МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1874ВЕ8Т

Руководство пользователя

2014

Содержание

Введение

В настоящем руководстве КФДЛ.431295.048 приведено описание архитектуры, функционального построения, системы команд и особенностей применения микросхемы 1874ВЕ8Т. Микросхема представляет собой СБИС однокристального 16-разрядного микроконтроллера с тактовой частотой до 40 МГц, ОЗУ 2 Кбайт, расширенным ОЗУ 2 Кбайт, внутренней памятью программ (типа EEPROM) объемом 16K×16 бит, блоком кодирования по ГОСТ 28147–89, восемью 16-разрядными АЦП с возможностью дифференциального включения входов и блоком цифровых компараторов, 14-разрядным ЦАП, 3-канальным ШИМ, портами ввода-вывода типа USART, контроллерами интерфейсов CAN, LIN, SPI, I2C, устройством высокоскоростного ввода-вывода HSIO, отладочным модулем OCDS и сторожевым таймером.

Изделие служит цели развития отечественной элементной базы для применения в различных цифровых системах управлениях, радиосвязи и изделиях, требующих точные аналогово-цифровые и цифро-аналоговые преобразования.

Настоящее руководство может служить практическим пособием по применению микроконтроллера для разработчиков систем на основе ИС 1874ВЕ8Т.

1 Назначение и область применения

Архитектура микроконтроллера ориентирована на создание цифровых управляющих систем, функционирующих в режиме реального времени с возможностью адаптации и модификации под конкретные приложения. Изделие может служить элементной базой для цифровых систем управления различной аппаратурой, силовой электроникой, автомобильной техникой и т. д. Наличие встроенных аппаратных сдвигателя, умножителя и делителя и возросшая производительность ядра позволят использовать микроконтроллер при решении задач обработки сигналов.

Наличие средств инструментальной отладки и встроенного отладочного модуля обеспечивает как эффективное проектирование систем на основе микроконтроллера, так и возможность смены алгоритма работы при создании модификаций систем.

Возможность гибкого управления энергопотреблением микроконтроллера (три режима пониженного энергопотребления, возможность отключения неиспользуемых блоков) позволит использовать микросхему в критичных к потреблению приложениях.

2 Краткое техническое описание ИС 1874ВЕ8Т

2.1 Функциональные параметры микросхемы

Функциональные параметры ИС 1874ВЕ8Т:	
разрядность данных, бит	16
частота следования импульсов тактового сигнала, МГц	до 40
динамически конфигурируемая шина данных, бит	8 или 16
встроенная память программ типа EEPROM, бит	16К×16
регистровое ОЗУ, бит	2024×8
адресуемая память, бит	64К×8
сервер PTS	1
параллельный 8-разрядный порт ввода-вывода	5
16-разрядный таймер/счетчик	2
блок высокоскоростного ввода-вывода	1
3-канальный ШИМ	1
генератор псевдослучайных последовательностей (ПСП)	1
блок кодирования по ГОСТ 28147–89	1
универсальный последовательный порт USART	2
контроллеры интерфейсов:	
CAN (сдвоенный)	1
LIN	1
SPI	1
I2C	1
АЦП/разрядность	8/16
максимальное количество используемых АЦП при дифференциальном включении входов	4
ЦАП/разрядность	1/14
режим энергопотребления	3
16-разрядный сторожевой таймер	1
модуль отладки OCDS	1

Микросхема выполнена в металлокерамическом планарном корпусе с четырехсторонним расположением выводов 4235.88-1 и предназначена для ручной и автоматической сборки в соответствии с ГОСТ РВ 20.39.412–97. Масса микросхемы – не более 4,5 г.

Герметизация микросхемы осуществляется шовной роликовой сваркой. Показатель герметичности микросхемы по эквивалентному нормализованному потоку – не более $6{,}65 \cdot 10^{-3}$ Па·см^3/с.

Микросхемы не имеют собственных резонансных частот ниже 100 Гц.

Структурная схема микросхемы 1874ВЕ8Т приведена на рисунке 2.1.

Условное графическое обозначение микросхемы приведено на рисунке 2.2.

Функциональное назначение выводов микросхем приведено в таблице 2.1.

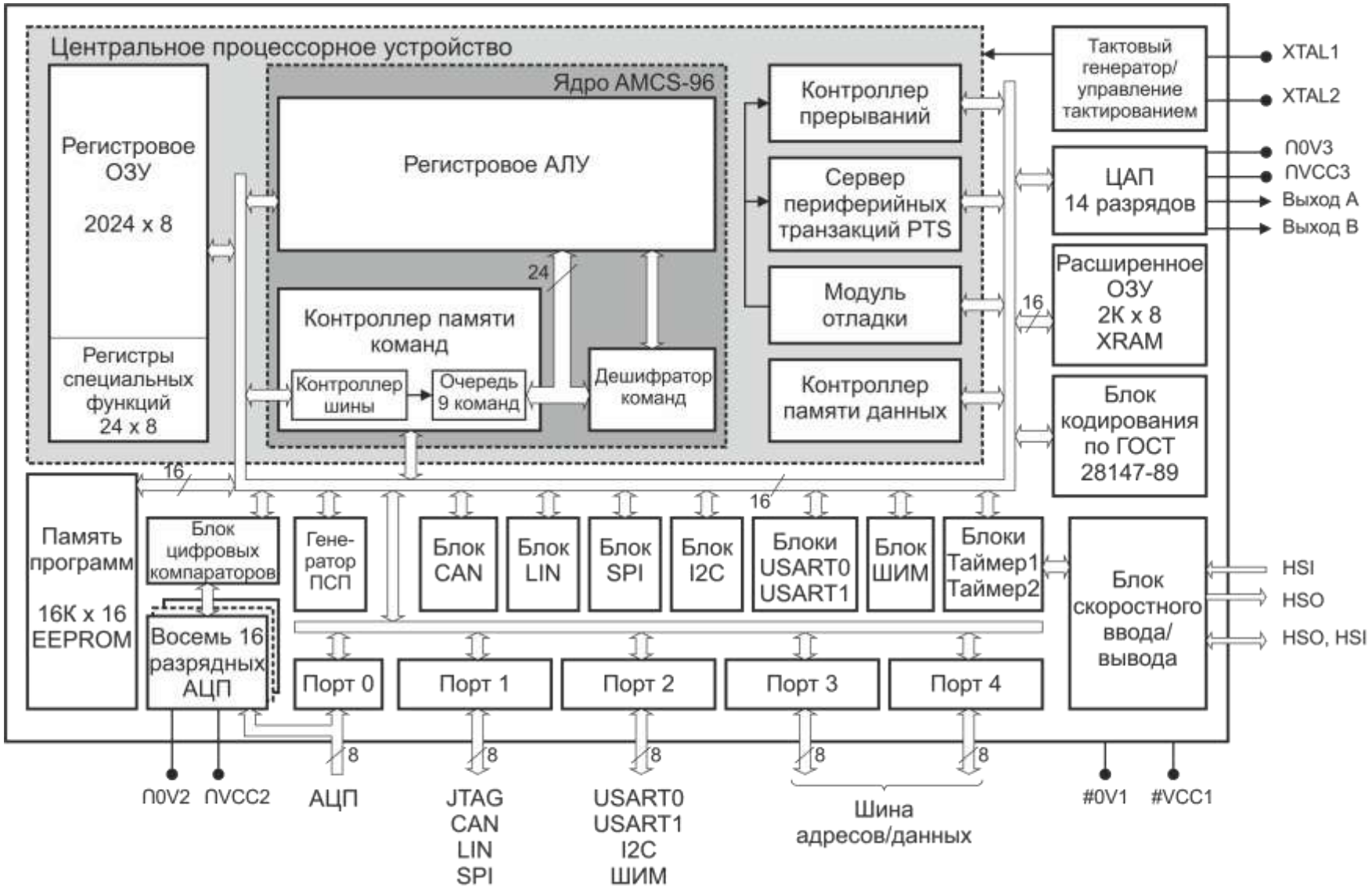


Рисунок 2.1 – Структурная схема микроконтроллера 1874ВЕ8Т

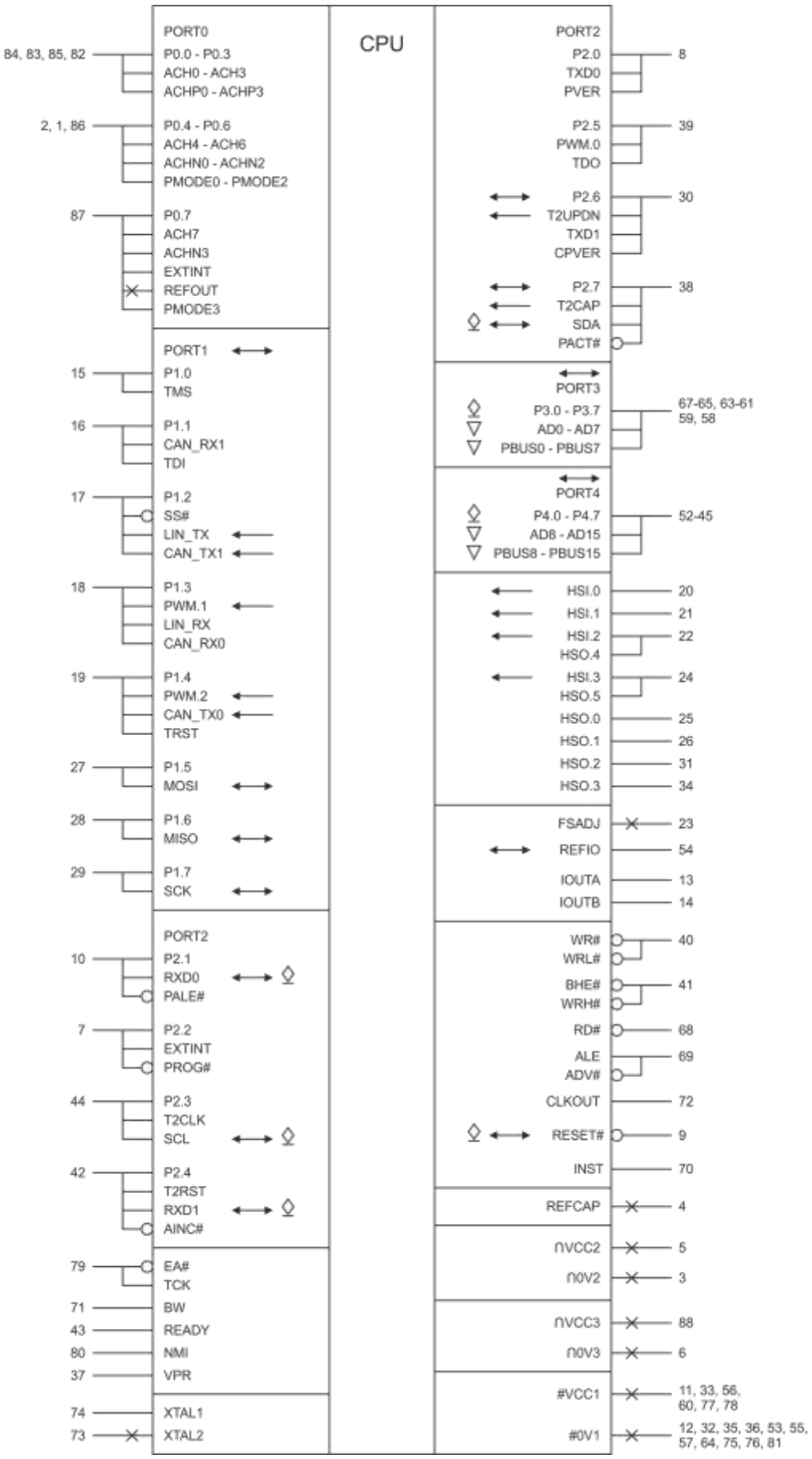


Рисунок 2.2 – Условное графическое изображение микросхемы 1874ВЕ8Т

Таблица 2.1 – Функциональное назначение выводов микроконтроллера

Обозначение вывода	Номер вывода	Функциональное назначение	Тип вывода
1	2	3	4
P0.0	84	Вход «порт 0, 0 разряд»	I
ACH0		Вход АЦП, канал 0	I
ACHP0		Прямой вход АЦП, канал 0	I
P0.1	83	Вход «порт 0, 1 разряд»	I
ACH1		Вход АЦП, канал 1	I
ACHP1		Прямой вход АЦП, канал 1	I
P0.2	85	Вход «порт 0, 2 разряд»	I
ACH2		Вход АЦП, канал 2	I
ACHP2		Прямой вход АЦП, канал 2	I
P0.3	82	Вход «порт 0, 3 разряд»	I
ACH3		Вход АЦП, канал 3	I
ACHP3		Прямой вход АЦП, канал 3	I
P0.4	2	Вход «порт 0, 4 разряд»	I
ACH4		Вход АЦП, канал 4	I
ACHN0		Инверсный вход АЦП, канал 0	I
PMODE0		Выбор режима программирования*	I
P0.5	1	Вход «порт 0, 5 разряд»	I
ACH5		Вход АЦП, канал 5	I
ACHN1		Инверсный вход АЦП, канал 1	I
PMODE1		Выбор режима программирования*	I
P0.6	86	Вход «порт 0, 6 разряд»	I
ACH6		Вход АЦП, канал 6	I
ACHN2		Инверсный вход АЦП, канал 2	I
PMODE2		Выбор режима программирования*	I
P0.7	87	Вход «порт 0, 7 разряд»	I
ACH7		Вход АЦП, канал 7	I
EXTINT		Вход «сигнал внешнего прерывания»	I
ACHN3		Инверсный вход АЦП, канал 3	I
REFOUT		Буферизированный вывод опорного напряжения АЦП	–
PMODE3		Выбор режима программирования*	I
P1.0	15	Вход/выход «порт 1, 0 разряд»	I/O
TMS		Вход выбора режима JTAG	I
P1.1	16	Вход/выход «порт 1, 1 разряд»	I/O
CAN_RX1		Вход 1 порта CAN	I
TDI		Вход данных JTAG	I
P1.2	17	Вход/выход «порт 1, 2 разряд»	I/O
SS#		Вход выбора ведомого порта SPI	I
LIN_TX		Выход порта LIN	O
CAN_TX1		Выход 1 порта CAN	O
P1.3	18	Вход/выход «порт 1, 3 разряд»	I/O
PWM.1		Выход ШИМ, 1 канал	O
LIN_RX		Вход порта LIN	I
CAN_RX0		Вход 0 порта CAN	I

Продолжение таблицы 2.1

1	2	3	4
P1.4	19	Вход/выход «порт 1, 4 разряд»	I/O
PWM.2		Выход ШИМ, 2 канал	O
CAN_TX0		Выход 0 порта CAN	O
TRST		Вход сброса JTAG	I
P1.5	27	Вход/выход «порт 1, 5 разряд»	I/O
MOSI		Вход данных ведомого/выход данных ведущего порта SPI	I/O
P1.6	28	Вход/выход «порт 1, 6 разряд»	I/O
MISO		Выход данных ведомого/вход данных ведущего порта SPI	I/O
P1.7	29	Вход/выход «порт 1, 7 разряд»	I/O
SCK		Вход/выход тактовой частоты порта SPI	I/O
P2.0	8	Выход «порт 2, 0 разряд»	O
TXD0		Выход последовательных данных USART0	O
PVER		Выход «верификация»*	O
P2.1	10	Вход «порт 2, 1 разряд»	I
RXD0		Вход/выход последовательных USART0	I/O/2
PALE#		Вход фиксации адреса/команды*	I
P2.2	7	Вход «порт 2, 2 разряд»	I
EXTINT		Вход «сигнал внешнего прерывания»	I
PROG#		Вход фиксации данных*	I
P2.3	44	Вход «порт 2, 3 разряд»	I
T2CLK		Вход «синхронизация таймера 2»	I
SCL		Вход/выход синхронизации порта I2C	I/O/2
P2.4	42	Вход «порт 2, 4 разряд»	I
T2RST		Вход «сброс таймера 2»	I
RXD1		Вход/выход последовательных данных USART1	I/O
AINC#		Вход «автоинкремент»*	I
P2.5	39	Выход «порт 2, 5 разряд»	O
PWM.0		Выход ШИМ, 0 канал	O
TDO		Выход данных JTAG	O
P2.6	30	Вход/выход «порт 2, 6 разряд»	I/O
T2UPDN		Вход «выбор режима таймера 2»	I
TXD1		Выход последовательных данных USART1	O
CPVER		Выход «ошибка программирования»*	O
P2.7	38	Вход/выход «порт 2, 7 разряд»	I/O
T2CAP		Вход «выборка данных таймера 2»	I
SDA		Вход/выход данных шины I2C	I/O/2
PACT#		Выход «подтверждение программирования»*	O
P3.0	67	Вход/выход «порт 3, 0 разряд»	I/O/2
AD0		Вход/выход «адрес-данные, 0 разряд»	I/O/Z
PBUS0		Вход/выход «адрес-команда-данные, 0 разряд»*	I/O/Z
P3.1	66	Вход/выход «порт 3, 1 разряд»	I/O/2
AD1		Вход/выход «адрес-данные, 1 разряд»	I/O/Z
PBUS1		Вход/выход «адрес-команда-данные, 1 разряд»*	I/O/Z
P3.2	65	Вход/выход «порт 3, 2 разряд»	I/O/2
AD2		Вход/выход «адрес-данные, 2 разряд»	I/O/Z
PBUS2		Вход/выход «адрес-команда-данные, 2 разряд»*	I/O/Z

Продолжение таблицы 2.1

1	2	3	4
P3.3	63	Вход/выход «порт 3, 3 разряд»	I/O/2
AD3		Вход/выход «адрес-данные, 3 разряд»	I/O/Z
PBUS3		Вход/выход «адрес-команда-данные, 3 разряд»*	I/O/Z
P3.4	62	Вход/выход «порт 3, 4 разряд»	I/O/2
AD4		Вход/выход «адрес-данные, 4 разряд»	I/O/Z
PBUS4		Вход/выход «адрес-команда-данные, 4 разряд»*	I/O/Z
P3.5	61	Вход/выход «порт 3, 5 разряд»	I/O/2
AD5		Вход/выход «адрес-данные, 5 разряд»	I/O/Z
PBUS5		Вход/выход «адрес-команда-данные, 5 разряд»*	I/O/Z
P3.6	59	Вход/выход «порт 3, 6 разряд»	I/O/2
AD6		Вход/выход «адрес-данные, 6 разряд»	I/O/Z
PBUS6		Вход/выход «адрес-команда-данные, 6 разряд»*	I/O/Z
P3.7	58	Вход/выход «порт 3, 7 разряд»	I/O/2
AD7		Вход/выход «адрес-данные, 7 разряд»	I/O/Z
PBUS7		Вход/выход «адрес-команда-данные, 7 разряд»*	I/O/Z
P4.0	52	Вход/выход «порт 4, 0 разряд»	I/O/2
AD8		Вход/выход «адрес-данные, 8 разряд»	I/O/Z
PBUS8		Вход/выход «адрес-команда-данные, 8 разряд»*	I/O/Z
P4.1	51	Вход/выход «порт 4, 1 разряд»	I/O/2
AD9		Вход/выход «адрес-данные, 9 разряд»	I/O/Z
PBUS9		Вход/выход «адрес-команда-данные, 9 разряд»*	I/O/Z
P4.2	50	Вход/выход «порт 4, 2 разряд»	I/O/2
AD10		Вход/выход «адрес-данные, 10 разряд»	I/O/Z
PBUS10		Вход/выход «адрес-команда-данные, 10 разряд»*	I/O/Z
P4.3	49	Вход/выход «порт 4, 3 разряд»	I/O/2
AD11		Вход/выход «адрес-данные, 11 разряд»	I/O/Z
PBUS11		Вход/выход «адрес-команда-данные, 11 разряд»*	I/O/Z
P4.4	48	Вход/выход «порт 4, 4 разряд»	I/O/2
AD12		Вход/выход «адрес-данные, 12 разряд»	I/O/Z
PBUS12		Вход/выход «адрес-команда-данные, 12 разряд»*	I/O/Z
P4.5	47	Вход/выход «порт 4, 5 разряд»	I/O/2
AD13		Вход/выход «адрес-данные, 13 разряд»	I/O/Z
PBUS13		Вход/выход «адрес-команда-данные, 13 разряд»*	I/O/Z
P4.6	46	Вход/выход «порт 4, 6 разряд»	I/O/2
AD14		Вход/выход «адрес-данные, 14 разряд»	I/O/Z
PBUS14		Вход/выход «адрес-команда-данные, 14 разряд»*	I/O/Z
P4.7	45	Вход/выход «порт 4, 7 разряд»	I/O/2
AD15		Вход/выход «адрес-данные, 15 разряд»	I/O/Z
PBUS15		Вход/выход «адрес-команда-данные, 15 разряд»*	I/O/Z
HSI.0	20	Вход «быстрый ввод, канал 0»	I
HSI.1	21	Вход «быстрый ввод, канал 1»	I
HSI.2	22	Вход «быстрый ввод, канал 2»	I
HSO.4		Выход «быстрый вывод, канал 4»	O
HSI.3	24	Вход «быстрый ввод, канал 3»	I
HSO.5		Выход «быстрый вывод, канал 5»	O
HSO.0	25	Выход «быстрый вывод, канал 0»	O
HSO.1	26	Выход «быстрый вывод, канал 1»	O
HSO.2	31	Выход «быстрый вывод, канал 2»	O

Продолжение таблицы 2.1

1	2	3	4
HSO.3	34	Выход «быстрый вывод, канал 3»	O
EA# TCK	79	Вход «внешний доступ» Вход тактовой частоты JTAG	I I
BW	71	Вход «разрядность внешней шины»	I
READY	43	Вход «готовность»	I
NMI	80	Вход «немаскируемое прерывание»	I
RESET#	9	Вход/выход «сброс»	I/O/2
INST	70	Выход «чтение команды»	O
ALE ADV#	69	Выход «разрешение записи адреса» Выход «адрес действителен»	O O
WR# WRL#	40	Выход «запись» Выход «запись младшего байта»	O O
BHE# WRH#	41	Выход «разрешение записи старшего байта» Выход «запись старшего байта»	O O
RD#	68	Выход «чтение»	O
FSADJ	23	Вывод подстройки выходного тока ЦАП	–
REFCAP	4	Вывод подключения внешнего источника опорного напряжения АЦП/Вывод подключения фильтрующего конденсатора источника опорного напряжения АЦП	–
REFIO	54	Вход/выход опорного напряжения ЦАП Вывод подключения фильтрующего конденсатора источника опорного напряжения ЦАП	I/O –
CLKOUT	72	Выход «системный тактовый сигнал»	O
IOUTA	13	Выход тока ЦАП	O
IOUTB	14	Комплементарный выход ЦАП	O
XTAL1	74	Вывод подключения кварцевого резонатора/ Вход тактового сигнала	– I
XTAL2	73	Вывод подключения кварцевого резонатора	–
VPR	37	Вход в режим программирования*/Вход «старт с альтернативного адреса» Вход «возврат из режима пониженного потребления»	I I
#VCC1	11, 33, 56, 60, 77, 78	Выводы питания 3,3 В цифровой части микросхемы	–
∩VCC2	5	Вывод питания 3,3 В аналого-цифрового преобразователя	–
∩VCC3	88	Вывод питания 3,3 В цифро-аналогового преобразователя	–
#0V1	12, 32, 35, 36, 53, 55, 57, 64, 75, 76, 81	Общие выводы цифровой части микросхемы	–

Окончание таблицы 2.1

1	2	3	4
∩0V2	3	Общий вывод аналого-цифрового преобразователя	–
∩0V3	6	Общий вывод цифро-аналогового преобразователя	–

Примечания

1 В графе «Тип вывода» используются обозначения: I – вход, O – выход, Z – третье состояние, 2 – режим открытого стока.

2 Знаком «*» отмечены функции, которые реализуются только при использовании записанной в ПЗУ стандартной программы-монитора, разработанной ОАО «НИИЭТ».

2.2 Электрические параметры микросхемы

Электрические параметры микросхемы при приёмке и поставке и предельно допустимые значения параметров приведены в таблицах 2.2 и 2.3 соответственно.

Номинальное значение напряжения питания микросхемы 1874BE8T – 3,3 В. Допустимое отклонение напряжения питания ± 10 %. Амплитуда пульсаций напряжения питания – не более 30 мВ. Напряжение источника опорного напряжения – от 2,6 до 3,6 В. Допустимое отклонение напряжения питания от крайних значений – минус 1 % для напряжения 2,6 В и 1 % для напряжения 3,6 В.

Микросхема является стойкой к климатическим воздействиям и сохраняет свои параметры в процессе и после воздействия на них климатических факторов, приведенных в таблице 4 ОСТ В 11 0998–99, в том числе:

- пониженной рабочей температуры среды – минус 60 ºС;
- повышенной рабочей температуры среды – 125 ºС;
- повышенной предельной температуры – 150 ºС.

Разработанная микросхема 1874BE8T устойчива к воздействию механических факторов по ОСТ В 11 0998–99 и технологическим воздействиям при изготовлении радиоэлектронной аппаратуры по ОСТ В 11 0998–99.

Таблица 2.2 – Электрические параметры при приемке и поставке в диапазоне рабочих температур окружающей среды от минус 60 до 125 ºС

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 RD#, WR#, ALE, BHE#, INST, CLKOUT, RESET#, P1.0 – P1.7, P2.0/TXD0, P2.1/RXD0 (в режиме 0), P2.3/SCL, P2.4/RXD1 (в режиме 0), P2.5/PWM.0, P2.6/TXD1, P2.7/SDA, P3.0 – P3.7, P4.0 – P4.7, HSO.0 – HSO.3, HSI.2/HSO.4, HSI.3/HSO.5, В, I_{OL} = 6 мА, U_{CC1} = 3,0 В	U_{OL}	–	0,4	–60 ± 3 25 ± 10 125 ± 5
2 Выходное напряжение высокого уровня по выводам P1.0 – P1.7, P2.6/TXD1, P2.7/SDA, В, I_{OH1} = –100 мкА, U_{CC1} = 3,0 В	U_{OH1}	U_{CC1} – 0,9	–	
3 Выходное напряжение высокого уровня по выводам RD#, WR#, ALE, BHE#, INST, CLKOUT, P2.0/TXD0 (в режиме 0), P2.4/RXD1, P2.5/PWM.0, P3.0 – P3.7, P4.0 – P4.7, HSO.0 – HSO.3, HSI.2/HSO.4, HSI.3/HSO.5, В, I_{OH2} = –6 мА, U_{CC1} = 3,0 В	U_{OH2}	U_{CC1} – 0,9	–	

Продолжение таблицы 2.2

1	2	3	4	5
4 Выходное опорное напряжение АЦП на выводе REFOUT, В, $U_{CC2} = U_{CC1} = 3{,}3$ В, $C_{REFCAP} = 0{,}1$ мкФ	U_{REFOUT}	1,125	1,375	
5 Выходное опорное напряжение ЦАП, В, $U_{CC3} = U_{CC1} = 3{,}3$ В, $I_{OUTFS} = 20$ мА	U_{REFIO}	1,14	1,26	
6 Ток утечки низкого уровня по входам P2.2 – P2.4, P3.0 – P3.7, P4.0 – P4.7, HSI.0 – HSI.3, EA#, BW, READY, VPR, NMI, мкА, $U_{IL} = 0$ В, $U_{CC1} = U_{CC2} = 3{,}6$ В	I_{ILL1}	–10	–	
7 Ток утечки высокого уровня по входам P2.2 – P2.4, P3.0 – P3.7, P4.0 – P4.7, HSI.0 – HSI.3, EA#, BW, READY, VPR, RESET#, мкА, $U_{IH} = U_{CC1}$, $U_{CC1} = U_{CC2} = 3{,}6$ В	I_{ILH1}	–	10	
8 Ток утечки низкого уровня по входам P0.0 – P0.7, мкА, $U_{IL} = 0$ В, $U_{CC1} = U_{CC2} = 3{,}6$ В	I_{ILL2}	–0,5	–	
9 Ток утечки высокого уровня по входам P0.0 – P0.7, мкА, $U_{IH} = U_{CC1}$, $U_{CC1} = U_{CC2} = 3{,}6$ В	I_{ILH2}	–	0,5	
10 Входной ток низкого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА, $U_{IL} = 0{,}45$ В, $U_{CC1} = 3{,}6$ В	I_{IL1}	–300	–	–60 ± 3
11 Входной ток высокого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА, $U_{IH} = 2{,}0$ В, $U_{CC1} = 3{,}6$ В	I_{IH1}	–1 000	–	25 ± 10 125 ± 5
12 Входной ток низкого уровня по выводу RESET#, мкА, $U_{IL} = 0$ В, $U_{CC1} = 3{,}6$ В	I_{IL2}	–800	–	
13 Входной ток высокого уровня по выводу NMI, мкА, $U_{IH} = 2{,}4$ В, $U_{CC1} = 3{,}6$ В	I_{IH2}	50	–	
14 Динамический ток потребления цифровой части в режиме RESET по выводам #VCC1, мА, $U_{CC1} = 3{,}6$ В, $f_{CI} = 33$ МГц	I_{OCC1}	–	50	
15 Ток потребления цифровой части в режиме холостого хода по выводам #VCC1, мА, $U_{CC1} = 3{,}6$ В, $f_{CI} = 33$ МГц	I_{CCI1}	–	50	
16 Ток потребления цифровой части в режиме хранения по выводам #VCC1, мА, $U_{CC1} = 3{,}6$ В, $f_{CI} = 33$ МГц	I_{CCS1}	–	3	
17 Динамический ток потребления цифровой части в режиме Slow, мА, $U_{CC1} = 3{,}6$ В, $f_{CI} = 33$ МГц	$I_{OCCSLOW}$	–	30	
18 Динамический ток потребления АЦП по выводу ∩VCC2, мА, $U_{CC2} = U_{CC1} = 3{,}6$ В	I_{OCC2}	–	35	

Окончание таблицы 2.2

1	2	3	4	5
19 Динамический ток потребления ЦАП по выводу ∩VCC3, мА, U_{CC3} = 3,6 В, U_{CC1} = 3,6 В, f_{CI} = 33 МГц, I_{OUTFS} = 20 мА	I_{OCC3}	–	30	–60 ± 3 25 ± 10 125 ± 5
20 Интегральная нелинейность ЦАП, LSB, U_{CC3} = U_{CC1} = 3,3 В, I_{OUTFS} = 20 мА	E_L	–7	7	125 ± 5
		–5	5	–60 ± 3 25 ± 10 85 ± 3
21 Дифференциальная нелинейность ЦАП, LSB, U_{CC3} = U_{CC1} = 3,3 В, I_{OUTFS} = 20 мА	E_{LD}	–4	4	
		–6	6	125 ± 5
22 Погрешность коэффициента усиления ЦАП, % от полной шкалы, U_{CC3} = U_{CC1} = 3,3 В, I_{OUTFS} = 20 мА	GE	–0,5	0,5	
23 Погрешность смещения ЦАП, % от полной шкалы, U_{CC1} = U_{CC3} = 3,3 В, I_{OUTFS} = 20 мА	OE	–0,02	0,02	
24 Общие гармонические искажения ЦАП, дБ, U_{CC1} = U_{CC3} = 3,3 В, I_{OUTFS} = 20 мА, f_{OUT} = 125 кГц	THD_1	–	–70	
25 Выходной шум ЦАП, pA $\sqrt{Hz}$, U_{CC1} = U_{CC3} = 3,3 В, I_{OUTFS} = 20 мА, f_{OUT} = 125 кГц	ON	–	20	–60 ± 3 25 ± 10 125 ± 5
26 Общие гармонические искажения АЦП, дБ, U_{CC2} = U_{CC1} = 3,3 В, PGA = 0 дБ, f_{IN} = (0 – 4) кГц, f_S = (8; 64) кГц	THD_2	–	–76	
27 Отношение сигнал/(шум + искажения) в каналах АЦП, дБ, U_{CC2} = U_{CC1} = 3,3 В, PGA = 0 дБ, f_{IN} = (0 – 4) кГц, f_S = (8; 64) кГц	SINAD	73	–	
28 Функциональный контроль U_{CC1} = (3,0; 3,6) В, f_{CI} = 40 МГц	ФК	–	–	

Примечания

1 Параметры I_{ILL1}, I_{ILH1}, I_{ILL2}, I_{ILH2} при температуре минус 60 ºС не измеряются, а гарантируются нормой при температуре (25 ± 10) ºС.

2 Параметры E_L и E_{LD} ЦАП при повышенной температуре 85 ºС не измеряются, а гарантируются измерениями при температуре 125 ºС и подтверждаются периодическими испытаниями.

3 Параметры THD_2, SINAD измеряются в режиме дифференциальных входов.

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

Наименование параметра, единица измерения, режим 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_{CC1}	3,0	3,6	–0,3	5,0
2 Напряжение питания АЦП, В*	U_{CC2}	3,0	3,6	–0,3	5,0
3 Напряжение питания ЦАП, В*	U_{CC3}	3,0	3,6	–0,3	5,0
4 Входное напряжение низкого уровня, В*	U_{IL}	–0,5	0,8	–0,6	–
5 Входное напряжение высокого уровня тактового сигнала XTAL1, В*	U_{IHCI}	$0,7U_{CC1}$	U_{CC1}	–	U_{CC1} + 0,6
6 Входное напряжение высокого уровня сигнала RESET#, В*	U_{IHRST}	$0,6U_{CC1}$	U_{CC1}	–	U_{CC1} + 0,6
7 Входное напряжение высокого уровня, В*	U_{IH}	$0,2U_{CC1}$+1	U_{CC1}	–	U_{CC1} + 0,6
8 Напряжение между выводами #0V1 и ∩0V2, В	U_{0V1}–U_{0V2}	–0,01	0,01	–0,3	0,3
9 Диапазон преобразования каналов АЦП (на входах ACH.0 – ACH.7), В PGA = 0 дБ, U_{REF} = 1,25 В, U_{CC2} = 3,3 В	U_{FSR}	1,44	**	–	–
10 Выходной ток высокого уровня по выводам P1.0 – P1.7, P2.6, P2.7, мкА*	I_{OH1}	–100	–	–200	–
11 Выходной ток высокого уровня по выводам P3.0 – P3.7, P4.0 – P4.7, TXD0/P2.0, RXD0/P2.1, мА*	I_{OH2}	–6	–	–12	–
12 Выходной ток низкого уровня, мА*	I_{OL}	–	6	–	12
13 Выходной ток полной шкалы ЦАП, мА, U_{CC3} = 3,3 В	I_{OUTFS}	2	20	–	–
14 Частота следования импульсов тактового сигнала, МГц	f_{CI}	0,001	40	–	–
15 Длительность фронта и спада входных сигналов, нс	t_{LH}, t_{HL}	–	5	–	–
16 Емкость выходной нагрузки, пФ	C_L	–	100	–	200

Примечание – Не допускается одновременная подача двух и более предельных режимов.

_____

* Время работы в предельном режиме должно быть не более 5 с.

** Верхнее практическое значение диапазона преобразования U_{FSR} определяется потребителем, исходя из условий применения, с учетом зависимости, приведенной на рисунке 19.4а .

3 Архитектура изделия

16-разрядный микроконтроллер 1874ВЕ8Т предназначен для выполнения вычислительных инструкций с повышенным быстродействием и высокоточных аналогово-цифровых и цифро-аналоговых преобразований. Он имеет общие архитектуру и набор инструкций с другими микросхемами серии 1874. Базируется на ядре AMCS-96 разработки НИИЭТ, которое отличается увеличенной производительностью (в среднем на 30 – 40 % относительно других контроллеров серии 1874, реализующих архитектуру MCS-96, при одинаковой частоте) благодаря введению конвейеризации процесса выборки – выполнения команд, полной оптимизации процесса выполнения инструкций, наличию аппаратных сдвигателя, умножителя, делителя и других конструктивных особенностей. Реализуемая ядром микроконтроллера архитектура AMCS-96 полностью программно совместима с ядрами MCS-96 архитектуры.

ОЗУ микроконтроллера доступно быстрым способам адресации; отключаемое расширенное ОЗУ доступно при косвенном способе адрессации. ПЗУ типа EEPROM имеет защиту и удобный механизм программирования. Поддерживается возможность начала выполнения программы из разных областей внутренней и внешней памяти. Для отладки программ используется встроенная система отладки с возможностью перезагрузки микросхемы. Внешняя шина адреса/данных конфигурируемая.

Микроконтроллер имеет в своем составе сервер периферийных транзакций PTS, позволяющий передавать данные в моменты простоя шины данных. Это обеспечивает транзакции без остановки ЦПУ, что повышает общее быстродействие системы.

Наличие контроллеров последовательных интерфейсов CAN, LIN, SPI и I2C позволяет использовать микроконтроллер в различных системах контроля и сбора информации, а наличие подсистемы высокоскоростного ввода-вывода с разрешающей способностью в один машинный цикл, позволяет использовать его в быстродействующих системах управления.

Для выполнения высокоточных аналого-цифровых и цифро-аналоговых преобразований используются восемь 16-разрядных АЦП с возможностью параллельного преобразования каналов и дифференциального включения входов и 14-разрядный высокоскоростной ЦАП. Для более эффективного использования ресурсов имеется блок цифровых компараторов, позволяющий отслеживать выход параметров процессов за границы допустимых значений.

Микроконтроллер имеет гибкую систему управления энергопотреблением. Помимо режимов пониженного энергопотребления (POWERDOWN), холостого хода (IDLE) и медленной работы (SLOW), существует возможность отключения неиспользуемых периферийных устройств.

3.1 Особенности архитектуры

Структурная схема микросхемы 1874ВЕ8Т приведена на рисунке 3.1.

Микроконтроллер реализует архитектуру AMCS-96, полностью программно совместимую с MCS-96 фирмы Intel с изменениями, направленными на увеличение производительности.

Это 16-разрядная быстродействующая микросхема высокой степени интеграции, ориентированная на решение задач управления процессами в реальном масштабе времени. В результате применения микросхемы ожидается увеличение функциональности, надежности и производительности встроенных систем управления за счёт использования полностью нового быстродействующего 16-разрядного ядра, высокоточных 16-разрядных аналого-цифровых и 14-разрядного цифро-аналогового преобразователей, большого объема внутрикристального ОЗУ, энергонезависимой памяти программ типа EEPROM, устройства высокоскоростного ввода-вывода и последовательных портов приема/передачи. Гибкое

управление энергопотреблением позволит использовать микроконтроллер 1874ВЕ8Т в критичных к потреблению приложениях.

Архитектура микроконтроллера с увеличенной производительностью, развитой системой встроенных функциональных блоков и мощной системой команд ориентирована на создание систем, реализующих функций управления и вычисления в режиме реального времени (в том числе систем управления различной аппаратурой).

Ключевые особенности микроконтроллера:

- быстродействующая архитектура типа «регистр-регистр»;
- аппаратный умножитель 16 на 16 за один машинный цикл;
- аппаратный делитель 32 на 16 за два машинных цикла;
- динамически конфигурируемая разрядность шины данных – 8 или 16 бит.
- устройство высокоскоростного ввода-вывода с разрешающей способностью один машинный цикл;
- два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета.

Микроконтроллер содержит полный набор команд, включающий операции с битами, байтами, словами (16 бит), двойными беззнаковыми словами (32 бита), длинные знаковые операции (32 бита), работу с флагами, а также переходы и вызовы подпрограмм. Все стандартные логические и арифметические команды работают как с байтами, так и со словами. Команды перехода по установке и очистке бита могут работать как с регистрами специальных функций (SFR), так и с другими байтами регистрового файла. Эти быстрые битовые операции позволяют ускорить функции ввода-вывода.

Операции над байтами и словами составляют основу системы команд. Ассемблер ASM-96 использует суффикс «B» в мнемонике для операций над байтами (отсутсвие суффикса указывает на операции над словами).

Длинные и двухсловные операции включают операции сдвигов, нормализацию, умножения и деления. Деление 32-битного числа на 16-битное порождает 16-битное частное и 16-битный остаток. Умножение 16-битного числа на 16-битное образует 32-битный результат. Обе операции могут выполняться с числами со знаком или без знака. Команды нормализации и соответствующий флаг обеспечивают аппаратную поддержку пакета программ для выполнения операций над числами с плавающей запятой (FPAL-96).

3.2 Краткое описание функционирования микросхемы

Процессорное ядро

Микроконтроллеры построены по архитектуре Фон Неймана. Они имеют внешнюю системную магистраль для обмена данными с внешней памятью и дополнительными периферийными устройствами. Система команд микроконтроллеров поддерживает широкий набор методов адресации, в т. ч. битовую адресацию.

Архитектура микроконтроллера, называемая архитектурой типа «регистр-регистр», принципиально отличается от архитектуры микроконтроллеров других серий. Такая архитектура обеспечивает достижение более высокой производительности и упрощает работу с периферией. Этим, в первую очередь, объясняются возможности эффективного решения на базе микроконтроллера достаточно сложных задач управления в реальном времени.

Главным отличительным признаком архитектуры микроконтроллера является полностью обновленное ядро, базирующееся на регистровом файле. Большое число универсальных легкодоступных регистров исключает «узкие места», свойственные архитектуре, использующей специальные регистры-аккумуляторы, и обеспечивает быстрое переключение контекста. Все устройства микроконтроллера реализуют операции

с байтами, словами, несколько операций с 32-битовыми операндами, а также команды перехода по битам.

Структурная схема ядра AMCS-96 разработки ОАО «НИИЭТ» представлена на рисунке 3.1.

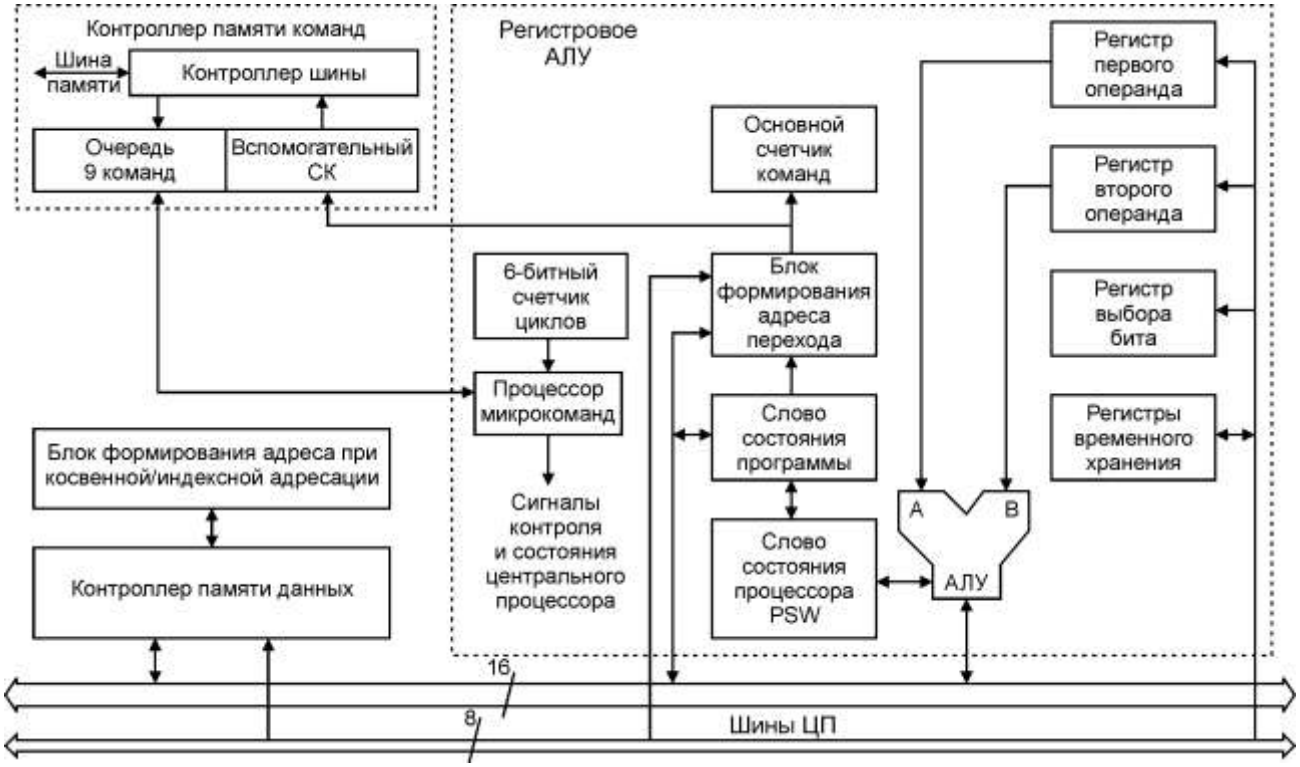


Рисунок 3.1 – Структурная схема ядра микроконтроллера

Команды поступают в блок очереди команд, который может хранить до девяти команд длиной до 7 байт каждая. Процесс выборки и выполнения команд независимый, что позволяет минимизировать время простоя как контроллера команд, так и ЦПУ. Выборка команд может осуществляться или из внутреннего ПЗУ, или через интерфейс из внешней памяти в режиме 16/8 бит.

По сигналу команды попадают в процессор микрокоманд, где происходит их декодирование и выполнение. Выполнение команды состоит из последовательности выполнений микроинструкций. Нужная инструкция выбирается счетчиком циклов, обнуляющимся при старте выполнения каждой новой команды.

Контроллер памяти данных выполняет выборку нужных данных из ОЗУ, расширенного ОЗУ, области периферийных устройств, внутреннего ПЗУ или внешней памяти.

Блок формирования адреса при косвенной/индексной адресации позволяет формировать адрес без использования функций АЛУ. В своем составе содержит сумматор/вычитатель.

Основной счетчик команд содержит адрес команды, следующей за выполняемой. После сброса загружается начальным адресом, с которого начинается выполнение программы. По выбору значения начального адреса – 2080h, 9000h, 0A000h.

Если переход, прерывание, вызов подпрограммы или возврат из подпрограммы изменяет адресную последовательность, то АЛУ загружает соответствующий адрес в PC и очищает очередь.

ЦПУ выполняет вычисления в АЛУ. Слова вводятся в АЛУ через входы A и B. АЛУ выполняет все логические и арифметические операции за один (для деления – два) машинных цикла. В своем составе содержит встроенные аппаратные сумматоры/вычитатели, сдвигатели, умножитель, делитель и др.

Регистр слова состояния процессора PSW содержит бит I, который производит общее разрешение обслуживания всех маскируемых прерываний, бит PSE, разрешающий или запрещающий работу сервера периферийного обмена, и шесть битовых флагов, отражающих состояние программы пользователя.

Способы адресации

Система команд микроконтроллера содержит следующие типы адресации: прямая регистровая, косвенная, косвенная с автоинкрементом, непосредственная, короткая индексная и длинная индексная. Эти типы адресации увеличивают гибкость и скорость выполнения команд устройствами микроконтроллера. Каждая команда использует, по крайней мере, один из способов адресации.

Прямая регистровая адресация и непосредственная адресация выполняются наиболее быстро. Прямая регистровая адресация обеспечивает доступ к файлу регистров и SFR. Непосредственная адресация использует информацию, следующую за кодом команды, как операнд.

Оба режима косвенной адресации используют значение слова в регистре, как адрес операнда. Косвенная адресация с автоинкрементом увеличивает адресное слово на единицу после операции с байтом и на два – после операции со словом. Этот способ адресации обеспечивает легкий доступ к справочным таблицам.

Длинная индексная адресация обеспечивает прямой доступ к любой ячейке 64К адресного пространства. Этот способ формирует адрес операнда добавлением 16-битного значения к регистровому слову. Индексирование с нулевым регистром позволяет иметь прямую адресацию к любой ячейке. Короткая индексная адресация формирует адрес операнда добавлением 8-битного значения к регистровому слову.

Кроме того, имеются расширенные версии команд прерываемой и непрерываемой передачи блока. Множество способов адресации делает легким программирование на языке ассемблера и обеспечивает отличную взаимосвязь с языками высокого уровня. Команды ассемблера состоят из мнемоники, за которой следуют адрес или данные.

Прерывания

Основной функцией микроконтроллера является обеспечение управления устройствами в реальном времени. Схема контроллера прерываний позволяет событиям реального времени управлять выполнением программы. Когда событие вызывает прерывание, ядро обслуживает это прерывание перед выполнением следующей команды. Встроенная периферия, внешний сигнал или команда могут выставить запрос на прерывание. В простейшем случае микроконтроллер получает запрос прерывания, осуществляет обслуживание и возвращается к прерванной программе.

В разработанном микроконтроллере 17 векторов прерывания, а также два дополнительных вектора прерывания – пошаговое выполнение программы и неопознанный код операции, используемых в системах отладки или платах-прототипах. Когда контроллер прерывания определяет одно из семнадцати прерываний, он устанавливает соответствующий бит в один из двух регистров ожидания прерываний. Отдельные прерывания разрешаются либо запрещаются установкой или очисткой бит в регистре маски прерываний. Когда контроллер прерывания производит обработку прерывания, он делает вызов программы обслуживания прерываний, а затем очищает соответствующий бит ожидания.

Имеются две возможности обслуживания прерываний: программное обслуживание прерываний через контроллер прерываний и микропрограммное обслуживание прерываний через блок PTS. Можно выбрать вариант обслуживания прерываний для каждого из маскируемых прерываний. Немаскируемые прерывания (NMI, программное

прерывание, прерывание по неподдерживаемому коду, прерывание отладочного модуля) всегда обрабатываются подпрограммами обработки прерываний.

Сервер периферийных транзакций PTS позволяет осуществлять аппаратную обработку прерываний. Он содержит набор встроенных алгоритмов, исходные данные для которых должны быть размещены программой пользователя в ПЗУ или во внешней памяти. Алгоритмы PTS охватывают, в основном, пересылки данных. Прерывания, обслуживаемые PTS, отрабатываются параллельно работе ЦПУ в моменты простоя шины данных, тем самым не мешая выполнению основной программы. Пересылка может осуществляться и в режиме IDLE.

Блок PTS может выполнять операции:

- передача блока информации из одного места памяти в другое;
- последовательный опрос нескольких каналов АЦП;
- загрузка данных в блок HSO;
- выгрузка данных из блока HSI.

3.3 Некоторые особенности периферийных устройств

Стандартные порты ввода-вывода

Микроконтроллеры имеют пять 8-разрядных портов ввода-вывода. Порт 0 – входной, он же порт аналоговых входов для АЦП. Порт 1 – квазидвунаправленный, он разделяет выводы с двумя выходами модуля ШИМ. Порт 2 содержит три типа линий – квазидвунаправленные, входные и выходные. Порты 3 и 4 являются двунаправленными портами с открытым стоком, они могут использоваться, как интерфейс шины адресов/данных.

Таймеры-счетчики

Микроконтроллер имеет два независимых 16-разрядных многофункциональных таймера-счетчика. В одном из таймеров используется внутренняя синхронизация (режим работы – таймер реального времени), в другом – внешняя (счетчик внешних событий). Содержимое таймеров может быть в любое время считано, программно изменено или сброшено внешним сигналом. Таймер-счетчик внешних событий подсчитывает положительные или отрицательные фронты входных сигналов и может работать как в режиме прямого (инкремента), так и обратного (декремента) счета.

Источником счетных импульсов может быть либо системный синхросигнал с фиксированной частотой (возможно предварительно деленной в заданное число раз), либо один из входов микроконтроллера. В первом случае устройство выполняет функции таймера, так как фактически считает интервалы времени постоянной длительности.

Во втором случае устройство выполняет функции счетчика событий (отрицательных или положительных перепадов) на входе микросхемы. Счет может производиться либо в одном, либо в обоих направлениях. В последнем случае направление счета определяется уровнем сигнала на соответствующем входе микросхемы. При переходе содержимого счетчика из наибольшего состояния в наименьшее, и наоборот, могут генерироваться соответствующие внутренние запросы на прерывания.

Внутренняя синхронизация увеличивает значение таймера 1 и таймера 2 каждый восьмой такт процессора, равный трем периодам тактовой частоты. Таймер 2 может иметь внешнюю синхронизацию. При внешней синхронизации значение таймера 2 увеличивается при каждом положительном или отрицательном перепаде. Любой внешний или внутренний источник может сбросить таймер 2. Таймер 1 и таймер 2 могут формировать прерывания при переполнении. Микроконтроллер также позволяет использовать таймеры для задания скорости передачи информации последовательным портом и для управления работой сторожевого таймера.

Сторожевой таймер

Сторожевой таймер позволяет восстанавливать нормальную работу при сбоях программ. Если таймер разрешен, он будет вызывать аппаратный сброс, если программа не очищает его каждые 64К машинных циклов.

Когда таймер переполняется, он переводит линию RESET в состояние низкого уровня не менее, чем на два машинных цикла, сбрасывая микроконтроллер и другие устройства, подключенные к его выводу RESET#.

Устройства высокоскоростного ввода-вывода HSI/HSO

HSI может записывать время внешних событий с разрешающей способностью в один машинный цикл, следить за четырьмя независимыми линиями и захватывать значение таймера 1, когда происходит запрограммированиое событие. Инициализировать захват могут четыре типа событий: нарастание входного сигнала (передний фронт), спад сигнала (задний фронт), нарастание и спад или каждый восьмой передний фронт. HSI может хранить восемь значений таймера 1, семь в семиуровневой области FIFO и один – в регистре хранения HSI. Чтение регистра хранения HSI не перегружает значения, размещенные в FIFO ранее. Устройства HSI могут сформировать прерывание при загрузке регистра хранения HSI или когда загружаемое значение является четвертым или шестым элементом FIFO.

HSO может инициализировать события в определенные моменты, базирующиеся на значениях таймера 1 или таймера 2. Эти программируемые события заключаются в запуске аналого-цифрового преобразования, сбросе таймера 2, формировании до четырех программных временных задержек и установке или очистке одной или нескольких линий из шести выходных линий HSO. HSO хранит ожидаемые события и соответствующее время в ассоциативном запоминающем устройстве (АЗУ). Этот блок хранит до восьми команд. Каждая команда определяет время действия, тип действия, имеет ли место прерывание, идет ссылка на таймер 1 или таймер 2. Каждый такт HSO сравнивает значение ячеек АЗУ на совпадение с текущим временем и инициализирует определенные события при совпадении времени.

Последовательные порты USART

Последовательные порты имеют один синхронный 0-й режим и три (1, 2 и 3) асинхронных. Асинхронные режимы полностью дуплексные, т. е. данные могут передаваться и приниматься одновременно. Приемник буферизован, что означает – прием второго байта может начаться до считывания первого. Передатчик также дважды буферизован. Наиболее общее использование синхронного нулевого режима – расширение возможностей устройств ввода-вывода микроконтроллеров, использующих регистры сдвигов. Режим 1 – стандартный 10-битный асинхронный режим, используемый для нормальных последовательных коммуникаций. Режимы 2 и 3 – 9-битные режимы общего пользования для межпроцессорных коммуникаций.

Контроллер интерфейса SPI

Последовательный периферийный интерфейс предназначен для быстрого синхронного обмена информацией между микроконтроллером и периферией или между двумя микроконтроллерами. Кроме того, пользователи могут использовать контроллер SPI для внутрисхемного программирования памяти программ или отладки программы при использовании отладочного модуля.

Контроллер интерфейса SPI может работать как мастер, так и как ведомое устройство. Регистр приема имеет дополнительный буфер, что позволяет принимать очередной байт данных, пока не прочитан ранее принятый байт. Дополнительный буфер регистра передачи позволяет организовать непрерывную передачу неограниченного числа байт.

Максимальная частота передачи данных равна четверти значения частоты генератора (так, при внешней частоте микроконтроллера 40 МГц – частота передачи

данных составит 10 МГц). Завершение передачи сопровождается установкой в «1» соответствующего флага.

Блок ШИМ

Встроенный ШИМ предназначен для генерации ШИМ-сигнала на выходах микросхемы без участия процессора. Скважность импульса на выходах блока является переменной величиной, импульсы повторяются каждые 256 или 512 тактов входного синхросигнала. Блок ШИМ имеет три выхода. Модулируемые сигналы могут использоваться в системах управления двигателями.

Аналого-цифровые преобразователи

Восемь АЦП конвертируют аналоговые напряжения на входе в цифровой эквивалент. Разрешающая способность – 16 бит с программируемым временем преобразования. Любой преобразователь может начать преобразование немедленно или преобразование всех каналов может быть запущено в запрограммированное время блоком HSO. Завершение каждого преобразования АЦП сопровождается прерыванием. АЦП имеет отдельные выводы питания ∩VCC2 и ∩0V2, что исключает влияние помех по цифровым линиям питания на аналого-цифровое преобразование. Имеется возможность выдачи опорного напряжения на вывод P0.7. Для уменьшения действия синфазных помех возможно дифференциальное включение входов, при этом максимальное количество одновременно работающих каналов равно четырем. Для каждого из каналов существует возможность организации режима цифрового компаратора, при котором выдача прерываний происходит при выходе результата преобразования за заданные границы.

Цифро-аналоговый преобразователь

Разрешающая способность – 14 бит. После записи кода в управляющий регистр ЦАП на выходах микроконтроллера появляется ток, соответствующий этому коду. Возможен перевод ЦАП в режим пониженного энергопотребления. Микроконтроллер имеет отдельный вывод FSADJ для подстройки тока преобразователя.

Модуль отладки (OCDS)

Этот функциональный блок предназначен для упрощения процесса отладки программного обеспечения. По выбранному типу события (требуемые данные на шине данных, требуемый адрес операнда на шине адреса, значение счетчика команд, выход в область внешнего ПЗУ) этот блок инициализирует заданное действие:

- немаскируемое прерывание;
- переход в режим IDLE (выход из режима по прерыванию или сбросу);
- аппаратный сброс.

Аппаратный сброс вместе с настройкой контроля выхода в область внешнего ПЗУ может использоваться для контроля выполнения программ и защиты внешней памяти от несанкционированного доступа.

Энергосберегающие режимы работы

Режим IDLE. Работают только встроенные функциональные устройства. Микроконтроллер находится в режиме ожидания разрешенного прерывания от внешнего или внутреннего устройства.

Режим POWERDOWN. Прекращение внутренней синхронизации и отключение генератора. Внутреннее ОЗУ и большинство периферийной памяти сохраняет свое значение, если сохраняется напряжение питания. Ток потребления составляет несколько микроампер.

Режим SLOW. Замедление работы ЦПУ и периферийных устройств за счет увеличения длительности машинного цикла в два раза. Режим может включаться/выключаться программно без сброса микроконтроллера.

Также существует возможность независимого отключения/включения тактового сигнала периферийных устройств, что позволяет гибко управлять общим энергопотреблением.

4 Типы данных и адресация

В данном разделе рассматриваются типы операндов и способы адресации, обеспечиваемые архитектурой микроконтроллера, а также приводятся краткие рекомендации по программированию.

4.1 Типы операндов

Система команд поддерживает множество типов данных, которые обычно используются в управляющих алгоритмах.

Типы переменных операндов указаны заглавными буквами, например, BYTE – 8-битная беззнаковая переменная в инструкции.

В таблице 4.1 даётся общий обзор этих типов операндов.

Таблица 4.1 – Типы операндов

Тип операнда	Наличие знака и число бит		Возможные значения	Адресация
Бит (BIT)	без-знаковые	1	1 или 0	Как к компоненту байта
Байт (BYTE)		8	от 0 до (2^8–1) (от 0 до 255)	К байту
Слово (WORD)		16	от 0 до (2^{16}–1) (от 0 до 65 535)	К младшему байту (адрес должен быть кратным двум)
Двойное слово (DOUBLE WORD)		32	от 0 до (2^{32}–1) (от 0 до 4 294 967 295)	К младшему байту младшего слова в регистровом файле (адрес должен быть кратным четырем)
Короткое целое (SHORT INTEGER)	знаковые	8	от $(-2)^7$ до (2^7–1) (от –128 до 127)	К байту
Целое (INTEGER)		16	от $(-2)^{15}$ до (2^{15}–1) (от –32 768 до 32 767)	К младшему байту (адрес должен быть кратным двум)
Длинное целое (LONG INTEGER)		32	от $(-2)^{31}$ до (2^{31}–1) (от –2 147 483 648 до 2 147 483 647)	К младшему байту младшего слова в регистровом файле (адрес должен быть кратным четырем)
Примечание – 32-битные переменные поддерживаются только как операнды в инструкциях сдвига, как делимое при инструкции деления 32 на 16 и как результат действия умножения 16 на 16. Для совместимости с программными средствами сторонних производителей необходимо придерживаться правил, принятых в программировании на языке «Си» для адресации 32-битных операндов.				

Для каждого типа операнда есть соответствующий эквивалент для ассемблера и «Си». Ниже представлены типы операндов и их эквиваленты в порядке: тип операнда – эквивалент ассемблера/ эквивалент «Си».

BYTE – BYTE/ unsigned char
SHORT INTEGER – BYTE/ char
WORD – WORD/ unsigned int

INTEGER	–	WORD/ int
DOUBLE WORD	–	LONG/ unsigned long
LONG INTEGER	–	LONG/ long

Операнд бит (BIT)

BIT – одноразрядная переменная, которая может иметь булевы значения «TRUE» и «FALSE». Архитектура требует, чтобы к биту обращались как к компонентам байта или слова, так как прямая адресация бита не поддерживается.

Операнд байт (BYTE)

BYTE – 8-битная переменная без знака, которая может иметь значения от 0 до 255 (2^8–1). Арифметические операторы и операторы сравнения могут быть применены к операндам байтам, но результат должен интерпретироваться по модулю 256 арифметически. Логические действия с байтами осуществляются побитно. Биты в пределах байта нумеруются от 0 до 7, бит 0 – младший бит. Нет никаких ограничений выравнивания для байт, так что они могут быть помещены по любому адресу памяти.

Операнд слово (WORD)

WORD – 16-битная беззнаковая переменная, состоящая из двух байт (старшего и младшего), которая может иметь значения от 0 до 65 535 (2^{16}–1). Арифметические операции и операции сравнения могут быть применены к операндам словам, но результат должен интерпретироваться по модулю 65 536 арифметически. Логические действия со словами осуществляются побитно. Биты в пределах слова нумеруются от 0 до 15, бит 0 – младший бит.

Слова должны быть выровнены по границе четного байта адреса пространства. Младший байт слова находится по четному адресу, а старший байт находится в следующем (нечетном) адресе. Адрес слова – это адрес его младшего байта. Действия над словами, расположенными по нечетным адресам, не гарантируются.

Операнд двойное слово (DOUBLE WORD)

DOUBLE WORD – 32-битная переменная без знака, состоящая из двух слов (старшего и младшего), которая может принимать значения от 0 до 4 294 967 295 (2^{32}–1). Архитектура непосредственно поддерживает операнды двойные слова только как операнды в командах сдвига, делимого при делении 32/16 и произведения при умножении 16×16. Для этих действий переменная двойное слово должна находиться в младшем файле регистров и должна быть выровнена по адресу, который является кратным четырем. Адрес двойного слова – это адрес его самого младшего байта (четный адрес). Младшее слово двойного слова находится всегда в младшем адресе, даже когда данные находятся в стеке. Это означает, что старшее слово должно быть выдвинуто в стек первым.

Действия над двойным словом, которые непосредственно не поддерживаются, могут быть легко осуществлены с двумя словами. Например, следующие последовательности инструкций с 16 битами исполняют 32-битное сложение и 32-битное вычитание соответственно.

```
ADD   REG1, REG3      ; (2-операндное сложение)
ADDC  REG2, REG4
SUB   REG1, REG3      ; (2-операндное вычитание)
SUBC  REG2, REG4
```

Операнд короткое целое (SHORT INTEGER)

SHORT INTEGER – 8-битная переменная со знаком, которая может принимать значения от –128 (-2^7) до 127 (2^7–1). Арифметические действия, результат которых оказывается вне допустимого диапазона, устанавливают флаги переполнения в регистре PSW. Числовой результат тот же самый, что и результат эквивалентного действия над байтами. Нет никаких ограничений выравнивания для коротких целых, так что они могут быть помещены по любому адресу памяти.

Операнд целое (INTEGER)

INTEGER – 16-битная переменная со знаком, состоящая из двух байт (старшего и младшего), которая может принимать значения от –32 768 (-2^{15}) до 32 767 ($2^{15}-1$). Арифметические действия, результат которых оказывается вне допустимого диапазона, устанавливают флаги переполнения в регистре PSW. Числовой результат тот же самый, что и результат эквивалентного действия над словами.

Целое должно быть выровнено по границе четного байта в адресном пространстве. Младший байт целого находится по четному адресу, а старший байт находится в следующем (нечетном) адресе. Адрес целого – адрес его младшего байта. Действия над целыми, расположенными по нечетным адресам, не гарантируются.

Операнд длинное целое (LONG INTEGER)

LONG INTEGER – 32-битная переменная со знаком, состоящая из двух слов (старшего и младшего), которая может принимать значения от –2 147 483 648 (-2^{31}) до 2 147 483 647 ($2^{31}-1$). Архитектура непосредственно поддерживает операнды длинные слова только как операнды в командах сдвига, как делимое при делении 32/16 и как произведение при умножении 16×16. Для этих действий переменная длинное слово должна быть в младшем файле регистров и должна быть выровнена по адресу, который является кратным четырем. Адрес длинного целого – это адрес его самого младшего байта (четный адрес).

Действия над длинными целыми, которые непосредственно не поддерживаются, могут быть легко осуществлены с двумя целыми. Например, следующие последовательности инструкций с 16 битами исполняют 32-битное сложение и 32-битное вычитание соответственно.

```
ADD     REG1, REG3        ; (2-операндное сложение)
ADDC    REG2, REG4
SUB     REG1, REG3        ; (2-операндное вычитание)
SUBC    REG2, REG4
```

4.2 Способы адресации

Система команд микроконтроллера допускает следующие способы адресации: прямая регистровая, непосредственная, косвенная, косвенная с автоинкрементом, короткая индексная, длинная индексная, индексная с использованием нулевого регистра и с использованием указателя стека. Каждая команда использует, по крайней мере, один из способов адресации. Имеются расширенные версии команд прерываемой и непрерываемой передачи блока. Команды ассемблера состоят из мнемоники, за которой следуют адрес или данные. В представленных ниже примерах используются обозначения:

AX, BX, CX	–	16-разрядные регистры;
DX	–	8-разрядный регистр;
AH, AL	–	старший и младший байты регистра AX;
BH, BL	–	старший и младший байты регистра BX;
SP	–	регистр указателя стека;
MEM_WORD(reg)	–	содержимое ячейки памяти, адрес которой указан в регистре reg, в качестве которого может использоваться любой 8/16-разрядный регистр файла регистров.

Прямая регистровая адресация

Наиболее простой и быстрый способ адресации. Предполагает размещение операнда в файле регистров. Операнд имеет короткий байтовый адрес. С использованием оконной адресации памяти прямая регистровая адресация может быть распространена на операнды

с адресом, превышающим FFh. Для удобства целесообразно работать с символическими именами переменных, предварительно определив их.

Пример:

```
ADD CX, AX, BX          ; CX ← AX + BX
INCB DX                 ; DX ← DX + 1
```

Непосредственная адресация

Непосредственная адресация позволяет брать значение операнда непосредственно из поля команды (указывается после символа «#»). Для операций с переменными байтами и короткими целыми это будет 8-разрядное поле. Для операций с переменными типа слово и целое – 16-разрядное поле. Команда может содержать только один непосредственный операнд; для других операндов необходимо использовать прямую регистровую адресацию.

Операнды могут задаваться в любой системе счисления – двоичной (b), восьмеричной (o или q), шестнадцатеричной (h) или десятичной (d или отсутствие буквы)

Пример:

```
ADD AX, #340            ; AX ← AX + 340
PUSH #1234h             ; SP ← SP – 2, SP ← 1234h
DIVB AX, #10            ; AL ← AX/10, Ah ← AX MOD 10
```

Косвенная адресация

Косвенная адресация осуществляет доступ к операнду, адрес которого размещен в переменной типа слово регистрового файла. Вычисляемый адрес должен соответствовать правилам выравнивания. Следует заметить, что косвенная адресация позволяет обращаться к операнду в любом месте адресного пространства, включая файл регистров. 8-битное поле внутри команды выбирает регистр, который содержит косвенный адрес. Команда может содержать только одну косвенную ссылку, обращение к добавочным операндам осуществляется с помощью прямой регистровой адресации.

Пример:

```
LD AX, [AX]             ; AX ← MEM_WORD(AX)
ADDB AL, BL, [CX]       ; AL ← BL + MEM_BYTE(CX) (ADDB_3op)
POP [AX]                ; MEM_WORD(AX) ← MEM_WORD(SP), SP ← SP + 2
```

Косвенная адресация с автоинкрементом

Косвенная адресация, при выполнении которой содержимое регистра указателя адреса автоматически увеличивается на единицу при работе с байтами и короткими целыми, и на два – при работе со словами и целыми.

Пример:

```
LD AX, [BX]+            ; AX ← MEM_WORD(BX), BX ← BX + 2
ADDB AL, BL, [CX]+      ; AL ← BL + MEM_BYTE(CX)
                        ; CX ← CX + 1 (ADDB_3op)
PUSH [AX]+              ; SP ← SP – 2
                        ; MEM_WORD(SP) ← MEM_WORD(AX)
                        ; AX ← AX + 2
```

Короткая индексная адресация

При короткоиндексной адресации адрес одного из операндов вычисляется с помощью двух 8-битных полей. Одно 8-битное поле выбирает в файле регистров переменную типа слово, содержащую адрес. Второе 8-битное поле в команде имеет знаковое расширение и суммируется с переменной типа слово для формирования исполнительного адреса операнда. Исполнительный адрес может находиться на 128 байт ранее от текущего адреса или на 127 байт после него. Команда может содержать только один такой операнд, другие операнды задаются с помощью прямой регистровой адресации.

Пример:

```
LD AX, 12[BX]           ; AX ← MEM_WORD(BX+12)
```

```
MULB AX, BL, 3[CX]          ; AX ← BL * MEM_BYTE(CX+3), (MULB_3op)
```

Длинная индексная адресация

Длинноиндексная адресация похожа на короткоиндексную адресацию, за исключением того, что 16-битное поле берётся из команды и добавляется к переменной типа слово для формирования адреса операнда. Такая адресация может применяться только для одного операнда в команде, для других операндов необходимо использовать прямую регистровую адресацию.

Пример:

```
AND AX, BX, TABLE[CX]       ; AX ← BX AND MEM_WORD(TABLE+CX)
                            ; (AND_3op)
ST AX, TABLE[BX]            ; MEM_WORD(TABLE_BX) ← AX
ADDB AL, BL, LOOKUP[CX]     ; AL ← BL + MEM_BYTE(LOOKUP+CX)
                            ; (ADDB_3op)
```

Адресация с использованием регистра нуля

Кроме источника фиксированного нуля для расчетов и сравнения, регистр нуля ZERO_REG может использоваться как переменная типа слово в длинноиндексной адресации. Такая комбинация позволяет адресоваться к любому участку памяти. Так как этот способ использует индексную адресацию, то доступ осуществляется медленнее, чем при прямой регистровой адресации.

Пример:

```
ADD AX, 1234[ZERO_REG]      ; AX ← AX + MEM_WORD(1234) (ADD_2op)
POP 5678[ZERO_REG]          ; MEM_WORD(5678) ← MEM_WORD(SP)
                            ; SP ← SP + 2
```

Адресация с использованием указателя стека

Байты с адресами 18h и 19h файла регистров содержат указатель стека SP, к которому адресуются по адресу 18h. Кроме обычных операций, указатель стека SP может также использоваться как переменная типа слово в косвенной адресации для доступа к вершине стека или в короткоиндексной адресации для доступа к данным внутри стека.

Пример:

```
PUSH [SP]           ; Копирование вершины стека
LD AX, 2[SP]        ; AX ← NEXT_TO_TOP
```

4.3 Выборы способов адресации на языке ассемблера

Ассемблер упрощает выбор режимов адресации. Это упрощает задачу программирования, и эти возможности ассемблера следует применять там, где только это возможно.

Прямая адресация

Ассемблер выбирает между прямой регистровой адресацией и адресацией с использованием нулевого регистра, в зависимости от местоположения операнда в памяти. Необходимо обратиться к операнду с его символическим именем. Если операнд находится в младшем файле регистров, ассемблер выбирает прямую адресацию. Если операнд находится в другой области памяти, ассемблер выбирает адресацию с нулевым индексом.

Индексная адресация

Ассемблер выбирает между короткой индексной и длинной индексной адресациями в зависимости от значения индекса. Если значение может быть выражено в восьми битах, язык ассемблера выбирает короткую индексную адресацию. Если значение больше 8-битного, выбирается длинная индексная адресация.

4.4 Стандарты и соглашения программного обеспечения

При разработке программного обеспечения рекомендуется, чтобы использовались соглашения, принятые для редактирования на основе языка С. Эти стандарты применимы и для ассемблера, и для программной среды языка С, и они обеспечивают совместимость между этими средами.

Использование регистров

В 256-байтном младшем файле регистров содержатся регистры специальных функций центрального процессора и указатель стека. Остаток младшего файла регистров и старший файл регистров доступны для использования. Периферийные регистры специальных функций (SFR) и распределенные в карте памяти SFR расположены в верхней области адресного пространства. Периферийные SFR могут быть перемещены через окна в младший файл регистров для прямого доступа. Распределенные в карте памяти SFR не могут быть использованы в режиме окон. Можно использовать косвенную или индексную адресацию, чтобы получить доступ к ним. Все SFR доступны побайтно и пословно, если не определено иначе.

Язык программирования С адаптирован к простой эффективной стратегии, размещает восемь байтов, начинающихся в адресе 1Ch, во временное хранение и рассматривает оставшуюся область в файле регистров как сегмент запоминающего устройства, размещённый там, где необходимо.

Примечание – Использование любого SFR как базового или индексного регистра для косвенных или индексных инструкций может вызвать непредсказуемые результаты, потому что внешние события могут изменить содержимое SFRs. Так как некоторые SFRs очищаются при чтении, допустимо использование SFR как операнда в инструкциях «чтение-модификация-запись» (например, XORB).

Адресация 32-битных операндов

32-битные операнды (двойное слово и длинное целое) сформированы двумя смежными 16-битными словами в запоминающем устройстве. Младшее двойное слово находится всегда в младшем адресе, даже когда данные находятся в стеке (это означает, что старшее слово должно быть выдвинуто в стек первым). Адрес 32-битного операнда – адрес его младшего байта.

Аппаратные средства поддерживают 32-разрядные типы данных как операнды в инструкциях сдвига, делимое в делении 32 на 16 и произведение в умножении 16 на 16. Для этих действий 32-битный операнд должен располагаться в младшем файле регистров и должен быть выровнен по адресу, кратному четырем.

Соединение подпроцедур

Параметры передаются к подпроцедурам через стек. Параметры выдвигаются в стек от самого правого параметра налево. Параметры на восемь битов выдвигаются в стек со старшим неопределенным байтом. 32-битные параметры выдвигаются в стек как два 16-битных; старшая часть параметра выдвигается в стек первой. Как пример, рассмотрим следующую процедуру:

Void example_procedure (char param1, long param2, int param3)

Когда эта процедура выполняется, стек будет содержать параметры в следующем порядке:

word param3
low word of param2
high word of param2
byte param1
return address ← Stack Pointer.

Если процедура возвращает значение к коду запроса (в противоположность изменению более глобальных переменных), результат возвращается в область временного

хранения (TMPREG0 в этом примере), начинающуюся в 1Ch. TMPREG0 рассматривается или как 8-, 16-, или 32-битная переменная, в зависимости от типа процедуры.

Стандарт вызова, принятый языком С, имеет несколько следующих ключевых особенностей.

Процедуры могут всегда предполагать, что восемь байтов файла регистров, начинающиеся в 1Ch, могут использоваться для временного хранения в пределах тела процедуры.

Код, который вызывает процедуру, должен учитывать, что процедура изменяет восемь байтов файла регистров, начинающихся в 1Ch.

Код, который вызывает процедуру, должен предполагать, что процедура изменяет слово состояния процессора PSW, потому что процедуры не сохраняют и не восстанавливают PSW.

Результат процедур всегда возвращается в переменную TMPREG0.

Язык С позволяет определять прерывание процедур, которые выполняются. Прерывание процедур не соответствует правилам нормальных процедур. Параметры нельзя передать к этим процедурам, и они не могут возвратить результаты. Так как прерывание процедуры можно выполнить по существу в любое время, они должны сохранить и восстанавливать значения PSW и TMPREG0.

4.5 Защита и руководящие принципы программного обеспечения

В микроконтроллере реализовано несколько механизмов восстановления после ошибок программного обеспечения и аппаратных средств. Прерывание по невыполнимому коду инструкции обеспечивает защиту от выполнения некорректного кода инструкции. Команда аппаратного сброса RST может вызвать сброс, если программный счетчик выходит за границы. Код инструкции RST – FFh, поэтому процессор сбросит себя, если попробует выполнить инструкцию от незапрограммированных ячеек в энергонезависимом запоминающем устройстве или из шины, линии которой подключены к высокому уровню. Сторожевой таймер может сбросить микроконтроллер в случае аппаратной или программной ошибки. Блок отладки может инициализировать сброс в случае выхода в недопустимые области памяти.

Рекомендуется заполнить неиспользованные области кодом с NOP и периодическими переходами к процедуре обслуживания ошибок или RST инструкции. Это особенно важно для кодов окружения таблиц поиска. Везде, где место позволяет, необходимо окружить каждую таблицу семью NOP (потому что самая длинная инструкция устройства имеет семь байт) и RST или переходами к процедуре обслуживания ошибок. Так как RST – однобайтовая инструкция, NOP не нужны, если RST используются вместо переходов к процедуре ошибки.

При использовании сторожевого таймера для защиты программного обеспечения рекомендуется сбрасывать таймер только в одном месте программы, сокращая возможность нежелательного сброса. Секция кода, который сбрасывает сторожевой таймер, должна контролировать другие секции кода для правильного выполнения инструкций. Это может быть сделано проверкой переменных, чтобы удостовериться, что они – в пределах разумных значений.

5 Распределение памяти

Микроконтроллер имеет 64 Кбайт адресного пространства, большая часть которого предназначена для программ или данных. Адресное пространство общее для адресов и команд, но выполнение команд из некоторых областей памяти ограничено. Так, невозможно выполнение команд из областей регистрового файла и расширенного ОЗУ. При попытке чтения команд из этих адресов будет происходить обращение к внешней памяти.

На рисунке 5.1 показано разделение памяти для данных.

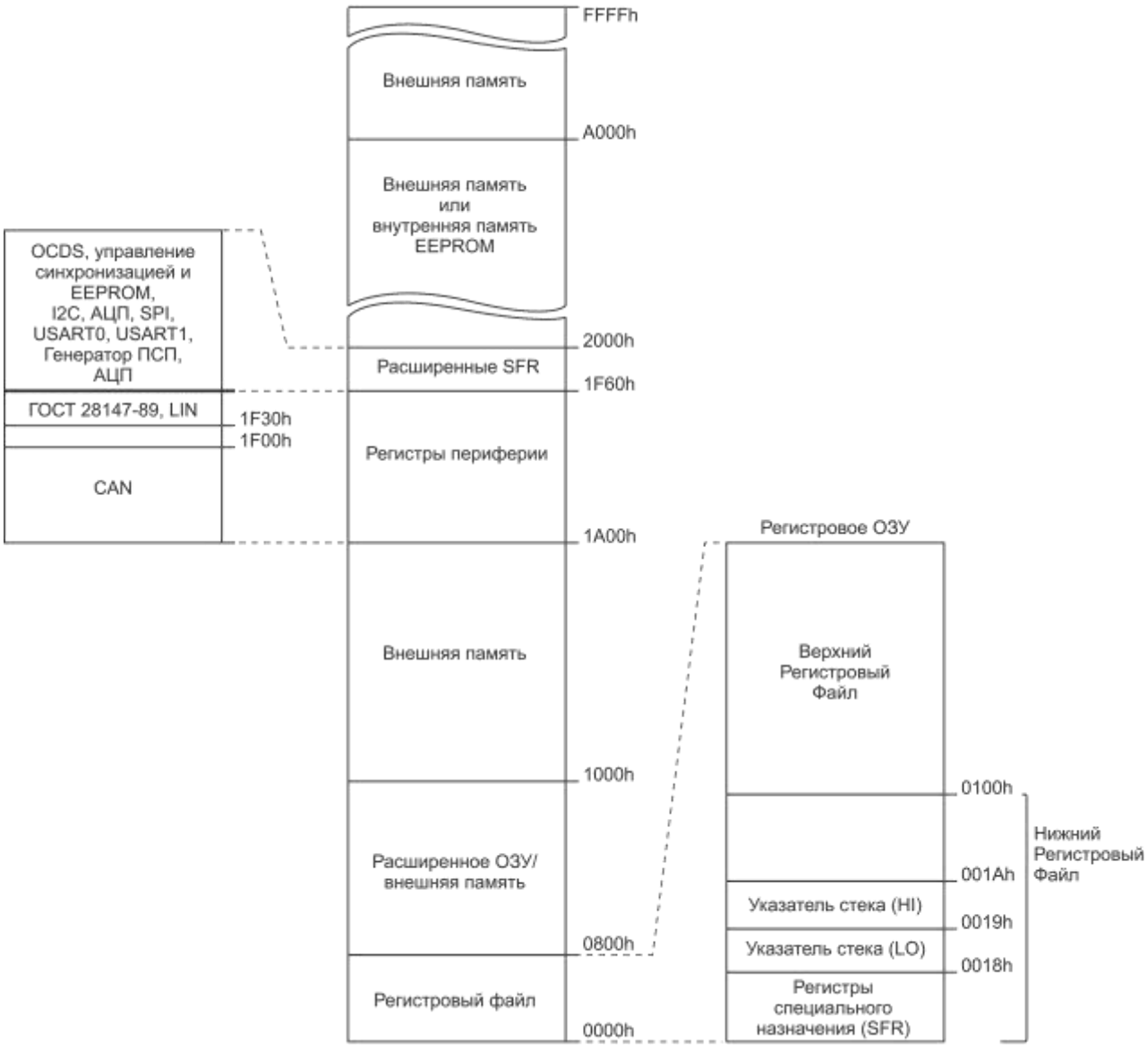


Рисунок 5.1 – Карта адресов памяти для данных

Распределение памяти для команд показано на рисунке 5.2.

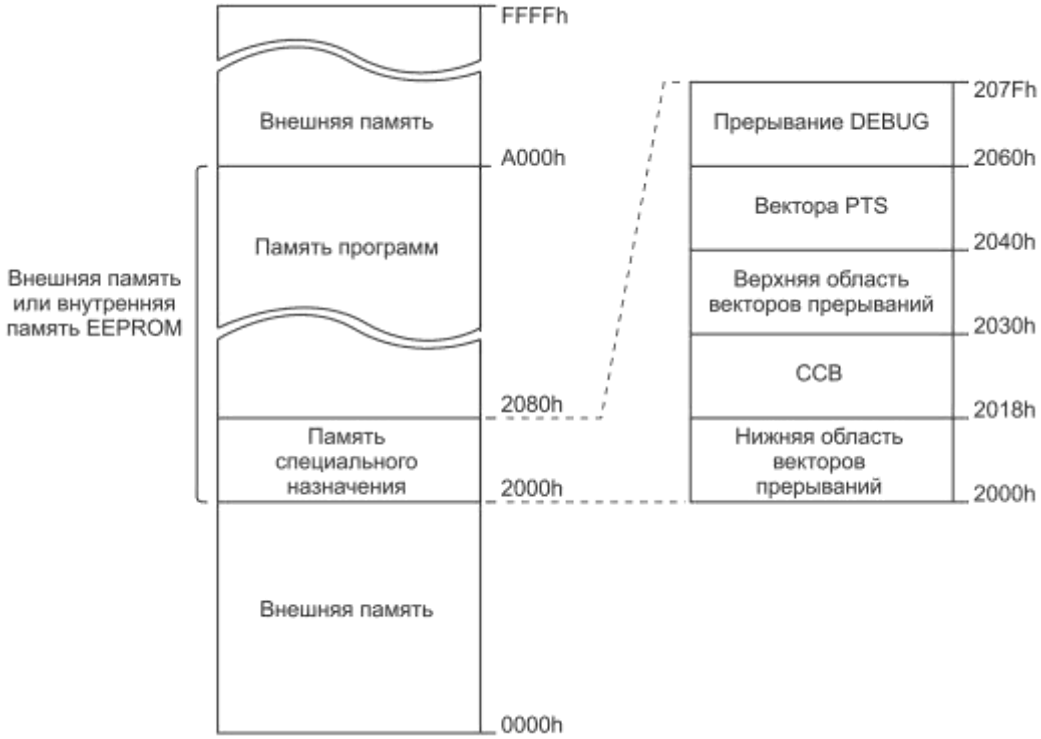


Рисунок 5.2 – Карта адресов памяти для команд

В таблице 5.1 приводятся начальные и конечные адреса памяти при обращении к данным, в таблице 5.2 – при обращении к командам. Команды могут располагаться либо в области внутренней EEPROM, либо в области внешней памяти.

Таблица 5.1 – Старшие адреса памяти при обращении к данным

Обозначение	Диапазон адресов
Внешняя память	A000h – FFFFh
Внешняя или внутренняя память EEPROM	2000h – 9FFFh
Расширенные SFR	1F70h – 1FFFh
Внешняя память	1000h – 1F6Fh
Расширенное ОЗУ/внешняя память	0800h – 0FFFh
Регистровый файл (включая SFR)	0000h – 07FFh

Таблица 5.2 – Старшие адреса памяти при обращении к командам

Обозначение	Диапазон адресов
Внешняя память	A000h – FFFFh
Программная память *	2080h – 9FFFh
Память специального назначения *	2000h – 207Fh
Внешняя память	0000h – 1FFFh

_____

* Может находиться как в области внутренней EEPROM, так и во внешней памяти.

5.1 Разделы внешней памяти для данных

Область выше расширенных SFR или внутренней EEPROM и область выше расширенного ОЗУ или регистрового файла всегда назначается на внешнюю память данных. Обращение к ней осуществляется через шину внешних адресов/данных. Разделение памяти данных и памяти команд осуществляется сигналом INST.

Расширенные регистры SFR

Область расширенных регистров SFR содержит регистры управления программированием ПЗУ, тактированием периферийных блоков, расширенного управления АЦП, отладочным модулем и порты 3 и 4. Порты 3 и 4 читаются и записываются как слово по адресу 1FFEh (порт 3 – младший байт; порт 4 – старший байт). Эти порты работают либо как порты ввода-вывода, либо как мультиплексированная шина адрес/данные, либо их комбинация, в зависимости от EA# сигнала при сбросе.

Расширенное ОЗУ

Область расширенного ОЗУ предназначена для хранения больших массивов данных, доступ к которым осуществляется косвенными методами адресации. Обращение к этой области занимает больше машинных циклов, чем обращение к регистровому файлу прямым методом, но намного меньше, чем к внешней ОЗУ. Расширенное ОЗУ имеет функцию отключения. При этом область 0800h – 0FFFh адресуется в область внешней памяти.

5.2 Программная память и память специального назначения

Программная память и память специального назначения занимают раздел памяти 32 Кбайт (рисунок 5.2 и таблица 5.2), который может постоянно располагаться либо во внутренней EEPROM, либо во внешней памяти. Эта область является общей для команд и данных в случае работы ИС в режиме включенного ПЗУ, что позволяет программировать ПЗУ путем записи нужных данных в нужные ячейки.

Выбор внутренней или внешней памяти

Доступ к адресам программной памяти и адресам памяти специального назначения осуществляется в зависимости от того, где находится данный раздел: во внутреннем EEPROM или во внешней памяти, причём он не может находиться и там, и там одновременно. Значение вывода внешнего доступа EA# во время переднего фронта сигнала RESET# определяет вид доступа: внутренний или внешний. Это значение запоминается в устройстве и не может быть изменено до сброса устройства. Если EA# имеет низкий уровень, то внутреннее EEPROM недоступно, и все обращения к этому адресному диапазону направляются прямо во внешнюю память. Если EA# имеет высокий уровень, то обращение к этому адресному диапазону осуществляется во внутреннем EEPROM. Время включения EEPROM составляет 5 мкс, поэтому необходимо обеспечить соответствующую длительность сигнала RESET при первом использовании режима работы с внешней памятью или переключении в этот режим. Время выполнения программы из внутренней памяти равно времени выполнения программы из внешней памяти, имеющей 16-битный доступ с одним дополнительным циклом ожидания.

Примечание – Вывод EA# должен иметь высокий уровень для надежного функционирования ЦПУ без внешней памяти.

Память программ

Программная память располагается по адресам 2080h – 9FFFh. Существует возможность запуска программы с различных начальных адресов.

Стандартный режим работы. Начальный адрес – 2080h.

Режим старта из области внешней памяти A000h. Для этого надо удерживать сигналы VPR и ALE на низком уровне во время сброса. Может использоваться только при работе в режиме внутренней памяти команд.

Режим старта с начального адреса 9000h. Для этого надо удерживать VPR на низком уровне, а ALE в высоком во время сброса. Если выбран режим работы с внутренней памятью, то начнется выполнение программы – монитора, реализующей функции программирования ПЗУ.

Примечание – Так как исходное значение ячеек внутренней памяти EEPROM равно 00h, то рекомендуется записывать FFh в неиспользуемые ячейки программной памяти, так как при неправильном ходе выполнения программы это приведет к перезагрузке схемы.

5.3 Память специального назначения

Память специального назначения расположена по адресам 2000h – 207Fh (см. рисунок 5.2). Она содержит несколько зарезервированных ячеек памяти, векторы для периферийного сервера PTS и стандартных прерываний. В таблице 5.3 приведен список адресов памяти специального назначения а также начальные и конечные адреса.

Верхняя и нижняя области векторов прерываний содержат адреса подпрограмм обслуживания прерываний. PTS-векторы содержат адреса управляющих блоков PTS.

Таблица 5.3 – Адреса памяти специального назначения

Обозначение	Диапазон адресов
Общего назначения	2062h – 207Fh
Вектор прерывания DEBUG	2060h
Общего назначения	205Eh – 205Fh
Векторы PTS	2040h – 205Dh
Старшие векторы прерывания	2030h – 203Fh
Общего назначения	2019h – 202Fh
CCB	2018h
Общего назначения	2014h – 2017h
Младшие векторы прерывания	2000h – 2013h

Байт конфигурации кристалла CCB

Байт конфигурации кристалла CCB – первый байт, выбранный из памяти после сброса устройства. Последовательность сброса загружает CCB во внутренний регистр, называемый регистром конфигурации кристалла (CCR). Регистр CCR управляет защитой внутренней памяти, режимом READY, сигналом управления шиной, разрядностью шины и режимом POWERDOWN.

5.4 Регистровый файл

Регистровый файл разделён на верхний и нижний регистровые файлы (см. рисунок 5.2). Верхний регистровый файл содержит регистровое ОЗУ общего назначения. Нижний регистровый файл содержит регистровое ОЗУ общего назначения, указатель стека (SP) и регистры специальных функций (SFR). В таблице 5.4 приведены начальные и конечные адреса.

Таблица 5.4 – Адреса памяти регистрового файла

Обозначение	Диапазон адресов
Верхний регистровый файл	0100h – 07FFh
Регистровое RAM (ОЗУ) общего назначения	001Ah – 00FFh
Указатель стека SP	0018h – 0019h
Регистры специального назначения SFR	0000h – 0017h

Регистровое ОЗУ общего назначения

Ячейки 001Ah – 00FFh содержат регистровое RAM (ОЗУ) общего назначения. Ячейки указателя стека 0018h – 0019h могут также использоваться в качестве регистрового ОЗУ общего назначения, когда операции со стеком не проводятся. РАЛУ может прямо обращаться к этой памяти, используя прямую регистровую адресацию.

Верхний регистровый файл также содержит регистровое ОЗУ общего назначения (0100h – 07FFh). Обычно РАЛУ обращается к этой памяти, используя косвенную или индексную адресацию. Технология вертикальных окон позволяет РАЛУ использовать прямую регистровую адресацию для доступа к ОЗУ в верхнем регистровом файле. Создание вертикальных окон обеспечивает быстрое переключение на задачи по прерываниям и ускоряет выполнение программ.

Указатель стека SP

Ячейки памяти 0018h и 0019h содержат указатель стека SP. SP содержит адрес вершины стека. SP должен указывать на адрес слова (четный), который на два байта больше, чем желаемый стартовый адрес. Перед тем, как ЦПУ выполнит вызов подпрограммы или программы обслуживания прерывания, оно дважды декрементирует содержимое SP и копирует (PUSH) адрес следующей команды из программного счётчика в стек. Затем загружает адрес подпрограммы или программы обслуживания прерываний в программный счётчик. После завершения выполнения подпрограммы или программы обслуживания прерывания, ЦПУ возвращается из подпрограммы (командой RET) и загружает (командой POP) содержимое вершины стека (т. е. адрес возврата) в программный счётчик.

Подпрограммы могут быть вложенными. Это значит, что каждая подпрограмма может вызывать другую подпрограмму. ЦПУ засылает содержимое программного счётчика в стек каждый раз при выполнении вызова подпрограммы. Стек растёт вниз по мере добавления содержимого. Глубину стека ограничивает только величина доступной памяти. Каждый раз при возврате из подпрограммы ЦПУ выгружает адрес из вершины стека, и потом вершина стека перемещается на следующий адрес возврата. Когда операции со стеком не проводятся, ячейки SP могут использоваться в качестве регистрового ОЗУ общего назначения.

Инициализация указателя стека

Пользователь программ должен загрузить двухбайтный адрес в указатель стека. Выбор адреса, на два байта большего, чем желаемый стартовый адрес, делается потому, что указатель стека автоматически декрементируется перед тем, как ЦПУ засылает первый байт адреса возврата в стек. Следует помнить, что стек растёт вниз, что требует достаточного количества ячеек для максимального числа адресов, занесённых в стек. Стек может находиться во внутреннем регистровом файле, расширенном ОЗУ, либо во внешнем ОЗУ.

Регистры специальных функций SFR

Ячейки 0000h – 0017h обеспечивают доступ к регистрам специальных функций SFR через три горизонтальных окна. РАЛУ осуществляет прямое управление всеми периферийными модулями, кроме портов 3 и 4, через SFR.

При использовании SFR в качестве базового или индексного регистра при косвенной или индексной адресациях следует отдавать отчёт в том, что содержимое SFR непредсказуемо. Внешние события могут менять содержимое SFR, и некоторые SFR очищаются при считывании.

Функции многих SFR различны в зависимости от того, считывается из них или записывается в них информация. По этой причине никогда не следует использовать SFR в качестве операнда в команде чтение-модификация-запись. Не следует использовать зарезервированные SFR – в них следует записать 00h или оставить в исходном состоянии. При считывании зарезервированные биты и SFR могут возвращать неопределённые значения.

5.5 Горизонтальные окна

В микроконтроллере применяется технология горизонтальных окон для обращения к регистрам специального назначения. Используются три окна, каждое из которых позволяет «видеть» 24 байта блока памяти с адресами от 0000h до 0017h. Это горизонтальное окно 0, горизонтальное окно 1 и горизонтальное окно 15 (см. рисунок 5.3).

Рисунок 5.3 – Горизонтальные окна

Таблица 5.5 – Горизонтальные окна адресации

Адрес	Активное горизонтальное окно					Адрес
	0		**15**		**1**	
	Запись	Чтение	Запись	Чтение	Запись/Чтение	
17h	PWM0_ CONTROL	IOS2		PWM0_ CONTROL	PWM2_ CONTROL	17h
16h	IOC1	IOS1		IOC1	PWM1_ CONTROL	16h
15h	IOC0	IOS0		IOC0		15h
14h	**WSR**					14h
13h	INT_MASK1					13h
12h	INT_PEND1					12h
11h	SP_CON0	SP_STAT0		SP_CON0	DACVAL	11h
10h	IOPORT2		–			10h
0Fh	IOPORT1				–	0Fh
0Eh	BAUD_RATE0	IOPORT0				0Eh
0Dh	TIMER2		T2CAPTURE			0Dh
0Ch					IOC3	0Ch
0Bh	IOC2	TIMER1		IOC2	–	0Bh
0Ah	WATCHDOG			WATCHDOG		0Ah
09h	INT_PEND					09h
08h	INT_MASK					08h
07h	SBUF_TX0	SBUF_RX0		SBUF_TX0	PTSSRV	07h
06h	HSO_ COMMAND	HSI_STATUS		HSO_ COMMAND		06h
05h	HSO_TIME	HSI_TIME		HSO_TIME	PTSSEL	05h
04h						04h
03h	HSI_MODE	–		HSI_MODE	–	03h
02h	–	–		–	–	02h
01h	ZERO_REG					01h
00h						00h

Для управления переключением окон используется регистр WSR. Этот регистр «виден» в каждом окне по одному адресу (14h) и всегда доступен как для записи, так и для чтения. Доступ к остальным регистрам SFR осуществляется иначе. Так, например, регистр сторожевого таймера WATCHDOG, расположенный по адресу 0Ah, доступен для записи только при включенном нулевом горизонтальном окне, а для чтения – при включенном 15.

В таблице 5.5 указано, какие регистры SFR доступны для записи/чтения в зависимости от активного горизонтального окна. Если регистр 16-разрядный и занимает две ячейки в одном окне, то младший байт регистра всегда расположен по четному адресу, а старший – по нечетному. Серым цветом выделены регистры, всегда доступные для записи и чтения, вне зависимости от того, какое окно активно.

5.6 Вертикальные окна

Технология вертикальных окон заключается в том, чтобы создать в области младшего файла регистров «окно» – область, на которую можно отобразить какую-либо другую область памяти (другое «окно»), в том числе с 16-разрядными адресами, и иметь возможность использовать прямую регистровую адресацию для обращения к регистрам, к которым стандартно можно обратиться только косвенно или посредством индексной адресации. «Окно» в области младшего файла регистров – это базовое вертикальное окно.

Всю область ОЗУ можно разделить на 64 окна объемом по 32 байта, 32 окна объемом по 64 байта или 16 окон объемом по 128 байт (см. рисунок 5.4). Нумерация окон начинается с области, начальный адрес которой 0000h и используется для указания окна, которое будет «просматриваться» посредством базового вертикального окна. Начальный адрес базового окна зависит от количества окон. Так, при 64 окнах базовое окно будет располагаться в области адресов с 00E0h по 00FFh, при 32 – с 00C0h по 00FFh, при 16 – с 0080h по 00FFh.

Для управления включением/переключением вертикальных окон используется регистр WSR. Так, для включения режима 32-байтных вертикальных окон старшие два бита регистра WSR должны быть в состоянии 01b, при этом оставшиеся шесть бит задают порядковый номер окна, которое будет спроецировано на область базового вертикального окна. Для примера выбрано окно с номером 2Eh, расположенное в области адресов с 05C0h по 05DFh, которое проецируется в базовое окно. Теперь для обращения к ячейке памяти, расположенной по адресу 05C0h, достаточно указывать адрес 00E0h (см. рисунок 5.5).

Для включения режима 128-байтных вертикальных окон старшие четыре бита регистра WSR должны быть в состоянии 0001b, при этом оставшиеся четыре бита задают порядковый номер окна, которое будет спроецировано на область базового вертикального окна. Для примера выбрано окно с номером 0Dh, расположенное в области адресов с 0680h по 06FFh, которое проецируется в базовое окно. Теперь для обращения к ячейке памяти, расположенной по адресу 0680h достаточно указывать адрес 0080h (см. рисунок 5.6).

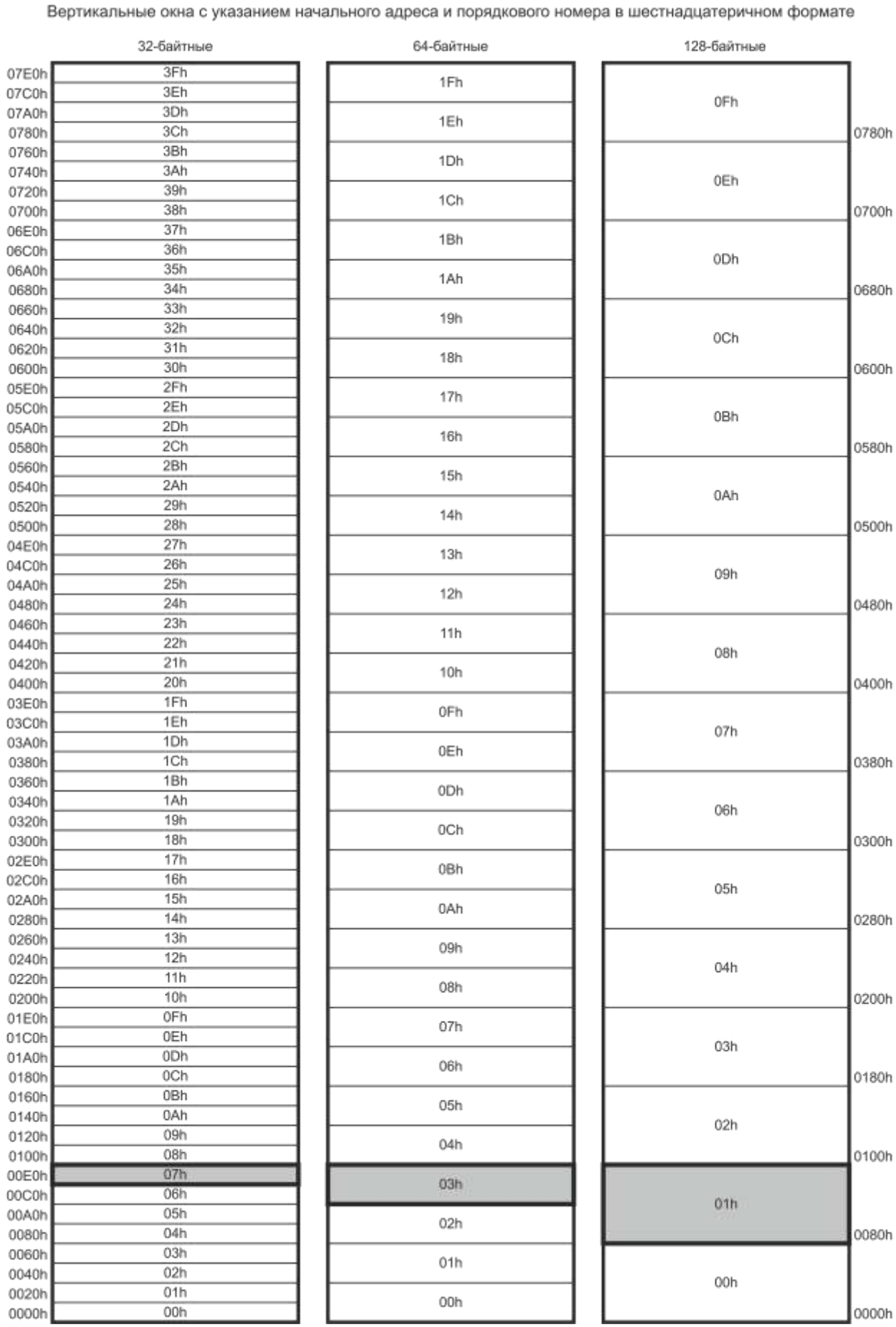


Рисунок 5.4 – Три варианта разделения ОЗУ на окна

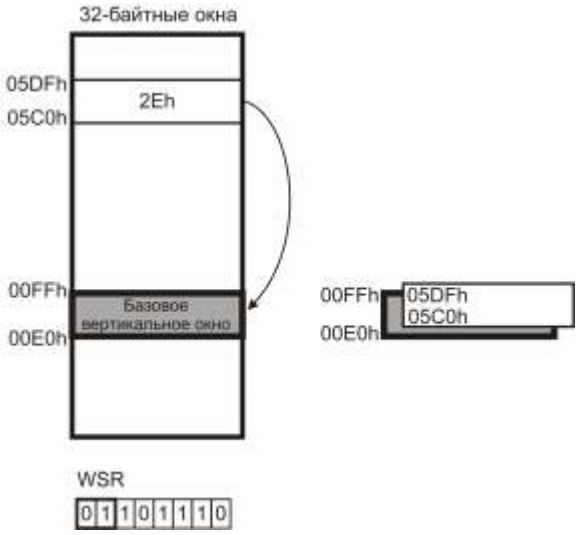


Рисунок 5.5 – Выбор вертикального окна при работе с 32-байтными окнами

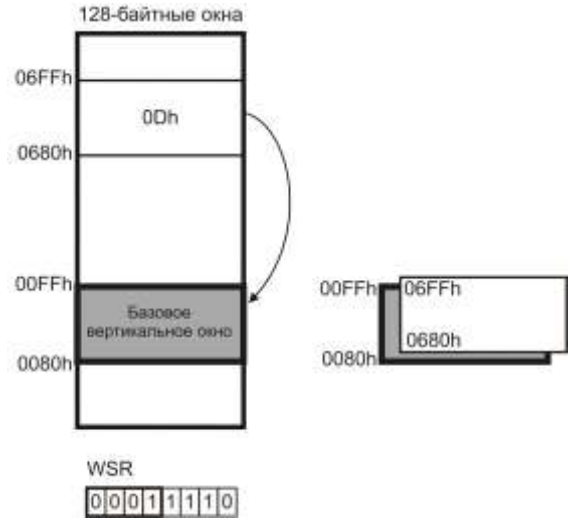


Рисунок 5.6 – Выбор вертикального окна при работе с 128-байтными окнами

Важно помнить:

- при работе с вертикальными окнами (в старшем нибле регистра WSR находится значение, отличное от 0000b) механизм горизонтальных окон не работает. По умолчанию активно горизонтальное окно 0;

- при работе с горизонтальными окнами (в старшем нибле регистра WSR находится значение 0000b) механизм вертикальных окон не работает.

Команда PUSHA автоматически сохраняет содержимое WSR в стеке, а команда POPA восстанавливает прежнее значение WSR из стека. В процедуре обслуживания прерывания после команды PUSHA можно выбрать необходимое вертикальное окно и быстро записать в него результаты расчетов или, наоборот, считать значения переменных.

Восстановление прежнего значения WSR произойдет автоматически после команды POPA в конце процедуры обслуживания прерывания (перед RET).

Ниже приведен фрагмент программы, иллюстрирующий один из возможных механизмов определения переменной в верхнем регистровом файле, и способы доступа к этой переменной посредством прямой регистровой и индексной адресации. В режиме вертикальных окон выбирается окно с номером 07h, которому принадлежит ячейка памяти, расположенная по адресу 0382h.

```
        VAR1 EQU   0382H :WORD ; 16-разрядный адрес ячейки памяти

        PUSHA                  ; загрузка содержимого WSR в стек
        LDB   WSR, #17H        ; режим 128-байтных окон, выбрано окно с номером 07h

;Команда с обычной индексной адресацией
        ADD AX, VAR1[0]        ; AX ← AX + VAR1 + 0

;Те же действия, но с использованием окна
        ADD AX, 82H            ; AX ← AX + VAR1

;Команды с обычной индексной адресацией (после сложения переменных регистров AX и
;VAR1 результат нужно сохранить в регистре VAR1)

        ADD AX, VAR1[0]        ; AX ← AX + VAR1 + 0
        ST AX, VAR1[0]         ; VAR1 ← AX

;Те же действия, но с использованием окна
        ADD 82H, AX            ;VAR1 ← VAR1 + AX

        POPA                   ; перегрузка первоначальной информации из стека в WSR
```

6 Прерывания

Основной функцией микроконтроллера является обеспечение управления устройствами в режиме реального времени. В состав микроконтроллера входят два блока обработки прерываний – контроллер прерываний, позволяющий осуществлять программное обслуживание прерываний, и сервер периферийных транзакций (далее PTS), являющийся аппаратным обработчиком прерываний без задействования ЦПУ.

В микроконтроллере выделены 17 векторов прерываний для 44 источников прерывания. Встроенная периферия, внешний сигнал или команда могут сформировать запрос на прерывание. В простейшем случае центральный процессор получает запрос на прерывание, приостанавливает выполнение текущей программы, обслуживает запрос и возвращается к выполнению программы.

6.1 Блоки обслуживания прерываний

Для маскируемых прерываний может быть выбран любой из двух блоков обработки прерываний. Немаскируемые прерывания (в том числе специальные – пошагового исполнения, NMI, DEBUG и по неподдерживаемому коду) всегда обрабатываются контроллером прерываний (т.е. только программно).

Контроллер прерываний

Контроллер прерываний обслуживает прерывания совместно с подпрограммами обработки прерываний. Когда аппаратура обнаруживает прерывание, она генерирует и выполняет специальный вызов подпрограммы обслуживания прерывания. При этом в стек помещается содержимое программного счётчика, а затем счетчик команд PC загружается значением, соответствующим вектору прерывания (верхние и нижние области векторов прерываний, расположенные в специально отведённой области внешней памяти, содержат адреса подпрограмм обслуживания прерываний). Далее ЦПУ выполняет подпрограмму обслуживания прерывания, после чего программный счетчик PC загружается значением из стека, и продолжается выполнение прерванной программы.

Сервер периферийных транзакций PTS

PTS представляет собой блок аппаратной обработки прерываний (только маскируемых), который может использоваться вместо стандартных программ обработки прерываний и содержащий набор встроенных алгоритмов, исходные данные для которых должны быть размещены во внутреннем ОЗУ и/или во внешней памяти.

PTS предназначен для обеспечения некоторой последовательности событий, выполняемых на микропрограммном уровне в заданное время без участия процессора. Такими событиями могут быть:

- передача блока информации из одного места памяти (или устройства ввода-вывода) в другое;
- выгрузка данных из блока HSI;
- загрузка данных в блок HSO.

Алгоритмы PTS охватывают, в основном, пересылки данных. Пересылка может осуществляться и в режиме IDLE.

PTS обслуживает прерывания параллельно работе основной программы, не модифицирует стек или PSW и осуществляет передачу данных только во время простоя шины данных (моменты, когда осуществляются арифметические операции, переходы, простои вследствие пустой очереди команд и т.п.). Пример передачи данных представлен на рисунке 6.1. Более подробное описание блока PTS представлено ниже.

Рисунок 6.1 – Пример передачи PTS данных

PTS работает в пяти специальных микропрограммных режимах, которые позволяют ему выполнять отдельные задачи гораздо быстрее, чем подпрограммам обслуживания прерываний. Если возникает запрос на прерывание, то специальный декодировщик приоритетов выбирает соответствующий вектор (из таблицы векторов PTS) и вызывает специальный блок управления (PTSCB). Каждое прерывание PTS генерирует один цикл, показанный на рисунке 6.2.

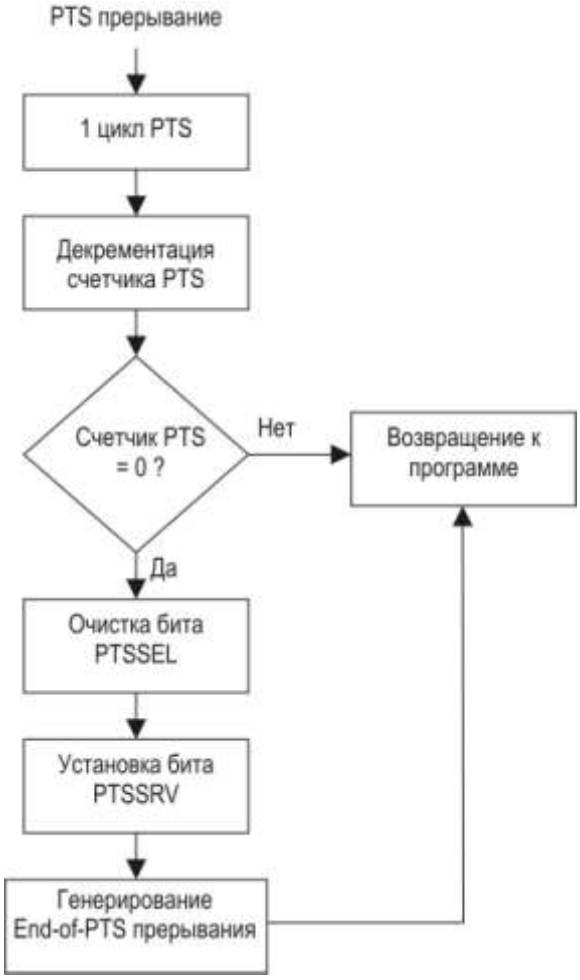


Рисунок 6.2 – Цикл прерывания PTS

На рисунке 6.3 представлен алгоритм обслуживания стандартного прерывания и прерывания PTS.

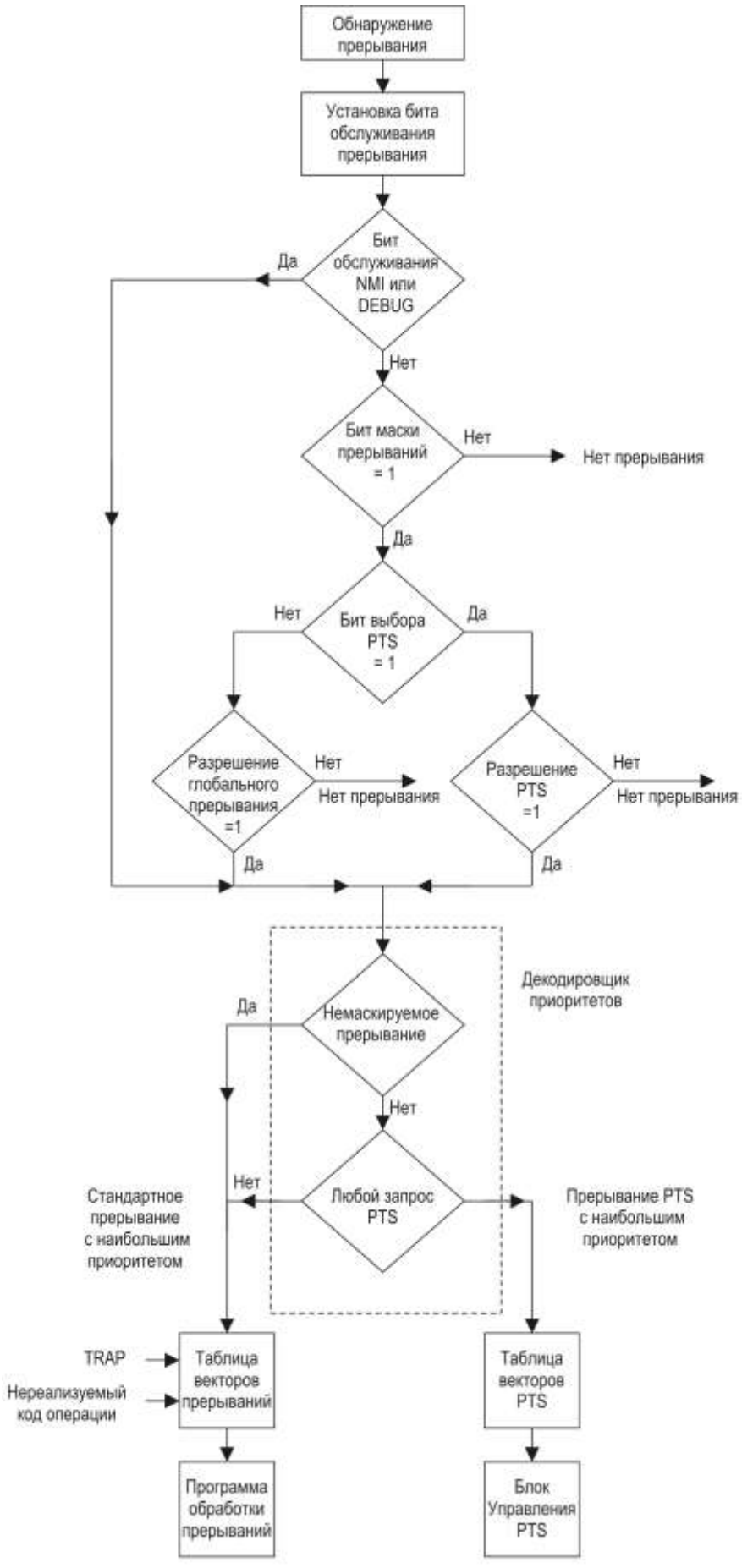


Рисунок 6.3 – Алгоритм обслуживания стандартного прерывания и прерывания PTS

6.2 Управление прерываниями

Приоритеты прерываний

Прерывания по неподдерживаемому коду и программные прерывания не имеют приоритетов и напрямую проходят в контроллер прерываний для обслуживания. Контроллер прерываний выбирает соответствующий вектор (ячейку памяти), расположенный в памяти специального назначения (см. рисунок 6.2). Вектор (ячейка памяти) содержит стартовый адрес программы обработки прерываний.

Все остальные прерывания имеют собственные уровни приоритета, определяемые декодировщиком приоритетов (уровень 16 является высшим, 0 – низшим).

В таблице 6.1 представлены все векторы прерываний с указанием их номеров, адресов и приоритетов.

Таблица 6.1 – Векторы прерываний

Источник прерывания	Номер	Вектор прерывания	Ячейка вектора прерывания	Ячейка вектора PTS	При-ори-тет
Неподдерживаемый код	–	UNIOPCODE	2012h	–	–
Команда TRAP	–	SWTRAP	2010h	–	–
Модуль отладки	INT16	DEBUG	2060h	–	16
NMI	INT15	NMI	203Eh	–	15
Буфер FIFO HSI полный	INT14	HSIFIFOFULL	203Ch	205Ch	14
Внешнее прерывание по выводу P2.2	INT13	EXTINT1	203Ah	205Ah	13
Переполнение таймера 2	INT12	T2OVF	2038h	2058h	12
Захват значения таймера 2	INT11	T2CAP	2036h	2056h	11
Четвёртая запись в FIFO модуля HSI	INT10	HSIFIFO4	2034h	2054h	10
Флаги RI модулей USART0 и USART1, 1 линия прерываний CAN	INT09	RI_USART, CAN1	2032h	2052h	9
Флаги TI модулей USART0 и USART1, 0 линия прерываний CAN	INT08	TI_USART, CAN0	2030h	2050h	8
Внешнее прерывание по выводу P2.2 или по выводу P0.7	INT07	EXTINT	200Eh	204Eh	7
Прерывание SPI, I2C, LIN, 2 линия прерываний CAN	INT06	SERPORT, CAN2	200Ch	204Ch	6
Программное прерывание 0/1/2/3, сброс таймера 2, запуск АЦП	INT05	SWTIMER	200Ah	204Ah	5
Вход 0 модуля HSI	INT04	HSI0	2008h	2048h	4
Выходы 0 – 5 модуля HSO	INT03	HSOINT	2006h	2046h	3
Заполнение FIFO или загрузка фиксирующего регистра модуля HSI	INT02	HSIDATAV	2004h	2044h	2
Завершение аналого-цифрового преобразования или срабатывание цифровых компараторов	INT01	ADCOM	2002h	2042h	1
Таймер 1 или таймер 2	INT00	TOVF	2000h	2040h	0
Примечание – Каждое из маскируемых прерываний с номерами INT14 – INT00 может быть назначено на PTS. Любое из PTS прерываний имеет приоритет над всеми другими маскируемыми прерываниями.					

Любой запрос прерывания PTS имеет больший приоритет, чем запрос маскируемого прерывания. Если нет запроса прерывания от модуля отладки и немаскируемого прерывания (NMI), то декодировщик приоритетов устанавливает высший приоритет для запроса прерываний PTS, и контроллер прерываний выбирает соответствующий вектор PTS (в области адресов 2040h – 205Dh внешней памяти), который содержит стартовый адрес специального блока управления PTS. Если нет и прерывания PTS, то декодировщик приоритетов устанавливает наивысший приоритет запросам стандартных прерываний, и контроллер прерываний выбирает соответствующий вектор (из памяти специального назначения внешней памяти), который содержит стартовый адрес соответствующей подпрограммы обработки прерывания.

Изменение приоритетов прерываний

Приоритеты маскируемых прерываний, определенные по умолчанию, можно перепрограммировать посредством регистров масок прерываний INT_MASK и INT_MASK1. Можно задать, какие прерывания будут обслуживаться подпрограммой обработки прерывания, а какие – в цикле PTS. Ниже представлен вариант подпрограммы обработки запроса прерывания от модуля USART0, которая запрещает все прерывания, кроме EXTINT (вектор EXTINT с приоритетом 7, вектор REC с приоритетом 9).

```
Serial_int:          ; Вход в подпрограмму обработки прерывания
PUSHA                ; Сохранение регистров PSW, INT_MASK и WSR
DI                   ; Запрет обслуживания всех прерываний
LDB INT_MASK1, #20h  ; Разрешение внешнего прерывания
EI                   ; Разрешение обслуживания всех прерываний
POPA                 ; Восстановление регистров PSW, INT_MASK и WSR
RET                  ; Выход из подпрограммы обработки прерывания
```

Вектор прерывания REC, соответствующий прерыванию модуля USART0, расположен по адресу 2032h, и должен содержать адрес, по которому расположена метка Serial_int.

Пояснение к приведенной выше программе

После обнаружения прерывания и определения его приоритета, аппаратная часть запускает механизм обслуживания прерывания. Значение программного счетчика PC помещается в стек, а сам он загружается значением из ячейки вектора, соответствующего немаскируемому прерыванию с наибольшим приоритетом. Далее никакое другое прерывание не будет обслужено до тех пор, пока не будет выполнена первая команда подпрограммы обработки прерывания (в примере это команда PUSHA).

Команда PUSHA, которая гарантированно будет выполнена, помещает в стек значения регистров PSW, INT_MASK1 и WSR, после чего очищает регистры PSW и INT_MASK1. Очистка регистра PSW, помимо сброса арифметических флагов, сбрасывает флаги I и PSE, что в совокупности с очисткой регистра INT_MASK1 маскирует все маскируемые прерывания, запрещает стандартную процедуру обслуживания прерываний и PTS.

Команда DI запрещает обслуживание всех прерываний, после чего программирование регистра INT_MASK1 (например, посредством команды LDB), позволяет разрешить и/или запретить прерывания от разных источников, тем самым создавая возможность программного распределения приоритетов (в примере разрешается только внешнее прерывание с вектором EXTINT). После этого обслуживание всех прерываний разрешается командой EI, но запросы на прерывания начнут обслуживаться только после выполнения команды, следующей за командой EI (в примере после команды POPA). В приведенном примере между командами EI и POPA не выполняется никаких действий, но фактически в этом месте может располагаться программа пользователя.

В конце программы обработки прерывания находится команда POPA, которая загружает регистры PSW, INT_MASK1 и WSR их исходными значениями, которые хранились в стеке (естественно, все значения, записанные в эти регистры в течение

выполнения программы обработки прерывания, теряются). Запросы на прерывания начнут обслуживаться только после выполнения команды, следующей за командой POPA, т.е. после выполнения команды RET, что автоматически исключает ситуацию запуска механизма обслуживания следующего прерывания до завершения обслуживания текущего, которая могла бы привести к переполнению стека.

Внешние прерывания и временные ограничения

Поддерживаемые внешние источники прерываний указаны в таблице 6.2.

Таблица 6.2 – Внешние источники прерываний

Вывод микроконтроллера	Альтернативная функция вывода	Номер прерывания	Вектор прерывания
NMI	–	INT15	NMI
P0.7	EXTINT	INT07	EXTINT
P2.2	EXTINT	INT07	EXTINT
		INT13	EXTINT1
	HSI.0	INT04	HSI0
P2.7	T2CAP	INT11	T2CAP

На всех выводах, которые используются в качестве источников прерываний, активным уровнем сигнала является уровень логической единицы, поэтому аппаратно обнаруженный переход сигнала из «нуля» в «единицу» интерпретируется как запрос прерывания. Для схемы мониторинга внешних прерываний важно, чтобы высокий уровень сигнала на выводе сохранялся в течение не менее двух тактов синхросигнала микроконтроллера.

В системе команд микроконтроллера есть шесть особых команд – DI, EI, POPA, POPF, PUSHA и PUSHF. Каждая их этих команд (помимо своего основного назначения) запрещает обслуживание всех прерываний до тех пор, пока не будет выполнена следующая за ней команда.

Прерывания также запрещаются после:

- прерывания по неподдерживаемому коду;
- прерывания по команде TRAP (программное прерывание).

Отклик на прерывание

От момента обнаружения запроса на прерывание до выборки первой команды подпрограммы обработки прерывания (или цикла PTS) проходит некоторое время, которое напрямую зависит от того, в какой фазе выполнения команды был сформирован запрос. Если запрос на прерывание возникает не позднее, чем за четыре машинных цикла до окончания выполнения команды, то его обслуживание начнется по окончании выполнения команды. В случае более позднего появления запроса отклик на него будет получен только по окончании выполнения следующей команды.

Если прерывание обрабатывается контроллером прерываний, то после запуска механизма обслуживания требуется 16 машинных циклов для выбора соответствующего вектора прерывания, загрузки стека и перехода к выборке первой команды. Если используется стек, расположенный во внешней памяти, то добавляется еще несколько машинных циклов, что обусловлено работой внешней шины. В случае, если прерывание поступает на блок PTS, то сразу же запускается цикл PTS.

Таким образом, время реакции на прерывание обозначается в машинных циклах (мц):

- для контроллера прерываний время отклика t_1 вычисляется следующим образом:

4 мц + t_2 (время выполнения команды) + 16 мц;

- для блока PTS время отклика t_1 составит: 4 мц + t_2 (время выполнения команды).

Примечание – Самая быстрая обработка прерывания будет в случае, если стек находится в области внутреннего ОЗУ и запрос поступил во время короткой команды или

простоя ЦПУ. При выполнении операций «выбор вектора» и «запись в стек» первой выполняется операция «выбор вектора».

Программирование прерываний

Любое из прерываний с номерами от INT00 до INT14 может быть направлено для обслуживания как в контроллер прерываний, так и в блок PTS. Задать вариант обслуживания для каждого прерывания можно посредством регистра PTSSEL. Биты регистра с 14 по 0 соответствуют прерываниям с номерами INT14 – INT00 в том же порядке. По умолчанию все биты регистра PTSSEL сброшены, и все запросы на прерывания направляются в контроллер прерываний.

Для глобального разрешения обслуживания прерываний следует установить бит I регистра PSW. Бит устанавливается командой EI. Для запрета обслуживания всех прерываний, т.е. для сброса бита I используется команда DI.

Помимо глобального разрешения обслуживания, каждое прерывание должно быть индивидуально разрешено посредством соответствующего бита. Биты управления прерываниями находятся в регистрах масок INT_MASK и INT_MASK1. Прерываниям с номерами INT15 – INT8 соответствуют биты с седьмого по нулевой регистра INT_MASK1, а прерываниям с номерами INT7 – INT0 соответствуют биты с седьмого по нулевой регистра INT_MASK в том же порядке (см. рисунок 6.4).

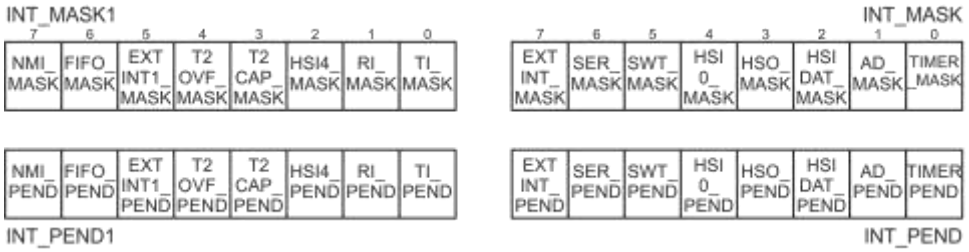


Рисунок 6.4 – Регистры масок прерываний и ждущих прерываний

Дополнительными регистрами обслуживания прерываний являются регистры ждущих прерываний INT_PEND и INT_PEND1. Как только система мониторинга прерываний обнаруживает запрос, устанавливается соответствующий бит регистра, являющийся индикатором. Прерываниям с номерами INT15 – INT08 соответствуют биты с седьмого по нулевой регистра INT_PEND1, а прерываниям с номерами INT07 – INT00 соответствуют биты с седьмого по нулевой регистра INT_PEND в том же порядке.

Индикаторы обнаруженных запросов прерываний устанавливаются всегда, независимо от состояния бита I регистра PSW как для немаскированных, так и для замаскированных прерываний. Регистры INT_PEND и INT_PEND1 всегда доступны для чтения и записи, что позволяет не только следить за возникающими запросами прерываний, но и отменять или формировать их программно, обнуляя или устанавливая соответствующие биты. Аппаратно каждый индикатор запроса на прерывание сбрасывается сразу после запуска механизма обслуживания соответствующего прерывания.

Для некоторых прерываний, помимо глобального и индивидуального разрешения, требуется дополнительное разрешение, которое осуществляется посредством регистра IOC1:

- бит HSI_INT задает источник прерывания для вектора HSIDATAV (INT02);
- биты T2_OVF_INT и T1_OVF_INT – для вектора TOVF (INT00);
- бит EXTINT_SRC – для вектора EXTINT (INT07).

На рисунке 6.5 показаны источники прерываний и соответствующие им векторы, а также управляющие биты. Не показанные на рисунке 6.5 линии прерываний не имеют управляющих бит, т.е. при возникновении запроса на прерывание сразу осуществляется переход по вектору прерывания.

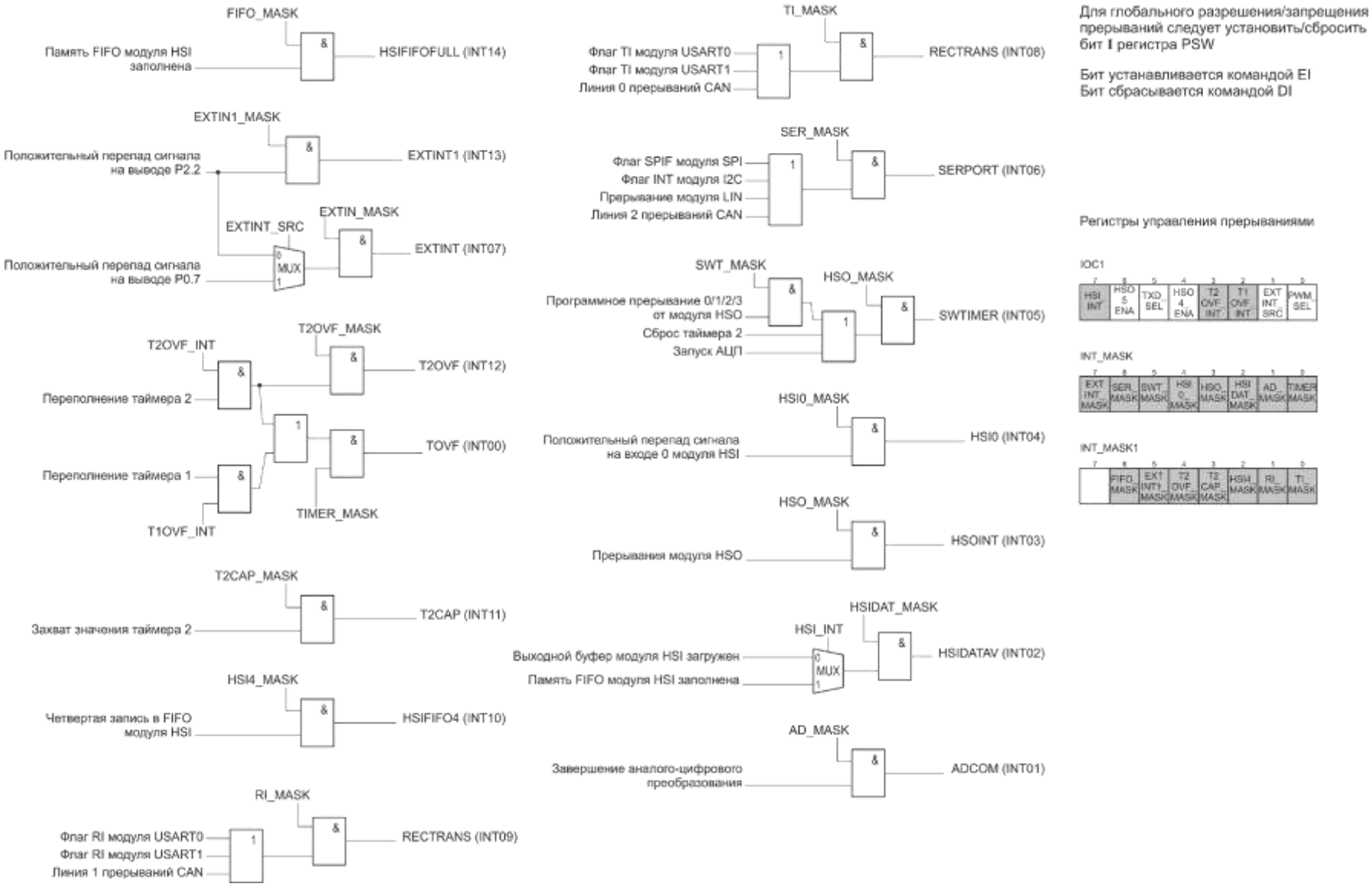


Рисунок 6.5 – Источники прерываний, управляющие биты и векторы прерываний

6.3 Специальные прерывания

Микроконтроллер поддерживает четыре специальных прерывания:
- по неподдерживаемому коду (вектор UNIOPCODE);
- по команде TRAP (программное прерывание с вектором SWTRAP);
- модуля отладки (вектор DEBUG);
- внешнее по выводу NMI микроконтроллера (вектор NMI).

Эти прерывания обслуживаются только контроллером прерываний (не могут быть назначены на PTS), не могут быть замаскированы и на них не влияет состояние бита I регистра PSW. Только прерывания с векторами NMI и DEBUG проходят через детектор передачи и декодировщик приоритетов, остальные два – напрямую в контроллер прерываний.

Неподдерживаемый код. При попытке ЦПУ выполнить неподдерживаемый код, появившийся в результате программной или аппаратной ошибки, произойдет переход по вектору UNIOPCODE с адресом 2012h. Вектор должен содержать стартовый адрес подпрограммы обработки ошибки.

Программное прерывание. Используется при отладке программ и генерации программных прерываний и позволяет вырабатывать прерывания после выполнения каждой команды. Вектор SWTRAP, расположенный по адресу 2010h, должен содержать стартовый адрес подпрограммы обслуживания прерывания.

Прерывание модуля отладки. Вектор прерывания DEBUG расположен по адресу 2060h. Прерывание имеет наивысший приоритет и используется при отладке программ.

Немаскируемое прерывание. Внешнее прерывание. Формируется сигналом, приходящим на вывод NMI микроконтроллера. Имеет 15 уровень приоритета и используется для экстренного прерывания микроконтроллера. Программа обслуживания прерывания вызывается косвенно через вектор NMI с адресом 203Eh.

Если вывод NMI не используется, его необходимо заземлить.

6.4 Блок PTS

Разрешением работы блока PTS управляет бит PSE регистра PSW. Бит устанавливается командой EPTS (разрешение PTS) и сбрасывается командой DPTS (запрещение PTS).

Для того, чтобы запрос на прерывание был направлен в блок PTS, следует установить соответствующий бит в регистре PTSSEL.

Прежде чем направлять запросы прерываний в блок PTS, следует сформировать управляющие блоки. Для каждого прерывания формируется отдельный блок (PTSCB), состоящий из восьми байт, каждый из которых становится управляющим регистром. Задаются регистры: PTSCOUNT, PTSCON, PTSSRC (формируется двумя регистрами PTSSRC_H и PTSSRC_L), PTSDST (формируется двумя регистрами PTSDST_H и PTSDST_L) и PTSBLOCK (см. рисунок 6.6). Блок определяет режим, общее число передач, общее число циклов и адреса источника и приемника передач данных.

Все блоки размещаются в регистровом ОЗУ по четным адресам. Адрес начала блока записывается в ячейку вектора, который соответствует прерыванию. Если в управляющем блоке имеются неиспользуемые байты, то они могут использоваться, как регистры общего назначения ОЗУ.

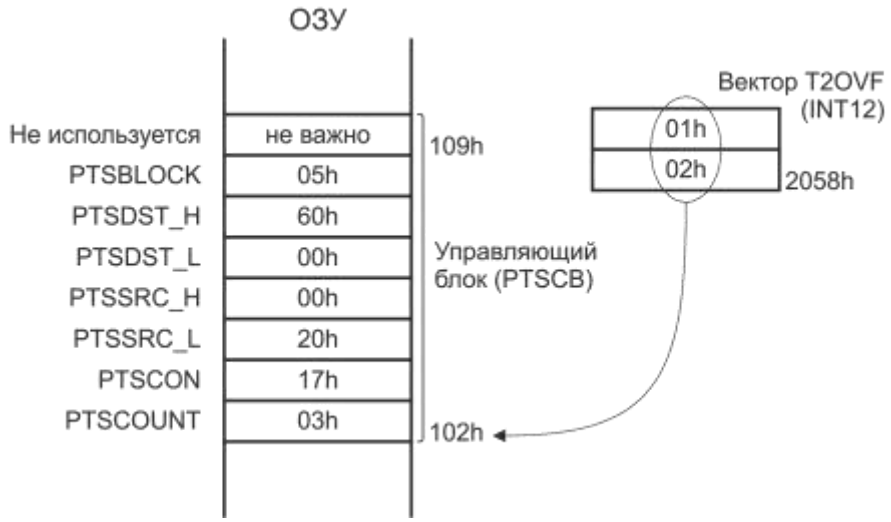


Рисунок 6.6 – Структура управляющего блока (PTSCB) обслуживания прерывания от таймера 2 с вектором T2OVF (INT12)

Блок PTS поддерживает четыре режима функционирования – одиночной передачи, блочной передачи, режимы HSO и HSI. Конфигурация управляющих блоков меняется в зависимости от режима работы. На рисунке 6.7 показаны варианты структур управляющих блоков для каждого режима.

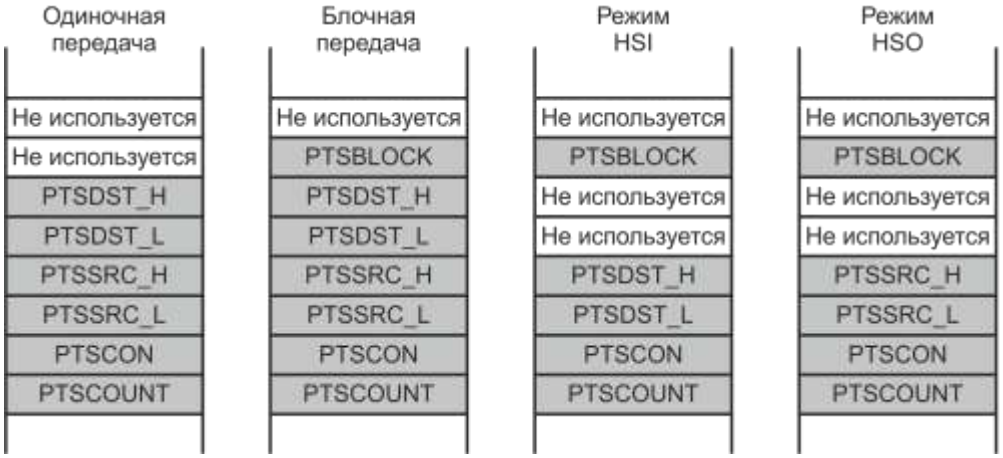


Рисунок 6.7 – Структура управляющего блока в зависимости от режима работы

Регистр PTSCOUNT

В начальном байте каждого управляющего блока всегда располагается счетный регистр PTSCOUNT, который задает число циклов PTS (максимально – 256), которые будут последовательно выполнены без участия центрального процессора. Регистр PTSCOUNT декрементируется в конце каждого цикла PTS и как только содержимое регистра станет равно нулю, соответствующий бит в регистре PTSSEL сбросится, а в регистре PTSSRV установится индикатор запроса на прерывание END-OF-PTS. Прерывание END-OF-PTS является стандартным прерыванием и обрабатывается контроллером прерываний.

Пример. Чтобы направить запрос прерывания от таймера 2 с вектором T2OVF (INT12) на блок PTS, следует установить бит T2OVF_SEL регистра PTSSEL. При возникновении запроса на прерывание аппаратная часть обратится к ячейке памяти с адресом 2058h (см. таблицу 6.1), чтобы получить адрес начала соответствующего управляющего блока (на рисунке 6.5 этот адрес равен 102h).

Когда счетчик циклов обнулится, аппаратная часть сгенерирует прерывание END-OF-PTS и в регистре PTSSRV установится индикатор T2OVF_SRV, при этом бит T2OVF_SEL сбросится. Далее прерывание END-OF-PTS будет направлено на обслуживание в контроллер прерываний с вектором T2OVF, которому соответствует ячейка с адресом 2038h (адрес, используемый контроллером прерываний при стандартной процедуре обслуживания прерывания от таймера 2). Как только контроллер прерываний запустит механизм обслуживания прерывания END-OF-PTS, бит T2OVF_SRV будет сброшен.

Для того, чтобы последующие прерывания от таймера 2 направлялись на PTS, бит T2OVF_SEL следует снова установить.

Регистр PTSSRV фактически является аналогом регистров INT_PEND и INT_PEND1 и позволяет не только следить за возникающими запросами прерываний END-OF-PTS, но и отменять или формировать их программно, обнуляя или устанавливая соответствующие биты. Каждому прерыванию, которое может быть направлено в блок PTS, соответствует один бит регистра PTSSRV. Прерываниям с номерами INT14 – INT00 соответствуют биты регистра с 14 по 0 в том же порядке.

Регистр PTSCON

Основной регистр управляющего блока, предназначенный для задания режима его работы и конфигурации. В режимах одиночной и блочной передачи этот регистр имеет один формат, а в режимах HSO/HSI – другой.

Регистр PTSSRC

Регистр используется в режимах одиночной и блочной передач и в режиме HSO и содержит адрес ячейки памяти источника.

В режиме одиночной передачи биты SI и SU регистра PTSCON определяют, будет ли PTSSRC инкрементироваться.

В режиме блочной передачи бит SI определяет, будет ли PTSSRC инкрементироваться после каждой передачи, а бит SU – будет ли PTSSRC сохранять свое последнее значение или восстанавливать первоначальное.

В режиме HSO бит SU определяет, будет ли PTSSRC обновляться в конце цикла обмена.

Регистр PTSDST

Регистр используется в режимах одиночной и блочной передач и в режиме HSI и содержит адрес ячейки памяти приемника.

В режиме одиночной передачи биты DI и DU регистра PTSCON определяют, будет ли PTSDST инкрементироваться.

В режиме блочной передачи бит DI определяет, будет ли PTSDST инкрементироваться после каждой передачи, а бит DU – будет ли PTSDST сохранять свое последнее значение или восстанавливать первоначальное.

В режиме HSO бит DU определяет, будет ли PTSDST обновляться в конце цикла обмена.

Регистр PTSBLOCK

Регистр используется во время блочной передачи и в режимах HSI и HSO и управляет числом передач.

Режим одиночной передачи

В режиме одиночной передачи в течение цикла PTS передается один байт (если установлен бит BW регистра PTSCON) или одно слово (если бит BW сброшен) из одной ячейки памяти в другую. Этот режим обычно используется с прерываниями по последовательному порту ввода-вывода или концу аналого-цифрового преобразования. Количество передач задается регистром PTSCOUNT. Адреса ячеек памяти источника и приемника байта/слова задаются регистрами PTSSRC и PTSDST, которые могут указывать на любую ячейку памяти (в случае передачи слова адрес ячейки должен быть четным). Пары бит SU-SI и DU-DI управляют состояниями регистров PTSSRC и PTSDST

(могут инкрементироваться независимо друг от друга). Оба бита каждой пары должны иметь одинаковые значения. Запись 0b-1b или 1b-0b запрещена.

Пример режима одиночной передачи.

На рисунке 6.8 показано состояние управляющего блока в начале обслуживания прерывания и в конце. Счетный регистр задает восемь циклов PTS. Каждый цикл пересылает одно слово (бит BW очищен) из ячейки с адресом 20h в ячейку внешней памяти. После первой передачи слова в ячейку с адресом 6000h значение регистра PTSDST инкрементируется на 2 (установлен бит DI), значение счетного регистра уменьшается на единицу. Состояние регистра PTSSRC не изменяется, поскольку биты SU и SI не установлены. Далее цикл повторяется восемь раз.

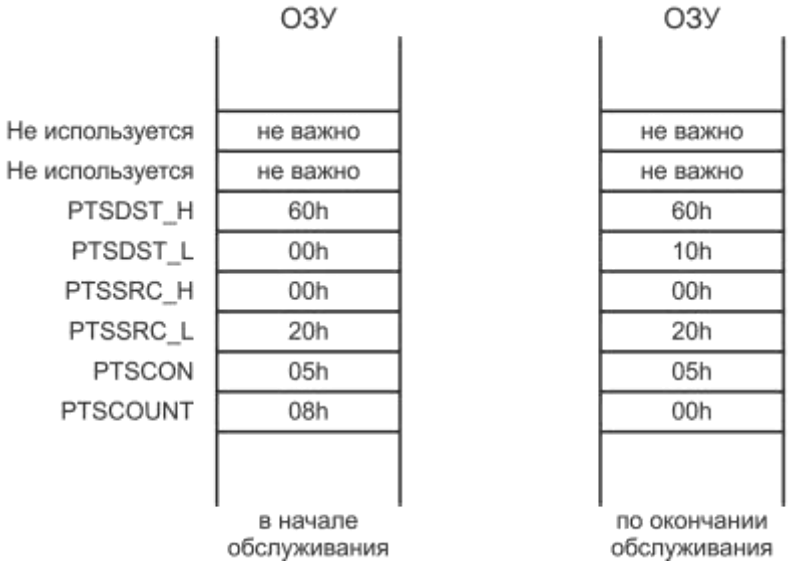


Рисунок 6.8 – Управляющий блок в режиме одиночной передачи

По окончании восьми передач будет заполнена область внешней памяти с адреса 6000h по 600Fh. Счетный регистр обнулится, и сформируется прерывание END-OF-PTS. Состояние регистра PTSDST не сбросится в начальное, поскольку установлен бит DU.

Режим блочной передачи

В режиме блочной передачи в течение цикла PTS может быть передано от одного до 32 байт или слов (в зависимости от состояния бита BW) из одной области памяти в другую. Количество передаваемых байт/слов задается регистром количества блоков PTSBLOCK. Количество циклов PTS задается регистром PTSCOUNT. Адреса ячеек памяти источника и приемника байт/слов задаются регистрами PTSSRC и PTSDST, которые могут указывать на любую ячейку памяти (в случае передачи слова адрес ячейки должен быть четным). В отличие от режима одиночной передачи, биты SU, SI, DU и DI могут устанавливаться независимо друг от друга. Установка бита SI/DI включает автоинкремент регистра PTSSRC/PTSDST, т.е. после каждой передачи байта/слова значение регистра будет увеличиваться на один/два. Установка бита SU/DU задает режим работы регистра PTSSRC/PTSDST, когда этот регистр сохраняет значение адреса, которое было в нем на момент завершения цикла PTS. Таким образом, одновременная установка битов SU-SI/DU-DI задает режим последовательного увеличения содержимого регистра PTSSRC/PTSDST.

Пример режима блочной передачи.

На рисунке 6.9 показано состояние управляющего блока в начале обслуживания прерывания и в конце. Счетный регистр задает три цикла PTS. В каждом цикле должно быть передано пять байт, что задается регистром PTSBLOCK (значение 05h) и битом BW (установлен).

Первый байт передается из ячейки с адресом 20h в ячейку с адресом 6000h, после чего регистры PTSSRC и PTSDST инкрементируются (биты SI и DI установлены). Далее передаются остальные четыре байта. Таким образом, заполняются ячейки с адресами 6000h – 6004h. На этом заканчивается первый цикл PTS. Регистр PTSCOUNT декрементируется. Поскольку бит DU установлен, то в регистре PTSDST сохраняется значение 6005h, которое было в нем на момент окончания первого цикла PTS. В тоже время регистр PTSSRC восстанавливает свое исходное значение 0020h, поскольку бит SU не установлен. Далее цикл PTS повторяется.

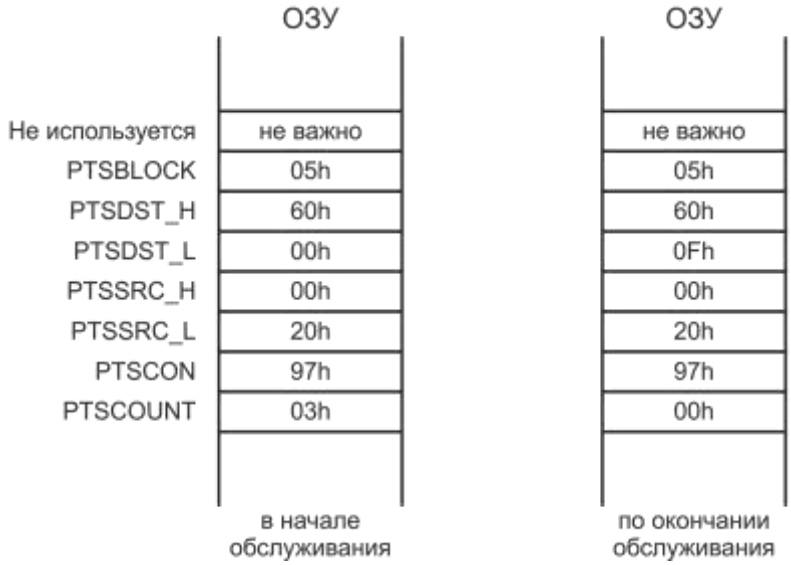


Рисунок 6.9 – Управляющий блок в режиме блочной передачи

Таким образом, по окончании трех циклов PTS значения из области памяти с адресами 20h – 24h будут последовательно скопированы в области с адресами 6000h – 6004h, 6005h – 6009h, 600Ah – 600Eh.

Режим HSI

В режиме HSI блок PTS загружает содержимое памяти FIFO модуля HSI во внутреннюю, либо внешнюю память, формируя так называемую «Таблицу». Адрес начала «Таблицы» находится в регистре PTSDST. Любое прерывание модуля HSI может быть использовано для запуска цикла PTS. Значение, записанное в регистр PTSBLOCK, указывает, сколько слов из памяти FIFO будет передано в «Таблицу» в течение каждого цикла PTS (максимально семь). Необходимо ввести значение, соответствующее прерыванию, которое запускает цикл PTS. Например, четвертая запись в память FIFO может явиться источником прерывания модуля HSI с вектором HSIFIFO4 (INT10). Если это прерывание использовалось для запуска цикла PTS, то в регистр PTSBLOCK следует записать значение 04h, таким образом, четыре значения из памяти FIFO будут переписаны в «Таблицу».

Данные памяти FIFO доступны посредством последовательного чтения регистров HSI_STATUS и HSI_TIME. Регистр HSI_STATUS содержит биты событий и текущее состояние выводов HSI. Регистр HSI_TIME содержит время обнаружения события.

С каждой передачей значения регистров HSI_STATUS и HSI_TIME последовательно копируются в «Таблицу» (см. рисунок 6.10, где «xxx» – адрес начала «Таблицы»). Для того, чтобы регистр PTSDST сохранял значение адреса, в конце каждого цикла PTS должен быть установлен бит UPDT регистра PTSCON (см. конфигурацию регистра для режима HSI/HSO).

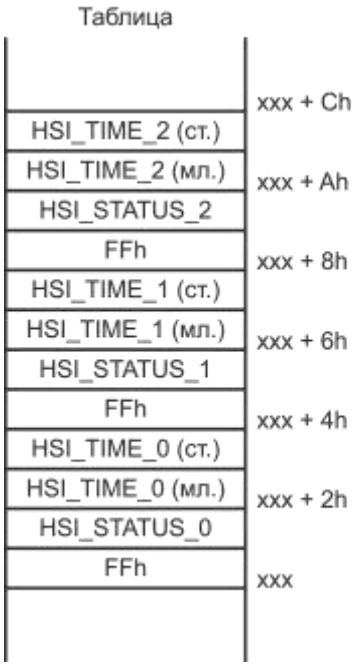


Рисунок 6.10 – Порядок заполнения «Таблицы» для режимов HSI и HSO

На рисунке 6.11 показано состояние управляющего блока в начале обслуживания прерывания и в конце. Задано 10 циклов PTS, в каждом из которых будет передано семь слов данных (значения регистров HSI_STATUS и HSI_TIME) из памяти FIFO модуля HSI в «Таблицу» с начальным адресом 100h. Начальный адрес инкрементируется в конце каждой передачи и обновляется по окончании каждого цикла PTS.

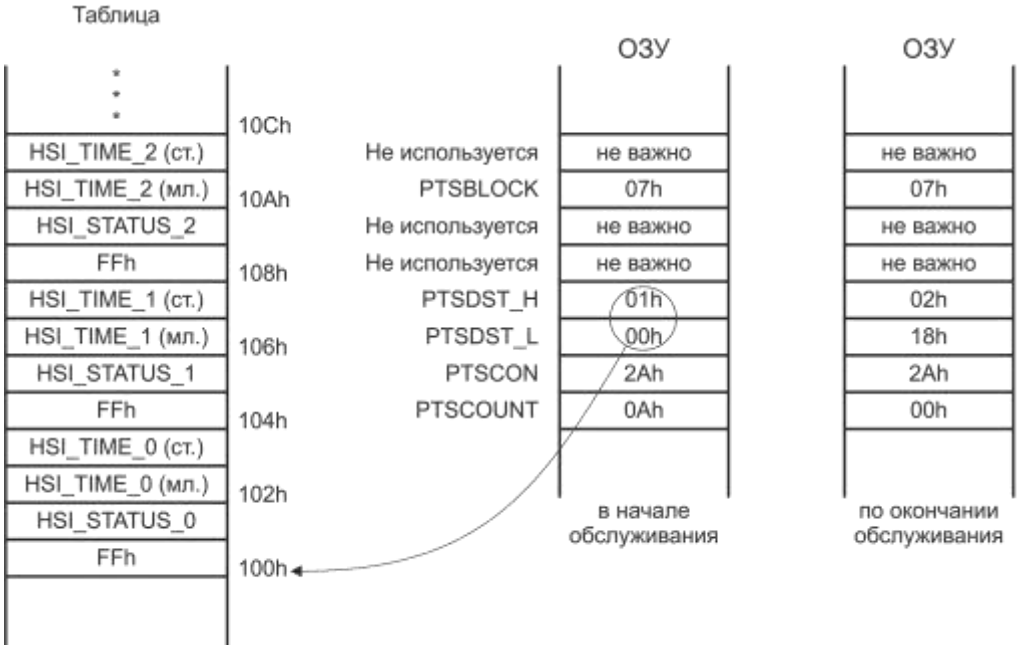


Рисунок 6.11 – Управляющий блок обслуживания прерывания модуля HSI

Режим HSO

В режиме HSO блок PTS загружает содержимое «Таблицы», расположенной во внутренней или внешней памяти, в ассоциативное запоминающее устройство (АЗУ) модуля HSO, т.е. выполняет действия, обратные тем, что выполняются в режиме HSI. Адрес начала «Таблицы» находится в регистре PTSSRC.

Все восемь слов АЗУ являются 24-разрядными. 16 бит каждого слова задают время выполнения команды, а оставшиеся 8 бит – код команды и ее параметры. Эти данные следует разместить в «Таблице» в последовательности, указанной на рисунке 6.10.

В одном цикле PTS от одного до восьми слов данных загружается последовательно через регистры HSI_COMMAND и HSI_TIME в АЗУ модуля HSO. Если в следующем цикле загружается девятое слово, а АЗУ заполнено, то это слово помещается в предварительный буфер, где находится до тех пор, пока не освободится место в АЗУ.

Любое прерывание модуля HSO может быть использовано для запуска цикла PTS. Значение, записанное в регистр PTSBLOCK, указывает, сколько записей будет скопировано из «Таблицы» в модуль HSO в течение каждого цикла PTS (максимально восемь). Для того, чтобы регистр PTSSRC сохранял значение адреса в конце каждого цикла PTS, бит UPDT регистра PTSCON должен быть установлен.

Пример режима HSO.

На рисунке 6.12 показано состояние управляющего блока в начале обслуживания прерывания и в конце. Задано 10 циклов PTS, в каждом из которых будет передано восемь слов данных из «Таблицы» с начальным адресом 100h в модуль HSO. Начальный адрес инкрементируется в конце каждой передачи и обновляется по окончании каждого цикла PTS.

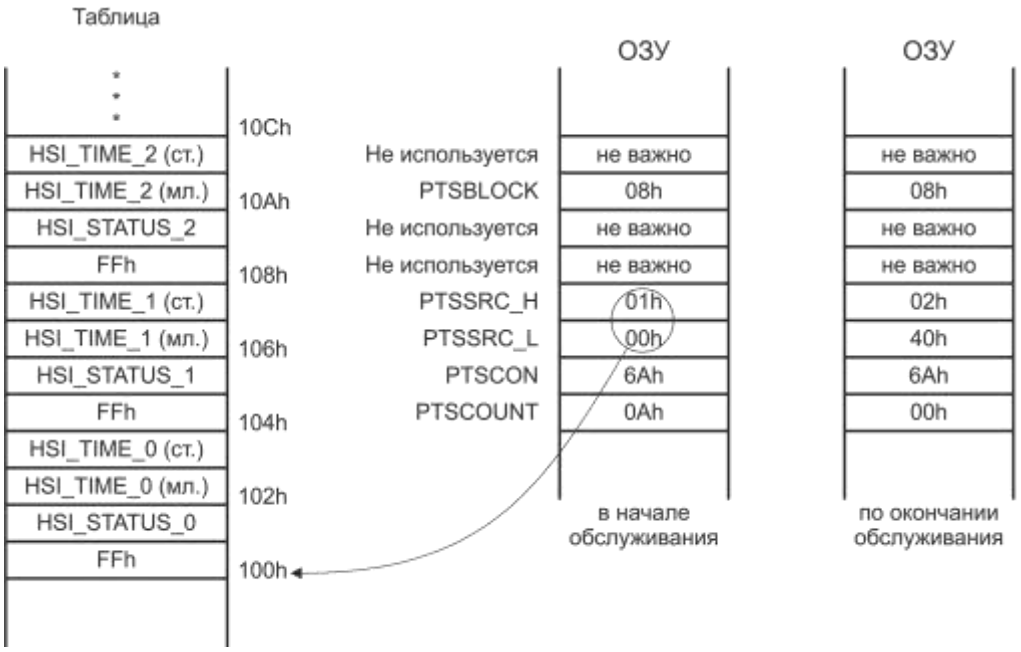


Рисунок 6.12 – Управляющий блок обслуживания прерывания модуля HSO

7 Сторожевой таймер

Сторожевой таймер позволяет восстанавливать нормальную работу при сбоях программ. Если работа сторожевого таймера разрешена, он будет вызывать аппаратный сброс, если программа не очищает его до истечения времени выполнения 64К машинных циклов.

Когда таймер переполняется, он переводит линию RESET в состояние низкого уровня не менее, чем на два машинных цикла, сбрасывая микроконтроллер и другие устройства, подключенные к его выводу RESET#.

Для очистки и включения сторожевого таймера следует последовательно друг за другом записать значения 1Eh и E1h в регистр WATCHDOG. Регистр счетчика таймера является 16-разрядным, но для пользователя доступен только его старший байт, получить который можно посредством чтения регистра WATCHDOG.

Сразу после записи значений 1Eh и E1h в регистр WATCHDOG, в счетчик сторожевого таймера будет аппаратно загружено значение 0000h, которое далее будет инкрементироваться каждый машинный цикл. Для программного сброса сторожевого таймера следует своевременно записывать значения 1Eh и E1h в регистр WATCHDOG.

После инициализации сторожевой таймер может быть выключен только аппаратно, после сброса микроконтроллера.

8 Порты ввода-вывода

Порты ввода-вывода служат для передачи информации между микроконтроллером и окружающими его устройствами. Посредством портов конфигурируется микроконтроллер, формируются управляющие сигналы для внешних устройств и считывается информация о состоянии окружения микроконтроллера.

8.1 Функциональные возможности

Микроконтроллер имеет пять 8-битных портов ввода-вывода. Выводы портов выполняют функции ввода и/или вывода информации (входы и выходы). Имеются квазидвунаправленные выводы и двунаправленные с открытым стоком.

Все выводы портов имеют дополнительные (альтернативные) функции.

Вывод вход

Любой вывод порта, определенный как вход, может быть только считан. Любая операция записи недопустима или игнорируется. Функциональная схема входа показана на рисунке 8.1.

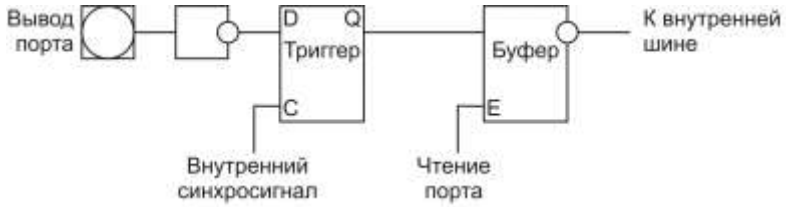


Рисунок 8.1 – Функциональная схема входа

Входной сигнал с вывода порта инвертируется и запоминается в триггере, который синхронизируется внутренним сигналом тактирования микроконтроллера. По сигналу чтения порта (пока сигнал CLOCKOUT имеет низкий уровень) содержимое триггера заносится в буфер и вместе с этим выставляется на внутреннюю шину.

Следует помнить, что частота изменения сигнала на выводе должна быть как минимум в два раза меньше рабочей частоты микроконтроллера.

Входные схемы не имеют выходных драйверов, ток утечки на входе и емкостная нагрузка очень малы.

Вывод выход

Любой вывод порта, определенный как выход, может быть только записан. Выходная схема не имеет буфера для чтения, поэтому при считывании будет получено неопределенное значение. Функциональная схема выхода с защелкой и драйверами показана на рисунке 8.2.

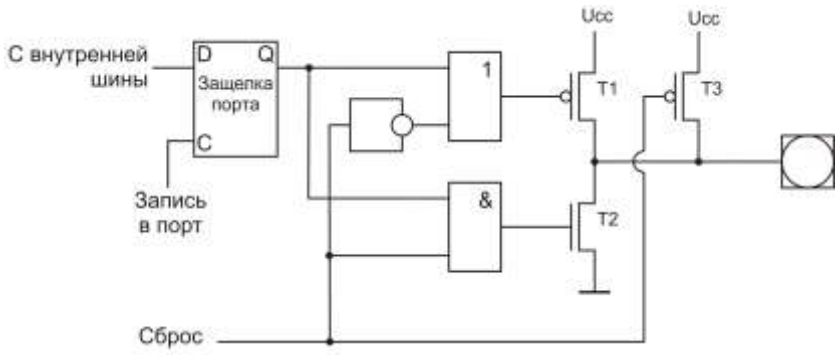
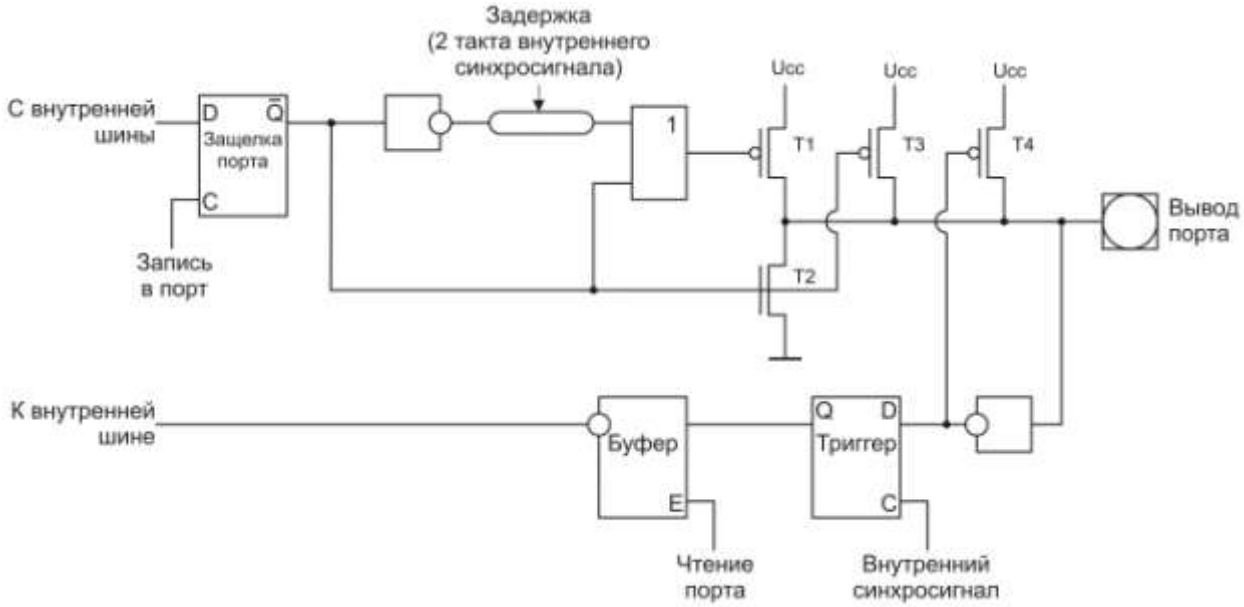


Рисунок 8.2 – Функциональная схема выхода

По сигналу записи в порт (пока сигнал CLOCKOUT имеет низкий уровень) новое значение запоминается в защелке порта и выставляется на выводе. Состояние вывода будет сохраняться до тех пор, пока не будет записано новое значение в защелку порта или не произойдет сброс микроконтроллера.

Квазидвунаправленный вывод

Квазидвунаправленный вывод может одновременно функционировать как в режиме входа, так и в режиме выхода, при этом управлять переключением не нужно. Такой вывод имеет активный низкий уровень и пассивный высокий уровень сигнала. При высоком уровне сигнала на выводе последний может функционировать как вход. В случае удержания на выводе низкого уровня сигнала внешнее устройство не может перевести вывод в состояние логической единицы. Функциональная схема квазидвунаправленного вывода показана на рисунке 8.3.



Принятые условные обозначения:
T1, T2 – сильные драйверы;
T3 – очень слабый драйвер;
T4 – слабый драйвер.

Рисунок 8.3 – Функциональная схема квазидвунаправленного вывода

Схема квазидвунаправленного вывода состоит из двух схем – схемы ввода сигналов и схемы вывода. Для приема сигналов с вывода используются триггер и буфер, для вывода – защелка порта и специальная логическая схема драйверов.

Запись единицы в защелку порта закрывает сильный драйвер T2 и открывает очень слабый драйвер T3. Для быстрого переключения вывода порта в единицу сильный драйвер T1 открывается на один такт внутреннего сигнала синхронизации микроконтроллера, а затем закрывается, оставляя открытым только T3. До тех пор, пока нагрузка на выводе мала, драйвер T3 и дополнительный драйвер T4 удерживают на выводе логическую единицу.

Для уменьшения суммарного тока, протекающего через вывод при подаче внешним устройством логического нуля, слабый драйвер T4 закрывается (при достижении уровня примерно 2 В).

На рисунке 8.4 приведена переходная характеристика вывода при открытых драйверах T3 и T4. По мере уменьшения напряжения U_I уменьшается ток I_I. Как только

величина тока достигнет порогового значения I_{TL}, драйвер Т4 закроется и включенным останется только очень слабый драйвер Т3.

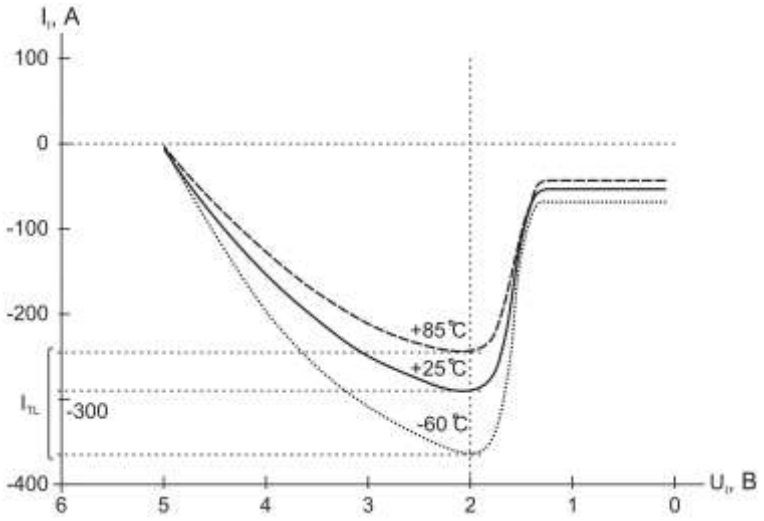


Рисунок 8.4 – Передаточная характеристика квазидвунаправленного выхода

В случае, если вывод не подключен, драйвер Т3 является подтяжкой и поддерживает на выводе маленький ток, который формирует логическую единицу.

Временные характеристики квазидвунаправленного вывода аналогичны характеристикам выводов-выходов и выводов-входов, т.е есть запись/чтение происходят, пока сигнал CLOCKOUT имеет низкий уровень.

Двунаправленный вывод с открытым стоком

Работа двунаправленного вывода с открытым стоком аналогична работе квазидвунаправленного вывода, за исключением того, что отсутствуют сильные и слабые драйверы. Функциональная схема вывода показана на рисунке 8.5.

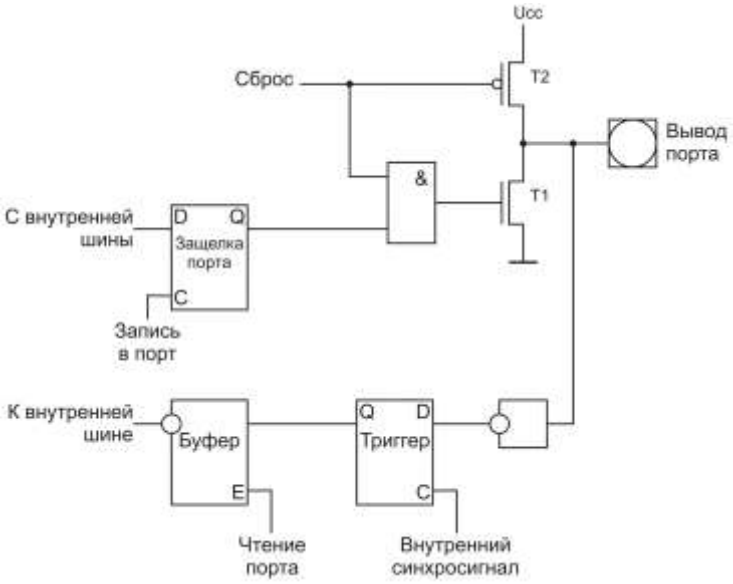


Рисунок 8.5 – Функциональная схема двунаправленного вывода с открытым стоком

Запись нуля в защелку порта открывает сильный драйвер Т1, и на выводе устанавливается логический ноль. В этом случае внешнее устройство не может перевести вывод в состояние логической единицы. Состояние вывода будет сохраняться до тех пор,

пока не будет записано новое значение в защелку порта или не произойдет сброс микроконтроллера.

Запись единицы в защелку порта закроет драйвер T1, и вывод окажется в неподключенном состоянии. Снаружи уровень сигнала на выводе может быть подтянут до уровня логической единицы использованием подтягивающего резистора.

Порт 0

Выводы порта 0 могут функционировать только как входы. Каждый вывод одновременно выполняет функции как цифрового, так и аналогового входа (порт 0 выполняет функции цифрового порта, также и при включенном АЦП). Состояние порта может быть прочитано посредством регистра IOPORT0, который всегда доступен только для чтения.

Следует помнить, что чтение регистра IOPORT0 во время аналого-цифрового преобразования возвращает неопределенное значение.

Седьмой разряд порта 0 дополнительно является входом внешнего прерывания EXTINT.

Порт 1

Все выводы порта 1 квазидвунаправленные и по умолчанию функционируют, как входы-выходы общего назначения. Получать информацию о состоянии выводов порта, а также управлять их состоянием можно посредством регистра IOPORT1 (всегда доступен для записи/чтения), каждый бит которого соответствует одному выводу. Для того, чтобы вывод мог функционировать как вход, в соответствующем бите регистра IOPORT1 должна находиться единица. Для конфигурирования порта можно воспользоваться командой LDB. Например, установить на младших разрядах единицы, а на старших – нули:

LDB IOPORT1, #00001111b.

Когда на состояние выводов оказывает влияние внешнее устройство, то следует внимательнее относиться к операциям чтения, изменения и записи регистра IOPORT1 (т.е. считыванию состояния порта и его изменению). Если какой-либо вывод удерживается внешним устройством в состоянии логического нуля, то невозможно установить на этом выводе высокий уровень записью единицы в соответствующий бит регистра IOPORT1.

Низкий уровень сигнала на выводе может быть установлен записью нуля в соответствующий бит регистра IOPORT1. Поскольку в этом случае низкий уровень сигнала поддерживается сильным драйвером (см. рисунок 8.3), то подача высокого уровня от внешнего устройства приведет к протеканию большого тока короткого замыкания, что может негативно сказаться на работе как микроконтроллера, так и внешнего устройства.

Вывод 0 порта одновременно является входом-выходом общего назначения и входом SPW_DIN модуля SpaceWire. При использовании модуля SpaceWire нулевой бит регистра IOPORT1 должен быть установлен.

Вывод 1 порта одновременно является входом-выходом общего назначения и входом SPW_SIN модуля SpaceWire. При использовании модуля SpaceWire первый бит регистра IOPORT1 должен быть установлен.

Вывод 2 порта одновременно является входом-выходом общего назначения и входом выбора ведомого SS# модуля SPI и мультиплицирован с выходом HSO.0 модуля HSO. При использовании модуля SPI в качестве ведомого устройства второй бит регистра IOPORT1 должен быть установлен, чтобы позволить внешнему мастеру активировать и дезактивировать ведомого. Если требуется постоянное активное состояние ведомого, можно записать ноль в указанный бит (не рекомендуется). При использовании модуля SPI в качестве мастера состояние второго бита регистра IOPORT1 не оказывает влияния на работу модуля. Для включения альтернативной функции следует установить бит HSO.0/1_ENA регистра IOC1.

Вывод 3 порта является входом-выходом общего и мультиплицирован с выходом SPW_DOUT модуля SpaceWire. Для включения альтернативной функции следует установить бит SPWD_SEL регистра IOC3.

Вывод 4 порта является входом-выходом общего и мультиплицирован с выходом SPW_SOUT модуля SpaceWire. Для включения альтернативной функции следует установить бит SPWS_SEL регистра IOC3.

Вывод 5 порта одновременно является входом-выходом общего назначения и входом-выходом данных MOSI модуля SPI и мультиплицирован с выходом HSO.1 модуля HSO. При использовании модуля SPI пятый бит регистра IOPORT1 должен быть установлен. Для включения альтернативной функции следует установить бит HSO.0/1_ENA регистра IOC1.

Вывод 6 порта одновременно является входом-выходом общего назначения и входом-выходом данных MISO модуля SPI и мультиплицирован с выходом HSO.2 модуля HSO. При использовании модуля SPI шестой бит регистра IOPORT1 должен быть установлен. Для включения альтернативной функции следует установить бит HSO.2/4_ENA регистра IOC1.

Вывод 7 порта одновременно является входом-выходом общего назначения и входом-выходом синхросигнала SCK модуля SPI и мультиплицирован с выходом HSO.4 модуля HSO. При использовании модуля SPI седьмой бит регистра IOPORT1 должен быть установлен. Для включения альтернативной функции следует установить бит HSO.2/4_ENA регистра IOC1.

Порт 2

Порт имеет один вывод, работающий как вход, два – как выход, и пять выводов, которые являются входами/выходами. Получать информацию о состоянии выводов порта, а также управлять их состоянием можно посредством регистра IOPORT2, но следует помнить, что вывод-вход не может быть запрограммирован, а состояние вывода-выхода не может быть прочитано.

В режиме программирования микроконтроллера выводы порта реализуют функции контроля взаимодействия с внешней памятью.

Вывод 0 порта мультиплицирован с выходом TXD0 модуля USART0. По умолчанию вывод функционирует как выход общего назначения. Для включения альтернативной функции следует установить бит TXD_SEL регистра IOC1.

Вывод 1 порта является входом общего назначения, а также выводом RXD0 модуля USART0 (при приеме данных функционирует как обычный вход, при передаче – как выход с открытым стоком).

Вывод 2 порта одновременно является входом общего назначения и входом внешнего прерывания EXTINT.

Вывод 3 порта одновременно является входом общего назначения и входом сигнала синхронизации для таймера 2, а также выводом SCL модуля I2C (при приеме синхросигнала функционирует как обычный вход, при передаче – как выход с открытым стоком).

Вывод 4 порта одновременно является входом общего назначения, входом сигнала сброса для таймера 2, а также функционирует как вывод RXD1 модуля USART1 (обычный вход-выход).

Вывод 5 порта является выходом общего назначения и мультиплицирован с выходом нулевого канала модуля ШИМ. Для включения альтернативной функции следует установить бит PWM_SEL регистра IOC1.

Вывод 6 порта – квазидвунаправленный вывод. Одновременно является входом-выходом общего назначения и входом сигнала задания направления счета для таймера 2. Вывод мультиплицирован с выходом TXD1 модуля USART1. Для включения альтернативной функции следует установить бит TXD_SEL регистра IOC1.

Вывод 7 порта – квазидвунаправленный вывод. Одновременно является входом

-выходом общего назначения, входом сигнала захвата состояния таймера 2, а также выводом SDA модуля I2C (при приеме синхросигнала функционирует как обычный вход, при передаче – как выход с открытым стоком).

Порты 3 и 4

Получать информацию о состоянии выводов портов 3 и 4, а также управлять их состоянием можно посредством регистра IOPORT34 (запись и чтение следует производить пословно).

Чтобы использовать порты 3 и 4 как порты входы/выходы, микроконтроллер должен быть сконфигурирован для работы с внутренним ПЗУ.

Каждый вывод портов 3 и 4 может быть двунаправленным выводом с открытым стоком, либо разрядом двунаправленной системной шины адрес/данные. Обе функции могут быть скомбинированы. За конфигурацию отвечает вывод EA# микроконтроллера, который также определяет, доступны ли порты.

Если EA# имеет низкий уровень во время системного сброса, то порты 3 и 4 используются только для поддержки системной шины адрес/данные. Чтение или запись портов по адресу 1FFEh интерпретируется как доступ к внешней шине, а не к портам.

Когда EA# имеет высокий уровень во время сброса, порты 3 и 4 используются как входы/выходы с открытым стоком и дополнительно подерживают системную шину адресов/данных. Ноль, записанный в защелку порта, открывает транзистор с мощным низким выходным уровнем, тогда как единица закрывает выходной транзистор и переводит вывод в плавающее состояние. Запись единицы на выводы портов разрешает использовать их как входы. Чтение или запись в порты интерпретируется как внутренний доступ, если не выполняется цикл внешней шины. Порты 3 и 4 остаются портами ввода/вывода с открытым стоком все время, пока выполняется цикл внешней шины.

Любой доступ к адресу, не интерпретируемому как внутренний адрес, вызывает выполнение цикла доступа к внешней шине. Чтобы выполнить цикл шины, порты 3 и 4 временно переключаются на функцию поддержки системной шины адрес/данные. Это значит, что выводы портов имеют мощные высокие и низкие уровни во время цикла шины записи адреса и данных и имеют плавающие состояния во время цикла шины считывания данных. После завершения цикла выводы портов переключаются обратно на функцию портов I/O. Состояние вывода порта после переключения зависит от последнего значения, записанного в защелку порта.

В проектах, где используется системная шина адрес/данные, значение, запрограммированное в защелке порта, имеет несколько назначений. Значение защелки порта сохраняется на шине во время простоя (когда порты 3 и 4 не являются системной шиной). Запись нуля в порты 3 и 4 означает, что на шине во время простоя сохраняется низкий уровень, тогда как запись единицы означает, что шина находится в плавающем состоянии во время простоя. Плавающая шина может привести к увеличению потребления тока, но его легко ограничить подключением выводов шины через резисторы к напряжению питания. Использование шины с низким уровнем снимает необходимость в резисторах, но необходимо быть уверенным, что никакое другое устройство не сможет подключиться к шине в то же самое время (возможно короткое замыкание).

Когда порты 3 и 4 используются исключительно как системная шина адреса/данные, имеющая нагрузочные резисторы, то на шине устанавливается значение 0FFFFh, если команда считывается из несуществующей ячейки памяти. Выборка операнда 0FFh приводит к сбросу микроконтроллера.

Выборка команды из несуществующей ячейки памяти говорит об аппаратной или программной ошибке, поэтому сброс микроконтроллера полезен и предотвращает дальнейшую некорректную работу устройства.

Программирование портов 3 и 4

LD и ST команды требуют использования внутренних вспомогательных регистров, это необходимо для переноса информации порта в нижний регистровый файл перед использованием данных. Если данные уже во внутренней памяти, то в команде LD нет необходимости. Пример записи значения слова в порты 3 и 4:

```
LD      intreg, portdata        ; регистр← данные
ST      intreg, 1FFEh           ; регистр → порт 3 и 4
```

Для конфигурирования портов 3 и 4 сначала записью в регистры портов была установлена конфигурация портов на ввод. Это переводит порты в высокоимпедансное состояние. Порты при сбросе находятся в режиме входов. Если в порт были записаны нули, то на выводы, которые должны быть входами, необходимо записать единицы. Чтение портов 3 и 4, если в них предварительно были записаны нули, идет в следующем порядке:

```
LD      intrega, #0FFFFh        ; установка порта изменяет режим схемы
ST      intrega, 1FFEh          ; регистр → порты 3 и 4
                                ; ST и LD не
                                ; нужны, если предварительно были
                                ; записаны единицы
LD      intregb, 1FFEh          ; регистр ← порты 3 и 4
```

Форматы команд LD и ST аналогичны, только изменяется направление передачи и приема.

Аппаратное подключение к квазидвунаправленным портам

Когда выводы квазидвунаправленных портов используются как входы, то уровень сигнала на них задается внешним устройством. В тоже время состояние битов регистра порта также оказывает влияние на соответствующие выводы. Если внешнее устройство удерживает на выводе высокий уровень сигнала, а в соответствующий этому выводу бит регистра порта записывается ноль, то возникает высокий ток короткого замыкания.

Решение этой проблемы может быть как программным, так и аппаратным. Программно достаточно не записывать нули в соответствующие биты регистра порта. Аппаратное решение заключается в том, чтобы последовательно с выводом соединить резистор в 1 кОм.

Программирование портов

После сброса микроконтроллера каждый вывод порта переходит в режим, который назначен по умолчанию (в таблице 2.1 в колонке «Обозначение вывода» основная функция указана левее, а альтернативная – правее, соответственно, в колонке «Функциональное назначение» основная функция прописана в первой строке). Изменить режим работы можно, если включить альтернативную функцию. Так, например, вывод 1 порта 2 по умолчанию является входом, но при альтернативном функционировании в качестве вывода RXD0 модуля USART0 становится двунаправленным выводом с открытым стоком.

Большинство выводов микроконтроллера выполняет основную и альтернативную функцию одновременно. Например, выводы 5, 6 и 7 порта 1, которые являются выводами общего назначения и выводами модуля SPI, или выводы портов 0 и 5, которые являются входами общего назначения и входами АЦП.

Все выводы портов микроконтроллера программируются посредством регистров портов и дополнительно задействованными регистрами IOC1 и IOC3.

9 Управление синхронизацией периферийных устройств

Микроконтроллер имеет функцию управления синхронизацией внутренних периферийных устройств. Если пользователь не использует какое-либо периферийное устройство, то его можно отключить, тем самым уменьшив общее потребление. Включение/выключение происходит записью соответствующих битов в регистр управления CLKC. Также в регистре CLKC находится бит управления режимом SLOW, в котором увеличивается длительность машинного цикла в два раза по сравнению с нормальным режимом работы микроконтроллера. Структура схемы синхронизации представлена на рисунке 9.1.

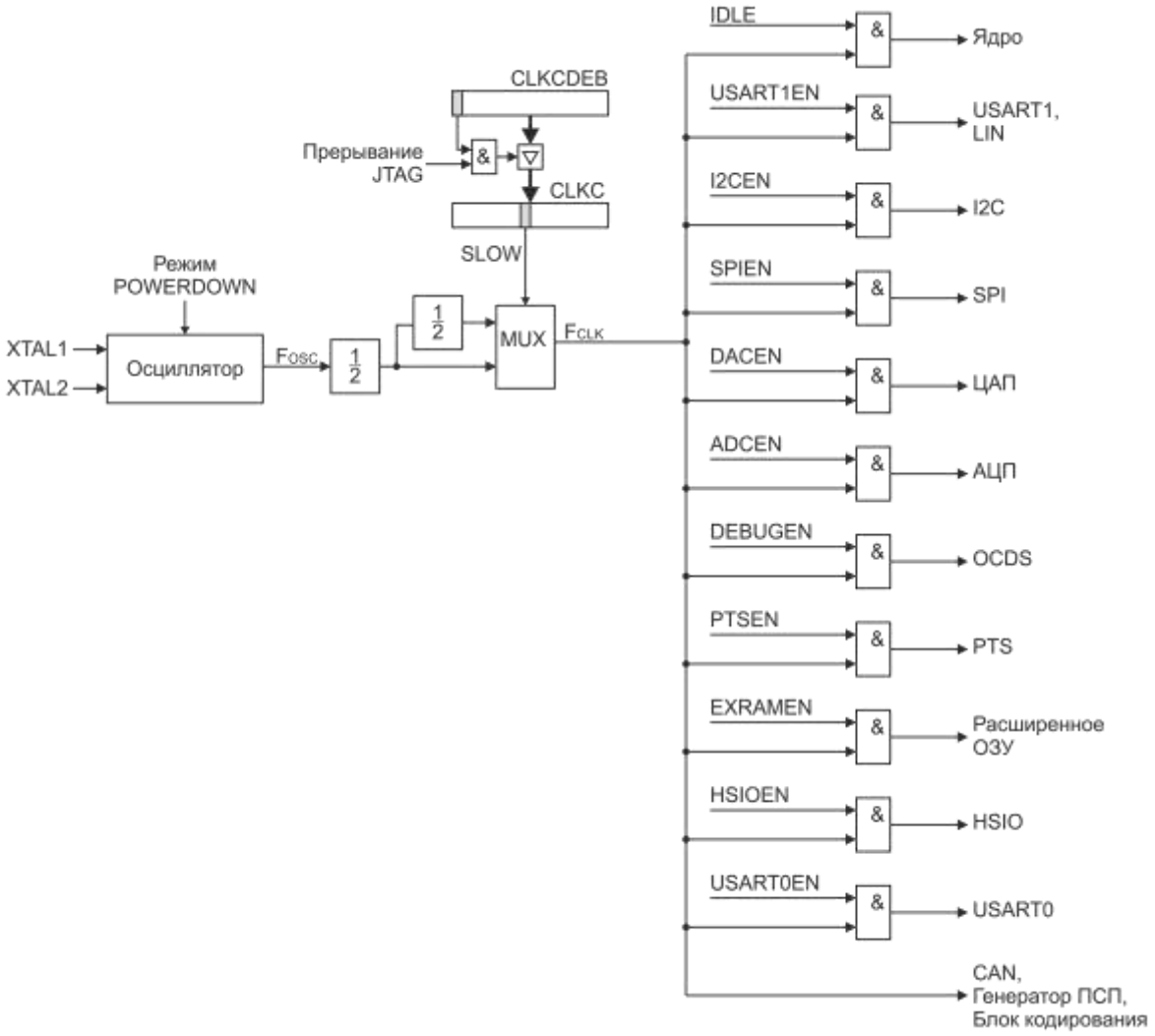


Рисунок 9.1 – Структура схемы синхронизации

При выключении устройств запись в их регистры невозможна, а при чтении будет выдаваться последнее значение, принятое регистрами перед отключением.

При отключении тактовых сигналов с блоков АЦП и ЦАП эти блоки автоматически переводятся в режим SLEEP, что позволяет существенно уменьшить энергопотребление схемы. Однако, следует помнить, что выход из этих режимов – длительный процесс и определяется временем срабатывания аналоговых частей соответствующих устройств.

Не рекомендуется выключение периферийных устройств во время их работы, так как это может привести к непредсказуемым последствиям. Выводы схемы, связанные с

отключаемыми устройствами, останутся в состоянии, которое было на момент отключения.

Модуль отладки может генерировать событие, при котором происходит перезапись регистра CLKC. Этот режим предназначен для остановки необходимых периферийных устройств с целью сохранения в связанных с устройством регистрах результатов.

В целях уменьшения энергопотребления после сброса цифро-аналоговый преобразователь находится в режиме пониженного потребления. Для включения необходимо установить бит DACEN в регистре CLKC . При выключении ЦАП возможно как чтение, так и запись связанного с ним управляющего регистра.

10 Модуль отладки (OCDS)

Модуль предназначен для упрощения процесса отладки программного обеспечения пользователя, а также для контроля хода выполнения программы. Блок формирует несколько типов откликов по нескольким типам событий. События могут быть следующие:

- обращение к адресам внешней памяти;
- появление на шине адресов заданного адреса операнда;
- появление на шине данных заданного слова;
- появление на шине данных заданного младшего байта;
- появление на шине данных заданного старшего байта;
- превышение значения основного счетчика команд заданной величины;
- точное совпадение значения основного счетчика команд с заданной величиной.

При обнаружении того или иного из указанных событий может формироваться отклик:

- прерывание DEBUG (2060h);
- переход в режим IDLE;
- аппаратный сброс микроконтроллера.

Для задания адресов, байт/слов данных и значений счетчика команд используются регистры модуля отладки.

Регистр DEBADDR

Используется, если требуется отследить появление на шине адресов желаемого значения. Как только значение слова адреса операнда на шине и значение, записанное в поле DEBADDRVAL, совпадут, это будет считаться событием. Для программирования отклика используется поле DEBADDREN регистра DEBCTRL.

Регистр DEBDATA

Используется, если требуется отследить появление на внутренней шине данных желаемого значения. Как только значение слова данных на шине и значение, записанное в поле DEBDATAWORD, совпадут, это будет считаться событием. Для программирования отклика используется поле DEBDATAEN регистра DEBCTRL.

Регистр DEBDATH и DEBDATL

Используются, если требуется отследить появление на внутренней шине данных желаемого значения только старшего или только младшего байта данных, что будет являться событием. Для программирования откликов используются поля DEBDATHEN и DEBDATLEN регистра DEBCTRL.

Регистры DEBPCEQ и DEBPCEQ2

Используются, если требуется отследить момент, когда счетчик команд достигнет желаемого значения. Как только значение счетчика команд и значение, записанное в поле DEBPCEQUAL, совпадут, это будет считаться событием. Для программирования отклика используется поле DEBPCEQEN регистра DEBCTRL.

Регистр DEBPC

Используется, если требуется отследить момент, когда счетчик команд превысит желаемое значение. Как только значение счетчика команд станет больше значения, записанного в поле DEBPCLESS, это будет считаться событием. Для программирования отклика используется поле DEBPCEN регистра DEBCTRL.

Важно помнить, что значение счетчика команд на момент возникновения события указывает адрес команды, следующей за выполняемой, и может отличаться от реального хода выполнения программы (при наличии перехода). При защите от сбоев не стоит настраивать регистр DEBPC на адрес, располагающийся непосредственно за последней командой в адресном пространстве (обычно команда перехода), так как возникновение этого значения в блоке основного счетчика команд не является ошибкой (но перехода по нему не будет).

Обращение к внешней памяти

Если требуется отследить момент, когда происходит выборка команды из области внешней памяти, то для этого нужно запрограммировать отклик посредством поля EXTMEMEN регистра DEBCTRL. Поскольку по умолчанию значение поля EXTMEMEN равно 00b, то логика модуля отладки не будет воспринимать обращение к внешней памяти как событие до тех пор, пока состояние поля не будет перепрограммировано.

Функционирование модуля отладки и управление его работой

Для программирования откликов на события используется регистр DEBCTRL. По умолчанию состояние регистра равно 0000h, что запрещает любые действия модуля отладки. Логика модуля OCDS начинает отслеживать желаемое событие сразу же после того, как для этого события будет запрограммирован отклик (т.е. значение соответствующего поля не равно 00b).

Для наглядности на рисунке 10.1 представлена структурная схема логики модуля OCDS, позволяющая формировать программируемые отклики на желаемые события. Семь линий откликов «прерывание DEBUG» далее объединяются по ИЛИ и формируют, таким образом, одну линию прерывания DEBUG, показанную на рисунке 10.2. Аналогично объединяются между собой линии «переход в IDLE» и «Сброс».

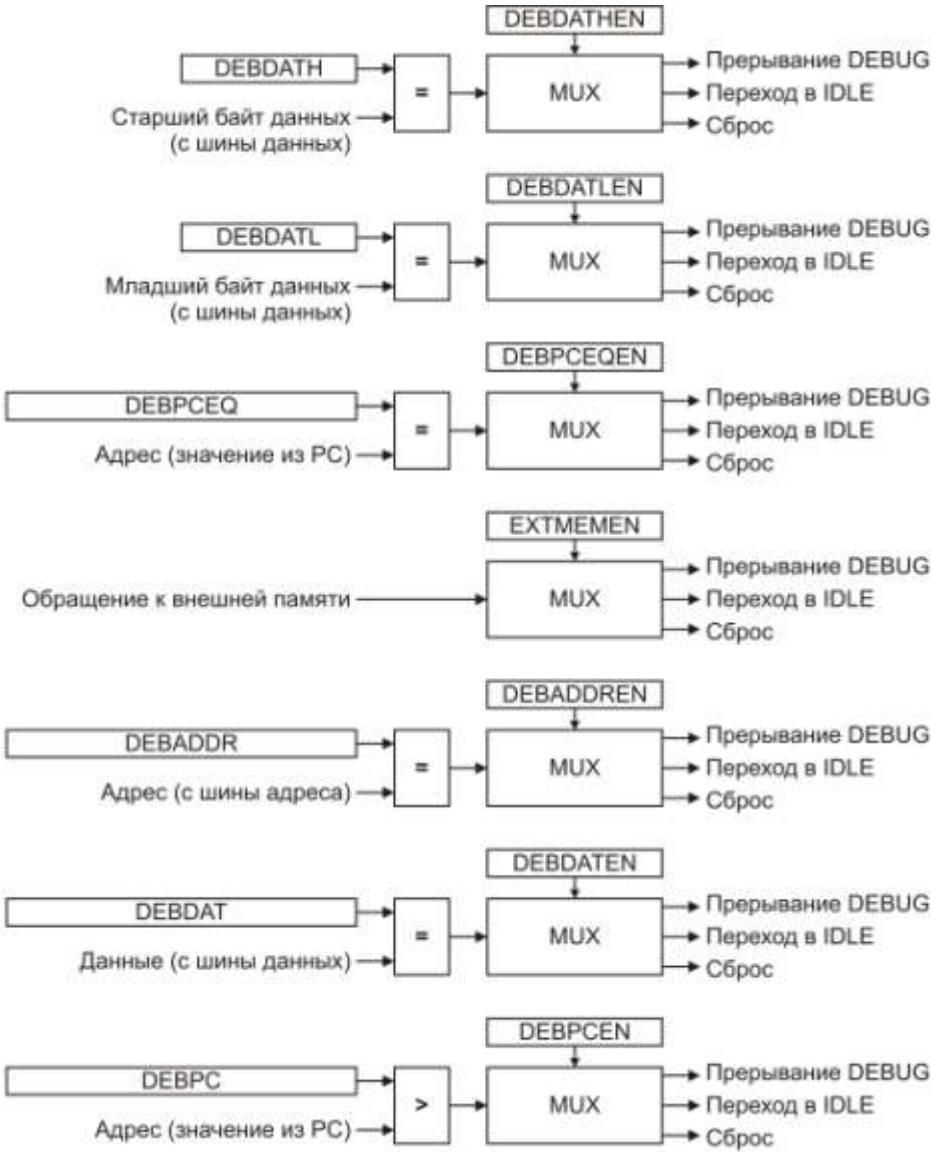


Рисунок 10.1 – Структурная схема работы модуля отладки

Прерывание DEBUG имеет самый высший приоритет и является немаскируемым. Оно не может назначаться на PTS. Время отклика на запрос определяется так же, как и для стандартных прерываний. Данное прерывание имеет бит ожидания, но он не доступен для записи и чтения.

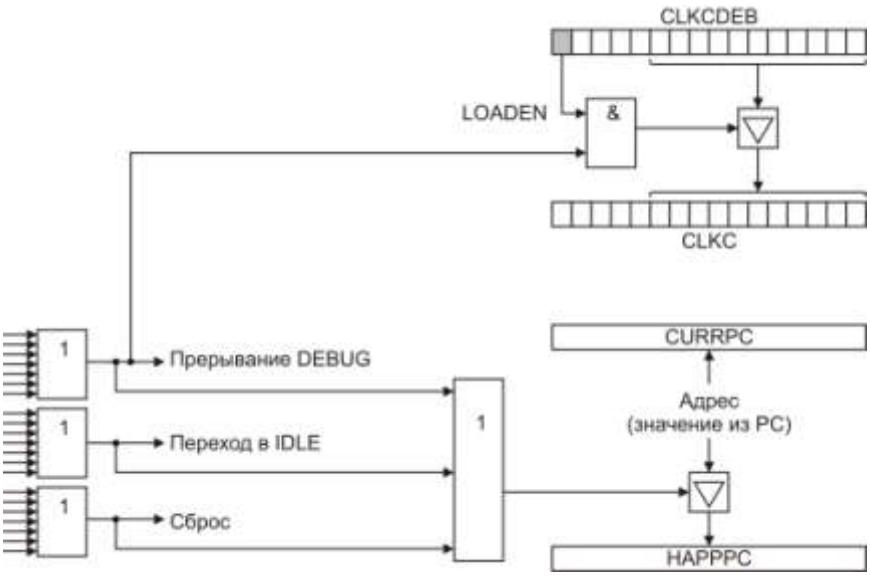


Рисунок 10.2 – Дополнительная структурная схема работы модуля отладки

Регистр HAPPPC

Из схемы на рисунке 10.2 видно, что при обнаружении любого события, независимо от запрограммированного отклика, будет происходить запись текущего состояния счетчика команд PC в регистр HAPPPC. Особенностью регистра является то, что он не сбрасывается во время сброса микроконтроллера

Регистр CURRPC

На рисунке 10.2 также указан регистр CURRPC. Значение, находящееся в этом регистре, всегда совпадает со значением счетчика команд.

Управление синхронизацией периферийных устройств

Одновременно с возникновением прерывания DEBUG модуля отладки может осуществляться перезапись регистра CLKC (регистр управления тактовыми сигналами периферийных устройств), что позволяет отключать/включать периферию. При этом управляющие регистры и регистры состояний будут сохранять свои значения, которые были на момент отключения.

Следует помнить, что после отключения блока его регистры доступны только для чтения.

Для того, чтобы осуществить перезапись регистра CLKC, необходимо установить бит LOADEN регистра CLKCDEB. Далее, как только будет сформировано прерывание DEBUG, логика модуля OCDS побитно перезагрузит регистр CLKC значением, которое хранится в регистре CLKCDEB (см. рисунок 10.2).

11 Блок кодирования по ГОСТ 28147–89

Блок кодирования/декодирования (далее блок кодирования) представляет собой аппаратную реализацию методов кодирования данных на основе алгоритмов ГОСТ 28147–89.

Блок кодирования позволяет производить кодирование/декодирование данных (блоками по четыре слова) тремя симметричными методами:

- простой заменой;
- гаммированием;
- гаммированием с обратной связью.

Важным дополнением является возможность блока кодирования формировать на основе исходных данных контрольную комбинацию, так называемую, имитовставку (аналог контрольной суммы), которая позволяет создавать дополнительную защиту данных.

Функциональное описание

В состав блока входят регистры управления CDCON, CDDATNUM (оба доступны только побайтно), CDSTATE и основные, и дополнительные регистры для кодирования исходных данных CDUW, CDAUTH, CDKEY и CDSBSTN, а также регистр исходных/полученных данных CDDATA.

Посредством регистра управления CDCON осуществляется конфигурирование блока, а также запуск и остановка (при необходимости) кодирования.

Дополнительным регистром для задания числа блоков данных, подлежащих обработке, является регистр CDDATNUM. Фактически, регистр CDDATNUM является счетчиком и находящееся в нем значение уменьшается на единицу каждый раз, по окончании обработки очередного блока данных. Достижение регистром нулевого состояния указывает на то, что все данные обработаны. Одновременно с этим в регистре состояния CDSTATE устанавливается флаг DONE.

При использовании регистра CDDATNUM, его состояние должно быть задано до начала процесса кодирования/декодирования (до установки бита START).

В случае, если к моменту установки бита START состояние регистра равно нулю, то окончание преобразования всего потока данных должно быть указано своевременной установкой бита STOP.

Блок оперирует с 8-байтными блоками исходных данных. Таким образом, если имеется массив данных, он должен быть разбит на блоки по восемь байт (по четыре слова). Если последний блок данных меньше 8 байт, то он должен быть дополнен любым набором бит до размера в 64 бита.

Для загрузки одного блока данных используется регистр данных CDDATA. Регистр является 64-битным. Запись в регистр происходит пословно по адресу, указанному в таблице 11.1. Если данные поступают в порядке D0, D1, D2 и т.д., то их размещение в регистре CDDATA будет соответствовать указанному на рисунке 11.1.

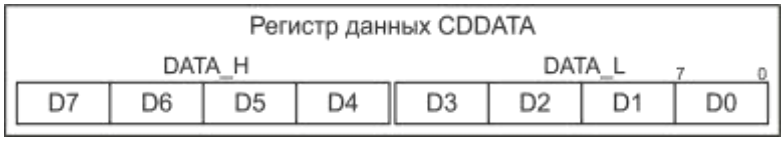


Рисунок 11.1 – Структура регистра CDDATA

В пределах регистра данные разбиваются на два 32-разрядных подблока с названиями DATA_H и DATA_L.

Помимо источника данных для преобразования, регистр CDDATA также является приемником обработанных данных, для получения которых требуется четырехкратное

использование команды чтения. Порядок получения данных при чтении – D0, D1, D2 и т. д. Для записи и чтения регистра CDDATA используется один адрес.

В состав блока кодирования входят ключевое устройство с регистром CDKEY и блок замены с регистром CDSBSTN.

Регистр CDKEY является 256-битным регистром ключа. Запись в регистр происходит пословно. Если данные поступают в порядке K0, K1, K2 и т.д., то их размещение в регистре CDKEY будет соответствовать указанному на рисунке 11.2. Регистр CDKEY доступен только для записи. На рисунке 11.3 приведена структура регистра CDSBSTN.

Регистр ключа CDKEY

KEY7	K31	K30	K29	K28
KEY6	K27	K26	K25	K24
KEY5	K23	K22	K21	K20
KEY4	K19	K18	K17	K16
KEY3	K15	K14	K13	K12
KEY2	K11	K10	K9	K8
KEY1	K7	K6	K5	K4
KEY0	K3	K2	K1	K0

Рисунок 11.2 – Структура регистра CDKEY

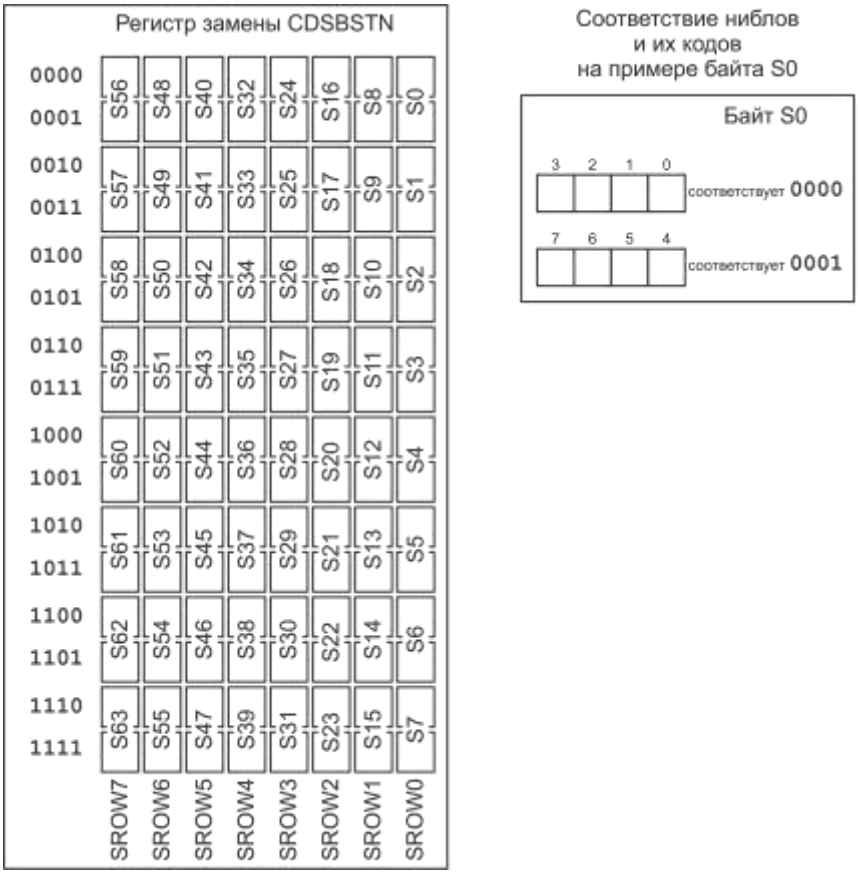


Рисунок 11.3 – Структура регистра CDSBSTN

В пределах регистра данные разбиваются на восемь подблоков с названиями KEY0, KEY1, KEY2, KEY3, KEY4, KEY5, KEY6, KEY7. Каждый из этих подблоков также является ключом и используется в процессе кодирования/декодирования. Выбор ключа осуществляет ключевое устройство под управлением логики.

Регистр CDSBSTN является 512-битным регистром замены. Запись в регистр происходит пословно. Если данные поступают в порядке S0, S1, S2 и т.д., то их размещение в регистре CDSBSTN будет соответствовать указанному на рисунке 11.3. Регистр CDSBSTN доступен только для записи.

В пределах регистра данные разбиваются на восемь подблоков с названиями SROW0, SROW1, SROW2, SROW3, SROW4, SROW5, SROW6, SROW7. Структура регистра CDSBSTN, показанная на рисунке 11.3, соответствует таблице подстановки, применяемой в процессе кодирования/декодирования, с восемью столбцами и 16 строками. Каждому полубайту соответствует код (двоичный порядковый номер).

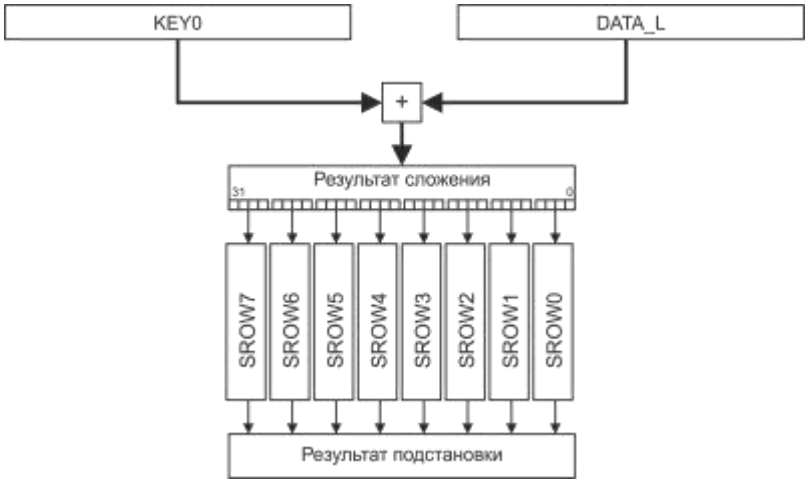


Рисунок 11.4 – Частичная функциональная схема алгоритма режима простой замены

Согласно алгоритму кодирования/декодирования, двойное слово обрабатываемых данных разбивается на восемь полубайт, каждому из которых соответствует один из столбцов (см. рисунок 11.4). Значение нибла является кодом выбора полубайта из регистра замены. Выбранные полубайты в свою очередь образуют двойное слово результата подстановки.

Помимо рассмотренных ключа и таблицы подстановки, для кодирования/ декодирования данных, в некоторых случаях требуются дополнительные исходные данные, называемые синхропосылкой. Для хранения синхропосылки используется регистр CDUW. Регистр является 64-битным. Запись в регистр происходит пословно. Если данные поступают в порядке UW0, UW1, UW2 и т.д., то их размещение в регистре CDUW будет соответствовать указанному на рисунке 11.5. Регистр CDUW доступен только для записи. В пределах регистра данные разбиваются на два 32-разрядных подблока с названиями UW_H и UW_L.

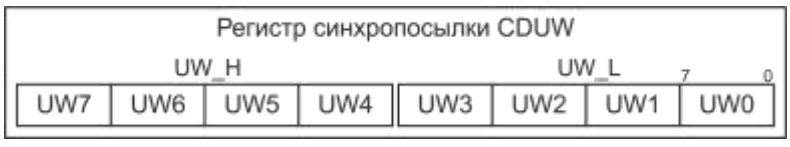


Рисунок 11.5 – Структура регистра CDUW

Для обеспечения имитозащиты данных применяется имитовставка, которая может вычисляться параллельно с процессом кодирования/декодирования. Результат вычисления, представляющий собой конкатенацию двойных слов AU_H и AU_L, помещается в регистр CDAUTH. Размещение данных в регистре CDAUTH будет соответствовать указанному на рисунке 11.6.

Рисунок 11.6 – Структура регистра CDAUTH

Регистр является 64-битным и доступен только для чтения. Для получения имитовставки требуется четырехкратное использование команды чтения. Порядок получения данных при чтении – AU0, AU1, AU2 и т. д.

В связи с тем, что почти все регистры блока имеют разрядность выше восьми, в процессе работы возможны ситуации некорректного задания исходных данных (недозагрузка регистра) или неполного прочтения вычисленных.

В зависимости от ситуации, логика блока установит соответствующий флаг ошибки или установит флаг предупреждения в регистре CDSTATE. Регистр CDSTATE является регистром состояния блока и доступен только для чтения.

Примечание – Блок кодирования не генерирует прерывания, поскольку время, затрачиваемое на кодирование/декодирование, во много меньше времени выполнения одной (любой) команды из системы команд микроконтроллера. Это означает, что после запуска преобразования можно следующей командой уже считывать результат.

Таблица 11.1 – Адреса доступа к 64-/256-/512-битным регистрам блока кодирования

Мнемоническое название регистра	Разряд-ность, бит	Адрес доступа к регистру	Обязательное количество последовательных циклов обращения к регистру пословно	Состояние после сброса
CDDATA	64	1F30h	4 (запись/чтение)	00h
CDKEY	256	1F34h	16 (только запись)	00h
CDSBSTN	512	1F36h	32 (только запись)	00h
CDUW	64	1F38h	4 (только запись)	00h
CDAUTH	64	1F3Eh	4 (только чтение)	00h

Режимы работы

В зависимости от состояния битового поля MODE регистра CDCON, блок может функционировать в одном из трех режимов кодирования/декодирования данных.

Режим простой замены

Режим кодирования блоков данных, опирающийся на алгоритм, в котором блоки открытых данных кодируются с помощью ключа и таблицы замены с получением блока закодированных данных. Декодирование данных происходит в обратном, относительно кодирования, порядке с теми же ключом и таблицей замены.

Режим гаммирования и гаммирования с обратной связью

Режимы кодирования блоков данных, опирающиеся на алгоритм, в которых открытые данные кодируются с помощью гамм-кодов, получаемых на основе закодированных с помощью ключа и таблицы замены синхровставок с получением блоков

закодированных данных. Декодирование данных происходит в обратном, относительно кодирования, порядке с теми же синхровставкой, ключом и таблицей замены.

Вычисление имитовставки (доступно для всех режимов)

Вычисление имитовставки может происходить параллельно с выполнением основного алгоритма выбранного режима. Механизм вычисления имитовставки единообразен для всех режимов и включается установкой бита AUTH регистра CDCON. Следует помнить, что вычисление замедляет процесс кодирования/декодирования в 1,5 раза.

Вычисление опирается на алгоритм, в котором данные преобразуются с помощью ключа и таблицы замены с получением 64-битной имитовставки. По окончании преобразования всех данных имитовставка может быть считана из регистра CDAUTH.

Имитовставка, полученная вместе с закодированными данными, отделяется от них. Данные декодируются, и параллельно с этим вычисляется имитовставка. По окончании декодирования всех данных вычисленная имитовставка считывается из регистра CDAUTH и сравнивается с полученной. При несовпадении декодированные данные считаются ошибочными.

Время преобразования данных в каждом из режимов обратно пропорционально внешней тактовой частоте. Так, при внешней тактовой частоте 20 МГц и включенном делителе на два, время преобразования составляет менее 2 мкс, а с вычислением имитовставки – менее 3 мкс.

Для более подробного ознакомления с алгоритмами, лежащими в основе вычислений блока, следует обратиться к ГОСТ 28147–89.

Порядок работы с блоком

1 Запуск преобразования

- в регистр ключа CDKEY записываются 16 слов данных;
- в регистр замены CDSBSTN записываются 32 слова данных;
- если предполагается использование одного из режимов гаммирования, то в регистр синхропосылки CDUW записываются четыре слова данных;
- в регистр данных CDDATA записываются четыре слова данных, подлежащих кодированию/декодированию;
- если известно количество блоков данных и если предполагается использование регистра CDDATNUM (только побайтная запись), то соответствующее значение должно быть записано в регистр;
- в регистре управления CDCON (только побайтная запись) программируются биты, отвечающие за алгоритм преобразования данных, необходимость вычисления имитовставки, режим работы, и устанавливается бит START.

2 Получение данных и дальнейшая работа

- по окончании кодирования следует прочитать регистр CDSTATE и проверить состояние флага DONE, указывающего на то, что преобразование завершилось успешно;
- если флаг DONE установлен, то преобразованные данные могут быть прочитаны из регистра CDDATA;
- если обработанный блок данных был не последний (согласно состоянию регистра CDDATNUM и/или нулевому состоянию бита STOP), то в регистр CDDATA записывается очередной блок данных, после чего процесс преобразования запускается автоматически (устанавливать бит START не нужно);
- если обработанный блок данных был последним и имитовставка не вычислялась, цикл работы с блоками данных считается завершенным;
- если был установлен бит AUTH и, соответственно, была вычислена имитовставка, она может быть прочитана из регистра CDAUTH, после чего цикл работы с блоками данных считается завершенным.

Особенности работы блока

Для обработки только одного блока загруженных данных нужно записать значение 01h в регистр CDDATNUM и установить бит START или одновременно установить биты START и STOP.

Количество циклов записи/чтения 32-/256-/512-и разрядных регистров должно точно соответствовать указанному в таблице 11.1. Если число записанных/прочитанных слов окажется меньше или больше указанного, регистр будет считаться незаполненным/непрочитанным.

Если на момент запуска преобразования в регистре CDDATA и/или регистре CDAUTH находятся непрочитанные данные, то это не влияет на запуск, а логика устанавливает флаг/флаги предупреждения в регистре CDSTATE. Следует помнить, что непрочитанные данные будут утеряны, вследствие записи поверх них новых вычисленных значений.

До перезаписи по окончании преобразования, данные в регистре CDDATA сохраняются и могут быть прочитаны многократно.

Независимо от состояния регистра CDDATNUM, установка бита STOP перед записью очередного блока данных будет означать, что этот блок данных последний. Следует помнить, что регистр CDDATNUM не сбрасывается аппаратно в случае преждевременного завершения обработки блоков данных. Поэтому следует контролировать состояние этого регистра и, при необходимости, перепрограммировать его перед каждой установкой бита START.

В любой момент все регистры (кроме CDCON) и конечный автомат преобразования блока кодирования могут быть сброшены установкой бита RST. При необходимости сброса только конечного автомата преобразования с сохранением состояния всех регистров нужно записать значение 00h в поле MODE.

12 Последовательные порты ввода-вывода USART

Микроконтроллер содержит два идентичных расширенных универсальных асинхронных приемопередатчика типа USART с поддержкой обнаружения ошибок посылки. Через приемопередатчики осуществляется прием и передача информации, представленной последовательным кодом (младшими битами вперед), в полном дуплексном режиме обмена.

Условные названия приемопередатчиков – USART0 и USART1. Далее символы «0» и «1», указывающие на принадлежность регистров, выводов и др., к тому или иному приемопередатчику не будут использоваться, если это не затрудняет понимание текста. Дальнейшее описание применимо как к USART0, так и к USART1 вследствие их полной функциональной идентичности.

В состав приемопередатчика USART входят принимающий и передающий сдвиговые регистры (недоступные программно), а также специальные буферные регистры SBUF_TX и SBUF_RX для записи данных для передачи и чтения полученных данных.

Запись байта в регистр SBUF_TX приводит к автоматическому переносу этого байта в сдвиговый регистр передатчика и инициирует начало передачи (некоторые особенности указаны ниже). По окончании передачи полученные данные можно получить, прочитав регистр SBUF_RX. Наличие буферного регистра хранения полученного байта позволяет совмещать чтение ранее принятого байта с приемом очередного. Однако, следует помнить, что если к моменту окончания приема очередного байта предыдущий байт не был считан из SBUF_RX, то он будет потерян (перезаписан новым байтом).

Блок USART имеет функцию фиксации ошибок посылки, при использовании которой отслеживаются все стоповые биты. При обнаружении пропущенного бита устанавливается флаг FE в регистре SCONx.

Последовательный порт USART может работать в четырех режимах – синхронном (режим 0) и асинхронных (режимы 1, 2, 3). Режим задается битовым полем SER_MODE регистра SP_CON.

12.1 Синхронный режим работы

Режим 0 (посылка 8 бит)

Синхронный режим, в основном используемый для записи и чтения внешнего параллельно-последовательного регистра. Посылки данных по 8 бит передаются или принимаются младшим битом вперед через вывод RxD микроконтроллера. Тактирование передачи/приема осуществляется синхросигналом, который генерируется блоком USART на выводе TxD (см. рисунок 12.1).

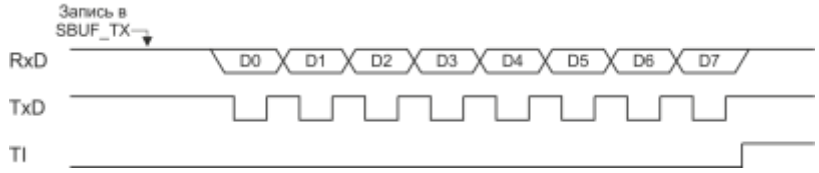


Рисунок 12.1 – Временная диаграмма передачи байта в режиме 0

Для инициализации передачи сначала следует очистить бит REN в регистре SP_CON (разрешение работы вывода RxD как выхода данных), а затем записать байт данных в регистр SBUF_TX. После записи данных в регистр SBUF_TX на аппаратном уровне происходит перезапись этого байта в передающий сдвиговый регистр, запуск генератора синхросигнала и инициализация передачи. С каждым тактом синхросигнала содержимое сдвигового регистра сдвигается вправо (в сторону младших битов) и поступает на вывод RxD. В освобождающиеся биты передающего сдвигового регистра записываются нули.

По окончании передачи восьмого бита устанавливается флаг TI в регистре SP_STAT, и формируется запрос на прерывание. Флаг TI не сбрасывается аппаратно с началом передачи очередного байта. Сброс флага происходит только при чтении регистра SP_STAT. Следует помнить, что в режиме 0 (и только в этом режиме) выход RxD функционирует как выход с открытым стоком. Поэтому на соответствующей линии передачи желательно использовать подтягивающий резистор номиналом 15 кОм (см. рисунок 12.2).

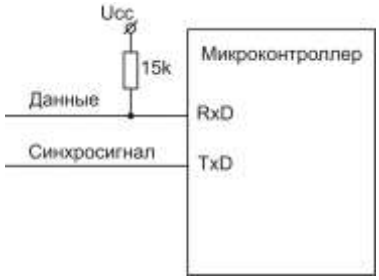


Рисунок 12.2 – Установка подтягивающего резистора

Для инициализации приема следует установить бит REN.

После обнаружения на входе RxD отрицательного импульса (длительностью не менее двух тактов XTAL1) запускается механизм приема данных. По окончании приема устанавливается флаг RI и формируется запрос на прерывание. Полученные данные могут быть прочитаны из регистра SBUF_RX.

Флаг RI не сбрасывается аппаратно. Сброс флага происходит только при чтении регистра SP_STAT.

Примечание – Пока флаг RI установлен, прием очередного байта невозможен. Если установлен бит REN, то сброс флага RI включает механизм приема данных. Для того, чтобы этого не произошло, следует очистить бит REN перед чтением регистра SP_STAT.

12.2 Асинхронные режимы работы

Режим 1 (посылка 10 бит)

В этом асинхронном режиме посылки данных по 10 бит передаются (младшим битом вперед) через вывод TxD микроконтроллера и принимаются (младшим битом вперед) через вывод RxD (см. рисунок 12.3).

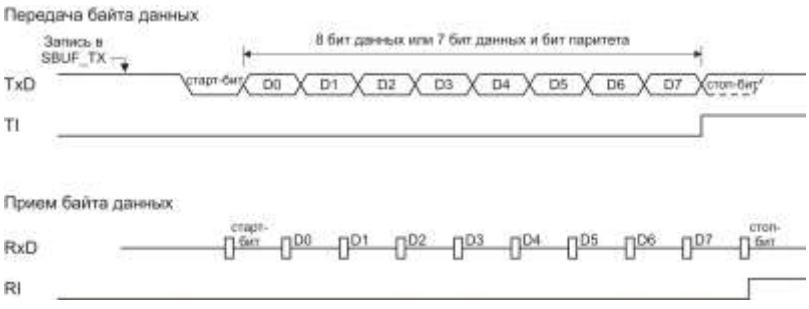


Рисунок 12.3 – Временная диаграмма передачи в режиме 1

Одна посылка состоит из старт-бита (всегда «0»), 8 бит данных и стоп-бита (всегда «1»). Если разрешен контроль по паритету (установлен бит PEN в регистре SP_CON), то посылка состоит из старт-бита, 7 бит данных, бита паритета (вместо восьмого бита данных) и стоп-бита.

Для инициализации передачи следует записать байт данных в регистр SBUF_TX. После записи байта в регистр SBUF_TX, на аппаратном уровне происходит перезапись этого байта в передающий сдвиговый регистр, запуск генератора синхросигнала для задания скорости передачи и инициализация передачи. Содержимое сдвигового регистра сдвигается вправо (в сторону младших битов) и поступает на выход TxD. В освобождающиеся биты передающего сдвигового регистра записываются нули. По окончании передачи девятого бита посылки и перед началом передачи стоп-бита устанавливается флаг TI и генерируется прерывание. Передача следующего байта данных начнется только после того, как будет отправлен стоп-бит текущей передачи.

Если бит REN установлен и модуль USART находится в режиме ожидания, то появление возрастающего фронта сигнала на входе RxD вызовет запуск генератора синхросигнала для определения скорости передачи и инициализирования приема. Перед окончанием стоп-бита устанавливается флаг RI и формируется запрос на прерывание.

Значение принятого бита 9 всегда сохраняется в бите RPE/RB8 регистра SP_STAT. В случае, если разрешен контроль по паритету, то состояние бита RPE/RB8 будет зависеть от результата проверки. Полученный байт данных переписывается в регистр SBUF_RX, откуда может быть прочитан.

В режиме 1 тактовый сигнал не передается, поэтому для согласованной работы устройств они должны быть настроены на одну скорость передачи. Скорость передачи задается регистром BAUD_RATE.

Режимы 2 и 3 (посылка 11 бит)

В этом асинхронном режиме посылки данных по 11 бит передаются (младшим битом вперед) через вывод TxD микроконтроллера и принимаются (младшим битом вперед) через вывод RxD (см. рисунок 12.4). Одна посылка состоит из старт-бита (всегда «0»), 8 бит данных, программируемого девятого бита данных и стоп-бита (всегда «1»).

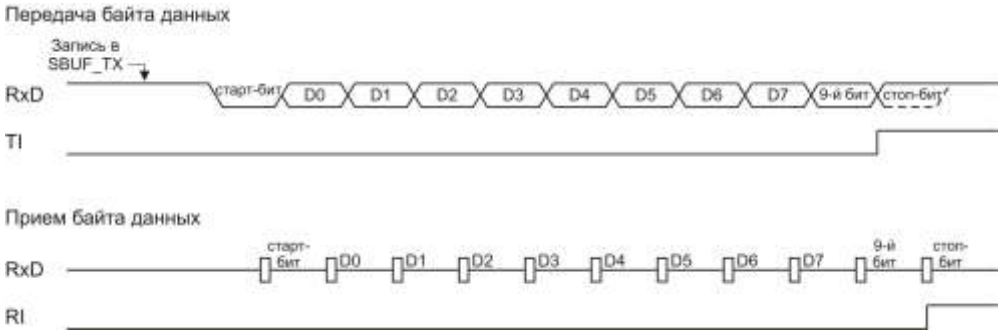


Рисунок 12.4 – Временная диаграмма передачи в режимах 2 и 3

Для инициализации передачи следует записать байт данных в регистр SBUF_TX. После записи байта в регистр SBUF_TX на аппаратном уровне происходит перезапись этого байта в передающий сдвиговый регистр, запуск генератора синхросигнала для задания скорости передачи и инициализация передачи. Содержимое сдвигового регистра сдвигается вправо (в сторону младших битов) и поступает на выход TxD. В освобождающиеся биты передающего сдвигового регистра записываются нули. По окончании передачи десятого бита посылки и перед началом передачи стоп-бита устанавливается флаг TI и генерируется прерывание. Передача следующего байта данных начнется только после того, как будет отправлен стоп-бит текущей передачи.

Десятый передаваемый бит – это бит, значение которого может быть задано посредством бита TB8 регистра SP_CON. Состояние бита TB8 должно быть задано до начала передачи (то есть до записи регистра SBUF_TX). По окончании передачи бит TB8 очищается аппаратно, поэтому его следует устанавливать (если необходимо) перед началом каждой передачи.

Если бит REN установлен и модуль USART находится в режиме ожидания, то появление возрастающего фронта сигнала на входе RxD вызовет запуск генератора синхросигнала для определения скорости передачи и инициализирование приема. Время установки флага RI и формирование запроса на прерывание идентично режиму 1. Значение десятого принятого бита всегда сохраняется в бите RPE/RB8 регистра SP_STAT. Полученный байт данных переписывается в регистр SBUF_RX, откуда может быть прочитан.

Следует помнить, что в режиме 2 установка флага RI и формирование запроса на прерывание допускается только, если установлен бит TB8, а контроль по паритету не может быть включен.

По формату передачи режим 3 идентичен режиму 2, поэтому временная диаграмма, представленная на рисунке 12.4, справедлива для обоих режимов. В отличие от режима 2, в режиме 3 может быть включен контроль по паритету. Если бит разрешения контроля по паритету (PEN) регистра SP_CON очищен, то значение десятого передаваемого бита задается состоянием бита TB8. Если бит PEN установлен, то десятый передаваемый бит является битом паритета. Поведение флага TI и запроса на прерывание идентично режиму 2.

При приеме, если бит PEN очищен, то значение десятого принятого бита сохраняется в бите RPE/RB8 регистра SP_STAT; если бит PEN установлен, то десятый принятый бит является битом паритета, а бит RPE/RB8 функционирует как флаг ошибки паритета. В отличие от режима 2, в режиме 3 состояние бита RB8 не влияет на установку флага RI и формирование запроса на прерывание. По окончании приема без ошибок полученный байт данных переписывается в регистр SBUF_RX, откуда может быть прочитан.

В режимах 2 и 3 тактовый сигнал не передается, поэтому для согласованной работы устройств они должны быть настроены на одну скорость передачи. Скорость передачи задается регистром BAUD_RATE.

12.3 Скорость приема/передачи

Скорость передачи/приема данных задается посредством регистра BAUD_RATE. Регистр позволяет указать источник синхросигнала для внутреннего генератора блока USART и скорость обмена информацией между устройствами.

Источником синхросигнала может быть вход XTAL1 или вход T2CLK микроконтроллера (см. рисунок 12.5). Если установлен бит SOURCE регистра BAUD_RATE, то источником является вход XTAL1 и значение частоты синхросигнала будет fusart = 0,5 × fxtal1, в противном случае, fusart = ft2clk.

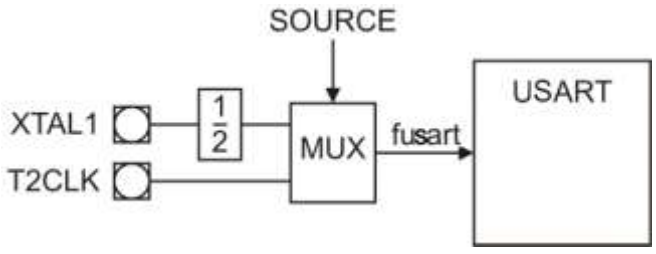


Рисунок 12.5 – Управление выбором источника синхронизации

Значение, записываемое в поле BR_VALUE регистра BAUD_RATE, связано со скоростью передачи данных, выраженной в бодах, следующими соотношениями:

$$BR\_VALUE_{DEC} = \left(\frac{fusart}{Bod} \right) - 1 \quad \text{(для режима 0);} \tag{12.1}$$

$$BR\_VALUE_{DEC} = \left(\frac{fusart}{Bod \times 8} \right) - 1 \quad \text{(для режимов 1, 2 и 3).} \tag{12.2}$$

В формулах 12.1 и 12.2 приняты обозначения:
- BR_VALUE_{DEC} – значение BR_VALUE в десятичном формате;
- fusart – частота синхросигнала, поступающего на блок (Гц);
- Bod – желаемая скорость передачи (бод).

Пример расчета значения BR_VALUE для желаемой скорости передачи данных, равной 9 600 бод, в режиме 1. В качестве источника синхросигнала выбран вход XTAL1 (SOURCE = 1b), на который подается сигнал с частотой 16 МГц.

$$BR\_VALUE_{DEC} = \left(\frac{8 \times 10^6}{9600 \times 8} \right) - 1 = 103. \tag{12.3}$$

При переводе в шестнадцатеричный формат получается значение 67h. Таким образом, в поле BR_VALUE следует записать значение 0067h. С учетом того, что бит SOURCE установлен, в регистр BAUD_RATE в итоге записывается значение 8067h.

Для режима 0 в поле BR_VALUE может быть записано максимальное значение 3FFFh и минимальное значение 0001h.

Для режимов 1, 2 и 3 в поле BR_VALUE может быть записано максимальное значение 3FFFh и минимальное значение 0000h (источник синхронизации XTAL1) или 0001h (источник синхронизации T2CLK).

Регистр BAUD_RATE является 16-разрядным с последовательной загрузкой. Значение, которое должно быть записано в регистр, разбивается на два байта – старший и младший. Первым по адресу регистра BAUD_RATE записывается значение младшего байта, а вторым, по тому же адресу – значение старшего байта.

12.4 Конфигурирование выводов TxD0, TxD1, RxD0 и RxD1

Функционирование нулевого и первого, четвертого и шестого выводов порта P2 микроконтроллера как входов/выходов модулей USART0 и USART1 соответственно является альтернативной функцией порта. Для включения альтернативной функции следует установить бит TXD_SEL в регистре IOC1. Этот бит одновременно разрешает работу выводов порта P2.0 и P2.6 как TxD0 и TxD1 соответственно. Для выключения альтернативной функции следует очистить бит TXD_SEL.

Альтернативные функции выводов P2.1 и P2.4 (RxD0 и RxD1) включаются битами REN регистров SP_CON0 и SP_CON1 соответственно. Когда эти биты установлены, выводы порта функционируют как входы модулей USART0 и USART1. Очистка бита REN запрещает прием данных на соответствующем выводе и переводит его в вывод общего назначения порта P2.

12.5 Прерывания

Управление прерываниями модулей USART0 и USART1 осуществляется посредством регистров INT_MASK, INT_MASK1, INT_PEND и INT_PEND1.

Если требуется формировать запрос на прерывание по окончании передачи или приема, то следует установить бит TI_RI_MASK в регистре INT_MASK1. При возникновении запроса на прерывание в регистре INT_PEND1 будет аппаратно установлен бит TI_RI_PEND. Также этот бит может быть установлен программно.

Следует помнить, что программная установка флагов RI и TI в регистрах SP_STAT0 и SP_STAT1 не приводит к формированию запросов прерываний, а сброс этих флагов при чтении SP_STAT0 и SP_STAT1 не влияет на состояние регистров INT_PEND и INT_PEND1.

13 Интерфейс I2C

Модуль I2C обеспечивает полную поддержку двухпроводного последовательного синхронного интерфейса I2C/SMBus. Результат такой совместимости – легкое соединение со многими запоминающими устройствами и устройствами ввода-вывода, включая EEPROM, SRAM, счетчики, АЦП, ЦАП, периферийные устройства.

Функциональные возможности модуля:

- совместимость с протоколами SMBus 1.1 и SMBus 2.0, ACCESS.Bus, I2C 2.1;
- поддержка скоростного/стандартного (FS) и высокоскоростного (HS) режимов;
- программирование действий мастера/ведомого;
- возможность подключения к шине нескольких ведущих устройств, т.е. поддержка режима мультимастер (MM);
- один программно задаваемый адрес;
- 7- или 10-битная адресация ведомого;
- поддержка адреса общего вызова.

Особые возможности SMBus:

- отслеживание времени простоя линии SCL;
- наличие функции отслеживания ошибок в пакетах данных (PEC) с использованием метода расчета контрольной суммы (CRC);
- поддержка адреса отклика мастера;
- поддержка полинга и контроля прерываний.

13.1 Протокол шины

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

Протокол поддерживает режим мультимастер, в котором шина может контролироваться одним или несколькими устройствами из подключенных к шине. Каждое устройство, подключенное к шине, имеет свой адрес и может быть как приемником, так и передатчиком (некоторые только приемниками).

Операции с данными

Устройство, которое начинает передачу данных, становится мастером. Мастер генерирует тактовый сигнал SCL, а также инициирует и завершает передачу данных по шине. За один такт сигнала SCL передается один бит данных по линии SDA (рисунок 13.1).

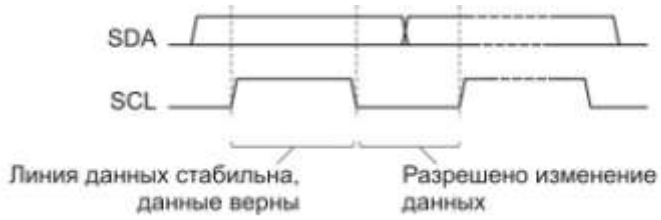


Рисунок 13.1 – Передача бита данных

Данные валидны (верны), пока уровень сигнала на линии SCL высокий. Когда на линии SCL низкий уровень сигнала, данные могут меняться.

Старт и стоп

Состояние старта формируется тогда, когда на линии SCL держится высокий уровень сигнала, а на линии SDA возникает перепад уровня сигнала из высокого в низкий (см. рисунок 13.2).

Состояние стопа (останова) формируется тогда, когда на линии SCL держится высокий уровень сигнала, а на линии SDA возникает перепад уровня сигнала из низкого в высокий (см. рисунок 13.2).

Состояния старта и стопа формирует только мастер.

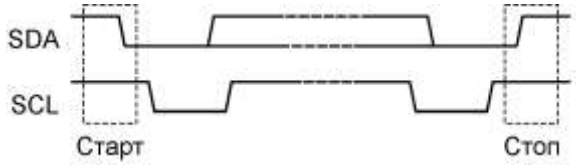


Рисунок 13.2 – Состояния старта и стопа

После того, как сформировано состояние старта, шина считается занятой и другие устройства не должны пытаться управлять ей. Шина считается занятой до тех пор, пока не будет сформировано состояние стопа. В середине передачи может быть сформировано состояние повторного старта, если мастеру нужно обратиться к другому ведомому или если требуется изменение направления передачи данных без потери контроля над шиной (см. рисунок 13.3).

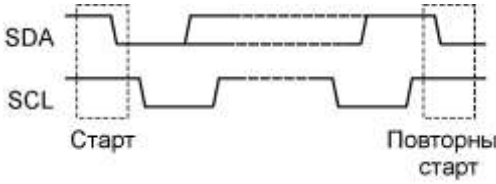


Рисунок 13.3 – Состояние повторного старта

Арбитраж

Арбитраж выполняется в момент времени, когда на линии SCL находится «1». Два устройства могут сгенерировать стартовое состояние в одно и то же время. Далее арбитраж будет продолжаться до тех пор, пока одно из устройств сформирует «0», а другое – «1» на линии SDA. Устройство, которое установило «1» на линии SDA, проигрывает арбитраж. На рисунке 13.4 приведен пример арбитража.

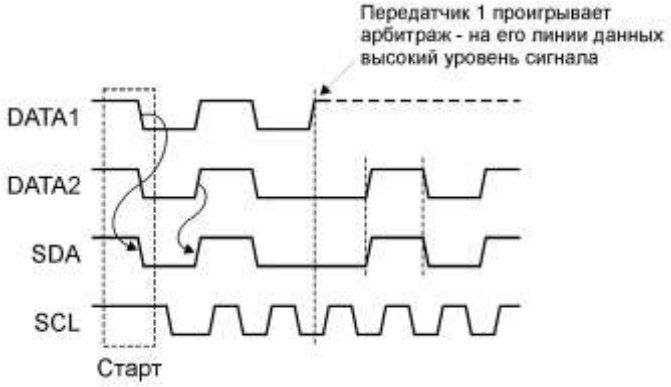


Рисунок 13.4 – Арбитраж на линии SDA

Два устройства передают свои данные DATA1 и DATA2 на линию SDA. В момент времени, когда очередной бит данных DATA1 равен «1», а бит данных DATA2 равен «0», второе устройство выигрывает арбитраж и продолжает передачу своих данных, а первое устройство прекращает передачу.

Если устройство проигрывает арбитраж во время передачи первого байта после старта (во время передачи адреса ведомого), оно становится ведомым приемником и мониторит передаваемый адрес на случай совпадения. Арбитраж также может быть проигран в режиме мастера приемника во время квитирования или в режиме ведомого передатчика во время ответа на адрес отклика на сигнал предупреждения.

В случае проигрывания арбитража в битовом поле MODE регистра SMBST устанавливается соответствующий код и генерируется прерывание.

Синхронизация

Синхронизация тактовых сигналов разных устройств, подключенных к шине I2C, реализуется в случаях, когда несколько устройств являются мастерами, и выполняется с использованием той особенности, что линия SCL реализована как монтажное «И» линий тактовых сигналов этих устройств. Для примера рассмотрим синхронизацию двух мастеров с линиями тактовых сигналов CLK1 и CLK2 (см. рисунок 13.5).

Рисунок 13.5 – Синхронизация

Линия SCL переводится в состояние «0» сразу, как только один из мастеров выставляет на своей линии тактового сигнала низкий уровень сигнала (CLK1 на рисунке 13.5). При этом его внутренний счетчик длительности низкого уровня сигнала сбрасывается и начинает отсчет. Второй мастер выставляет низкий уровень позже, и его счетчик также сбрасывается (CLK2).

Как только внутренний счетчик первого мастера переполнится, мастер выставит на линии CLK1 высокий уровень сигнала. Тем не менее, линия SCL будет по-прежнему оставаться в состоянии «0», удерживаемая вторым мастером. В связи с этим, первый мастер перейдет в состояние ожидания (см. рисунок 13.5). Когда переполнится счетчик второго мастера, он выставит на линии CLK2 высокий уровень сигнала, и в этот момент линия SCL перейдет в состояние «1». С этого момента внутренние счетчики длительности высокого уровня сигнала обоих мастеров начнут синхронный отсчет.

Каждая передача данных состоит из начального состояния «старт», состояний передач битов и состояния «стоп». Данные передаются старшим битом (MSB) вперед. Передача каждого байта завершается квитированием, т.е. приемник подтверждает окончание приема сигналом подтверждения (ACK). Ведомое устройство может увеличивать паузу между тактовыми импульсами, удерживая на линии SCL сигнал низкого уровня, пока происходит обработка принятых данных или подготовка данных для следующей передачи. Этот процесс может происходить после передачи любого бита/байта (см. рисунок 13.6).

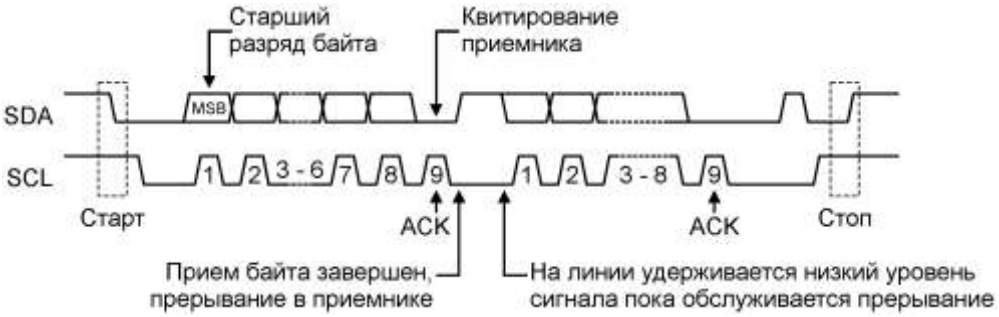


Рисунок 13.6 – Передача данных

Квитирование

Каждый байт посылки должен быть завершен квитированием, т.е. ответом на прием сигнала запроса подтверждения приема (ACK). На рисунках 13.6 и 13.7 показано положение момента квитирования в пределах посылки.

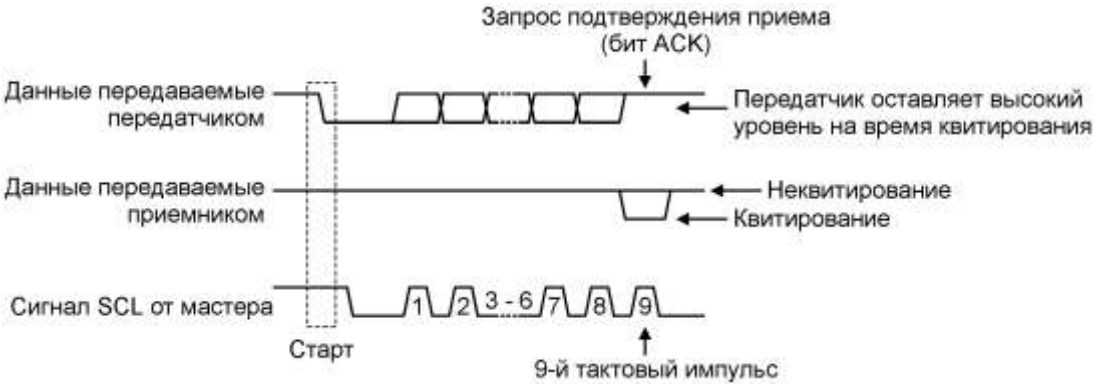


Рисунок 13.7 – Квитирование и неквитирование бита подтверждения ACK

Бит запроса подтверждения приема генерируется мастером. Передатчик (мастер или ведомый) в момент девятого такта синхросигнала оставляет линию SDA в состоянии «1» (бит ACK). В свою очередь, приемник должен сбросить линию SDA в «0» в течение времени, пока на линии SCL удерживается высокий уровень девятого импульса тактового сигнала, т.е. квитировать прием (см. рисунок 13.7). Если приемник не отвечает на запрос подтверждения и не подтверждает прием байта, то он оставляет линию SDA без изменений в состоянии «1», т.е. не квитирует прием.

Примечание – Все устройства, подсоединенные к шине I2C, в обязательном порядке должны квитировать бит ACK при получении байта с их собственным адресом. Этот механизм используется для отслеживания наличия отключившихся (самостоятельно или по каким-то причинам) от шины устройств.

Ведомое устройство имеет право не квитировать бит ACK в следующих случаях:

- если ведомый не может принять данные или он занят. Мастер, обнаружив неквитирование байта, должен сгенерировать состояние стопа и прервать передачу. Как альтернатива, ведомый может затянуть период низкого уровня сигнала тактирования на линии SCL для завершения своих операций и продолжить передачу;
- если ведомый обнаружил некорректную команду или некорректные данные. В этом случае, ведомый должен неквитировать принятый байт. Мастер, обнаружив неквитирование байта, должен сгенерировать состояние стопа и повторить передачу;
- если мастер функционирует как приемник, то, приняв байт, он должен сообщить ведомому об окончании данных неквитированием бита ACK, посланного ведомым. После

этого ведомый передатчик должен освободить линию SDA для того, чтобы мастер смог сгенерировать состояние завершения передачи (состояние стопа).

Формат передачи данных с 7-битной адресацией

На рисунке 13.8 показана передача адреса и двух байт данных. Каждому устройству, подключенному к шине, присваивается уникальный 7-битный адрес. Первые семь бит, передаваемые после старта, представляют собой адрес ведомого, восьмой бит (R/W#) определяет направление передачи – от ведомого (чтение, если R/W# = «1») или к ведомому (запись, если R/W# = «0»).

Рисунок 13.8 – Передача данных с 7-битной адресацией

Каждый ведомый, получивший байт адреса, сравнивает его со своим собственным адресом. Если адрес распознается как «свой», ведомый квитирует прием и далее, в зависимости от состояния бита R/W#, становится передатчиком или приемником.

Протокол SMBus/I2C позволяет генерировать адрес общего вызова для одновременного обращения ко всем устройствам, подключенным к шине. Первым передается адрес общего вызова (00h), затем следует байт назначения общего вызова. Ведомые, которые ожидают данные, квитируют этот байт и становятся приемниками, остальные игнорируют общий вызов.

Протокол SMBus/I2C поддерживает уникальную функцию – распознавание адреса отклика на сигнал предупреждения (Alert Response Address – ARA). В системах с несколькими ведомыми каждое устройство может послать мастеру сигнал предупреждения. Для этого используется дополнительная третья линия ALERT#, физически идентичная линиям SDA и SCL, реализованная по принципу монтажное «И». К этой линии также подключаются все устройства. Когда какому-то ведомому (или нескольким ведомым) необходимо обратиться к мастеру, он (или они) выставляет на линии ALERT# низкий уровень сигнала – это сигнал предупреждения (см. рисунок 13.9).

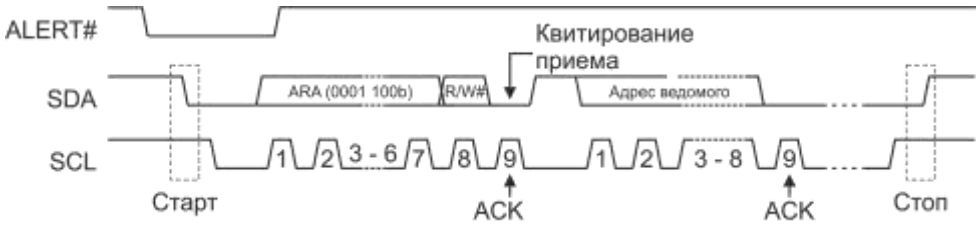


Рисунок 13.9 – Передача адреса отклика на сигнал предупреждения

Мастер, обнаружив «0» на линии ALERT#, обращается ко всем ведомым, посылая адрес отклика на сигнал предупреждения (ARA). Адрес состоит из семи битов (0001_100b) и бита R/W# = «1» (чтение). Ведомый, который отправил сигнал предупреждения, получив ARA, квитирует его и затем отправляет свой 7-битный адрес (восьмой бит может быть как «0», так и «1»), сообщая таким образом ведомому, какое именно устройство посылало сигнал предупреждения. Кроме этого, ведомый, который выставлял «0» на линии ALERT#, должен перестать удерживать линию, чтобы на ней установился высокий уровень сигнала. В том случае, если несколько устройств посылали сигнал предупреждения, то после получения ARA, свой адрес передает то устройство,

которое захватывает шину по стандартным правилам арбитража. Если после обслуживания ведомого мастер все еще обнаруживает на линии ALERT# низкий уровень сигнала, он понимает это как то, что сигнал предупреждения посылался несколькими ведомыми. Мастер снова отправляет ARA и затем общается со следующим ведомым. Появление на линии ALERT# высокого уровня сигнала означает, что все ведомые, которые требовали обращения, обслужены.

Примечание – Описываемый в настоящем ТО модуль I2C не имеет выделенной линии ALERT#. При необходимости, пользователь может задействовать свободный вывод микроконтроллера и программно реализовать возможность передачи сигнала предупреждения от ведомого к мастеру. В свою очередь, функция распознавания адреса отклика (ARA) и последующей отправки собственного адреса реализована полностью. Включить функцию можно установкой бита SMBARE в регистре SMBCTRL1.

Формат передачи данных с 10-битной адресацией

10-битная адресация позволяет адресовать до 1 024 ведомых устройств, с использованием резервной комбинации 1111_0xxb, которая передается по линии SDA сразу после старта. 10-битный формат полностью совместим с 7-битным форматом и может использоваться одновременно с ним, что позволяет соединять по шине I2C устройства с разной адресацией.

Основной идеей формата является передача 10-битного адреса в двух первых байтах, следующих сразу после старта. В первом байте передается значение 1111_0xxb, где «xx» – это два старших бита адреса и бит R/W# (на рисунке 13.10 обозначен символом «W» – запись), который должен быть равен «0», чтобы ведомый понял, что в следующем байте будут переданы остальные 8 бит адреса. Во втором байте передаются 8 бит адреса (см. рисунок 13.10).

Рисунок 13.10 – Передача данных ведомому с 10-битным адресом (для расшифровки обозначений, применяемых на рисунке, следует обратиться к таблице 13.1)

Чтобы осуществить чтение ведомого, которого адресует мастер, после второго байта адреса следует отправить бит повторного старта и затем комбинацию 1111_0xxb и бит R/W# (обозначен символом «R» – чтение), который на этот раз равен «1» (см. рисунок 13.11).

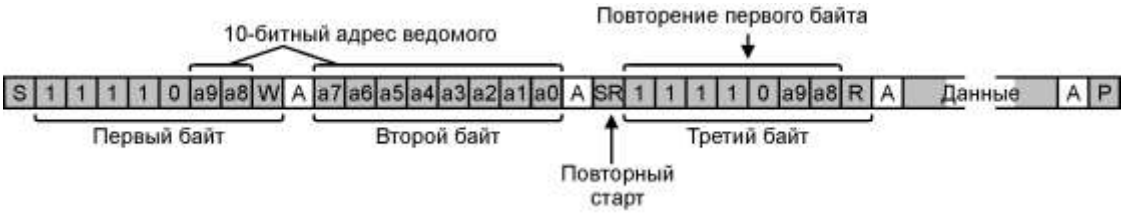


Рисунок 13.11 – Получение данных от ведомого с 10-битным адресом

На рисунках 13.10 и 13.11 биты посылки условно обозначены буквами S, W и др. или состояния битов указаны непосредственно «0» или «1». В дальнейшем на подобных рисунках, поясняющих содержимое посылки при передаче или приеме данных, будут применяться такие же и другие обозначения. Все обозначения, которые будут использоваться, указаны в таблице 13.1 с подробными пояснениями.

Таблица 13.1 – Условные обозначения, принятые на рисунках, показывающих содержимое посылок данных или адресов на линии SDA

Обозначение	Расшифровка обозначения
S	Состояние старта. Символом «S» обозначается стартовый бит посылки
SR	Состояние повторного старта
R/W	Бит указания направления передачи. В тексте настоящего описания он упоминается, как R/W#. Наличие этого обозначения в бите посылки указывает на то, что этот бит может быть равен как «0», так и «1»
R	Частный случай обозначения бита направления передачи R/W#. Если в обозначении бита стоит символ «R», то это указывает на то, что в данной посылке бит R/W# должен быть равен «1», т.е. направление передачи данных происходит от ведомого к мастеру (чтение)
W	Частный случай обозначения бита направления передачи R/W#. Если в обозначении бита стоит символ «W», то это указывает на то, что в данной посылке бит R/W# должен быть равен «0», т.е. направление передачи данных происходит от мастера к ведомому (запись)
A/A#	Бит квитированого/неквитированого приема, посылаемый приемником в ответ на запрос передатчика подтвердить прием. Наличие этого обозначения в бите посылки указывает на то, что этот бит может быть равен как «0», так и «1»
A	Частный случай обозначения бита A/A#. Если в обозначении бита стоит символ «A», то это указывает на то, что в данной посылке в ответ на запрос подтверждения приема байта произошло квитирование, т.е. приемник установил линию SDA в «0». В тексте настоящего описания квитированный бит запроса подтверждения приема обозначается как ACK
A#	Частный случай обозначения бита A/A#. Если в обозначении бита стоит символ «A#», то это указывает на то, что в данной посылке в ответ на запрос подтверждения приема байта произошло неквитирование, т.е. приемник не изменил линию SDA и оставил ее в состоянии «1». В тексте настоящего описания неквитированный бит запроса подтверждения приема обозначается как NACK
P	Состояние окончания передачи. Символом «P» обозначается стоповый бит посылки
Код мастера	8-битный код мастера. Значение 0000_1xxxb, где «xxx» – уникальный код каждого мастера в системе нескольких устройств
Адрес	7-битный адрес ведомого, передаваемый мастером
Адрес ведомого	Адрес ведомого, передаваемый во втором байте посылки. В режиме HS это 7-битный адрес, на что указывает идущий следом бит R/W#, в остальных случаях это восемь младших бит 10-битного адреса
Данные	Байт или несколько байт данных
GC	Байт адреса общего вызова (0000_0000b)
AR	Адрес отклика (0001_100b)
	Изображение числа в овале с линией, прикрепляющей его к изображению передачи битов, обозначает код операции и указывает момент, в который этот код записывается в поле MODE регистра SMBST
Цветное поле	Серым цветом обозначены биты, передаваемые от мастера к ведомому
Белое поле	Белым цветом обозначены биты, передаваемые от ведомого к мастеру

13.2 Функциональное описание

Структурная схема модуля I2C представлена на рисунке 13.12. Далее приводится краткое описание назначения блоков модуля.

Входные и выходные каскады линий SDA и SCL

Для обеих линий используются входные шумовые фильтры. В режиме FS эти фильтры подавляют любые импульсы входного сигнала, длительность которых не превышает один такт системного синхросигнала. Выходные каскады включают в себя понижающие (до уровня «0») устройства с открытым стоком. Функционирование входных и выходных каскадов зависит от состояния модуля I2C, т.е. включен или выключен.

Управление режимом работы и опрос состояния

Управление модулем осуществляют блоки управления режимом работы и скоростью передачи, регистров управления и состояния. В состав этих блоков входят следующие регистры:

- SMBST. Содержит биты, отражающие текущую конфигурацию модуля I2C (мастер или ведомый, передатчик или приемник) и бит флага прерывания;
- SMBCST. Является одновременно регистром управления шиной и регистром состояния шины;
- SMBCTRL1. Управляет генерированием состояний старта, повторного старта и останова, а также квитированием;
- SMBCTRL2 и SMBCTRL3. Устанавливают параметры тактового сигнала в режиме мастера и контролируют режим 10-битной адресации.

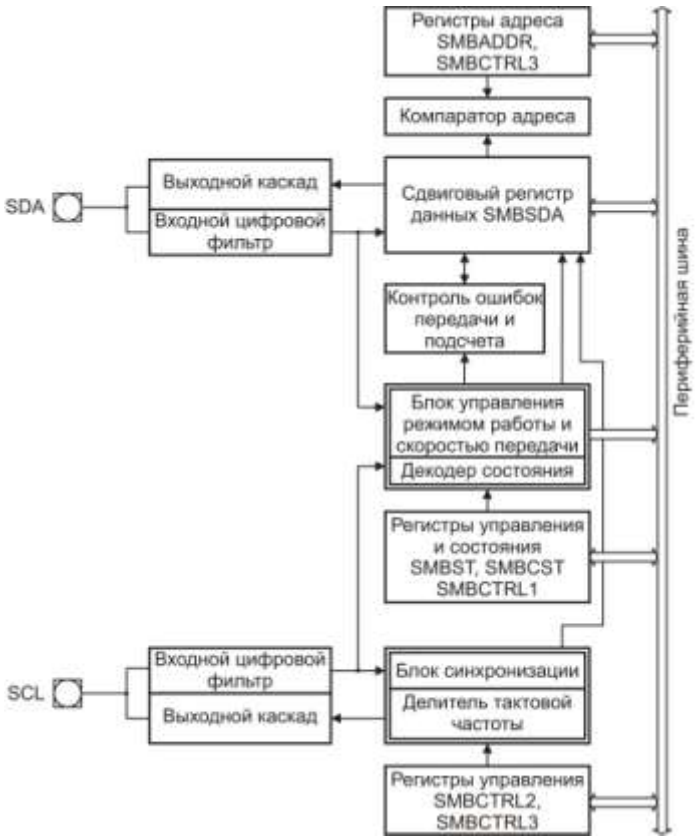


Рисунок 13.12 – Структурная схема модуля I2C

Регистры адреса и компаратор адреса

В регистр адреса SMBADDR может быть записан 7-битный адрес, который является адресом устройства при работе его в режиме ведомого. Распознавание адреса включается установкой бита SAEN.

Компаратор адреса сравнивает принятый 7-битный адрес со значением, хранящимся в поле ADDR. Если разрешено распознавание адреса общего вызова (установлен бит GCMEN регистра SMBCTRL1), то компаратор сравнивает принятый адрес со значением 0000_000b. Если разрешено распознавание адреса отклика на сигнал предупреждения (установлен бит SMBARE регистра SMBCTRL1), то компаратор сравнивает принятый адрес со значением 0001_100b.

Если включен режим 10-битной адресации (одновременно установлены биты SAEN и S10EN регистров SMBADDR и SMBCTRL3, соответственно), компаратор сравнивает старшие пять битов первого полученного байта со значением 1111_0b, а следующие два бита со значением второго и первого битов поля S10ADR регистра SMBCTRL3. Старший бит второго полученного байта сравнивается со значением нулевого бита поля S10ADR, а оставшиеся семь битов – со значением битового поля ADDR регистра SMBADDR (см. рисунок 13.13).

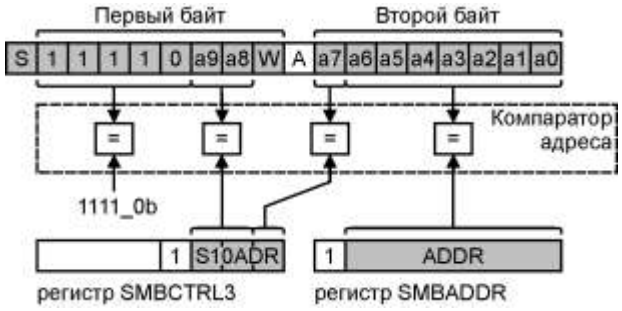


Рисунок 13.13 – Компаратор адреса в режиме 10-битной адресации

Сдвиговый регистр данных

Регистр SMBSDA представляет собой сдвиговый регистр, используемый для приема и передачи данных. Старший бит регистра передается/принимается первым, младший бит – последним. Запись в регистр SMBSDA возможна только, если установлен бит INT регистра SMBST. Регистр может быть прочитан в любой момент времени, но прочитанные данные будут гарантированно достоверными только при установленном бите INT. Регистр SMBSDA не очищается при сбросе и хранит случайные данные до тех пор, пока не будет перезаписан программно или аппаратно после приема байта.

Генерация тактового сигнала и синхронизация

Последовательный тактовый сигнал (выходной сигнал модуля I2C в режиме мастера) формируется генератором на базе системного тактового сигнала (с частотой fosc).

Модуль I2C может функционировать в двух глобальных режимах – стандартном/скоростном (FS) и высокоскоростном (HS).

В режиме FS используется 7-битный предделитель. Значение старших 6 бит определяется значением битового поле SCLFRQ регистра SMBCTRL2, а младший бит всегда равен нулю (деление производится только на четное число). Минимальный и максимальный коэффициенты деления – 8 и 128 соответственно. Блок синхронизации тактового сигнала производит синхронизацию генератора тактового сигнала и выходного сигнала SCL с тактовым сигналом других устройств, подключенных к шине.

В режиме HS используется 4-битный предделитель. Значение старших бит определяется значением битового поля HSDIV регистра SMBCTRL3, младший бит всегда равен нулю.

Определяемое спецификацией протокола SMBus наименьшее время ожидания на линии SCL составляет 25 мс. Если пауза между двумя тактовыми импульсами превысила 25 мс, то устройство должно прервать текущую передачу. Мастер должен сформировать состояние старта в процессе передачи или после ее окончания. Ведомый должен освободить шину. Устройства, обнаружившие данное состояние, должны восстановить свои соединения и ожидать формирования состояния старта в пределах 10 мс.

Для отслеживания периодов ожиданий на шине в модуле I2C имеется счетчик времени ожидания. Функциональная схема счетчика показана на рисунке 13.14.

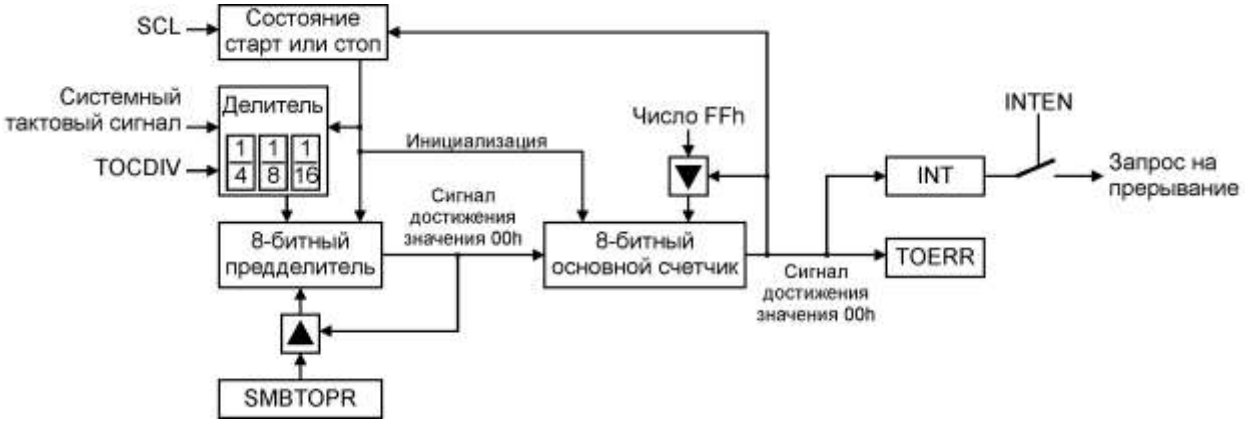


Рисунок 13.14 – Функциональная схема счетчика времени ожидания

Счетчик времени ожидания состоит из делителя, 8-битного программируемого предделителя и 8-битного основного счетчика. Все элементы счетчика времени ожидания начинают работу по отрицательному фронту сигнала на линии SCL (если работа счетчика разрешена). Положительный фронт сигнала на линии SCL сбрасывает значения делителя, предделителя и основного счетчика. Предделитель считает вниз, начиная со значения, записанного в регистр SMBTOPR. После достижения нуля счетчиком предделителя, он загружается значением из регистра SMBTOPR. Основной счетчик считает вниз от значения FFh. Каждое достижения нуля предделителем декрементирует значение основного счетчика. Обнуление основного счетчика и загрузка его значением FFh вызывает остановку основного счетчика, предделителя и делителя и установку флага TOERR в регистре SMBCST. Дополнительно устанавливается флаг INT и, если разрешено, генерируется прерывание.

Период времени ожидания определяется следующим выражением:

$$Тожид = Tosc \times TOCDIV \times (SMBTOPR + 1) \times 256, \quad (13.1)$$

где Tosc – период системного тактового сигнала с частотой fosc.

Арбитраж и обнаружение ошибок на шине

Арбитраж в режиме мастера передатчика может быть потерян в случае, когда два мастера одновременно формируют состояние старта и начинают передачу данных. Потеря арбитража может происходить как во время передачи адреса, так и во время передачи данных.

В случае потери приоритета при передаче байта адреса, мастер переходит в режим ведомого приемника и начинает принимать адрес. Если принятый адрес оказался «своим», модуль I2C далее функционирует в режиме ведомого. Если принятый адрес не оказался «своим», то модуль I2C переходит в режим безадресного ведомого.

В случае потери приоритета при передаче байта данных модуль I2C сразу переходит в режим безадресного ведомого.

Обнаружение и исправление ошибок на шине

Состояние ошибки на шине возникает в том случае, если во время передачи адреса/данных или во время квитирования на шине обнаруживаются состояния старта или стопа. При обнаружении ошибки на шине выполняются действия:

- в поле MODE регистра SMBST записывается код ошибки 1Fh;
- генерируется прерывание (если разрешено);
- модуль I2C переходит в режим безадресного ведомого;
- линии SDA и SCL освобождаются.

Обнаружение ошибки на шине может вызвать у простой шины некорректное формирование состояния старта и отключение модуля I2C. Поэтому для возврата к нормальной работе следует выполнить действия:

- выключить и снова включить модуль I2C (бит ENABLE в регистре SMBCTRL2);
- в течение времени простоя проверить, не подключен ли другой активный мастер к шине (бит BB регистра SMBCST должен быть обнулен);
- в режиме мастера шины сформировать состояние старта, передать адрес и затем сформировать состояние останова, таким образом, проведя синхронизацию всех ведомых устройств (в том числе и тех, которые не обнаружили ошибку на шине).

Режим IDLE

Переход в режим IDLE происходит при отключении внешнего сигнала тактирования модуля I2C записью нуля в бит I2CCLKEN регистра CLKREG. Переход в режим IDLE подобен программному выключению модуля I2C (очистка бита ENABLE в регистре SMBCTRL2). Регистры SMBCTRL1, SMBST и SMBCST очищаются, чтобы гарантировать нормальный старт после возобновления функционирования модуля.

Выход из режима IDLE осуществляется записью единицы в бит I2CCLKEN и включением модуля битом ENABLE.

13.3 Инициализация и функционирование

В целом модуль I2C поддерживает два базовых режима – режим FS и режим HS.

Стандартный/скоростной режим, или режим FS – стандартный режим работы, в котором модуль функционирует по умолчанию. Диапазон частот сигнала на линии SCL – от 23,6 до 750,3 кГц (при XTAL1 = 24 МГц).

Высокоскоростной режим или режим HS – режим работы, который включается программно. Режим HS значительно превосходит режим FS по скорости – диапазон частот сигнала на линии SCL от 0,25 до 2 МГц (при XTAL1 = 24 МГц).

Все операции режима HS начинаются в режиме FS в следующем порядке:

- стартовое состояние;
- 8-битный код мастера (значение 0000_1xxxb, где «xxx» – уникальный код каждого мастера в системе нескольких устройств);
- неквитирование.

Арбитраж на шине происходит в момент передачи несколькими мастерами своих уникальных кодов. Выигравший арбитраж мастер захватывает шину. В связи с такой организацией режима HS дальнейший арбитраж и синхронизация на шине не реализуются.

После выполнения вышеуказанных шагов устройства, поддерживающих режим HS, переключаются в этот режим. Мастер генерирует состояние повторного старта (SR), а затем передает адрес ведомого и бит направления передачи R/W# (см. рисунок 13.15. Расшифровка обозначений, принятых на рисунке, приведена в таблице 13.1).

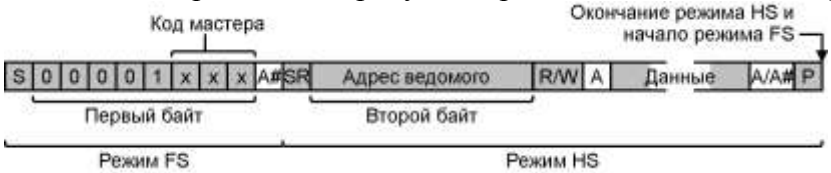


Рисунок 13.15 – Переход в режим HS и обратно в режим FS

Все передачи в режиме HS по формату идентичны передачам режима FS, что делает эти два режима полностью совместимыми.

Выход из режима HS происходит генерированием состояния окончания передачи (P), после которого все устройства переключаются обратно в режим FS.

В каждом из двух базовых режимов – FS и HS – модуль I2C может функционировать как мастер или ведомый, получать или передавать информацию.

Далее все режимы работы модуля I2C будут рассмотрены подробно.

Инициализация

Для начала работы следует произвести инициализацию:

1 Включить модуль I2C установкой бита ENABLE в регистре SMBCTRL2.

2 Если активен режим мастера, записать нужный коэффициент деления в битовое поле SCLFRQ в регистре SMBCTLR2 для выбора периода тактового сигнала SCL (для режима HS записать коэффициент деления в поле HSDIV регистра SMBCTRL3).

3 Если активен режим ведомого, необходимо:

- записать «собственный» адрес ведомого в битовое поле ADDR и установить бит SAEN регистра SMBADDR;

- для реализации 10-битной адресации записать старшие биты адреса в битовое поле S10AD и установить бит S10EN регистра SMBCTRL3;

- для включения функции распознавания адреса общего вызова установить бит GCMEN в регистре SMBCTRL1;

- для включения функции распознавания адреса отклика установить бит SMBARE в регистре SMBCTRL1.

4 При необходимости отслеживания периодов ожидания на шине записать желаемые значения в регистр SMBTOPR и в битовое поле TOCDIV (регистр SMBCST) для отсчета времени ожидания на линии SCL. Для автоматического отслеживания времени ожидания записать ненулевое значение в битовое поле TOCDIV регистра SMBCST.

5 Для разрешения формирования запроса на прерывание установить бит INTEN в регистре SMBCTRL1.

Функционирование

Модуль I2C может работать в режиме мастера или ведомого. Также он может функционировать как передатчик или приемник. Итого, модуль I2C поддерживает девять режимов:

- безадресный ведомый;
- мастер передатчик в режиме FS;
- мастер передатчик в режиме HS;
- мастер приемник в режиме FS;
- мастер приемник в режиме HS;
- ведомый передатчик в режиме FS;
- ведомый передатчик в режиме HS;
- ведомый приемник в режиме FS;
- ведомый приемник в режиме HS.

Передача информации по шине состоит из последовательности различных действий (начало передачи, прием данных и др.). Каждое действие называется состоянием (состояние старта, состояние останова и др.). После того, как то или иное состояние сформировано, его код аппаратно записывается в регистр SMBST в битовое поле MODE и может быть прочитано программно. В таблицах 13.2 и 13.3 приводятся все возможные состояния, их мнемонические обозначения и коды. На квитирование или неквитирование приема указывает запись «ACK» или «NACK», соответственно. Так, например, если мастер отправил байт адреса ведомому, который после получения квитировал прием, то на это будет указывать «ACK», а в поле MODE регистра SMBST будет записан код 04h, соответствующий состоянию с мнемоническим обозначением «MTADPA». Более

подробно каждый режим работы модуля I2C будет рассмотрен далее. На рисунках 13.16 – 13.23, поясняющих работу модуля в том или ином режиме, приняты обозначения, расшифровка которых приводится в таблице 13.1. Для получения дополнительной информации и понимания работы модуля I2C можно воспользоваться приложением В.

Таблица 13.2 – Коды функционирования модуля I2C в режиме FS

Режим		Код	Мнемоника	Описание состояния на момент записи кода в поле MODE регистра SMBST	ACK/ NACK
Общий		00h	IDLE	IDLE, нет доступной валидной информации о статусе	–
Мастер в режиме FS	–	01h	STDONE	Сформировано состояние старта	–
		02h	RSDONE	Сформировано состояние повторного старта	–
		03h	IDLARL	Потеря арбитража, переход в режим безадресного ведомого	–
	Передача	04h	MTADPA	Отправлен адрес ведомого	ACK
		05h	MTADNA	Отправлен адрес ведомого	NACK
		06h	MTDAPA	Отправлен байт данных	ACK
		07h	MTDANA	Отправлен байт данных	NACK
	Прием	08h	MRADPA	Отправлен адрес ведомого	ACK
		09h	MRADNA	Отправлен адрес ведомого	NACK
		0Ah	MRDAPA	Принят байт данных	ACK
		0Bh	MRDANA	Принят байт данных	NACK
	–	0Ch	MTMCER	Отправлен код мастера, обнаружена ошибка	ACK
Ведомый в режиме FS	Прием	10h	SRADPA	Принят адрес	ACK
		11h	SRAAPA	Принят адрес после потери арбитража	ACK
		12h	SRDAPA	Принят байт данных	ACK
		13h	SRDANA	Принят байт данных	NACK
	Передача	14h	STADPA	Принят адрес	ACK
		15h	STAAPA	Принят адрес после потери арбитража	ACK
		16h	STDAPA	Отправлен байт данных	ACK
		17h	STDANA	Отправлен байт данных	NACK
	Передача адреса отклика	18h	SATADP	Принят адрес отклика на предупреждение	ACK
		19h	SATAAP	Принят адрес отклика на предупреждение после потери арбитража	ACK
		1Ah	SATDAP	Отправлены данные в ответ на получение адреса отклика	ACK
		1Bh	SATDAN	Отправлены данные в ответ на получение адреса отклика	NACK
	–	1Ch	SSTOP	Обнаружено состояние останова ведомого	–
		1Dh	SGADPA	Принят адрес общего вызова	ACK
		1Eh	SDAAPA	Принят адрес общего вызова после потери арбитража	ACK
Общий		1Fh	BERROR	Обнаружена ошибка на шине (некорректное состояние старта или останова)	–

Примечание – Диапазон значений кодов 0Dh–0Fh зарезервирован и не доступен для использования. Дополнительная информация находится в приложении В.

Таблица 13.3 – Коды функционирования модуля I2C в режиме HS

Режим		Код	Мнемоника	Описание состояния на момент записи кода в поле MODE регистра SMBST	ACK/ NACK
Мастер в режиме HS	–	21h	HMTMCOK	Код мастера передан успешно, переход в режим HS	–
		22h	HRSDONE	Сформировано состояние повторного старта	–
		23h	HIDLARL	Потеря арбитража, переход в режим HS безадресного ведомого	–
	Передача	24h	HMTADPA	Отправлен адрес ведомого	ACK
		25h	HMTADNA	Отправлен адрес ведомого	NACK
		26h	HMTDAPA	Отправлен байт данных	ACK
		27h	HMTDANA	Отправлен байт данных	NACK
	Прием	28h	HMRADPA	Отправлен адрес ведомого	ACK
		29h	HMRADNA	Отправлен адрес ведомого	NACK
		2Ah	HMRDAPA	Принят байт данных	ACK
		2Bh	HMRDANA	Принят байт данных	NACK
Ведомый в режиме HS	Прием	30h	HSRADPA	Принят адрес	ACK
		32h	HSRDAPA	Принят байт данных	ACK
		33h	HSRDANA	Принят байт данных	NACK
	Передача	34h	HSTADPA	Принят адрес	ACK
		36h	HSTDAPA	Отправлен байт данных	ACK
		37h	HSTDANA	Отправлен байт данных	NACK
Примечание – Диапазоны значений кодов 2Ch–2Fh и 38h–3Fh, а также коды 20h, 31h и 35h зарезервированы и не доступны для использования. Дополнительная информация находится в приложении В.					

Режим безадресного ведомого

Режим работы по умолчанию (MODE = 00h). После включения модуль I2C начинает функционировать в режиме безадресного ведомого и непрерывно мониторит шину. При обнаружении состояния старта или повторного старта переходит в режим ведомого приемника. Для перехода в режим мастера передатчика нужно сформировать корректное состояние старта.

Переключение в режим безадресного ведомого происходит в случаях:

- стартовое состояние не было успешно сформировано, так как другое устройство удерживало на линии SCL низкий уровень сигнала;
- произошла потеря арбитража во время передачи байта данных в режиме мастера передатчика или во время передачи бита R/W# в режиме мастера приемника;
- произошла потеря арбитража во время ответа на полученный адрес отклика;
- неквитирование принятого адреса в режиме ведомого приемника (адрес не совпал со «своим» или запрещен);
- неквитирование в конце переданного байта в режиме ведомого передатчика;
- обнаружено состояние останова;
- обнаружена ошибка на шине;
- модуль I2C был сброшен;
- модуль I2C был выключен.

Режим FS мастера передатчика

Включение режима:

1 Переход в режим мастера передатчика происходит после успешного формирования состояния старта. Первый байт, передаваемый мастером сразу после старта, состоит из адреса ведомого и бита направления.

2 В зависимости от состояния бита направления (R/W#), модуль I2C далее функционирует как мастер передатчик (если R/W# = «0») или как мастер приемник (если R/W# = «1»). Для перехода в режим HS мастер может передать код мастера (0000_1xxxb) вместо первого байта адреса.

3 Переход в режим мастера произойдет после установки бита START в регистре SMBCTRL1. Если бит BB в регистре SMBCST сброшен, т.е. шина свободна, будет сгенерировано состояние старта. Если бит BB = 1b, то бит START останется установленным, а состояние старта будет сгенерировано по истечении времени, равного одному такту сигнала тактирования на линии SCL, после освобождения шины.

4 Как только стартовое состояние будет сгенерировано успешно, бит START сбросится, модуль I2C перейдет в состояние STDONE (в поле MODE запишется значение 01h), установится флаг INT, и линия SCL будет удерживаться в «0» до тех пор, пока флаг INT не будет сброшен. Если разрешено битом INTEN (регистр SMBCTRL1), сгенерируется прерывание.

Передача адреса и данных:

1 Пока удерживается флаг INT, программа записывает адрес ведомого и бит направления передачи в регистр данных SMBSDA (адрес записывается в биты с седьмого по первый).

2 После записи в регистр SMBSDA флаг INT сбрасывается программно установкой бита CLRST в регистре SMBCTRL1.

3 После сброса флага INT и по истечении времени, требуемого для установки данных, на линии SCL появляется тактовый сигнал и данные, хранящиеся в регистре SMBSDA, начинают передаваться по линии SDA.

4 После завершения передачи байта и получения ответа на запрос подтверждения передачи (ACK), т.е. после девятого такта сигнала тактирования на линии SCL, аппаратная часть анализирует квитирование/неквитирование передачи и устанавливает соответствующий код в поле MODE.

5 Во время передачи линии SCL и SDA постоянно мониторятся с целью выявления возможных конфликтов с другими устройствами, подключенными к шине. В случае обнаружения конфликта передача прерывается, и в поле MODE записывается код 11b (состояние SRAAPA – переход в режим ведомого приемника после потери арбитража) или код 03h (состояние IDLARL – переход в режим безадресного ведомого после потери арбитража).

6 Если бит направления равен единице и не обнаружено ошибок на шине, модуль I2C переходит в режим мастера приемника.

7 Если бит направления равен нулю и передача адреса ведомого завершена успешно (значение кода в поле MODE не равно 05h/1Fh), устанавливается флаг INT, указывая на то, что ожидается запись первого байта данных в регистр SMBSDA для дальнейшей передачи, и, если разрешено, генерируется прерывание. Пока флаг INT будет оставаться установленным, линия SCL будет удерживаться в «0».

8 Байт данных записывается программно в регистр SMBSDA, и передача продолжается.

9 Если ведомый приемник не квитирует отправленный ему байт данных, в поле MODE записывается код 0Bh (состояние MRDANA). На линии SCL будет установлен низкий уровень сигнала и, если разрешено, сгенерировано прерывание.

Для отслеживания ошибок в пакетах данных применяется механизм вычисления контрольной суммы (CRC) для нескольких байт данных. В режиме мастера передатчика установка бита PECNEXT в регистре SMBCST вызовет перенос содержимого регистра ошибок (не доступен программно) в регистр SMBSDA и инициирует передачу байта CRC (байт контрольной суммы) ведомому. Передача байта CRC должна выполняться после передачи последнего байта данных и перед формированием состояния останова или повторного старта.

Мастер передатчик контролирует шину и может адресовать любое ведомое устройство и изменять направление передачи без потери контроля над шиной, используя возможность формирования состояния повторного старта. Для формирования состояния следует:

1 Установить бит START.

2 В режиме мастера приемника прочитать последний полученный байт из регистра SMBSDA.

3 Сбросить флаг прерывания INT.

После этих действий будет освобождена линия SCL, сгенерировано состояние повторного старта и сгенерировано прерывание. В поле MODE будет записан код 02h (состояние RSDONE).

Модуль I2C может быть выведен из режима мастера передатчика генерированием состояния останова. Для этого необходимо:

1 Установить бит STOP в регистре SMBCTRL1.

2 В режиме мастера приемника прочитать последний полученный байт из регистра SMBSDA.

3 Сбросить флаг INT.

Вышеуказанные действия приведут к незамедлительному формированию состояния останова и очистке бита STOP.

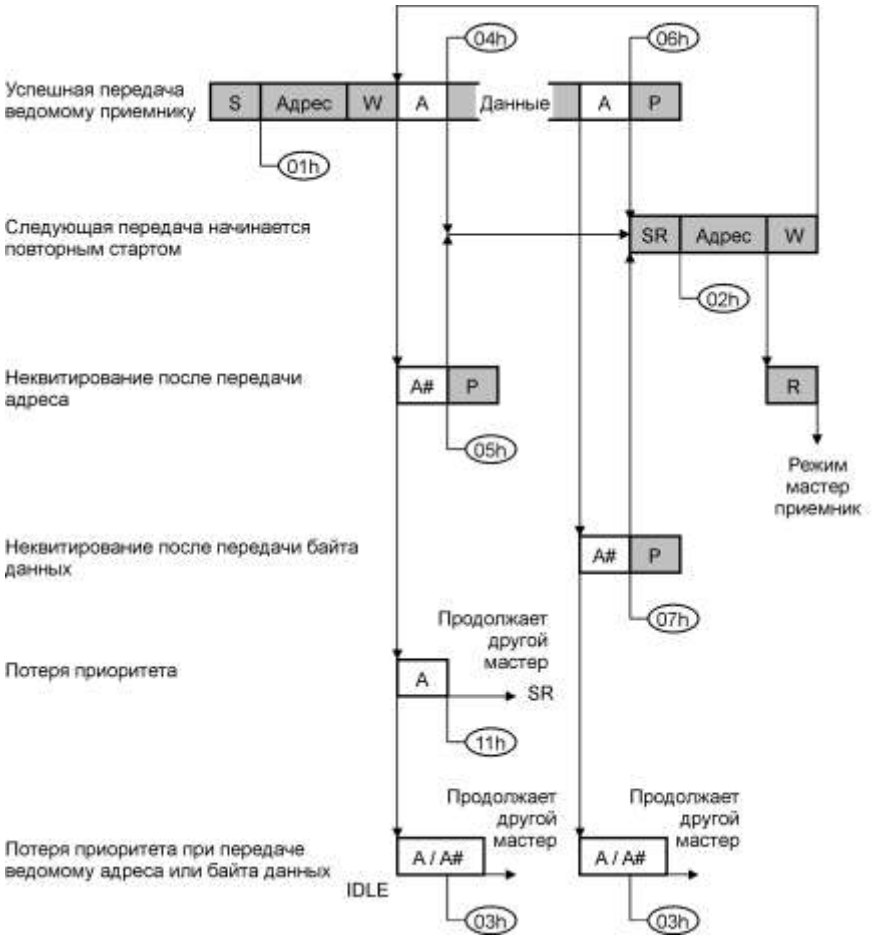


Рисунок 13.16 – Режим FS мастера передатчика

Состояние останова может быть сформировано только, если модуль I2C функционирует как мастер и контролирует шину (в поле MODE находится любое значение кода из диапазона 01h – 0Bh).

Дополнительно можно обратиться к приложению В.

На рисунке 13.16 представлено графическое пояснение к описанию режима.

Режим HS мастера передатчика

Переход в режим HS мастера передатчика происходит в том случае, если после состояния старта мастер передает код мастера (0000_1xxxb) вместо адреса ведомого. По окончании передачи кода мастера устанавливается флаг INT и, если разрешено, генерируется прерывание. Вслед за успешной передачей кода мастера в поле MODE записывается код 21h (состояние HMTMCOK), и мастер переходит в режим HS.

Далее необходимо сформировать состояние повторного старта, записав единицу в бит START и сбросить флаг INT записью единицы в бит CLRST.

После сгенерированного состояния повторного старта устанавливается флаг INT, и в поле MODE записывается код 22h (состояние HRSDONE). Дальнейший порядок действий по передаче адреса и данных аналогичен описанному режиму FS мастера передатчика.

Дополнительно можно обратиться к приложению В.

На рисунке 13.17 представлено графическое пояснение к описанию режима.

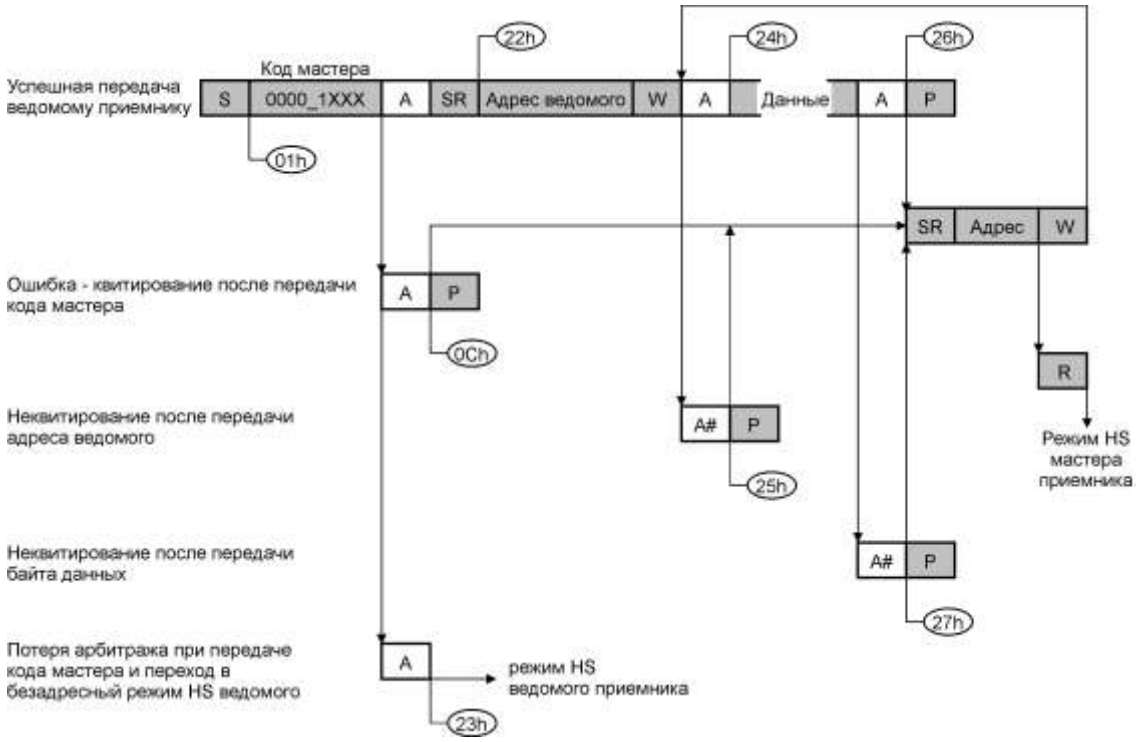


Рисунок 13.17 – Режим HS мастера передатчика

Режим FS мастера приемника

Переход в режим мастера приемника происходит после успешной передачи адреса ведомого с единичным битом направления (R/W# = «1»). В режиме мастера приемника модуль I2C получает данные от ведомого устройства, поэтому теряет контроль над шиной SDA. В тоже время мастер продолжает тактировать передачу и должен отвечать на бит ACK каждого принятого байта.

После каждого принятого байта устанавливается флаг INT, и пользовательская программа читает полученные данные из регистра SMBSDA. Линия SCL удерживается в «0», пока установлен флаг INT. После сброса флага INT может стартовать прием следующего байта. После этого (согласно протоколу SMBus) состояния повторного старта или стопа не должны генерироваться мастером, поскольку мастер теперь не является единственным контроллером линии SDA. В конце приема каждого байта мастер не квитирует прием, сообщая таким образом ведомому об успешном приеме.

После приема предпоследнего байта перед сбросом флага INT следует записать ноль в бит ACK регистра SMBCTRL1. В тоже время, если требуется отправка байта CRC, следует установить бит PECNEXT в регистре SMBCST. После сброса флага INT будет принят последний байт данных и неквитирован. По окончании приема мастер возвращается в режим передатчика и теперь может сгенерировать состояние повторного старта или останова.

Если механизм отслеживания ошибок включен, то последний переданный от ведомого байт будет байтом CRC. В случае, если результат вычисления контрольной суммы ненулевой, то установится флаг ошибки PECFAULT в регистре SMBCST.

Дополнительно можно обратиться к приложению В.

На рисунке 13.18 представлено графическое пояснение к описанию режима.

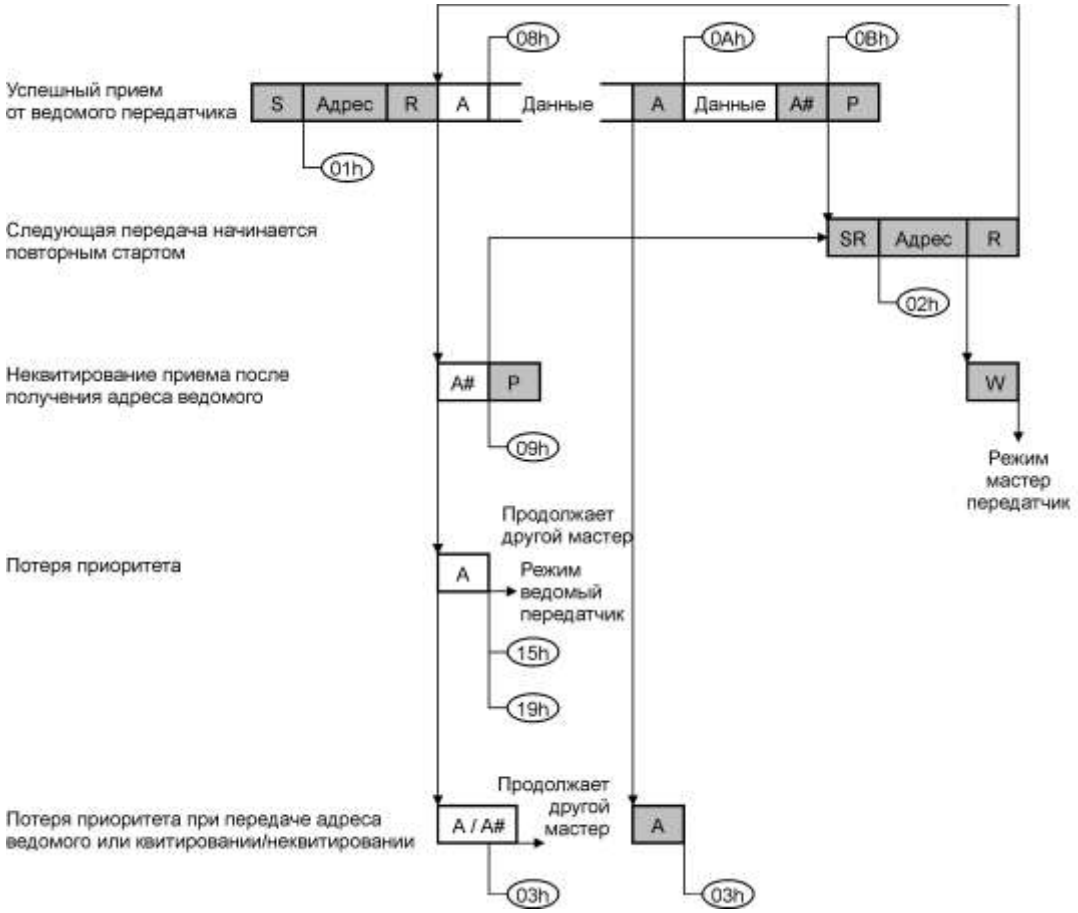


Рисунок 13.18 – Режим FS мастера приемника

Режим HS мастера приемника

Переход в режим HS мастера приемника происходит, если после переданного кода мастера и последовавшего за ним состоянием повторного старта, производится передача адреса ведомого с битом направления R/W# = «1». Модуль I2C переходит в режим HS мастера приемника, устанавливается флаг INT, а в поле MODE записывается соответствующий код из диапазона 28h – 2Bh.

Дополнительно можно обратиться к приложению В.

На рисунке 13.19 представлено графическое пояснение к описанию режима.

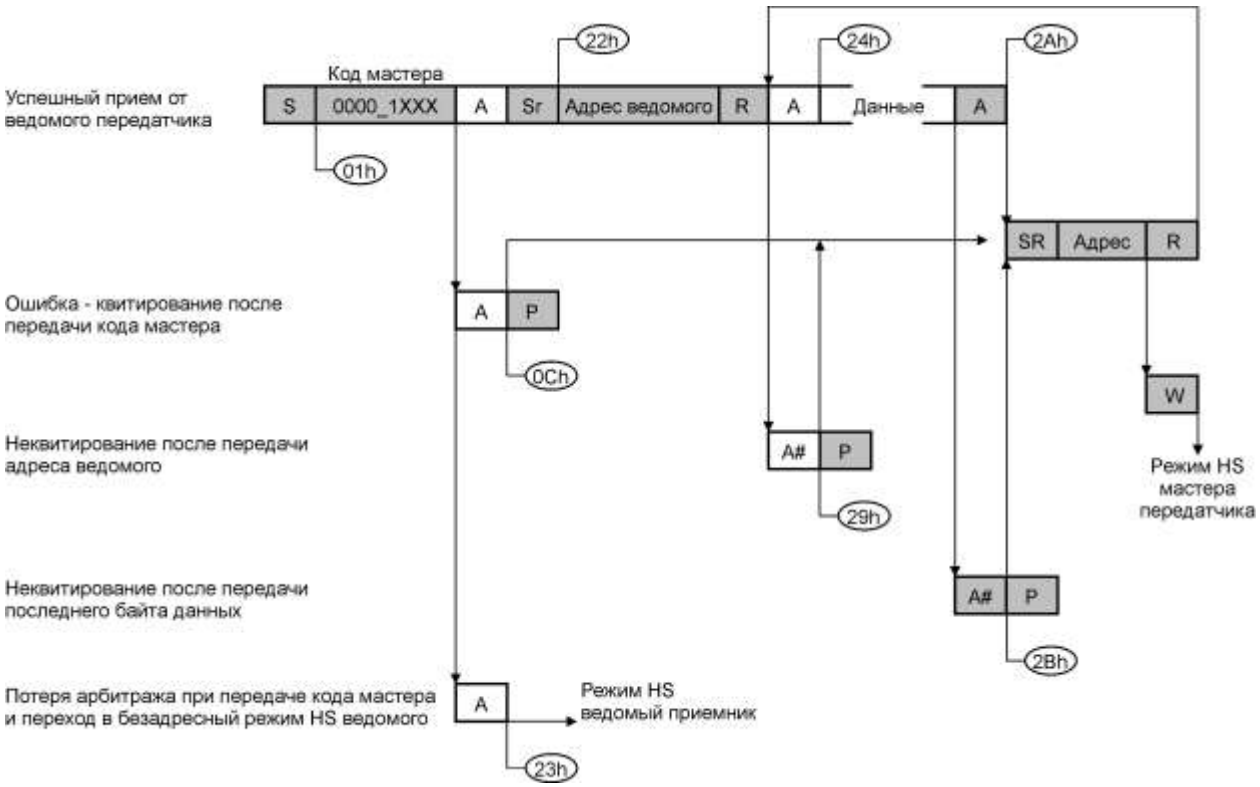


Рисунок 13.19 – Режим HS мастера приемника

Режим FS ведомого приемника

В этом режиме данные принимаются от мастера передатчика. Ведомый квитирует или не квитирует прием каждого байта.

После включения модуль I2C мониторит шину. При обнаружении состояния старта модуль I2C переключается в режим ведомого приемника и начинает принимать семь бит адреса и бит направления передачи от мастера. Мастер-передатчик может переключиться в режим ведомого приемника вследствие потери арбитража при передаче адреса.

После получения байта адреса ведомый сравнивает полученный адрес с:

- полем ADDR регистра SMBADDR, если установлен бит SAEN;
- со значением 0000_000b (адрес общего вызова), если установлен бит GCMEN;
- со значением 0001_100b (адрес отклика), если установлен бит SMBARE.

Квитирование приема производится, если принятый адрес совпал с «собственным» (запрограммированным пользователем) адресом общего вызова или адресом отклика. После обнаружения совпадения адреса и квитирования в поле MODE записывается соответствующий код и устанавливается флаг INT. Также, если разрешено битом INTEN, генерируется прерывание. Принятый байт (адрес и бит направления) переписывается в регистр SMBSDA.

В зависимости от состояния бита направления, модуль I2C переходит в режим ведомого передатчика (если R/W# = «1») или остается в режиме ведомого приемника (R/W# = «0»).

После каждого принятого байта устанавливается флаг INT, указывающий на то, что необходимо прочитать данные из регистра SMBSDA, а линия SCL удерживается в «0». После программного чтения регистра SMBSDA флаг INT сбрасывается (записью единицы в бит CLRST), и линия SCL освобождается.

Установка битов SAEN и S10EN включает режим 10-битной адресации ведомого приемника. После обнаружения состояния старта ведомый последовательно принимает два байта, в которых содержится адрес.

Последовательность передачи бит в посылке при 10-битной адресации и механизм распознавания адреса были рассмотрены выше.

После корректного приема ведомым двух байтов и совпадении принятого адреса с собственным байты сохраняются в регистре SMBSDA и сдвиговом регистре, прием квитируется, устанавливается флаг INT, а в поле MODE записывается соответствующий код состояния – 10h или 11h.

Если включен механизм обнаружения ошибок, последний байт, принятый от мастера передатчика, будет байтом CRC. Если результат вычисления контрольной суммы не нулевой, устанавливается флаг ошибки PECFAULT и передача не квитируется. Программа пользователя должна «знать» о количестве передаваемых мастером байт и устанавливать бит PECNEXT перед чтением предпоследнего байта из регистра SMBSDA и уже потом сбрасывать флаг INT. В результате будет аппаратно рассчитана контрольная сумма, и результат отправлен мастеру в момент передачи бита ACK. Если ошибок нет, будет выполнено квитирование (отправлен «0» в ответ на запрос ACK), если ошибки есть – неквитирование (отправлен «1» в ответ на запрос ACK).

Если ведомому приемнику нужно сообщить мастеру, что он не может более принимать данные, следует сначала установить бит ACK, а затем – бит CLRST (для сброса флага INT). Далее будет принят последний байт данных, который не будет квитирован (бит ACK = 1b) и установится флаг INT. После этого программа может прочитать последний полученный байт из регистра SMBSDA и сбросить флаг INT, после чего модуль I2C освободит шину.

Дополнительно можно обратиться к приложению В.

На рисунке 13.20 представлено графическое пояснение к описанию режима.

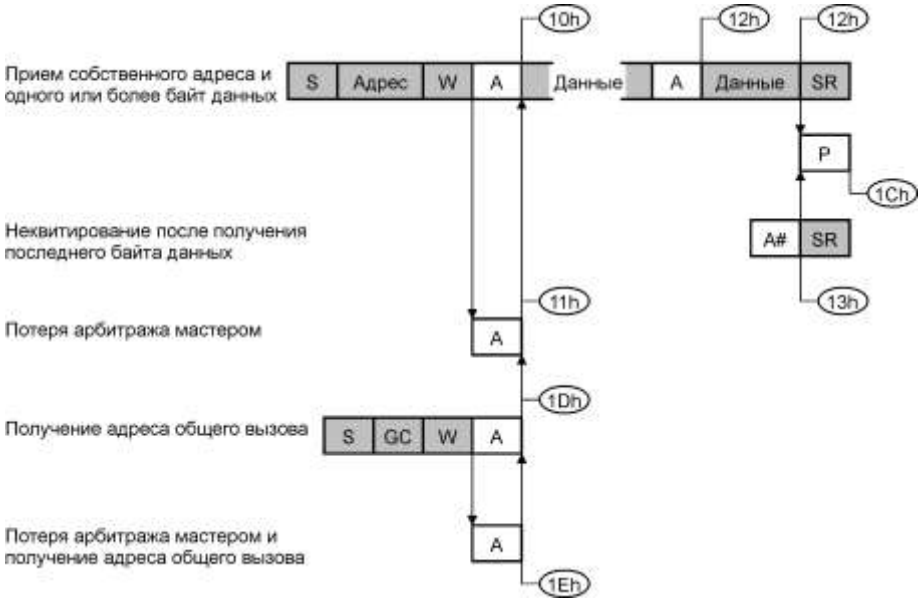


Рисунок 13.20 – Режим FS ведомого приемника

Режим HS ведомого приемника

Включение режима происходит после получения валидного кода мастера (0000_1xxxb). После передачи кода мастера формируется состояние повторного старта, а затем передается адрес ведомого с нулевым битом направления (R/W# = «0»). После получения байта адреса ведомый проверяет его на совпадение (см. ранее «Режим FS ведомого приемника»).

Дополнительно можно обратиться к приложению В.

На рисунке 13.21 представлено графическое пояснение к описанию режима.

Рисунок 13.21 – Режим HS ведомого приемника

Режим FS ведомого передатчика

В этом режиме данные передаются от ведомого передатчика к мастеру приемнику. Ведомый проверяет ответ мастера на бит ACK.

Переход в режим передатчика происходит из режима ведомого приемника. После получения собственного адреса и бита направления, равного единице (R/W# = «1»), ведомый становится передатчиком. Флаг INT устанавливается, указывая на то, что в регистр SMBSDA следует записать данные. Пока установлен флаг INT, линия SCL удерживается в «0». После записи данных в регистр SMBSDA следует сбросить флаг INT. После этого, по истечении времени, необходимого для установки данных на линии SDA, линия SCL освобождается, и данные начинают передаваться.

Передача данных аналогична передаче в режиме мастера передатчика. После каждого успешного приема байта устанавливается флаг INT, а в поле MODE записывается соответствующий код. Линия SCL удерживается в состоянии «0» до тех пор, пока флаг INT остается установленным. Флаг INT должен сбрасываться только после записи данных в регистр SMBSDA. Каждый последующий байт должен записываться в регистр SMBSDA до тех пор, пока в поле MODE не появится код 17h (состояние STDANA), указывающий на то, что мастер «не желает» далее принимать данные.

Вывод ведомого из режима передатчика осуществляется только мастером приемником. Мастер приемник должен не квитировать последний (согласно запланированному количеству) полученный байт данных. При обнаружении неквитирования переданных данных, модуль I2C переходит в режим безадресного ведомого и в поле MODE записывается код 00h (состояние IDLE). Далее ведомый мониторит шину в ожидании состояния старта или повторного старта.

Для работы в режиме с 10-битной адресацией следует осуществить действия, аналогичные описанным для режима FS ведомого приемника.

Сначала модуль I2C переходит в режим ведомого приемника и получает 10-битный адрес. Если программно не требуется никаких действий, то флаг INT не устанавливается, линия SCL не удерживается в «0» и поле MODE содержит соответствующую информацию о состоянии. Далее (см. ранее «Формат передачи данных с 10-битной адресацией»), вслед за вторым байтом адреса может последовать состояние повторного старта и затем повторная передача первого байта адреса с той лишь разницей, что бит направления содержит единицу (R/W# = «1»). Таким образом, после приема трех байт, если принятый 10-битный адрес окажется «своим», установится флаг INT и ведомый переключится в режим передатчика. В поле MODE запишется один из двух кодов – 14h или 15h.

Если включен механизм распознавания ошибок, то последний отправленный ведомым передатчиком байт будет байтом CRC. Программа должна «знать» количество

байт, посылаемых в пакете данных, и после отправки всех байт устанавливать бит PECNEXT (вместо записи очередных данных в регистр SMBSDA) для того, чтобы в регистр SMBSDA записался байт контрольной суммы.

В модуле I2C поддерживается функция распознавания адреса отклика, который передается мастером шины ко всем ведомым. Ведомое устройство, получившее адрес отклика (0001_100b), переключается в режим передатчика и начинает передавать свой собственный адрес.

Для включения функции распознавания адреса отклика следует установить бит SMBARE в регистре SMBCTRL1.

Модуль I2C реагирует на адрес отклика только при работе в режиме ведомого. В ответ на получение адреса отклика начать передачу адресов могут несколько ведомых. Ведомый, выигравший арбитраж, продолжает передачу, остальные – освобождают шину.

Дополнительно можно обратиться к приложению В.

На рисунке 13.22 представлено графическое пояснение к описанию режима.

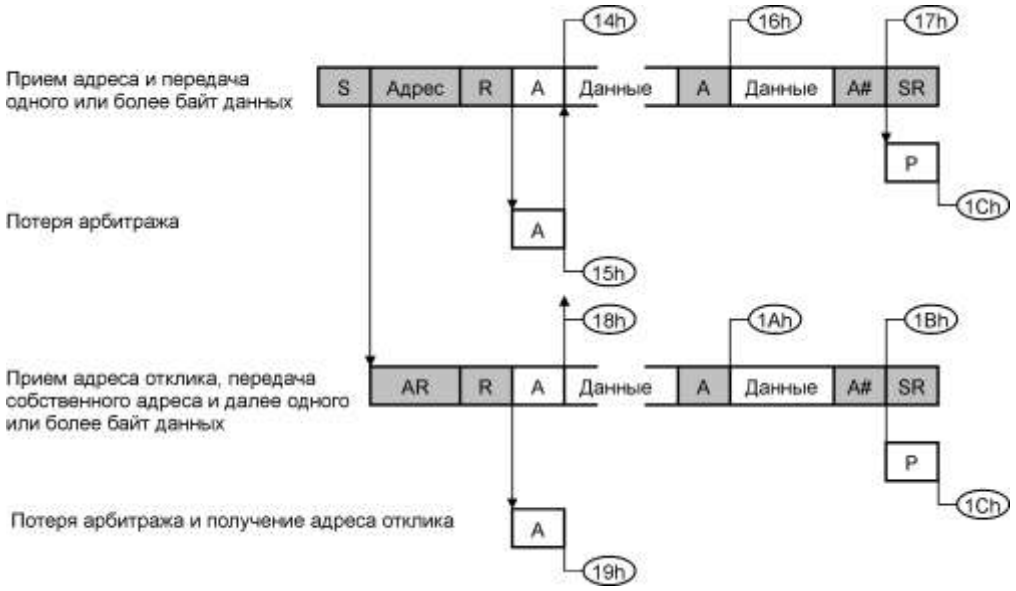


Рисунок 13.22 – Режим FS ведомого передатчика

Режим HS ведомого передатчика

Модуль I2C переходит в режим HS ведомого после получения валидного кода мастера (0000_1xxxb). Далее следует состояние повторного старта и передача адреса ведомого с единичным битом направления (R/W# = «1»). После этого ведомый переключается в режим HS ведомого передатчика. Функционирование в этом режиме в целом идентично режиму FS ведомого передатчика, с теми отличиями, что поддерживается более высокая скорость передачи, а значения кодов состояний (поле MODE) находятся в диапазоне 34h – 37h.

Дополнительно можно обратиться к приложению В.

На рисунке 13.23 представлено графическое пояснение к описанию режима.

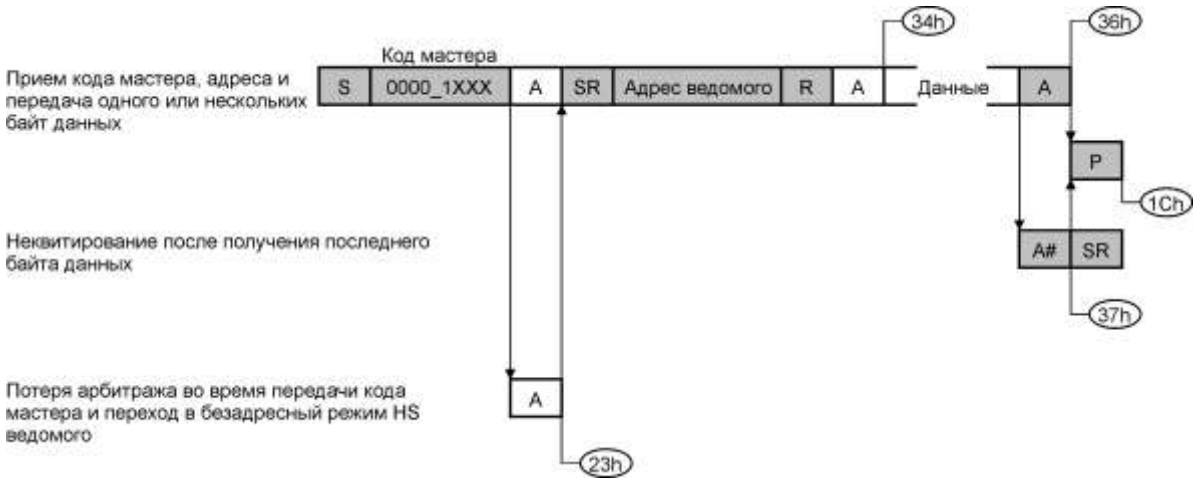


Рисунок 13.23 – Режим HS ведомого передатчика

Дополнительная информация о работе модуля

1 Когда модуль I2C выключен, бит BB регистра SMBCST очищен. Включения модуля в системе с более, чем одним мастером, может произойти в момент времени, когда по шине идет передача. Бит BB не сможет это показать. Во избежание создания ошибок на шине модуль I2C должен синхронизироваться с сигналами на шине прежде, чем сделать попытку стать мастером. Для этого следует дождаться момента, когда на шине не будет выявлена активность, т.е. периодически проверять бит BB через периоды времени, равные периоду ожидания на шине.

2 Бит BB позволяет мониторить шину и не допускать формирования ошибочных состояний старта в процессе передачи между другими устройствами на шине.

3 В некоторых случаях шина может «зависать» при активных (с нулевым уровнем) сигналах на линиях SDA и/или SCL. Источниками таких состояний могут быть необнаруженные ошибочные стартовые или стоповые состояния, сформировавшиеся в течение приема ведомых данных. Если считать, что причиной зависания явился модуль I2C, то возможны следующие два варианта развития событий:

а) если зависла линия SCL, ничего не будет происходить, а мастер, захвативший шину, должен освободить ее.

б) если зависла линия SDA, мастер должен освободить шину. Следует помнить, что в нормальном состоянии удерживать линию SCL может только текущий мастер шины. Последовательность действий для выхода из зависания следующая (при условии, что на шине только один мастер):

- выключить и включить модуль I2C для перевода его в режим безадресного ведомого;

- установить бит START для создания состояния старта;

- проверить, удерживается ли линия SDA в «0» (активное состояние) чтением бита TSDA регистра SMBCST. Если линия активна, отправить одиночный импульс по линии SCL, установив бит TGSCL в регистре SMBCST;

- проверить, что в поле MODE записан код 01b (состояние STDONE), который укажет на то, что состояние старта сформировано. Если нет, то повторять предыдущий и этот шаги до тех пор, пока линия SDA не освободится.

14 Синхронный последовательный интерфейс SPI

Последовательный периферийный интерфейс SPI предназначен для быстрого синхронного побайтного обмена информацией между микроконтроллером и периферийными устройствами или между двумя микроконтроллерами. Четырехпроводной интерфейс SPI организуется по принципу «Мастер» - «Ведомый», т.е. между одним ведущим и одним или несколькими ведомыми устройствами, не требует дополнительного оборудования для подключения SPI-совместимых устройств и обладает широкими возможностями для конфигурирования.

Модуль SPI может функционировать и как мастер, и как ведомое устройство. В режиме мастера модуль сам генерирует синхросигнал и таким образом управляет скоростью обмена информацией. Все подключенные устройства могут быть только ведомыми и должны выдавать и принимать данные синхронно. В режиме ведомого модуль ожидает появление синхросигнала и при его получении начинает выдавать и принимать данные. Модуль SPI также используется для последовательного внутрисхемного программирования микроконтроллера.

Модуль SPI имеет четыре вывода SCK, MISO, MOSI и SS#, которые объединены логически по «И» с выводами микроконтроллера P1.7, P1.6, P1.5 и P1.2 соответственно. Когда модуль выключен, он не влияет на работу выводов. В свою очередь, для нормального функционирования модуля следует записывать единицы в биты регистра IOPORT1, которые управляют указанными выводами микроконтроллера. Вывод P1.2 является входом выбора устройства SS#. Когда модуль SPI работает в режиме ведомого, подача низкого уровня сигнала на вход P1.2 активирует модуль. Как правило, управление уровнем сигнала на входе SS# осуществляется внешним мастером, который активирует ведомого для обмена данными и отключает его в остальное время. Тем не менее, установить низкий уровень сигнала на входе P1.2 можно записью нуля в соответствующий бит регистра IOPORT1. В режиме мастера состояние вывода P1.2 не оказывает влияния на работу модуля. Посредством выводов P1.7, P1.6, P1.5 и P1.2 модуль SPI может коммутироваться с внешними SPI-совместимыми устройствами.

Модуль SPI имеет два дополнительных буфера данных – один для передаваемых, а второй для принимаемых данных, что позволяет осуществлять непрерывную передачу и прием неограниченного числа байт с непрерывной синхронизацией.

Структура модуля

В состав модуля SPI входят два 8-разрядных сдвиговых регистра для передачи и приема данных, два буфера данных и регистр данных, блок синхронизации, регистры управления и блок коммутации и управления выводами. На рисунке 14.1 показана структурная схема взаимодействия блоков модуля SPI.

Синхронизирующим сигналом модуля SPI является сигнал Fspi, поступающий от коммутатора частоты синхронизации. Частота сигнала синхронизации микроконтроллера Fosc, подающегося на вывод XTAL1, аппаратно делится на два. Дополнительное влияние на частоту сигнала синхронизации Fspi оказывает бит SLOW регистра CLKC:

- если бит SLOW сброшен, то fspi = fosc/2;
- если бит SLOW установлен, то fspi = fosc/4.

Это следует учитывать при задании/расчете скорости передачи и приема данных.

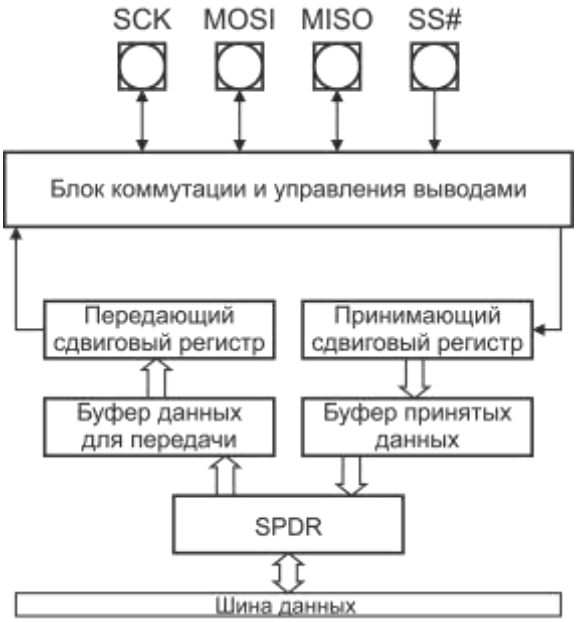


Рисунок 14.1 – Структурная схема взаимодействия блоков модуля SPI

14.1 Управляющие регистры

Управление и контроль состояния модуля SPI, загрузка и чтение полученных данных осуществляются посредством трех регистров: SPCR, SPSR и SPDR.

Регистр SPCR

По умолчанию, модуль SPI выключен. Регистр управления SPCR позволяет задать все основные параметры работы модуля.

Если бит SPIE установлен, то по окончании передачи каждого байта будет генерироваться запрос на прерывание с вектором SERPORT (INT06). Бит SER_MASK регистра INT_MASK является маскирующим битом прерывания. Бит SER_PEND регистра INT_PEND является индикатором сформированного запроса на прерывание.

Бит SPE является битом включения модуля SPI. Этот бит может быть сброшен в любой момент, что приведет к немедленному выключению модуля. Состояние всех битов регистра SPCR должно быть задано одновременно с установкой бита SPE. Если в процессе работы требуется изменить состояние какого-либо бита регистра SPCR (т.е. изменить конфигурацию модуля SPI), то сначала обязательно (!) нужно сбросить бит SPE.

Бит DORD задает порядок передачи и приема битов при обмене байтами. По умолчанию, передача и прием каждого байта осуществляется старшим битом вперед. Установка бита DORD меняет порядок передачи и приема на противоположный – младшим битом вперед.

Бит MSTR позволяет конфигурировать модуль SPI на работу в режиме мастера. По умолчанию, включен режим ведомого.

Бит CPOL задает полярность тактового сигнала. Если бит сброшен, то в отсутствие передачи на линии SCK мастером удерживается низкий уровень сигнала, в противном случае – высокий (см. рисунок 14.2).

Передним всегда считается тот фронт сигнала, который является началом передачи первого бита (т.е. передним может быть и положительный, и отрицательный фронт).

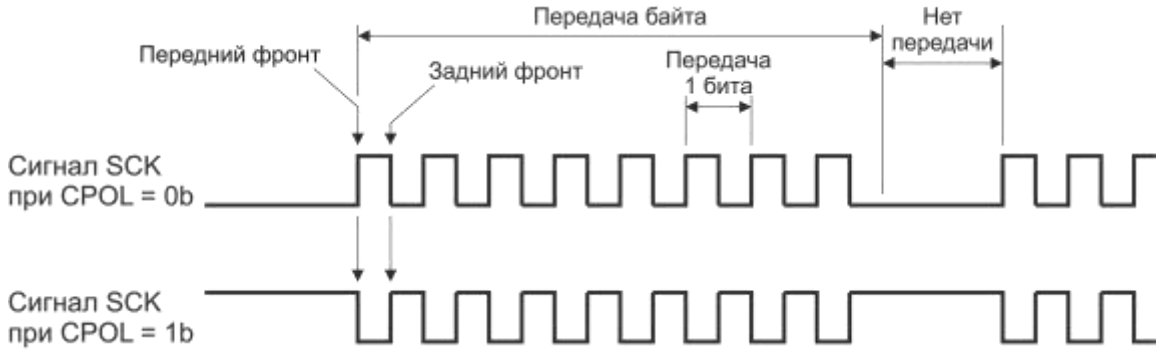


Рисунок 14.2 – Сигнал синхронизации SCK при разных состояниях бита CPOL

Бит CPHA задает порядок считывания и выставления данных. По умолчанию бит CPHA установлен, и выставление данных на линиях MISO и MOSI происходит по переднему фронту сигнала синхронизации, а считывание (выборка) – по заднему. Сброс бита CPHA меняет порядок на обратный. Комбинации битов CPOL и CPHA задают четыре режима обмена данными (см. рисунок 14.3).

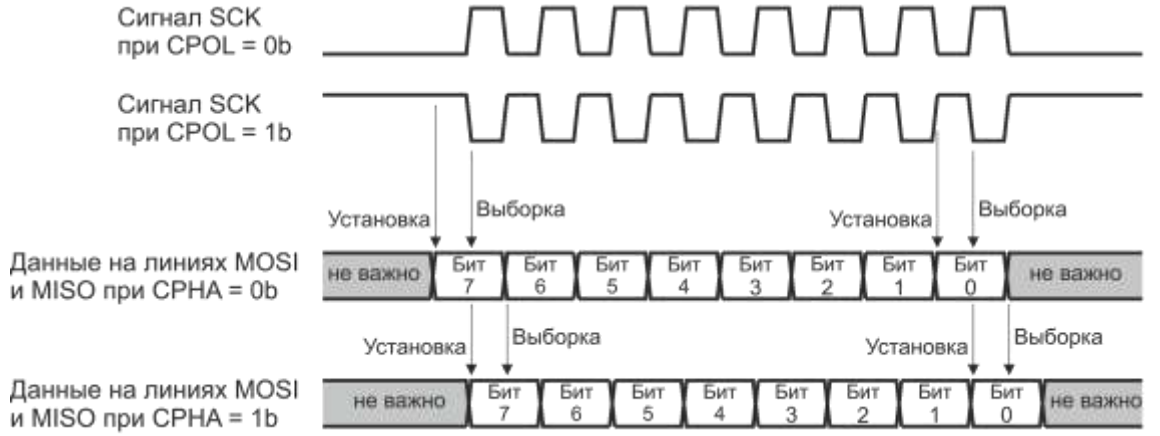


Рисунок 14.3 – Временные диаграммы передач данных в разных режимах

По умолчанию данные передаются старшим битом вперед. Установка и считывание данных выполняются мастером и ведомым одновременно (поэтому на рисунке 14.3 линии MOSI и MISO условно объединены). При CPHA = 0b установка первого бита на линию передачи данных происходит до появления переднего фронта сигнала SCK. К моменту появления переднего фронта сигнала SCK данные должны быть стабильными, чтобы мастер и ведомый смогли произвести корректную выборку. При CPHA = 0b выборка данных производится по переднему фронту сигнала синхронизации, а установка – по заднему. При CPHA = 1b установка первого бита (и последующих бит) на линию передачи данных происходит только тогда, когда появляется передний фронта сигнала SCK. Выборка данных происходит по заднему фронту. Во всех режимах до момента установки первого бита на линию передачи данных и после последней (восьмой) выборки не важно, какой уровень сигнала удерживается на линии.

Оставшиеся два бита SPR1 и SPR0 регистра задают скорость передачи данных. Если модуль SPI функционирует в режиме мастера, то именно он формирует синхросигнал SCK для передачи и приема данных. Синхросигнал формируется на основе сигнала тактовой частоты Fspi. Четыре комбинации состояний битов SPR1 и SPR0 позволяют программировать четыре скорости передачи данных (см. таблицу 14.1).

Таблица 14.1 – Комбинации битов SPR1, SPR0

SPR1	SPR0	Частота синхросигнала SCK	
		При SLOW = 0b	При SLOW = 1b
0	0	fspi/4	fspi/2
0	1	fspi/16	fspi/8
1	0	fspi/64	fspi/32
1	1	fspi/128	fspi/64

Если модуль SPI функционирует в режиме ведомого, то состояния битов SPR1 и SPR0 игнорируются. Ведомый передает данные на частоте сигнала SCK, который задается внешним мастером, при этом частота входного сигнала синхронизации fsck не должна превышать частоту fspi более, чем в четыре раза.

Регистр SPDR

Регистр данных SPDR позволяет загружать данные для передачи и считывать принятые данные (обращение по одному адресу). В режиме мастера запись в регистр SPDR автоматически инициализирует передачу и прием байта. В режиме ведомого запись в SPDR только подготавливает данные для передачи и должна производиться до начала передачи, т.е. до появления переднего фронта синхросигнала SCK.

В нормальном режиме работы данные передаются побайтно. И в режиме мастера, и в режиме ведомого запись очередного байта в регистр SPDR во время передачи игнорируется, передача не прерывается, но выставляется флаг WCOL, который является индикатором конфликта процессов записи и передачи.

В режиме буферизации данные могут передаваться непрерывным потоком с непрерывной синхронизацией. Режим включается установкой бита ENH. И в режиме мастера, и в режиме ведомого запись очередного байта в регистр SPDR во время передачи приводит к загрузке этого байта в буфер, передача не прерывается, и выставляется флаг WCOL, который в данном режиме является индикатором того, что очередной байт загружен и готов для передачи. Если по окончании передачи байта обнаруживается установленный флаг WCOL, то байт из буфера переносится в передающий регистр и передача данных продолжается, а флаг WCOL сбрасывается. Буфер может хранить только один байт данных – повторная запись приведет к потере ранее записанного байта.

В обоих режимах (нормальном и буферизации) по окончании передачи и приема каждого байта устанавливается флаг SPIF и формируется запрос на прерывание (если установлен бит SPIE регистра SPCR). Каждый принятый байт сохраняется и доступен для чтения посредством регистра SPDR до тех пор, пока не будет перезаписан очередным принятым байтом.

Регистр SPSR

По умолчанию модуль SPI функционирует в нормальном режиме, и данные передаются побайтно. Для включения режима буферизации следует установить бит ENH.

Бит SPIF является флагом окончания передачи и приема и функционирует, если установлен бит SPIE регистра SPCR. В обоих режимах (нормальном и буферизации) флаг SPIF устанавливается по окончании передачи и приема каждого байта. Для сброса флага необходимо выполнить последовательно действия – прочитать регистр SPSR, а затем прочитать/записать регистр SPDR. При этом между командой чтения регистра SPSR и командой чтения/записи регистра SPDR допускается размещение других команд. При выключении модуля SPI сбросом бита SPE флаг SPIF не сбрасывается. При включении модуля флаг SPIF всегда сбрасывается аппаратно.

Бит WCOL является флагом записи в регистр SPDR во время передачи байта. В нормальном режиме работы флаг сбрасывается одновременно с флагом SPIF. В режиме буферизации данных флаг сбрасывается одновременно с загрузкой очередного байта из буфера в передающий регистр. При выключении модуля SPI флаг WCOL сбрасывается аппаратно.

Бит LDEN является битом разрешения загрузки буфера в режиме буферизации данных. Во время каждого обмена байтами бит LDEN установлен в течение передачи первых четырех бит, а затем сбрасывается. Наиболее подходящее время для записи очередного байта для передачи – это время, когда бит LDEN установлен.

На состояние информационных битов SPIF, WCOL и LDEN оказывают влияния только аппаратные процессы. Чтение регистра SPSR не модифицирует эти биты.

Дополнительным битом управления передачей данных является бит DISSO. В режиме ведомого установка этого бита переводит выход MISO в высокоимпедансное состояние. Таким образом, модуль SPI может принимать данные, но ответной передачи не происходит. Это может использоваться в системах с одним мастером и несколькими параллельно соединенными и постоянно активными ведомыми (каждый со своим адресом). Выходы MISO всех ведомых заблокированы. Мастер передает байт адреса, все ведомые принимают его и к началу следующей передачи ведомый, распознавший свой адрес, сбрасывает бит DISSO, тем самым разрешая передачу данных к мастеру через вывод MISO. Регистр SPSR всегда доступен для записи, и в связи с этим бит DISSO может быть установлен/сброшен в любой момент.

14.2 Особенности функционирования

1 Мастер всегда должен включаться (битом SPE) раньше ведомого. Особенно это важно, если вход SS# ведомого заземлен (т.е. ведомый активен сразу после включения).

2 Деактивация ведомого вследствие появления высокого уровня сигнала на линии SS# сразу останавливает прием и передачу данных ведомым устройством, но не сбрасывает его передающие регистры. Для корректного продолжения работы следует вновь записать данные в регистр SPDR ведомого и только потом активировать его.

3 При выключении модуля SPI текущая передача прерывается, сдвиговые регистры обнуляются.

15 Контроллер интерфейса LIN

LIN (Local Interconnect Network) представляет собой последовательный коммуникационный протокол, предназначенный для создания локальных сетей обмена данными на коротких расстояниях. Сеть строится из единственного ведущего устройства (мастера) и множества ведомых устройств. Мастер контролирует все процессы, связанные с передачей данных (сообщений) по шине. Ведомые могут реагировать на сообщения мастера или других узлов сети, но отвечать они могут только будучи адресованными мастером. Шина LIN состоит из одного канала, по которому передаются синхросигналы и данные. Физическая среда канала представляет собой однопроводную линию, подключенную через подтягивающий резистор к шине питания. Высокий уровень сигнала на шине является рецессивным и показывает, что шина свободна. Низкий уровень сигнала на шине является доминантным и показывает, что шина занята.

Кадр сообщения

Вся информация, передаваемая по шине LIN, разделяется на кадры. Кадр сообщения состоит из двух частей – заголовка, посылаемого мастером, и ответа, который может формироваться как мастером, так и ведомым (см. рисунок 15.1).

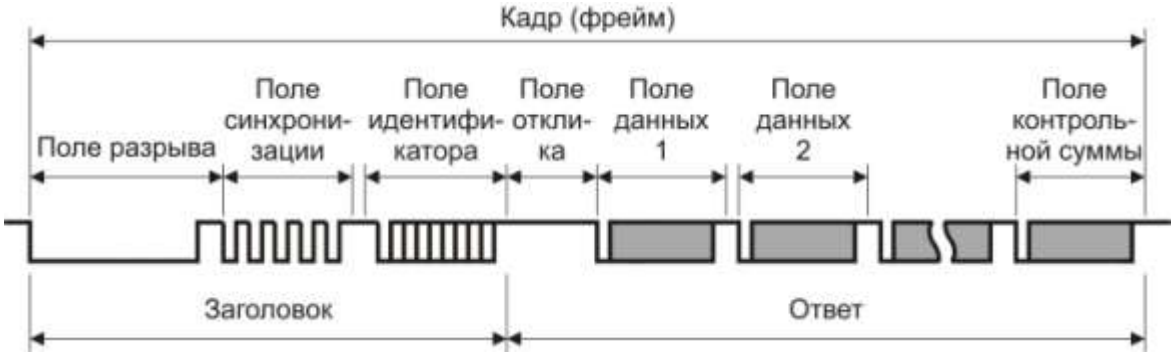


Рисунок 15.1 – Кадр сообщения

Заголовок состоит из:
- поля разрыва;
- поля синхронизации;
- поля идентификатора.

Ответ состоит из:
- поля отклика;
- полей данных;
- поля контрольной суммы.

Заголовок начинается с первого доминантного бита поля разрыва синхронизации и заканчивается со стоповым битом поля PID. Часть сообщения, начинающаяся со стопового бита PID и до стопового бита контрольной суммы, является ответом.

15.1 Байтовые поля

Каждое байтовое поле содержит 10 бит – старт бит, 8 бит данных, стоп (см. рисунок 15.2). Старт-бит указывает на начало байтового поля и является доминантным, в то время как стоп-бит является рецессивным. Восемь информационных битов могут быть как доминантными, так и рецессивными.

Рисунок 15.2 – Байтовое поле

Поле разрыва синхронизации указывает на начало передачи кадра сообщения. Это поле всегда передается мастером и указывает на необходимость ведомым подготовиться к приему поля синхронизации. Поле разрыва синхронизации состоит из двух частей – низкого уровня, длительность которого должна быть не менее 13 битов, и высокого уровня, длительностью не менее одного бита.

Длительность первой части выбрана таким образом, чтобы обеспечить различие между полем разрыва синхронизации и максимально возможной допустимой последовательностью нулевых битов в пределах одного кадра данных. Для ведомых устройств длительность поля разрыва может быть снижена до 9,5 бит.

Вторая часть поля необходима для обеспечения возможности обнаружения стартового бита следующего поля синхронизации (см. рисунок 15.3).

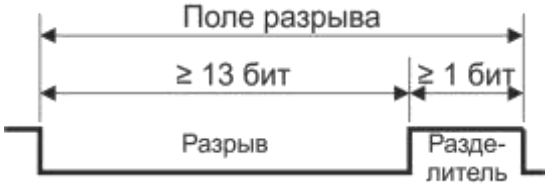


Рисунок 15.3 – Поле разрыва синхронизации

Поле синхронизации содержит сигналы, необходимые для синхронизации задающих генераторов ведомых устройств сети. Поле синхронизации является байтовым полем, имеющем значение 55h (см. рисунок 15.4).

Рисунок 15.4 – Поле синхронизации

Поле синхронизации содержит пять спадающих фронтов (пять переходов от рецессивного состояния к доминантному). Эти спадающие фронты должны использоваться ведомыми устройствами для настройки скорости обмена данными.

Контроллер интерфейса LIN может работать с идентификаторами (PID) как новой версии протокола 2.X, так и версией 1.X. Основное отличие в том, что в ранних версиях протокола PID использовался в том числе и для контроля количества полей данных в кадре сообщения. В версии 2.X количество полей данных определяется пользователем и не включено в PID.

В версии протокола 1.X в идентификационном поле находится информация о содержимом и длине сообщения. Это поле разделено на три секции: четыре идентификационных бита, два служебных бита, указывающих на длину сообщения и два бита проверки на четность (см. рисунок 15.5). Таким образом, происходит деление 64 идентификационных номеров на четыре комплекта, каждый из которых содержит 16 идентификаторов.

Рисунок 15.5 – Идентификационное поле в версии 1.X

В соответствии с протоколом 1.X количество полей данных в кадре данных определяется по таблице 15.1.

Таблица 15.1 – Количество полей данных в кадре данных

ID4	ID5	Количество полей данных
0	0	2
1	0	
0	1	4
1	1	8

Следует заметить, что в идентификационном поле не указывается длина сообщения, а только содержание. Это позволяет ведомым приемникам определить, все ли данные были приняты. Последние два бита идентификационного поля являются битами проверки на четность. Контроллер LIN использует алгоритм смешанного контроля четности, который гарантирует, что идентификационное поле никогда не будет состоять из одних нулей или единиц. Этот алгоритм позволяет только обнаруживать ошибки, но не исправляет их.

Биты проверки на четность формируются в соответствии с приведенными ниже формулами:

P0 = ID0 XOR ID1 XOR ID2 XOR ID4

P1 = NOT (ID1 XOR ID3 XOR ID4 XOR ID5)

В версии протокола 2.X идентификационное поле содержит только две секции – шесть идентификационных бит и два бита четности (см. рисунок 15.6). Четность вычисляется так же, как и в версии 1.X. Количество полей данных в кадре сообщения устанавливается пользователем. Сделать это можно посредством поля DATALEN регистра LINCON.

Рисунок 15.6 – Идентификационное поле в версии 2.X

В кадре сообщения может быть от одного до восьми полей данных. Их количество должно быть согласовано для всей сети. Байты данных передаются как часть поля данных. Передача данных осуществляется младшим значащим битом вперед (см. рисунок 15.7).

Рисунок 15.7 – Поле данных

Последним полем в кадре сообщения является поле контрольной суммы (см. рисунок 15.8).

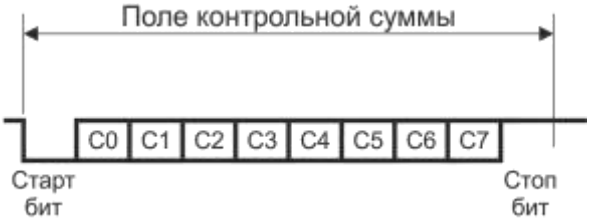


Рисунок 15.8 – Поле контрольной суммы

Контрольная сумма представляет собой инвертированную сумму по модулю 256 всех байт. Контроллер LIN может вычислять как обычную контрольную сумму (только для байт данных), так и расширенную – для байт данных и идентификатора.

Вычисление контрольной суммы представляет собой последовательное сложение элементов, а при превышении на очередном шаге значения 256 от промежуточной суммы отнимается 255. Конечный результат инвертируется. Свойства инвертированной суммы по модулю 256 таково, что если ее сложить с суммой всех байт, то результат будет равен FFh.

15.2 Режим сна

Мастер сети может быть инициатором вхождения в режим сна. Кадр сна представляет собой кадр сообщения, в котором первый байт данных равен 00h, а все другие – FFh. Ведомые устройства входят в режим сна автоматически или по сигналу от мастера. Запрос выхода из режима сна может послать любое устройство сети. Данный запрос представляет собой 5 доминантных бит (см. рисунок 15.9).

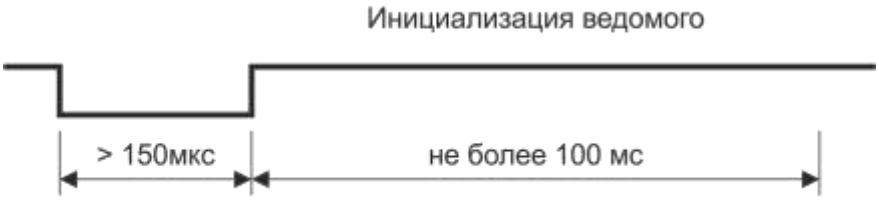


Рисунок 15.9 – Запрос выхода из сна

15.3 Регистры управления и контроля

Все регистры контроллера LIN доступны по адресам 1F40h – 1F5Eh.

Контроллеру LIN принадлежат 8 регистров данных. Четыре регистра, расположенные по адресам 1F40h – 1F46h, отведены под восемь байт данных, которые будут передаваться в кадре сообщения.

Количество отправляемых данных задается полем DATALEN регистра LINCON. Неиспользуемые регистры можно оставлять незаполненными.

Четыре регистра с адресами 1F48h – 1F4Eh отведены под восемь байт данных, которые будут получены. Количество принимаемых байт определяется полем PID кадра сообщения или полем DATALEN.

Старший байт регистра LINCON хранит информацию о количестве байт данных, передаваемых в одном кадре сообщения. Заполнять поле DATALEN необходимо только при использовании протокола версии 2.X. Младший байт содержит настройки протокола обмена и тип текущего кадра сообщения.

Поскольку в версии протокола 1.X число байт данных содержится в PID, а в версии протокола 2.X – задается полем DATALEN, следует правильно задавать состояние бита VER.

С учетом типа устройства – мастер или ведомый – поле MSGTYPE определяет режим работы – передача или прием (см. таблицу 15.2).

Таблица 15.2 – Тип кадра сообщения и соответствующие действия устройства

<table>
<tr><td rowspan="2">Тип кадра сообщения, заданный полем MSGTYPE</td><td colspan="2">Действия, в зависимости от типа устройства</td></tr>
<tr><td>Мастер</td><td>Ведомый</td></tr>
<tr><td>BUSWAIT</td><td colspan="2">Неактивен</td></tr>
<tr><td>HEADER</td><td rowspan="2">Передача</td><td rowspan="2">Прием</td></tr>
<tr><td>HEADER+RESPONSE</td></tr>
<tr><td>RESPONSE RECEIVE</td><td>Прием</td><td>Прием</td></tr>
<tr><td>RESPONSE TRANSMIT</td><td>Не влияет</td><td>Передача ответа</td></tr>
<tr><td>WAKE</td><td colspan="2">Передача запроса на пробуждение</td></tr>
<tr><td colspan="3">Примечание – Примененные в таблице типы означают: HEADER – в кадре передаются поля разрыва, синхронизации и идентификатора; RESPONSE – в кадре передаются поля данных и контрольной суммы.</td></tr>
</table>

В регистре состояния LINSTAT находятся биты, несущие информацию о действиях контроллера LIN, его текущем состоянии и ошибках передачи. Флаги регистра LINSTAT маскируются битами регистра разрешений прерываний LININTEN. По умолчанию состояние регистра разрешений прерываний 0000h и поэтому флаги не выставляются. Если установка того или иного флага разрешена, то одновременно с установкой генерируется прерывание. Установленные флаги не сбрасываются аппаратно. Для того, чтобы сбросить бит (флаг) в регистре LINSTAT, следует в этот бит записать единицу.

Биты TR, REC и FRAMESTAT не маскируются – устанавливаются и сбрасываются аппаратно. Биты TREN и RECEN управляют только генерированием прерываний. Поле FRAMESTATEN задает код режима, при входе в который будет генерироваться прерывание и выставляться соответствующий код режима в поле FRAMESTAT регистра LINSTAT.

Флаг TREND устанавливается, когда устройство в режиме передачи заканчивает передавать текущий кадр сообщения. Если передается HEADER, то флаг будет установлен после стоп-бита поля PID. Если передается RESPONSE или полный кадр сообщения, то сигнал будет выработан после стопового бита поля контрольной суммы.

Биты PIDREC, DATAREC и CSRREC устанавливаются, когда устройство принимает одно из соответствующих полей. После приема поля PID необходимо решить – адресован ли принятый кадр сообщения данному устройству или следует ожидать следующий кадр. После приема поля контрольной суммы можно начинать обработку принятых данных.

Можно выполнять обработку данных и после каждого принятого поля данных, не дожидаясь контрольной суммы, но в этом случае есть риск столкнуться с ситуацией, когда контрольная сумма не совпадет с принятой, а данные уже будут в обработке.

Регистр настройки длины поля разрыва синхронизации LINBREAK используется, если устройство в сети является ведомым. В регистре содержится 16-разрядное значение минимальной длины поля разрыва синхронизации, выраженное в периодах тактового сигнала. Значение вычисляется по формуле:

$$N = \frac{Fxtal}{2 \cdot Flin} \times \frac{Nbit}{16}$$

где Fxtal – тактовая частота микроконтроллера,
Flin – скорость передачи данных блоком LIN,
Nbit – минимальное количество бит в поле разрыва синхронизации.

Пример. Требуется рассчитать длину поля для микроконтроллера с частотой 20 МГц, скоростью LIN 20 Кб/с и стандартной длиной поля разрыва в 13 бит.

$$N = \frac{20 \times 10^6}{2 \times 20 \times 10^3} \times \frac{13}{16}$$

Результат равен 406. После перевода в шестнадцатеричный формат получается значение 0196h, после чего в регистр LINSPEED записывается вычисленное значение.

Если отклонение частоты велико, можно занижать длительность поля разрыва синхронизации, обеспечивая более уверенное его детектирование.

Регистр настройки скорости передачи LINSPEED используется, если устройство в сети является мастером. 16-разрядное значение вычисляется по формуле:

$$N = \frac{Fxtal}{4 \cdot Flin}$$

Так, для микроконтроллера с частотой 24 МГц и скоростью LIN 20 Кб/с рассчитанное значение равно 012Ch.

Регистр LINPIDTX передаваемого PID используется для записи значения идентификатора, который будет отправлен в следующем кадре сообщения. Загружаемое значение не зависит от версии протокола и типа контрольной суммы. При необходимости два неиспользованных разряда будут автоматически заменены на биты четности при отправке.

Регистр LINPIDRX принятого PID используется для считывания полученного значения идентификатора. Обновление регистра происходит одновременно с установкой флага PIDREC. Биты четности в данный регистр не записываются, но в случае нарушения четности будет установлен флаг PIDERR.

Регистр контрольной суммы LINCSR используется для считывания полученного значения контрольной суммы. Обновление регистра происходит одновременно с установкой флага CSRREC. Если значение контрольной суммы, вычисленное на основе полученных данных, не совпадает с принятой контрольной суммой, то устанавливается флаг CSRERR.

15.4 Программирование

Пример организации передачи главным устройством полного кадра сообщения (HEADER + RESPONSE). Тип контрольной суммы – расширенная, версия протокола 2.X, 4 байта данных в ответе, частота контроллера – 38,4 МГц, скорость LIN – 20 Кбит/с.

```
; установка скорости
LD AX, #001E0H
LD DX, #LINSPEED
ST AX, [DX]

; установка длительности поля BREAK
LD AX, #0030DH
LD DX, #LINBREAK
ST AX, [DX]

; настройка блока LIN 20 Кбит/с, 2.X, ENHCSR=1
LD AX, #07401H
LD DX, #LINCON
ST AX, [DX]

; заполнение 4 байт
LD AX, #BYTES10
LD DX, #LINDTX10
ST AX, [DX]+
LD AX, #BYTES32
ST AX, [DX]

; установка значения PID
LDB AX, #029H
LD DX, #LINPIDTX
STB AX, [DX]

; выполнение посылки HEADER + RESPONSE
LDB AX, #003H
LD DX, #LINCON
STB AX, [DX]
```

16 Контроллер интерфейса CAN

Последовательный интерфейс CAN (Controller Area Network) – интерфейс связи, эффективно поддерживающий распределенное управление в реальном масштабе времени с высокой помехозащищенностью. Протокол связи определен в спецификации CAN 2.0B.

Протокол CAN оптимизирован для систем, в которых должно передаваться относительно небольшое количество информации (по сравнению с Ethernet или USB) к любому или всем узлам сети. Множественный доступ с опросом состояния шины позволяет каждому узлу получить доступ к шине с учетом приоритетов. Неадресная структура сообщений позволяет организовать многоабонентскую доставку данных с сокращением трафика шины. Быстрая устойчивая передача информации с системой контроля ошибок позволяет отключать неисправные узлы от шины, что гарантирует доставку критических по времени сообщений.

Область применения CAN протокола: от высокоскоростных сетей связи до электропроводов в автомобиле. Высокая скорость передачи данных (до 1 Мбит/с), хорошая помехозащищенность протокола, защита от неисправности узлов – делают шину CAN подходящей для индустриальных приложений управления типа DeviceNet.

CAN имеет асинхронную последовательную структуру шины с одним логическим сегментом сети. CAN сеть может состоять из двух или более узлов с возможностью подключения/отключения узлов от шины без перенастройки других устройств (см. рисунок 16.1).

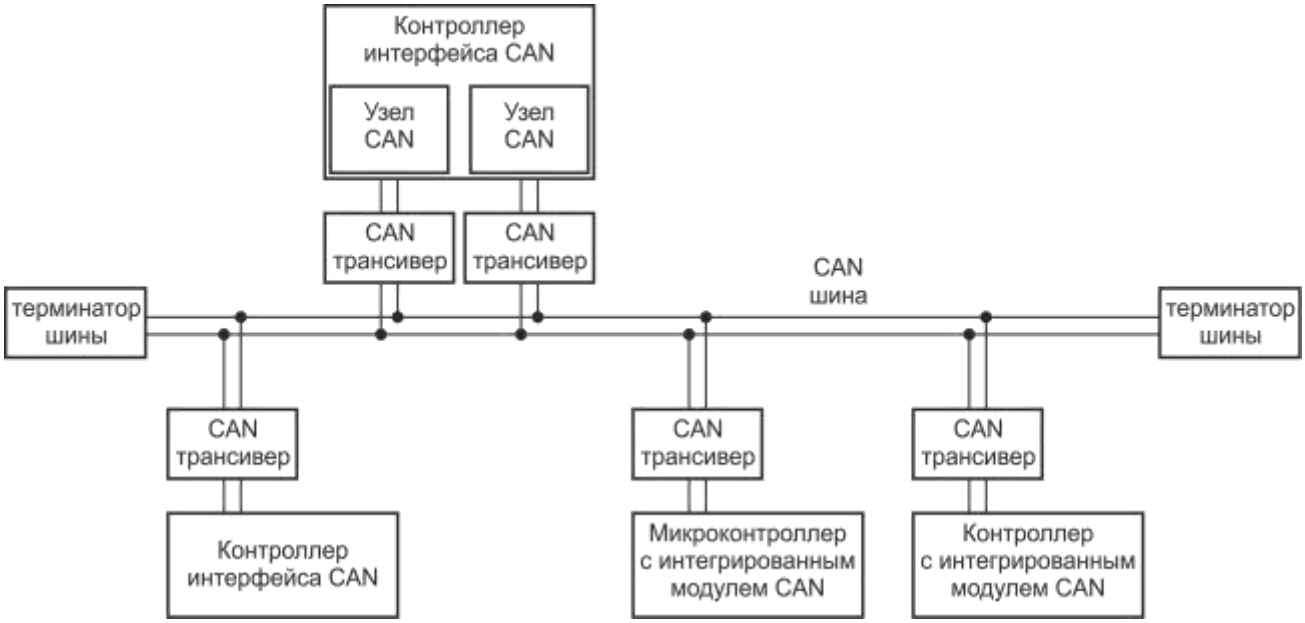


Рисунок 16.1 – Общая структура CAN сети

Логика шины работает по механизму монтажного И, в котором рецессивный бит соответствует логической единице, а доминантный – логическому нулю. Пока ни один узел не формирует доминантный бит, шина находится в рецессивном состоянии. Появление на шине доминантного бита (выставленного одним или несколькими узлами) создает доминантное состояние шины. Отсюда следует, что при выборе среды передачи данных необходимо точно определить, какое состояние будет доминантным, а какое – рецессивным. Одним из наиболее распространенных и дешевых вариантов линии связи является пара скрученных проводов. Линии шины тогда называются CANH и CANL и могут быть подключены непосредственно к устройствам. Не существует никакого дополнительного стандарта на среду передачи данных.

При использовании в качестве линии связи пары скрученных проводов с нагрузочными резисторами на концах можно получить максимальную скорость передачи данных 1 Мбит/с при длине линии до 40 м. Для линий связи протяженностью более 40 м необходимо снизить скорость передачи данных (для линии 1000 м скорость шины должна быть не более 40 Кбит/с). Из-за дифференциального характера линии связи шина CAN малочувствительна к электромагнитным помехам. Экранирование шины значительно снизит воздействие внешнего электромагнитного поля, что особенно важно для высокоскоростных режимов работы.

Двоичная информация кодируется. Доминантным является низкий уровень, рецессивным – высокий. Для гарантированной синхронизации данных всеми узлами шины используется принцип «бит-стаффинга». Это означает, что при последовательной передаче пяти бит одинаковой полярности передатчик вставляет один дополнительный бит противоположной полярности перед передачей остальных битов. Приемник также проверяет полярность и удаляет дополнительные биты.

В CAN протоколе при передаче данных приемные узлы не адресуются, а указывается идентификатор передатчика. С помощью идентификатора указывается содержание сообщения (например, применительно автомобиля – обороты, температура двигателя и т.д.) и степень приоритета сообщения. Более высокий приоритет у идентификатора, имеющего меньшее бинарное значение.

При коллективном доступе к шине используется неразрушающий арбитраж с опросом состояния шины. Перед началом передачи данных узел проверяет состояние шины (отсутствие активности на шине). При начале передачи сообщения узел становится управляющим шины, все остальные узлы переходят в режим приема. После приема сообщения (подтвержденного каждым узлом) каждый узел проверяет идентификатор в сообщении и сохраняет сообщение, если это требуется. В противном случае, сообщение сбрасывается. Если два или более узлов начинают передачу данных одновременно, поразрядный арбитраж позволяет избежать конфликта на шине. Каждый узел выдает на шину свой идентификатор (старший бит формируется первым) и контролирует ее состояние. Если узел посылает «1», а читает «0», значит, арбитраж потерян, и узел переключается в режим приема. Это происходит тогда, когда идентификатор конкурирующего узла имеет меньшее бинарное значение. Таким образом, узел с высоким приоритетом выигрывает арбитраж без необходимости повторять сообщение. Все остальные узлы будут пытаться передать сообщение после освобождения шины. Данный механизм не позволяет передавать сообщения одновременно разными узлами. Для этого программно должно быть обеспечено, чтобы узлы, передающие данные, не имели одинаковых идентификаторов. Оригинальная спецификация в версии CAN 2.0b (так называемая расширенная версия CAN) определяет возможность идентификатора иметь длину 11 или 29 бит.

Протокол CAN предусматривает следующие типы сообщений:

- сообщение данных (стандартное и расширенное);
- удаленный запрос данных;
- сообщение об ошибке;
- сообщение о перезагрузке.

16.1 Типы и структура сообщений CAN

Стандартное сообщение данных

Формируется, когда узел желает передать данные. Формат сообщения показан на рисунке 16.2.

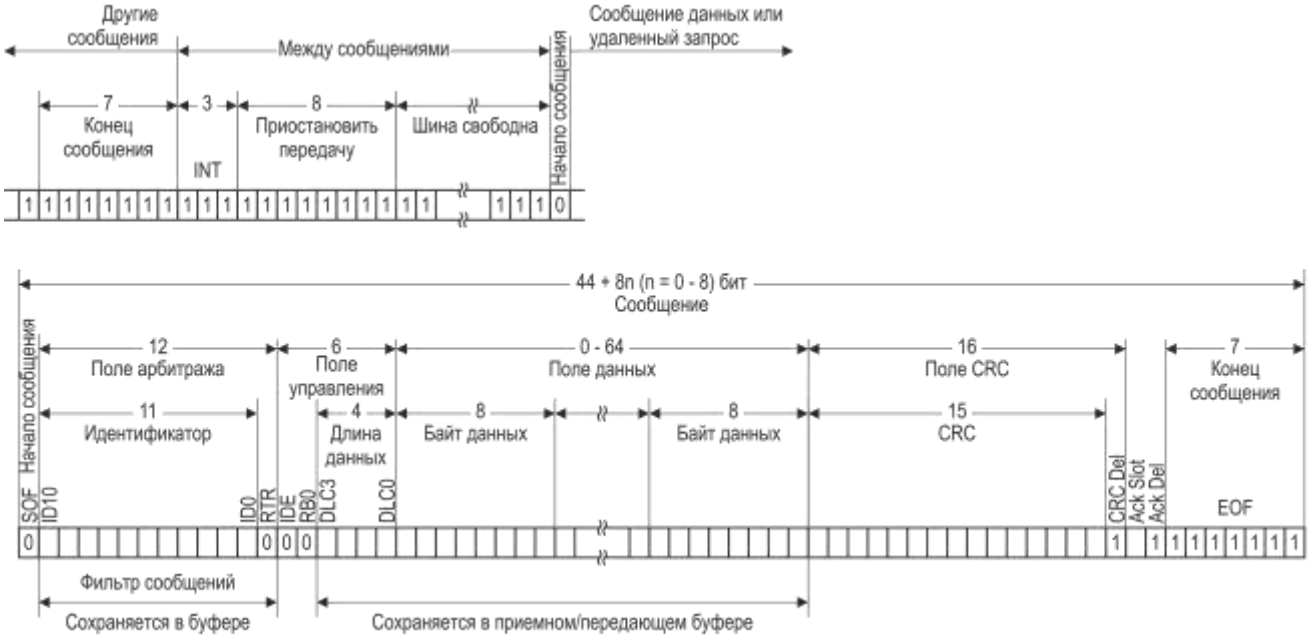


Рисунок 16.2 – Стандартное сообщение данных

Стандартное сообщение имеет в своем составе:

- бит SOF – доминантный («0») бит начала сообщения для жесткой синхронизации всех узлов;

- поле арбитража (12 бит), включающее поле ID идентификатора (11 бит) и бит RTR передачи по удаленному запросу (RTR = «0» соответствует сообщению данных, RTR = «1» соответствует удаленному запросу);

- поле управления (6 бит), включающее бит IDE указатель расширенного идентификатора (IDE = «0» соответствует стандартному идентификатору, IDE = «1» соответствует расширенному идентификатору), бит RB0 резервный доминантный бит и поле DLC числа байт данных (4 бита), которое указывает, сколько байт данных содержится в сообщении (допустимые значения – от 0 до 8, другие значения использоваться не могут);

- поле данных (от 0 до 64 бит), содержащее целое число байт данных;

- поле контрольной суммы CRC (16 бит), включающее поле CRC (15 бит), используемое для обнаружения возможных ошибок передачи данных и бит CRC Del рецессивный разделитель CRC;

- поле подтверждения (2 бита), включающее бит ACK Slot подтверждения передачи (передающий узел выдает рецессивный бит, а любой узел, который принял сообщение без ошибок, заменяет его сформированным доминантным битом) и бит ACK Del рецессивный разделитель подтверждения;

- поле EOF конца сообщения (7 бит).

Между передачами двух любых сообщений шина должна оставаться в рецессивном состоянии как минимум в течение времени появления 3 бит (поле INT простоя). Если после появления трех рецессивных битов (поле INT) ни один узел не начал передачу,

шина переходит в состояние бездействия IDLE и находится в рецессивном состоянии до появления доминантного бита сообщения.

Расширенное сообщение данных

Формируется, когда узел желает передать данные. Формат сообщения показан на рисунке 16.3.

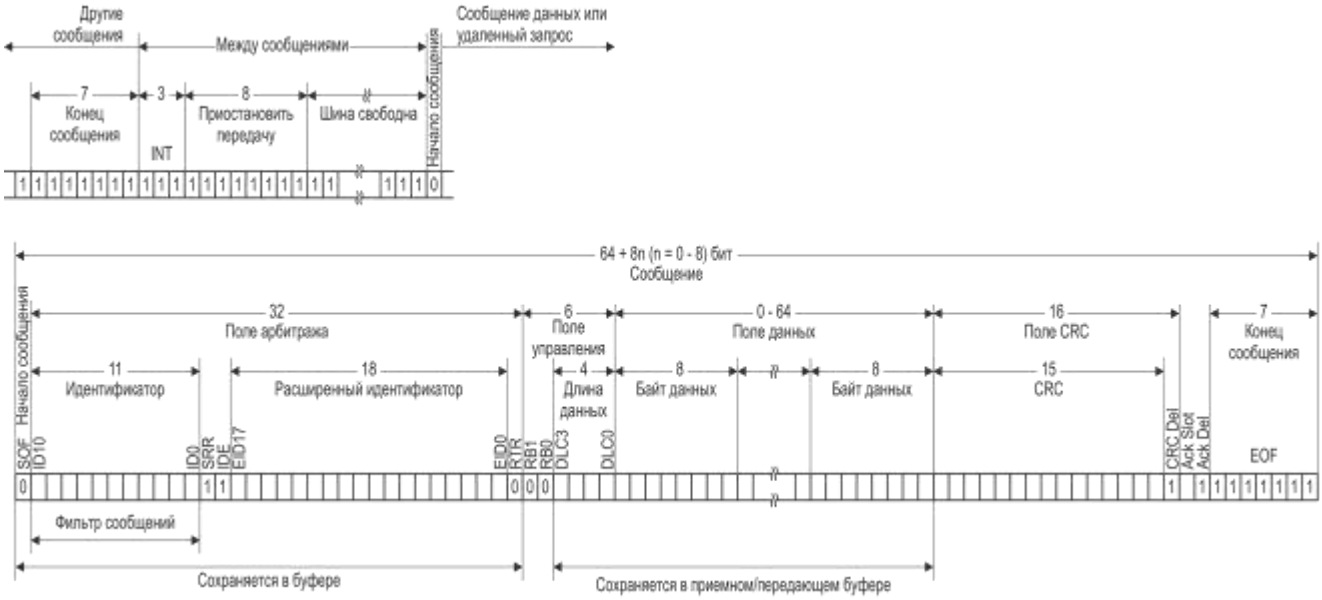


Рисунок 16.3 – Расширенное сообщение данных

Расширенное сообщение имеет в своем составе:

- бит SOF – доминантный («0») бит начала сообщения для жесткой синхронизации всех узлов;
- поле арбитража (38 бит), включающее поле стандартного идентификатора (11 бит), бит SRR заменитель удаленного запроса, бит IDE указатель расширенного идентификатора (рецессивный, что соответствует расширенному идентификатору) и поле расширенного идентификатора (18 бит);
- бит RTR передачи по удаленному запросу (RTR = «0» соответствует сообщению данных, RTR = «1» соответствует удаленному запросу);
- поле управления (6 бит), включающее бит RB0 резервный доминантный бит, бит RB1 резервный доминантный бит и поле DLC числа байт данных (4 бита), которое указывает, сколько байт данных содержится в сообщении (допустимые значения – от 0 до 8, другие значения использоваться не могут);
- поле данных (от 0 до 64 бит), содержащее целое число байт данных;
- поле контрольной суммы CRC (16 бит), включающее поле CRC (15 бит), используемое для обнаружения возможных ошибок передачи данных, и бит CRC Del рецессивный разделитель CRC;
- поле подтверждения (2 бита), включающее бит ACK Slot подтверждения передачи (передающий узел выдает рецессивный бит, а любой узел, который принял сообщение без ошибок, заменяет его сформированным доминантным битом), и бит ACK Del рецессивный разделитель подтверждения;
- поле EOF конца сообщения (7 бит).

Удаленный запрос данных

Формируется, когда узлу требуются данные другого узла. Узел назначения посылает удаленный запрос с идентификатором источника. Соответствующий узел источника (распознавший свой идентификатор) посылает стандартное или расширенное сообщение в ответ на запрос.

Удаленный запрос данных существует в стандартном и расширенном вариантах (на рисунке16.4 представлен вариант удаленного запроса со стандартным идентификатором).

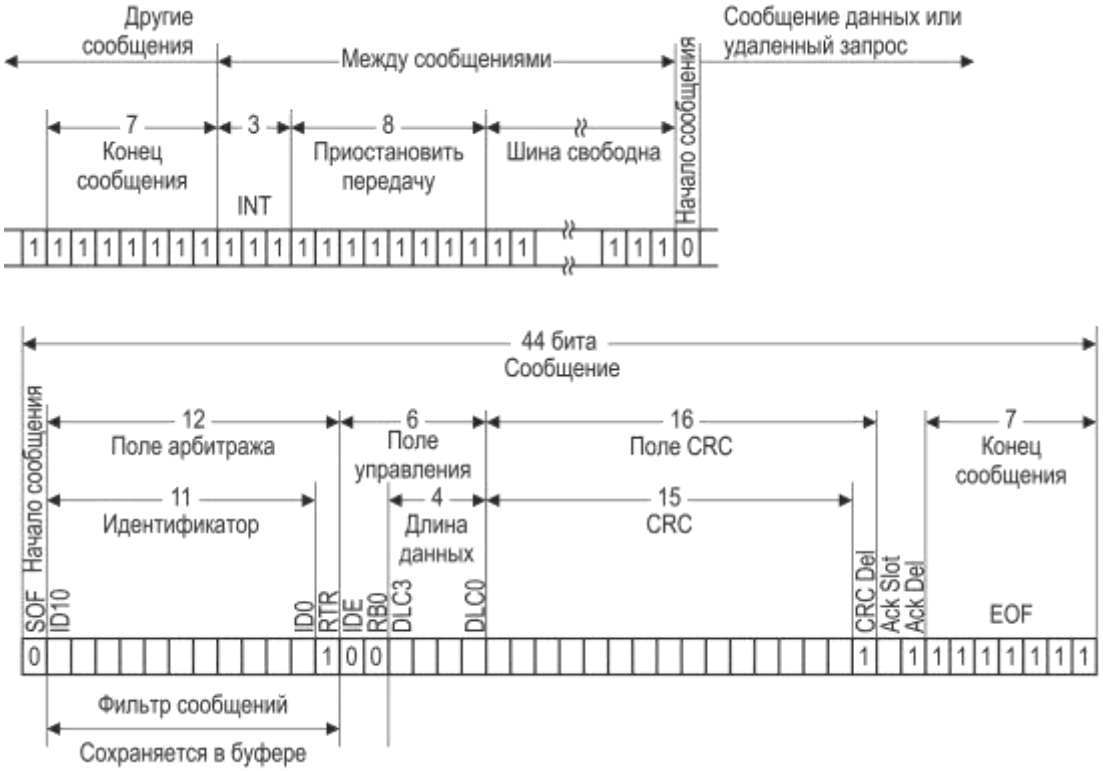


Рисунок 16.4 –Удаленный запрос данных (стандартный формат)

Имеются только два отличия содержимого удаленного запроса от сообщения данных:

- бит RTR в удаленном запросе передается в рецессивном состоянии;
- поле данных отсутствует (в сообщении не передается никаких данных, значение в поле DLC любое в пределах от 0 до 8).

В самом маловероятном случае, когда одновременно формируется удаленный запрос и устройство пытается передать данные с одинаковыми идентификаторами, арбитраж будет выигран устройством, передающим данные, из-за доминантного состояния бита RTR.

Узел, который посылал запрос, получает данные немедленно.

Сообщение об ошибке

Формируется любым узлом, который обнаруживает ошибку на шине. Формат сообщения показан на рисунке 16.5.

Сообщение об ошибке состоит из двух полей: поля разделителя ошибки и поля флага ошибки. Возможны два типа поля флага ошибки, в зависимости от вида ошибки узла, обнаружившего ее.

Если ошибку обнаружил активный узел (как в примере на рисунке 16.5), тогда он прерывает передачу текущего сообщения, формируя флаг активной ошибки. Флаг активной ошибки состоит из шести последовательных доминантных битов, которые нарушают правила бит-стаффинга (правила наполнения и передачи битов на шине). Остальные узлы также обнаруживают ошибку и начинают формировать сообщение об ошибке. Таким образом, поле флага ошибки может содержать от 6 до 12 доминантных битов (сформированных одним узлом или более). Поле флага ошибки дополняется разделителем ошибки, состоящим из 8 рецессивных битов и позволяющим перезапускать связь с шиной после обнаружения ошибки. После перехода шины в нормальное состояние

узлы возобновляют передачу данных, остановленный узел повторяет передачу сообщения, переданного до этого с ошибкой.

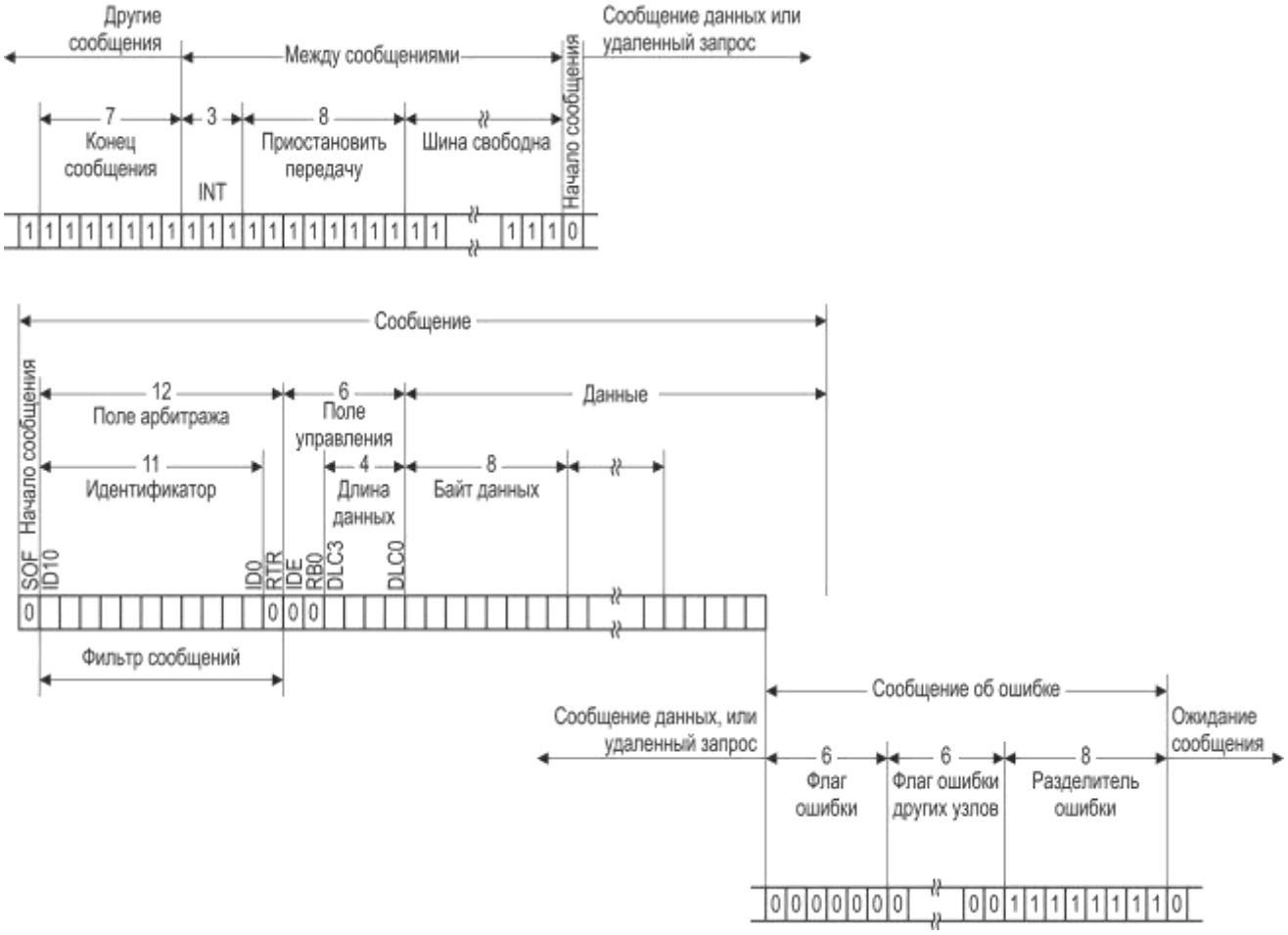


Рисунок 16.5 – Сообщение об ошибке

Если ошибку обнаружил пассивный узел, тогда он формирует флаг пассивной ошибки, состоящий из 6 последовательных рецессивных битов, и затем разделитель ошибки. Таким образом, сообщение о пассивной ошибке состоит из 14 рецессивных битов. Это не нарушает правила бит-стафинга на шине и не оказывает влияния на передачи других узлов. Исключение составляет узел, который передает данные узлу, обнаружившему ошибку. В этом случае правила бит-стафинга нарушаются и передача данных прекращается. После передачи пассивной ошибки узел должен ожидать 6 последовательных рецессивных битов для восстановления связи с шиной.

Сообщение о перезагрузке

Формат сообщения о перезагрузке аналогичен формату сообщения об ошибке, но может быть сформирован только, когда шина простаивает.

Сообщение о перезагрузке проиллюстрировано на рисунке 16.6.

Разделитель перезагрузки состоит из 8 последовательных рецессивных битов.

Узел может сформировать сообщение о перезагрузке в двух случаях:

- между сообщениями обнаружен доминантный бит, что является ненормальным во время простоя шины;

- для задержки передачи нового сообщения.

Узел может последовательно сформировать не более двух сообщений перезагрузки.

Флаг перезагрузки состоит из 6 последовательных доминантных битов. Другие узлы обнаруживают перезагрузку и начинают формировать ее самостоятельно. Поэтому на шине во время выполнения перезагрузки может быть до 12 доминантных битов.

Рисунок 16.6 – Сообщение о перезагрузке

16.2 Структура и функционирование модуля CAN

В состав контроллера CAN входят два идентичных независимых узла CAN0 и CAN1, ОЗУ для хранения сообщений, которое является общим для узлов, и система управления контроллером. Между узлами есть отличие лишь в подключении к выводам микроконтроллера, в связи с этим при описании работы узлов индексы 0 и 1 будут заменены символом «x». Дополнительно в контроллере реализовано «логическое И» выходов узлов с выводом результата на выводы микроконтроллера.

Контроллер CAN имеет следующие функциональные особенности:

- соответствие ISO 11898;
- функционирование согласно спецификации CAN 2.0b (активная версия);
- отдельные управляющие регистры для каждого из двух узлов;
- программируемая скорость передачи информации до 1 Мбит/с;
- гибкий и полный контроль передачи сообщений и обработки ошибок;
- 3 линии прерываний;
- 16 объектов сообщений для хранения сообщений и их параметров в ОЗУ. Каждый объект сообщения может быть привязан к любому из узлов, сконфигурирован для передачи или приема как стандартных, так и расширенных сообщений и удаленных запросов. Каждый объект имеет индивидуальную маску для фильтрации принимаемых сообщений. Объекты сообщений могут иметь разные уровни приоритета, могут объединяться для построения структур FIFO произвольных размеров (до 16 объектов в одной структуре). Кроме того, реализована возможность попарного соединения объектов для формирования шлюзов для автоматической передачи сообщений между узлами. Параллельно с вышеуказанными свойствами объекты сообщений могут организовываться в списки с постоянно доступной реорганизацией (совместимость с TwinCan-устройствами, которые не имеют списков).

Структура контроллера интерфейса CAN приведена на рисунке 16.7.

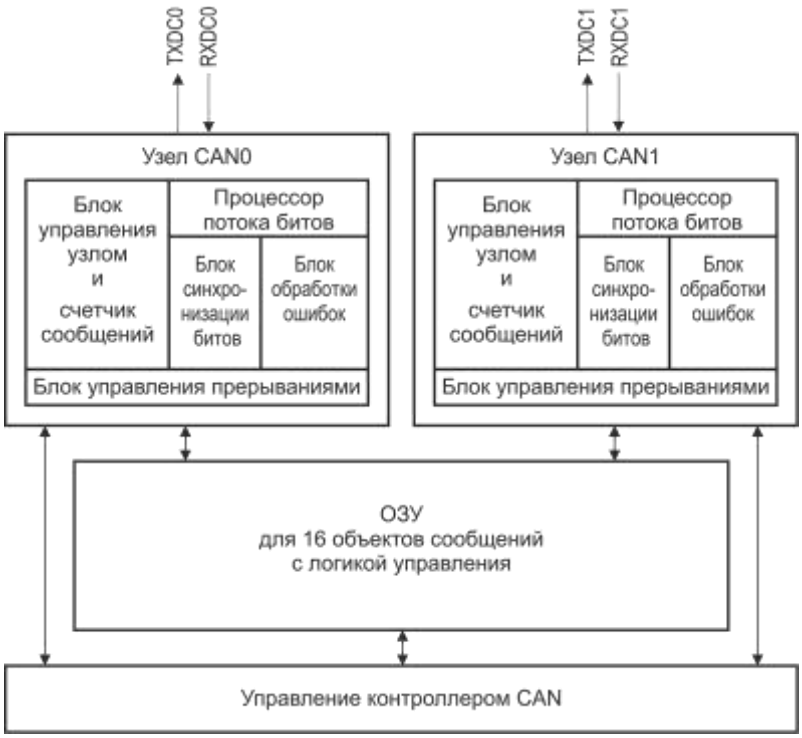


Рисунок 16.7 – Структура контроллера интерфейса CAN

Синхронизация

Тактирующим сигналом контроллера CAN является сигнал Fclc (Fin), приходящий с генератора тактовых сигналов. На основе этого сигнала посредством программируемого дробного делителя частоты формируется внутренний сигнал Fcan (Fout), синхронизирующий работу контроллера и являющийся базовым синхросигналом для передачи/приема сообщений по внешней шине CAN.

Включение контроллера CAN

По умолчанию, после сброса микроконтроллера контроллер CAN выключен. На это также указывает состояние флага DISS регистра CLC. Когда контроллер выключен, этот флаг установлен.

Для включения контроллера CAN следует записать ноль в бит DISR регистра CLC. После этого флаг DISS сбросится. Рекомендуется проверять состояние флага DISS, перед началом программирования регистров контроллера, которые не доступны в выключенном состоянии.

Выключение контроллера CAN

Программно можно перевести контроллер CAN в режим выключения установкой бита DISR. Контроллер завершает все текущие операции, после чего устанавливает флаг DISS и отключает внутреннее тактирование, в связи с чем все регистры становятся недоступными для обращения.

Простой шины

Между передачами сообщений шина CAN находится в рецессивном состоянии. Для выполнения условий простой шины необходимо, чтобы было получено, как минимум, три рецессивных бита после завершения передачи/приема очередного сообщения.

Анализ работы контроллера CAN

Для анализа работы контроллера доступны два режима – общего анализа и внутренней петли.

Режим общего анализа включается установкой бита CALM регистра NCRx узла и позволяет осуществлять независимый мониторинг работы узла, не затрагивая шину CAN. В этом режиме сообщения данных и удаленные запросы отслеживаются без участия узла в

операциях на шине. Выходы узла находятся в рецессивном состоянии. Узел может получать сообщения данных, сообщения удаленных запросов и сообщения об ошибках, но работа узла на передачу запрещена. Полученные сообщения данных/удаленных запросов остаются без подтверждения (бит подтверждения остается в рецессивном состоянии), но принимаются и сохраняются (при совпадении идентификаторов) в соответствующих объектах сообщений. В ответ на входящие сообщения не выдается подтверждение, и не генерируются сообщения об ошибках. На удаленные запросы не выдаются сообщения данных, а сами сообщения данных не могут быть переданы установкой бита запроса передачи TXRQ регистра состояния объекта сообщений MOSTATn. Прерывания после приема генерируются (если это разрешено) для всех принятых сообщений, не содержащих ошибок.

Режим внутренней петли включается установкой бита LBM регистра NPCRx и позволяет проводить внутреннее тестирование контроллера CAN, а также отладку управляющей программы без доступа к внешней шине CAN. Внутренняя петля состоит из внутренней шины CAN (внутри контроллера CAN) и переключателя выбора шины для каждого узла (см. рисунок 16.8). С помощью переключателя каждый узел CAN может быть подключен либо к внутренней шине (режим внутренней петли), либо к внешней шине (нормальный режим работы). Если выбран режим внутренней петли, то на внешнем передающем выводе узла CAN поддерживается рецессивный уровень сигнала, а состояние принимающего вывода игнорируется.

Если оба узла CAN функционируют в режиме внутренней петли, они взаимодействуют друг с другом посредством внутренней шины CAN, не оказывая влияние на работу других модулей, функционирующих в нормальном режиме.

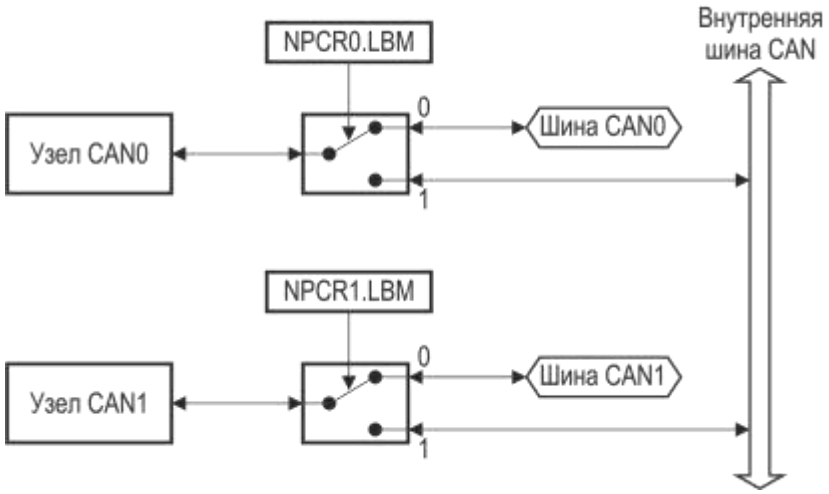


Рисунок 16.8– Режим внутренней петли

Дробный делитель

Дробный делитель позволяет генерировать частоту fout из входной тактовой частоты fin (fclc) путем программирования делителя посредством регистра FDR.

Примечание – Задаваемое значение входной частоты fin зависит от длительности передачи одного бита информации и должно быть n-кратно ей. Поскольку длительность передачи бита определяется количеством квантов времени (Ntq, см. далее), то для расчета частоты fin в МГц следует пользоваться формулой:

$$fin = n \times Ntq,$$

где Ntq – количество квантов времени tq,

n – целое число, начиная с 1 (для задания кратности).

Дробный делитель делит частоту fin путем умножения на величину 1/val или величину 1024/val для любого val от 0 до 1023, выдавая на выходе тактовый сигнал fout (fcan).

На рисунке 16.9 показана блок-схема дробного делителя. Логика дробного делителя работает по-разному, в зависимости от режима, задаваемого полем DM.

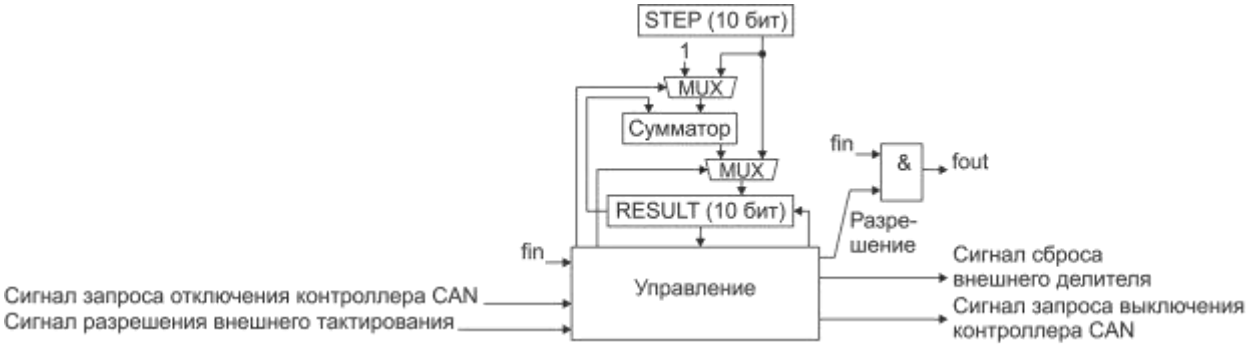


Рисунок 16.9 – Схема дробного делителя

В режиме нормального деления (DM = 01b) делитель работает как перегружаемый счетчик с шагом инкрементирования, равным единице. Состояние счетчика доступно посредством поля RESULT. Каждый раз при переполнении (т.е. когда RESULT = 3FFh), формируется импульс сигнала Fout, после чего в счетчик загружается значение из поля STEP.

Выходная частота fout определяется по формуле:

fout = fin× 1/ (1024 – STEPd),

где STEPd – значение поля STEP в десятичном формате.

Отсюда следует, что для получения частоты fout = fin, значение STEP должно быть равно 3FFh. На рисунке 16.10 показано формирование сигнала Fout при значении STEP = = 3FDh (1021d).

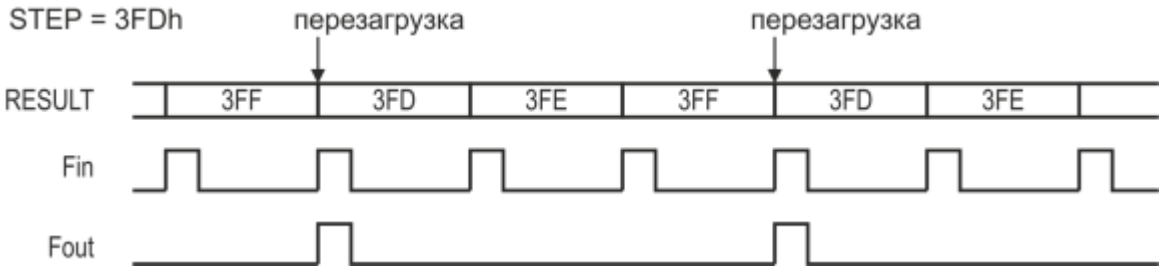


Рисунок 16.10 – Формирование сигнала с частотой fout в нормальном режиме

В режиме дробного деления (DM = 10b) делитель работает как перезагружаемый счетчик, но шаг инкрементирования в этом случае равен значению поля STEP. Если результат инкрементирования значения RESULT на величину STEP превышает 3FFh, возникает переполнение счетчика, формируется импульс сигнала Fout, после чего в счетчик загружается значение, на которое результат инкрементирования превысил 3FFh.

Выходная частота fout определяется по формуле:

fout = fin× STEPd / 1024d.

В целом, режим дробного деления позволяет программировать частоту fout с более высокой точностью, чем нормальный режим, но сигнал может иметь «джиттер» периода,

не превышающий одного периода fin, в связи с чем не рекомендуется использовать режим дробного деления при высоких скоростях передач.

На рисунке 16.11 показано формирование сигнала Fout при значении STEP = 234h (564d).

fout = fin× 564/1024 = 0,55 × fin

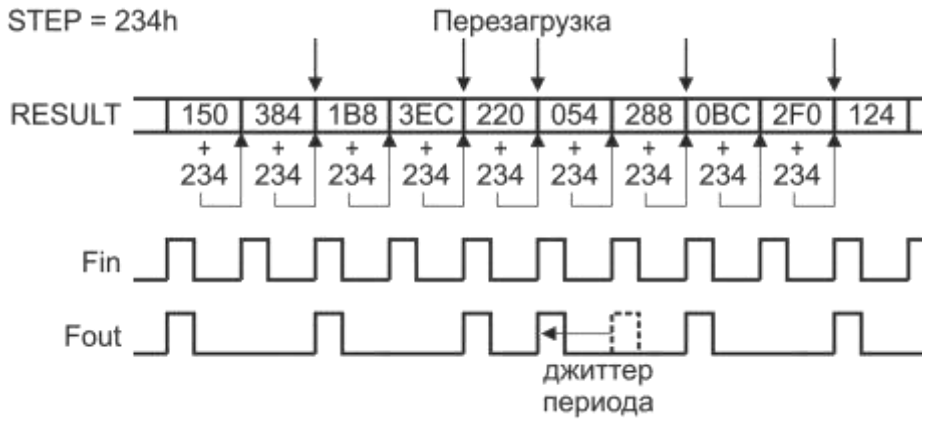


Рисунок 16.11 – Формирование сигнала с частотой fout в режиме дробного деления

Процесс выключения делителя начинается одновременно с возникновением запроса выключения контроллера CAN.

Контроллер сообщений

Управляет обменом сообщениями между CAN узлами и памятью сообщений и выполняет следующие функции:

- фильтрация входящих сообщений для определения корректного объекта сообщений для сохранения полученных данных;
- определение объекта сообщений, содержимое которого будет передано в первую очередь (для каждого узла индивидуально);
- передача содержимого объекта сообщений к CAN узлу с параллельной вставкой в сообщение битов управления и состояния;
- осуществление буферизации FIFO и функционирования шлюза;
- объединение битов уведомления ждущих обработки сообщений.

Управление прерываниями модуля CAN

На рисунке 16.12 показана структура формирования запроса на прерывание.

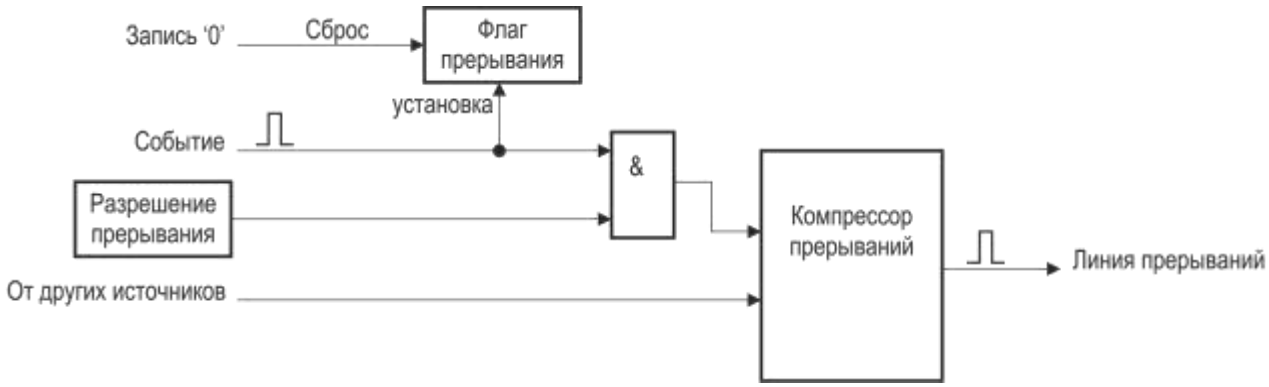


Рисунок 16.12 – Структура формирования запроса на прерывание

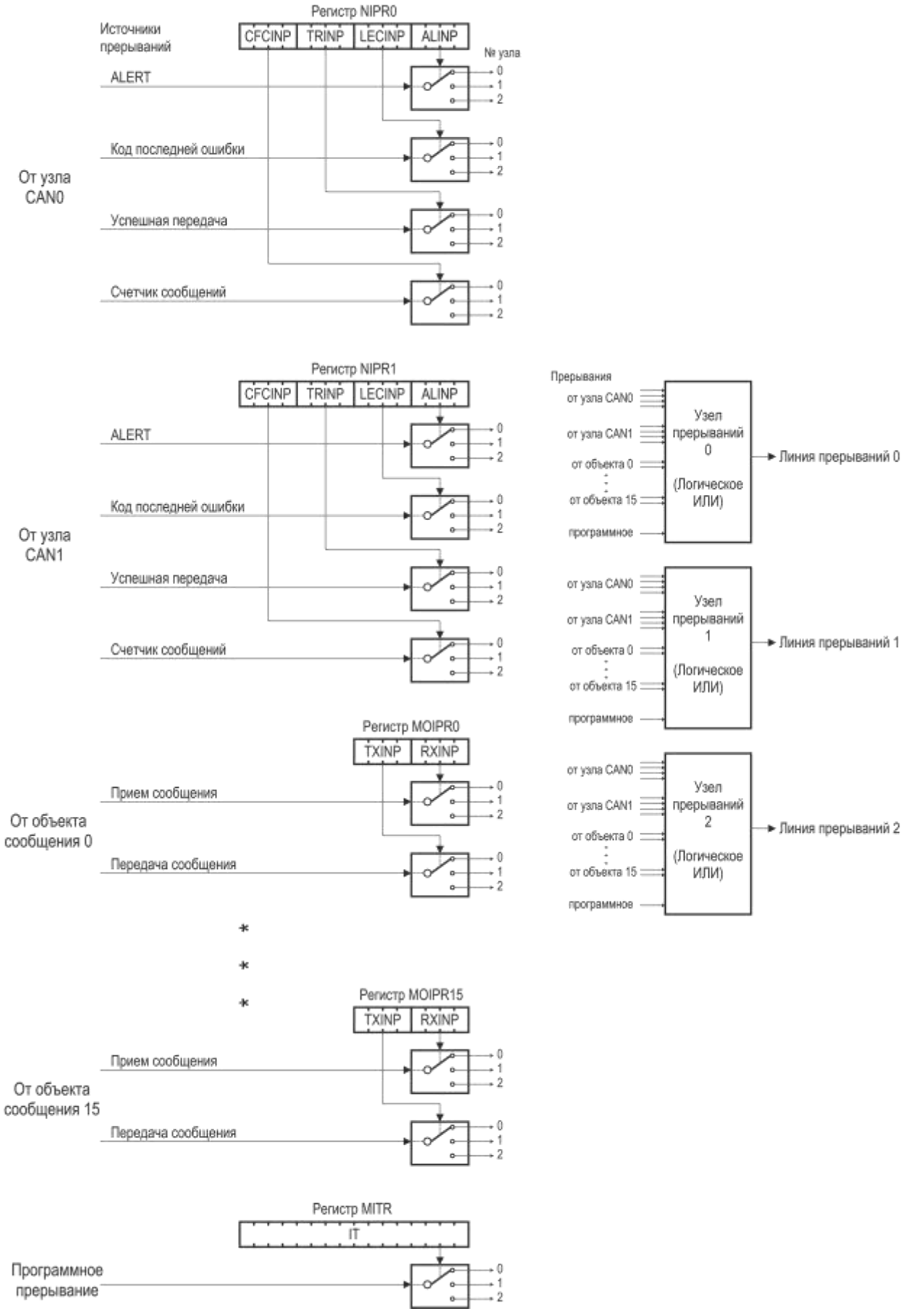


Рисунок 16.13 – Компрессор прерываний

Событие, по которому должен быть сгенерирован запрос на прерывание, устанавливает флаг прерывания и (если разрешено) формирует запрос на прерывание на одной из 16 линий прерываний. Импульс запроса на прерывание генерируется независимо от состояния флага прерывания. Флаг прерывания может быть сброшен программно, записью нуля. Если к одной линии прерываний подключены несколько источников прерываний, то появление импульса от любого источника сформирует запрос на прерывание. Логика управления прерываниями использует схему компрессии прерываний.

Источниками прерываний являются:

- CAN узлы (8 источников – по 4 для каждого узла);
- объекты сообщений (32 источника – по 2 для каждого объекта);
- программное прерывание (источник – регистр MITR).

Каждый аппаратный источник прерывания управляется 4 битами указателя прерываний, который определяет для него одну из трех линий прерываний, что позволяет коммутировать на одну линию несколько источников прерываний. На рисунке 16.13 представлен компрессор прерываний.

Когда объект сообщения n генерирует запрос на прерывание по окончании приема или передачи сообщения, запрос передается на линию прерываний, выбранную в битовом поле RXINP или TXINP регистра MOIPRn объекта сообщения n. Если количество объектов сообщений больше, чем количество линий прерываний, то на одну линию могут приходить несколько запросов прерываний. Для разрешения конфликтов на линиях прерываний в модуле CAN предусмотрен механизм распределения приоритетов для объектов сообщений.

Выводы микроконтроллера, связанные с выводами контроллера CAN

Выводы для приема и передачи сообщений закреплены за выводами порта P1 (см. таблицу 2.1). Каждый узел CAN имеет одну входную и одну выходную линии, которые посредством выводов микроконтроллера соединяются с внешней CAN-шиной.

16.3 Узел модуля CAN

Каждый узел CAN имеет свою собственную логику управления и выдачи информации о состоянии и может быть сконфигурирован и работать независимо от другого узла.

Режим конфигурации включается установкой бита CCE регистра NCRx. Режим конфигурации позволяет изменять параметры синхронизации битов и состояния счетчиков ошибок.

Конфигурация прерываний задается битами TRIE, ALIE и LECIE:

- бит TRIE управляет разрешением прерывания после передачи сообщения;
- бит ALIE управляет разрешением прерываний по ошибке.
- бит LECIE управляет разрешением прерывания по коду последней ошибки.

Регистр NSRx отражает текущее состояние, содержит информацию о передачах и ошибках узла.

Блок управления узлом

Координирует работу:

- разрешает/запрещает действия узла на шине;
- разрешает/запрещает и генерирует различные события, касающиеся работы узла (ошибка на шине, успешное завершение передачи сообщения), которые приводят к формированию запросов на прерывания;
- управляет счетчиком сообщений.

Блок синхронизации битов

Согласно стандарту ISO 11898 время передачи одного бита разделено на сегменты, которые, в свою очередь, составлены из целочисленных отрезков времени, называемых

квантами времени tq (см. рисунок 16.14). Квант времени – фиксированная единица времени, получаемая из частоты синхронизации и делителя модуля CAN.

Сегмент синхронизации Tsync позволяет синхронизировать начало обмена данными между передатчиком и приемником. Длительность сегмента всегда равна одному кванту времени.

Сегмент распространения – Tprop. Используется для компенсации физического времени запаздывания сигнала в пределах сети. Длительность сегмента рассчитывается с учетом времени прохождения сигнала от передатчика к приемнику и обратно, входной задержки компаратора и задержки выхода драйвера и может составлять от 1 до 8 квантов времени.

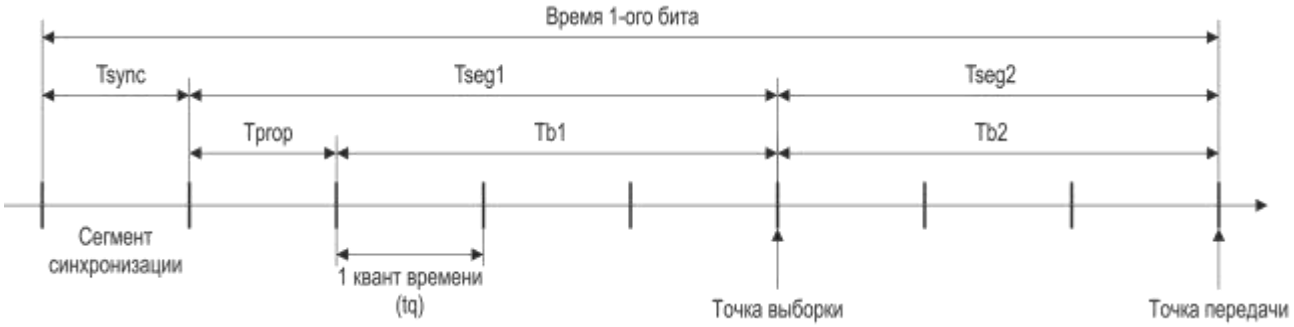


Рисунок 16.14 – Структура одного бита

Сегменты буфера фазы 1 и буфера фазы 2 – Tb1 и Tb2, расположенные до и после точки выборки, используются для компенсации смещения фазы тактовых частот источника и приемника, обнаруживаемой после появления сегмента синхронизации, а также для оптимального расположения точки выборки полученного бита.

Точка выборки – момент, когда читается состояние шины для определения принятого бита. Как правило, длительность временного интервала от начала бита до точки выборки составляет 60 – 70 % времени бита, в зависимости от системных параметров.

Сегмент распространения и сегмент буфера фазы 1 вместе составляют сегмент параметра 1 (Tseg1), который определяется битовым полем TSEG1 регистра синхронизации битов NBTRx (может быть записан, только если установлен бит CCE регистра NCRx). Согласно стандарту ISO, минимальная длительность сегмента параметра 1 должна составлять три кванта времени.

Сегмент параметра 2 (Tseg2) определяется битовым полем TSEG2 регистра NBTRx и охватывает сегмент буфера фазы 2. Минимальная длительность сегмента параметра 2 составляет два кванта времени.

Согласно стандарту ISO, минимальная длительность одного бита, получающаяся сложением сегментов Tsync, Tseg1 и Tseg2 не должна быть менее 8 квантов времени.

Максимальная длительность бита – 25 квантов времени.

Примечание – Минимальное номинальное время передачи одного бита составляет 1 мкс, что соответствует скорости передачи 1 Мбит/с.

Формулы вычисления значений сегментов и времени одного бита Tbit:

при DIV8 = 0 значение кванта времени $tq = (BRP + 1) / fout$;
при DIV8 = 1 значение кванта времени $tq = 8 \times (BRP + 1) / fout$;
$Tsync = 1 \times tq$;
$Tseg1 = (TSEG1 + 1) \times tq \geq 3tq$;
$Tseg2 = (TSEG2 + 1) \times tq \geq 2tq$;
$Tbit = Tsync + Tseg1 + Tseg2 \geq 8tq$.

Чтобы компенсировать смещение фазы между частотами генераторов различных узлов шины, каждое устройство должно синхронизироваться по фронту смены уровня сигнала на шине от рецессивного к доминантному. Как только фронт обнаруживается, логика синхронизации сравнивает его текущее положение с ожидаемым и выполняет настройку значений параметров Tseg1 и Tseg2.

Контроллер CAN использует два механизма синхронизации – аппаратный и ресинхронизацию (синхронизация с восстановлением тактовых интервалов).

Аппаратная синхронизация выполняется по каждому фронту смены уровня сигнала на шине от рецессивного к доминантному. При аппаратной синхронизации временные интервалы сегментов, из которых складываются времена битов, не изменяются в течение всего сообщения.

Ресинхронизация выполняется автоматическим удлинением сегмента Tseg1 или укорачиванием сегмента Tseg2. Максимальное значение изменения сегментов колеблется в пределах от 1 до 4 квантов времени. Синхронизация выполняется только при появлении фронта смены уровня сигнала на шине от рецессивного к доминантному. Фиксированное значение максимального числа последовательных бит одинаковой полярности гарантирует своевременное восстановление синхронизации. Смещение фазы фронта смены уровня сигнала на шине отслеживается относительно сегмента синхронизации и измеряется в квантах времени.

Если величина фазового смещения меньше или равна запрограммированному значению ширины перехода ресинхронизации Tsjw, выполняется аппаратная синхронизация.

Если величина смещения фазы больше, чем Tsjw, а фазовое смещение положительно, то удлиняется сегмент Tseg1, в случае отрицательного фазового смещения укорачивается сегмент Tseg2.

Значение Tsjw определяется полем SJW регистра NBTRx по формуле:

$Tsjw = (SJW + 1) \times tq$.

Помимо прочего, должны соблюдаться следующие правила:

$Tseg1 \geq Tsjw + Tprop$ и $Tseg2 \geq Tsjw$.

Соотношения между максимальным отклонением частоты fout и сегментами буферов фаз и шириной перехода ресинхронизации следующие:

$\Delta fout \leq T/2 \times (13 \times Tbit - Tb2)$,

$\Delta fout \leq Tsjw / 20 \times Tbit$,

где T – меньшее из Tb1 и Tb2.

В итоге:

- Tsync составляет 1 квант времени;
- Tprop – от 1 до 8 квантов времени;
- Tb1 – от 1 до 8 квантов времени;
- Tb2 – выбирается равным двум квантам времени или равным сегменту Tb1, если его значение более двух квантов времени;
- Tsjw может составлять максимально 4 кванта времени, однако, в типовых приложениях достаточно 1.

Корректные значения параметров синхронизации битов должны быть записаны в регистр NBTRx (доступен, если установлен бит CCE) до окончания инициализации (до сброса бита INIT регистра NCRx), т.е. до начала работы CAN узла.

Процессор потока битов

Процессор потока битов формирует (на основе содержимого объектов сообщений) сообщения данных и удаленные запросы непосредственно перед отправкой на шину CAN. Процессор потока управляет генератором CRC (генератор контрольной суммы) и добавляет контрольную сумму к сообщению. После вставки битов начала (SOF) и конца

(EOF) сообщения, процессор потока начинает передачу сообщения по правилам арбитража шины CAN. В течение всего времени передачи сообщения процессор потока битов ведет мониторинг шины. Если обнаруживается несовпадение текущего (определяемого мониторингом) и ожидаемого (выдаваемого CAN узлом) уровней напряжения на шине, генерируется ошибка и соответствующий ей запрос на прерывание. Код возникшей ошибки отражается в битовом поле LEC регистра NSRx.

Корректность получаемых данных проверяется и подтверждается или не подтверждается кодом CRC. В случае неподтверждения возникает ошибка, генерируется запрос на прерывание и код ошибки выставляется в регистре NSRx. Кроме этого, на шину выдается сообщение об ошибке.

После получения сообщения, не содержащего ошибок, и разбиения его на идентификатор и пакет данных полученная информация записывается в буфер блока обработки сообщений, формируется соответствующее прерывание, и обновляются регистры состояния.

Блок обработки ошибок

Блок обработки ошибок CAN узла предназначен для выявления ошибок в работе устройств узла. В составе блока есть два счетчика: счетчик ошибок приема (поле REC в регистре NECNTx) и счетчик ошибок передачи (поле TEC). Инкрементированием и декрементированием счетчиков управляет процессор потока битов.

Если процессор потока битов сам выявляет ошибку в процессе передачи, то счетчик ошибок передачи (поле TEC) инкрементируется на 8. Инкрементирование на 1 происходит, если об ошибке сообщено внешним CAN-устройством путем генерирования сообщения об ошибке. Направление передачи с ошибочным сообщением и узел, сообщивший об ошибке передачи, указывают на соответствующие узлы CAN в регистрах NECNTx, что используется для анализа ошибки.

В зависимости от значений счетчиков ошибок, узел CAN может находиться в одном из трех состояний:

- активной ошибки;
- пассивной ошибки;
- отключен от шины.

Узел находится в состоянии активной ошибки, если значение каждого из счетчиков ошибок меньше 128. Узел в состоянии активной ошибки присоединен к шине и посылает флаг активной ошибки при обнаружении ошибок.

Узел находится в состоянии пассивной ошибки, если значение хотя бы одного из счетчиков ошибок больше или равно 128. Узел подключен к шине, но при обнаружении ошибок посылает флаг пассивной ошибки. После передачи узел в состоянии пассивной ошибки будет ждать инициализации дальнейшей передачи.

Узел находится в состоянии отключения от шины, если значение счетчика ошибок TEC больше или равно 256. О том, что CAN узел находится в состоянии отключения от шины, сигнализирует флаг BOFF регистра NSRx. Узел в состоянии отключения от шины не может работать с шиной (выходные передатчики отключены).

Флаг EWRN регистра NSRx устанавливается, когда хотя бы один из счетчиков достиг или превысил лимит ошибок, определенный в битовом поле EWRNLVL регистра NECNTx. Как только значения обоих счетчиков перестанут превышать лимит ошибок, флаг EWRN сбросится.

Счетчик сообщений

Счетчик сообщений может использоваться для проверки последовательности передаваемых битов объектом сообщений или получения информации о завершении передачи/приема сообщения соответствующего узла CAN. Подсчет сообщений осуществляется 16 разрядным счетчиком, который управляется регистром NFCRx. Битовые поля CFMOD и CFSEL определяют режим работы и событие для инкрементирования счетчика.

Каждый узел CAN имеет в своем составе 16-разрядный счетчик сообщений/ синхросчетчик, который подсчитывает количество принятых и переданных сообщений. Битовое поле CFSEL определяет один из трех режимов работы счетчика.

В режиме подсчета сообщений после успешной передачи и/или успешного приема сообщения, содержимое счетчика копируется в битовое поле CFCVAL регистра MOIPRn объекта сообщений n, участвующего в пересылке данных. После чего счетчик сообщений инкрементируется.

Прерывания узла CAN

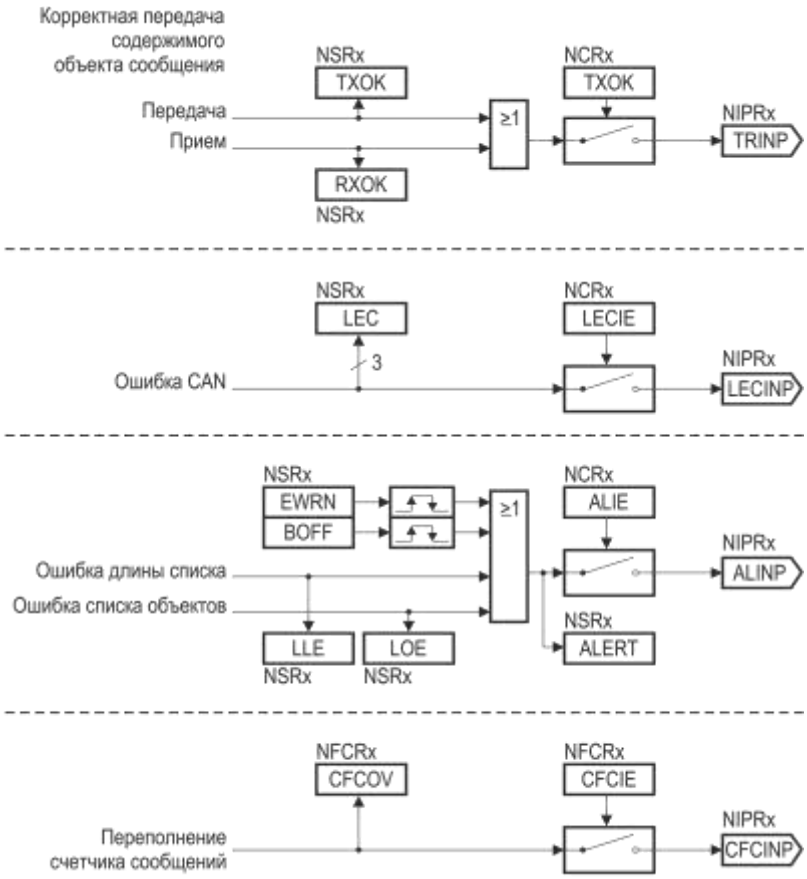


Рисунок 16.15 – Прерывания CAN узла

Узел может генерировать запросы на прерывания (см. рисунок 16.15) в случае:

- успешной передачи/приема сообщения;
- обнаружения кода последней ошибки;
- переполнения счетчика сообщений;
- состояния ALERT (состояние, возникающее, когда хотя бы один из счетчиков ошибок узла достиг значения своего лимита, изменяется состояние «отключен от шины», возникает ошибка длины списка или ошибка списка объектов).

После каждой успешной передачи или успешного приема сообщения генерируется (если разрешено соответствующими битами TXOK и RXOK) прерывание. Битовое поле TRINP регистра NIPRx задает одну (из трех) линию прерывания.

Прерывание узла при возникновении кода последней ошибки формируется (если разрешено битом LECIE), если после модификации поля LEC его значение больше нуля. Битовое поле LECINP задает линию прерывания.

Прерывание узла при переполнении счетчика сообщений генерируется, если оно разрешено битом CFCIE регистра NFCRx. Битовое поле CFCINP задает линию прерывания.

ALERT прерывание может быть сформировано (если разрешено битом ALERT) любым из следующих событий:

- изменением состояния бита BOFF;
- изменением состояния бита EWRN;
- ошибкой длины списка, которая также выставляет бит LLE;
- ошибкой элемента списка, которая также выставляет бит LOE;
- выставлением аппаратоно бита INIT.

Битовое поле ALINP задает линию прерывания.

В дополнение к аппаратным прерываниям есть возможность программного генерирования прерываний с использованием регистра прерываний MITR. Запись единицы в n-й разряд битового поля IT генерирует сигнал запроса прерывания на соответствующей ему n-ой линии прерываний (одной из 16). Установка нескольких битов приводит к параллельному генерированию запросов прерываний на соответствующих установленным битам линиях прерываний.

16.4 Объекты сообщений

Расположение регистров управления и состояния объектов сообщений представлено на рисунке 16.16, где для примера взят пятый объект сообщения. Расположение регистров всех объектов сообщений идентично.

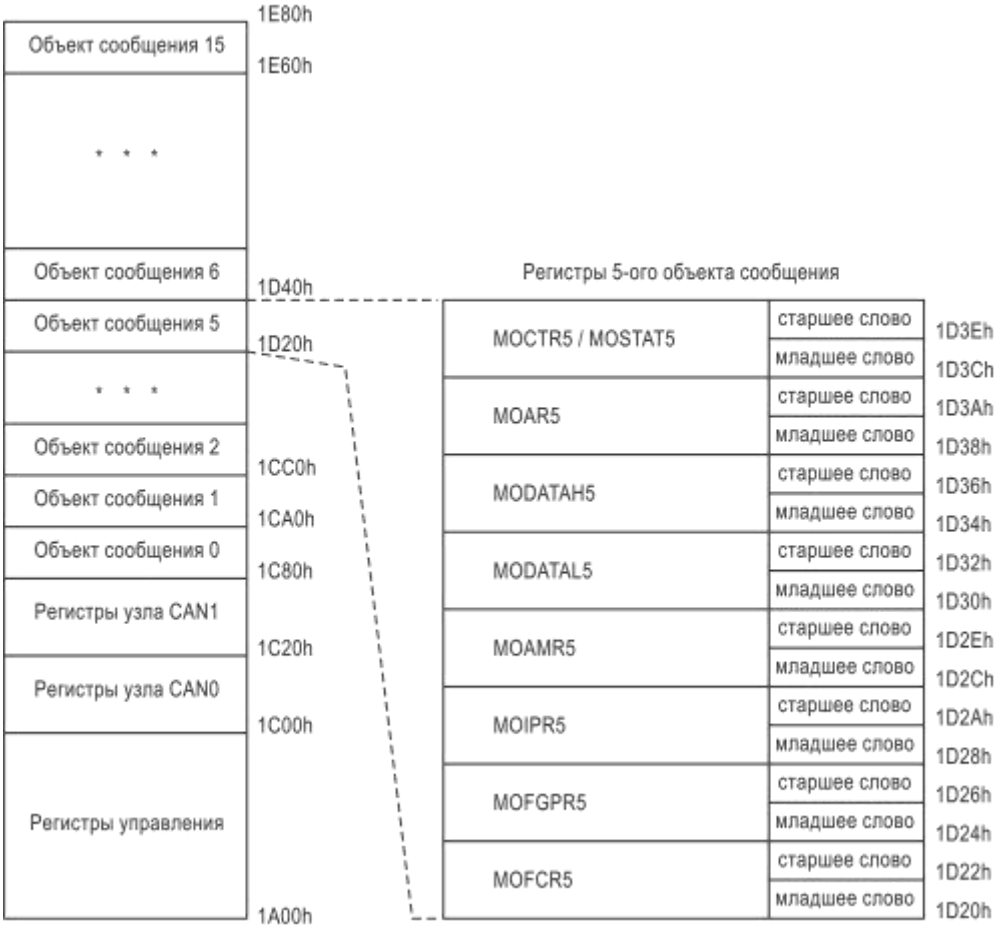


Рисунок 16.16 – Структура объектов сообщений

Регистры управления и состояния объектов сообщений

В состав каждого объекта сообщения входят девять 32-разрядных регистров:

- управления и состояния – MOCTRn (только запись) и MOSTATn (только чтение), расположенные по одному адресу;
- арбитража – MOARn;
- данных – MODATAHn и MODATALn;
- маски – MOAMRn;
- указателя прерываний – MOIPRn;
- указателя FIFO/шлюза – MOFGPRn;
- управления функционированием – MOFCRn.

Контроллер списка

Структура списка объектов сообщений

Объекты сообщений модуля CAN могут быть организованы в восемь списков (см. рисунок 16.17).

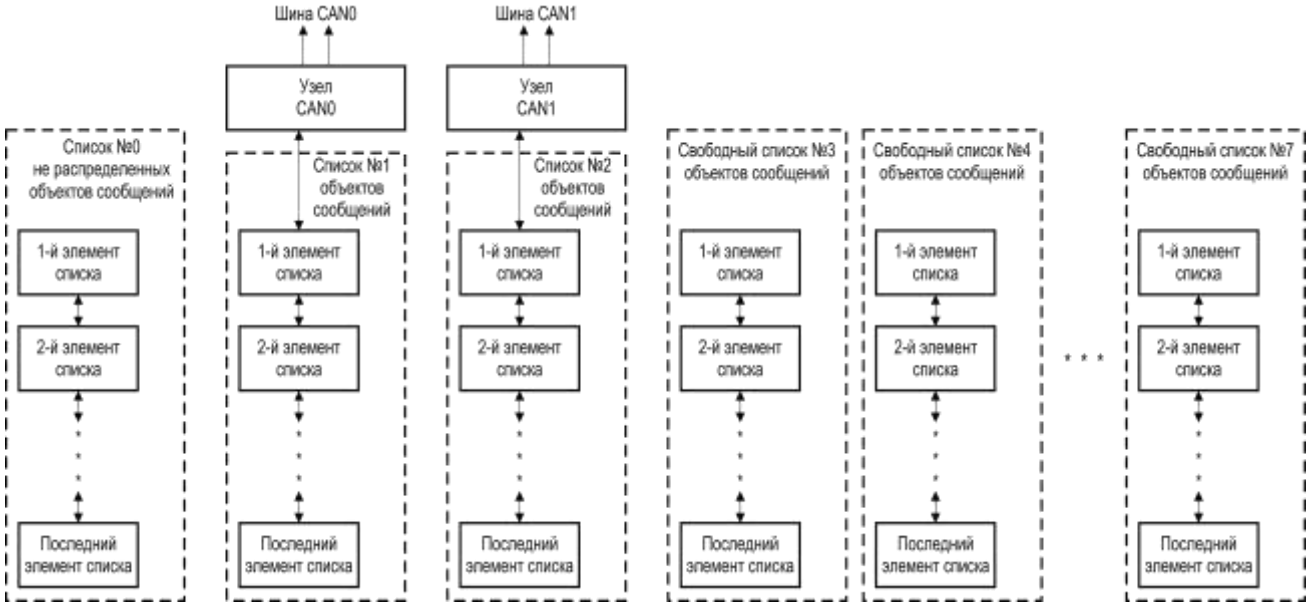


Рисунок 16.17 – Списки контроллера CAN

Каждый объект сообщения может быть добавлен в один из списков. Каждый узел CAN имеет свой список и соответствующий регистр списка. Регистр LIST1 отражает состояние списка №1 узла CAN0, регистр LIST2 – списка №2 узла CAN1.

Примечание – Узел может оперировать только с теми объектами сообщений, которые занесены в принадлежащий ему список.

Положение объекта сообщения n в списке определяется посредством регистра MOSTATn, который содержит указатели на предшествующий ему и следующий за ним элементы списка (объекты). Нераспределенные между узлами CAN объекты сообщений по умолчанию организуются в отдельный список №0, состояние которого отражается в регистре LIST0. Остальные пять списков с номерами от 3 до 7 являются свободными (не принадлежат ни одному узлу) и имеют соответствующие регистры LIST3 – LIST7.

Примечание – Объекты сообщений, распределенные в списки с 3 по 7, не могут быть использованы узлами CAN.

Механизмы FIFO и шлюза (см. далее) оперируют с объектами сообщений, независимо от их распределения по спискам, что дает возможность работы со всеми восемью списками. Следовательно, при использовании механизмов FIFO и шлюза следует внимательно следить за содержимым списков.

На рисунке 16.18 представлен вариант, когда объекты сообщений с номерами 3, 5 и 16 занесены в список № 2, принадлежащий узлу CAN1. Состояние списка отражено в регистре LIST2.

Значение поля BEGIN регистра LIST2 указывает на первый элемент списка (объект сообщения 5). Значение поля END указывает на последний элемент списка (объект сообщения 3). Количество элементов списка (количество объектов сообщений в списке) отражается в поле SIZE (значение SIZE всегда на единицу меньше количества элементов списка). Бит EMPTY является индикатором заполнения списка. Если список пуст, бит EMPTY установлен, в противном случае, бит сброшен.

Каждый объект сообщений содержит номер списка (поле LIST регистра MOSTATn), к которому он относится, а также указатели PNEXT и PPREV на следующий по списку объект сообщения и предшествующий соответственно. Поле PPREV первого по списку объекта сообщения должно указывать на этот же объект. Поле PNEXT последнего по списку объекта сообщения должно указывать на этот же объект.

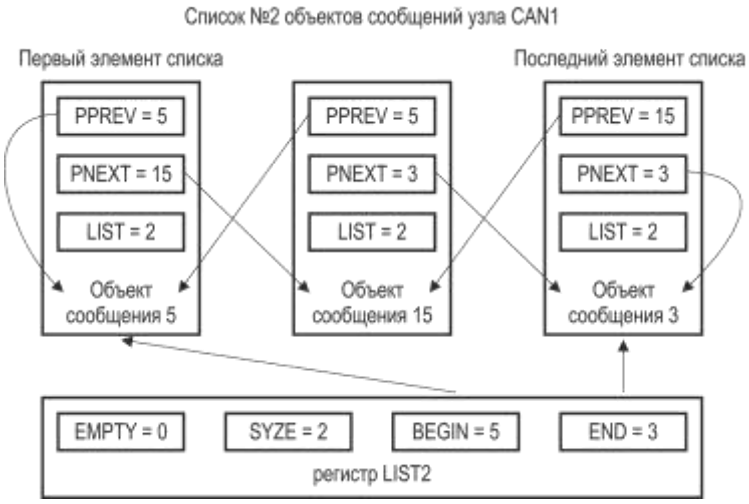


Рисунок 16.18 – Пример списка объектов сообщений

На рисунке 16.18 указатель PPREV пятого объекта сообщения (первого в списке) имеет значение 5h, а указатель PNEXT третьего объекта сообщения (последнего в списке) имеет значение 3h. Значение поля LIST всех трех объектов сообщений равно 2h.

Объект сообщения, у которого LIST = 0h относится к нулевому списку нераспределенных объектов. После сброса все объекты сообщений считаются нераспределенными. По умолчанию, порядок элементов списка № 0 следующий: объект сообщений n–1 является предыдущим объекта сообщения n, а объект сообщения n+1 – следующим.

Панель команд списка

Для просмотра структуры списка объектов сообщений узла достаточно обратиться к соответствующим регистрам LIST1/LIST2 и MOSTATn.

Структура списка управляется и изменяется посредством контроллера списка, который, в свою очередь, управляется панелью команд, основное назначение которой – упрощение внесения изменений в структуру списка, отслеживание этих изменений и проверка их корректности.

Регистр панели команд PANCTR полностью определяет действия контроллера списка по построению и реорганизации списков объектов сообщений.

Панель команд запускается записью соответствующей команды в битовое поле PANCMD. До записи кода команды в поле PANCMD должны быть записаны соответствующие аргументы команды в битовые поля PANAR1 и PANAR2.

Примечание – Запись новых значений в поля PANAR1 и PANAR2 не изменяет сразу их содержимого. Новые значения сначала попадают в специальный теневой регистр. Далее, одновременно с записью кода команды в поле PANCMD, новые значения из теневого регистра переносятся в поля PANAR1 и PANAR2.

С записью корректного кода команды выставляется флаг BUSY, и в дальнейшем все попытки записи в регистр PANCTR игнорируются. Флаг BUSY остается активным, а панель команд заблокированной до тех пор, пока не завершится выполнение записанной команды.

После сброса микроконтроллера контроллер списка формирует список №0 нераспределенных объектов сообщений. Во время этой операции флаг BUSY установлен, и все обращения к ОЗУ объектов сообщений запрещены. ОЗУ становится доступным только после сброса флага BUSY.

Примечание – После сброса ОЗУ объектов сообщений автоматически инициализируется контроллером списка для обеспечения каждого объекта сообщений корректным указателем на список. По окончании этой операции флаг BUSY сбрасывается.

В случае появления команды динамического распределения, по которой какой-либо элемент забирается из списка №0 и переносится в другой указанный список, наряду с битом BUSY, устанавливается бит RBUSY. Это указывает на то, что значения битовых полей PANAR1 и PANAR2 будут обновлены контроллером списка следующим образом:

- номер объекта сообщения, переносимого из списка № 0 нераспределенных объектов сообщений, записывается в PANAR1;

- если установлен бит ERR (седьмой бит поля PANAR2), значит, список №0 пуст и выполнение команды завершается; если бит ERR сброшен – список №0 не пуст и команда выполняется.

Результаты выполнения команды динамического распределения записываются до того, как контроллер списка начнет процесс распределения. Как только результаты станут доступны, бит RBUSY сбрасывается. Это позволяет пользователю запрограммировать настройки желаемого объекта сообщения, в то время как контроллер списка распределяет объекты. Во время операций со списками доступ к объектам сообщений не запрещен, но следует помнить, что любой доступ к регистрам ОЗУ объектов сообщений в течение процесса распределения объектов вносит задержку (в процесс), равную длительности доступа.

Код команды «нет операции» автоматически записывается в битовое поле PANCMD.

Новая команда может быть записана в любое время, когда бит BUSY сброшен.

Все битовые поля регистра PANCTR, исключая биты BUSY и RBUSY, могут быть записаны программно, что делает возможным сохранять и восстанавливать значения регистра PANCTR, если панель команд используется независимой подпрограммой обработки прерываний. Если возникает такая ситуация, то любые задачи, которые используют панель команд и которые могут прерывать выполнение других задач, тоже использующих панель команд, будут опрашивать состояние флага BUSY. До тех пор, пока флаг BUSY будет оставаться установленным, содержимое регистра PANCTR будет сохранено в соответствующей области памяти до операции восстановления. Как только подпрограмма обработки прерываний закончится, содержимое регистра PANCTR будет восстановлено.

До того, как объект сообщения, занесенный в список активного узла CAN, будет перенесен на другую позицию этого же списка или перенесен в другой список, бит MSGVAL регистра MOSTATn объекта сообщения n должен быть очищен.

Примечание – Если требуется перераспределить объекты сообщений в списки повторно, необходимо приостановить работу узлов CAN (установить бит INIT регистра NCRx), а после занесения объектов в списки возобновить ее (сбросить бит INIT).

16.5 Прием сообщения

После завершения приема сообщение сохраняется в объекте сообщения в соответствии с установленным алгоритмом (см. рисунок 16.19). Помимо сохранения данных в объекте сообщения, модуль CAN осуществляет обмен данными с ЦП.

Бит MSGVAL корректности объекта сообщения

При приеме сообщения информация сохраняется в объекте сообщения только в том случае, если установлен бит MSGVAL регистра MOSTATn. Если ЦП очищает бит MSGVAL, модуль CAN останавливает запись в объект сообщения, и далее объект может быть реконфигурирован центральным процессором с последующей записью в него информации без участия модуля CAN.

Бит RTSEL распределения объекта сообщений

Реконфигурация объекта сообщения центральным процессором во время работы модуля CAN (например, сброс бита MSGVAL, изменение объекта сообщения и повторная установка бита MSGVAL) происходят следующим образом:

- объект сообщения получает приоритет;
- ЦП очищает бит MSGVAL для реконфигурации объекта сообщения;
- после реконфигурации ЦП снова устанавливает бит MSGVAL;
- завершается получение сообщения;
- если установлен бит MSGVAL, полученные данные сохраняются в объекте сообщения, генерируется запрос на прерывание, устанавливается соответствующий флаг;
- если сконфигурировано, производятся шлюзовые и FIFO операции.

Примечание – После реконфигурации объекта сохранение данных по завершении получения сообщения может быть нежелательным. Запретить запись данных в объект сообщения можно посредством бита RTSEL.

После получения объектом сообщения приоритета его бит RTSEL устанавливается контроллером CAN, открывая, таким образом, объект сообщения для записи. После приема сообщения контроллер CAN дополнительно проверяет возможность записи в объект сообщения, а именно – установлен ли все еще бит RTSEL. И только в том случае, если бит RTSEL установлен, полученные данные сохраняются в объекте сообщения (вместе со всеми последующими действиями, которые указаны выше).

Если во время операций контроллера CAN объект сообщения становится некорректным (сброс бита MSGVAL), бит RTSEL должен быть сброшен до того, как бит MSGVAL будет установлен снова, или, по крайней мере, одновременно с ним. Это необходимо для предотвращения сохранения старой информации в объекте сообщения.

Реконфигурация объекта сообщения должна происходить следующим образом:

- сброс бита MSGVAL;
- реконфигурация объекта сообщения, пока бит MSGVAL сброшен;
- сброс бита RTSEL и далее установка бита MSGVAL.

Бит RXEN разрешения приема

Полученное с шины сообщение может быть сохранено в объекте сообщения только в случае, если установлен бит RXEN. Контроллер CAN проверяет состояние бита RXEN только во время фильтрации (см. далее) принимаемого сообщения. После того, как сообщение принято, состояние бита не имеет значения и не оказывает влияния на дальнейшее сохранение данных в объекте сообщения.

Бит RXEN позволяет управлять блокированием объекта сообщения – после сброса бита RXEN полученное сообщение сохраняется в объекте сообщения, который получил приоритет, но в сохранении последующих сообщений этот объект не принимает участия.

Флаги RXUPD, NEWDAT и MSGLST

Индикатором процесса сохранения (изменения) данных в объекте сообщения является флаг RXUPD, который выставляется с началом процесса сохранения (изменения) и сбрасывается с его окончанием.

После сохранения полученного сообщения (идентификатора, бита IDE, кода длины данных, поля данных, в случае сообщения данных) выставляется флаг NEWDAT. Если к моменту выставления (завершение сохранения/изменения данных) флаг NEWDAT был уже установлен, выставляется флаг MSGLST, который говорит о том, что произошла потеря данных.

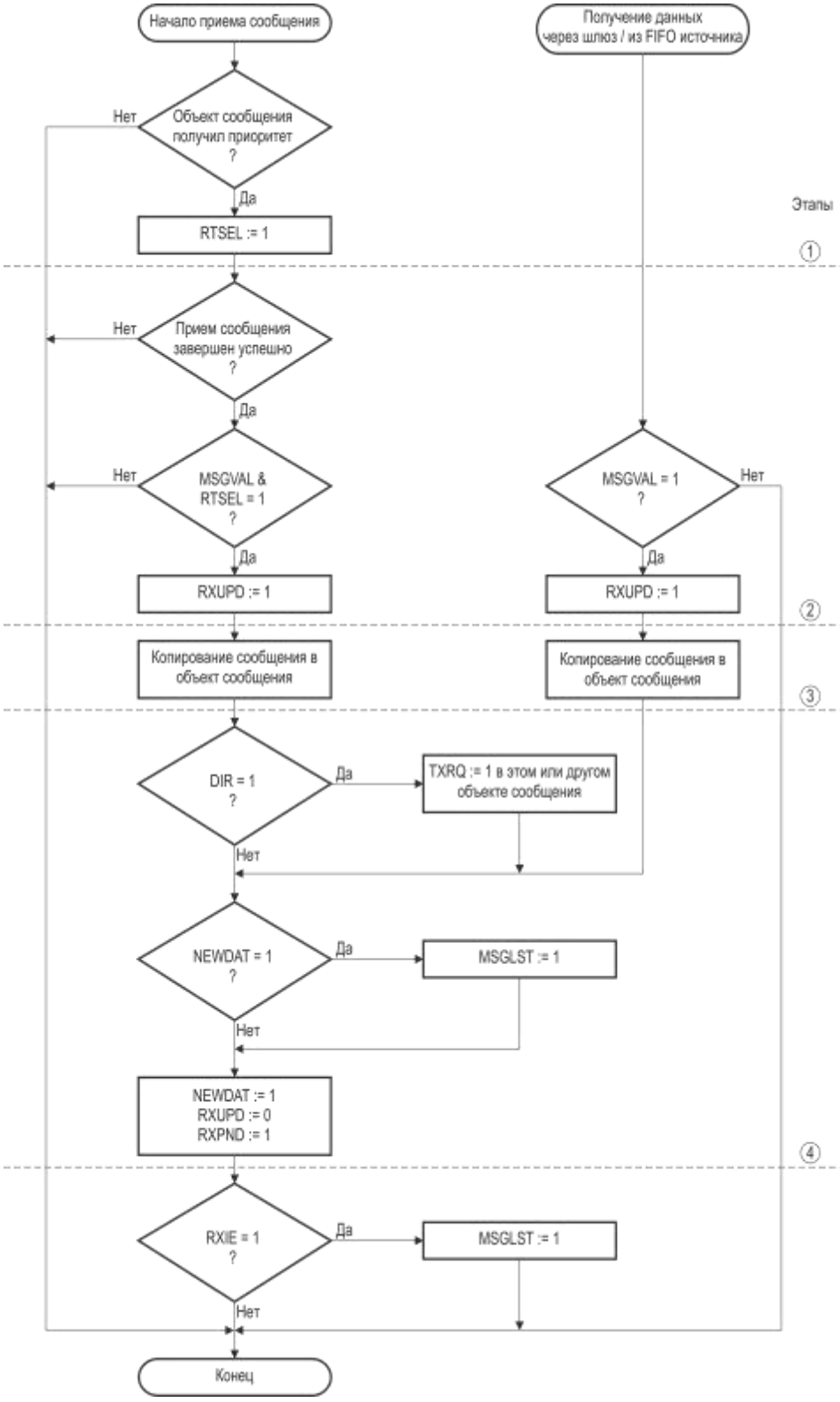


Рисунок 16.19 – Прием сообщения

Флаги RXUPD и NEWDAT позволяют произвести чтение корректных данных из объекта сообщения во время текущих операций контроллера CAN. Рекомендуемая последовательность действий следующая:

- сброс флага NEWDAT;
- чтение данных (идентификатор, данные и т. д.) из объекта сообщения;
- проверка флагов NEWDAT и RXUPD – оба флага должны быть сброшены. В случае невыполнения этого условия возвращение к первому действию;
- если флаги NEWDAT и RXUPD сброшены, то содержимое объекта сообщения корректно и не используется контроллером CAN в течение операции чтения.

Поведение флагов RXUPD, NEWDAT и MSGLST идентично как для сообщений данных, так и для сообщений удаленных запросов.

16.6 Передача сообщения

Алгоритм передачи сообщений показан на рисунке 16.20. Одновременно с копированием данных (идентификатора, бита IDE, бита RTR, равного биту DIR, кода длины данных и собственно данных) из объекта сообщения, содержимое которого должно быть передано во внутренний передающий буфер соответствующего узла CAN, для контроля соблюдения четкой последовательности выполнения всех операций устанавливаются биты состояния.

Биты MSGVAL, TXEN0, TXEN1 и TXRQ

Сообщение может быть передано только в случае, когда все четыре бита установлены.

Бит RTSEL

Бит RTSEL выставляется после того, как объект сообщения получает приоритет для передачи своего содержимого. Когда данные объекта сообщения копируются в передающий буфер, бит RTSEL проверяется, и если он установлен, сообщение передается. После успешной передачи сообщения бит RTSEL проверяется снова, и если он установлен, осуществляются дальнейшие операции.

Для полной и завершенной реконфигурации корректного объекта сообщения должны быть выполнены следующие шаги:

- очистка бита MSGVAL;
- реконфигурация объекта сообщения, пока бит MSGVAL сброшен;
- сброс бита RTSEL и установка бита MSGVAL.

Сброс бита RTSEL гарантирует как полное отключение объекта сообщения от текущей передачи, так и то, что никакие операции (копирование данных в передающий буфер, включая сброс бита NEWDAT, очистка бита TXRQ, прерывание сообщения и т. д.), относящиеся к старой конфигурации этого объекта сообщения, не повлияют на новую конфигурацию после установки бита MSGVAL.

Флаг NEWDAT

После завершения передачи содержимого объекта сообщения в передающий буфер узла CAN, флаг NEWDAT аппаратно сбрасывается, тем самым обозначая, что объект сообщения открыт для записи новых данных.

Если после успешной передачи сообщения (на CAN-шину) флаг NEWDAT все еще остается сброшенным (в объект сообщения не были записаны новые данные), флаг TXRQ аппаратно сбрасывается. Если же флаг NEWDAT был установлен программно (в связи с необходимостью передачи новых данных), флаг TXRQ не сбрасывается, тем самым разрешая передачу новых данных.

Дополнительные режимы передачи

Дополнительно имеются два режима, каждый из которых может быть выбран индивидуально:

- режим передачи данных с защитой от повторений;
- режим однократной пересылки данных.

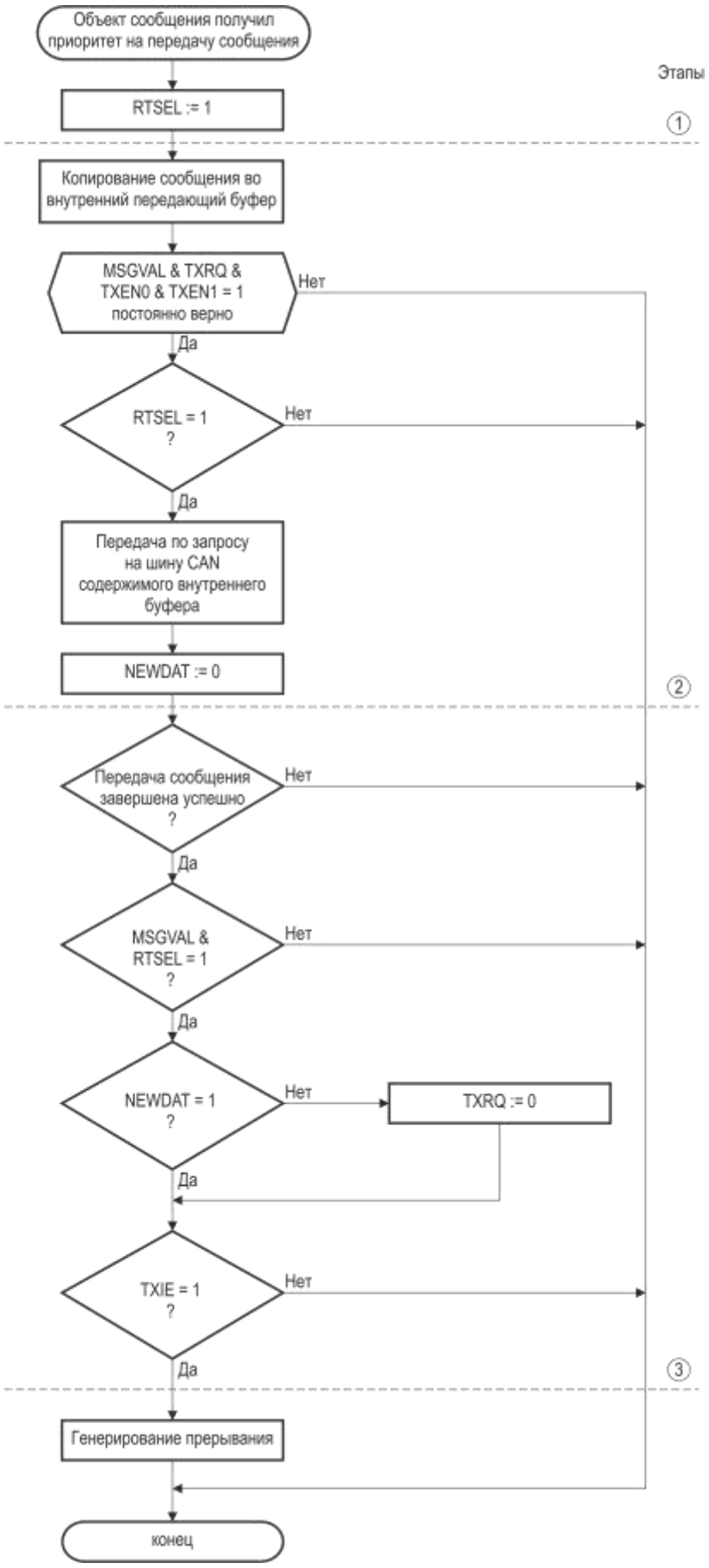


Рисунок 16.20 – Передача сообщения

Режим передачи данных с защитой от повторения

Выбирается установкой бита SDT регистра MOFCRn.

После приема сообщения и сохранения его в выбранном объекте сообщения, бит MSGVAL этого объекта аппаратно сбрасывается, что делает объект некорректным, а, следовательно, недоступным для последующей записи.

После однократной успешной передачи данных бит MSGVAL передающего объекта сообщения будет сброшен, и объект станет недоступным для передачи.

При приеме сообщения удаленного запроса поведение объекта сообщения n, который его принял, зависит от состояния бита FRREN регистра MOFCRn этого объекта.

Если бит FRREN установлен, то в ответ на сообщение удаленного запроса будут переданы данные из объекта, на который указывает поле CUR объекта сообщения, принявшего удаленный запрос. После успешной передачи бит MSGVAL объекта, принявшего удаленный запрос, будет сброшен. Объект сообщения, передавший данные, останется корректным.

Примечание – Если бит FRREN сброшен, то в ответ на сообщение удаленного запроса объект сообщения выполнит передачу собственных данных. По окончании передачи бит MSGVAL этого объекта сообщения не сбросится, т.е. объект останется корректным и может участвовать в дальнейших операциях (защита от повторений не включается).

Режим однократной пересылки данных

Выбирается установкой бита STT регистра MOFCRn.

Бит TXRQ сбрасывается, когда содержимое объекта сообщения копируется в передающий буфер узла CAN. Таким образом, в дальнейшем, при неудачной (вследствие ошибок) пересылке сообщения по CAN-шине, повторной передачи не будет.

16.7 Фильтрация сообщений

Контроллер CAN использует фильтрацию для контроля приема и передачи сообщений.

Фильтрация при получении сообщений

При получении узлом CAN сообщения определяется объект сообщения, в котором будут сохранены получаемые данные в случае успешного приема.

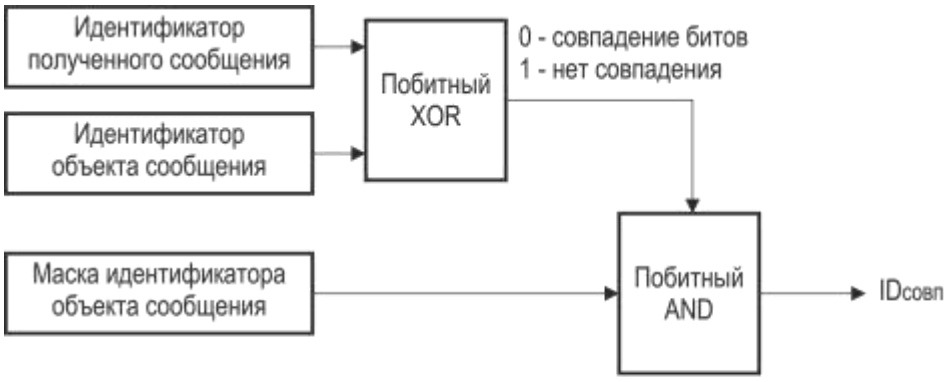


Рисунок 16.21 – Проверка идентификатора полученного сообщения

Объект сообщения считается корректным для приема, если одновременно соблюдаются условия:

-объект сообщения распределен в список объектов сообщений узла, который принимает сообщение;

- бит MSGVAL установлен;

- бит RXEN установлен;

- бит DIR равен биту RTR принимаемого сообщения. Если бит DIR установлен (объект передачи) объект сообщения может принять только сообщение удаленного запроса. Если бит DIR сброшен (объект приема), объект сообщения может принять только сообщение данных;

- если бит MIDE установлен, то бит IDE получаемого сообщения оказывает следующее влияние:

- если бит IDE (регистр MOARn) установлен, то бит IDE принимаемого сообщения должен быть равен единице (расширенный идентификатор);

- если бит IDE сброшен, бит IDE принимаемого сообщения должен быть равен нулю (стандартный идентификатор);

- если бит MIDE сброшен, значение бита IDE принимаемого сообщения не важно, т.е. допускаются сообщения как со стандартным, так и с расширенным идентификатором;

- идентификатор полученного сообщения полностью (побитно) совпадает с идентификатором, хранящимся в регистре MOARn объекта сообщений, за исключением битов, закрытых маской регистра MOAMRn, значение которых не важно. На рисунке 16.21 показан пример проверки идентификатора.

Среди всех объектов сообщений, которые отвечают указанным выше критериям, для сохранения полученного сообщения выбирается объект с наивысшим приоритетом. Для задания приоритета используется поле PRI в регистре MOARn. Объект сообщения, у которого значение поля PRI меньше, имеет больший приоритет. При равенстве значений поля PRI приоритетным считается объект сообщения, который предшествует следующему в списке.

Фильтрация при передаче сообщений

Когда требуется передача содержимого какого-либо объекта сообщения, в соответствующих управляющих регистрах выставляются флаги, указывающие на необходимость передачи. Объект сообщения считается корректным для передачи, если одновременно соблюдаются условия:

- объект сообщения распределен в список объектов сообщений CAN узла;

- флаг MSGVAL установлен;

- флаг TXRQ установлен;

- флаги TXEN0 и TXEN1 установлены.

Может возникнуть ситуация, когда передачи требуют одновременно несколько объектов сообщений. Среди всех объектов, которые отвечают указанным выше критериям, для передачи выбирается объект с наивысшим приоритетом.

Объект сообщения, у которого значение поля PRI (регистр MOARn) меньше, имеет больший приоритет. При равенстве значений поля PRI разных объектов приоритет определяется следующим образом:

- при PRI = 10b – согласно правилам арбитража передачи сообщения;

- при PRI = 01b/11b приоритет имеет объект сообщения, который предшествует следующему в списке.

Объект сообщения, являющийся корректным для передачи и имеющий приоритет, будет осуществлять передачу первым. Остальные объекты сообщений будут переданы по очереди, согласно их приоритетам.

Объект сообщения определяется как стандартный объект сообщения, если в регистре MOFCRn значение битового поля MMC равно нулю. Стандартный объект сообщения может принимать и передавать сообщения, согласно правилам, описанным выше.

На рисунке 16.22 показано формирование запроса на передачу объекта сообщения.

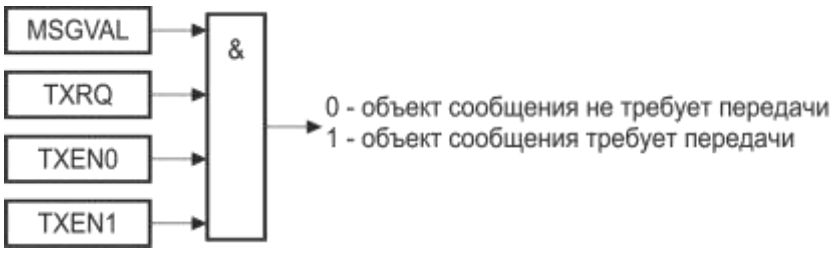


Рисунок 16.22 – Формирование запроса на передачу объекта сообщения

16.8 FIFO структура объектов сообщений

Регистр MOFGPRn объекта сообщения n содержит установки указателей на объекты сообщений, которые используются при операциях FIFO и шлюзовых операциях.

В случае сильной загрузки ЦП обработка серии сообщений может быть затруднена – например, вследствие получения и/или передачи большого числа сообщений за малые промежутки времени. Для таких случаев предусмотрена система буферов быстрого ввода-вывода, так называемая FIFO структура, которая может функционировать автоматически и позволяет избежать потери принимаемых сообщений, минимизировать время подготовки сообщений к отправке, а также генерировать прерывания по окончании операций.

Допускается организация нескольких параллельных FIFO структур. Число структур и их составляющих зависит только от количества доступных объектов сообщений. FIFO структура может быть создана, изменена и удалена в любой момент времени, даже во время операций контроллера CAN.

На рисунке 16.23 представлена основная FIFO структура. Она состоит из одного базового объекта и n-ого числа вспомогательных объектов. Вспомогательные объекты объединяются последовательно в списки (подобно спискам объектов сообщений). Базовый объект может быть занесен в любой список. Хотя на рисунке базовый объект не относится ни к одному из списков, он может быть вставлен в любую последовательность вспомогательных объектов. Это означает, что базовый объект одновременно является и вспомогательным объектом (шлюзовые операции не возможны). Порядковые номера объектов сообщений (0, 1, 2 и т. д.) не имеют никакого значения при FIFO операциях с объектами.

Базовый объект не нуждается в обязательном занесении его в какой-либо список, в отличие от вспомогательных объектов, которые должны быть определены в общий список (так как они последовательно связаны). С помощью указателей (битовые поля BOT, CUR и TOP регистра MOFGPRn) можно присоединять базовый объект к вспомогательным объектам, независимо от того, принадлежат базовый и вспомогательный объекты одному списку или разным спискам.

Минимальная FIFO структура может состоять из одного объекта сообщения, который будет одновременно являться и базовым, и вспомогательным (фактически не используется). Максимальная FIFO структура может включать в себя все 64 объекта сообщений.

В базовом объекте FIFO границы установлены: поле BOT указывает на самый младший элемент FIFO структуры, поле TOP – на самый старший элемент, поле CUR – на вспомогательный объект, который в настоящий момент выбран котроллером CAN для передачи сообщения. Как только начинается передача, в CUR записывается номер следующего по списку вспомогательного объекта сообщения (CUR = PNEXT используемого объекта). Если значение битового поля CUR достигло номера старшего элемента списка (CUR = TOP), то следующим значением будет BOT (реализация автоматического перехода в начало списка). Таким образом, реализуется замкнутая FIFO

структура, в которой битовые поля TOP и BOT устанавливают связь между началом и концом списка.

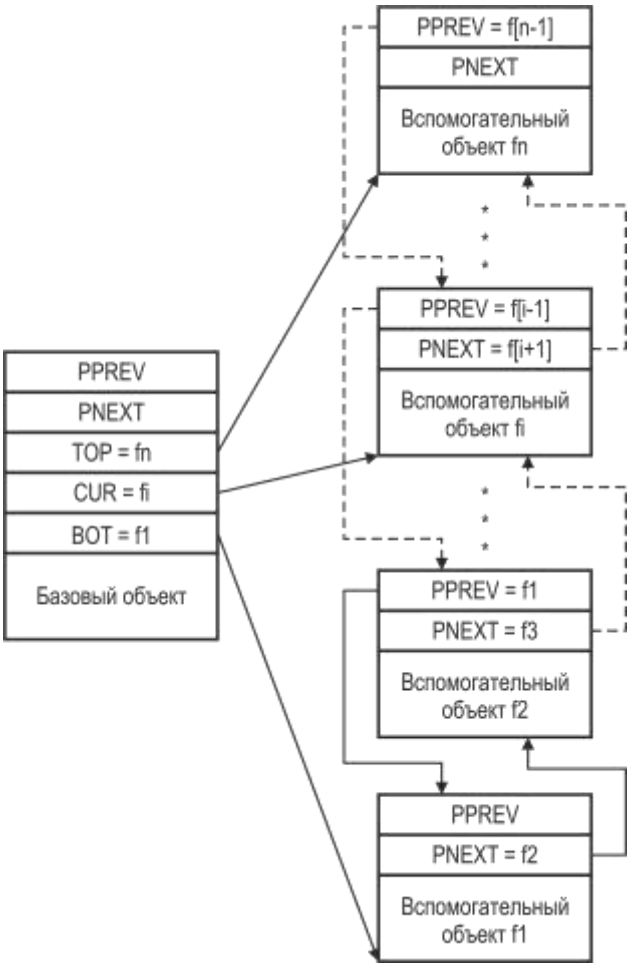


Рисунок 16.23 – FIFO структура с базовым объектом и n вспомогательными объектами

Битовое поле SEL позволяет определить вспомогательный объект в пределах списка, для которого генерируется прерывание всякий раз, когда указатель CUR достигает значения указателя SEL. Также битовое поле SEL позволяет отследить окончание запланированной передачи серии сообщений или выдать прерывание, предупреждающее о том, что FIFO структура становится заполненной.

FIFO структура для приема

FIFO структура для приема используется для буферизации входящих сообщений данных и удаленных запросов.

FIFO структура для приема активируется записью значения 0001b в битовое поле MMC регистра MOFCRn базового объекта. Эта запись автоматически определяет объект как базовый объект приема FIFO. Типы вспомогательных объектов FIFO не имеют значения при операциях.

Когда базовый объект FIFO получает сообщение от узла CAN, которому он принадлежит, сообщение сохраняется не в этом базовом объекте, а во вспомогательном объекте сообщения, на который указывает битовое поле CUR. При этом по умолчанию предполагается, что для вспомогательного объекта MMC = 0000b (действительное значение MMC игнорируется), и никаких операций фильтрации принимаемого сообщения не производится.

Одновременно с приемом сообщения текущее значение указателя CUR базового объекта меняется на номер следующего по списку вспомогательного объекта FIFO

структуры. Этот вспомогательный объект будет использован для приема следующего сообщения.

Если установлен флаг OVIE регистра MOFCRn базового объекта и значение указателя CUR становится равным значению указателя SEL, генерируется прерывание переполнения. Это прерывание генерируется на узле прерываний с указателем TXINP базового объекта сразу после сохранения полученного сообщения во вспомогательном объекте. Прерывания генерируются, если это разрешено битом TXIE.

Следует помнить, что сообщение сохраняется в базовом и вспомогательном объектах FIFO, только если установлен бит MSGVAL.

Во избежание непосредственного приема сообщения вспомогательным объектом, как если бы он был независимым объектом и не принадлежал FIFO структуре, флаги RXEN всех вспомогательных объектов должны быть сброшены. Состояние флага RXEN не важно в случае, когда вспомогательный объект занесен в список, не связанный с узлом CAN.

FIFO структура для передачи

FIFO структура для передачи используется для буферизации серий сообщений данных или удаленных запросов, которые должны быть отправлены. FIFO структура для передачи состоит из базового объекта и одного или более вспомогательных объектов.

FIFO структура для передачи активируется записью значения 0010b в поле MMC регистра MOFCRn базового объекта. В отличие от FIFO структуры для приема, в битовые поля MMC вспомогательных объектов (FIFO структуры для передачи) должно быть записано значение 0011b. Указатели CUR всех вспомогательных объектов должны указывать на базовый объект FIFO передачи (чтобы инициализироваться программно).

Флаги TXEN1 всех вспомогательных объектов сообщений, за исключением одного, на который указывает указатель CUR базового объекта, должны быть программно сброшены. Флаг TXEN1 указанного объекта должен быть установлен. Указатель CUR базового объекта может быть инициализирован для любого вспомогательного объекта.

При определении корректности объектов сообщений FIFO структуры для начала FIFO-операций базовый объект должен быть определен первым как корректный, т.е. MSGVAL должен быть установлен.

В случае необходимости удаления FIFO структуры, прежде, чем начнется операция удаления, все вспомогательные объекты, принадлежащие этой FIFO структуре, должны быть определены как некорректные (биты MSGVAL должны быть сброшены).

FIFO структура для передачи использует флаги TXEN1 всех своих объектов для выбора сообщения для передачи. В результате фильтрации право передавать сообщение получает тот объект, у которого выставлен флаг TXEN1. После передачи сообщения флаг TXEN1 аппаратно сбрасывается, а в указатель CUR записывается номер следующего объекта, требующего отправки сообщения, для которого уже выставлен (аппаратно) свой флаг TXEN1 и так далее для всей FIFO структуры.

Если установлен флаг OVIE регистра MOFCRn базового объекта и значение указателя CUR становится равным значению указателя SEL, генерируется прерывание переполнения. Это прерывание генерируется на узле прерываний с указателем RXINP базового объекта после завершения операций получения сообщения. Прерывания приема базового объекта генерируются, если это разрешено битом RXIE.

Рекомендации по программированию регистров для FIFO структуры

1 Для передающего базового объекта:

- сбросить бит MSGVAL;
- задать поля CUR, BOT, TOP, SEL;
- записать значение 0010b в поле MMC, задать DLC, установить биты OVIE и RXIE (если необходимо).

Примечание – Состояние регистров MOARn и MOAMRn передающего базового объекта не важно, поскольку в передаче участвуют передающие вспомогательные

объекты и принимающий базовый объект. Поле RXINP указывает линию, на которую будет выдаваться прерывание переполнения (CUR = SEL).

2 Для передающих вспомогательных объектов:

- сбросить бит MSGVAL;

- установить биты DIR, TXEN1 (только для того вспомогательного объекта, на который указывает поле CUR передающего базового объекта, у остальных вспомогательных объектов бит TXEN1 должен быть сброшен), TXEN0;

- записать в поле CUR номер передающего базового объекта;

- записать значение 0011b в поле MMC, задать DLC.

Примечание – Значения регистров MOARn передающих вспомогательных объектов должно совпадать со значением регистра MOAR принимающего базового объекта, так как процесс передачи фактически происходит между ними (или иного принимающего объекта, если на приеме используется не FIFO структура).

3 Для принимающего базового объекта:

- установить бит RXEN;

- задать поля CUR, BOT, TOP, SEL;

- записать значение 0001b в поле MMC, задать DLC, установить биты OVIE и TXIE (если необходимо).

Примечание – Значение регистра MOARn принимающего базового объекта должно быть равно значению регистров MOARn передающих вспомогательных объектов передачи. Поле TXINP указывает, на какую линию будет выдаваться прерывание переполнения (прерывание после операции сохранения полученного сообщения во вспомогательных объектах при CUR = SEL).

4 Для принимающих вспомогательных объектов:

- сбросить бит RXEN (не требуется, если вспомогательные объекты занесены в список, не связанный с узлом CAN);

- задать поле DLC (состояние поля MMC не важно).

Примечание – Состояние регистров MOARn принимающих вспомогательных объектов не важно.

5 Установить бит MSGVAL, в первую очередь, у передающего базового объекта, а затем у всех остальных объектов.

6 Установить бит TXRQ для всех передающих вспомогательных объектов, начиная с того, на который указывает поле CUR передающего базового объекта.

16.9 Шлюзы

Режим позволяет реализовывать автоматическую передачу информации через шлюз между двумя независимыми шинами CAN без участия ЦП.

Шлюз можно сформировать на уровне объектов сообщений и осуществлять передачу информации между узлами CAN. Шлюз может быть сформирован между двумя любыми объектами сообщений, принадлежащими разным узлам CAN. Количество шлюзов зависит только от количества объектов сообщений, допускающих формирование шлюзов.

Режим шлюза активируется записью значения 0100b в битовое поле MMC регистра MOFCRn объекта сообщения n и инициализирует его как шлюзовый объект-источник. Объект сообщения, который будет являться шлюзовым объектом-приемником, выбирается указателем CUR объекта-источника. Для формирования шлюза достаточно, чтобы объект-приемник был корректным (установлен бит MSGVAL). Остальные параметры не влияют на возможность осуществления передачи между объектами от источника к приемнику.

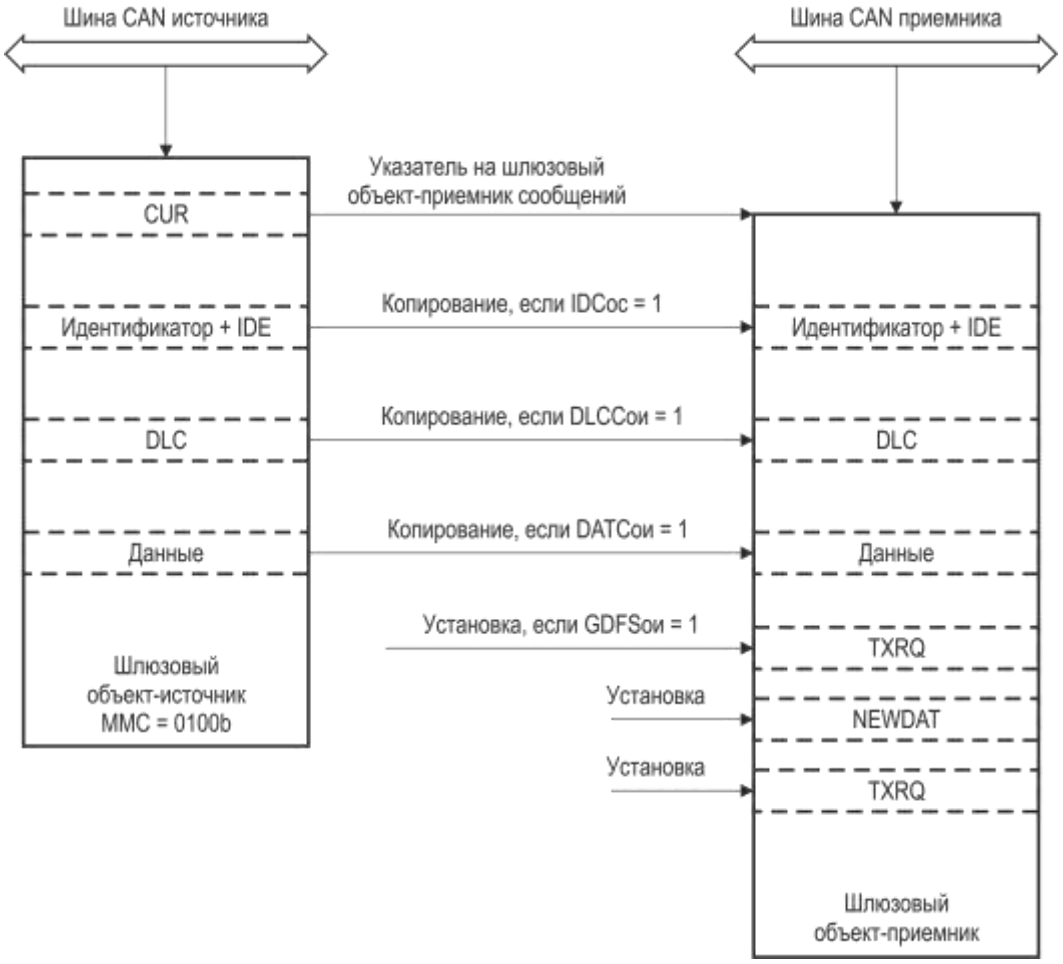


Рисунок 16.24 – Передача через шлюз от источника к приемнику

Шлюзовый объект-источник (см. рисунок 16.24) функционирует как обычный объект сообщений с тем отличием, что возможны дополнительные действия контроллера CAN при приеме и сохранении сообщения в объекте-приемнике:

1 Если установлен флаг DLCC регистра MOFCRn объекта-источника, код длины данных DLC копируется из шлюзового объекта-источника в шлюзовый объект-приемник.

2 Если установлен флаг IDC объекта-источника, идентификатор ID и расширение IDE копируются из шлюзового объекта-источника в шлюзовый объект-приемник.

3 Если установлен флаг DATC объекта-источника, байты данных, хранящиеся в двух регистрах MODATALn и MODATAHn объекта-источника, копируются из шлюзового объекта-источника в шлюзовый объект-приемник. Копируются все 8 байт данных, вне зависимости от значения поля DLC.

4 Если установлен флаг GDFS объекта-источника, то устанавливается бит запроса передачи TXRQ объекта-приемника.

5 Устанавливаются флаги RXPND и NEWDAT регистра MOSTATn объекта-приемника.

6 Если установлен флаг RXIE регистра MOSTATn объекта-приемника, то генерируется запрос на прерывание.

7 Указатель CUR объекта-источника переводится на следующий объект-приемник по правилам FIFO структуры. Сформировать шлюз между объектом-источником и одним объектом-приемником (значение указателя CUR будет оставаться неизменным) возможно программированием:

TOP = BOT = CUR = номер объекта-приемника.

Организация шлюза «объект-источник – объект-приемник» аналогична организации FIFO структуры «базовый объект-вспомогательный объект», что указывает на возможность формирования шлюза с интегрированным FIFO-приемником. На рисунке 16.24 объект сообщения слева – шлюзовый объект-источник, а объект сообщения справа – шлюзовый объект-приемник.

При получении сообщения данных (объект-источник является объектом приема, т.е. его бит DIR сброшен) и при получении удаленного запроса (объект-источник является объектом передачи) через шлюз используется один и тот же механизм.

Удаленные запросы

После получения узлом CAN сообщения удаленного запроса и сохранения его в объекте сообщения выставляется бит запроса передачи для ответа на удаленный запрос (отправка сообщения данных) или для автоматического повторения запроса. Если сброшен бит FRREN объекта сообщения, в который было сохранено сообщение удаленного запроса, то выставляется флаг TXRQ этого объекта.

У объекта сообщений, передающего сообщение удаленного запроса, должны быть выставлены биты регистра MOSTATn следующим образом: DIR = 0 (объект передает сообщение удаленного запроса), TXEN0 = 1, TXEN1 = 1, далее MSGVAL = 1, TXRQ = 1. Значение идентификатора, содержащегося в регистре MOARn передающего объекта сообщений, должно быть равно значению идентификатора (или подходить, если есть маскируемые биты идентификатора у принимающего объекта, установленные в регистре MOAMRn) принимающего объекта сообщений, чтобы сообщение удаленного запроса было принято принимающим объектом другого узла. Само сообщение удаленного запроса должно содержать идентификатор принимающего объекта сообщений, поэтому значение регистра MODATALn передающего объекта сообщений должно быть равно значению регистра MOARn принимающего объекта.

У объекта сообщений, принимающего сообщение удаленного запроса, должны быть выставлены биты регистра MOSTATn следующим образом: DIR = 1 (объект принимает сообщение удаленного запроса); TXEN0 = 1 и TXEN1 = 1 (если отвечать на запрос будет сам); RXEN = 1; далее MSGVAL = 1. Регистры объекта MODATALn и MODATAHn должны содержать данные, которые будут выдаваться в ответ на запрос.

Если установлен бит FRREN объекта сообщения, который принял сообщение удаленного запроса, то флаг TXRQ устанавливается у того объекта, на который указывает поле CUR объекта, принявшего удаленный запрос. При этом указатель CUR не меняет своего значения.

Последовательность записи в регистры объекта сообщений, передающего сообщение удаленного запроса, аналогична указанной выше.

У объекта сообщений, принимающего сообщение удаленного запроса, должны быть выставлены биты регистра MOSTATn следующим образом: DIR = 1 (объект принимает сообщение удаленного запроса), RXEN = 1, далее MSGVAL = 1. Битовое поле CUR регистра MOFGPRn должно указывать на номер объекта сообщения (должен находиться в том же узле, что и принявший сообщение удаленного запроса объект), содержащего данные, предназначенные для передачи в ответ на поступивший удаленный запрос.

У объекта сообщений, хранящего данные для отправки в ответ на запрос, должны быть установлены следующие биты регистра MOSTATn: DIR = 1 (объект передает сообщение данных), TXEN0 = 1, TXEN1 = 1, далее MSGVAL = 1. Бит TXRQ регистра MOSTATn объекта сообщения, хранящего данные для отправки в ответ на запрос, устанавливается автоматически при приеме сообщения удаленного запроса принимающим объектом сообщения.

Прием ответа на запрос (переданного сообщения данных) осуществляется стандартным объектом сообщения запрашивающего узла CAN (обмен данными происходит между объектом сообщения, хранящим данные для отправки в ответ на запрос, и объектом сообщения запрашивающего узла CAN).

Несмотря на то, что механизм удаленных запросов работает независимо от типа объекта сообщения, он наиболее полезен при использовании шлюзов, для формирования удаленных запросов на шине шлюзового объекта-источника после получения удаленного запроса на шине шлюзового объекта-приемника. В зависимости от значения бита FRREN шлюзового объекта-приемника, есть два варианта обработки удаленного запроса, возникшего с той стороны шлюза, где расположен объект-приемник (при условии, что происходит передача из объекта-источника в объект-приемник, т. е. DIR (источника) = 0 и DIR (приемника) =1).

1 Обработка запроса шлюзового объекта-приемника с FRREN = 0b:

- сообщение удаленного запроса принимается шлюзовым объектом-приемником;
- бит TXRQ шлюзового объекта-приемника устанавливается автоматически;
- сообщение данных с текущей информацией, хранящейся в объекте-приемнике, передается на шину приемника.

2 Обработка запроса шлюзового объекта-приемника с FRREN = 1b:

- сообщение удаленного запроса принимается шлюзовым объектом-приемником;
- бит TXRQ шлюзового объекта-источника (объект должен быть указан в поле CUR объекта-приемника), устанавливается автоматически;
- сообщение данных передается объектом-источником на шину CAN источника;
- получатель удаленного запроса в ответ выдает сообщение данных на шину источника;
- сообщение данных сохраняется в объекте-источнике;
- сообщение данных копируется в объект-приемник (через шлюз);
- выставляется бит TXRQ объекта-приемника (при условии, что $GDFS_{источника} = 1$);
- новые данные, сохраненные в объекте-приемнике, передаются на шину приемника, в ответ на удаленный запрос на шине приемника.

Рекомендации по записи в регистры в режиме шлюза при передаче удаленного запроса с FRREN = 1.

Обмен запрос-данные происходит в данном случае между стандартным объектом сообщений одного узла и объектом-приемником шлюза другого узла. Но при этом данные для ответа на запрос в шлюзовый объект-приемник поступают по шлюзу от объекта-источника. При получении удаленного запроса от объекта сообщения объектом-приемником флаг TXRQ устанавливается не у самого объекта-приемника, а у объекта-источника, благодаря установленному биту FRREN и битовому полю CUR (указывает на объект-источник) объекта-приемника. Данные из MODATALn и MODATAHn объекта-источника копируются в MODATALn и MODATAHn объекта-приемника (установлен бит DATC регистра MOFCRn объекта-источника), вследствие чего автоматически устанавливается бит TXRQ регистра MOCTRn объекта-приемника (установлен бит GDFS объекта-источника шлюза), и осуществляется передача сообщения данных (ответ на запрос) запрашивающему объекту сообщения.

После успешного приема/передачи сообщения ЦП получает уведомление о завершении операции для задания дальнейших действий, связанных с объектом сообщения.

16.10 Прерывания объектов сообщений

После сохранения принятого сообщения в объект сообщения или успешной передачи формируется соответствующее прерывание. Каждый объект сообщения может формировать прерывания. Каждое прерывание направляется на одну из трех выходных линий прерываний. Прерывания приема (после сохранения сообщения) также формируются после операций FIFO и шлюзовых операций. Флаги TXPND и RXPND всегда устанавливаются после успешной операции передачи/приема, независимо от состояния соответствующих флагов разрешения прерываний.

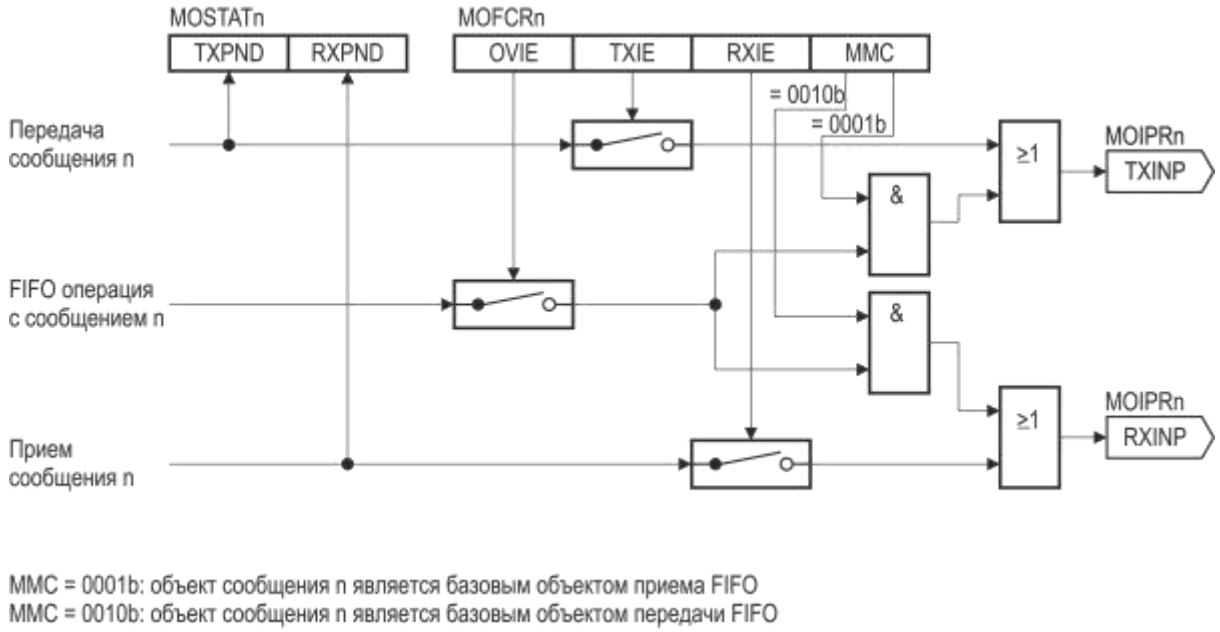


Рисунок 16.25 – Распределение прерываний

Объект сообщения может формировать FIFO прерывания. Если флаг OVIE регистра MOFCRn установлен, то формирование FIFO прерывания будет зависеть от типа объекта сообщений (см. рисунок 16.25):

- если объект сообщения является принимающим базовым объектом, то выходная линия прерываний для этого объекта определяется битовым полем TXINP регистра MOIPRn;
- если объект сообщения является передающим базовым объектом, то выходная линия прерываний определяется битовым полем RXINP.

Ждущие сообщения

Когда генерируется запрос на прерывание (после приема/передачи сообщения), в регистре ждущих прерываний MSPND0 выставляется флаг ждущего сообщения. Каждому объекту сообщений выделено два бита регистра – бит для операций приема и бит для операций передачи. Позиция флага ждущего сообщения определяется демультиплексорами DMUX.

В зависимости от значения поля MPSEL регистра MCR, реализуется один из двух режимов выбора и установки флагов ждущих сообщения (см. рисунок 16.26):

- Режим 1 в случае MPSEL = 0h;
- Режим 2 в случае MPSEL = Fh;

Если нет необходимости в определении источника прерывания (прием или передача сообщения), то можно использовать любой из двух режимов, в противном случае, следует использовать второй режим.

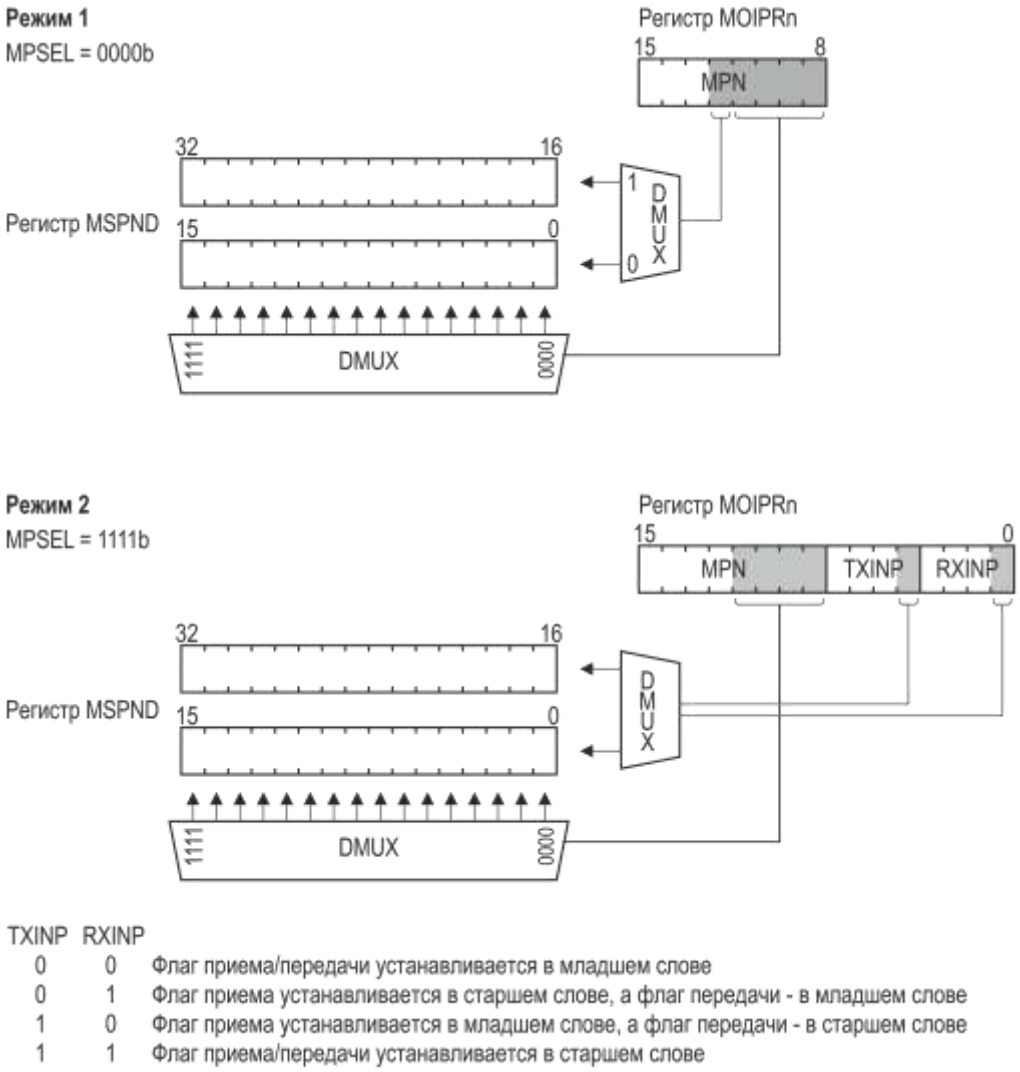


Рисунок 16.26 – Два режима выбора и установки флагов ждущих сообщений

В первом режиме установка флага ждущего сообщения происходит следующим образом:

- 5 бит поля MPN выбирает старшее или младшее слово регистра MSPND0, в котором будет установлен флаг ждущего сообщения;

- четыре младших бита поля MPN (на рисунке 16.26 выделены серым цветом) выбирают позицию флага (от 0 до 31), который будет установлен в регистре MSPND0.

Во втором режиме при определении позиции флага ждущего сообщения принимаются в расчет значения поля MPN и полей RXINP (для приема) и TXINP (для передачи). При этом для флагов могут использоваться любые биты выбранного регистра MSPNDx. Установка флага ждущего сообщения происходит следующим образом:

- четыре младших бита поля MPN (на рисунке 16.26 выделены серым цветом) совместно с нулевыми битами полей TXINP и RXINP выбирают позицию флага (от 0 до 31). Фактически нулевой бит поля TXINP/RXINP выбирает старшее или младшее слово регистра MSPND0, а четыре бита поля MPN задают позицию в выбранном слове.

Регистр MSPND0 может быть записан программно. Биты, в которые записываются единицы, остаются без изменений, а биты, в которые записываются нули, очищаются. Такой механизм записи позволяет избежать конфликта между одновременной аппаратной установкой и программной очисткой битов регистра.

Регистр MSPND0 связан с регистром индекса сообщения MSID0, который отражает позицию самого младшего бита из всех установленных в регистре MSPND0. Регистр

MSID0 доступен только для чтения и обновляется незамедлительно после изменения (как аппаратного, так и программного) содержимого регистра MSPND0.

Регистр маски индекса сообщения MSIMASK содержит маску для регистра MSPND0. Только незакрытые маской биты могут обслуживаться. Регистр MSIMASK используется одновременно для регистров MSPND0 и MSID0.

16.11 Рекомендации по программированию модуля CAN

Для корректного запуска и работы контроллера и конфигурирования его узлов следует соблюдать порядок программирования регистров:

- записать регистр CLC;
- записать регистр FDR;
- в регистре NCRx установить биты INIT и CCE, после чего регистры NBTRx и NPCRx станут доступны для записи и чтения, а регистр NECNTx – только для чтения;
- записать регистр NPCRx;
- записать регистр NIPRx;
- записать регистр NBTRx;
- записать регистр NFCRx (если необходимо);
- в регистре NCRx сбросить биты INIT и CCE, после чего регистры NBTRx и NPCRx будут не доступны для записи.
- при необходимости, записать регистры MSIMASK, MCR и MSPND;
- записать регистр PANCTR.

Для корректной работы объектов сообщений регистры каждого из них в обязательном порядке должны быть проинициализированы. Исключение составляют объекты сообщений, использование которых не предусматривается. Для таких объектов в регистре MOCTRn бит MSGVAL должен оставаться сброшенным.

Примерный порядок инициализации регистров объекта сообщений n:

- установить бит DIR в регистре MOSTATn для передачи сообщения данных/приема удаленного запроса или сбросить бит DIR для приема сообщения данных/передачи удаленного запроса; установить биты TXEN0 и TXEN1 (для передачи) или RXEN (для приема) в регистре MOCTRn;
- записать регистр MOARn;
- записать регистр MOAMRn (если необходимо);
- записать регистр MOFCRn;
- записать регистр MOFGPRn (если будут использоваться FIFO структуры);
- записать регистр MOIPRn;
- записать регистры MODATALn и MODATAHn;
- распределить объекты сообщений в списки с помощью регистра PANCTR;
- установить бит MSGVAL корректности объекта сообщения в регистре MOCTRn (для неиспользуемых объектов этот бит должен быть сброшен);
- для активирования передачи установить бит TXRQ регистра MOCTRn.

Примечание – Нарушение последовательности программирования регистров перед началом работы может привести к некорректной работе узлов или невозможности функционирования контроллера CAN в целом.

17 Генератор псевдослучайных последовательностей (ПСП)

Проблема кодирования больших сообщений и потоков данных появилась сравнительно недавно – с появлением средств мультимедиа и сетей с высокой пропускной способностью, обеспечивающих передачу мультимедийных данных.

В современных ИС и информационных системах начинают применяться технологии, которые требуют передачи больших объемов данных. Это немыслимо без использования современных технологий кодирования.

Наиболее распространенным является потоковое кодирование данных. Система защиты не ждет, когда закончится передаваемое сообщение, а сразу же осуществляет его кодирование и передачу.

Наиболее очевидным является побитовое сложение входящей последовательности (сообщения) с некоторым бесконечным или периодическим ключом, получаемым, например, от генератора псевдослучайных последовательностей (ПСП). Такой генератор псевдослучайной битовой последовательности реализован в микроконтроллере 1874ВЕ8Т на основе сдвиговых регистров с линейной обратной связью.

Принципы построения схем генерации псевдослучайных последовательностей

Сдвиговый регистр представляет собой последовательность битов. Количество битов определяется длиной сдвигового регистра. Если длина равна n битам, то регистр является n-битовым сдвиговым регистром. Всякий раз, когда нужно извлечь бит, все биты сдвигового регистра сдвигаются вправо на одну позицию. Новый крайний левый бит является функцией всех остальных битов регистра. На выходе сдвигового регистра оказывается один бит (обычно младший значащий). Периодом сдвигового регистра называется длина получаемой последовательности до начала ее повторения.

Простейшим примером является линейный сдвиговый регистр с обратной связью (LFSR). Обратная связь представляет собой просто функцию «исключающее ИЛИ» некоторых битов регистра, перечень этих битов называется отводной последовательностью. Иногда такой регистр называется конфигурацией Фиббоначи.

Для того, чтобы конкретный LFSR имел максимальный период, многочлен, образованный из отводной последовательности и константы 1, должен быть примитивным по модулю 2. Степень многочлена является длиной сдвигового регистра.

Сами по себе LFSR являются хорошими генераторами псевдослучайных последовательностей, но они обладают некоторыми нежелательными неслучайными свойствами. Последовательные биты линейны, что делает их бесполезными для шифрования. Для LFSR длины n внутреннее состояние представляет собой предыдущие n выходных битов генератора. Даже если схема обратной связи хранится в секрете, она может быть определена по 2n выходным битам генератора с помощью алгоритма Берлекэмпа — Мэсси.

Кроме того, большие случайные числа, генерируемые с использованием идущих подряд битов этой последовательности, сильно коррелированны и для некоторых типов приложений вовсе не являются случайными. Несмотря на это, LFSR часто используются для создания алгоритмов кодирования.

Основной подход при проектировании генератора потока ключей на базе LFSR прост. Сначала берется один или несколько LFSR, обычно с различными длинами и различными многочленами обратной связи. Если длины взаимно просты, а все многочлены обратной связи примитивны, то у образованного генератора будет максимальная длина. Ключ является начальным состоянием регистров LFSR. Бит выхода представляет собой нелинейную функцию некоторых битов регистров LFSR. Эта функция называется комбинирующей функцией, а генератор в целом – комбинационным генератором.

Можно ввести ряд усложнений. В некоторых генераторах для различных LFSR используется различная тактовая частота, иногда частота одного генератора зависит от

выхода другого. Управление тактовой частотой может быть с прямой связью, когда выход одного LFSR управляет тактовой частотой другого LFSR, или с обратной связью, когда выход одного LFSR управляет его собственной тактовой частотой.

Хотя все эти генераторы чувствительны, по крайней мере, теоретически, к вскрытиям вложением и вероятной корреляцией, многие из них безопасны до сих пор.

В микроконтроллере 1874ВЕ8Т генератор ПСП реализован по типу – двусторонний генератор «стоп – пошел».

В этом генераторе используется два LFSR с одинаковой длиной n. Выходом генератора является «исключающее ИЛИ» выходов каждого LFSR. Если выход LFSR-1 в момент времени t-1 равен нулю, а в момент времени t-2 – единице, то LFSR-2 не тактируется в момент времени t. Наоборот, если выход LFSR-2 в момент времени t-1 равен «0», а в момент времени t-2 – «1», и если LFSR-2 тактируется в момент времени t, то LFSR – 1 не тактируется в момент времени t.

Управление генератором псевдослучайных последовательностей

В качестве линейных сдвиговых регистров с обратной связью выбраны 16-разрядные регистры с отводной последовательностью, обеспечивающей максимальный период такого генератора.

Управление генератором ПСП осуществляется с помощью четырех регистров SFR: PRNGREG1, PRNGREG2, PRNGCON и PRNGRES.

Регистры PRNGREG1 и PRNGREG2 перед началом работы генератора необходимо проинициализировать значениями (отличными от нуля), которые и будут являться ключом. По одному и тому же ключу генератор будет выдавать одинаковую последовательность битов. Регистр PRNGRES является регистром, с которого можно снимать выходную псевдослучайную последовательность битов в виде слов, байтов или битов.

Для запуска кодирования следует установить бит RUN в регистре PRNGCON. Его надо устанавливать каждый раз, чтобы выполнилось необходимое число сдвигов. Количество сдвигов задается полем NUMSHIFTS.

Для повышения защиты кодируемой информации рекомендуется совместно со встроенным генератором комбинировать собственные программные реализации LFSR и подобных генераторов. Это приведет к усложнению схемы данного генератора и к увеличению его периода, что в свою очередь расширит область его применения. Несмотря на то, что программные реализации LFSR медленны, существуют определенные приемы, позволяющие увеличить производительность. Например, рекомендуется писать подобные генераторы на ассемблере, а не на языке С. Также, чтобы вычислить новый левый бит для LFSR за одну операцию, возможно использование битовых масок, которые позволят работать со словами размерности регистров, а не делать это за несколько операций для каждого бита в отдельности.

При использовании генератора ПСП не для кодирования, а, например, при измерениях, встроенный генератор вполне может удовлетворить запросы без дополнительных методов повышения защиты.

18 Таймеры и модуль высокоскоростного ввода-вывода HSIO

Два 16-разрядных счетчика – таймер 1 и таймер 2 – и два модуля – модуль высокоскоростного ввода HSI и модуль высокоскоростного вывода HSO, входящие в состав микроконтроллера – формируют единый модуль HSIO, который может вести подсчет импульсов и импульсных последовательностей, измерять ширину поступающих импульсов, формировать импульсные последовательности (в том числе с ШИМ) и генерировать прерывания.

18.1 Таймер 1

Независимый 16-разрядный однонаправленный счетчик, инкрементирующийся в начале каждого восьмого машинного цикла. Регистр TIMER1 содержит значение счетчика таймера и всегда доступен для записи/чтения. Инициализация таймера 1 происходит после записи в регистр TIMER1 значения, отличного от нуля. Если в дальнейшем таймер функционирует совместно с модулем HSI и/или HSO, то следует уделять большое внимание моменту программирования регистра таймера, поскольку это может вносить изменения во временные соотношения между ожидаемыми и/или запрограммированными событиями. Запущенный счетчик таймера 1 может быть остановлен только сбросом микроконтроллера. Функциональная схема блока таймера 1 показана на рисунке 18.1.

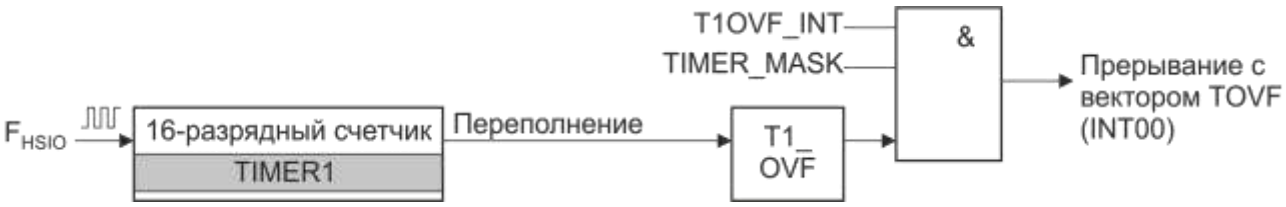


Рисунок 18.1 – Функциональная схема блока таймера 1

Бит T1OVF_INT регистра IOC1 и бит TIMER_MASK регистра INT_MASK управляют формированием запросов на прерывания по переполнению таймера. Индикатором запроса на прерывание является бит TIMER_PEND регистра INT_PEND. Флагом переполнения является бит T1_OVF регистра IOS1 (следует помнить, что чтение регистра IOS1 сбрасывает его биты с пятого по нулевой). Прерыванию соответствует вектор TOVF (INT00).

18.2 Таймер 2

Независимый 16-разрядный реверсивный счетчик с блоком выбора источника синхронизации и регистром захвата. Таймер может функционировать в режиме нормального счета, переключаясь с каждым восьмым тактом синхросигнала, или в режиме скоростного счета – с каждым тактом. Максимальная скорость счета таймера – один машинный цикл, т.е. два такта внешней тактовой частоты микроконтроллера.

Регистр TIMER2 содержит значение счетчика таймера и всегда доступен для записи/чтения. Инициализация таймера происходит после записи в регистр TIMER2 значения, отличного от нуля. Если в дальнейшем таймер функционирует совместно с модулем HSO, то следует уделять большое внимание моменту программирования регистра таймера, поскольку это может вносить изменения во временные соотношения между запрограммированными событиями. Запущенный счетчик таймера 2 может быть остановлен только сбросом микроконтроллера, в тоже время обнуление счетчика можно контролировать. Счетчик таймера обнуляется аппаратно по переполнению, при возникновении запрограммированного события, или может быть обнулен в нужный

момент программно. Регистр захвата T2CAPTURE служит регистром сохранения значения таймера 2, которое находится в его счетчике на момент появления положительного фронта сигнала на выводе P2.7 (T2CAP) микроконтроллера. Сигнал на входе T2CAP должен оставаться без изменений не менее длительности двух тактов сигнала CLOCKOUT. Регистр T2CAP всегда доступен для чтения и записи. Функциональная схема блока таймера 2 показана на рисунке 18.2.

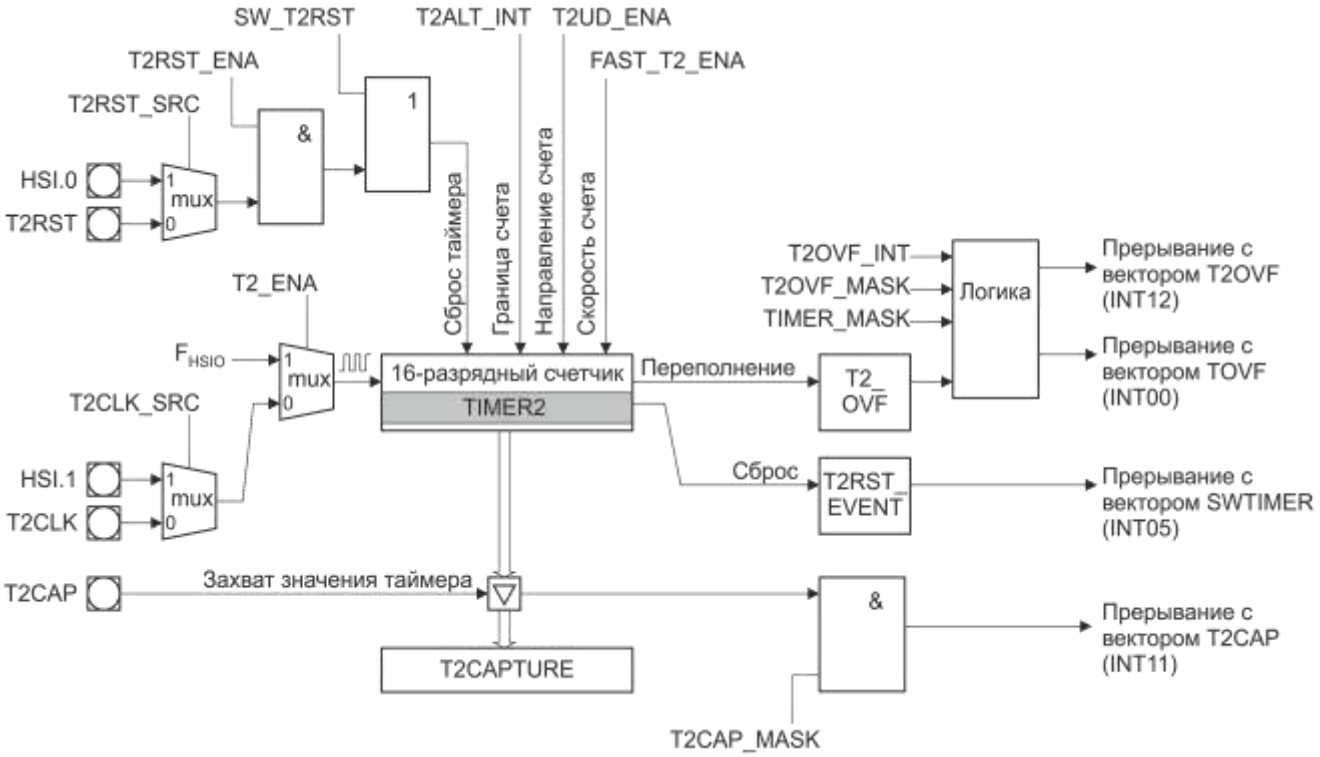


Рисунок 18.2 – Функциональная схема блока таймера 2

Источник синхронизации таймера 2 задается битами T2_ENA и T2CLK_SRC регистров IOC3 и IOC0 соответственно.

Граница переполнения/опустошения таймера (FFFFh/0000h или 7FFFh/8000h), направление и режим счета задаются битами T2ALT_INT, T2UD_ENA и FAST_T2_ENA регистра IOC2.

Биты T2RST_ENA, T2RST_SRC и SW_T2RST регистра IOC0 управляют сбросом таймера. Кроме того, таймер может быть сброшен командой T2RST (код 0Eh), выполняемой модулем HSO.

Бит T2OVF_INT регистра IOC1, бит T2OVF_MASK регистра INT_MASK1 и бит TIMER_MASK регистра INT_MASK управляют формированием запросов на прерывания по переполнению таймера. Бит T2CAP_MASK регистра INT_MASK1 управляет формированием запросов на прерывания по захвату значения таймера. Индикаторами запросов на прерывания являются биты T2OVF_PEND и T2CAP_PEND регистра INT_PEND1 и бит TIMER_PEND регистра INT_PEND. Флагом переполнения является бит T2_OVF регистра IOS1. Флагом сброса является бит T2RST_EVENT регистра IOS2.

Векторы прерываний:

- по переполнению – T2OVF (INT12) и TOVF (INT00);
- по захвату значения – T2CAP (INT11);
- по сбросу – SWTIMER (INT05).

Примечание – Если генерирование прерывания по переполнению таймера 2 разрешено, то вектор прерывания будет выбран в зависимости от состояния битов T2OVF_MASK и TIMER_MASK. Если оба бита установлены, то сначала будет обслужено прерывание с вектором T2OVF, а затем – с вектором TOVF, в соответствии с приоритетом. Таким образом, одно прерывание может быть дважды обслужено двумя разными подпрограммами, что может быть нежелательным. Это следует учитывать при программировании.

Состояния входов микроконтроллера T2CLK, T2UP_DN, T2CAP и T2RST фиксируются в моменты, когда внутренний сигнал синхронизации CLKOUT имеет низкий уровень. Для гарантированного и правильного детектирования состояние сигналов на входах должно оставаться стабильным, по крайней мере, в течение одного такта сигнала CLOCKOUT. Хотя захват состояний выводов происходит в каждом такте сигнала CLOCKOUT, опрос портовых защелок производится каждые 8 тактов, если таймер 2 работает в режиме нормального счета. Поэтому время реакции на изменение уровня того или иного сигнала на входе зависит от момента этого изменения в границах периода переключения счетчика таймера и может составлять от одного до восьми тактов сигнала CLOCKOUT. В режиме скоростного счета время реакции на изменения сигналов T2CLK, T2UP_DN и T2CAP составляет один такт сигнала CLOCKOUT.

В отличие от других входных сигналов, сигнал T2RST обрабатывается всегда одинаково – только каждый восьмой такт сигнала CLOCKOUT – независимо от режима работы таймера 2.

При одновременной установке сигналов T2CLK, T2CAP и T2RST аппаратная последовательность действий всегда следующая:

- захват значения таймера 2;
- сброс таймера 2 (обнуление счетчика таймера);
- переключение счетчика таймера.

При одновременной установке сигналов T2CLK и T2CAP (T2RST не установлен) сначала всегда производится захват значения таймера 2, а потом переключение счетчика таймера.

18.3 Модуль высокоскоростного ввода HSI

Основным назначением модуля является мониторинг состояния своих выводов HSI.0 – HSI.3 и фиксирование времени обнаружения до восьми запрограммированных событий с использованием таймера 1 (и только его) с возможностью последующей выдачи сохраненной информации. Функциональная схема модуля показана на рисунке 18.3.

Функции выводов HSI.0, HSI.1, HSI.2/HSO.4 и HSI.3/HSO.5 микроконтроллера, как входов модуля HSI, требуют включения посредством битов HSI0_ENA, HSI1_ENA, HSI2_ENA и HSI3_ENA регистра IOC0. Для програмирования выводов HSI.2/HSO.4 и HSI.3/HSO.5 дополнительно используется регистр IOC1.

Каждый вход оснащен защелкой, которая с каждым тактом синхросигнала CLOCKOUT фиксирует состояние соответствующего входа, детектором фронта входного сигнала и счетчиком восьми фронтов. Для гарантированного и правильного детектирования длительность стабильного состояния сигнала на входе должна быть не менее длительности одного такта сигнала CLOCKOUT. Таким образом реализованы четыре канала захвата модуля HSI. Для каждого канала независимо может быть задан один из четырех возможных режимов мониторинга состояния соответствующего вывода:

- детектирование каждого восьмого положительного фронта входного сигнала;
- детектирование каждого положительного фронта входного сигнала;
- детектирование каждого отрицательного фронта входного сигнала;
- детектирование каждого фронта входного сигнала.

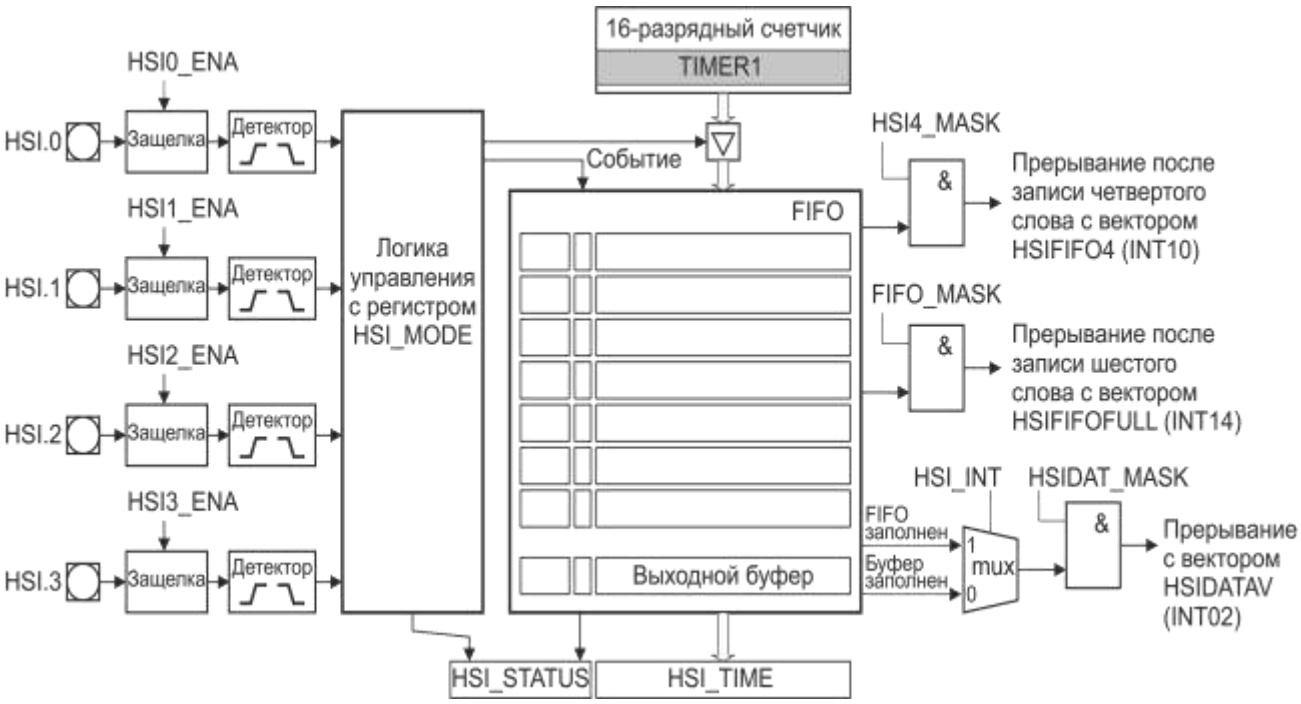


Рисунок 18.3 – Функциональная схема модуля HSI

Режимы задаются посредством регистра HSI_MODE. Как только на выводе обнаруживается запрограммированное событие, в накопительный блок FIFO загружается 16-разрядное значение таймера 1 (время события), бит-индикатор события и три служебных бита. Время загрузки в FIFO составляет один такт сигнала CLOCKOUT. Если два события обнаруживаются одновременно, они записываются с одинаковыми временами.

Если задан режим детектирования каждого восьмого положительного фронта входного сигнала, то активируется счетчик восьми фронтов. Через каждые восемь переключений счетчик подает сигнал на логику управления блока HSI. Особенность счетчика заключается в том, что при смене режима мониторинга счетчик не сбрасывается. Следует помнить, что при повторном включении режима отслеживания восьми фронтов счетчик может иметь ненулевое значение.

Накопительный блок FIFO состоит из запоминающего устройства для семи 20-разрядных расширенных слов и выходного 20-разрядного буфера. Таким образом, заполненный блок FIFO может хранить информацию о восьми событиях. Как только FIFO заполняется полностью, он перестает принимать информацию до тех пор, пока не появится свободное место. Информация о событии, обнаруженном первым, т.е. расширенное слово, записанное первым, находится в выходном буфере, остальные слова – в памяти FIFO в порядке, соответствующем порядку детектирования событий.

По мере заполнения выходного буфера и далее памяти блока FIFO модуль HSI может генерировать прерывания (в скобках указан соответствующий вектор прерывания):

- слово из памяти FIFO (в том числе и первое слово) загружено в выходной буфер или память FIFO заполнена (HSIDATAV); источник определяется битом HSI_INT;
- в память FIFO записано шестое слово, при этом состояние выходного буфера не учитывается (HSIFIFOFULL);
- в память FIFO записано четвертое слово, при этом состояние выходного буфера не учитывается (HSIFIFO4);

В таблице 18.1 представлены комбинации битов регистров IOC1, INT_MASK1 и INT_MASK для программирования прерываний модуля HSI.

В зависимости от того, какие биты управления и/или масок были установлены, при формировании запросов прерываний устанавливаются биты ожидания в регистрах INT_PEND1 и INT_PREND. Регистр IOS1 содержит два флага прерываний модуля HSI – флаг HSI_RDY выходного буфера и флаг FIFO_FULL заполнения памяти FIFO.

Таблица 18.1 – Связь источников прерываний и векторов прерываний

Запись в память FIFO	Запись в выходной буфер	Регистры и биты				Вектор прерывания
		IOC1	INT_MASK1		INT_MASK	
		HSI_INT	FIFO_ MASK	HSI4_ MASK	HSIDAT_ MASK	
–	Да	0	–	–	1	HSIDATAV (INT02)
Шестая	–	1	–	–	1	
Шестая	–	–	1	–	–	HSIFIFOFULL (INT14)
Четвертая	–	–	–	1	–	HSIFIFO4 (INT10)
Примечание – Символ «–» указывает на то, что состояние выходного буфера и состояние бита неважно.						

Чтение информации из блока FIFO происходит через выходной буфер. Начиная с первого записанного слова (которое уже находится в буфере), все остальные слова в том же порядке, в каком были записаны в блок FIFO, последовательно загружаются в буфер и затем становятся доступными для чтения.

Для чтения информации из блока FIFO последовательно выполняются два действия:

1 Чтение регистра HSI_STATUS.

В регистре находятся четыре пары битов – по одной для каждого канала. В паре один бит с приставкой «_EVENT» в названии указывает на номер канала, в котором было зафиксировано событие. Второй бит с приставкой «_STAT» в названии отражает текущее состояние сигнала на входе канала (именно текущее, а не то, что было на момент обнаружения события). Таким образом, считывая состояние регистра HSI_STATUS можно точно определить, к какому каналу относится слово, находящееся в выходном буфере – соответствующий бит будет установлен в то время, как остальные будут сброшены.

2 Чтение регистра HSI_TIME.

Регистр содержит значение, которое соответствует времени фиксирования события в блоке FIFO.

Сразу после прочтения регистра HSI_TIME очередное слово из памяти FIFO загружается в выходной регистр, и значения в регистрах HSI_STATUS и HSI_TIME обновляются. Поэтому важно соблюдать указанную последовательность чтения регистров.

После того, как хотя бы одно слово будет прочитано из выходного буфера, блок FIFO возобновит фиксацию времени событий до следующего своего заполнения.

18.4 Модуль высокоскоростного вывода HSO

Основным назначением модуля является сравнение запрограммированного значения/значений времени с текущим значением подключенного таймера и выполнение заданной команды (нескольких команд) при их совпадении. Выполняя заданные команды, модуль HSO может управлять своими выходами HSO.0 – HSO.5, генерировать программные прерывания, сбрасывать таймер 2 и запускать аналого-цифровое преобразование. Модуль HSO позволяет формировать независимые импульсные последовательности, в том числе и с ШИМ. Функциональная схема модуля показана на рисунке 18.4.

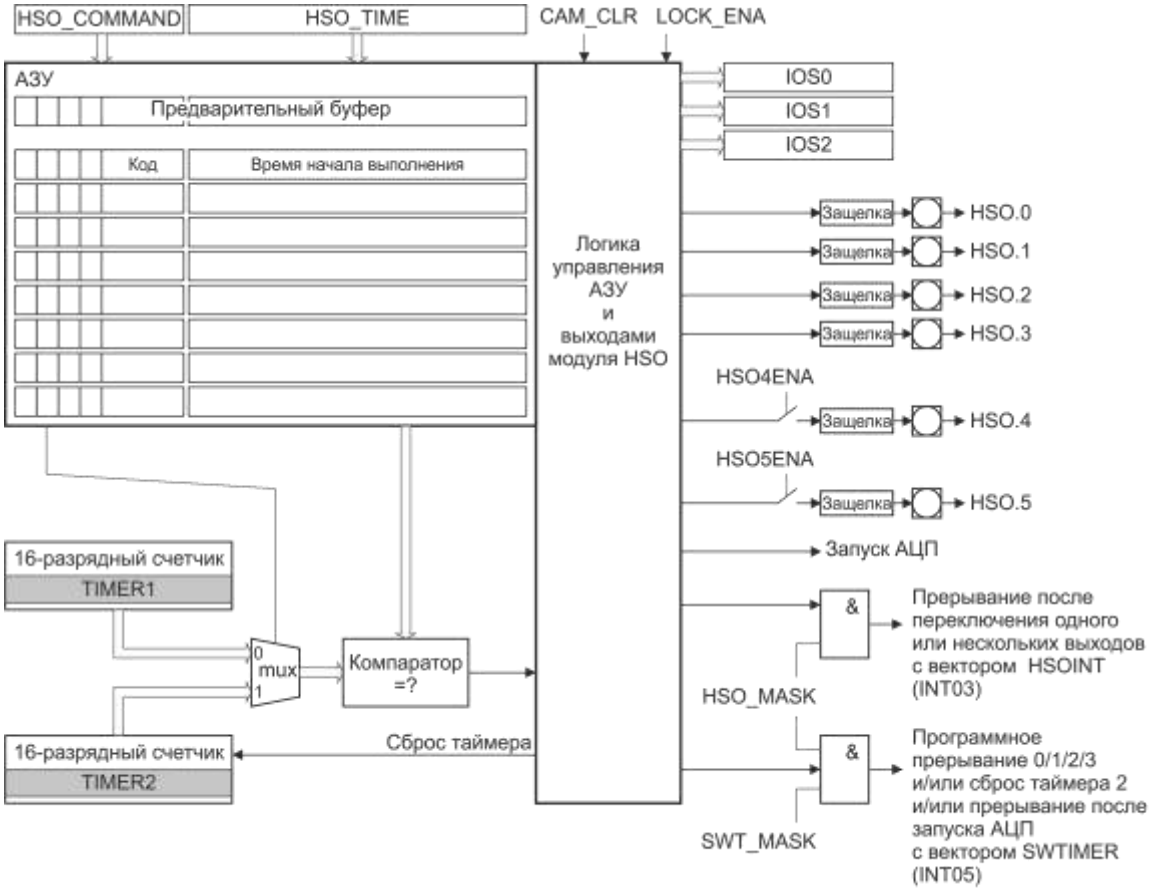


Рисунок 18.4 – Функциональная схема модуля HSO

Модуль HSO имеет четыре независимых вывода – HSO.0, HSO.1, HSO.2, HSO.3 – и два общих с модулем HSI вывода – HSI.2/HSO.4 и HSI.3/HSO.5, для програмирования которых используется регистр IOC1. Каждый из этих двух выводов может одновременно работать и как выход HSO, и как вход HSI.

Состояние выводов отражают флаги HSO0_STAT, HSO1_STAT, HSO2_STAT, HSO4_STAT и HSO5_STAT регистра IOS0.

Основным блоком модуля HSO является ассоциативное запоминающее устройство (АЗУ) для восьми 24-разрядных расширенных слов с 24-разрядным предварительным буфером хранения.

Каждое слово АЗУ включает в себя 16-разрядное значение времени для задания момента выполнения команды, четыре бита команды и четыре бита параметров.

Для записи каждого слова в АЗУ последовательно выполняются действия:

- проверка состояния флагов HR_STAT и HRCAM_STAT регистра IOS0;
- запись регистра HSO_COMMAND;
- запись регистра HSO_TIME.

1 Проверка состояния флагов HR_STAT и HRCAM_STAT регистра IOS0.

При программировании модуля HSO все слова сначала размещаются в АЗУ, минуя предварительный буфер. В АЗУ последовательно можно записать максимум восемь слов. После того, как АЗУ будет заполнено, установится флаг HRCAM_STAT. Девятое записанное слово автоматически будет размещено в предварительном буфере, после чего установится флаг HR_STAT. При дальнейшей работе каждый раз, когда будет освобождаться ячейка АЗУ, в нее будет переписываться слово из предварительного буфера. Цель проверки указанных флагов заключается в том, чтобы определить, свободен ли буфер или нет, поскольку неконтролируемая запись в буфер (при заполненном АЗУ) может привести к потере ранее записанной и подготовленной для загрузки в АЗУ информации.

2 Запись регистра HSO_COMMAND.

Если проверка флагов HR_STAT и HRCAM_STAT показала, что хотя бы одна ячейка АЗУ свободна и/или свободен предварительный буфер, то можно загрузить очередное слово. В первую очередь записываются параметры команды и ее код.

Бит CAM_LOCK позволяет зафиксировать команду в АЗУ, в результате чего команда будет выполняться столько раз, сколько подключенный таймер будет достигать значения времени выполнения. Незафиксированная команда будет выполнена всего один раз, после чего ячейка АЗУ, в которой она была размещена, полностью будет очищена и готова для загрузки нового слова. Для глобального разрешения фиксирования всех команд следует установить бит LOCK_ENA регистра IOC2.

В зависимости от состояния бита TIMER_SEL, может быть подключен как таймер 1, так и таймер 2.

В зависимости от выполняемой команды (команд) модуль HSO может оказывать влияние на свои выходы HSO.0 – HSO.5. Влияние заключается в том, что модуль может устанавливать на этих выходах как высокий, так и низкий уровень сигнала. Бит PIN_CMD задает уровень сигнала, который будет установлен на выходе/выходах в результате выполнения команды.

Установленный бит HSOINT_ENA разрешает прерывания, генерируемые ячейкой АЗУ. Прерывания маскируются битом HSO_MASK регистра INT_MASK. Индикатором сгенерированного запроса на прерывание является бит HSO_PEND регистра INT_PEND.

Оставшиеся четыре бита – поле CMD_TAG – задают код одной из 14 доступных команды.

Восемь команд с кодами от 0h до 7h и команда с кодом Ch управляют выходами HSO.0 – HSO.5. Результат выполнения любой из девяти команд – переключение соответствующего выхода (или выходов) в состояние, заданное битом PIN_CMD. Выполнение команды сопровождается генерированием прерывания с вектором HSOINT (INT03). Поскольку для всех девяти команд выделен только один вектор прерывания, то при обслуживании прерывания нужно считывать регистр IOS2, чтобы определить источник/источники прерывания. Комбинация состояний битов HSO0_EVENT, HSO1_EVENT, HSO2_EVENT, HSO3_EVENT, HSO4_EVENT и HSO5_EVENT будет указывать на источник/источники прерывания.

Четыре команды с кодами от 8h до Bh позволяют формировать программные прерывания. Результат выполнения любой из четырех команд – формирование запроса на прерывание с вектором SWTIMER (INT05). Поскольку для всех четырех команд выделен только один вектор прерывания, то при обслуживании прерывания нужно считывать регистр IOS1, чтобы определить источник/источники прерывания. Комбинация состояний битов SWTF0, SWTF1, SWTF2 и SWTF3 будет указывать на источник/источники прерывания.

Результат выполнения команды с кодом Eh – сброс таймера 2 и формирование запроса на прерывание с вектором SWTIMER (INT05). Для подтверждения источника прерывания следует проверять состояние флага T2RST_EVENT регистра IOS2.

Результат выполнения команды с кодом Fh – запуск аналого-цифрового преобразования и формирование запроса на прерывание с вектором SWTIMER (INT05). Для подтверждения источника прерывания следует проверять состояние флага AD_EVENT регистра IOS2.

3 Запись регистра HSO_TIME.

Запись в регистр завершает процесс заполнения одной ячейки (одного 24-разрядного слова) памяти АЗУ. Регистр HSO_TIME позволяет задать момент времени начала выполнения команды. Каждой команде в АЗУ соответствует определенное значение времени, при этом несколько команд могут иметь одинаковые значения времени.

Работа блока HSO начинается сразу после запуска таймера. Все значения времени, записанные в АЗУ, сравниваются с таймером одновременно в момент его переключения (для каждого слова может быть задан любой из двух таймеров). В этом процессе участвуют только те значения времени, которые расположены непосредственно в ячейках АЗУ. Информация, находящаяся в предварительном буфере хранения, не используется до тех пор, пока не будет переписана в освободившуюся ячейку АЗУ.

Если при очередном переключении таймера обнаруживается совпадение с запрограммированным значением времени, то одновременно со следующим переключением таймера выполнится соответствующая команда. Если совпадений несколько, то выполняются несколько команд.

Если результаты выполнения двух или нескольких команд, или одной команды, записанной несколько раз, являются противоположными, то выходы остаются в неизменном состоянии. Например, если в одной ячейке АЗУ записана команда HSO0, выполнение которой переключает выход HSO.0 в единицу, а в другой ячейке записана та же команда, но выполнение которой переключает выход HSO.0 в ноль, а значения времени совпадают, то состояние выхода HSO.0 не изменится. После этого, если команды не зафиксированы, они будут удалены из АЗУ.

Пример:

```
LDB     HSO_COMMAND, #00100000b
LDB     HSO_TIME, #2000h
LDB     HSO_COMMAND, #00000000b
LDB     HSO_TIME, #2000h
```

При программировании времени выполнения команды можно задавать значение времени непосредственно

```
LDB     HSO_COMMAND, #[параметры и команда]
LDB     HSO_TIME, #[время]
```

или как смещение относительно текущего значения таймера (в примере – таймера 1)

```
LDB     HSO_COMMAND, #[параметры и команда]
ADD     HSO_TIME, TIMER1, #[смещение]
```

Следует помнить, что загружаемое значение времени должно, как минимум на две единицы, превышать текущее значение счетчика таймера (т.е. «смещение» ≥ 02h), в

противном случае, их совпадение произойдет только через период таймера (65 535/32 767 переключений). Это ограничение справедливо как для таймера 1, так и для таймера 2.

Для выполнения одной команды из АЗУ требуется один такт синхросигнала таймера. Если несколько команд имеют одинаковое значение времени, то для выполнения этих команд требуется несколько тактов. Таким образом, если значение времени одно для всех команд в АЗУ, то нужно восемь тактов. По этой причине подключенный к модулю HSO таймер должен переключаться каждый восьмой такт синхросигнала. Это важно учитывать при работе с таймером 2, поскольку он имеет возможность функционировать в режиме ускоренного счета.

Допускается использование режима ускоренного счета таймера 2 в случае необходимости быстрого выполнения одной (!) команды, при условии отсутствия в АЗУ других команд с совпадающими значениями времени. В случае наличия нескольких команд с одинаковыми значениями времени будет выполнена только одна команда. Кроме того, если эта команда зафиксирована, то она и только она будет выполняться в каждом периоде таймера. Если команда не была зафиксирована, то после выполнения она будет удалена из АЗУ. Оставшиеся команды будут выполняться аналогично.

Направление счета таймера 2 может быть любым и может меняться в процессе работы, но желательно программировать таймер на счет только в одном направлении.

Если таймер 2 запрограммирован на сброс внешним сигналом (установлен бит T2RST_SRC регистра IOC0), то не следует задавать значение времени выполнения команды, равным «0000h». Поскольку внешний сброс таймера является асинхронным по отношению к работе модуля HSO, то переключение таймера после сброса может произойти до того, как компаратор успеет произвести сравнение и есть вероятность того, что состояние «0000h» таймера не будет обнаружено, а, следовательно, не будет выполнена запрограммированная команда.

19 Аналого-цифровой преобразователь и блок цифровых компараторов

Аналого-цифровой преобразователь, реализованный в контроллере, построен на основе сигма-дельта (Σ−Δ) преобразователя, реализующего алгоритм сверхвыборки, когда норма осуществления выборки во много раз превышает требуемую частоту (блок-схема АЦП представлена на рисунке 19.1). Такая реализация позволяет значительно уменьшить шумы и гарантирует стабильно высокие динамические параметры преобразователя. Схемы цифровых компараторов осуществляют контроль результатов преобразования и генерируют прерывания при их выходе за пределы заданных значений (рисунок 19.2).

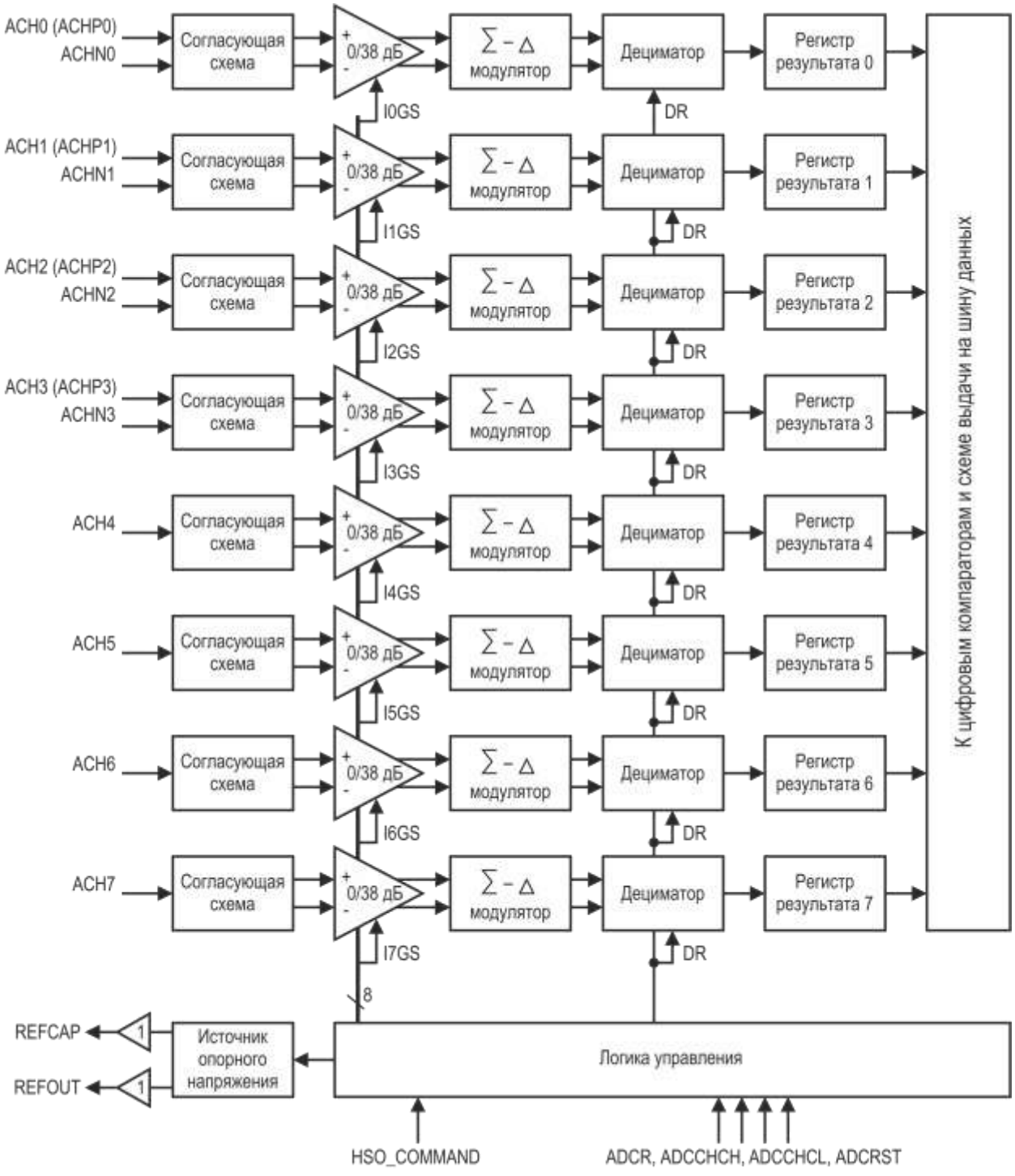


Рисунок 19.1 – Блок-схема аналого-цифрового преобразователя

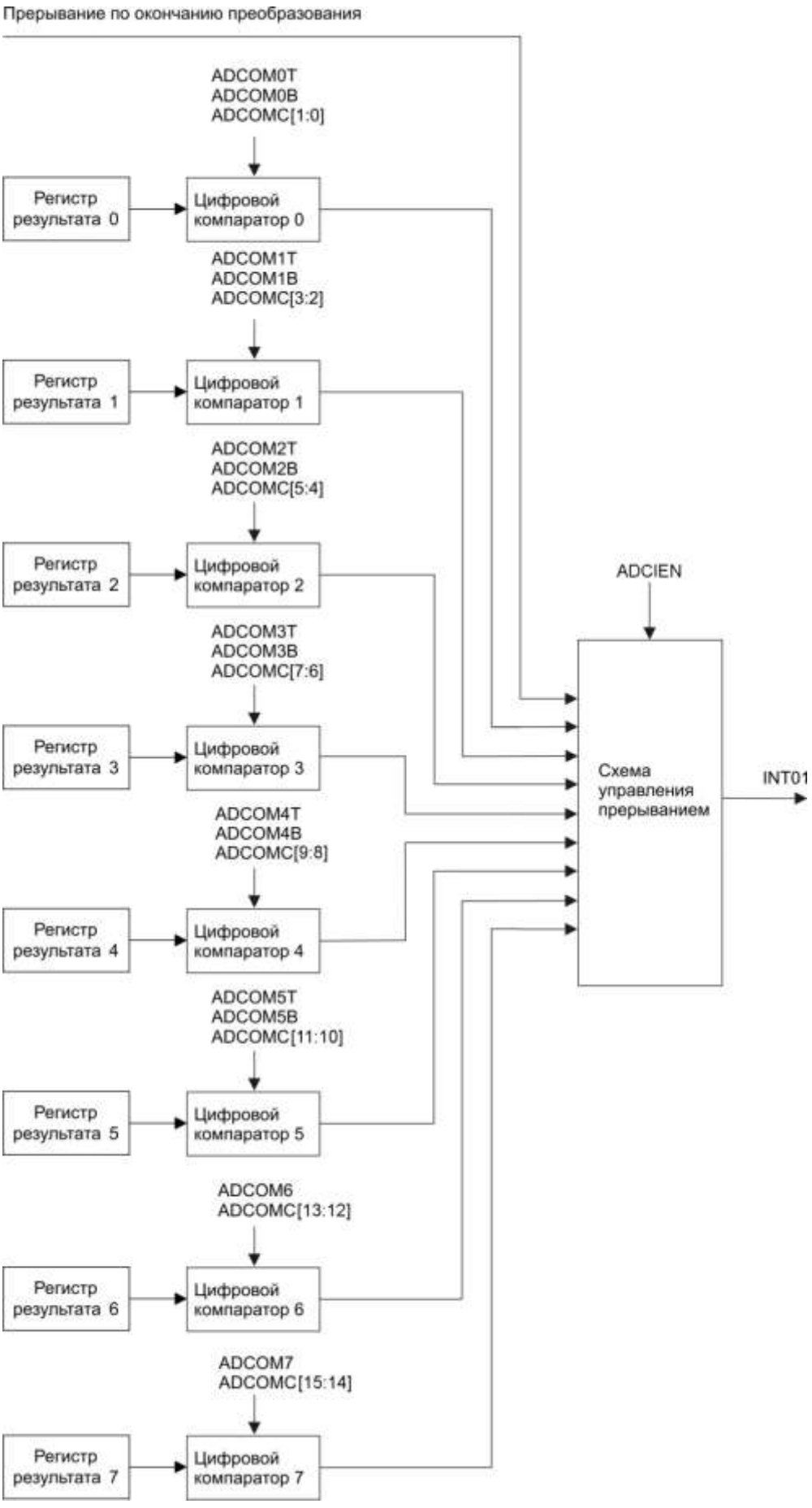


Рисунок 19.2 – Блок-схема цифровых компараторов

19.1 Обзор функций АЦП

Модуль АЦП (блок-схема на рисунке 19.1) преобразует входное напряжение сигнала на каждом из восьми каналов в 16-битное цифровое значение. Основными частями модуля АЦП являются:

- восемь идентичных схем преобразования, каждая из которых включает согласующую схему, усилитель входного сигнала, сигма-дельта модулятор, дециматор и 16-разрядный регистр результата преобразования;

- восемь аналоговых входов АЦП: четыре первых входа по умолчанию подключены к позитивным дифференциальным (дифф.) входам UINP каналов 0 – 3, четыре вывода ACHNx в режиме дифференциального входа через нормально замкнутые цепи коммутатора $К_1$ (коммутаторы $К_0$, $К_2$, $К_3$ не показаны) подключены к инверсным входам UINN каналов 0 – 3 АЦП (при «0» в разрядах 4 – 7 регистра ADCRST). При дифференциальном включении максимальное количество одновременно работающих каналов АЦП равно четырем (восемь выводов – по два входа на канал). Инверсия входов UINN в дифференциальных усилителях каналов АЦП установлена занесением единицы в разряды регистра ADCRST. Упрощенная блок-схема переключения входов в каналах АЦП в режим одиночного (однопроводного) входа приведена на рисунке 19.3.

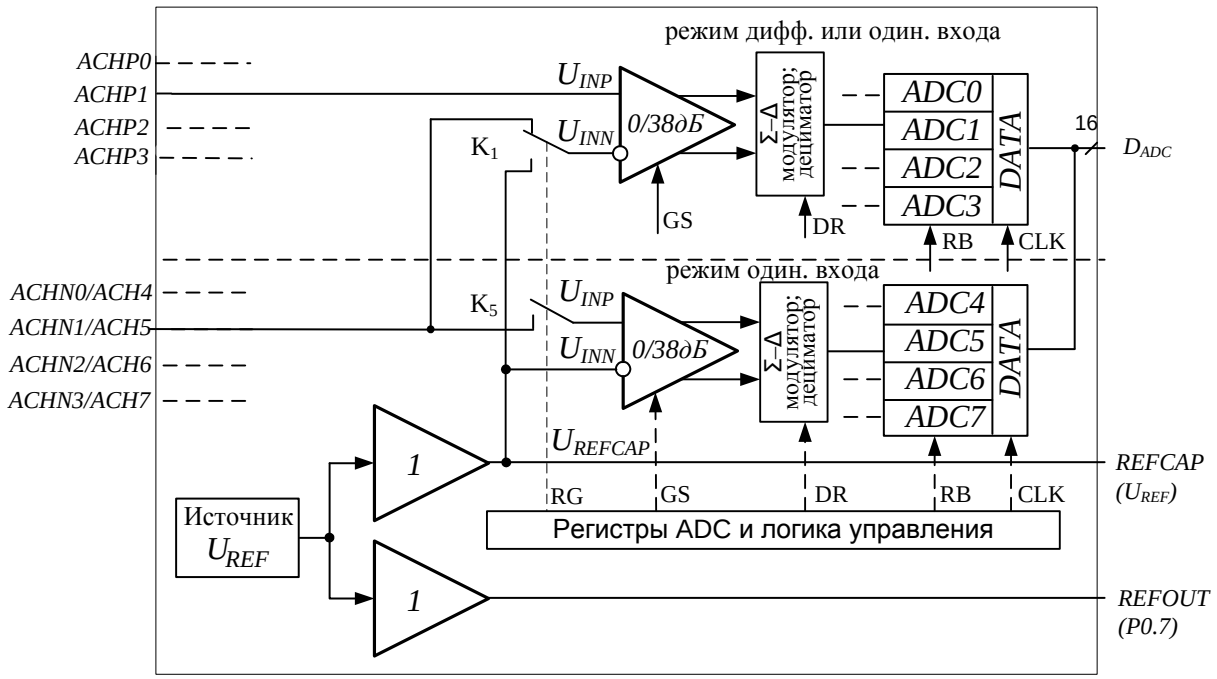


Рисунок 19.3 – Упрощенная блок-схема переключения режимов в каналах АЦП

Для переключения каналов АЦП в режим одиночного входа в соответствующие разряды регистра ADCRG заносится «1», при этом коммутаторы $К_1$…$К_5$ переключаются в нижнее положение (цепи коммутаторов шести каналов не показаны). При этом в каналах 0 – 3 АЦП входы UINN дифференциального усилителя переключаются на шину опорного напряжения U_{REF}, а на входы U_{INP} дифференциальных усилителей каналов 4 – 7 подключаются выводы ACH4 – ACH7. Каналы 4 – 7 АЦП используются только в режиме одиночного входа;

- регистры ADCR, ADCCHCH, ADCCHCL, ADCRST управляют временем преобразования, коэффициентом усиления входных сигналов, включением преобразований, запуском преобразования выбранных каналов и другими функциями АЦП;

- внутренний источник опорного напряжения АЦП микросхемы выполнен с использованием напряжения запрещенной зоны полупроводника. Опорное напряжение U_{REF} через буфер подается к каналам АЦП и на вывод REFCAP, к которому подключается конденсатор емкостью не менее 0,1 мкФ с минимальной длиной проводников;

- на вывод REFCAP для повышения стабильности опорного источника может подаваться внешний источник напряжения U_{REF} в диапазоне от 1,125 до 1,6 В с низким внутренним сопротивлением и выходным током не менее 2 мА. Точность поддержания входного опорного напряжения, заданного в выше указанном диапазоне, в процессе преобразования АЦП не менее 0,5 U_{LSB};

- при включении входов АЦП со смещением по переменному току по схемам на рисунках 19.6 и 19.8 предусмотрен буферизированный вывод опорного напряжения REFOUT установкой бита RU (CRC: 6) регистра управления. При внешнем U_{REF}, для схем согласования входных уровней АЦП, потребителям следует использовать этот же источник, так как выходной буфер REFOUT не имеет связи с выводом REFCAP. Блок-схема организации U_{REF} в каналах АЦП показана на рисунке 19.3;

- запуск преобразования АЦП осуществляется с помощью команды HSO при тактировании сигналом CLKC периферийных блоков микроконтроллера. Преобразуемый сигнал поступает на согласующую схему канала, обеспечивающую ограничение уровня, а также выбор полярности сигнала. С ограничителя сигнал поступает на усилитель с программируемым дискретным коэффициентом усиления в диапазоне от 0 до 38 дБ, далее – на сигма-дельта модулятор и дециматор. Усилитель с программируемым коэффициентом усиления каждого канала представляет собой схему на переключаемых конденсаторах и является частью сигма-дельта модулятора;

- цифровой фильтр выполняет две важные функции. Во-первых, это перемещение шума квантования за пределы требуемой полосы частот, который сформирован аналоговым модулятором, и, во-вторых, это прореживание потока битов высокой частоты до более низкой частоты 15-битных слов;

- сглаживающий прореживающий фильтр – это цифровой фильтр с характеристикой в форме sinc3, который понижает частоту дискретизации на значение, заданное в регистре управления. Делитель прореживания позволяет пользователю гибко подстроить значение выборки АЦП к требуемой программе контроллера;

- АЦП выполняет несколько функций, таких как: временная дискретизация, квантование по уровню, кодирование, аналого-цифровые преобразования. Передаточная характеристика преобразования напряжения ΔU_{IN} между входами канала АЦП в режиме дифференциального и одиночного входа приведена на рисунке 19.4 сплошной линией. В дифференциальном режиме передаточную характеристику АЦП также можно показать двумя прямыми относительно опорного напряжения U_{REF} по входам UINP" и UINN".

19.2 Соотношение режимных параметров в каналах АЦП

Термины, обозначения и соотношения режимных параметров АЦП приведены в соответствии с ГОСТ 29109–91.

В соответствии с рисунком 19.4 начальная и конечная точки преобразования входного напряжения АЦП U_{FS-} и U_{FS+} определяются экспериментально с суммарной погрешностью не более одного U_{LSB} (в идеальном случае) из соотношений (19.1) и (19.2) или могут быть рассчитаны из уравнения передаточной характеристики АЦП (19.7), при известном коэффициенте преобразования.

$$U_{FS-} = U_{FS-,1} - 0{,}5 \cdot U_{LSB}, \qquad (19.1)$$

где $U_{FS-,1}$ – напряжение первого межкодового перехода к коду 8001h;

U_{LSB} – напряжение единицы младшего разряда. В данном случае шаг квантования определяется выражением (19.4).

Входное напряжение в конечной точке АЦП при переходе к коду 7FFFh равно:

$$U_{FS+} = U_{FS(nom)+} - 0{,}5 \cdot U_{LSB}. \qquad (19.2)$$

Область значений входного напряжения АЦП, ограниченная точками (U_{FS-}–U_{FS+}), называется практическим диапазоном преобразования АЦП и обозначается U_{FSR}.

$$U_{FSR} = (U_{FS+} - U_{FS-}) - U_{LSB} = 2^N \cdot U_{LSB} - U_{LSB} = U_{LSB}(2^N - 1), \qquad (19.3)$$

где N – число разрядов.

Шаг квантования (единица младшего разряда) определяется отношением:

$$U_{LSB} = \frac{U_{FSR}}{2^N - 1} = \frac{U_{FSR(nom)}}{2^N}, \qquad (19.4)$$

где $U_{FSR(nom)}$ – номинальный диапазон преобразования АЦП, определяемый выражением:

$$U_{FSR(nom)} = U_{FSR} + U_{LSB} = 2^N \cdot U_{LSB}. \qquad (19.5)$$

Единица старшего знакового разряда определяется соотношением:

$$U_{MSB} = \frac{U_{FSR(nom)}}{2}. \qquad (19.6)$$

Передаточная характеристика преобразования напряжения ΔU_{IN} канала АЦП в режиме одиночного или дифференциального входа (показана сплошной линией $U_{FS-} - U_{FR+}$ на рисунке 19.4) описывается уравнением:

$$D_{ADC} = (U_{INP} - U_{INN}) \cdot K_G, \qquad (19.7)$$

где U_{INP} – входное напряжение на позитивном входе АЦП;

U_{INN} – входное напряжение на негативном входе АЦП;

D_{ADC} – выходной цифровой код АЦП;

K_G – коэффициент преобразования АЦП.

.

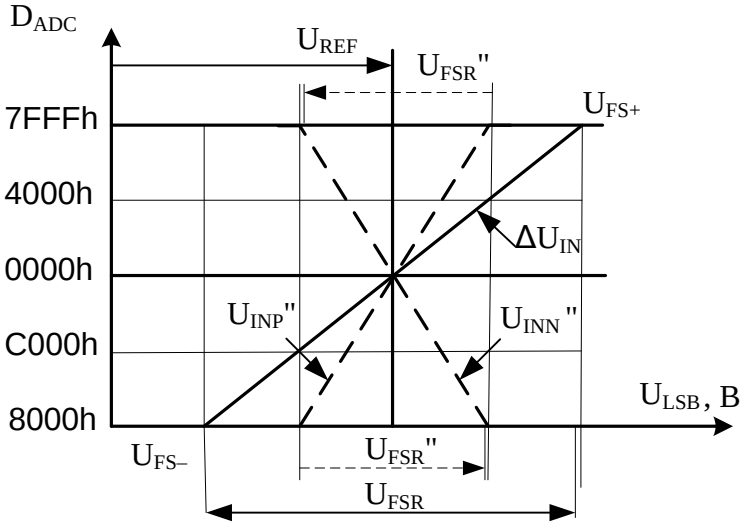


Рисунок 19.4 – Передаточная характеристика каналов АЦП

В соответствии с (19.7) и с учетом (19.5) коэффициент преобразования канала АЦП равен:

$$K_G = \frac{K_{PGA} \cdot D_{ADC}}{U_{FSR(nom)}} = \frac{K_{PGA}}{U_{LSB}}, \qquad (19.8)$$

где K_{PGA} – коэффициент усиления АЦП, при PGA = 0 дБ равен единице, при этом коэффициент преобразования АЦП равен:

$$K_G = \frac{1}{U_{LSB}}. \qquad (19.9)$$

Для передаточной характеристики по входам относительно U_{REF} (рисунок 19.4, пунктирные линии) диапазон преобразования и шаг квантования делятся пополам. При этом коэффициент преобразования по каждому из входов относительно U_{REF}, с учетом (19.4) и (19.9), равен:

$$K_G'' = \pm\frac{1}{0{,}5U_{LSB}}. \qquad (19.10)$$

Выше приведенные соотношения режимных параметров позволяют определить и рассчитать шаг квантования, коэффициент преобразования и допустимые входные напряжения АЦП. Ниже приведены примеры 1 – 5 расчета диапазона входных напряжений и выходного кода АЦП с комментарием по применению.

Пример 1 – Из уравнения (19.7) рассчитаем допустимый диапазон напряжений преобразования между входами дифференциального усилителя 16-разрядного АЦП в режиме дифференциального или одиночного входа ($U_{INN} = U_{REF}$) при PGA = 0 дБ. Используя типовую зависимость, рисунок 19.4а, при номинальном U_{REF} = 1,25 В и температуре 25 °С, определяем U_{FSR} = 1,5 В. U_{FSR} также можно определить, используя выше приведенные соотношения (19.1) – (19.5). При U_{FSR} = 1,5 В (из (19.9) числовой коэффициент преобразования K_G равен 43 690,8. По определению, цифровой код АЦП 16-разрядного АЦП равен $D_{ADC} = 2^{16} = 65\,536$. Относительно опорного напряжения код АЦП одной полярности составит $D_{ADC} = \pm 2^{15} = \pm 32\,768$. Максимальная разница входного напряжения $\Delta U_{IN} = (U_{FS+} - U_{FS-}) = D_{ADC} / K_G = \pm 32\,768/43\,690{,}8 = \pm 0{,}75$ В (напряжение на входах относительно U_{REF}). Напряжение между входами АЦП ΔU_{IN} должно быть не более диапазона преобразования U_{FSR} = 1,5 В (см. рисунки 19.4, 19.4б).

В то же время в дифференциальном режиме входные напряжения ($U_{INP}'' - U_{INN}''$) относительно опорного напряжения при K_G'' = ±87 381 (19.10) из уравнения (19.7), при коде $D_{ADC} = 2^{15} = 32\,768$ не более: $\Delta U_{IN}'' = (U_{INP}'' - U_{INN}'') = D_{ADC}/K_G'' = 32\,768$ (±87 381) = = ±0,375 В (**U_{FSR}**" не более 0,75 В).

Пример 2 – Из уравнения (19.7) при известном коэффициенте преобразования K_G (см. (19.9)) можно определить выходной код АЦП. Для входного напряжения ΔU_{IN} = 0,375 В = = ¼U_{FSR}: выходной код $D_{ADC} = \Delta U_{IN} \cdot K_G = 0{,}375 \cdot 43\,690{,}8 = 16\,384$= 4000h (см. рисунок 19.4).

Пример 3 – Расчет входных напряжений преобразования при PGA = 38 дБ (дополнение к примеру 1). При PGA = 38 дБ коэффициент усиления АЦП равен $K_{PGA/38дБ}$ = = 80. Коэффициет преобразования АЦП в соответствии с (19.8) также увеличится в 80 раз и составит: $K_{G/38дБ} = 43\,690{,}8 \cdot 80 = 3\,495\,264$. Диапазон преобразования на входах АЦП соответственно уменьшится в 80 раз и составит 18,75 мВ при U_{LSB} = 0,5722 мкВ.

При расчетах также возможно использование уравнения АЦП (19.7) с полученным выше числовым коэффициентом преобразования $K_{G/38дБ}$. При выходном цифровой коде $D_{ADC} = \pm 2^{15} = \pm 32\,768$. Границы диапазона преобразуемых напряжений на входах АЦП в данном примере равны: $\Delta U_{IN} = (U_{FS+} - U_{FS-}) = D_{ADC}/K_{G/38дБ} = \pm 32\,768/3\,495\,264 = \pm 9{,}375$ мВ.

Из примера следует, что напряжение на входах АЦП, при PGA = 38 дБ, не более ±9,375 мВ относительно U_{REF} и между входами АЦП не более $U_{FSR/38дБ}$ = 18,75 мВ.

В то же время в дифференциальном режиме входные напряжения ($U_{INP}'' - U_{INN}''$) относительно опорного напряжения при $K_G''_{/38дБ} = \pm 3\,495\,264 \cdot 2 = \pm 6\,990\,528$ из уравнения

(19.7) не более: $\Delta U_{IN}'' = (U_{INP}'' - U_{INN}'') = D_{ADC}/K_G''_{/38дБ} = 32\,768 / (\pm 6\,990\,528) = \pm 4{,}6875$ мВ и не более $U_{FSR}''_{/38дБ} = 9{,}375$ мВ. Диапазон преобразования сигма-дельта модулятора при этом равен 1,5 В.

При работе с PGA = 38 дБ в 80 раз повышаются требования к стабильности опорного напряжения АЦП и уровню электрических помех на входах АЦП.

Пример 4 − Расчет входных напряжений и кодов преобразования АЦП в диапазоне рабочих температур ведется аналогично выше изложенному с учетом типовой зависимости, приведенной на рисунке 19.4а. Из рисунка 19.4а видно, что при заданном U_{REF} диапазон преобразования АЦП ограничивается пониженной температурой среды, и именно эти значения U_{FSR} следует использовать для расчета шага квантования и коэффициента преобразования при выбранном U_{REF}. При номинальном опорном напряжении U_{REF} = 1,25 В, PGA = 0 дБ и U_{CC2} = 3,3 В диапазон преобразования U_{FSR} при пониженной температуре среды минус 60 °С − не менее 1,44 В (см. таблицу 2.3).

При этом верхнее практическое значение диапазона преобразования U_{FSR} определяется потребителем, исходя из выбранного U_{REF} и диапазона рабочих температур, с учетом типовой зависимости на рисунке 19.4а.

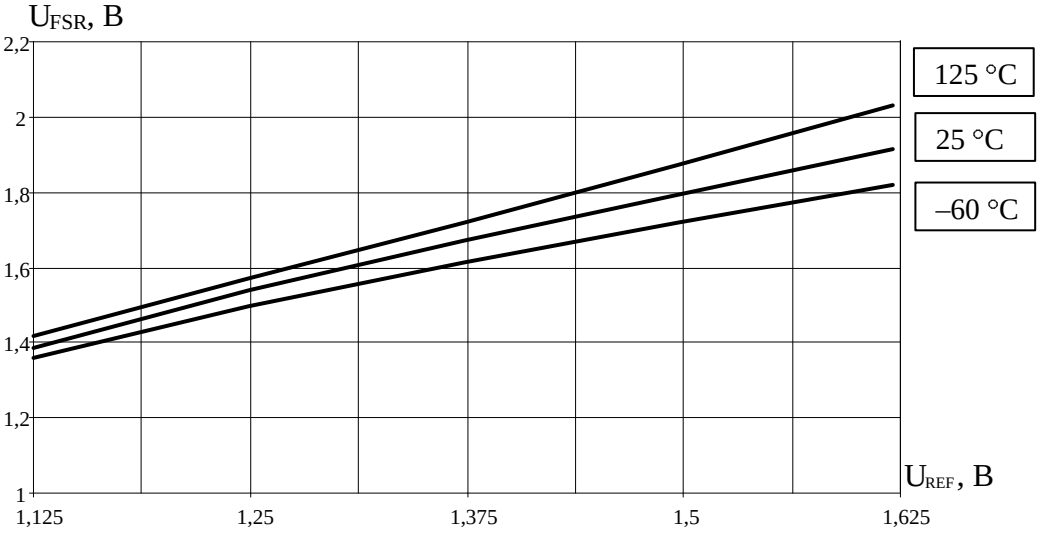


Рисунок 19.4а − Типовая зависимость диапазона преобразования каналов АЦП от опорного напряжения в диапазоне рабочих температур при PGA = 0 дБ, U_{CC2} = 3,3 В

Пример 5 − Пример временных диаграммм при оцифровке входного синусоидального напряжения ΔU_{IN} с частотой f_{IN} = 1кГц между входами АЦП и входных напряжений U_{INP}'' и U_{INN}'' относительно опорного напряжения при PGA= 0 дБ приведен на рисунке 19.4б.

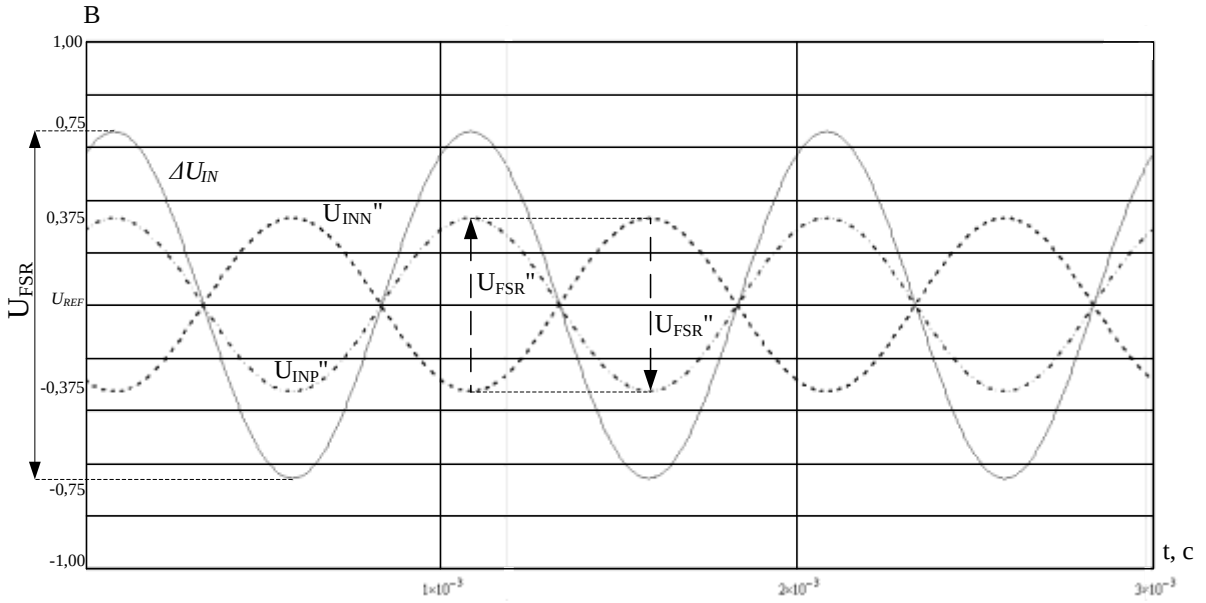


Рисунок 19.4б − Временные диаграммы входных напряжений АЦП при f_{IN} = 1 кГц

Комментарий к примерам

При PGA = 0 дБ диапазон входных напряжений преобразования между входами дифференциального усилителя АЦП в режиме дифференциального и одиночного входа равен диапазону преобразования U_{FSR} сигма-дельта модулятора АЦП (см. рисунок 19.4а). Входные напряжения при непрерывном преобразовании АЦП должны иметь 10 % запас от границ номинального диапазона преобразования. Дифференциальное включение входов АЦП позволяет избавиться от синфазной помехи.

В дифференциальном режиме возможна асимметричная подача входных напряжений относительно опорного напряжения, при этом результат преобразования (выходной код) определяется разницей входных напряжений в пределах диапазона преобразования АЦП.

Входы каналов АЦП допускают подачу входных напряжений от 0 В до U_{CC2} без гарантий результатов АЦП вне диапазона преобразования, что может использоваться при поиске границ диапазона преобразования U_{FS-}, U_{FS+} или при других оригинальных методах частного применения каналов АЦП с учетом передаточных характеристик, приведенных на рисунке 19.4.

19.3 Регистры управления и программирование АЦП

Существуют следующие способы управления АЦП:

- расширенное управление с помощью регистров управления;
- запуск преобразования с помощью команды модуля HSO.

С помощью расширенного управления можно осуществить одновременный запуск любого количества выбранных каналов с требуемыми параметрами коэффициента усиления по входу для каждого из каналов. В АЦП реализована возможность запуска преобразования выбранного канала с помощью команды модуля HSO.

Время запуска преобразования

Существует несколько способов запуска преобразования. Но, независимо от способа запуска для включения каналов АЦП в работу, требуется определенное время. Во время первого запуска преобразования, после включения контроллера, требуется время, равное 100 мкс, необходимое для включения источника опорного напряжения. Если запущен старт преобразования какого-либо канала, то преобразование начнется автоматически, сразу после запуска источника опорного напряжения. Все остальные запуски не требуют дополнительного времени ожидания и начинаются непосредственно после установки

соответствующих битов. Источник опорного напряжения остается включенным во время работы контроллера даже в том случае, когда никакое преобразование не выполняется. Исключение составляют режим сохранения мощности (POWERDOWN) и отключение АЦП через регистр CLKC. Переход контроллера в эти режимы вызывает выключение источника опорного напряжения. После выхода контроллера из режима сохранения мощности для запуска источника опорного напряжения также требуется 100 мкс.

Запуск преобразования командой HSO

В АЦП предусмотрена возможность запуска преобразования всех каналов с помощью команды блока высокоскоростного вывода сигналов HSO. Запуск преобразования возможен лишь в том случае, если бит GPU регистра ADCR установлен.

Модуль HSO включает преобразование при выполнении команды CMD_TAG = 0Fh. Процесс преобразования запускается, когда таймер 1 или таймер 2 (в зависимости от того, какой из них выбран в качестве базового) достигнет запрограммированного значения. После этого бит GPU в регистре ADCR будет установлен, что вызовет немедленное начало преобразования всех каналов.

Механизм расширенного управления АЦП

Механизм расширенного управления АЦП обеспечивает гибкое управление каждым каналом, скоростью выборки, запуск преобразования любых выбранных каналов, а также перевод любого канала в режим непрерывного преобразования.

Индивидуальное управление каналами

Регистры ADCCHCH и ADCCHCL предназначены для управления коэффициентом усиления и запуском отдельных каналов модуля АЦП.

После запуска преобразования бит старта преобразования этого канала остается установленным в течение всего времени и соответствует непрерывному режиму преобразования. Если необходимо остановить преобразование, то необходимо сбросить биты PUIx (x – номер канала от 0 до 7).

Включение режимов дифференциальных входов осуществляется битами регистра ADCR.

Одновременный запуск преобразования всех каналов

Функциональные возможности АЦП позволяют осуществить одновременный запуск преобразования всех каналов установкой бита GPU регистра ADCR.

Прерывание по завершению преобразования

Запуск и окончание преобразования выбранных каналов, независимо от их количества, осуществляется одновременно. Сразу после завершения преобразования выставляется запрос прерывания по окончанию преобразования, и на следующий такт ЦПУ в регистры результата заносятся результаты преобразования активных каналов. В случае работы блока АЦП в режиме цифровых компараторов возможно отключение прерывания по окончанию преобразования сбросом бита INTEN регистра ADCR.

19.4 Регистры управления основными и дополнительными функциями АЦП

Кроме регистров управления преобразованием в АЦП, предусмотрены дополнительные регистры управления основными и дополнительными функциям.

Регистр ADCR осуществляет глобальное управление сбросом АЦП, одновременный запуск преобразования всех каналов, включение буферизированного выхода REFOUT, выключение АЦП и включение дифференциального режима работы каналов (только для каналов 0 – 3).

Бит INTEN регистра ADCR управляет генерацией прерывания по окончанию преобразований. Бит «0» (третий бит) регистра ADCR предназначен для тестирования. Необходимо следить, чтобы этот бит был сброшен, иначе АЦП может неправильно функционировать. Бит DIV2 предназначен для дополнительного деления частоты на входе АЦП на два (всего АЦП, а не только аналоговой части). Рекомендуется выбирать скорость

работы АЦП битами DR и MCD, а бит DIV2 использовать в случае, если требуемую частоту не удается получить с помощью других способов.

В схеме присутствует программируемый делитель тактовой частоты, который позволяет пользователю уменьшить частоту тактового сигнала АЦП в один, два, три, четыре или пять раз для формирования внутреннего опорного тактового сигнала (DMCLK), используемого для вычисления значений выборки. Коэффициент деления задается битовым полем MCD регистра ADCR.

Битовое поле DR регистра ADCR определяет скорость выборки. Делитель прореживания позволяет пользователю гибко подстроить значение выборки АЦП к требуемой программе. Максимальное значение скорости выборки соответствует DMCLK/256. По умолчанию значение выборки соответствует скорости DMCLK/512.

Входное согласующее устройство на входе каждого канала позволяет независимо инвертировать входной сигнал. Общее разрешение инверсии входного сигнала осуществляется установкой бита INV регистра управления ADCR. Биты регистра ADCRST определяют каналы, на входах которых требуется произвести инверсию.

Бит PDOFF вместе с битом PUREF управляют включением/выключением опорного напряжения АЦП. Если бит PDOFF установлен, то опорное напряжение будет выключаться при переводе микроконтроллера в режим POWERDOWN или запрещению подачи на модуль АЦП тактового сигнала (регистр CLKC). Если бит PDOFF сброшен, то управление ведется с помощью бита PUREF (1 – опорное напряжение включено, 0 – выключено). Необходимо помнить, что включение опорного напряжения происходит автоматически при запуске преобразования любого из каналов. Для выключения опорного напряжения необходимо завершить все преобразования.

С помощью установки бита RMOD можно сбросить аналоговые модуляторы, чьи биты выбора каналов CHREx регистра ADCRST установлены в единицы. Для корректной работы бит RMOD должен быть сброшен.

После осуществления сброса аналоговых модуляторов потребуется выполнение не менее трех циклов преобразования для каждого из каналов для получения правильного результата.

Бит SEEN предназначен для включения режимов прямых (однополярных) и дифференциальных каналов. Установка бита включает режим одиночного входа тех каналов, чьи биты выбора каналов CHREx установлены в единицу. Сброс битов выбора каналов в нули задает режим дифференциального включения входов (только для каналов 0 – 3). Не рекомендуется включать режим дифференциального включения входов для каналов 4 – 7, так как это приведет к неправильному функционированию АЦП.

Бит RU предназначен для включения буферизированного выхода опорного напряжения АЦП. Опорное напряжение подается через ключ на вывод 7 порта 0. Данное напряжение может применяться как для внешних схем, так и для калибровки канала АЦП в микроконтроллере.

19.5 Регистры результата преобразования

В состав АЦП входят восемь 16-разрядных регистров результата преобразования ADCRES0 – ADCRES7, по одному для каждого канала. После окончания преобразования выбранных каналов, независимо от способа запуска преобразования, результат преобразования для каждого задействованного канала одновременно считывается с внутренней шины АЦП и помещается в соответствующий регистр результата, стирая старое значение. Содержимое регистров результата незадействованных каналов не изменяется.

Важно помнить, что после первого запуска преобразования канала, произведенного после включения питания, сброса аналоговой части АЦП, вывода контроллера или АЦП из режима сохранения мощности, требуется проведение трех циклов преобразования для

получения корректного результата преобразования. После завершения каждого из этих циклов результат преобразования будет передаваться в регистр результата канала, но лишь после третьего цикла преобразования результат преобразования будет правильным. Все последующие преобразования канала, запущенные после этих трех циклов, выполняются за один цикл с выдачей корректного результата. Это условие необходимо выполнять для каждого канала АЦП.

19.6 Интерфейс с АЦ преобразователем

АЦП имеет независимые выводы для питания аналоговой и цифровой частей. Для питания аналоговой части схемы используются выводы ∩VCC3 и ∩0V3. Цифровая часть модуля подключена к шинам питания микроконтроллера. Информацию о подключении выводов питания аналоговой части АЦП и требования к источнику питания можно найти в соответствующем разделе руководства.

Выходное напряжение внутреннего источника выдается на вывод REFOUT. Вывод REFCAP используется для подключения внешнего буферного конденсатора. Температурная нестабильность напряжения REFCAP – 50 ppm/ºC. Минимальное сопротивление нагрузки, подключенной к REFOUT, составляет 1 кОм, максимальная емкость – 100 пФ.

Входное сопротивление аналого-цифровых преобразователей по входам ACHx, ACHPx, ACHNx составляет 25 кОм при частоте DMCLK = 16 МГц. Входное сопротивление обратно пропорционально частоте DMCLK ($4 \cdot 10^{11}$/DMCLK).

Постоянное смещение сигнала аналогового входа при включении в режиме одиночного входа выполняется с использованием внутреннего источника опорного напряжения. Пример такого подключения представлен на рисунке 19.5.

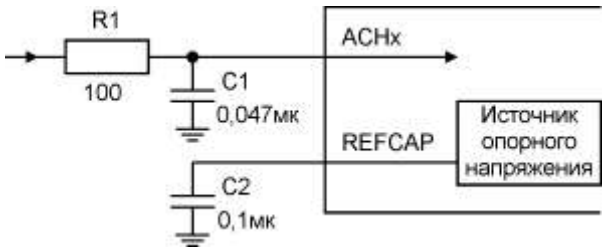


Рисунок 19.5 – Пример схемы подключения канала со смещением по постоянному току (режим одиночного входа)

В том случае, если смещение входного сигнала не равно уровню внутреннего источника опорного напряжения, должно использоваться подключение по переменному току. Смещение входа в этом случае может быть реализовано согласно схеме, приведенной на рисунке 19.6.

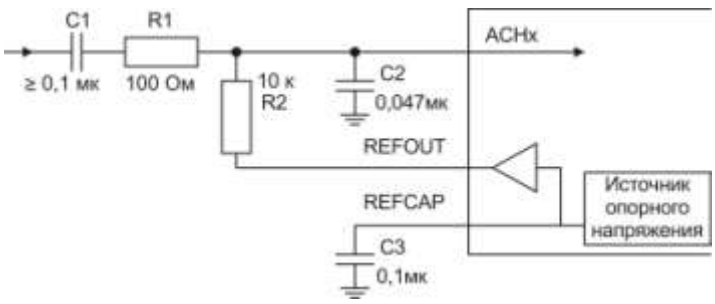


Рисунок 19.6 – Пример схемы подключения канала со смещением по переменному току (режим одиночного входа)

Пример включения АЦП в режиме дифференциальных входов представлен на рисунках 19.7 и 19.8.

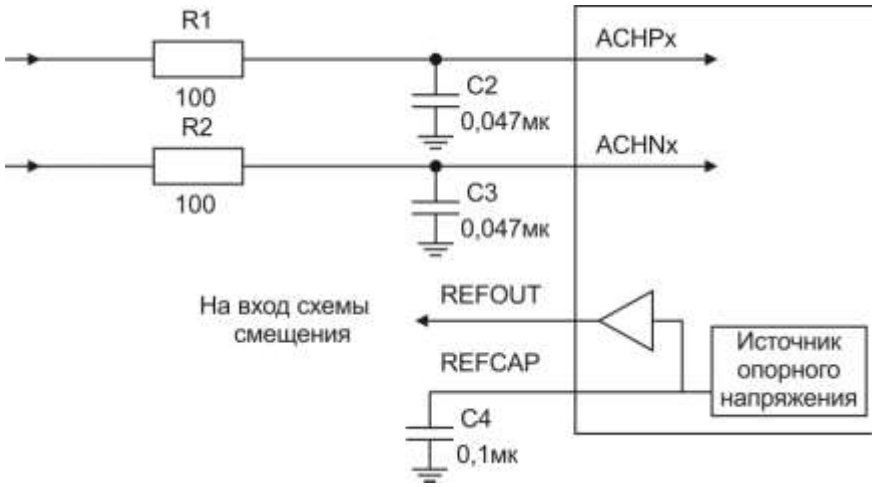


Рисунок 19.7 – Пример схемы подключения канала со смещением по постоянному току (режим дифференциальных входов)

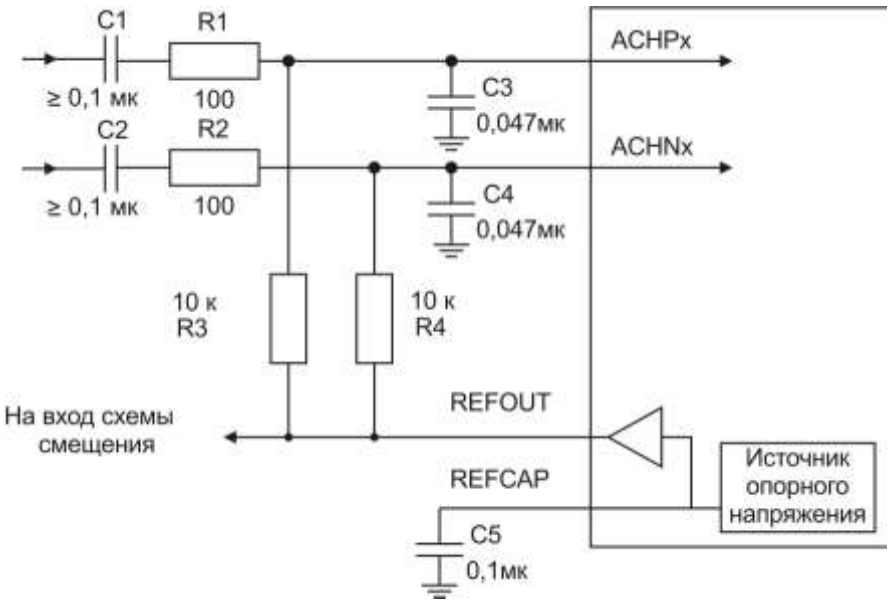


Рисунок 19.8– Пример схемы подключения канала со смещением по переменному току (режим дифференциальных входов)

Порт 0 может быть использован для операций с аналоговыми и цифровыми сигналами в одно время. Однако, при чтении порта на аналоговые цепи может быть наведен небольшой шум, который способен повлиять на точность результата преобразования, протекающего в текущее время. По этой причине не рекомендуется использовать АЦП во время чтения порта 0.

19.7 Блок цифровых компараторов

Блок цифровых компараторов (рисунок 19.2) позволяет управлять генерированием прерываний АЦП. Так как основной режим работы АЦП – режим непрерывного преобразования каналов, то имеет смысл отключение прерывания по окончанию

преобразования и включение прерываний по выходу контролируемого параметра за заданный диапазон. Основные функциональные возможности блока компараторов:

- генерация прерывания в случае, если результат преобразования меньше, либо равен заданному значению (рисунок 19.9);
- генерация прерывания в случае, если результат преобразования больше, либо равен заданному значению (рисунок 19.10);
- генерация прерывания в случае, если результат преобразования попадает в диапазон, заданный двумя регистрами границ – верхней и нижней (режим доступен только для каналов 0 – 5) (рисунок 19.11);
- генерация прерывания в случае, если результат преобразования выходит из диапазона, заданного двумя регистрами границ – верхней и нижней (режим доступен только для каналов 0 – 5) (рисунок 19.12).

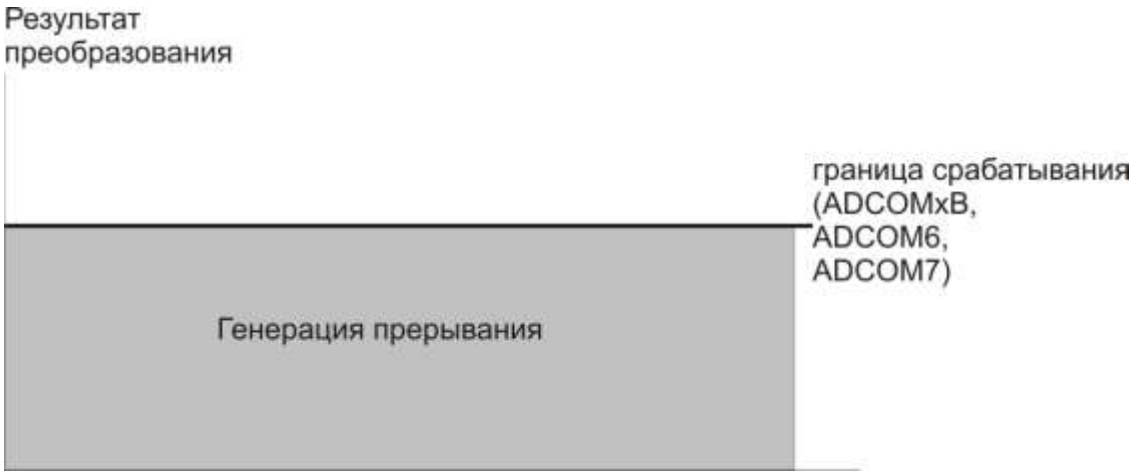


Рисунок 19.9 – Срабатывание компараторов по нижней границе

Рисунок 19.10 – Срабатывание компараторов по верхней границе

Рисунок 19.11 – Срабатывание компараторов по попаданию в диапазон

Рисунок 19.12 – Срабатывание компараторов по выходу из диапазона

В случае, если АЦП не используется, компараторы могут контролировать, например, результаты арифметических расчетов. Так как регистры результатов аналого-цифровых преобразований доступны для записи, то они могут фигурировать в командах умножения, сложения, пересылки и т. д. То есть, если организовать программу так, чтобы результаты вычислений хранились в регистрах результата АЦП, то возможен их аппаратный контроль. Так как каждый цифровой компаратор имеет свой приоритет, то возможна организация сложных условий сравнения (например, прерывание срабатывает, когда условие цифрового компаратора 0 не выполнилось, а компаратора 1 – выполнилось).

Блок состоит из восьми компараторов, которые каждый машинный цикл сравнивают значение, записанное в регистре результата преобразования соответствующего канала АЦП, со значением, указанным в задающих регистрах. Компараторы каналов 0 – 5 имеют по два 16-разрядных регистра, задающих как верхнее (ADCOMxT), так и нижнее (ADCOMxB) граничные значения. Компараторы каналов 6 – 7 имеют по одному 16-разрядному регистру (ADCOM6, ADCOM7), поэтому не подходят для контроля диапазона. Управляющий регистр ADCOMC определяет, какие типы контроля использовать для каждого цифрового компаратора. Регистр ADCIEN содержит флаги срабатывания компараторов и биты разрешения генерации прерывания INT01 каждым из компараторов. Флаги срабатывания компараторов устанавливаются вне зависимости, разрешена ли генерация прерывания INT0 этим каналом или нет.

Каждый цифровой компаратор имеет свой приоритет. Самый большой приоритет имеют компараторы 0 и 4. Распределение приоритетов компараторов представлено в таблице 19.1.

Таблица 19.1 – Распределение приоритетов цифровых компараторов

Номер компаратора	Приоритет	Группа приоритетов
0	4	1
1	3	1
2	2	1
3	1	1
4	4	2
5	3	2
6	2	2
7	1	2

Для срабатывания прерывания цифрового компаратора необходимо, чтобы оно было разрешено в регистре ADCIEN, выполнилось условие выбранного компаратора и не выполнились условия компараторов с более бысоким приоритетом из группы.

В случае, если произошло срабатывание цифрового компаратора, выставляется соответствующий флаг в регистре ADCIEN. В случае, если разрешено прерывание при срабатывании какого-либо компаратора, блок генерирует прерывание INT01.

Так как по умолчанию после сброса все компараторы настроены на фиксацию момента выхода за пределы или равенства результата преобразования нижней границы, а значение результатов всех преобразований и значения нижних границ в начальный момент равно 0000h, то биты AD0FLAG и AD4FLAG после сброса, по умолчанию, установлены.

20 Цифро-аналоговый преобразователь

Микроконтроллер 1874ВЕ8Т имеет встроенный 14-разрядный цифро-аналоговый преобразователь, позволяющий получать на выходе ИС ток, пропорциональный записанному в управляющий регистр значению.

Схемы применения цифро-аналоговых преобразователей относятся не только к области преобразования код – аналог. Пользуясь их свойствами, можно определять произведения двух или более сигналов, строить делители функций, аналоговые звенья, управляемые от микроконтроллеров, такие как аттенюаторы, интеграторы. Важной областью применения ЦАП являются также генераторы сигналов, в том числе сигналов произвольной формы.

ЦАП могут использоваться в следующих областях:

- обработка чисел, имеющих знак;
- перемножители и делители функций;
- аттенюаторы и интеграторы на ЦАП;
- системы прямого цифрового синтеза сигналов:
    - бытовая техника;
    - обработка звука и видео;
    - системы цифровой обработки сигналов;
    - цифровое телевидение;
    - радиолокация;
    - управление процессами в промышленности;
    - медицинское оборудование;
    - телеметрия;
    - измерительные приборы и диагностическое оборудование;
- цифровая обработка сигналов.

Характеристики блока ЦАП:

- 14-бит разрешение (16 384 значений выходного тока);
- дифференциальные токовые выходы – максимум тока от 2 до 20 мА;
- потребляемая мощность блока 135 мВт;
- режим пониженного потребления 15 мВт;
- внутренний источник опорного напряжения 1,2 В.

20.1 Функциональный обзор ЦАП

Структурная схема блока представлена на рисунке 20.1.

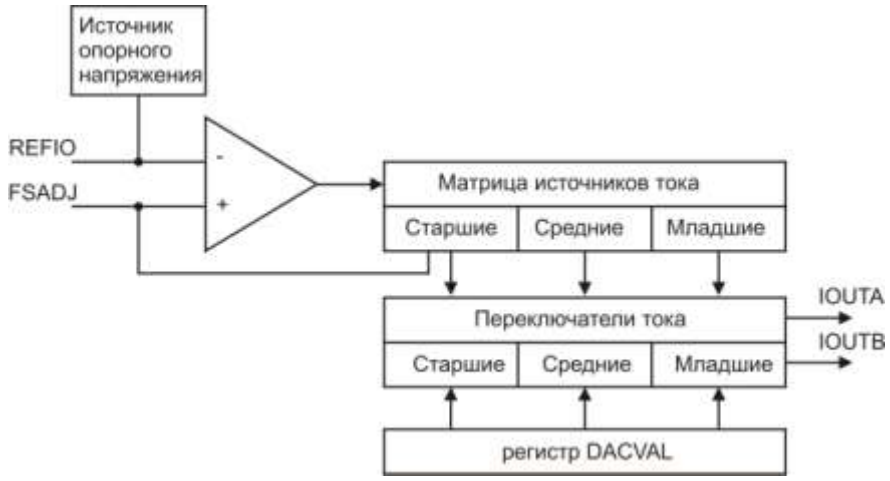


Рисунок 20.1 – Структурная схема блока ЦАП

ЦАП содержит следующие функциональные блоки:

- источник опорного напряжения 1,2 В (ИОН);
- управляющий усилитель (УУ);
- матрицу источников тока;
- переключатели тока.

Микросхема имеет независимые выводы для питания аналоговых блоков цифро-аналогового преобразователя. Для питания аналоговых блоков используются выводы ∩VCC3 и ∩0V3.

Блок работает от внутреннего источника опорного напряжения.

Микроконтроллер имеет два токовых выхода IOUTA и IOUTB, которые могут включаться в схеме как на отдельные нагрузки, так и на дифференциальную нагрузку. Дифференциальное напряжение U_{DIFF}, формируемое на нагрузках R_{LOAD}, образуется между U_{OUTA} и U_{OUTB} и может быть преобразовано в однополярное напряжение через трансформатор или дифференциальный усилитель.

Выходное сопротивление токовых выходов можно представить как эквивалент параллельного соединения PMOS ключей с типовым сопротивлением 50 кОм и емкостью 5 пФ.

Токи выходов IOUTA и IOUTB поддерживают свое значение на этих выходах в диапазоне напряжений от минус 1,0 до 1,25 В.

Диапазон выходных напряжений на выходах стабилизированного тока IOUTA и IOUTB в положительной области незначительно зависит от тока полной шкалы I_{OUTFS}. Он ухудшается незначительно от его номинального 1,25 В для I_{OUTFS} = 20 мА до 1,0 В для I_{OUTFS} = 2 мА. Для оптимальной линейности токов I_{OUTA} и/или I_{OUTB} необходимо использовать на выходе микросхемы преобразователь ток–напряжение, что позволяет сохранять неизменное выходное сопротивление. Включение микросхемы с пониженным перепадом напряжения на выходах IOUTA и IOUTB в дифференциальном или в несимметричном включении снижает зависимость сигнала от выходного сопротивления, таким образом, улучшаются характеристики сигнала.

Значительное улучшение характеристик искажений и шумов реализуется дифференциальным включением нагрузки. Оптимальные искажения достигаются, когда максимальный размах сигнала на выходах IOUTA и IOUTB не превышает 0,5 В.

Шумовые характеристики и характеристики искажений слабо зависят от цифрового и аналогового питания, так же как и от тока полной шкалы I_{OUTFS}. При аналоговом напряжении питания 3.0 В обеспечивается максимальный уровень тока источника тока и дифференциальных ключей и обеспечивается улучшение коэффициента нелинейных искажений. Несмотря на то, что максимальный выходной ток можно установить в пределах от 2 до 20 мА, при его значении 20 мА обеспечиваются наилучшие характеристики шума и характеристики искажений. На характеристики шума влияет напряжение цифрового питания (U_{CC1}), выходная частота сигнала и тактовая частота. В заключение приведем оптимальные условия для цифро-аналогового преобразования:

- дифференциальное включение токовых выходов;
- размах положительного напряжения на IOUTA и IOUTB ограничен до 0,5 В;
- выходной ток полной шкалы I_{OUTFS} равен 20 мА.

Источник опорного напряжения

ЦАП имеет внутренний источник опорного напряжения (ИОН). Выходное напряжение внутреннего источника выдается на выход REFIO с нагрузочной способностью не более 100 нА. В этом случае к выходу REFIO обязательно должен быть подключен внешний керамический конденсатор емкостью не менее 0,1 мкФ для обеспечения стабильности опорного напряжения. Допустимо использование внешнего источника опорного напряжения с низким выходным сопротивлением для исключения влияния внутреннего источника.

Управляющий усилитель

Управляющий усилитель позволяет регулировать выходной ток полной шкалы (I_{OUTFS}) в диапазоне от 2 до 20 мА. Для установки тока полной шкалы используется внешний регулировочный резистор (R_{SET}), подключаемый между выводами FSADJ и ∩0V2. При величине сопротивления резистора 2 кОм обеспечивается максимальный ток полной шкалы 20 мА, при сопротивлении 20 кОм – минимальный ток 2 мА. АЧХ усилителя корректируется внутренней емкостью 150 пФ.

Ток полной шкалы I_{OUTFS} является функцией опорного напряжения и сопротивления внешнего резистора:

$$I_{OUTFS} = 32 \times I_{REF},$$

где $I_{REF} = U_{REFIO}/R_{SET}$.

Токовые выходы

Цифро-аналоговый преобразователь имеет комплементарные токовые выходы IOUTA и IOUTB. Сумма токов на выходах ($I_{OUTA} + I_{OUTB}$) равна току полной шкалы I_{OUTFS}. Токи вытекают из выходов во внешнюю нагрузку, подключаемую к общему выводу ∩0V2.

Ток выхода IOUTA практически равен I_{OUTFS}, когда все биты входных данных установлены в состояние «1» (т. е. код ЦАП = 16 383), в то время как на выходе IOUTB нет тока. Ток выходов является функцией входного кода и тока полной шкалы:

$$I_{OUTA} = (DACVALUE/16\,384) \times I_{OUTFS},$$

$$I_{OUTB} = ((16\,383 - DACVALUE)/16\,384) \times I_{OUTFS},$$

где код DACVALUE = 0, 1, 2 … 16 383 – десятичное представление кода.

Токовые выходы обычно подключаются напрямую к нагрузочным сопротивлениям или нагружаются дифференциально на трансформатор. Если требуется соединение по постоянному току, выходы IOUTA и IOUTB должны быть напрямую подключены к правильно подобранным нагрузочным сопротивлениям R_{LOADA}, R_{LOADB}, подключаемым вторым выводом к выводу аналоговой земли ∩0V2. В качестве нагрузки может использоваться кабель с сопротивлением 50 или 75 Ом. Напряжения несимметричного выхода IOUTA и IOUTB соответственно рассчитываются:

$$U_{OUTA} = I_{OUTA} \times R_{LOADA},$$

$$U_{OUTB} = I_{OUTB} \times R_{LOADB}.$$

Напряжения U_{OUTA} и U_{OUTB} во всем диапазоне выходных токов не должны превышать предельно допустимых величин минус 1,0 В и 1,25 В.

Дифференциальное выходное напряжение U_{DIFF} определяется разностью токов I_{OUTA} и I_{OUTB}, соответственно:

$$U_{DIFF} = (I_{OUTA} - I_{OUTB}) \times R_{LOAD},$$

где $R_{LOAD} = R_{LOADA} = R_{LOADB}$.

Заменив значения I_{OUTA}, I_{OUTB} и U_{DIFF}, получаем следующее выражение:

$$U_{DIFF} = \{(2\times DACVALUE - 16\,383)/16\,384\} \times (32R_{LOAD}/R_{SET}) \times U_{REFIO}.$$

Последние два равенства показывают преимущества применения микросхемы в дифференциальном включении выходов. Во-первых, это помогает ликвидировать синфазные помехи, возникающие из-за шума I_{OUTA} и I_{OUTB}, искажений и синфазных отклонений токов. Во-вторых, обеспечивается удвоенное выходное напряжение U_{DIFF} по сравнению с напряжениями несимметричного выхода (U_{OUTA} и U_{OUTB}), таким образом, обеспечивается двойная мощность сигнала в нагрузке.

Улучшение температурного дрейфа обеспечивается для несимметричных (U_{OUTA} и U_{OUTB}) и дифференциальных (U_{DIFF}) выходов посредством подбора сопротивления R_{LOAD}, R_{SET}.

20.2 Установка выходных токов

Установку выходных токов обеспечивают переключатели тока. Необходимый набор токов выдается матрицей источников тока. При этом выходные токи матрицы разделены на три группы: старшие, средние и младшие.

Старшие токи (32 тока) равны опорному току I_{REF}. Один старший ток выдается на выход FSADJ, остальные (31 ток) поступают на переключатели тока. Подключение старших токов к выходам IOUTA, IOUTB определяется состоянием битов DB13 – DB9 управляющего регистра ЦАП.

Средние токи (15 токов) равны I_{REF} /16. Подключение средних токов к выходам IOUTA, IOUTB определяется состоянием битов DB8 – DB5.

Младшие токи (5 токов) равны соответственно I_{REF} /32, I_{REF} /64, I_{REF} /128, I_{REF} /256 и I_{REF} /512. Подключение младших токов к выходам IOUTA, IOUTB определяется состоянием битов DB4 – DB0. Для корректной работы блока необходимо записывать значение DACVAL целым словом, а не побайтно.

20.3 Включение ЦАП

В целях уменьшения общего потребления энергии системой возможен перевод блока в режим пониженного энергопотребления. Существует два способа управления этим переводом. Если установлен бит SLEEPC регистра DACVAL, то перевод в режим пониженного потребления производится автоматически. Это делается обнулением соответствующего бита регистра CLKCL (бит 6) или переводом всего микроконтроллера в режим POWERDOWN. Если бит SLEEPC сброшен, то управление ведется битом SLEEP («1» – ЦАП выключен, «0» – ЦАП включен). По умолчанию блок цифро-аналогового преобразователя выключен. Для включения необходимо установить бит CLKCL.6.

21 Широтно-импульсный модулятор PWM

Встроенный трехканальный широтно-импульсный модулятор предназначен для генерации ШИМ сигналов на выходах микросхемы без участия процессора. ШИМ позволяет формировать импульсы с переменной скважностью и периодом следования.

Функциональные особенности

В состав ШИМ входит селектор опорного сигнала MUX, 8-битный счетчик, три идентичных формирователя сигналов pwm0, pwm1, pwm2 с блоком управления выводами (см. рисунок 21.1). Каждый формирователь состоит из двух регистров и компаратора.

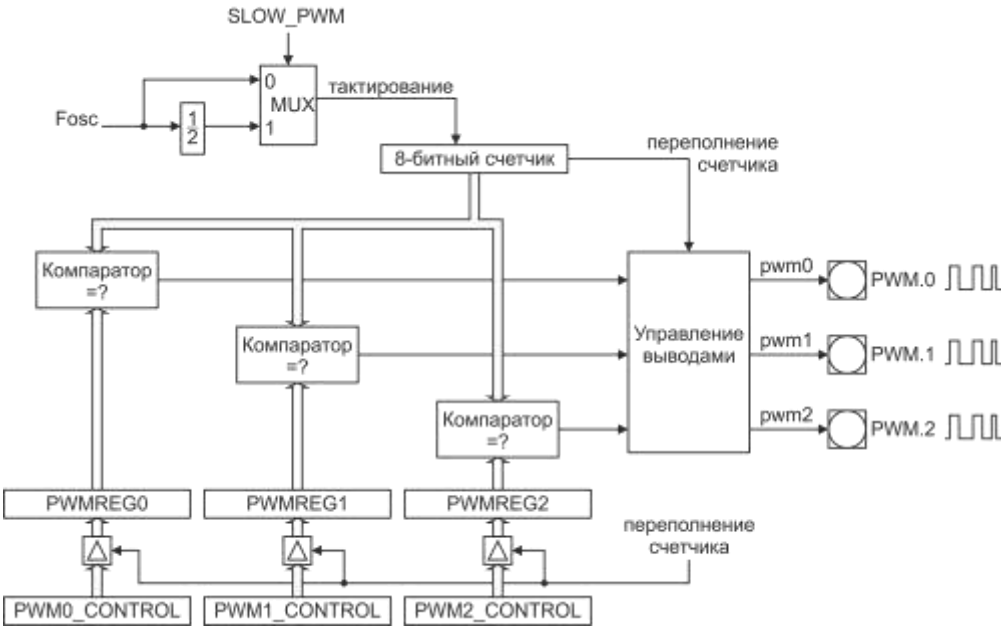


Рисунок 21.1 – Блок-схема модуля PWM

8-битный счетчик тактируется сигналом синхронизации, поступающим с селектора MUX. Счетчик инкрементируется с каждым тактом синхросигнала и по достижении значения FFh обнуляется. Одновременно с этим вырабатывается сигнал переполнения счетчика, который подается на логику управления выводами.

Регистры PWM0_CONTROL, PWM1_CONTROL и PWM2_CONTROL содержат запрограммированные значения длительностей импульсов формируемых сигналов pwm0, pwm1 и pwm2. Каждый раз при переполнении счетчика значения регистров PWM0_CONTROL, PWM1_CONTROL и PWM2_CONTROL загружаются в программно недоступные регистры PWMREG0, PWMREG1 и PWMREG2 соответственно. Такой механизм позволяет точно и легко управлять скважностью формируемых сигналов.

Посредством компараторов содержимое регистров PWMREG0, PWMREG1 и PWMREG2 постоянно сравнивается с содержимым счетчика. С каждым компаратором связан триггер, который удерживает соответствующую линию вывода формируемого сигнала в том или ином логическом состоянии.

Начальное состояние сигналов на выводах PWM.0, PWM.1 и PWM.2 – логическая единица. Компаратор, обнаруживший совпадение значений регистра и счетчика, вырабатывает сигнал, поступающий на логику управления выводами, и на соответствующем выводе устанавливается логический ноль. По сигналу переполнения счетчика на всех выводах устанавливается логическая единица.

На рисунке 21.2 показан пример формирования импульсов на выводах PWM.0, PWM.1 и PWM.2 при постоянных значениях регистров PWM0_CONTROL, PWM1_CONTROL и PWM2_CONTROL.

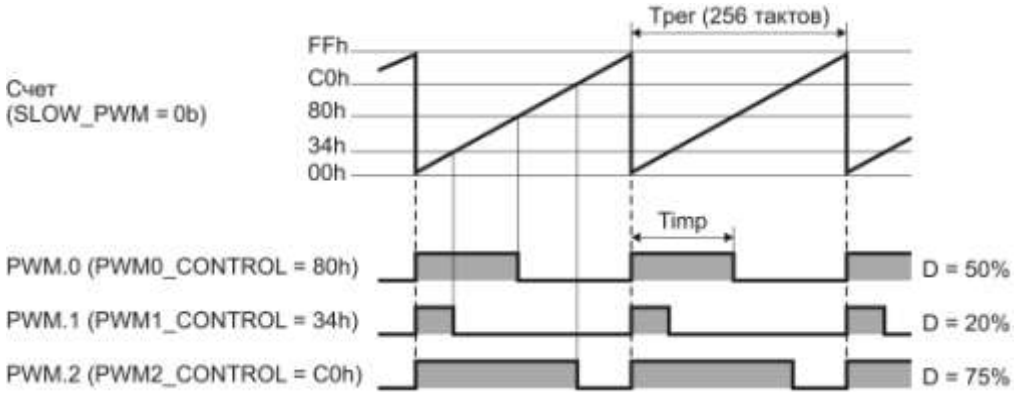


Рисунок 21.2 – Формирование импульсов

Управление модуляцией сигналов

Базовым сигналом тактирования счетчика является сигнал Fpwm, поступающий на блок ШИМ. По умолчанию счетчик будет инкрементироваться с частотой 256 тактов.

Для понижения частоты переключения счетчика следует включит делитель на два посредством бита SLOW_PWM в регистре IOC2.

Период счетчика Tper, мкс, составляет 256 или 512 тактов синхросигнала (см. рисунок 21.2) и, при заданном коэффициенте деления k (1 или 2) и известной частоте синхросигнала Fpwm, МГц, может быть рассчитан по формуле

$$Tper = 256\times \frac{k}{Fpwm} . \qquad (21.1)$$

Для задания длительности импульса Timp (см. рисунок 21.2) каждого сигнала используются регистры PWM0_CONTROL, PWM1_CONTROL и PWM2_CONTROL.

Регистры длительности импульса позволяют управлять коэффициентами заполнения сигналов и таким образом осуществлять широтно-импульсную модуляцию. На рисунке 21.2 для каждого сигнала указан его коэффициент заполнения D (в процентах). Соотношение между длительностью импульса Timp и коэффициентом D следующее

$$D = \frac{IMP_{DEC}}{256} \times 100\ \%, \qquad (21.2)$$

где IMP_{DEC} – значение поля IMP в десятичном формате.

Если в регистр длительности импульса будет записано значение 00h, то при очередном переполнении счетчика на соответствующем выводе будет установлен логический ноль и будет удерживаться до тех пор, пока в регистр длительности импульса не будет записано значение, отличное от нуля.

Если значение IMP равно FFh, то переключение соответствующего вывода в ноль будет происходить каждый раз, когда счетчик будет достигать значения FFh. После сброса счетчика вывод будет переключаться в единицу. Как видно, длительность удержания уровня логического нуля в этом случае составит один такт синхросигнала счетчика.

На рисунке 21.3 показана форма выходного сигнала на выводе PWM.0 для случая, когда значение регистра PWM0_CONTROL равно FFh. Из рисунка также видно, что счетчик переключается с частотой, которая в два раза меньше частоты сигнала Fpwm (обусловлено установленным битом SLOW_PWM).

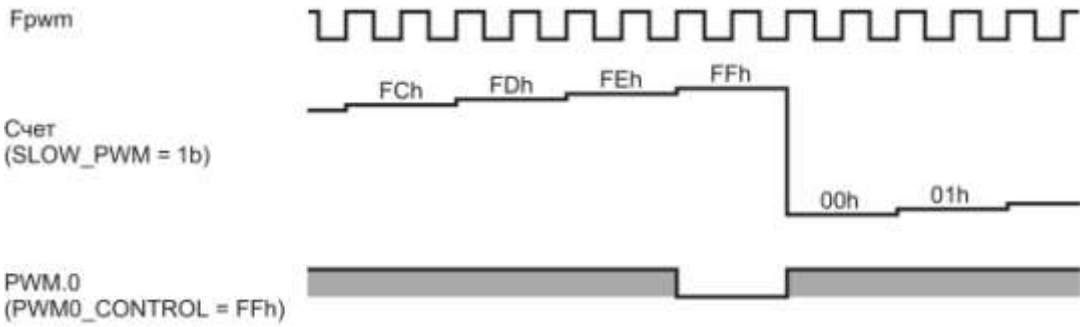


Рисунок 21.3 – Форма выходного сигнала при IMP = FFh

Пример программирования

fosc = 40 МГц и, как следствие, Fpwm = fosc/2 = 20 МГц.

Устанавливается бит SLOW_PWM, что соответствует выбору делителя «2» для частоты сигнала тактирования счетчика. Для расчета длительности периода счетчика используется формула (21.1):

Tper = 256 × k / Fpwm = 256 × 2 / 20 = 25,6 мкс

Коэффициенты заполнения D0, D1, D2 сигналов:
- на выводе PWM.0 – 0,5;
- на выводе PWM.1 – 0,2;
- на выводе PWM.2 – 0,75.

Длительности импульсов:
- Timp0 = Tper × D0 = 25,6 × 0,5 = 12,8 мкс;
- Timp1 = Tper × D1 = 25,6 × 0,2 = 5,12 мкс;
- Timp2 = Tper × D2 = 25,6 × 0,75 = 19,2 мкс.

Согласно формуле (21.2):

- $IMP_{DECPWM0}$ = 256 × 0,5 = 128 и, следовательно, в регистр PWM0_CONTROL следует записать значение 80h;
- $IMP_{DECPWM1}$ = 256 × 0,2 = 51,2 и, следовательно, в регистр PWM1_CONTROL следует записать значение 34h;
- $IMP_{DECPWM2}$ = 256 × 0,75 = 192 и, следовательно, в регистр PWM2_CONTROL следует записать значение C0h.

Этому примеру соответствует поведение сигналов, представленное на рисунке 21.2.

Управление выводами

Передача выходных сигналов ШИМ является альтернативной функцией выводов микроконтроллера P2.5, P1.3 и P1.4. Управление альтернативной функцией осуществляется посредством регистров IOC1 и IOC3.

22 Специальные режимы работы

Микроконтроллер поддерживает три специальных режима работы с пониженным энергопотреблением и режим электрической изоляции.

Режим Idle – работают только встроенные функциональные устройства, а микроконтроллер находится в режиме ожидания разрешенного прерывания от внешнего или внутреннего устройства.

Режим Powerdown – вся внутренняя синхронизация фиксируется в состоянии логического нуля, и генератор отключается. Внутреннее ОЗУ и большинство периферийной памяти сохраняют своё значение, если сохраняется напряжение питания. В этом режиме ток потребления составляет всего несколько микроампер.

Режим Slow – замедляется работа как ЦПУ, так и периферийных устройств, за счет увеличения длительности машинного цикла в два раза. Может включаться и выключаться программно без сброса микроконтроллера.

Дополнительный режим Once – микроконтроллер электрически изолирован от других устройств на печатной плате.

Помимо этих режимов, существует возможность независимого отключения тактовых сигналов периферийных устройств. Это позволяет гибко управлять общим потреблением, используя наиболее оптимальный режим в каждый момент времени работы системы.

22.1 Режим Idle

Тактирование ЦПУ зафиксировано в состоянии логического нуля, но периферия тактируется, и сигнал CLKOUT остается активным. ЦПУ останавливает выполнение команд, но внутреннее ОЗУ, таймеры/счетчики, последовательный порт, сервер периферийных транзакций и система прерываний продолжают функционировать. Потребление энергии уменьшается по сравнению с нормальным режимом работы.

Выводы управления шиной (ALE, RD#, WR#, INST и BHE#) переведены в неактивное состояние. Если порты 3 и 4 использовались, как порты ввода-вывода, они будут сохранять значения, записанные в их защелках. Если эти порты использовались как шина адрес/данные, то выводы будут «плавающими». Исключение – использование PTS передач во внешнюю память.

Если таймер WDT разрешен, то он продолжает работать, поэтому микроконтроллер должен выходить из режима Idle каждые 64К такта, чтобы сбросить таймер.

Вход в режим Idle

Осуществляется по команде IDLPD #1.

Выход из режима Idle

Прерывание или аппаратный сброс выведут устройство из режима Idle. Из-за того, что вся периферия остается активной в этом режиме, любой разрешенный источник прерывания может выработать прерывание. По прерыванию возобновляется тактирование ЦПУ и начинается выполнение соответствующей подпрограммы обслуживания прерывания. Только прерывание, назначенное на обработку стандартным методом, может вывести микроконтроллер из режима Idle. Если обработка осуществляется PTS, то микроконтроллер может выполнять передачи (в том числе во внешнюю память), не выходя из режима Idle. При этом во время передачи выводы управления внешней шиной принимают свои активные состояния.

Если микроконтроллер был выведен из режима Idle стандартной процедурой обработки прерываний, то после завершения подпрограммы обслуживания прерывания ЦПУ выбирает и затем выполняет команду, которая следует за командой IDLPD #1.

22.2 Режим Powerdown

Режим функционирования с очень низким потреблением энергии. Все сигналы внутреннего тактирования зафиксированы в состоянии логического нуля, генератор отключен, блок АЦП переведен в режим Sleep (по умолчанию). Ток потребления АЦП снижается до тока утечки устройства. Если значение входного напряжения сохраняется, то регистры специальных функций (SFR) и регистры ОЗУ сохраняют данные.

Выводы управления шиной (ALE, RD#, WR#, INST и BHE#) переведены в неактивное состояние. Все выходы сохраняют значения, находящиеся в их защелках. Если порты 3 и 4 использовались как порты ввода-вывода (EA# = 0), их выводы будут удерживать последнее выведенное значение. Если порты 3 и 4 использовались как шина адрес/данные (EA# = 1), их выводы будут «плавающими».

Запрещение режима Powerdown

Режим запрещен, когда бит PD регистра CCR сброшен.

Переход в режим Powerdown

До перехода в режим обязательно должен быть завершен обмен через последовательный порт и завершены все аналоговые преобразования. В противном случае возможна потеря данных. Кроме того, если сторожевой таймер разрешен, его регистр должен быть очищен, чтобы устройство смогло выйти из режима Powerdown «чисто», иначе WDT может сбросить микроконтроллер до стабилизации тактового генератора. В режиме Powerdown счетчик WDT не меняет своего значения.

Переход в режим Powerdown осуществляется по команде IDLPD #2.

Примечание – На выводе EXTINT должен удерживаться низкий уровень сигнала, пока устройство находится в режиме Powerdown.

Выход из режима Powerdown

Выход из режима Powerdown обеспечивается одним из трех способов.

1 Удержание низкого уровня сигнала на выводе VPR в течение как минимум 50 нс приведет к разблокированию внутреннего генератора и возобновлению тактирования микроконтроллера.

2 Удержание высокого уровня сигнала на выводе EXTINT в течение как минимум 50 нс. Вывод EXTINT обычно используется как вход внешнего прерывания. В режиме Powerdown он используется как вход, чувствительный к уровню. Бит EXTINT_SRC регистра IOC1 определяет, какой из двух входов EXTINT будет использоваться. Сброшенный бит EXTINT_SRC выбирает вывод P2.2, а установленный – P0.7.

В обоих случаях не будет сброса микроконтроллера, а продолжится выполнение программы с команды, следующей за командой IDLPD #2.

3 Удержание низкого уровня сигнала на выводе RESET# до тех пор, пока генератор не стабилизируется. Схема генератора должна характеризоваться определенным временем наступления стабильной генерации. Когда используется внешнее тактирование, RESET# должен удерживаться в низком уровне, по крайней мере, один такт. Микроконтроллер выйдет из режима Powerdown, произойдет сброс и начнется выполнение программы с начального адреса.

22.3 Режим Slow

Микроконтроллер имеет дополнительный режим уменьшения энергопотребления, при котором внутренняя частота уменьшается в два раза и длительность машинного цикла становится равной четырем тактам сигнала XTAL. Это требует пересчета временных соотношений для периферийных устройств, поскольку длительности всех тактовых сигналов увеличатся в два раза.

Вход в режим Slow

Осуществляется установкой бита SLOW регистра CLKC .

Выход из режима Slow

Осуществляется сбросом бита SLOW регистра CLKC**.**

Данная функция может быть использована на определенных этапах выполнения программы или в случае, если все вычислительные возможности необходимы только в определенные моменты времени.

В режиме Slow доступны режимы Idle и Powerdown, механизмы работы всех периферийных блоков не меняются. Вход и выход из режима Slow занимает один машинный цикл и может осуществляться в любой момент времени.

22.4 Режим Once

Режим тестирования, который электрически изолирует микроконтроллер от других устройств на печатной плате. Все выводы находятся в режиме высокого сопротивления (кроме квазидвунаправленных выводов, которые находятся в режиме слабой единицы), а схема – в режиме Powerdown. В режиме Once на выводе RESET# должен удерживаться высокий уровень сигнала.

Вход в режим Once

Осуществляется, если во время нарастающего фронта сигнала #RESET удерживать низкий уровень сигнала на выводе P2.0.

Выход из режима Once

Осуществляется подачей сигнала RESET#. Вывод P2.0 при этом может быть неподключенным или находиться в состоянии логической единицы.

Для уменьшения энергопотребления в режиме Once одновременно с электрической изоляцией происходит прекращение внутреннего тактирования и выключение генератора (включается режим Powerdown).

Следует помнить, что контроллер может выйти из режима Powerdown, но при этом оставаться в режиме электрической изоляции. Чтобы этого не произошло, необходимо удерживать вывод VPR в состоянии логической единицы, а вывод P2.2 в состоянии логического нуля.

23 Интерфейс внешней памяти

Микроконтроллер допускает подключение различных устройств, реализующих функцию внешней памяти по 8/16-разрядному программируемому интерфейсу с внутренним контролем готовности для медленных устройств.

Подключение внешней памяти и контроль ее функционирования осуществляется посредством выводов и сигналов, указанных в таблице 23.1. Название сигнала совпадает с названием вывода микроконтроллера. Если сигнал имеет бит управления (бит регистра CCR), то этот бит и его состояние указаны под названием вывода микроконтроллера в скобках.

Таблица 23.1 – Сигналы интерфейса внешней памяти

Вывод микро-контроллера	Дополни-тельная функция	Описание сигнала
1	2	3
P4.7 – P4.0 P3.7 – P3.0	AD15 –AD8 AD7 – AD0	Системная 8/16-разрядная мультиплексированная шина адреса/данных. В течение адресной фазы шинного цикла адрес выставляется на шину и фиксируется во внешних устройствах по сигналу ALE/ADV#
ALE (ALE = 1b)		Выходной сигнал с высоким активным уровнем. Информирует о том, что на шине установлен и доступен верный адрес и начался шинный цикл. Сигнал ALE активен только в начале шинного цикла, а потом он переходит в неактивное состояние. Сигнал ALE может использоваться внешним устройством как стробирующий сигнал захвата адреса в регистр-защелку
	ADV# (ALE = 0b)	Выходной сигнал с низким активным уровнем. Информирует о том, что на шине установлен и доступен верный адрес и начался шинный цикл. В отличие от ALE, сигнал ADV# активен в течение всего шинного цикла. Сигнал ADV# может использоваться внешним устройством как стробирующий сигнал захвата адреса в регистр-защелку
BHE# (WR = 1b)		Выходной сигнал с низким активным уровнем. Становится активным только во время записи слова или отдельно старшего байта во внешнюю память. Сигнал BHE# используется одновременно с нулевым разрядом адреса A0
	WRH# (WR = 0b)	Выходной сигнал с низким активным уровнем. В режиме 16-разрядной шины становится активным во время записи слова или отдельно старшего байта. В режиме 8-разрядной шины становится активным во время записи слова или отдельно старшего/младшего байта
BW (BW0 = 1b)	–	Входной сигнал задания разрядности внешней шины, если установлен бит BW0 регистра CCR. Высокий уровень сигнала BW устанавливает 16-разрядную шину, низкий – 8-разрядную. Если бит BW0 сброшен, то шина автоматически устанавливается как 8-разрядная независимо от сигнала BW

Окончание таблицы 23.1

1	2	3
EA#	–	Входной сигнал с низким активным уровнем. В активном состоянии разрешает доступ к внешней памяти. Поскольку микроконтроллер не имеет внутренней памяти программ, вывод EA# желательно заземлить
INST	–	Выходной сигнал с высоким активным уровнем. Становится активным только во время цикла чтения внешней памяти. Высокий уровень сигнала INST информирует о том, что происходит выборка команды, низкий – чтение данных. Сигнал INST может использоваться в приложениях, когда требуется независимость областей памяти команд и данных (байт конфигурации CCB и векторы прерываний считаются данными)
RD#	–	Выходной сигнал с низким активным уровнем. Становится активным во время чтения внешней памяти
READY	–	Входной сигнал с высоким активным уровнем, используемый при работе с медленными внешними устройствами. В активном состоянии информирует микроконтроллер о том, что внешняя память готова для передачи или получения данных
WR# (WR = 1b)		Выходной сигнал с низким активным уровнем. Становится активным во время записи внешней памяти
	WRL# (WR = 0b)	Выходной сигнал с низким активным уровнем. В режиме 16-разрядной шины становится активным во время записи слова или отдельно младшего байта. В режиме 8-разрядной шины становится активным во время записи слова или отдельно старшего/младшего байта

23.1 Регистр CCR и конфигурирование шины

Первый байт, выбираемый из внешней памяти (EA# = 0) из ячейки с адресом 2018h после сброса микроконтроллера, является байтом конфигурации кристалла CCB (байт обычно программируется однократно, когда пользовательская программа откомпилирована и не переопределяется во время обычных операций). Этот байт загружается в регистр CCR, который является регистром конфигурации кристалла. Однажды загруженный, после сброса, регистр CCR не может быть изменен до следующего сброса микроконтроллера. Регистр CCR управляет режимом READY работы внешних устройств, разрядностью и режимом работы внешней шины управления, а также контролирует возможность перехода микроконтроллера в режим пониженного энергопотребления Powerdown.

Если на выводе READY удерживается низкий уровень сигнала во время загрузки регистра CCR, контроллер шины автоматически вставляет три цикла ожидания, что позволяет производить выборку байта CCB из медленной внешней памяти. Количество циклов ожидания можно запрограммировать посредством битов IRC1 и IRC0.

При загрузке байта CCB неважно, какой используется режим включения внешней памяти, так как временная диаграмма загрузки байта с четного адреса и в 16-разрядном, и в 8-разрядном включении одинакова.

Конфигурирование шины

Внешняя шина микроконтроллера является динамической и позволяет изменять разрядность во время работы. Возможностью изменения разрядности шины управляет бит BW0 регистра CCR. Если этот бит сброшен, то шина аппаратно конфигурируется как фиксированная 8-разрядная. В дальнейшем разрядность шины не может быть изменена до сброса микроконтроллера и изменения состояния бита BW0 (см. рисунок 23.1).

Если бит BW0 установлен, то конфигурация шины будет зависеть от состояния сигнала на входе BW. Низкий уровень сигнала будет включать 8-разрядный режим шины с использованием выводов порта 3. Порт 4 в этом случае может функционировать как порт общего назначения. Высокий уровень сигнала на входе BW будет включать 16-разрядный режим шины (см. рисунок 23.1). Переключение уровня сигнала BW нужно производить во время неактивного уровня сигнала RD#/WR#.

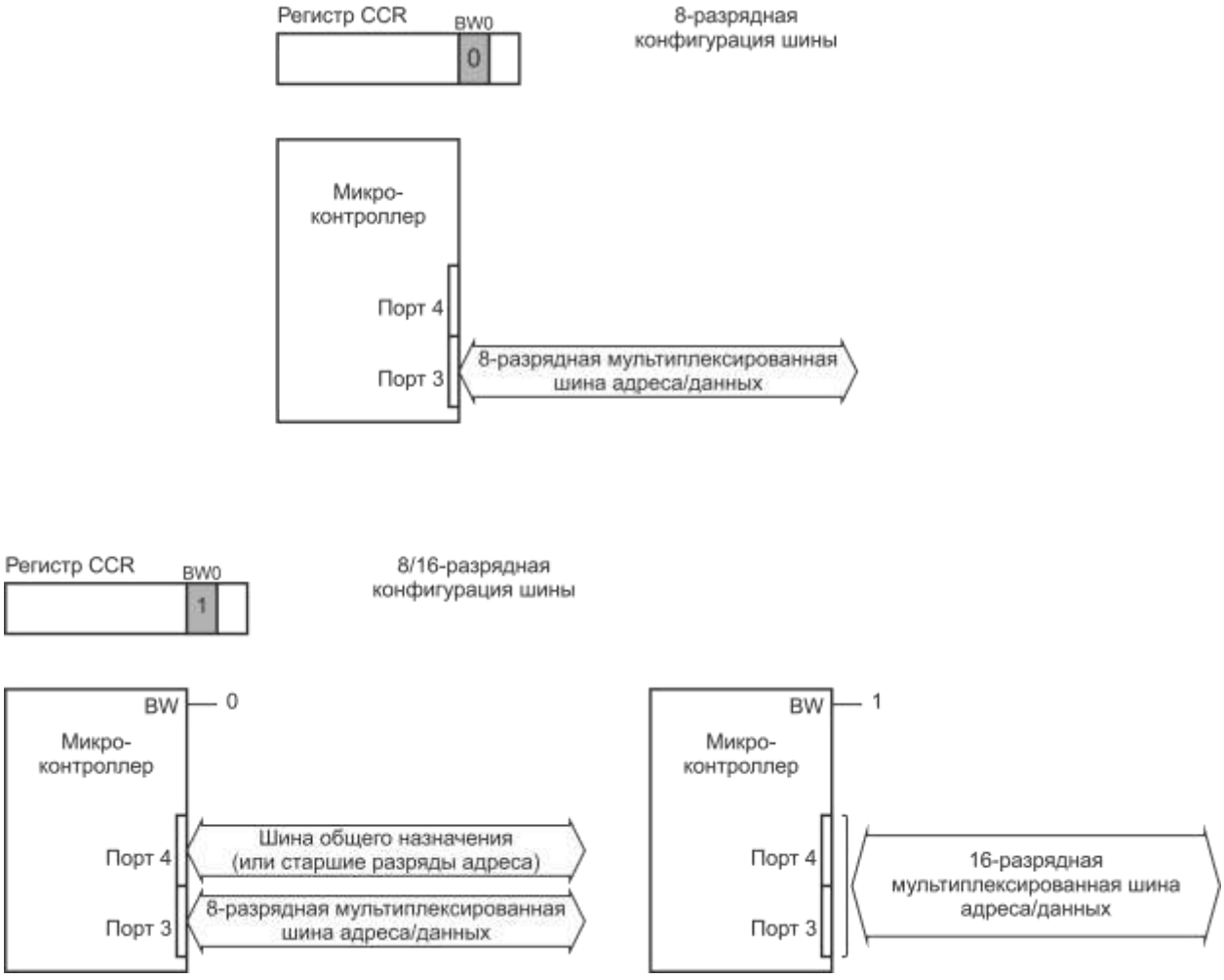


Рисунок 23.1 – Возможные варианты конфигурации шины адреса/данных

Динамическая конструкция шины расширяет границы аппаратного взаимодействия устройств. Например, если микроконтроллер взаимодействует с двумя внешними устройствами памяти – 16-разрядной памятью программ и 8-разрядной памятью данных, – а сигнал BW дополнительно заведен на входы «выбора» этих устройств, то переключение сигнала BW будет попеременно активировать память программ и память данных и соответственно менять разрядность шины.

Следует помнить, что в случае работы с 8-разрядной шиной будет происходить снижение производительности, поскольку для записи/чтения слова потребуется

дополнительный шинный цикл, а предварительная выборка в очередь команд будет недоиспользована.

16-разрядная шина

Используются выводы порта 4 для передачи старших разрядов адреса/данных и выводы порта 3 для передачи младших разрядов адреса/данных мультиплексированной шины AD15 – 0. Упрощенные временные диаграммы для внешних циклов записи и чтения показаны на рисунке 23.2.

По сигналу ALE на шине выставляется адрес, который запоминается в регистре-защелке. До тех пор, пока сигнал ALE в активном состоянии, на шине находится верный адрес.

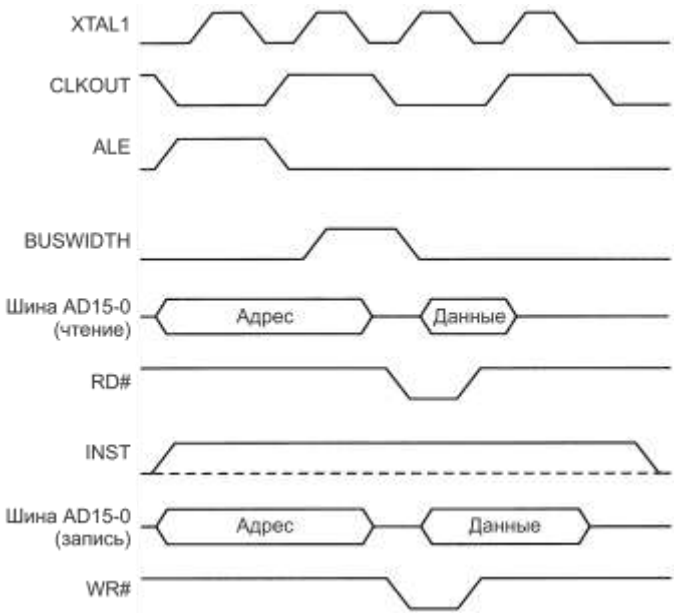


Рисунок 23.2 – Упрощенные временные диаграммы обмена по 16-разрядной внешней шине

В цикле чтения после сброса сигнала ALE контроллер шины переключает сигнал RD# в активное состояние. По ниспадающему фронту сигнала RD# устройство внешней памяти должно выставить данные на шину. Активный уровень сигнала INST во время чтения указывает на то, что идет выборка команды.

В цикле записи после сброса сигнала ALE контроллер шины переключает сигнал WR# в активное состояние. После обнаружения ниспадающего фронта сигнала WR# устройство внешней памяти должно начать считывание данных с шины.

8-разрядная шина

Используются выводы порта 3 для передачи младших разрядов адреса/данных мультиплексированной шины AD7 – 0. Линии AD15 – 8 не мультиплексируются. Старшие разряды адреса не меняются в течение всего шинного цикла. Упрощенные временные диаграммы для внешних циклов записи и чтения показаны на рисунке 23.3.

По сигналу ALE на шине выставляется адрес, который запоминается в регистре-защелке. До тех пор, пока сигнал ALE в активном состоянии, на шине находится верный адрес.

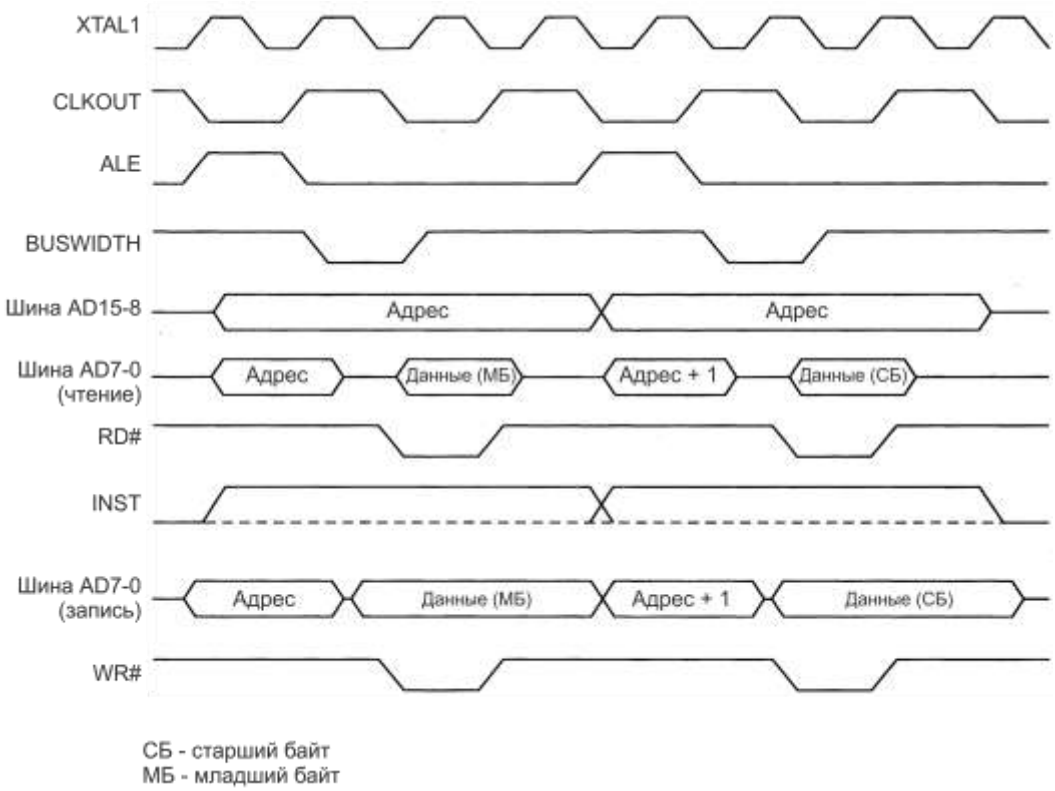


Рисунок 23.3 – Упрощенные временные диаграммы обмена по 8-разрядной внешней шине

В цикле чтения после сброса сигнала ALE контроллер шины переключает сигнал RD# в активное состояние. По ниспадающему фронту сигнала RD# устройство внешней памяти должно выставить данные на шину. При чтении слова контроллер шины выполняет два цикла чтения последовательно – сначала принимается младший байт, а затем старший. Активный уровень сигнала INST во время чтения указывает на то, что идет выборка команды.

В цикле записи после сброса сигнала ALE контроллер шины переключает сигнал WR# в активное состояние. После обнаружения ниспадающего фронта сигнала WR# устройство внешней памяти должно начать считывание данных с шины. При записи слова контроллер шины выполняет два цикла записи последовательно – сначала передается младший байт, а затем старший.

Длительность нормального шинного цикла чтения/записи составляет два машинных такта (два такта сигнала CLKOUT).

Управление длительностью шинного цикла

При работе с медленными внешними устройствами памяти может потребоваться увеличение длительности шинного цикла записи/чтения. Микроконтроллер имеет специальный вывод READY, который соединяется с внешним устройством памяти или другим устройством, контролирующим работу памяти. Когда внешнему устройству требуется дополнительное время для готовности к доступу, оно переводит сигнал READY в низкий уровень и удерживает его в таком состоянии до полной готовности. Как только на выводе READY обнаруживается низкий уровень сигнала, контроллер шины начинает вставлять дополнительные циклы ожидания (см. рисунок 23.4). Каждый цикл ожидания равен одному машинному циклу. Количество циклов задается битами IRC1 и IRC0 регистра конфигурации кристалла CCR. Если заранее неизвестно, какое количество циклов ожидания может потребоваться, то следует установить оба бита. В этом случае

контроллер шины будет вставлять дополнительные циклы ожидания до тех пор, пока на выводе READY будет удерживаться низкий уровень сигнала.

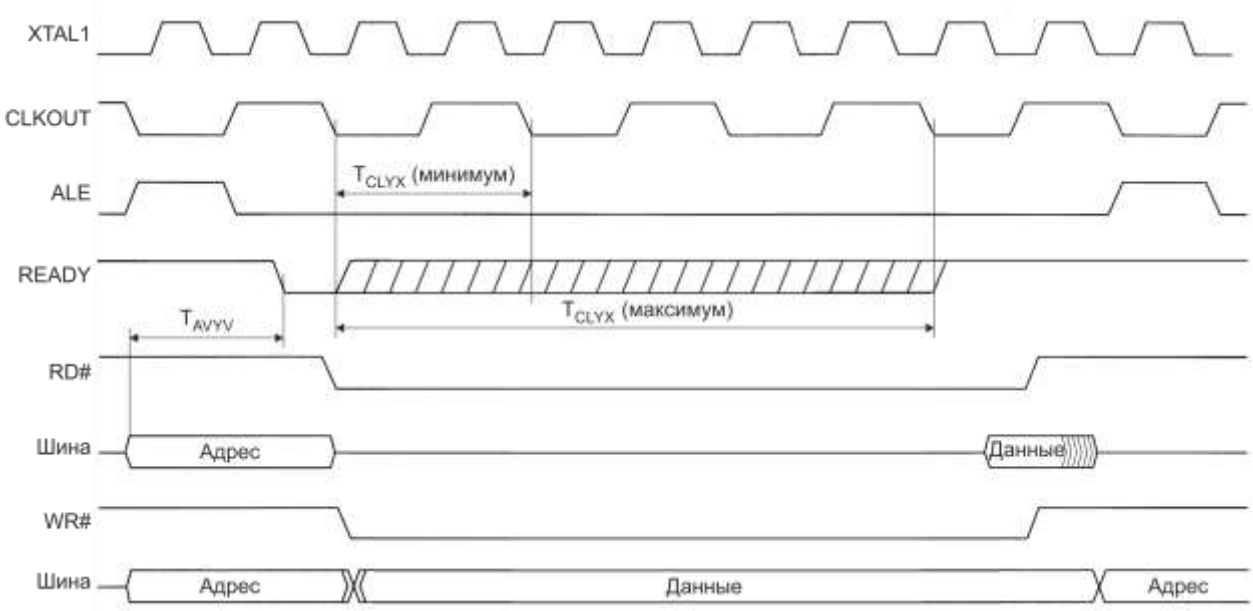


Рисунок 23.4 – Временные диаграммы для сигнала READY

Если необходимо, чтобы в каждый цикл записи/чтения вставлялась задержка в один, два или три цикла ожидания (в зависимости от состояния битов IRC1 и IRC0), то постоянное переключение сигнала READY не требуется – достаточно постоянно удерживать низкий уровень. В противном случае на сигнал READY накладывается ограничение. Проверка состояния вывода READY происходит после выставления адреса на шину по сигналу ALE в промежуток времени T_{AVYV} (см. рисунок 23.4), поэтому переключение в низкий уровень сигнала READY должно происходить не позднее окончания этого временного промежутка. Далее низкий уровень сигнала должен удерживаться до тех пор, пока не будут вставлены все циклы ожидания, т.е. переключение в высокий уровень должно происходить только по окончании временного промежутка T_{CLYX} (см. рисунок 23.4).

23.2 Управление шиной

В зависимости от того, какие группы сигналов управляют шинными циклами чтения/записи, реализуются четыре режима управления шиной. Каждому режиму соответствует своя комбинация состояний битов ALE и WR регистра CCR (см. таблицу 23.2).

Таблица 23.2 – Режимы управления шиной

Режим	Биты регистра CCR		Активные сигналы управления шиной			
	ALE	WR				
Стандартный	1	1	ALE	RD#	WR#	BHE#
Строб записи	1	0	ALE	RD#	WRL#	WRH#
Строб достоверного адреса	0	1	ADV#	RD#	WR#	BHE#
Строб достоверного адреса и записи	0	0	ADV#	RD#	WRL#	WRH#

Стандартный режим

По сигналу ALE на шине устанавливается адрес, сигнал BHE# выбирает область памяти, которая адресуется старшим байтом адреса. Для записи используются сигналы BHE# и WR# (см. рисунок 23.5). Для чтения используется сигнал RD#.

Рисунок 23.5 – Стандартный режим управления шиной

При 16-разрядной конфигурации шины возможна как запись слова данных, так и отдельно младшего или старшего байта. Для этого используются два сигнала записи WRH# и WRL#, передаваемые через выводы BHE# и WR# соответственно. Схема формирования этих сигналов показана на рисунке 23.6.

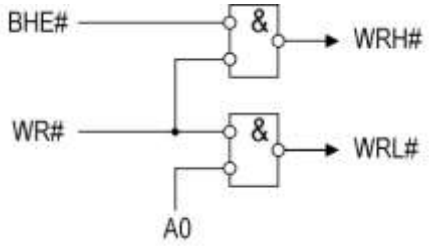


Рисунок 23.6 – Дешифрация сигналов WRL# и WRH#

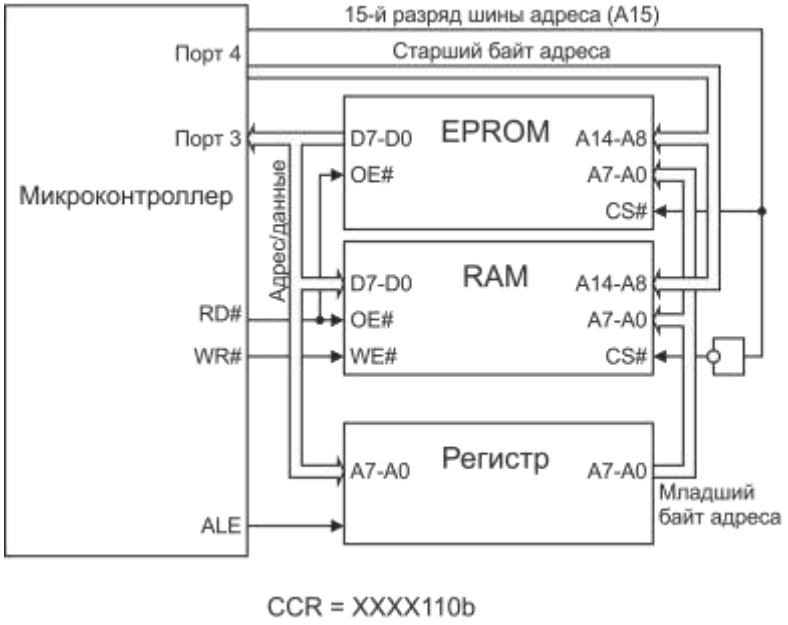


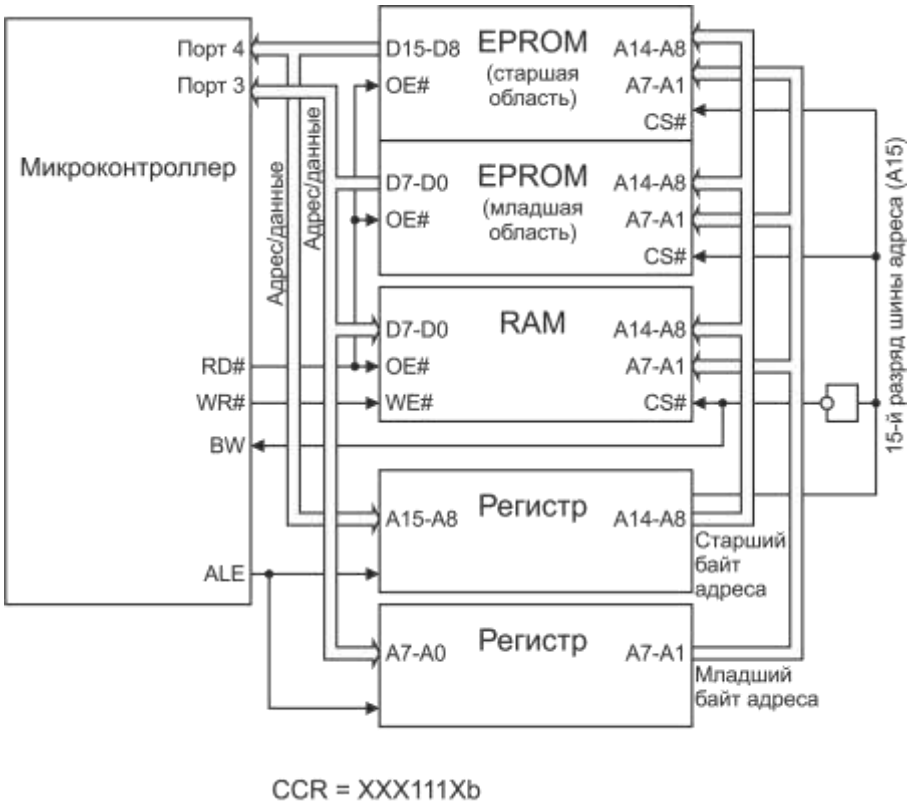
Рисунок 23.7 – Взаимодействие микроконтроллера и внешней памяти

Для чтения данных двух сигналов не требуется – достаточно одного сигнала RD#. При чтении на шину всегда выставляется слово, а в случае, если нужен только один байт,

то второй просто отбрасывается. На рисунке 23.7 показана схема взаимодействия микроконтроллера и внешней памяти с использованием 8-разрядной шины.

Старший разряд адреса (A15) является сигналом выбора устройства. Когда сигнал A15 имеет высокий уровень, выбирается ОЗУ, иначе – ПЗУ. Сигнал ALE дополнительно является стробирующим сигналом для защелкивания адреса (A7-0) в регистре.

На рисунке 23.8 показана схема взаимодействия микроконтроллера и внешней памяти с использованием динамической16-разрядной шины.



CCR = XXX111Xb

Рисунок 23.8 – Взаимодействие микроконтроллера и внешней памяти

Старший разряд адреса (A15) является сигналом выбора устройства. Когда сигнал A15 имеет высокий уровень, выбирается ОЗУ, иначе – ПЗУ. Кроме того, он управляет разрядностью шины, поскольку подключен на вход BW. Сигнал ALE дополнительно является стробирующим сигналом для защелкивания адреса (A15-0) в регистрах.

Режим строба записи

В этом режиме отсутствует внешняя дешифрация адреса при записи старшего и младшего байт данных во внешнюю память при 16-разрядной конфигурации шины. По сигналу ALE на шине устанавливается адрес. Для записи слова или только младшего байта (с четным адресом) используется сигнал WRL#. Для записи слова или только старшего байта (с нечетным адресом) используется сигнал WRH#. При 8-разрядной конфигурации шины сигналы WRL# и WRH# устанавливаются одновременно как для четных, так и для нечетных адресов (см. рисунок 23.9). Для чтения используется сигнал RD#.

Рисунок 23.9 – Режим строба записи

На рисунке 23.10 показана схема взаимодействия микроконтроллера и внешней памяти с использованием динамической 16-разрядной шины.

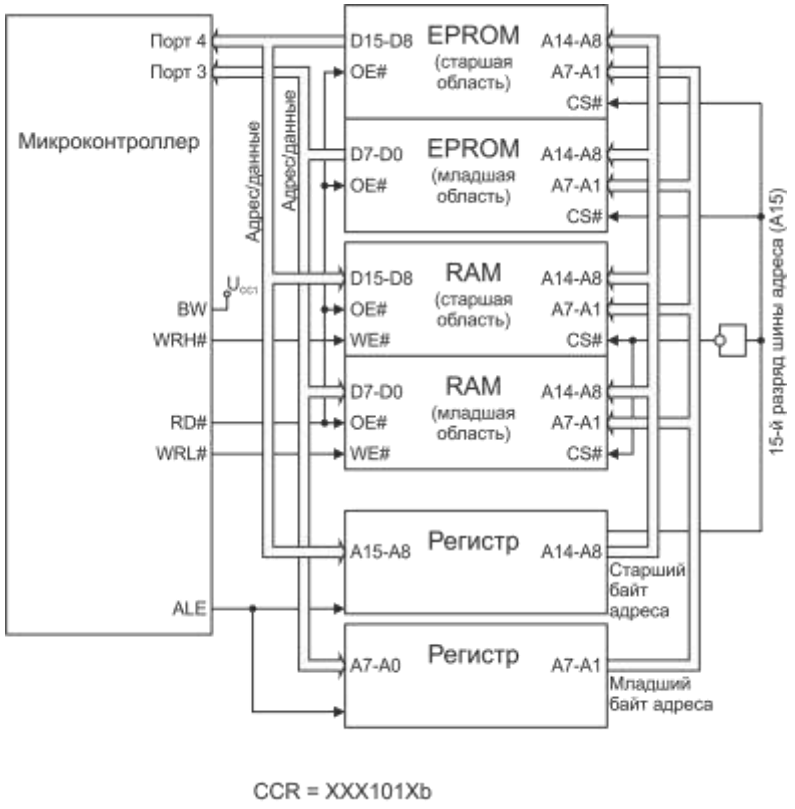


Рисунок 23.10 – Взаимодействие микроконтроллера и внешней памяти

Старший разряд адреса (A15) является сигналом выбора устройства. Когда сигнал A15 имеет высокий уровень, выбирается ОЗУ, иначе – ПЗУ. Сигнал ALE дополнительно является стробирующим сигналом для защелкивания адреса (A15-0) в регистрах. Младший разряд адреса (A0) не используется ОЗУ, поскольку записью старшего и младшего байт управляют сигналы WRH# и WRL#.

Режим строба достоверного адреса

В этом режиме вместо сигнала ALE используется сигнал ADV#, который становится активным после того, как на шине установится адрес (см. рисунок 23.11). Сигнал BHE# выбирает область памяти, которая адресуется старшим байтом адреса. Для записи используются сигналы BHE# и WR#. Для чтения используется сигнал RD#.

Рисунок 23.11 – Режим строба достоверного адреса

Основное отличие работы сигнала ADV# от сигнала ALE в том, что он активен в течение всего шинного цикла. Это позволяет использовать его не только как стробирующий сигнал для защелкивания адреса, а также как сигнал выбора устройства внешней памяти. На рисунке 23.12 показана схема взаимодействия микроконтроллера и внешнего ПЗУ с использованием 8-разрядной шины.

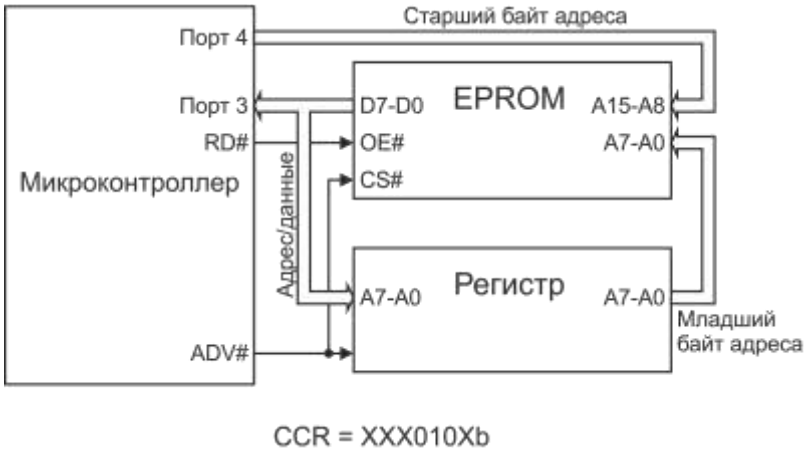


Рисунок 23.12 – Взаимодействие микроконтроллера и внешней памяти

Когда шинные циклы взаимодействия с внешней памятью следуют последовательно друг за другом, вид сигналов ALE и ADV# идентичен. Различие становится очевидным во время простоя шины, поскольку в это время сигнал ADV# имеет высокий неактивный уровень (см. рисунок 23.13).

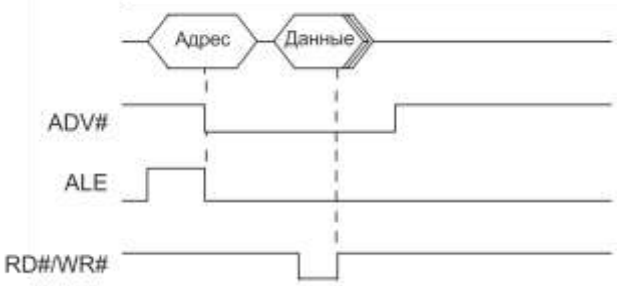


Рисунок 23.13 – Сравнение поведения сигналов ALE и ADV#

На рисунке 23.14 показана схема взаимодействия микроконтроллера и внешнего ПЗУ с использованием динамической 16-разрядной шины.

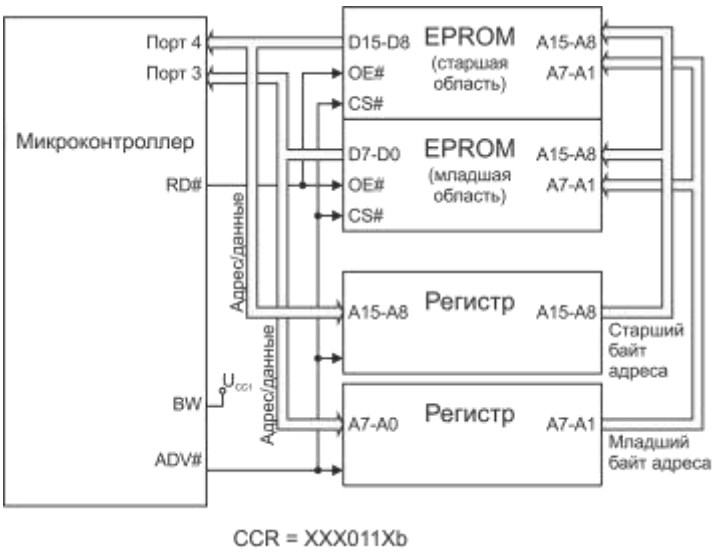


Рисунок 23.14 – Взаимодействие микроконтроллера и внешней памяти

Режим строба достоверного адреса и записи

В этом режиме используются сигналы ADV#, WRH# и WRL#, RD# (см. рисунок 23.15). Режим применяется в простых системах взаимодействия микроконтроллера и внешних устройств с использованием 16-разрядной шины.

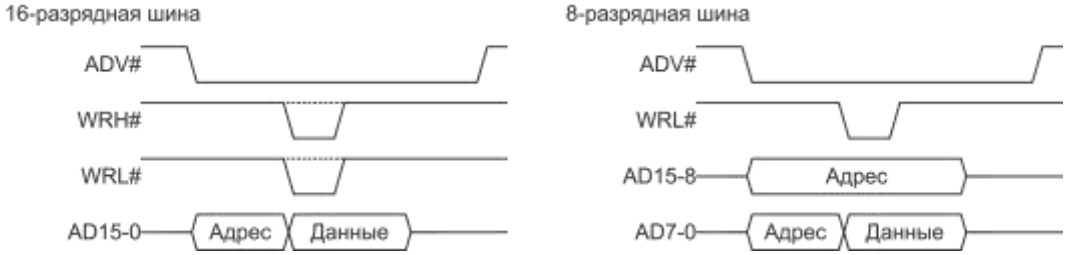


Рисунок 23.15 – Временные соотношения для режима строба достоверного адреса и записи

На рисунке 23.16 показана схема взаимодействия микроконтроллера и внешнего ОЗУ с использованием 16-разрядной шины.

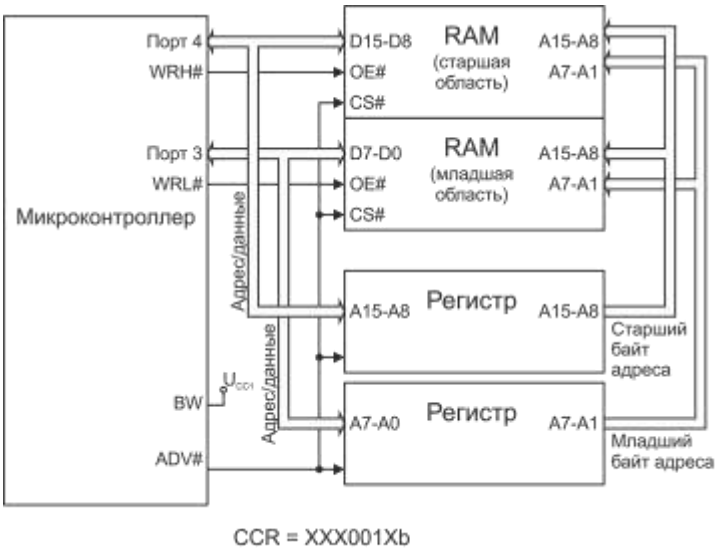


Рисунок 23.16 – Взаимодействие микроконтроллера и внешней памяти

24 Программирование постоянной памяти

Микросхема 1874BE8T содержит многократно программируемое запоминающее устройство (типа EEPROM). В микросхеме под ПЗУ отводится 32 Кбайт адресного пространства с 2000h по 9FFFh.

Адреса с 9000h по 9FFFh отводятся для программы программирования ПЗУ или для отладочных программ. При удержании входа VPR в низком уровне во время сброса, в основной счетчик команд загружается 9000h вместо 2080h и начинается выполнение программы-монитора, которая реализует режимы программирования, описанные в данном разделе (кроме режима RUN-TIME, поддерживаемого аппаратно). Область, выделенная под эту программу-монитор, не имеет никаких дополнительных ограничений на запись пользователем, поэтому может быть использована им как область для резервного старта или как основная область команд (т. е. можно переписать программу, которая идет по умолчанию). Но в этом случае пользователь сам должен позаботиться о возможности программирования ИС, расположив свою программу по удобным ему адресам.

Данный раздел предназначен для помощи потребителю в программировании микроконтроллеров, которое может осуществляться несколькими способами. Могут использоваться автоматический режим (AUTO), режим с использованием специального программатора (SLAVE), режим программирования во время исполнения программы (RUN-TIME) или режим программирования через последовательный порт 0 (SERIAL). Также можно выбрать режим вывода содержимого памяти (ROM-DUMP) для верификации содержимого ROM. Появилась возможность производить полный цикл разработки программ в интегрированной среде разработки CodeMaster-96, в том числе доступны режимы отладки и программирования ИС с помощью аппаратного отладочного устройства JEM-96. В разделе описаны режимы программирования, их выбор и работа в них.

24.1 Общее описание

Когда выбирается режим программирования (за исключением режима RUN-TIME), устройство отзывается на него выполнением внутреннего алгоритма, который находится в области ПЗУ, начиная с адреса 9000h. В общем случае внутренний алгоритм выполняет функции программирования под руководством внешних источников информации.

Выбор потребителем режима программирования определяет, какой алгоритм выбирается из ПЗУ. Чтобы выполнить программирование правильно, устройство должно иметь следующую минимальную схему включения: XTAL1 подключен, неиспользованные входы связаны с логической единицей или нулем, как этого требует режим, питание и земля подключены. Условия работы, заданные в описании устройства, должны быть выполнены.

В данном разделе рассматриваются требования для каждого режима программирования.

24.2 Режимы программирования

Внутренняя программа микроконтроллера поддерживает пять способов программирования EEPROM.

Режим автоматического программирования AUTO дает возможность самопрограммирования микроконтроллера из внешнего EEPROM, без использования специального программатора. Этот режим позволяет потребителю использовать прототип схемы проектируемого устройства для программирования.

Режим программирования SLAVE требует специального программатора. При использовании этого режима потребитель может программировать или верифицировать отдельные слова или наборы слов в EEPROM.

Режим RUN-TIME позволяет программировать отдельные ячейки EEPROM во время обычного выполнения программы полностью под управлением программного

обеспечения. В этом случае потребителю не нужно специально устанавливать режим программирования.

Режим программирования через последовательный порт 0 SERIAL. В этом случае управляющие команды и данные поступают через порт USART0.

Режим ROM-DUMP позволяет верифицировать содержимое EEPROM путем записи массива из EEPROM во внешнюю память.

Данные режимы (кроме RUN-TIME) реализуются по умолчанию встроенным программным обеспечением НИИЭТ. Пользователь может переписать это программное обеспечение, реализовав свой алгоритм записи.

Режим RUN-TIME является аппаратным режимом и поддерживается ИС изначально. Его нельзя переписать.

В случае, если по каким-либо причинам испортилось встроенное программное обеспечение, есть возможность альтернативного старта с адреса 0A000h (всегда внешний адрес) для перепрограммирования. В этом случае алгоритм перезаписи ПЗУ должен содержаться во внешней памяти.

Выбор режима программирования

Для выбора режима программирования используются сигналы PMODE (PMODE.0 – PMODE.3). В таблице 24.1 описаны функции PMODE, указываются значения PMODE и режимы программирования.

Следует ввести микроконтроллер в режим программирования удержанием сигнала VPR в низком уровне во время нарастания сигнала RESET#.

PMODE выбирает один из пяти режимов (режим RUN-TIME не выбирается). Выводы P0.4 – P0.7 задают шестнадцатеричное значение PMODE. Четыре режима – это SLAVE, ROM-DUMP, AUTO и SERIAL. PMODE фиксируется командами (неаппаратно) в самом начале программы-монитора и должен быть стабилен некоторое время после выполнения сброса. Также предусмотрена возможность сброса параметров отладки, сохраняемых в EEPROM, путём обеспечения соответствующего состояния на выводах PMODE.

Таблица 24.1 – Описание функций PMODE и его значений

P0.7 (PMODE3)	P0.6 (PMODE2)	P0.5 (PMODE1)	P0.4 (PMODE0)	Режим работы
0	0	0	1	DEBUG-SPI *
0	0	1	0	DEBUG-USART0 *
0	0	1	1	DEBUG-USART1 *
0	1	0	0	RESET-DEBUG-CONF
0	1	0	1	SLAVE
0	1	1	0	ROM-DUMP
1	1	0	0	AUTO
1	1	1	0	SERIAL

Примечания

1 В таблице знаком * отмечены состояния выводов, влияющие на выбор режима работы в интегрированной среде разработки CodeMaster-96 с помощью аппаратного отладочного устройства JEM-96.

2 Состояние RESET-DEBUG-CONF соответствует сбросу параметров отладки, хранящихся в EEPROM.

24.3 Функции выводов в режиме программирования

Чтобы обслуживать различные режимы программирования, некоторые выводы приобретают новые функции. В таблице 24.2 описаны эти новые функции. Назначение выводов в режиме программирования – на рисунке 24.1.

Таблица 24.2 – Описание функций выводов в режиме программирования

Имя функции	Тип вывода	Режим	Описание
AINC# P2.4	Вход	SLAVE	Автоинкремент. Активный низкий уровень на входе разрешает автоинкремент. Автоинкремент позволяет считывать и записывать в ячейки EEPROM последовательно, без передачи адреса при каждом обмене
CPVER P2.6	Выход	SLAVE	Полная верификация программирования. Высокий уровень на этом выводе в режиме SLAVE показывает, что все ячейки запрограммированы верно
EA#	Вход	Все режимы програм-мирования, исключая RUN-TIME	Внешний доступ. Если на EA# подано «1» во время нарастания сигнала RESET#, а на вход VPR подан «0», устройство входит в режим программирования. EA# фиксируется только при нарастании сигнала RESET#. Изменение уровня на входе EA# после сброса не дает эффекта
PACT# P2.7	Выход	AUTO	Программирование активно. В режиме программирования AUTO низкий уровень на выходе показывает, что идет процесс программирования, а высокий уровень индицирует окончание программирования
PALE# P2.1	Вход	SLAVE	В режиме SLAVE по заднему фронту PALE считывается адрес/команда из портов 3, 4
PBUS Порты 3, 4	Вход/ Выход	Все режимы програм-мирования, исключая RUN-TIME	Шина адрес/команда/ данные. Используется как системная шина для доступа к внешней памяти в режимах ROM-DUMP и автопрограммирования. В режиме SLAVE выводы PBUS требуется подсоединять к #VCC1 через резистор
PMODE P0.4 – P0.7	Вход	Все режимы програм-мирования, исключая RUN-TIME	Выбор режима. Определяет, какой алгоритм выбран для программирования EEPROM. Сигналы PMODE фиксируются после сброса устройства и должны сохранять свое значение во время операции. В таблице 24.1 приведены значения PMODE и режимы программирования, им соответствующие
PROG# P2.2	Вход	SLAVE	Программирование. В режиме SLAVE по спаду сигнала фиксируются данные на PBUS и начинается программирование, по фронту программирование заканчивается
PVER P2.0	Выход	AUTO SLAVE	Верификация программирования. В режимах SLAVE и AUTO высокий уровень сигнала PVER – при успешном программировании ячейки, низкий – при обнаружении ошибки
VPR	Вход	Все режимы	Вход в режимы программирования. Осуществляется подачей низкого напряжения во время нарастания сигнала RESET#

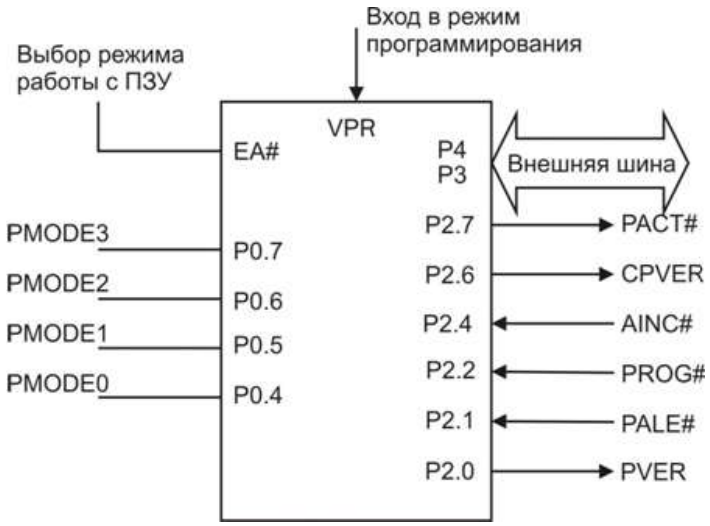


Рисунок 24.1 – Назначение выводов в режиме программирования

24.4 Распределение памяти

Содержимое памяти программ, находящейся по адресам 2080h – 9FFFh, и памяти специального назначения (2000h – 207Fh) определяется пользователем. В таблице 24.3 описаны начальные и конечные адреса памяти специального назначения микроконтроллера шестнадцатеричной и десятичной систем исчисления, на рисунке 24.2 – карта памяти.

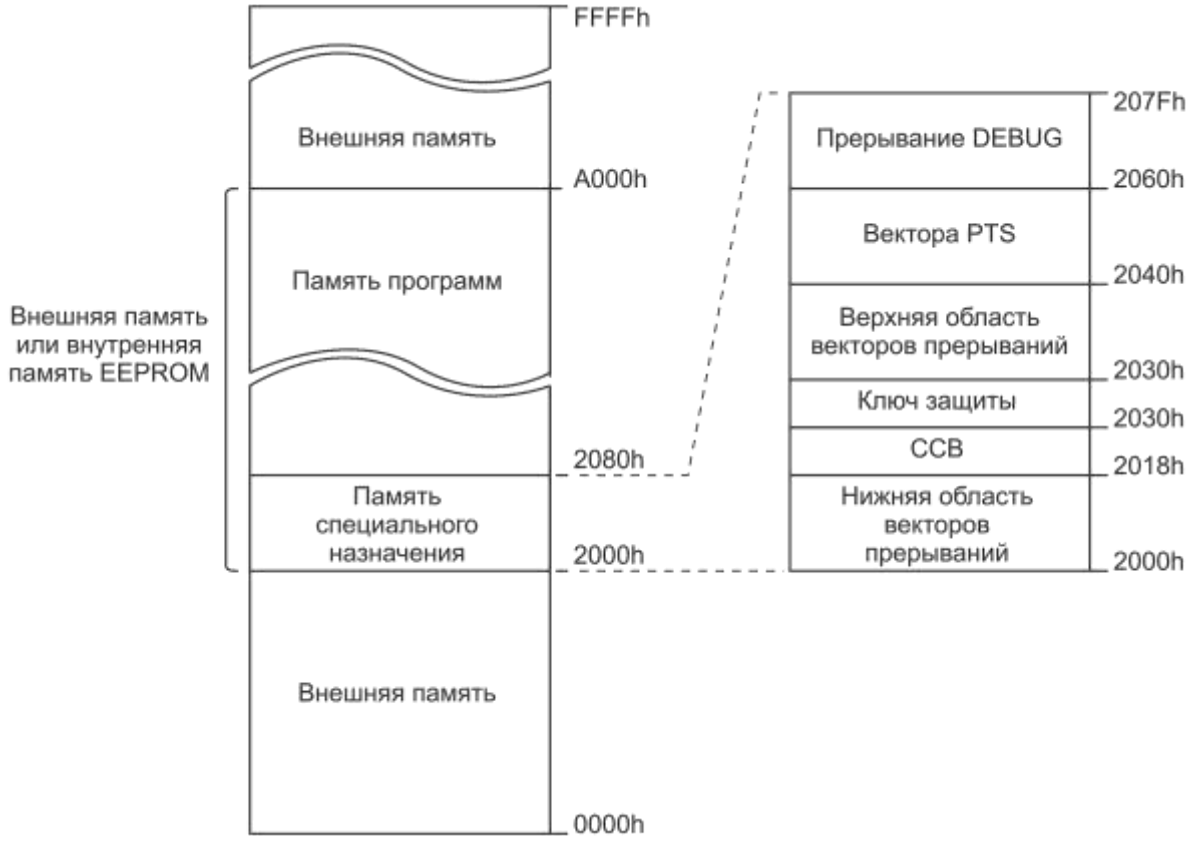


Рисунок 24.2 – Карта памяти специального назначения микроконтроллера

Таблица 24.3 – Адреса памяти специального назначения микроконтроллера

Обозначения	Диапазон адресов
Общего назначения	2062h – 207Fh
Вектор прерывания DEBUG	2060h
Общего назначения	205Eh – 205Fh
Векторы PTS	2040h – 205Dh
Старшие векторы прерывания	2030h – 203Fh
Общего назначения	2019h – 202Fh
CCB	2018h
Общего назначения	2014h – 2017h
Младшие векторы прерывания	2000h – 2013h

24.5 Последовательность включения питания

1 Подать напряжение низкого уровня на вывод RESET#, пока U_{CC1} не стабилизируется. В это время VPR и EA# выводы могут быть не подключены.

2 После стабилизации генератора и U_{CC1}, оставить устройство в состоянии сброса и подать напряжение U_{OH2} на EA#. После стабилизации EA# подать напряжение U_{OL} на вывод VPR.

3 Установить значение PMODE для выбора алгоритма программирования. Значения для каждого режима приведены в таблице 24.1.

4 Подать высокий уровень на вывод RESET#.

5 Работать с выбранным алгоритмом программирования.

24.6 Защита памяти

Аппаратная и программная защита – это отдельные части в схеме защиты памяти. Аппаратная защита предохраняет от внешней записи и считывания следующим образом. В регистре состояния схемы (CCR) устанавливаются биты кода безопасности, CCR.6 (LOC0) и CCR.7 (LOC1), запрещая чтение и/или запись в EEPROM. Попытка записи вызовет цикл записи контроллера шины, однако, записи не произойдет. Попытка чтения ПЗУ при выполнении программы из внешней памяти приведет к сбросу ИС.

Программная защита предотвращает неавторскую запись или верификацию EEPROM следующим образом. Механизм защиты по ключу ограничивает доступ во внутреннюю память. Если устройство введено в режим программирования, то алгоритм режима требует проверки ключа перед началом работы. Если проверка не пройдена, устройство зацикливается, пока не произойдет сброс.

Также для защиты возможно использование соответствующих функций модуля отладки.

Биты кода защиты в CCR

Регистр состояния схемы CCR устанавливает структуру шин и другие параметры устройства во время программирования. Процедура сброса загружает CCR из ячейки с адресом 2018h, где расположен байт конфигурации устройства CCB. Это обеспечивает защиту как в течение нормального выполнения программы, так и во время функционирования программы-монитора. Установка битов LOC1 или LOC0 воздействует на режим RUN-TIME и на любые попытки доступа к внутренней памяти от внешних устройств. При вводе микроконтроллера в режим программирования всего лишь меняется начальный адрес запуска программы (на 9000h), но CCB загружается с того же адреса, что и при нормальной работе (2018h).

Следует установить LOC0, чтобы сделать невозможной запись и предотвратить дальнейшее программирование области ПЗУ 2000h–9FFFh. Попытка записи вызывает цикл записи контроллера шины, однако, записи не произойдет.

Следует установить LOC1, чтобы запретить программе во внешней памяти доступ к содержимому EEPROM. Внутренняя память недоступна для считывания. Чтение данных может происходить, если основной счетчик команд находится в области 2000h – 9FFFh. При выполнении кода из внешней памяти любое чтение из области внутреннего ПЗУ приведет к сбросу ИС (рисунок 24.3).

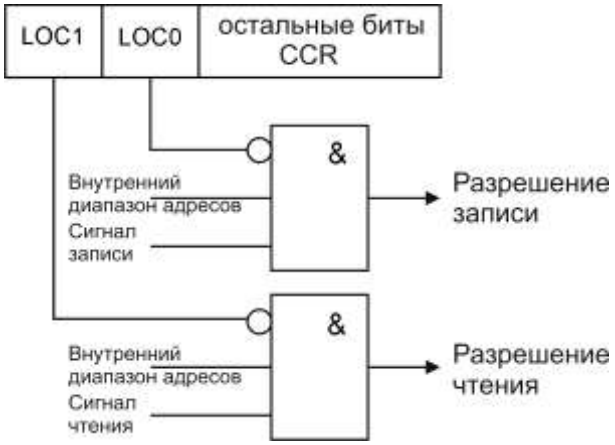


Рисунок 24.3 – Байт конфигурации кристалла

Режимы программирования требуют проверки ключа защиты до начала выполнения программирования. По умолчанию ПЗУ содержит нули, поэтому при первом программировании необходимо подавать нули в качестве ключа защиты.

Ключ защиты

Ключ – это 128-битная область во внутренней памяти с адреса 2020h до 202Fh. В этих ячейках по умолчанию содержатся нули. Для активации ключа защиты необходимо записать пользовательские данные в эти ячейки. Ячейки ключа защиты доступны при чтении из внешней памяти, и их использование не имеет смысла без установки LOC1. Разделы о режимах AUTO, SLAVE, ROM-DUMP, SERIAL содержат информацию о проверке ключа в каждом режиме. Если проверка ключа не пройдена, устройство входит в бесконечный цикл и должно быть сброшено для выхода из него.

Программирование ключа защиты

Если во время записи ключа произошел скачок напряжения или сброс, может случайно записаться неизвестный ключ, делающий устройство недоступным для последующего программирования. Чтобы избежать этого, основной массив EEPROM программируют до программирования битов кода защиты CCB. Но запись битов LOC1 и LOC0 не является необратимой, то есть микроконтроллер можно полностью стереть для последующего использования. Если включена защита записи ПЗУ, то подача команды на стирания любого блока памяти приведет к полной очистке памяти. Если же необходимо не допустить возможности стирания ПЗУ при защите, то необходимо обеспечивать соответствующую программную или аппаратную (при помощи блока отладки) блокировку доступа к регистру управления программированием. Так как процедура программирования состоит всегда из двух шагов – записи регистра ROMC и подачи непосредственной команды записи данных, то возможно прерывание последовательности их выполнения прерыванием DEBUG. Если были запрограммированы биты защиты, то защита включится только после перезагрузки микроконтроллера.

24.7 Режим RUN-TIME

Этот режим является аппаратно поддерживаемым и обеспечивает формирование нужных управляющих сигналов для корректного программирования EEPROM. Ячейки

EEPROM можно программировать во время обычного выполнения программы. При этом микроконтроллер должен работать в режиме работы с внутренней памятью. Перед непосредственной подачей команды записи необходимо выбрать соответствующий тип операции в управляющем регистре ROMC. Операцией может быть: запись слова, очищение строки памяти или очищение целого блока памяти.

Большая часть битов регистра ROMC является зарезервированной для режимов тестирования ПЗУ. Установка этих битов может привести к непредсказуемым последствиям.

Для стирания блоков памяти используется бит BW.

В таблице 24.4 представлены начальные и конечные адреса блоков, в таблице 24.5 – адреса строк на примере первого блока памяти.

Таблица 24.4 – Адреса блоков памяти EEPROM

Номер блока	Диапазон адресов
1	2000h – 27FFh
2	2800h – 2FFFh
3	3000h – 37FFh
4	3800h – 3FFFh
5	4000h – 47FFh
6	4800h – 4FFFh
7	5000h – 57FFh
8	5800h – 5FFFh
9	6000h – 67FFh
10	6800h – 6FFFh
11	7000h – 77FFh
12	7800h – 7FFFh
13	8000h – 87FFh
14	8800h – 8FFFh
15	9000h – 97FFh
16	9800h – 9FFFh

Таблица 24.5 – Адреса строк в первом блоке памяти EEPROM

Номер строки	Диапазон адресов
1	2000h – 200Fh
2	2010h – 201Fh
3	2020h – 202Fh
4	2030h – 203Fh
…	…
…	…
…	…
128	27F0h – 27FFh

Ниже представлен пример программы, осуществляющей стирание первого блока памяти и записи значения 5555h в ячейку с адресом 2090h.

```
LD AX,#1FFCh
LDB BX,#05h
STB BX,[AX]          ;enable erasing block

LD AX,#2000h
```

```
LD BX,#0000h
ST BX,[AX]

LD AX,#1FFCh
LDB BX,#01h
STB BX,[AX]          ;enable programming cell

LD AX,#2090h
LD BX,#5555h
ST BX,[AX]
```

Все режимы программирования используют RUN-TIME режим, только с разными источниками входных данных (последовательный порт, порты 3 и 4, интерфейс внешней памяти).

При записи слова микроконтроллер формирует управляющие сигналы для EEPROM нужной длительности. Для корректного осуществления программирования они не должны быть меньше установленных величин.

Регистр PPW хранит коэффициент для вычисления длительности программирующего импульса. Длительность должна быть не менее 4,5 мс.

Регистр PPSETUP хранит коэффициент для вычисления времени установки данных относительно переднего фронта программирующего импульса. Время установки должно быть не менее 1 мкс.

Регистр PPHOLD хранит коэффициент для вычисления времени удержания данных относительно заднего фронта программирующего импульса. Время удержания должно быть не менее 1 мкс.

Общая длительность как программирования одного слова, так и стирания одной строки или одного блока суммируется из времени установки, длительности программирующего импульса и времени удержания (рисунок 24.4).

Рисунок 24.4 – Программирующий импульс EEPROM

Длительность программирующего импульса рассчитывается по формуле

$$T_{PPW} = (PPW \times 16) \times Tмц,$$

где PPW – значение в десятичном представлении;

Тмц – длительность машинного цикла.

По умолчанию значение PPW соответствует 4,8 мс на частоте 33 МГц.

Время установки программирующего импульса рассчитывается по формуле

$$T_{PPSETUP} = PPSETUP \times Tмц,$$

где PPSETUP – значение в десятичном представлении;

Тмц – длительность машинного цикла.

По умолчанию значение PPW соответствует T_{PPW} = 4,8 мс на частоте 33 МГц.

Время установки программирующего импульса рассчитывается по формуле

$$T_{PPSETUP} = PPSETUP \times Тмц,$$

где PPSETUP – значение в десятичном представлении;

Тмц – длительность машинного цикла.

По умолчанию значение PPSETUP соответствует $T_{PPSETUP}$ = 1,2 мкс для частоты 33 МГц.

Время удержания программирующего импульса рассчитывается по формуле

$$T_{PPHOLD} = PPHOLD \times Тмц,$$

где PPHOLD – значение в десятичном представлении;

Тмц – длительность машинного цикла.

По умолчанию значение PPHOLD соответствует T_{PPHOLD} = 1,2 мкс для частоты 33 МГц.

После подачи команды на запись в EEPROM микроконтроллер сам формирует сигналы нужной длительности, а на время программирования переводит ядро в режим пониженного энергопотребления IDLE. В случае, если планируется программирование всей памяти или стирание большого количества блоков, рекомендуется отключить периферийные устройства (особенно АЦП и ЦАП). Это позволит уменьшить потребление цифровой части схемы, обеспечив более стабильное питание ПЗУ.

24.8 Режим программирования AUTO

Этот режим является альтернативным режимом, обеспечивающим программирование с минимальными затратами. Можно программировать микроконтроллер без использования программирующего устройства. Прибор программирует сам себя, выбирая данные из внешнего EEPROM (ячейки 0A000h–0FFFFh и 0000h–0FFFh, смотри таблицу 24.6). CCB загружается в регистр CCR. Микроконтроллер получает данные через внешнюю шину; поэтому CCB должен соответствовать системе памяти, установленной в режиме программирования, которая необязательно соответствует системе памяти в разрабатываемом проекте. В начале работы алгоритма режима AUTO осуществляется проверка кода защиты. Если проверка не прошла, устройство входит в бесконечный цикл и должно быть возвращено в исходное состояние через сброс (смотри таблицу 24.7).

Таблица 24.6 – Карта памяти автоматического режима

Адреса внешнего EEPROM	Адреса внутреннего EEPROM	Описание
0000h – 0FFFh	8000h – 8FFFh	Зарезервированные для данных и команд ячейки
A000h – FFFFh	2000h – 7FFFh	Зарезервированные для данных и команд ячейки
A020h – A02Fh	2020h – 202Fh	Ключ во время проверки

Таблица 24.7 – Функции выводов в режиме AUTO

Имя	Тип	Описание
PACT# (P2.7)	Выход	Активность программирования. Нуль указывает на продолжающееся программирование
PBUS (порты 3 и 4)	Вход / Выход	Шина адреса/управления/данных. Используется для доступа к внешней памяти
PVER (P2.0)	Выход	Проверка. Сигнал корректируется после каждого такта. Высокий уровень сигнала указывает на успешное программирование, низкий – на ошибку

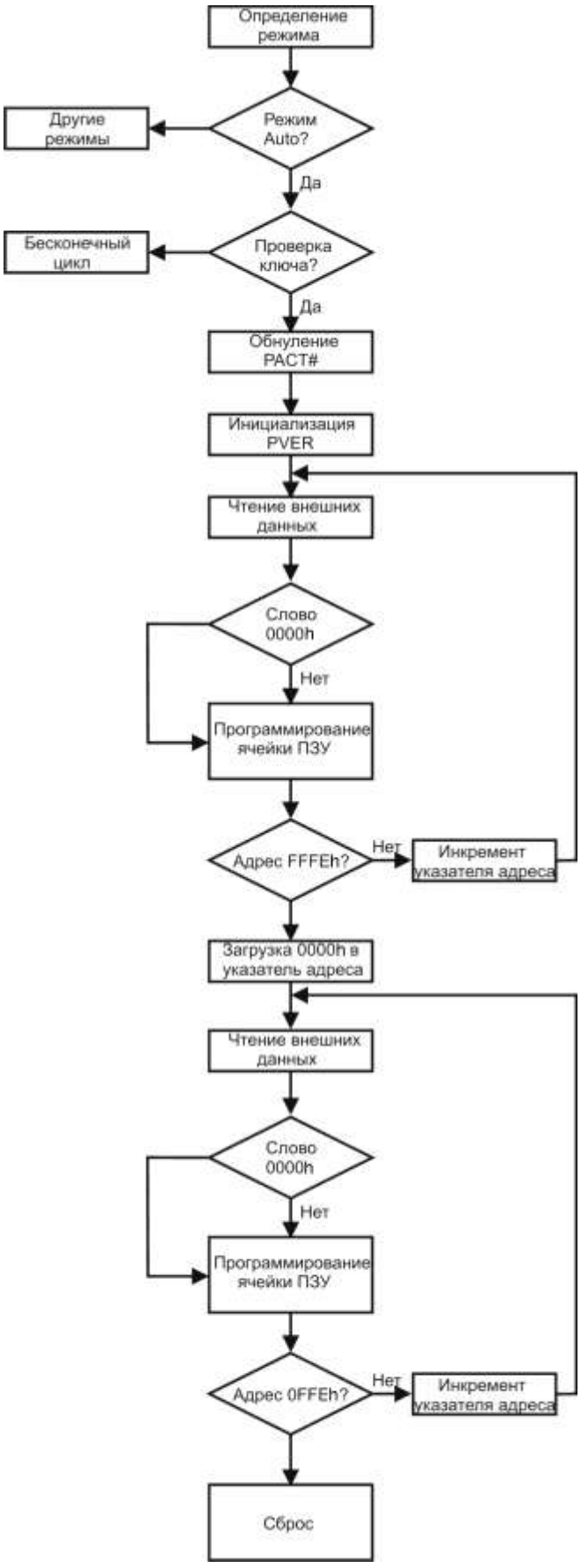


Рисунок 24.5 – Алгоритм режима AUTO

Алгоритм режима автоматического программирования проверяет ключ до начала работы. Сравнивается внешняя область A020h – A02Fh с внутренней 2020h – 202Fh. Внешняя и внутренняя области должны совпадать, иначе устройство входит в бесконечный цикл, предотвращающий программирование.

Чтобы войти в режим AUTO, необходимо установить PMODE в 0Ch и подать питание в последовательности, приведенной в подразделе 24.5. На рисунке 24.5 описана программа, содержащаяся в ПЗУ.

Последовательность операций в режиме AUTO:

1 Определение режима программирования (чтение PMODE).

2 Если выбран режим программирования AUTO– продолжение алгоритма. (PMODE = 0Ch).

3 Сравнение ключа защиты с данными во внешней памяти по адресам 0A020h – 0A02Fh.

4 Обнуление PACT# (низкий уровень P2.7).

5 Инициализация PVER (высокий уровень P2.0).

6 Чтение внешних данных, начинающихся с 0A000h.

7 Если слово не 0000h, выполняется программирование ячейки).

8 Если адрес внешней ячейки памяти равен 0FFFEh (соответствует 24 Кбайт записанной информации), то в указатель адреса загружается 0000h и выполняются последовательно действия по чтению из внешней памяти и программированию.

9 Когда адрес достигнет 0FFEh, то происходит сброс ИС и переход в режим нормальной работы.

24.9 Режим программирования SLAVE

В течение этого режима каждая внутренняя ячейка EEPROM доступна для записи и проверки по отдельности.

Существует подрежим, в котором адрес инкрементируется автоматически, и возможно чтение или запись в ячейки ПЗУ, расположенные последовательно, без необходимости передачи адреса по шине для каждого чтения или записи. Скорость процесса увеличивается, так как устраняется необходимость формировать и вводить каждый последующий адрес.

При входе в этот режим происходит проверка ключа защиты. Если ключ неверен, запись или считывание содержимого любой ячейки запрещены.

Примечание – В данном режиме программирования порты 3 и 4 функционируют в режиме с открытым стоком, поэтому требуется подключение подтягивающих резисторов к выводам этих портов. Не допускается программирование микроконтроллера в режиме SLAVE с помощью программатора для схем 1874BE76T, так как в них используется подача высокого напряжения (12,5 В) на выводы VPR и EA#. Это приведет к выходу из строя ИС 1874BE8T.

Режим SLAVE

Микроконтроллер в режиме SLAVE реагирует на команду Program Word или Dump Word. Порты (PBUS) служат для передачи адреса, данных и выборки команд. Некоторые из выводов порта 2 обеспечивают квитирование установления связи. Наименьший значащий бит порта 3 выбирает команду Program Word (P3.0 = 1) или Dump Word (P3.0 = 0). Старшие 15 битов содержат адрес программируемого или считываемого слова. 16 битов вместе представляют собой комбинацию адреса/команды. Адрес размещается с 2000h по 8FFFh. В таблице 24.8 описаны функции сигналов в режиме SLAVE.

Таблица 24.8 – Функции выводов в режиме программирования SLAVE

Имя	Тип	Описание
PALE# (P2.1)	Вход	Используется как разрешающий сигнал для чтения адресов и команд в устройство. PALE# обычно фиксируется на высоком уровне пользователем. PALE# разрешает чтение адресов и команд с третьего и четвертого портов
CPVER (P2.6)	Выход	Заключительная проверка программы. Когда работа закончена, высокий уровень показывает, что все ячейки записаны успешно, а низкий уровень указывает на ошибку, случившуюся во время одной из операций
AINC# (P2.4)	Вход	Автоматический инкремент. Низкий активный уровень разрешает режим автоматического инкремента. Он позволяет осуществлять чтение или запись последовательных ячеек без передачи каждый раз адреса по шине. В режиме Program Word: AINC# проверяется после записи в каждую ячейку. Если AINC# установлен, адрес инкрементируется и вводится следующее слово данных. В режиме Dump Word: AINC# проверяется после каждого слова, которое было записано в порты 3 и 4. Если AINC# установлен, адрес инкрементируется и устройство готово к выводу следующего слова данных
PBUS (порты 3 и 4)	Вход/ Выход	Шина адреса/управления/данных. Используется для чтения и записи команд, адресов и данных. Порты 3 и 4 используются в режиме ввода-вывода с открытым стоком (не как системная шина). Добавив внешние нагрузочные резисторы, можно считывать данные из устройства во время процедуры Dump Word
PROG# (P2.2)	Вход	Начало программирования. PROG# используется как разрешающий сигнал для чтения данных в течение процедуры Program Word. PROG# обычно фиксируется на высоком уровне пользователем. PROG# разрешает чтение данных из портов 3/4 и программирование внутреннего ПЗУ (EEPROM). Если PROG# остается установленным, в ячейку записываются те же данные в течение 25 импульсов по 500 мкс. По этой причине данные в портах должны оставаться неизменными, пока PROG# установлен. Когда PROG# станет высоким, работа продолжается. PROG# используется как разрешение установления связи для вывода данных через порт в процедуре Dump Word. PROG# устанавливается в высокий уровень пользователем. PROG# разрешает вывод содержимого ячейки ПЗУ (EEPROM) через порт. Когда PROG# станет высоким, работа продолжается
PVER (P2.0)	Выход	Проверка правильности программирования. Сигнал корректируется после каждого программирующего импульса. Высокий уровень указывает на успешное программирование, низкий – на обнаруживаемую ошибку

Проверка ключа защиты в режиме SLAVE

Алгоритм программирования в режиме SLAVE в начале работы проверяет ключ защиты. Для этого необходимо первыми командами задать программирование восьми последовательных слов с 2020h по 202Fh. Алгоритм сравнивает записываемые величины с содержимым ячеек 2020h – 202Fh внутреннего ПЗУ. Если проверка прошла, то предоставляется доступ к устройству, иначе оно зависает и должно быть возвращено

в исходное состояние через сброс. Устройство запрещает запись или вывод содержимого любой ячейки, если проверка не прошла.

Алгоритм режима SLAVE

Последовательность включения питания (подраздел 24.5) и таблица значений PMODE (таблица 24.1) содержат информацию, необходимую для входа в режим SLAVE. Алгоритм включает три функциональных блока: декодер адрес/команда, процедуры программирования слова (Program Word) и считывания слова (Dump Word).

На рисунке 24.6 показана программа дешифрации адреса/данных, на рисунке 24.7 – алгоритм Program Word, на рисунке 24.8 – алгоритм Dump Word.

Команды Program Word и Dump Word

В таблице 24.9 определены обозначения временных интервалов во временных диаграммах команд Program Word и Dump Word. На рисунках 24.9 и 24.10 приведены временные характеристики этих сигналов.

Таблица 24.9 – Обозначения временных характеристик

Обозначение	Описание
Tshll	Сброс высокого уровня до первого низкого PALE#
Tlllh	Длительность PALE# импульса
Tavll	Время установки адреса
Tllax	Время удержания адреса
Tpldv	Время от низкого PROG# до действительного слова вывода
Tphdx	Удержание данных слова вывода
Tdvpl	Время установки данных
Tpldx	Время удержания данных
Tplph	Длительность импульса PROG#
Tphll	Время от высокого PROG# до следующего низкого PALE#
Tlhpl	Время от высокого PALE# до низкого PROG#
Tphpl	Время от высокого PROG# до следующего низкого PROG#
Tphil	Время от высокого PROG# до низкого AINC#
Tilih	Длительность импульса AINC#
Tilvh	PVER удерживается после низкого AINC#
Tilpl	Время от низкого AINC# до низкого PROG#
Tphvl	Время от высокого PROG# до действительного PVER

Последовательность операций при дешифрации адреса/данных (рисунок 24.6)

1 Определение режима программирования (чтение PMODE).

2 Если выбран режим SLAVE (PMODE = 5h), продолжение алгоритма.

3 Проверка ключа безопасности.

4 Если PALE# активен, переход к следующему шагу, иначе – повторная операция.

5 Чтение данных с портов 3/4.

6 Если PVER установлен (ошибки нет), – переход к шагу 8. Если PVER не установлен – очистить CPVER, установить PVER, затем переход к шагу 7.

7 Ожидание сброса PALE#.

8 Проверка диапазона адресов портов 3 и 4.

9 Если P3.0 = 1, переход к процедуре Program Word (режим команд). Если P3.0 = 0, переход к процедуре Dump Word (режим адреса). Используется слово в портах 3 и 4 как адрес программирования или считывания.

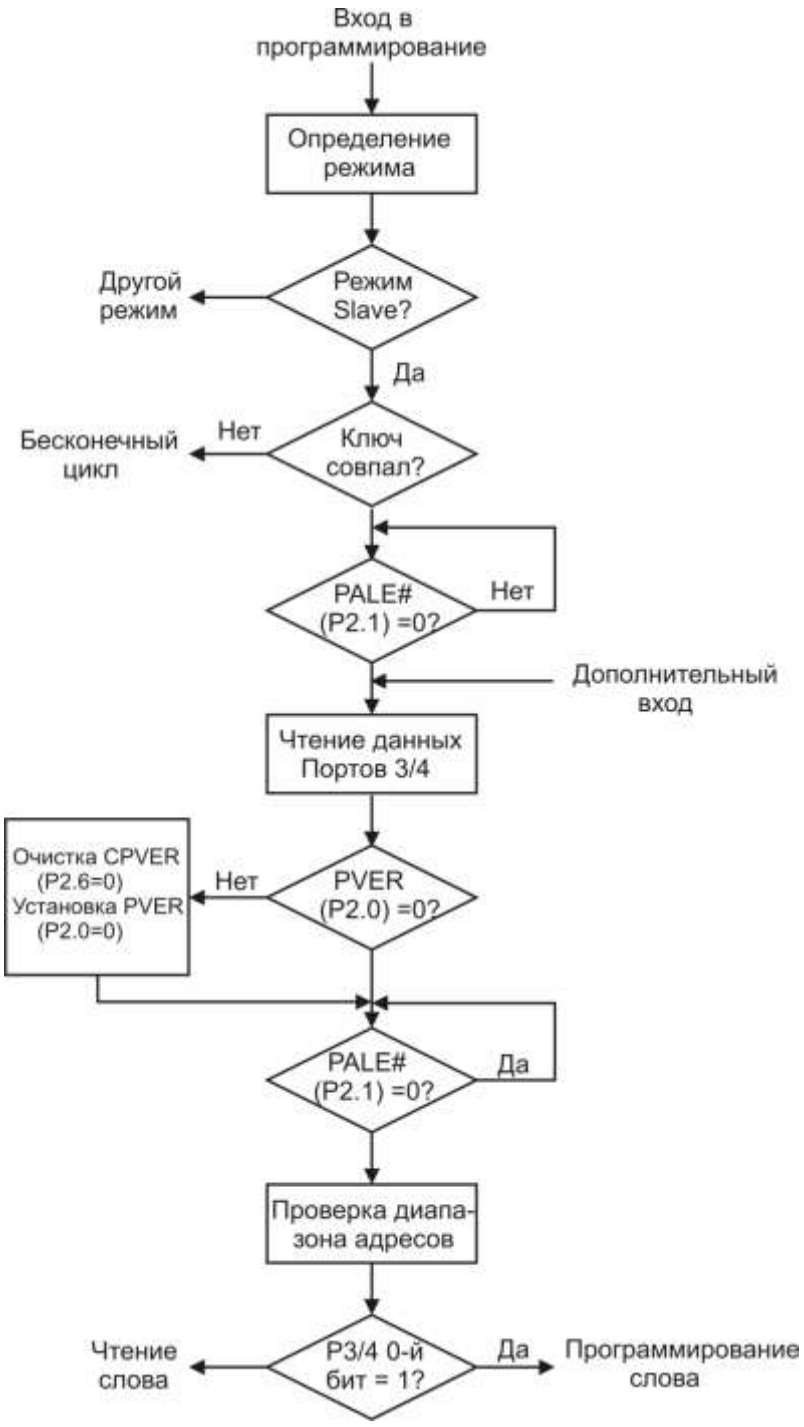


Рисунок 24.6 – Алгоритм дешифратора адреса/данных

Последовательность операций в процедуре Program Word (рисунок 24.7).

1 Ожидание установки PROG# в активный уровень.

2 При установлении PROG# – чтение данных из портов 3 и 4.

3 Вызов алгоритма программирования.

4 Если верификация пройдена, – установка PVER, иначе – очистка PVER.

5 Удержание PROG# в активном уровне, пока не завершится программирование.

6 Если PALE# активен, повторный вход в процедуру дешифрации адреса/команд и чтение следующего адреса/команды.

7 Если PALE# сброшен, – проверка AINC#.

8 Если AINC# активен, – адрес инкрементируется на два, затем верификация.

9 Если PVER установлен (нет ошибки), – повторный вход в процедуру Program Word.

10 Если PVER сброшен (ошибка верификации), – очистка CPVER, установка PVER, затем повторный вход в процедуру Program Word.

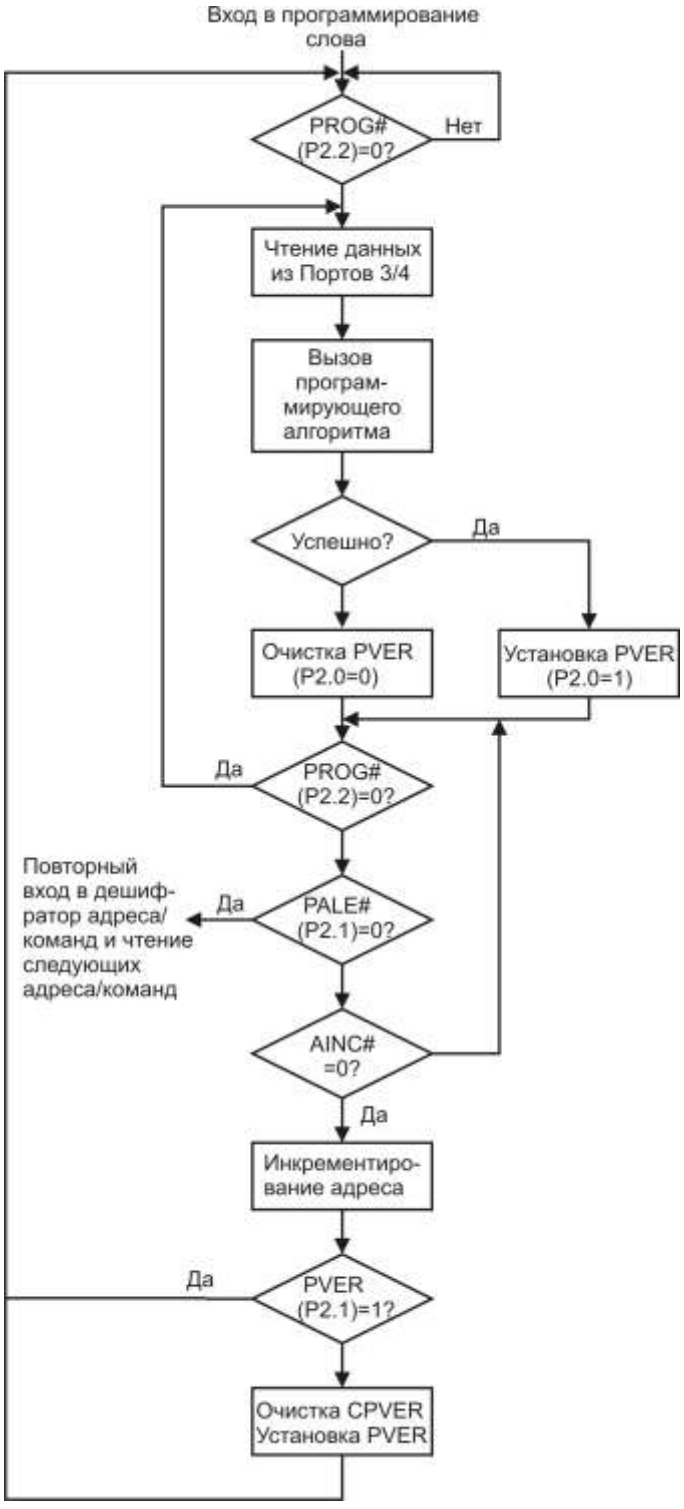


Рисунок 24.7 – Процедура Program Word

Рисунок 24.8 – Процедура Dump Word

Последовательность операций процедуры Dump Word (рисунок 24.8).

1 Чтение данных из EEPROM.

2 Ожидание установки PROG# в активный уровень.

3 При установлении PROG# – передача данных через порты 3 и 4.

4 Ожидание сброса PROG#.

5 Запись всех единиц в порты 3 и 4. (Это позволяет перевести выводы портов в третье состояние, избежав конфликта с процедурами Program Word и дешифрации адреса/команд, использующими порты 3 и 4 как входы.)

6 Если PALE# установлен, – возврат в процедуру дешифрации адреса/команд. Если PALE# сброшен – чтение AINC#.

7 Если AINC# не установлен, – возврат к шагу 6.

8 Если AINC# установлен, – адрес инкрементируется на два, затем повторный вход в эту процедуру.

На рисунке 24.9 показаны временные диаграммы команды Program Word с повторением импульсов программирования и автоинкрементом. Единица на P3.0 выбирает команду Program Word. Командой 3501h программируется ячейка слова, расположенная по адресу 3500h. Устройство получает входной сигнал PALE#, что показывает активность команды; установка PALE# фиксирует команду и адрес в портах 3 и 4 (PBUS). Установка PROG# фиксирует данные в портах 3 и 4 для начала программирования; устройство считает или выводит слово. Длительность импульса PROG# определяет длительность программирующего импульса. По нарастающему фронту сигнала PROG# идет автоматическая проверка содержимого только что

запрограммированной ячейки. Установка сигнала PVER# показывает успешное программирование. AINC# служит для автоматического инкремента адреса.

На рисунке 24.10 представлена команда Dump Word. Ноль на P3.0 выбирает команду Dump Word. Посылка команды «2100h» помещает слово из адреса внутренней памяти 2100h в порты 3 и 4. PROG# управляет подключением устройства к шине. В режиме Dump Word сигнал на выводе AINC# может оставаться активным или переключаться из одного состояния в другое. Вывод PROG# автоматически инкрементирует адрес.

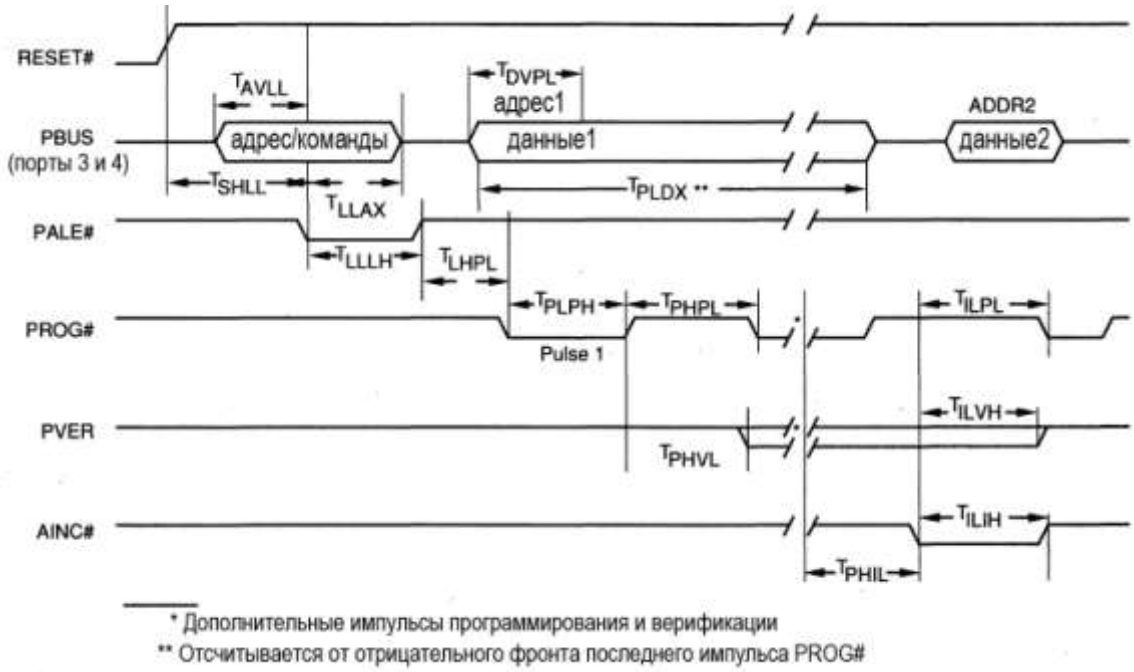


Рисунок 24.9 – Временные диаграммы в режиме Program Word

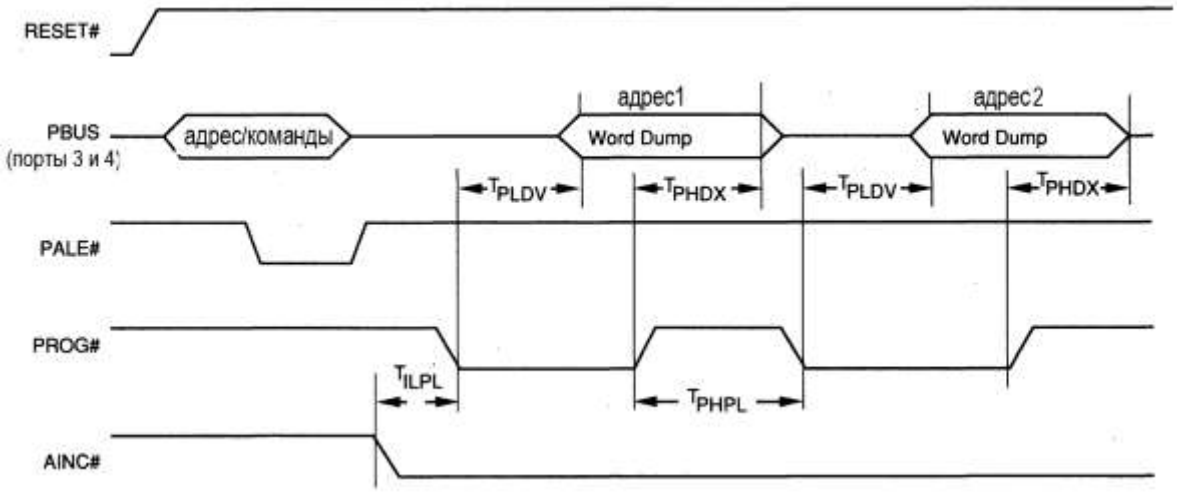


Рисунок 24.10 – Временные диаграммы режима Dump Word

24.10 Режим вывода содержимого памяти (ROM-DUMP)

Этот режим легко позволяет проверить содержимое EEPROM. Последовательность включения питания и таблица значений PMODE (таблица 24.1) помогут войти в режим ROM-DUMP (таблица 24.10).

После проверки ключа защиты режим записывает всю область внутренней памяти во внешнюю память. Ключ защиты во внутренней памяти должен соответствовать

при проверке ключу защиты в соответствующей области внешней памяти. Если ключ не совпадает, устройство входит в бесконечный цикл и может быть выведено из него только сбросом.

Таблица 24.10 – Карта памяти режима ROM-DUMP

Устройство	Адреса внутреннего EEPROM	Адреса внешней памяти
МК	2000h – 8FFFh	2000h – 8FFFh
МК Ключ защиты	2020h – 202Fh	2020h – 202Fh

24.11 Режим программирования через последовательный порт SERIAL

Данные в этом режиме передаются и принимаются через выводы TXD0 (P2.0) и RXD0 (P2.1). Эти выводы следует подключить к любому смарт-терминалу, способному работать в последовательном режиме. Любой хост, использующий интерфейс RS-232C (например, ПК), должен подключаться через специальный RS-232C приемопередатчик (драйвер).

В режиме программирования последовательного порта используются специальные RISM-команды, которые позволяют считывать и записывать внутреннюю память микроконтроллера. В следующих разделах описаны RISM-команды и представлены примеры их использования.

Выводы XTAL1 и XTAL2 подключаются к кварцевому резонатору с частотой от 10 до 40 МГц. Так, например, последовательная скорость двоичной передачи по умолчанию для частоты кварца 33 МГц составляет 9 600 бод.

Можно изменить скорость последовательной передачи, записав соответствующее значение в регистр BOUDE_RATE0.

24.12 Ввод микроконтроллера в режим программирования через последовательный порт SERIAL

Во время работы отладочной программы необходимо выбрать режим программирования, управляя сигналами PMODE (таблица 24.1). Чтобы войти в режим программирования через последовательный порт, значение PMODE должно быть 0Eh.

После входа в режим SERIAL микроконтроллер проверяет бит блокировки CCB0.7. Если CCB0.7 = 1, то происходит блокировка и микроконтроллер входит в бесконечный цикл. Если CCB0.7 = 0, то программирование продолжается. Ключи защиты, определенные в 2020h – 202Fh, должны быть не запрограммированы в этом режиме.

24.13 RISM-команды

После входа в режим программирования через последовательный порт микроконтроллер запускает отладочную подпрограмму TROM. Эта подпрограмма калибрует последовательный порт и его сигналы. Последовательный порт настраивается, чтобы работать в режиме 1 со скоростью двоичной передачи 9 600 бод на 33 МГц. Можно изменить эти параметры настройки по умолчанию после входа в режим SERIAL, записав значение скорости двоичной передачи в соответствующий регистр микроконтроллера.

Можно соединить микроконтроллер с любым смарт-терминалом через выводы RXD0 и TXD0. Команды, адреса и данные микроконтроллеру следует передавать через его вывод RXD0. Микроконтроллер передает данные, сдвигая их на выводе TXD0. Передавать информацию микроконтроллеру следует, используя 10-битовые фреймы (стартовый бит, 8 бит данных, начиная с LSB, стоповый бит). Аналогично микроконтроллер будет передавать информацию, сдвигая 10-битовые фреймы.

Выполнение передачи команд, соответствующих битам (00h – 014h). В каждый момент времени передается один байт данных или адреса, поэтому первым должен идти старший байт. Необходимо указывать полный адрес (16 бит). Так как микроконтроллер воспринимает данные в диапазоне 00h – 014h как команды, перед передачей байта данных

или адреса меньше 015h необходимо отправить команду SET_DLE_FLAG, которая укажет микроконтроллеру, что следующий байт не является командой. Микроконтроллер также отвечает на каждый полученный байт тем же самым байтом, т. е. тем самым свидетельствует об успешном приеме данных или успешном выполнении команды. Исключение составляет команда TRANSMIT, после этой команды микроконтроллер отправляет байт данных, считанных из памяти.

В таблице 24.11 перечислены RISM-команды микроконтроллера. В следующих разделах объясняется, как использовать эти RISM-команды, чтобы осуществлять чтение и запись слов (байт) внутренней памяти.

Таблица 24.11– RISM-команды последовательного порта

Байты команд	Команды	Описание
00h	SET_DLE_FLAG	Команда передается, когда байт адреса или данных меньше 015h. Этот код говорит микроконтроллеру, что следующий байт, который он получит, не будет командой. Микроконтроллер воспринимает значения в диапазоне 00h – 14h как команды, если им не предшествует команда SET_DLE_FLAG
02h	TRANSMIT	Команда подается сразу после передачи команд READ_BYTE или READ_WORD. Эта команда заставляет микроконтроллер начать передачу данных с его вывода TXD. Эту команду следует отправлять один раз после команды READ_BYTE и дважды после READ_WORD
04h	READ_BYTE	Команда чтения байта
05h	READ_WORD	Команда чтения слова
07h	WRITE_BYTE	Команда записи байта
08h	WRITE_WORD	Команда записи слова
0Ah	DATA_TO_ADDR	Команда подается сразу после передачи адреса
0Ch	ERASE	Команда очистки ПЗУ
0Fh	RESETSP	Команда сброса текущей последовательности команд

24.14 Чтение из внутренней памяти или RAM

Пять RISM-команд управляют чтением внутренней памяти:
- SET_DLE_FLAG (00h);
- DATA_TO_ADDR (0Ah);
- READ_BYTE (04h);
- READ_WORD (05h);
- TRANSMIT (02h).

На рисунке 24.11 представлен алгоритм для чтения слов из внутренней памяти, а также указано, какие изменения необходимо сделать для чтения из внутренней памяти отдельных байт.

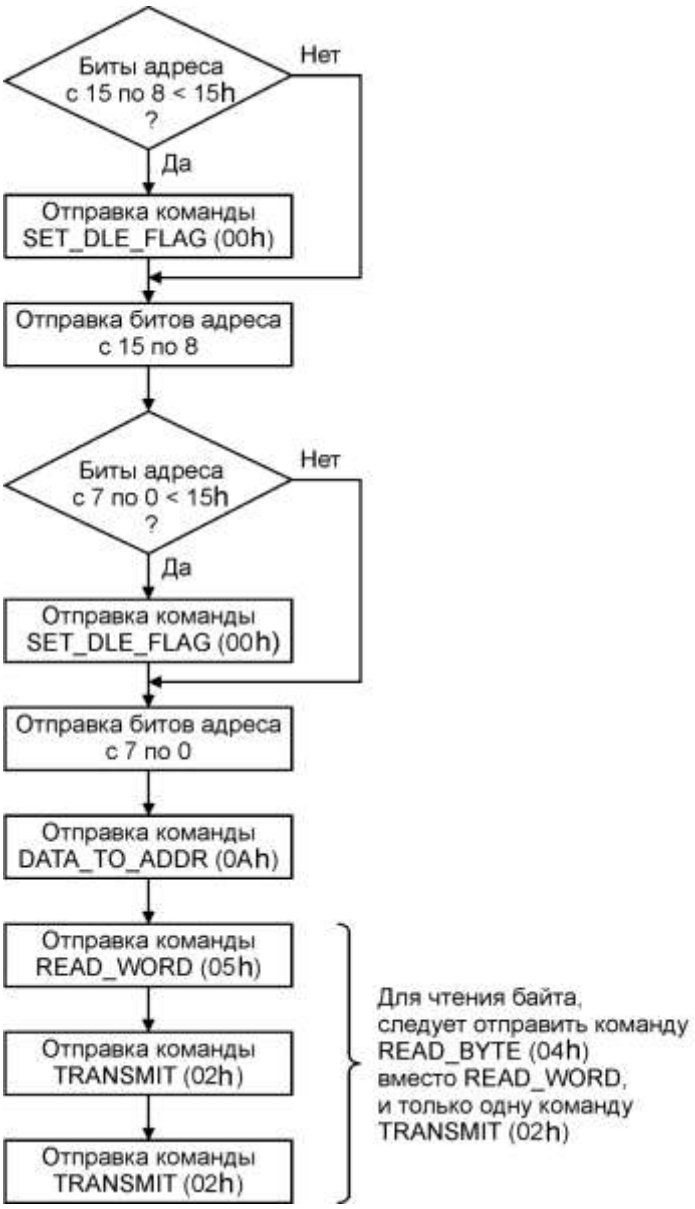


Рисунок 24.11 – Алгоритм чтения внутренней памяти микроконтроллера в режиме программирования через последовательный порт

24.15 Запись RAM

Четыре RISM-команды связаны с записью внутренней RAM:
- SET_DLE_FLAG (00h);
- DATA_TO_ADDR (0Ah);
- WRITE_BYTE (07h);
- WRITE_WORD (08h).

На рисунке 24.12 показан алгоритм для записи слов во внутреннюю память.

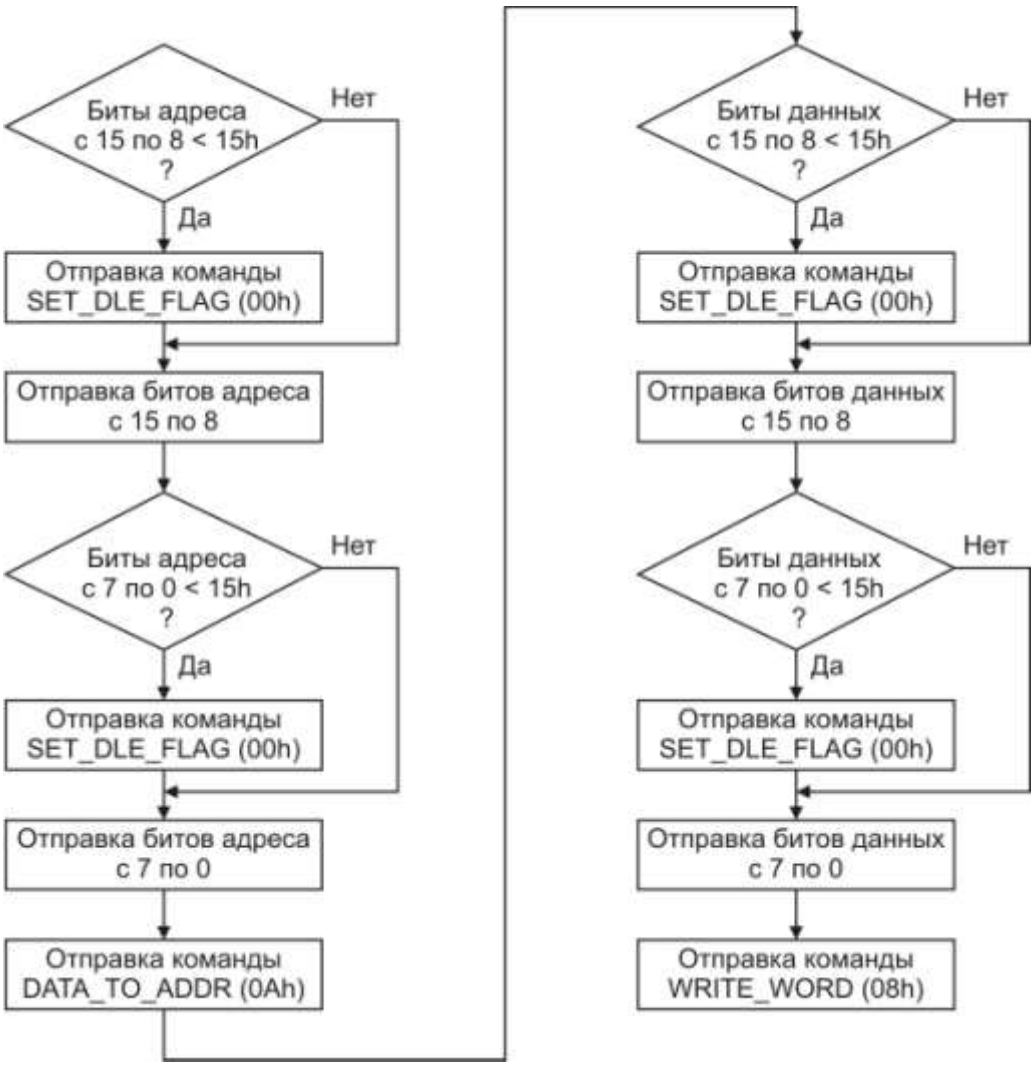


Рисунок 24.12 – Алгоритм записи во внутреннюю память микроконтроллера в режиме программирования через последовательный порт

Заключение

В настоящем руководстве КФДЛ.431295.048 приведено подробное описание архитектуры, функционального построения, системы команд и особенностей применения ИС 1874ВЕ8Т, которая представляет собой СБИС однокристального 16-разрядного микроконтроллера с тактовой частотой до 40 МГц, ОЗУ 2Кбайта, расширенным ОЗУ 2 Кбайта, восемью 16-разрядными АЦП с блоком цифровых компараторов, 14-разрядным ЦАП, 3-канальным ШИМ, двумя портами ввода-вывода USART, контроллерами интерфейсов CAN, LIN, SPI, I2C и внутренней памятью программ (типа EEPROM) объемом 16К×16. Микроконтроллер предназначен для применения в цифровых системах управления, для управления робототехническими комплексами, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, телекоммуникационной и другой технике.

Значения параметров, приведенные в настоящем руководстве, являются справочными.

Руководство КФДЛ.431295.048 может служить практическим пособием по применению микроконтроллеров для разработчиков систем на основе микроконтроллеров 1874ВЕ8Т и программистов.

Приложение А
(обязательное)
Список регистров микроконтроллера

Таблица А.1 представляет собой карту памяти микроконтроллера с адреса 1A00h по 1FFEh .

Таблица А.1– Карта памяти в диапазоне адресов 1A00h – 1FFEh

Область адресов	Блок, регистры которого расположены в указанной области
1A00h – 1EFEh	Контроллер CAN
1F00h – 1F2Eh	–
1F30h – 1F3Eh	Блок кодирования по ГОСТ 28147–89
1F40h – 1F5Eh	Контроллер LIN
1F60h – 1F6Eh	Генератор ПСП
1F70h – 1F8Eh	АЦП
1F90h – 1F9Eh	–
1FA0h – 1FA4h	Контроллер SPI
1FA6h – 1FADh	Блоки USART0 и USART1
1FAEh	–
1FB0h – 1FBEh	Контроллер I2C
1FC0h – 1FCFh	АЦП
1FD0h – 1FDEh	–
1FE0h – 1FEFh	Система отладки
1FF0h – 1FF6h	Управление синхронизацией и памятью
1FF8 – 1FFCh	–
1FFEh	Порты 3 и 4

В таблице А.2 представлен список регистров микроконтроллера, расположенных в алфавитном порядке их мнемонических обозначений. Далее в таблицах А.3 – А.126 представлены форматы регистров и информация о функциональном назначении их бит.

Таблица А.2– Регистры микроконтроллера

Мнемоническое обозначение	Название регистра	Адрес	Состояние после сброса
1	2	3	4
ADCCHCH	Регистр H управления АЦП	1FC4h	0000h
ADCCHCL	Регистр L управления АЦП	1FC2h	0000h
ADCIEN	Регистр управления прерываниями каналов АЦП	1F7Ch	0202h
ADCOM0B	Регистр нижней границы 0	1F70h	0000h
ADCOM1B	Регистр нижней границы 1	1F72h	0000h
ADCOM2B	Регистр нижней границы 2	1F74h	0000h
ADCOM3B	Регистр нижней границы 3	1F76h	0000h
ADCOM4B	Регистр нижней границы 4	1F78h	0000h
ADCOM5B	Регистр нижней границы 5	1F7Ah	0000h
ADCOM0T	Регистр верхней границы 0	1F80h	0000h
ADCOM1T	Регистр верхней границы 1	1F82h	0000h
ADCOM2T	Регистр верхней границы 2	1F84h	0000h
ADCOM3T	Регистр верхней границы 3	1F86h	0000h

Продолжение таблицы А.2

1	2	3	4
ADCOM4T	Регистр верхней границы 4	1F88h	0000h
ADCOM5T	Регистр верхней границы 5	1F8Ah	0000h
ADCOM6	Регистр границы 6	1F8Ch	0000h
ADCOM7	Регистр границы 7	1F8Eh	0000h
ADCOMC	Регистр управления цифровыми компараторами	1F7Eh	0000h
ADCR	Регистр управления запуском преобразования АЦП	1FC0h	3206h
ADCRES0	Регистр результата канала 0	1FD0h	0000h
ADCRES1	Регистр результата канала 1	1FD2h	0000h
ADCRES2	Регистр результата канала 2	1FD4h	0000h
ADCRES3	Регистр результата канала 3	1FD6h	0000h
ADCRES4	Регистр результата канала 4	1FD8h	0000h
ADCRES5	Регистр результата канала 5	1FDAh	0000h
ADCRES6	Регистр результата канала 6	1FDCh	0000h
ADCRES7	Регистр результата канала 7	1FDEh	0000h
ADCRST	Регистр управления сбросом каналов АЦП	1FC6h	00FFh
BAUD_RATE0	Регистр скорости передачи порта USART0	0Eh (0/-)	00h
BAUD_RATE1	Регистр скорости передачи порта USART1	1FA6h	00h
CCR	Регистр конфигурации кристалла	–	1Fh
CDAUTH	Регистр имитовставки блока кодирования	1F3Eh	00h
CDCON	Регистр управления блока кодирования	1F32h	00h
CDDATA	Регистр данных блока кодирования	1F30h	00h
CDDATNUM	Регистр числа блоков данных блока кодирования	1F3Ah	00h
CDKEY	Регистр ключа блока кодирования	1F34h	00h
CDSBSTN	Регистр замены блока кодирования	1F36h	00h
CDSTATE	Регистр состояния блока кодирования	1F3Ch	00h
CDUW	Регистр синхропосылки блока кодирования	1F38h	00h
CLC	Регистр управления частотой CAN	1A00h	00000003h
CLKC	Регистр управления тактированием	1FF0h	FF3Fh
CLKCDEB	Регистр загрузки CLKC при отладке	1FF6h	0000h
CURRPC	Регистр текущего значения счетчика команд	1FE8h	XXXXh
DACVAL	Регистр значения тока ЦАП	10h	C000h
DEBADDR	Регистр ожидаемого значения адреса	1FE4h	0000h
DEBCTRL (старший байт)	Регистр управления модулем отладки	1FE7h	00h
DEBCTRL (младший байт)	Регистр управления модулем отладки	1FE6h	00h
DEBDATA	Регистр ожидаемого значения данных	1FE2h	0000h
DEBDATH	Регистр ожидаемого значения старшего байта данных	1FEFh	00h
DEBDATL	Регистр ожидаемого значения младшего байта данных	1FEEh	00h
DEBPC	Регистр ожидаемого значения счетчика команд	1FE0h	0000h
DEBPCEQ	Регистр ожидаемого значения счетчика команд	1FECh	0000h
FDR	Регистр делителя контроллера CAN	1A0Ch	0000 0000h
HAPPPC	Регистр значения счетчика команд	1FEAh	XXXXh
HSI_MODE	Регистр режима HSI	03h(0/15)	FFh

Продолжение таблицы А.2

1	2	3	4
HSI_STATUS	Регистр состояния HSI	06h (15/0)	X0X0 X0X0b
HSI_TIME	Регистр времени HSI	04h (15/0)	XXXXh
HSO_COMMAND	Регистр управления HSO	06h (0/15)	XXh
HSO_TIME	Регистр времени HSO	04h (0/15)	XXXXh
ID	Регистр идентификации	1A08h	002B C051h
INT_MASK	Регистр маски прерываний	08h	00h
INT_MASK1	Регистр маски прерываний 1	13h	00h
INT_PEND	Регистр ждущих прерываний	09h	00h
INT_PEND1	Регистр ждущих прерываний 1	12h	00h
IOC0	Регистр 0 управления вводом/выводом	15h (0/15)	000000X0b
IOC1	Регистр 1 управления вводом/выводом	16h (0/15)	21h
IOC2	Регистр 2 управления вводом/выводом	0Bh (0/15)	X0000000b
IOC3	Регистр 3 управления вводом/выводом	0Ch (1)	F2h
IOPORT0	Регистр ввода/вывода порта 0	0Eh (-/0) или 1FF8h	XXh
IOPORT1	Регистр ввода/вывода порта 1	0Fh (0)	FFh
IOPORT2	Регистр ввода/вывода порта 2	10h (0)	C1h
IOPORT34	Регистр ввода/вывода портов 3 и 4	1FFEh	FFFFh
IOS0	Регистр 0 состояния ввода/вывода	15h (15/0)	00h
IOS1	Регистр 1 состояния ввода/вывода	16h (15/0)	00h
IOS2	Регистр 2 состояния ввода/вывода	17h (15/0)	00h
LINBREAK	Регистр настройки длины поля BREAK	1F54h	01E7h
LINCON	Регистр управления LIN	1F50h	0801h
LINCSR	Регистр контрольной суммы LIN	1F5Ch	XX00h
LINDRX10	Регистр 0 и 1 байтов принятых данных	1F48h	0000h
LINDRX32	Регистр 2 и 3 байтов принятых данных	1F4Ah	0000h
LINDRX54	Регистр 4 и 5 байтов принятых данных	1F4Ch	0000h
LINDRX76	Регистр 6 и 7 байтов принятых данных	1F4Eh	0000h
LINDTX10	Регистр 0 и 1 байтов данных для передачи	1F40h	0000h
LINDTX32	Регистр 2 и 3 байтов данных для передачи	1F42h	0000h
LINDTX54	Регистр 4 и 5 байтов данных для передачи	1F44h	0000h
LINDTX76	Регистр 6 и 7 байтов данных для передачи	1F46h	0000h
LININTEN	Регистр разрешения прерывания LIN	1F5Eh	0000h
LINPIDTX	Регистр передаваемого PID	1F58h	XX00h
LINPIDRX	Регистр принятого PID	1F5Ah	XX00h
LINSPEED	Регистр настройки скорости передачи LIN	1F56h	012Ch
LINSTAT	Регистр состояния LIN	1F52h	0000h
LIST0	Регистр списка №0 контроллера CAN	1B00h	007F 7F00h
LIST1	Регистр списка №1 узла CAN0	1B04h	0100 0000h
LIST2	Регистр списка №2 узла CAN1	1B08h	0100 0000h
LIST3	Регистр свободного списка №3	1B0Ch	0100 0000h
LIST4	Регистр свободного списка №4	1B10h	0100 0000h
LIST5	Регистр свободного списка №5	1B14h	0100 0000h
LIST6	Регистр свободного списка №6	1B18h	0100 0000h
LIST7	Регистр свободного списка №7	1B1Ch	0100 0000h
MCR	Регистр управления	1BC8h	0000 0000h
MITR	Регистр прерываний	1BCCh	0000 0000h
MOAMR0	Регистр маски объекта сообщения 0	1C8Ch	3FFF FFFFh

Продолжение таблицы А.2

1	2	3	4
MOAR0	Регистр арбитража объекта сообщения 0	1C98h	0000 0000h
MOCTR0	Регистр управления объектом сообщения 0	1C9Ch	0100 0000h
MODATAH0	Регистр данных (старшие четыре байта) объекта сообщения 0	1C94h	0000 0000h
MODATAL0	Регистр данных (младшие четыре байта) объекта сообщения 0	1C90h	0000 0000h
MOFCR0	Регистр управления функционированием объекта сообщения 0	1C80h	0000 0000h
MOFGPR0	Регистр указателя FIFO/шлюза объекта сообщения 0	1C84h	0000 0000h
MOIPR0	Регистр указателя прерываний объекта сообщения 0	1C88h	0000 0000h
MOSTAT0	Регистр состояния объекта сообщения 0	1C9Ch	0100 0000h
Регистры объектов сообщений с 1 по 14 (адреса и состояния после сброса см. в таблицах А.3 и А.4).			
MOAMR15	Регистр маски объекта сообщения 0	1E6Ch	3FFF FFFFh
MOAR15	Регистр арбитража объекта сообщения 0	1E78h	0000 0000h
MOCTR15	Регистр управления объектом сообщения 0	1E7Ch	0F0E 0000h
MODATAH15	Регистр данных (старшие четыре байта) объекта сообщения 0	1E74h	0000 0000h
MODATAL15	Регистр данных (младшие четыре байта) объекта сообщения 0	1E70h	0000 0000h
MOFCR15	Регистр управления функционированием объекта сообщения 0	1E60h	0000 0000h
MOFGPR15	Регистр указателя FIFO/шлюза объекта сообщения 0	1E64h	0000 0000h
MOIPR15	Регистр указателя прерываний объекта сообщения 0	1E68h	0000 0000h
MOSTAT15	Регистр состояния объекта сообщения 0	1E7Ch	0F0E 0000h
MSID0	Регистр индекса сообщения	1B80h	0000 0020h
MSIMASK	Регистр маски индекса сообщения	1BC0h	0000 0000h
MSPND0	Регистр ждущих прерываний	1B40h	0000 0000h
NBTR0	Регистр синхронизации битов узла CAN0	1C10h	0000 0000h
NBTR1	Регистр синхронизации битов узла CAN1	1C30h	0000 0000h
NCR0	Регистр управления узла CAN0	1C00h	0000 0001h
NCR1	Регистр управления узла CAN1	1C20h	0000 0001h
NECNT0	Регистр счетчика ошибок узла CAN0	1C14h	0060 0000h
NECNT1	Регистр счетчика ошибок узла CAN1	1C34h	0060 0000h
NFCR0	Регистр счетчика сообщений узла CAN0	1C18h	0000 0000h
NFCR1	Регистр счетчика сообщений узла CAN1	1C38h	0000 0000h
NIPR0	Регистр указателя прерываний узла CAN0	1C08h	0000 0000h
NIPR1	Регистр указателя прерываний узла CAN1	1C28h	0000 0000h
NPCR0	Регистр управления портом узла CAN0	1C0Ch	0000 0000h
NPCR1	Регистр управления портом узла CAN1	1C2Ch	0000 0000h
NSR0	Регистр состояния узла CAN0	1C04h	0000 0000h
NSR1	Регистр состояния узла CAN1	1C24h	0000 0000h
PANCTR	Регистр панели команд	1BC4h	0000 0301h
PPHOLD	Регистр коэффициента 3	1FF5h	–
PPSETUP	Регистр коэффициента 2	1FF4h	–

Окончание таблицы А.2

1	2	3	4
PPW	Регистр коэффициента 1	1FF2h	–
PSW	Слово состояния программы	–	–
PTSBLOCK	Регистр числа передач PTS	–	XXh
PTSCON	Регистр управления PTS	–	00h/02h
PTSCOUNT	Регистр количества циклов PTS	–	XXh
PTSDST	Регистр приемника PTS	–	XXXXh
PTSSRC	Регистр источника PTS	–	XXXXh
PTSSEL	Регистр выбора PTS	04h (1)	0000h
PTSSRV	Регистр обслуживания PTS	06h (1)	0000h
PRNGCON	Регистр управления генератора ПСП	1F64h	00h
PRNGREG1	Регистр ключа 0 генератора ПСП	1F60h	0000h
PRNGREG2	Регистр ключа 1 генератора ПСП	1F62h	0000h
PRNGRES	Регистр результата генератора ПСП	1F66h	0000h
PWM0_CONTROL	Регистр длительности импульса канала 0	17h (0/15)	00h
PWM1_CONTROL	Регистр длительности импульса канала 1	16h (1)	00h
PWM2_CONTROL	Регистр длительности импульса канала 2	17h (1)	00h
ROMC	Регистр управления программированием EEPROM	1FFCh	0000h
SBUF_RX0	Регистр принятых данных USART0	07h (15/0)	00h
SBUF_TX0	Регистр передаваемых данных USART0	07h (0/15)	00h
SBUF_RX1	Регистр принятых данных USART1	1FABh	00h
SBUF_TX1	Регистр передаваемых данных USART1	1FA9h	00h
SMBADDR	Регистр собственного адреса I2C	1FB8h	00h
SMBCST	Регистр управления и состояния I2C	1FB4h	00h
SMBCTRL1	Регистр 1 управления I2C	1FB6h	00h
SMBCTRL2	Регистр 2 управления I2C	1FBAh	00h
SMBCTRL3	Регистр 3 управления I2C	1FBEh	00h
SMBSDA	Сдвиговый регистр данных I2C	1FB0h	XXh
SMBST	Регистр состояния I2C	1FB2h	00h
SMBTOPR	Регистр загрузки предделителя	1FBCh	00h
SP	Регистр указателя стека	18h	1D18h
SPCR	Регистр управления SPI	1FA0h	04h
SPDR	Регистр данных SPI	1FA4h	00h
SPSR	Регистр состояния SPI	1FA2h	00h
SP_CON0	Регистр управления USART0	11h (0/15)	0Bh
SP_CON1	Регистр управления USART1	1FAFh	0Bh
SP_STAT0	Регистр состояния USART0	11h (15/0)	0Bh
SP_STAT1	Регистр состояния USART1	1FADh	0Bh
T2CAPTURE	Регистр захвата таймера 2	0Ch	XXXXh
TIMER1	Регистр таймера 1	0Ah (15/0)	0000h
TIMER2	Регистр таймера 2	0Ch (0)	0000h
WATCHDOG	Регистр сторожевого таймера	0Ah (0/15)	XXh
WSR	Регистр выбора окна	14h	X0h
ZERO_REG	Регистр нуля	00h	0000h

Примечание – Для регистров, адреса которых лежат в диапазоне от 00h до 17h, в скобках после адреса указаны номера горизонтальных окон, посредством которых осуществляется запись/чтение. Если цифра одна, значит, запись/чтение производится через одно окно. Отсутствие скобок указывает на то, что регистр доступен для записи/чтения по указанному адресу вне зависимости от того, какое окно активно.

Таблица А.3 – Адреса регистров объектов сообщений контроллера CAN

Объект со-общения n	Регистры объекта сообщения							
	MO FCRn	MO FGPRn	MO IPRn	MO AMRn	MO DATALn	MO DATAHn	MO ARn	MOSTATn MOCTRn
0	1C80h	1C84h	1C88h	1C8Ch	1C90h	1C94h	1C98h	1C9Ch
1	1CA0h	1CA4h	1CA8h	1CACh	1CB0h	1CB4h	1CB8h	1CBCh
2	1CC0h	1CC4h	1CC8h	1CCCh	1CD0h	1CD4h	1CD8h	1CDCh
3	1CE0h	1CE4h	1CE8h	1CECh	1CF0h	1CF4h	1CF8h	1CFCh
4	1D00h	1D04h	1D08h	1D0Ch	1D10h	1D14h	1D18h	1D1Ch
5	1D20h	1D24h	1D28h	1D2Ch	1D30h	1D34h	1D38h	1D3Ch
6	1D40h	1D44h	1D48h	1D4Ch	1D50h	1D54h	1D58h	1D5Ch
7	1D60h	1D64h	1D68h	1D6Ch	1D70h	1D74h	1D78h	1D7Ch
8	1D80h	1D84h	1D88h	1D8Ch	1D90h	1D94h	1D98h	1D9Ch
9	1DA0h	1DA4h	1DA8h	1DACh	1DB0h	1DB4h	1DB8h	1DBCh
10	1DC0h	1DC4h	1DC8h	1DCCh	1DD0h	1DD4h	1DD8h	1DDCh
11	1DE0h	1DE4h	1DE8h	1DECh	1DF0h	1DF4h	1DF8h	1DFCh
12	1E00h	1E04h	1E08h	1E0Ch	1E10h	1E14h	1E18h	1E1Ch
13	1E20h	1E24h	1E28h	1E2Ch	1E30h	1E34h	1E38h	1E3Ch
14	1E40h	1E44h	1E48h	1E4Ch	1E50h	1E54h	1E58h	1E5Ch
15	1E60h	1E64h	1E68h	1E6Ch	1E70h	1E74h	1E78h	1E7Ch

Таблица А.4 – Состояние регистров объектов сообщений после сброса

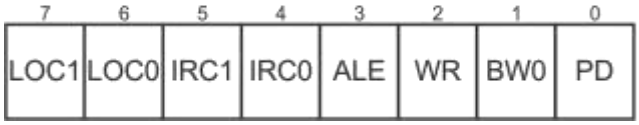
n (код)	MOCTRn / MOSTATn			MOAMRn
	PNEXT (биты 31–24)	PPREV (биты 23–16)	Биты 15–0	
0 (00h)	01h	00h	0000h	3FFF FFFFh
1 (01h)	02h	00h		
2 (02h)	03h	01h		
3 (03h)	04h	02h		
…	…	…		
13 (0Dh)	0Eh	0Ch		
14 (0Eh)	0Fh	0Dh		
15 (0Fh)	0Fh	0Eh		
Примечание – Регистры MOARn, MODATAHn, MODATALn, MOIPRn, MOFGPRn и MOFCRn после сброса микроконтроллера всегда инициализируются в 0000 0000h.				

Регистры ЦПУ

Таблица А.5 – Регистр конфигурации кристалла

CCR

значение после Сброса: 1Fh



Поле	Бит	Тип	Описание	
LOC1 LOC0	7, 6	Однократная аппаратная загрузка при сбросе микросхемы	Биты блокировки. Задают программную защиту внутренней памяти.	
			00	Нет защиты
			01	Защита от записи
			10	Защита от чтения
			11	Защита от чтения и записи
IRC1 IRC0	5,4		Биты управления готовностью внешних устройств. Определяют число циклов ожидания, которые могут быть введены, пока на выводе READY удерживается низкий уровень сигнала. Длительность одного цикла ожидания составляет один такт сигнала CLOCKOUT	
			00	1 цикл ожидания
			01	2 цикла ожидания
			10	3 цикла ожидания
			11	Вставка циклов ожидания до тех пор, пока сигнал READY в низком состоянии
ALE, WR	3, 2		Биты выбора режима. Определяют, какие сигналы будут генерироваться на шине управления в циклах записи и чтения внешней памяти (подробнее см. таблицу А.6)	
BW0	1		Бит управления разрядностью шины	
			0	8-разрядная шина; состояние вывода BW не важно
			1	8-разрядная шина, при BW = 0 16-разрядная шина, при BW = 1
PD	0		Бит разрешения включения режима PowerDown	
			0	Режим запрещен
			1	Режим разрешен

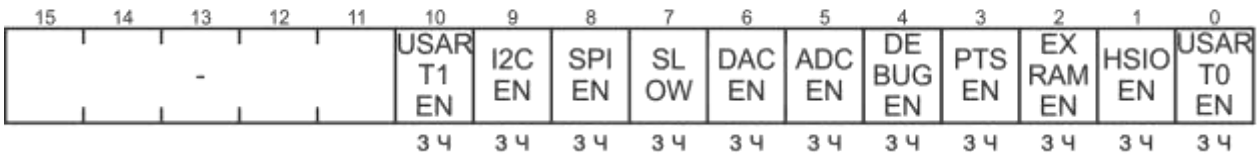
Таблица А.6 – Варианты конфигурации сигналов шины управления

Биты регистра CCR		Режим шины управления	Активный сигнал вывода на шине управления			
ALE	WR					
0	0	Строба записи и достоверного адреса	ADV#	RD#	WRH#	WRL#
0	1	Строба достоверного адреса	ADV#	RD#	BHE#	WR#
1	0	Строба записи	ALE	RD#	WRH#	WRL#
1	1	Стандартный	ALE	RD#	BHE#	WR#

Таблица А.7 – Регистр управления тактированием

CLKC

1FF0h значение после Сброса: FF3Fh



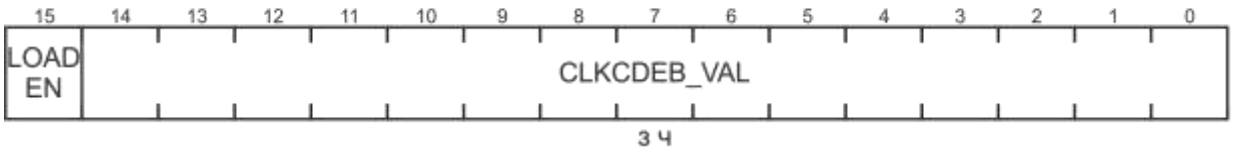
Поле	Бит	Описание		Время включения, мт	Время выключения, мт
USART1EN	10	Бит разрешения тактирования USART1 и LIN		1	1
I2CEN	9	Бит разрешения тактирования I2C		1	1
SPIEN	8	Бит разрешения тактирования SPI		1	1
SLOW	7	Бит включения режима медленного тактирования SLOW		1	1
		0	Выключен (Fclk = Fosc/2)		
		1	Включен (Fclk = Fosc/4)		
DACEN	6	Бит разрешения тактирования ЦАП		350 (при 40 МГц)	1
		0	Запрещено. При отключении тактового сигнала аналоговая часть переводится в режим SLEEP		
		1	Разрешено		
ADCEN	5	Бит разрешения тактирования АЦП		1 800 с момента первой команды на преобразование	1
		0	Запрещено. При отключении тактового сигнала аналоговая часть переводится в режим SLEEP		
		1	Разрешено. Сразу после установки бита может быть подана команда на преобразование		
DEBUGEN	4	Бит разрешения тактирования модуля отладки		1	1
PTSEN	3	Бит разрешения тактирования PTS		1	1
EXRAMEN	2	Бит разрешения работы расширенного ОЗУ (EXRAM)		1	1
		0	Запрещено. При выключении адреса EXRAM назначаются во внешнюю память		
		1	Разрешено		
HSIOEN	1	Бит разрешения тактирования HSIO		1	1
USART0EN	0	Бит разрешения тактирования USART0		1	1
–	15–11	Зарезервировано			

Примечание – Для бит 10 – 8, 4, 3, 1 и 0 действует правило: установленный бит разрешает синхронизацию соответствующего блока, сброшенный запрещает.

Таблица А.8 – Регистр загрузки регистра CLKC при отладке

CLKDEB

1FF6h значение после Сброса: 0000h



Поле	Бит	Описание
LOADEN	15	Бит разрешения загрузки регистра CLKC
		0 Запрещена
		1 Разрешена. Загрузка CLKC по прерыванию DEBUG
CLKCDEB_VAL	14–0	Биты, состояние которых будет загружаться в регистр CLKC одновременно с запросом на прерывание DEBUG. Загрузка возможна только при условии, что установлен седьмой бит регистра CLKDEB, соответствующий биту SLOW регистра CLKC. Использование регистра CLKDEB позволяет отключить тактирование определенных устройств для того, чтобы их регистры не изменили значение

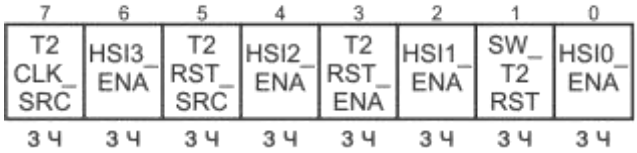
Таблица А.9 – Регистр 0 управления вводом/выводом

IOC0

15h значение после Сброса: 0000 00X0b

запись - горизонтальное окно 0

чтение - горизонтальное окно 15



Поле	Бит	Описание
1	2	3
T2CLK_SRC	7	Бит выбора внешнего источника синхронизации для таймера 2 (вывод P2.3)
		0 T2CLK
		1 HSI.1
HSI3_ENA	6	Бит разрешения захвата состояний вывода HSI.3
		0 Запрещено
		1 Разрешено
T2RST_SRC	5	Бит выбора источника внешнего сброса таймера 2
		0 Передний фронт сигнала на выводе T2RST
		1 Передний фронт сигнала на выводе HSI.0
HSI2_ENA	4	Бит разрешения захвата состояний вывода HSI.2
		0 Запрещено
		1 Разрешено

Окончание таблицы А.9

1	2	3	
T2RST_ENA	3	Бит разрешения внешнего сброса таймера 2	
		0	Запрещено
		1	Разрешено
HSI1_ENA	2	Бит разрешения захвата состояний вывода HSI.1	
		0	Запрещено
		1	Разрешено
SW_T2RST	1	Программный сброс таймера 2. При чтении этого бита всегда возвращается единица	
		0	Нет действий
		1	Запись единицы вызывает сброс таймера 2
HSI0_ENA	0	Бит разрешения захвата состояний вывода HSI.0	
		0	Запрещено
		1	Разрешено

Таблица А.10 – Регистр 1 управления вводом/выводом

IOC1

16h значение после Сброса: 21h

запись - горизонтальное окно 0

чтение - горизонтальное окно 15

7	6	5	4	3	2	1	0
HSI_INT	HSO5 ENA	TXD_SEL	HSO4 ENA	T2 OVF_INT	T1 OVF_INT	EXT INT_SRC	PWM_SEL
З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч

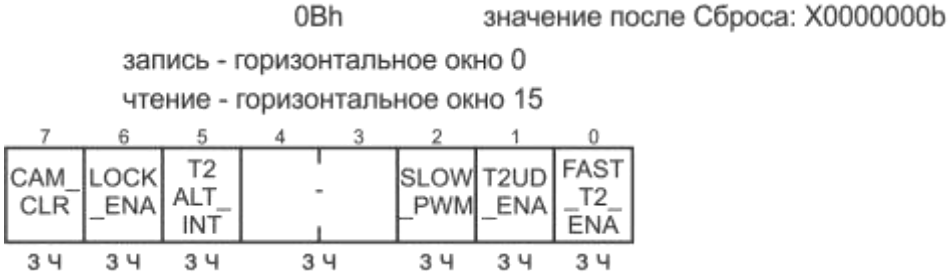
Поле	Бит	Описание	
1	2	3	
HSI_INT	7	Бит выбора источника прерывания HSI	
		0	Выходной буфер загружен
		1	Память FIFO заполнена
HSO5_ENA	6	Бит выбора функции вывода HSI.3	
		0	HSI.3
		1	HSO.5
TXD_SEL	5	Бит выбора режима работы выводов P2.0 и P2.6	
		0	Выводы общего назначения
		1	Выходы TXD0 и TXD1 последовательных портов
HSO4_ENA	4	Бит выбора функции вывода HSI.2	
		0	HSI.2
		1	HSO.4
T2OVF_INT	3	Бит разрешения прерывания по переполнению от таймера 2	
		0	Запрещено
		1	Разрешено
T1OVF_INT	2	Бит разрешения прерывания по переполнению от таймера 1	
		0	Запрещено
		1	Разрешено

Окончание таблицы А.10

1	2	3	
EXTINT_SRC	1	Бит выбора источника внешнего прерывания	
		0	Вывод P2.2
		1	Вывод P0.7
PWM_SEL	0	Бит выбора режима работы вывода P2.5	
		0	Вывод общего назначения
		1	Выход PWM.0 модуля ШИМ

Таблица А.11 – Регистр 2 управления вводом/выводом

IOIC2



Поле	Бит	Описание	
CAM_CLR	7	Бит очистки АЗУ модуля HSO. При чтении этого бита всегда возвращается единица	
		0	Нет действий
		1	Запись единицы вызывает очистку всего содержимого АЗУ
LOCK_ENA	6	Бит разрешения фиксации команд в АЗУ модуля HSO	
		0	Фиксация всех команд запрещена
		1	Фиксация всех команд разрешена. Параметры фиксации для каждой команды устанавливаются индивидуально посредством бита CAM_LOCK регистра HSO_COMMAND при загрузке в АЗУ
T2ALT_INT	5	Бит выбора границы переполнения таймера 2	
		0	FFFFh/0000h
		1	7FFFh/8000h
SLOW_PWM	2	Бит включения делителя на два входного синхросигнала	
		0	Выключен. Период выходного сигнала 256 машинных тактов
		1	Включен. Период выходного сигнала 512 машинных тактов
T2UD_ENA	1	Бит задания направления счета таймера 2	
		0	Таймер считает только вверх
		1	Если на выводе P2.6 (T2UPDN) ноль, то таймер считает вверх; Если на выводе P2.6 единица, то таймер считает вниз
FAST_T2_ ENA	0	Бит выбора режима работы таймера 2	
		0	Режим нормального счета. Счетчик таймера переключается каждые 8 тактов синхросигнала
		1	Режим скоростного счета. Счетчик таймера переключается с каждым тактом синхросигнала
–	4, 3	Зарезервировано	

Таблица А.12 – Регистр 3 управления вводом/выводом

IOC3

Поле	Бит	Описание	
PWM2_SEL	3	Бит включения альтернативной функции вывода P1.4. Состояние бита можно изменять без сброса устройства	
		0	Вывод общего назначения
		1	Вывод PWM2 блока ШИМ
PWM1_SEL	2	Бит включения альтернативной функции вывода P1.3. Состояние бита можно изменять без сброса устройства	
		0	Вывод общего назначения
		1	Вывод PWM1 блока ШИМ
T2_ENA	0	Бит выбора источника синхронизации таймера 2	
		0	Внешний источник (см. бит T2CLK_SRC регистра IOC0)
		1	Внутренний источник (внутренняя тактовая частота микроконтроллера)
–	7–4, 1	Зарезервировано	

Таблица А.13 – Регистр 0 состояния ввода/вывода

IOS0

Поле	Бит	Описание	
HR_STAT	7	Флаг состояния предварительного буфера модуля HSO. Устанавливается, когда производится запись в предварительный буфер	
		0	Буфер пуст
		1	Буфер не пуст
HRCAM_ STAT	6	Флаг состояния предварительного буфера и АЗУ модуля HSO	
		0	Буфер пуст и по крайней мере одно слово АЗУ пусто
		1	Буфер заполнен и/или заполнено АЗУ
HSO5_STAT	5	Бит состояния вывода HSO.5	
HSO4_STAT	4	Бит состояния вывода HSO.4	
HSO3_STAT	3	Бит состояния вывода HSO.3	
HSO2_STAT	2	Бит состояния вывода HSO.2	
HSO1_STAT	1	Бит состояния вывода HSO.1	
HSO0_STAT	0	Бит состояния вывода HSO.0	

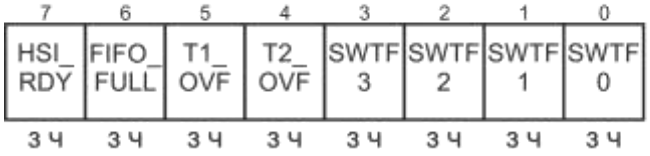
Таблица А.14 – Регистр 1 состояния ввода/вывода

IOS1

16h значение после Сброса: 00h

запись - горизонтальное окно 15

чтение - горизонтальное окно 0



Поле	Бит	Описание
HSI_RDY	7	Флаг загрузки выходного буфера модуля HSI. Устанавливается всегда, когда в выходной буфер загружается слово
		0 Состояние буфера не изменялось
		1 Буфер загружен
FIFO_FULL	6	Флаг заполнения памяти FIFO модуля HSI
		0 В памяти менее шести записанных слов
		1 В памяти шесть или более записанных слов
T1_OVF	5	Флаг переполнения таймера 1
		0 Нет действий
		1 Таймер переполнился
T2_OVF	4	Флаг переполнения таймера 2
		0 Нет действий
		1 Таймер переполнился
SWTF3	3	Флаг программного прерывания по команде SWT3 от модуля HSO
		0 Нет действий
		1 Сформирован запрос на прерывание
SWTF2	2	Флаг программного прерывания по команде SWT2 от модуля HSO
		0 Нет действий
		1 Сформирован запрос на прерывание
SWTF1	1	Флаг программного прерывания по команде SWT1 от модуля HSO
		0 Нет действий
		1 Сформирован запрос на прерывание
SWTF0	0	Флаг программного прерывания по команде SWT0 от модуля HSO
		0 Нет действий
		1 Сформирован запрос на прерывание

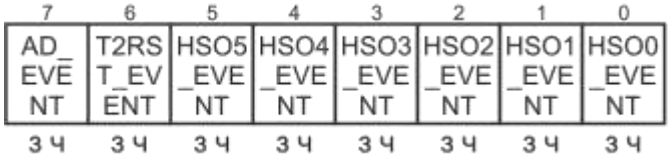
Таблица А.15 – Регистр 2 состояния ввода/вывода

IOS2

17h значение после Сброса: 00h

запись - горизонтальное окно 15

чтение - горизонтальное окно 0



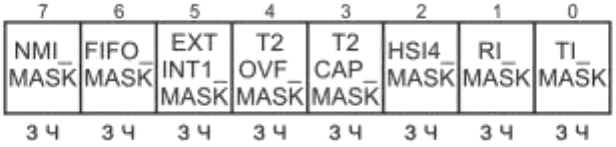
Поле	Бит	Описание	
AD_EVENT	7	Флаг запуска АЦП	
		0	Нет действий
		1	Команда модуля HSO запустила АЦ-преобразование
T2RST_EVENT	6	Флаг сброса таймера 2	
		0	Нет действий
		1	Команда модуля HSO сбросила таймер 2
HSO5_EVENT	5	Флаг переключения вывода HSO.5	
		0	Нет действий
		1	Команда модуля HSO переключила вывод
HSO4_EVENT	4	Флаг переключения вывода HSO.4	
		0	Нет действий
		1	Команда модуля HSO переключила вывод
HSO3_EVENT	3	Флаг переключения вывода HSO.3	
		0	Нет действий
		1	Команда модуля HSO переключила вывод
HSO2_EVENT	2	Флаг переключения вывода HSO.2	
		0	Нет действий
		1	Команда модуля HSO переключила вывод
HSO1_EVENT	1	Флаг переключения вывода HSO.1	
		0	Нет действий
		1	Команда модуля HSO переключила вывод
HSO0_EVENT	0	Флаг переключения вывода HSO.0	
		0	Нет действий
		1	Команда модуля HSO переключила вывод

Таблица А.16 – Регистр 1 маски прерываний

INT_MASK1

13h значение после Сброса: 00h

запись/чтение - любое горизонтальное окно



Поле	Бит	Описание
NMI_MASK	7	Нефункционирующий бит. Немаскируемое прерывание NMI (INT15) разрешено при любом состоянии этого бита
FIFO_MASK	6	Бит разрешения прерывания по окончании заполнения памяти FIFO модуля HSI (INT14)
EXTINT1_MASK	5	Бит разрешения внешнего прерывания EXINT (INT13). Вывод P2. 2 может генерировать или прерывание EXTINT (INT07) или прерывание EXTINT1 (INT13)
T2OVF_MASK	4	Бит разрешения прерывания по переполнению от таймера 2 (INT12)
T2CAP_MASK	3	Бит разрешения прерывания при захвате значения таймера 2 (INT11)
HSI4_MASK	2	Бит разрешения прерывания после четвертой записи в память FIFO модуля HSI (INT10)
RI_MASK	1	Бит разрешения прерываний (INT09): - по окончании приема от модулей USART0, USART1; - по линии прерываний 1 контроллера CAN
TI_MASK	0	Бит разрешения прерываний (INT08): - по окончании передачи от модулей USART0 и USART1; - по линии прерываний 0 контроллера CAN

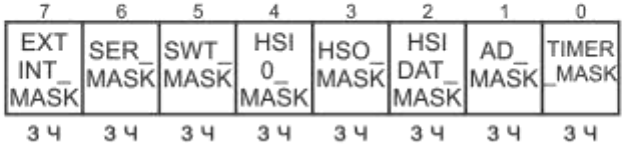
Примечание – Для разрешения прерывания следует установить соответствующий бит.

Таблица А.17 – Регистр маски прерываний

INT_MASK

08h значение после Сброса: 00h

запись/чтение - любое горизонтальное окно



Поле	Бит	Описание
EXTINT_ MASK	7	Бит разрешения внешнего прерывания EXTINT (INT07). Источник прерывания (вывод P0.7 или P2.2) выбирается битом EXTINT_SRC регистра IOC1

Окончание таблицы А.17

1	2	3
SER_MASK	6	Бит разрешения прерываний (INT6): - от контроллера SPI; - от контроллера I2C; - от контроллера LIN; - по линии прерываний 2 контроллера CAN
SWT_MASK	5	Бит разрешения программных прерываний по командам модуля HSO (INT05)
HSI0_MASK	4	Бит разрешения внешнего прерывания с вывода HSI.0 (INT04)
HSO_MASK	3	Бит глобального разрешения прерываний от HSO (INT03)
HSIDAT_ MASK	2	Бит разрешения прерывания по окончании формирования данных от HSI (INT02)
AD_MASK	1	Бит разрешения прерывания по окончании преобразования от АЦП (INT01)
TIMER_ MASK	0	Бит разрешения прерывания по переполнению от таймеров 1 и 2 (INT00)
Примечание – Для разрешения прерывания следует установить соответствующий бит.		

Таблица А.18 – Регистр 1 ждущих прерываний

INT_PEND1

12h значение после Сброса: 00h

запись/чтение - любое горизонтальное окно

7	6	5	4	3	2	1	0
NMI_ PEND	FIFO_ PEND	EXT INT1_ PEND	T2 OVF_ PEND	T2 CAP_ PEND	HSI4_ PEND	RI_ PEND	TI_ PEND
З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч

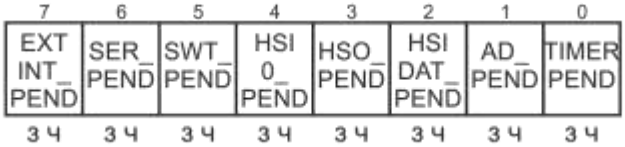
Поле	Бит	Описание
NMI_PEND	7	Индикатор запроса на прерывание NMI
FIFO_PEND	6	Индикатор запроса на прерывание по окончании заполнения памяти FIFO модуля HSI
EXTINT1_PEND	5	Индикатор запроса на прерывание с вывода EXINT1
T2OVF_PEND	4	Индикатор запроса на прерывание по переполнению таймера 2
T2CAP_PEND	3	Индикатор запроса на прерывание при захвате значения таймера 2
HSI4_PEND	2	Индикатор запроса на прерывание после четвертой записи в память FIFO модуля HSI
RI_PEND	1	Индикатор запроса на прерывание: - по окончании приема от модулей UART0 и UART1; - по линии прерываний 1 контроллера CAN
TI_PEND	0	Индикатор запроса на прерывание: - по окончании передачи от модулей UART0 и UART1; - по линии прерываний 0 контроллера CAN
Примечание – Установленный бит сигнализирует об ожидании обслуживания соответствующего запроса на прерывание. Индикатор запроса на прерывание сбрасывается аппаратно, как только происходит переход по соответствующему вектору прерывания		

Таблица А.19 – Регистр ждущих прерываний

INT_PEND

09h значение после Сброса: 00h

запись/чтение - любое горизонтальное окно



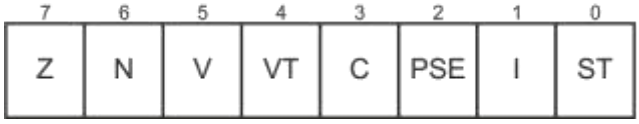
Поле	Бит	Описание
EXTINT_PEND	7	Индикатор запроса на прерывание с вывода EXINT
SER_PEND	6	Индикатор запроса на прерывание: - от контроллера SPI; - от контроллера I2C; - от контроллера LIN; - по линии прерываний 2 контроллера CAN
SWT_PEND	5	Индикатор запроса программного прерывания по команде модуля HSO
HSI0_PEND	4	Индикатор запроса на прерывание с вывода HSI.0
HSO_PEND	3	Индикатор запроса на прерывание от HSO
HSIDAT_PEND	2	Индикатор запроса на прерывание по окончании формирования данных от HSI
AD_PEND	1	Индикатор запроса на прерывание по окончании преобразования от АЦП
TIMERPEND	0	Индикатор запроса на прерывание по переполнению от таймеров 1 и 2

Примечание – Установленный бит сигнализирует об ожидании обслуживания соответствующего запроса на прерывание. Индикатор запроса на прерывание сбрасывается аппаратно, как только происходит переход по соответствующему вектору прерывания.

Таблица А.20 – Регистр состояния программы

PSW

значение после Сброса: --



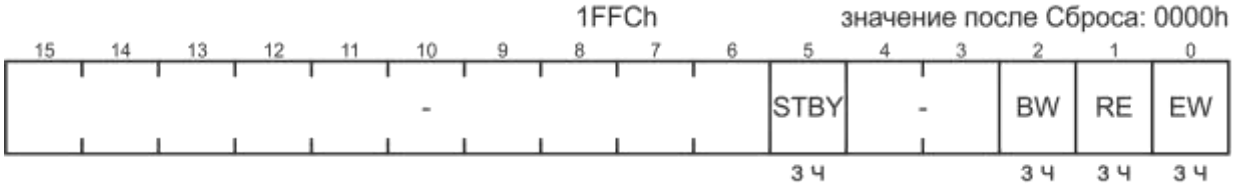
Поле	Бит	Описание
1	2	3
Z	7	Флаг нуля. Устанавливается, если результат операции равен нулю. При операциях сложения с переносом и вычитания с заемом этот флаг никогда не устанавливается, но он сбрасывается, если результат не нулевой. Таким образом, флаг нулевого результата показывает наличие нулевого или ненулевого результата при вычислениях с большой точностью

Окончание таблицы А.20

1	2	3
N	6	Флаг отрицательного результата. Устанавливается, если после выполнения операции получен отрицательный результат. Флаг правильный, если даже случилось переполнение. При всех сдвиговых операциях и операции нормализации этот флаг устанавливается равным старшему значащему биту результата, даже если счётчик сдвигов равен нулю
V	5	Флаг переполнения. Устанавливается, если после выполнения операции получен результат, выходящий за границы диапазона значений типа данных приёмника. При сдвиговых операциях (SHL, SHLB, SHLL) этот флаг устанавливается, если старший значащий бит операнда изменился в процессе сдвига. При операциях деления этот флаг устанавливается, если при выполнении команды: – DIVUB результат > 255 (FFh); – DIVU результат > 65 535 (FFFFh); – DIVB результат < –127 (81h) или > 127 (7Fh); – DIV результат < –32 767 (8001h) или > 32 767 (7FFFh)
VT	4	Дополнительный флаг переполнения. Устанавливается, когда флаг переноса C установлен. Очищается только командами CLRVT, JVT, JNVT. Это позволяет тестировать возможное состояние переполнения в конце последовательности родственных арифметических операций, что более эффективно, чем тестирование флага переполнения после любой операции
C	3	Флаг переноса. Устанавливается для индикации арифметического переноса из старшего разряда ALU или состояния, когда последний значащий бит выдвигается за пределы операнда. Если вычитание содержит заём, флаг переноса сбрасывается. Если значение сдвинутых битов < 1/2 LSB, то C = 0; если ≥ 1/2 LSB, то C = 1. Обычно результат округляется до большего значения при установленном флаге переноса. Флаг ST позволяет улучшить точность округления. Если значение сдвинутых битов: а) 0 и бит ST = 0, то C = 0; б) > 0 и < 1/2 LSB и ST = 1, то C = 0; в) = 1/2 LSB и ST = 0, то C = 1; г) > 1/2 LSB и < 1 LSB и ST = 1, то C = 1
PSE	2	Бит разрешения PTS. Устанавливается командой EPTS. Сбрасывается командой DPTS. Установленный бит PSE глобально разрешает работу сервера периферийных транзакций (PTS)
I	1	Бит глобального разрешения прерываний. Устанавливается командой EI. Сбрасывается командой DI. Установленный бит I разрешает обслуживание всех прерываний, кроме прерываний NMI, TRAP и неподдерживаемого кода
ST	0	Дополнительный флаг переноса (после операции умножения имеет неопределенное состояние). Устанавливается, чтобы показать, что во время сдвига вправо был установлен флаг переноса C, а затем сброшен (т.е. через перенос прошла единица). Флаг ST может использоваться вместе с флагом переноса C для более точного округления.

Таблица А.21 – Регистр управления программированием EEPROM

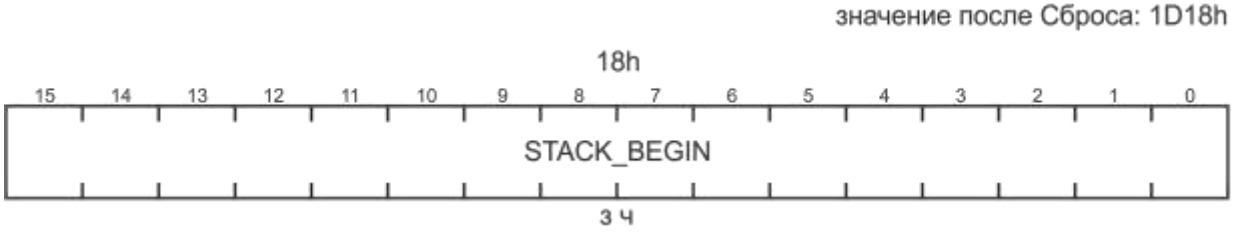
ROMC



Поле	Бит	Описание
STBY	5	Бит перевода ПЗУ в режим пониженного потребления. Не следует записывать единицу, если включен режим работы с внутренней памятью. При работе с внешней памятью этот бит автоматически после сброса устанавливается. Время перехода из режима пониженного потребления в активный режим – не менее 5 мкс.
BW	2	Бит разрешения стирания блока памяти. Бит EW должен быть установлен. Стирание осуществляется подачей команды пересылки нулевых данных по любому адресу, находящемуся в выбранном блоке. В микросхеме 16 блоков EEPROM, каждый емкостью 1К×16 бит. Если микросхема находится в режиме защиты ПЗУ от записи, то подача команды на стирание блока приведет к полному стиранию содержимого ПЗУ
RE	1	Бит разрешения стирания строки памяти. Бит EW должен быть установлен. Стирание осуществляется подачей команды пересылки нулевых данных по любому адресу, находящемуся в выбранной строке. В каждом блоке 128 строк по 16 байт.
EW	0	Бит разрешения записи слова в EEPROM. Запись осуществляется подачей любой команды пересылки. После каждой записи бит сбрасывается и требуется повторная установка бита.
–	15–6, 4, 3	Зарезервировано

Таблица А.22 – Регистр указателя стека

SP



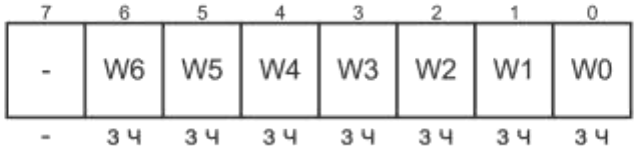
Поле	Бит	Описание
STACK_ BEGIN	15 – 0	Поле адреса вершины стека. Записанное значение должно быть четным и на два больше, чем желаемый адрес вершины стека

Таблица А.23 – Регистр выбора окна

WSR

14h значение после Сброса: X0h

запись/чтение - любое горизонтальное окно



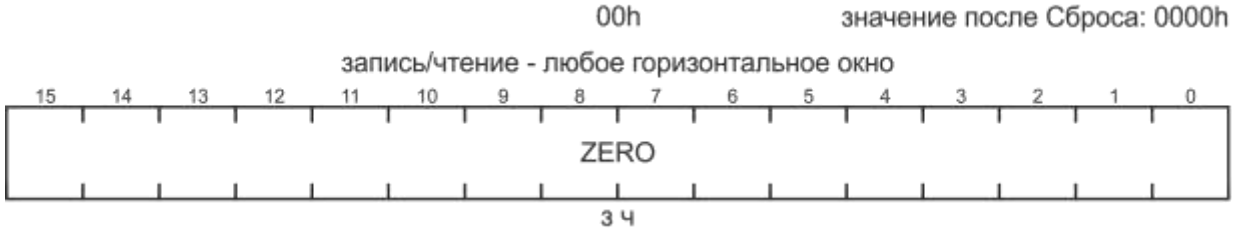
Поле	Бит	Описание	
W6	6	Установка этого бита выбирает 32-байтные вертикальные окна. В других случаях этот бит должен быть равен нулю	
W5	5	Установка этого бита выбирает 64-байтные вертикальные окна. В других случаях этот бит должен быть равен нулю	
W4	4	Установка этого бита выбирает 128-байтные вертикальные окна. В других случаях этот бит должен быть равен нулю	
W3–W0	3–0	Поле указания номера окна. В эти биты записывается номер желаемого горизонтального или вертикального окна. Размер окна выбирается установкой WSR.4, WSR.5 или WSR.6. Действующие горизонтальные окна – 0, 1 и 15. Действующие вертикальные окна – 0 – 15. Для выбора вертикальных окон 16 – 31 необходимо установить WSR.4 и WSR.6	
		0000	Горизонтальное окно 0
		0001	Горизонтальное окно 1
		1111	Горизонтальное окно 15
		Остальные значения зарезервированы	
–	7	Зарезервировано	

Таблица А.24 – Допустимые состояния регистра WSR

Допустимое состояние регистра WSR	Выбранный режим работы
0 1 numwin	32-байтные вертикальные окна. В битовом поле numwin задается порядковый номер окна от 00h до 3Fh
0 0 1 numwin	64-байтные вертикальные окна. В битовом поле numwin задается порядковый номер окна 00h до 1Fh
0 0 0 1 numwin	128-байтные вертикальные окна. В битовом поле numwin задается порядковый номер окна 00h до 0Fh
0 0 0 0 0 0 0 0	Горизонтальное окно 0
0 0 0 0 0 0 0 1	Горизонтальное окно 1
0 0 0 0 1 1 1 1	Горизонтальное окно 15

Таблица А.25 – Регистр нуля

ZERO_REG



Поле	Бит	Описание
ZERO	15–0	Нулевое слово (все биты всегда нули). При использовании регистра ZERO_REG в качестве источника при выполнении команды CMPL, двойное слово приемника сравнивается с 32-битным нулем

Регистры блока PTS

Таблица А.26 – Регистр выбора PTS

PTSSEL

Бит	Описание	
14, 13, …, 1, 0	Бит задания варианта обслуживания прерывания	
	0	Прерывание обслуживается контроллером прерываний
	1	Прерывание обслуживается блоком PTS
15	Зарезервировано	

Таблица А.27 – Регистр обслуживания PTS

PTSSRV

Бит	Описание	
14, 13, …, 1, 0	Индикатор запроса на прерывание END-OF-PTS от блока PTS	
	0	Нет запроса на прерывание
	1	Запрос на прерывание ожидает обслуживания
15	Зарезервировано	

Таблица А.28 – Регистр количества циклов PTS

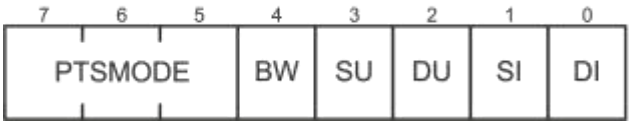
PTSCOUNT

Поле	Бит	Тип	Описание
PTS_COUNT	7–0	Запись	Поле задания количества циклов PTS (количество передач), которые будут выполняться блоком PTS без участия центрального процессора

Таблица А.29 – Регистр управления PTS в режиме одиночной/блочной передачи

PTSCON

значение после Сброса: 00h

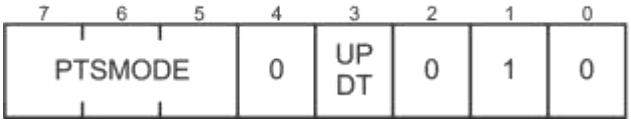


Поле	Бит	Тип	Описание	
PTS MODE	7–5	Запись	Выбор режима	
			000	Одиночная передача
			100	Блочная передача
BW	4	Запись	Бит выбора формата передачи	
			0	Передача слова
			1	Передача байта
SU	3	Запись	Бит изменения значения регистра адреса источника PTSSRC	
			0	В регистре сохраняется значение, которое было в начале цикла PTS
			1	В регистре сохраняется значение, которое возникло на момент окончания цикла PTS
DU	2	Запись	Бит изменения значения регистра адреса приемника PTSDST	
			0	В регистре сохраняется значение, которое было в начале цикла PTS
			1	В регистре сохраняется значение, которое возникло на момент окончания цикла PTS
SI	1	Запись	Бит автоинкремента регистра адреса источника PTSSRC	
			0	Нет действий
			1	Значение регистра инкрементируется в конце каждого цикла PTS: - на единицу при передаче байта; - на два при передаче слова
DI	0	Запись	Бит автоинкремента регистра адреса приемника PTSDST	
			0	Нет действий
			1	Значение регистра инкрементируется в конце каждого цикла PTS: - на единицу при передаче байта; - на два при передаче слова

Таблица А.30 – Регистр управления PTS в режиме HSI/HSO

PTSCON

значение после Сброса: 02h



Поле	Бит	Тип	Описание	
PTS MODE	7–5	Запись	Выбор режима обмена	
			001	HSI (соответствующий регистр PTSDST)
			011	HSO (соответствующий регистр PTSSRC)
UPDT	3	Запись	Бит изменения значения в регистре адреса сточника/приемника: - PTSSRC (в режиме HSO); - PTSDST (в режиме HSI)	
			0	В регистре сохраняется значение, которое было в начале цикла обмена
			1	В регистр сохраняется значение, которое возникло на момент окончания цикла обмена
–	4, 2–0	–	Зарезервировано. При чтении возвращаются указанные значения	

Таблица А.31 – Регистр адреса источника PTS

PTSSRC

значение после Сброса: XXXXh

Поле	Бит	Тип	Описание
PTSSRC_H	15–8	Запись	Старший байт адреса источника обмена
PTSSRC_L	7–0	Запись	Младший байт адреса источника обмена

Таблица А.32 – Регистр адреса приемника PTS

PTSDST

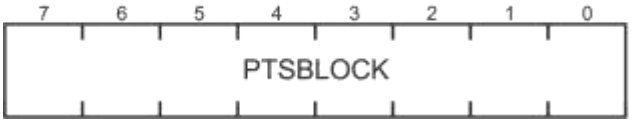
значение после Сброса: XXXXh

Поле	Бит	Тип	Описание
PTSDST_H	15–8	Запись	Старший байт адреса приемника PTS
PTSDST_L	7–0	Запись	Младший байт адреса приемника PTS

Таблица А.33 – Регистр числа передач PTS

PTSBLOCK

значение после Сброса: XXh

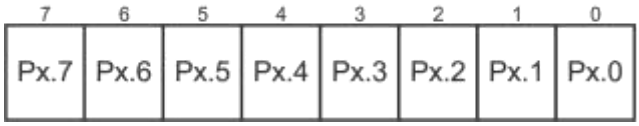


Поле	Бит	Тип	Описание
PTS BLOCK	7–0	Запись	Поле задания количества передаваемых байт/слов за один цикл PTS. Максимально возможное число, которое может быть записано в PTSBLOCK, зависит от режима работы, задаваемого полем PTSMODE регистра PTSCON. Так для: - режима блочной передачи это число – 32 (20h); - режима HSI – 7 (07h); - режима HSO – 8 (08h). Во всех режимах запись значения 00h в поле PTSBLOCK автоматически задает максимальное число, соответствующее режиму

Регистры портов

Таблица А.34 – Регистр ввода/вывода порта x

IOPORTx



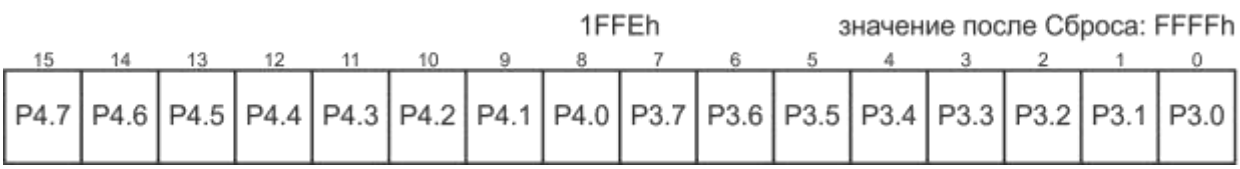
Поле	Бит	Тип	Описание
Px.7	7	См. таблицу А.35	Бит управления состоянием седьмого вывода порта x
Px.6	6		Бит управления состоянием шестого вывода порта x
Px.5	5		Бит управления состоянием пятого вывода порта x
Px.4	4		Бит управления состоянием четвертого вывода порта x
Px.3	3		Бит управления состоянием третьего вывода порта x
Px.2	2		Бит управления состоянием второго вывода порта x
Px.1	1		Бит управления состоянием первого вывода порта x
Px.0	0		Бит управления состоянием нулевого вывода порта x
Примечание – Символ x заменяет номер порта (см. таблицу А.35)			

Таблица А.35 – Регистры IOPORT0, IOPORT1, IOPORT2, IOPORT3, IOPORT4 и IOPORT5

Порт (x)	Регистр	Адрес	Значение после сброса
0	IOPORT0	0Eh (только чтение – горизонтальное окно 0) или 1FF8h (только чтение).	XXh
1	IOPORT1	0Fh (чтение/запись – горизонтальное окно 0)	FFh
2	IOPORT2	10h (чтение/запись – горизонтальное окно 0)	C1h

Таблица А.36 – Регистр ввода/вывода портов 3 и 4

IOPORT34

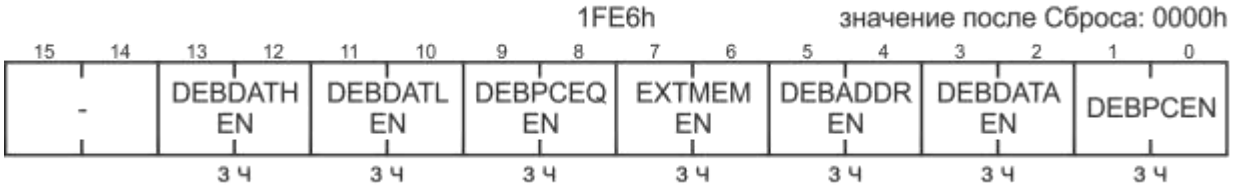


Поле	Бит	Тип	Описание
Px.7	7	Запись и чтение только как слово	Бит управления состоянием седьмого вывода порта x
Px.6	6		Бит управления состоянием шестого вывода порта x
Px.5	5		Бит управления состоянием пятого вывода порта x
Px.4	4		Бит управления состоянием четвертого вывода порта x
Px.3	3		Бит управления состоянием третьего вывода порта x
Px.2	2		Бит управления состоянием второго вывода порта x
Px.1	1		Бит управления состоянием первого вывода порта x
Px.0	0		Бит управления состоянием нулевого вывода порта x
Примечание – Символ x заменяет номер порта.			

Регистры модуля OCDS

Таблица А.37 – Регистр управления модулем отладки

DEBCTRL

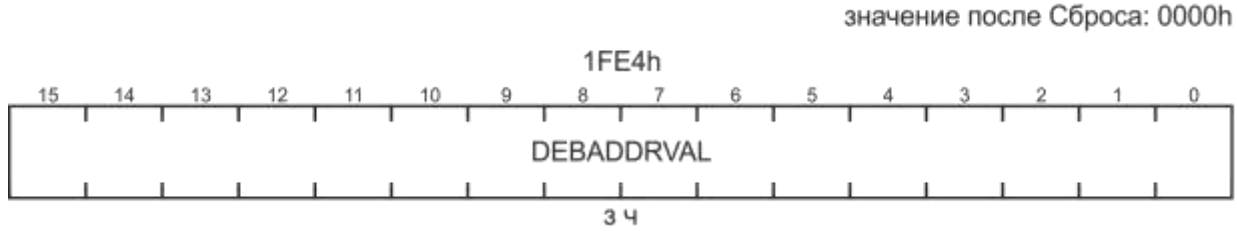


Поле	Бит	Описание события, после обнаружения которого система отладки выполнит запрограммированное действие
DEBDATHEN	13–12	Появление значения, заданного в регистре DEBDATH на старших разрядах внутренней шины данных
DEBDATLEN	11–10	Появление значения, заданного в регистре DEBDATL на младших разрядах внутренней шины данных
DEBPCEQEN	9–8	Совпадение текущего значения счетчика команд со значением регистра DEBPCEQ
EXTMEMEN	7–6	Выборка команды из области внешней памяти
DEBADDREN	5–4	Появление значения, заданного в регистре DEBADDR на внутренней шине адресов операндов
DEBDATAEN	3–2	Появление значения, заданного в регистре DEBDATA на внутренней шине данных
DEBPCEN	1–0	Текущее значение счетчика команд больше значения регистра DEBPC
–	15, 14	Зарезервировано

Примечание – Возможные программируемые действия:
00 – нет действий;
01 – прерывание DEBUG (2060h);
10 – переход в IDLE;
11 – сброс микроконтроллера.

Таблица А.38 – Регистр ожидаемого значения адреса

DEBADDR



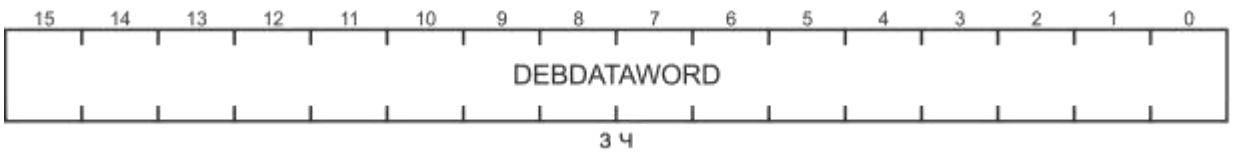
Поле	Бит	Описание
DEBADDRVAL	15–0	Значение адреса, при появлении которого на внутренней шине адресов выполняется действие, заданное полем DEBADDREN регистра DEBCTRL

Таблица А.39 – Регистр ожидаемого значения данных

DEBDATA

значение после Сброса: 0000h

1FE2h



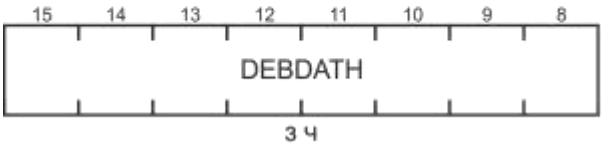
Поле	Бит	Описание
DEBDATAWORD	15–0	Значение данных, при появлении которых на внутренней шине данных выполняется действие, заданное полем DEBDATAEN регистра DEBCTRL

Таблица А.40 – Регистры ожидаемых значений старшего и младшего байт данных

DEBDATH

значение после Сброса: 00h

1FEFh



DEBDATL

значение после Сброса: 00h

1FEEh

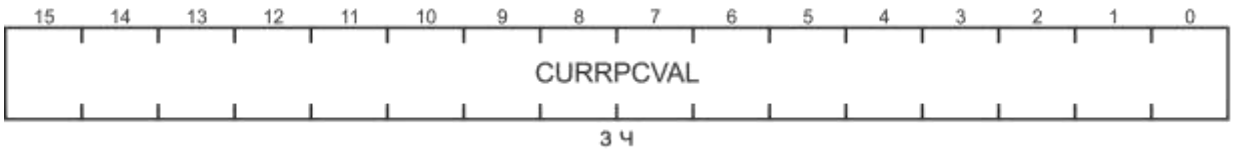
Поле	Бит	Описание
DEBDATH	15–8	Значение старшего байта данных, при появлении которого на старших разрядах внутренней шины данных выполняется действие, заданное полем DEBDATHEN регистра DEBCTRL
DEBDATL	7–0	Значение младшего байта данных, при появлении которого на младших разрядах внутренней шины данных выполняется действие, заданное полем DEBDATLEN регистра DEBCTRL

Таблица А.41 – Регистр текущего значения счетчика команд

CURRPC

значение после Сброса: XXXXh

1FE8h

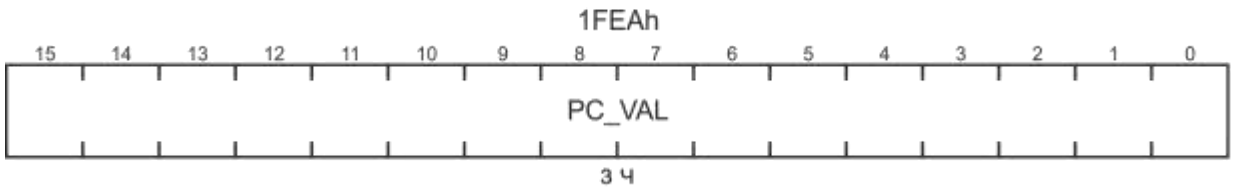


Поле	Бит	Описание
CURRPCVAL	15–0	Текущее значение основного счетчика команд

Таблица А.42 – Регистр значения счетчика команд

HAPPPC

значение после Сброса: XXXXh

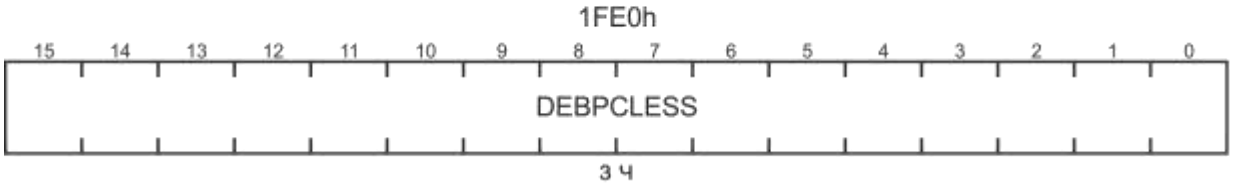


Поле	Бит	Описание
PC_VAL	15–0	Значение счетчика команд, которое было на момент возникновения ожидаемого события. Не сбрасывается во время системного сброса, сохраняя таким образом значение адреса команды, вызвавшей сброс микроконтроллера

Таблица А.43 – Регистр ожидаемого значения счетчика команд

DEBPC

значение после Сброса: 0000h

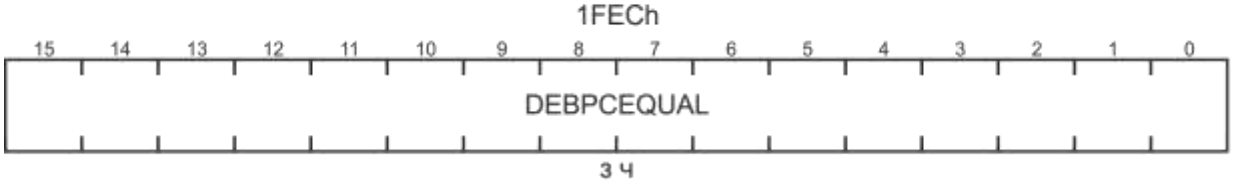


Поле	Бит	Описание
DEBPCLESS	15–0	Значение, при превышении которого счетчиком команд выполняется действие, заданное полем DEBPCEN регистра DEBCTRL

Таблица А.44 – Регистр ожидаемого значения счетчика команд

DEBPCEQ

значение после Сброса: 0000h



Поле	Бит	Описание
DEBPCEQUAL	15–0	Значение, при достижении которого счетчиком команд выполняется действие, заданное полем DEBPCEQEN регистра DEBCTRL

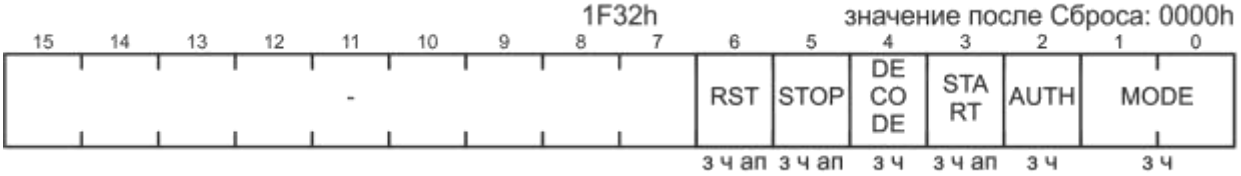
Регистры блока кодирования по ГОСТ 28147–89

Таблица А.45 – Адреса доступа к 64/256/512-битным регистрам блока кодирования

Мнемоническое обозначение регистра	Разрядность, бит	Адрес доступа к регистру	Обязательное количество последовательных циклов обращения к регистру, если формат данных – слово
CDDATA	64	1F30h	4 (запись/чтение)
CDKEY	256	1F34h	16 (только запись)
CDSBSTN	512	1F36h	32 (только запись)
CDUW	64	1F38h	4 (только запись)
CDAUTH	64	1F3Eh	4 (только чтение)

Таблица А.46 – Регистр управления

CDCON



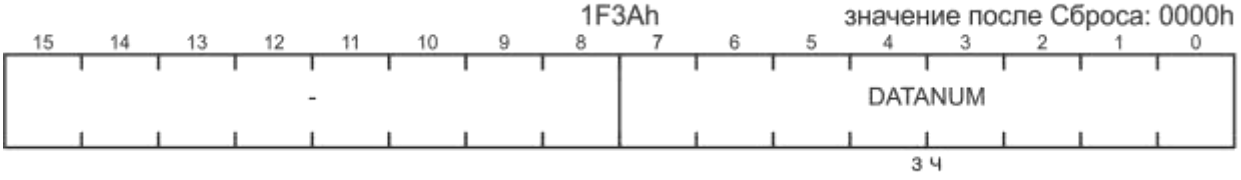
Поле	Бит	Описание
1	2	3
RST	6	Сброс блока кодирования. Установка этого бита приводит к сбросу всех регистров, за исключением регистра CDCON. Бит сбрасывается аппаратно
STOP	5	Флаг последнего блока данных. Установка этого бита перед записью очередного блока данных в регистр CDDATA будет означать, что записываемый блок данных является последним и по окончании его обработки процесс кодирования/ декодирования следует завершить (независимо от значения, хранящегося в регистре CDDATNUM). Бит сбрасывается аппаратно после начала обработки загруженного блока данных
DECODE	4	Бит выбора направления преобразования данных
		0 Кодирование
		1 Декодирование
START	3	Бит запуска. Установка этого бита после записи первого блока данных запускает процесс кодирования/декодирования. Для обработки одного или более блоков данных бит следует установить только один раз. Бит сбрасывается аппаратно
AUTH	2	Бит включения вычисления имитовставки (контрольной суммы)
		0 Имитовставка не вычисляется
		1 Параллельно с процессом кодирования/декодирования идет вычисление имитовставки
		Следует помнить, что вычисление имитовставки замедляет процесс обработки блока данных в 1,5 раза

Окончание таблицы А.46

1	2	3	
MODE	1–0	Поле выбора режим работы	
		00	Режим бездействия. Запись значения 00h во время процесса кодирования/ декодирования приводит к остановке работы и сбросу логики блока кодирования. В отличие от результата установки бита RST, все регистры сохраняют свое состояние
		01	Простая замена
		10	Гаммирование
		11	Гаммирование с обратной связью
–	15–7	Зарезервировано	

Таблица А.47 – Регистр числа блоков данных

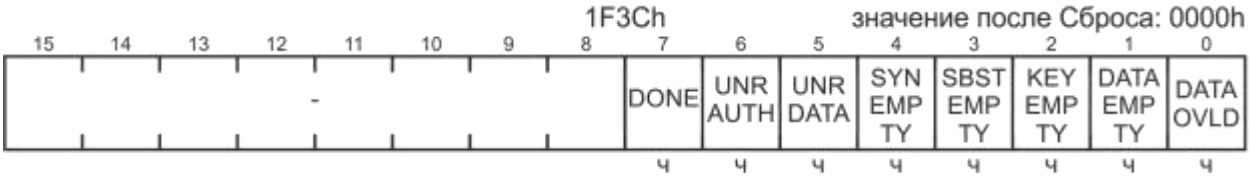
CDDATNUM



Поле	Бит	Описание
DATANUM	7–0	Число 64-разрядных блоков открытых данных, которые подлежат кодированию/декодированию
–	15–8	Зарезервировано

Таблица А.48 – Регистр состояния блока кодирования

CDSTATE



Поле	Бит	Описание
1	2	3
DONE	7	Флаг завершения преобразования
UNRAUTH	6	Предупреждение «Непрочитанная имитовставка». Флаг устанавливается в случае, если к моменту запуска преобразования (установка бита START) в регистре имитовставки CDAUTH находятся полностью или частично непрочитанные данные
UNRDATA	5	Предупреждение «Непрочитанные данные». Флаг устанавливается в случае, если к моменту запуска преобразования в регистре данных CDDATA находятся полностью или частично непрочитанные данные

Окончание таблицы А.48

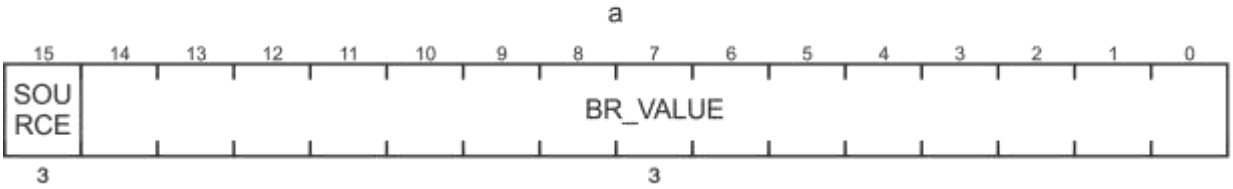
1	2	3
SYNEMPTY	4	Ошибка синхровставки. Флаг устанавливается в случае, если к моменту запуска преобразования в режиме гаммирования (MODE = 10/11b) полностью или частично не заполнен регистр синхровставки CDUW
SBSTEMPTY	3	Ошибка регистра замены. Флаг устанавливается в случае, если к моменту запуска преобразования полностью или частично не заполнен регистр замены CDSBSTN
KEYEMPTY	2	Ошибка ключа. Флаг устанавливается в случае, если к моменту запуска преобразования полностью или частично не заполнен регистр ключа CDKEY
DATAEMPTY	1	Ошибка данных. Флаг устанавливается в случае, если к моменту запуска преобразования полностью или частично не заполнен регистр данных CDDATA
DATAOVLD	0	Предупреждение «Перегрузка данных». Флаг устанавливается: - если в буфер данных CDDATA записывается 9-й байт данных до начала преобразования (до установки бита START); - если в буфер данных записывается байт данных во время преобразования блока данных
–	15–8	Зарезервировано

Регистры портов USART0 и USART1

Таблица А.49 – Регистр скорости передачи порта USARTx

BAUD_RATEx

значение после Сброса: 0000h



Поле	Бит	Описание
SOURCE	15	Бит выбора источника синхросигнала
		0 – Сигнал с вывода T2CLK
		1 – Сигнал с вывода XTAL1
BR_ VALUE	14–0	Скорость передачи. Для расчета значения, которое следует записать в это поле, следует воспользоваться нижеприведенными формулами. Для синхронного режима (Режим 0): $BR\_VALUE_{DEC}\left(\frac{fuart}{Bod}\right)-1,$ (A.1) где BR_VALUE_{DEC} – скорость передачи в десятичном формате, fuart – частота синхросигнала, поступающего на блок, Гц, Bod – желаемая скорость передачи, бод. Для асинхронных режимов (Режимы 1, 2, 3): $BR\_VALUE_{DEC}\left(\frac{fuart}{Bod\times 8}\right)-1.$ (A.2)

Примечание – Следует помнить, что при использовании синхросигнала с вывода T2CLK значение его частоты подставляется в формулы в неизменном виде, а при использовании синхросигнала с вывода XTAL1 значение частоты предварительно уменьшается вдвое. В таблице А.50 приведены значения BR_VALUE, рассчитанные для некоторых стандартных скоростей (x – номер порта 0, 1).

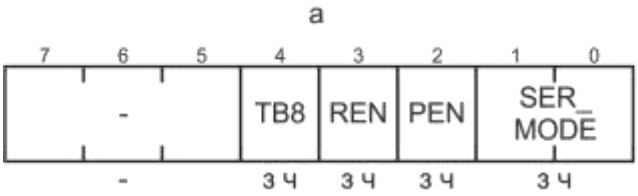
Таблица А.50 – Значения BR_VALUE в зависимости от источника синхросигнала

Скорость обмена	Источник синхросигнала и режим работы			
	XTAL1 (16МГц)		T2CLK (8 МГц)	
	Режим 0	Режимы 1, 2, 3	Режим 0	Режимы 1, 2, 3
9600 бод	8340h	8067h	0340h	0067h
4800 бод	8682h	80CFh	0682h	00CFh
2400 бод	8D04h	81A0h	0D04h	01A0h
1200 бод	9A0Ah	8340h	1A0Ah	0340h
600 бод	B415h	8682h	3415h	0682h

Таблица А.51 – Регистр управления USARTx

SP_CONx

значение после Сброса: 0Bh

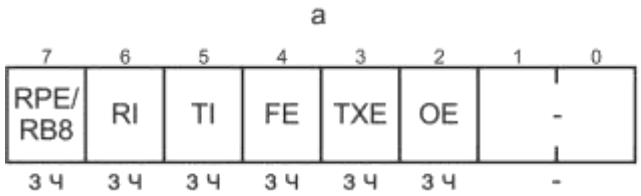


Поле	Бит	Описание	
TB8	4	Бит четности. 9-й бит данных, передаваемый в режимах 2 и 3	
REN	3	Бит разрешения приема	
		0	Прием данных на входе RxD запрещен
		1	Прием данных на входе RxD разрешен
PEN	2	Бит разрешения контроля по паритету	
		0	Запрещен
		1	Разрешен
SER_ MODE	1, 0	Поле выбор режима работы	
		00	Режим 0
		01	Режим 1
		10	Режим 2
		11	Режим 3
–	7–5	Зарезервировано	
Примечание – x – номер порта 0, 1.			

Таблица А.52 – Регистр состояния USARTx

SP_STATx

значение после Сброса: 0Bh



Поле	Бит	Описание
1	2	3
RPE/ RB8	7	Флаг ошибки паритета/принятый девятый бит. Если контроль по паритету разрешен (установлен бит PEN в регистре SP_CON), то флаг RPE установится, если будет обнаружена ошибка паритета в принятом байте. В случае отсутствия проверки по паритету, в бит RB8 будет записано значение девятого бита принятого байта (режимы 2 и 3)
RI	6	Флаг окончания приема. В режиме 0 – устанавливается аппаратно после приема 8-ого бита. В режимах 1, 2, 3 – устанавливается аппаратно в течение приема стоп-бита

Окончание таблицы А.52

1	2	3
TI	5	Флаг окончания передачи. В режиме 0 – устанавливается после передачи восьмого бита. В режимах 1, 2, 3 – устанавливается в течение передачи стоп-бита
FE	4	Флаг ошибки фрейма. Устанавливается, если при приеме байта не был получен стоп-бит
TXE	3	Флаг пустого буфера. Устанавливается, если буфер SBUF_TX и передающий сдвиговый регистр пусты. Флаг сбрасывается сразу же после записи в регистр SBUF_TX
OE	2	Флаг потери данных. Устанавливается, если регистр SBUF_RX не был прочитан до того, как в него был записан очередной принятый байт
–	1, 0	Зарезервировано
Примечание – x – номер порта 0, 1.		

Таблица А.53 – Регистр принятых данных USARTx

SBUF_RXx

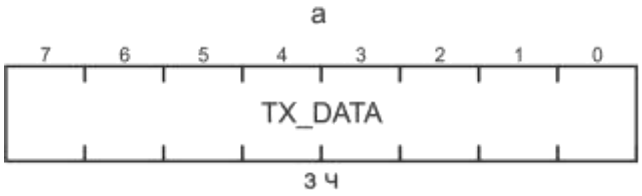
значение после Сброса: 00h

Поле	Бит	Описание
RX_DATA	7–0	Поле принятых данных
Примечание – x – номер порта 0, 1.		

Таблица А.54 – Регистр передаваемых данных USARTx

SBUF_TXx

значение после Сброса: 00h



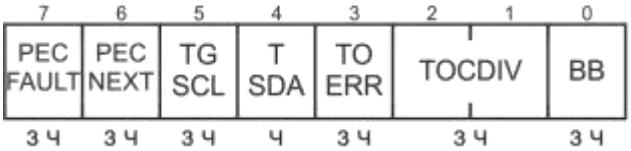
Поле	Бит	Описание
TX_DATA	7–0	Поле данных для передачи
Примечание – x – номер порта 0, 1.		

Регистры контроллера интерфейса I2C

Таблица А.55 – Регистр управления и состояния I2C

SMBCST

1FB4h значение после Сброса: 00h



Поле	Бит	Описание
1	2	3
PECFAULT	7	Флаг ошибки. Устанавливается в случае, если после расчета контрольной суммы для пакета данных и сравнения ее с полученной значение во внутреннем регистре ошибок ненулевое
PECNEXT	6	Бит управления отправкой байта контрольной суммы. Установка бита указывает на то, что следующий передаваемый байт будет байтом CRC (байт контрольной суммы). Реакция на установку бита PECNEXT зависит от режима работы. В режиме мастера передатчика установка бита PECNEXT вызовет загрузку результата вычисления CRC в регистр SMBSDA. После сброса флага INT начнется передача байта CRC. В режиме приемника установка этого бита будет указывать логике управления на то, что следующий байт, который будет принят, будет байтом CRC. В режиме ведомого приемника модуль I2C автоматически будет квитировать или не квитировать прием байта CRC, в зависимости от того, будет ли выявлена ошибка пакета данных или нет. В режиме мастера приемника по окончании приема байта CRC, будет отправлено значение бита ACK регистра SMBCTRL1
TGSCL	5	Бит переключения SCL. Бит позволяет переключать вывод SCL во время восстановления после ошибки. Когда на выводе SDA низкий уровень сигнала, запись «1» в бит TGSCL переключит вывод SCL на один такт. Когда на SDA высокий уровень сигнала, запись «1» в бит TGSCL игнорируется. Бит очищается аппаратно по окончании такта
TSDA	4	Бит тестирования SDA. Содержит текущее значение SDA. Этот бит можно использовать для отслеживания окончания процесса восстановления после ошибки, в течение которого ведомый постоянно поддерживает низкий уровень сигнала на выводе SDA
TOERR	3	Флаг ошибки простоя на шине. Если TOERR = 1b, это указывает на то, что на линии SCL был обнаружен простой. Флаг TOERR выставляется по обнулению основного счетчика времени простоя и может быть сброшен записью «1» в бит CLRST регистра SMBCTRL1

Окончание таблицы А.55

1	2	3	
TOCDIV	2, 1	Поле коэффициента делителя. Устанавливает коэффициент деления системного тактового сигнала, подаваемого на предделитель времени простоя линии SCL	
		00	Тактовый сигнал отсутствует
		01	Деление на 4
		10	Деление на 8
		11	Деление на 16
BB	0	Флаг занятости шины. Если BB = 1b, это указывает на то, что шина занята. Устанавливается, как только шина переходит в активное состояние (одновременное появление низкого уровня сигнала на выводах SDA и SCL или хотя бы на одном из них) или в стартовое состояние. Сбрасывается при выключении интерфейса I2C, либо при обнаружении состояния останова	

Таблица А.56 – Регистр 1 управления I2C

SMBCTRL1

1FB6h значение после Сброса: 00h

7	6	5	4	3	2	1	0
CLRST	SMBARE	GCMEN	ACK	-	INTEN	STOP	START
З	З Ч	З Ч	З Ч	-	З Ч	З Ч	З Ч

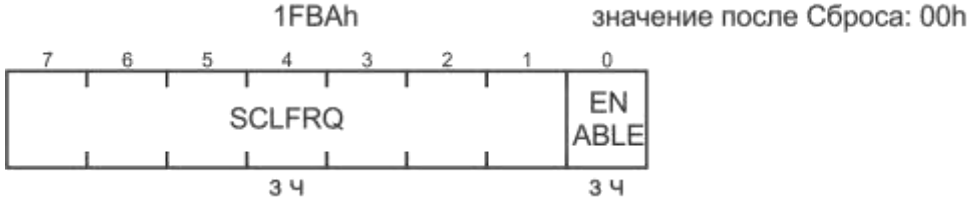
Поле	Бит	Описание	
1	2	3	
CLRST	7	Бит сброса флага прерывания INT. Запись «0» в бит CLR игнорируется. Запись «1» в бит CLR сбросит флаг INT в регистре SMBST. Чтение этого бита всегда возвращает «0»	
SMBARE	6	Бит управления реакцией на получение адреса отклика	
		0	Полученный адрес не проверяется на совпадение с адресом отклика
		1	Адрес, полученный сразу после старта, проверяется на совпадение с адресом отклика (0001_100b)
		Бит очищается при выходе ведомого из режима IDLE	
GCMEN	5	Бит управления реакцией на получение адреса общего вызова	
		0	Полученный адрес не проверяется на совпадение с адресом общего вызова
		1	Адрес, полученный сразу после старта, проверяется на совпадение с адресом общего вызова (0000_000b)
		Бит очищается при выходе ведомого из режима IDLE	
ACK	4	Бит квитирования приема. В режиме передатчика не используется. В режиме приемника (мастера/ведомого) содержит значение, которое передается в течение цикла отклика на запрос передатчика подтвердить прием. Передача нуля по окончании передачи байта (квитирование) означает, что данные успешно получены. Передача единицы (неквитирование) означает, что приемник не может продолжать работу по каким-либо причинам. Бит ACK очищается аппаратно по окончании цикла отклика	

Окончание таблицы А.56

1	2	3	
INTEN	2	Бит разрешения прерывания	
		0	Запрещено
		1	Разрешено
STOP	1	Бит останова. В режиме мастера установка бита STOP генерирует состояние останова, которое завершает или прерывает текущую передачу. После прекращения передачи бит STOP очищается аппаратно	
START	0	Бит старта. Этот бит устанавливается, когда требуется сформировать стартовое состояние на шине. Бит START очищается аппаратно по окончании цикла стартового состояния, а также при обнаружении ошибки на шине (состояние с кодом 1Fh)	
–	3	Зарезервировано	

Таблица А.57 – Регистр 2 управления I2C

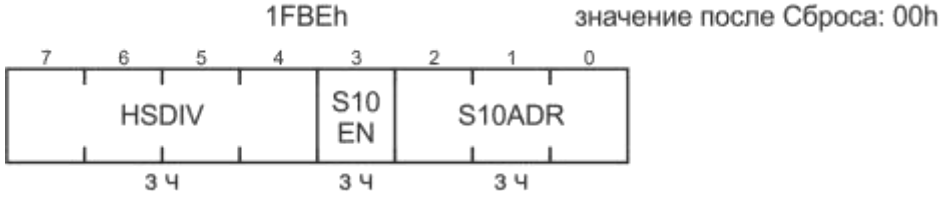
SMBCTRL2



Поле	Бит	Описание	
SCLFRQ	7–1	Поле выбора частоты f_{SCL} сигнала на выводе SCL в режиме мастера. Длительности высокого(T_{SCLH}) и низкого(T_{SCLL}) уровней сигнала SCL зависят от тактовой частоты fosc модуля I2C и рассчитываются по формуле: $T_{SCLH} = T_{SCLL} = 2 \times SCLFRQ \times (1/fosc)$. (A.3) Таким образом, частота сигнала на выводе SCL равна $f_{SCL} = 1/(T_{SCLH} + T_{SCLL})$. (A.4) В поле SCLFRQ можно записать любое значение в диапазоне от 04h до 7Fh. При попытке записи любого значения меньше «04h», оно будет записано со смещением 04h. Например, при записи числа 02h, к нему будет аппаратно добавлено смещение 04h и, в итоге, в поле SCLFRQ окажется значение «06h»	
ENABLE	0	Бит включения модуля I2C	
		0	Модуль выключен. Тактирование не осуществляется. Регистры SMBCTRL1, SMBST, SMBCST сброшены
		1	Модуль включен

Таблица А.58 – Регистр 3 управления I2C

SMBCTRL3



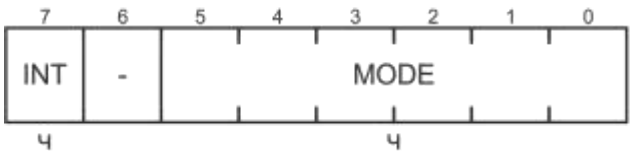
Поле	Бит	Описание	
HSDIV	7–4	Поле выбора частоты f_{SCL} сигнала на выводе SCL в режиме HS мастера. Длительности высокого(T_{HSCLH}) и низкого(T_{HSCLL}) уровней сигнала на выводе SCL зависят от тактовой частоты fosc модуля I2C и рассчитываются по формулам $T_{HSCLH} = HSDIV \times (1/fosc)$, (A.5) $T_{HSCLL} = 2 \times HSDIV \times (1/fosc)$. (A.6) Таким образом, частота сигнала на выводе SCL равна $f_{SCL} = 1/(T_{HSCLH} + T_{HSCLL})$. (A.7) В поле HSDIV можно записать любое значение в диапазоне от 2h до Fh. При попытке записи любого значения меньше «2h» в поле HSDIV, оно будет записано со смещением 2h. Например, при записи числа 1h к нему будет аппаратно добавлено смещение 2h и, в итоге, в поле SCLFRQ окажется значение «3h».	
S10EN	3	Бит разрешения 10-битной адресации ведомого	
		0	Запрещена
		1	Разрешена при условии, что установлен бит SAEN в регистре SMBADDR
S10ADR	2–0	Поле старших битов 10-битного адреса ведомого. Поле содержит старшие три разряда адреса ведомого при 10-битной адресации. Первый принятый байт адреса сравнивается со значением [11110b, S10ADR[2:1]], второй байт адреса – со значением [S10ADR[0], ADDR]	

Примечание – Квадратные скобки указывают на то, что заключенное в них число получается конкатенацией значений чисел, разделенных запятой; S10ADR[2:1] и S10ADR[0] – соответственно значения старших двух битов и младшего бита поля S10ADR; ADDR – значение поля регистра SMBADDR.

Таблица А.59 – Регистр состояния I2C

SMBST

1FB2h значение после Сброса: 00h



Поле	Бит	Описание
INT	7	Флаг прерывания. Устанавливается после 9-го такта сигнала SCL (когда SCL = 0) в любое запрограммированное время. Условия выставления флага INT: - во время приема/передачи как в режиме мастера, так и в режиме ведомого; - при совпадении адреса (адреса ведомого, адреса отклика или адреса общего вызова) содержимое регистра SMBSDA должно контролироваться программно для определения типа полученного адреса; - после успешного формирования стартового состояния или состояния повторного старта; - в случае неквитирования переданной информации; - при потере арбитража во время передачи последнего бита; - при обнаружении валидного состояния останова или состояния повторного старта; - при обнаружении ошибки на шине. Пока установлен флаг INT, на линии SCL удерживается низкий уровень сигнала. Флаг INT может быть сброшен установкой бита CLRST в регистре SMBCTRL1 или выключением модуля I2C (обнуление бита ENABLE в регистре SMBCTRL2) Условия выставления флага INT (не влияющие на уровень сигнала на линии SCL): - простой на линии SCL; - состояние останова в режиме ведомого (MODE = 1Ch); - потеря арбитража, вследствие чего ведомый переключился в безадресный режим (MODE = 03h или MODE = 23h); - неквитированная передача байта данных (MODE = 17h)
MODE	5–0	Код состояния. Возникновение того или иного состояния в течение функционирования модуля I2C сопровождается записью соответствующего кода в поле MODE.
–	6	Зарезервировано

Таблица А.60 – Регистр адреса I2C

SMBADDR

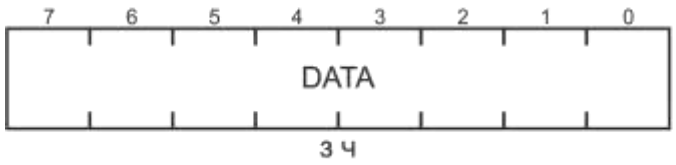
1FB8h значение после Сброса: 00h

Поле	Бит	Описание	
SAEN	7	Бит разрешения распознавания адреса	
		0	Безадресный режим
		1	Включена функция распознавания принятого адреса
ADDR	6–0	Поле собственного 7-битного адреса. При работе в режиме ведомого первые 7 бит, принятые после стартового состояния, сравниваются со значением ADDR. Если обнаружено совпадение и установлен бит SAEN, ведомый переходит в режим приемника или передатчика (в зависимости от состояния бита направления R/W#)	

Таблица А.61 – Регистр данных I2C

SMBSDA

1FB0h значение после Сброса: XXh

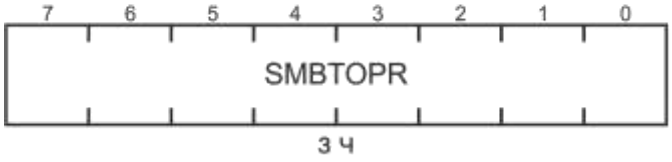


Поле	Бит	Описание
DATA	7–0	Поле данных

Таблица А.62 – Регистр загрузки предделителя

SMBTOPR

1FBCh значение после Сброса: 00h

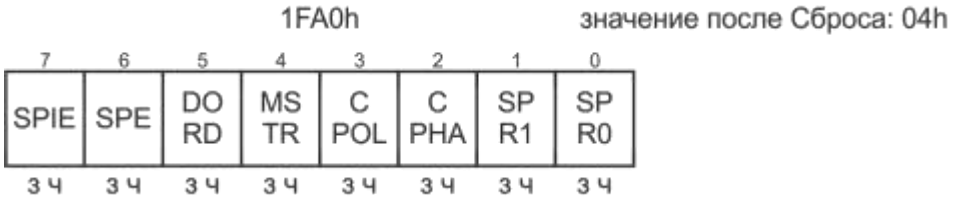


Поле	Бит	Описание
SMBTOPR	7–0	Поле значения перезагрузки предделителя

Регистры контроллера интерфейса SPI

Таблица А.63 – Регистр управления SPI

SPCR



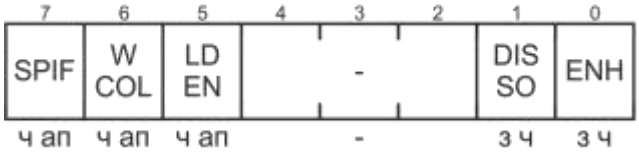
Поле	Бит	Описание	
SPIE	7	Бит разрешения запроса прерывания. Разрешает установку флага SPIF в регистре SPSR. Вместе с битом ES в регистре IE разрешает формирование запроса на прерывание от модуля SPI	
		0	Запрещено
		1	Разрешено
SPE	6	Бит разрешения работы модуля SPI	
		0	Выключен. Все выводы в высокоимпедансном состоянии
		1	Включен
DORD	5	Бит управления порядком передачи и приема данных	
		0	Старшим битом вперед
		1	Младшим битом вперед
MSTR	4	Бит конфигурации	
		0	Режим ведомого
		1	Режим мастера
CPOL	3	Бит полярности сигнала тактирования	
		0	В отсутствии передачи на выводе SCK удерживается сигнал низкого уровня
		1	В отсутствии передачи на выводе SCK удерживается сигнал высокого уровня
CPHA	2	Бит фазы тактового сигнала. Бит CPHA вместе с битом CPOL управляет соотношением фаз тактового сигнала и данных на линиях MOSI и MISO	
SPR1 SPR0	1, 0	Биты управления скоростью передачи данных	

SPR1	SPR0	Частота синхросигнала SCK	
		При SLOW = 0b	При SLOW = 1b
0	0	fspi/4	fspi/2
0	1	fspi/16	fspi/8
1	0	fspi/64	fspi/32
1	1	fspi/128	fspi/64

Таблица А.64 – Регистр состояния SPI

SPSR

1FA2h значение после Сброса: 00h



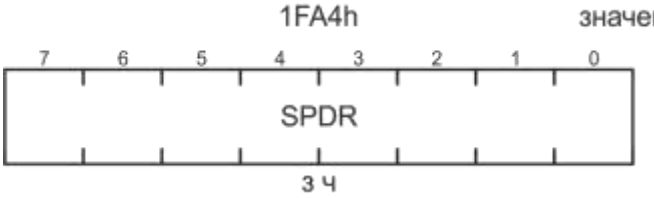
Поле	Бит	Описание
1	2	3
SPIF	7	Флаг прерывания модуля SPI. Устанавливается (если установлен бит SPIE) по окончании приема/ передачи каждого байта данных и является индикатором того, что данные получены, загружены в регистр SPDR и могут быть прочитаны. Одновременно с битом SPIF, если это разрешено битом SPIE регистра SPCR, формируется запрос на прерывание. В режиме буферизации данных (ENH = 1), если передается посылка из нескольких байт данных, флаг SPIF выставляется в конце передачи каждого байта, но запрос на прерывание формируется только один раз – по окончании передачи всей посылки. Для программного сброса флага следует сначала прочитать регистр SPSR. Только после этого чтение или запись регистра SPDR сбросит флаг SPIF. Аппаратно бит SPIF сбрасывается только при включении модуля SPI. При выключении модуля SPI значение бита SPIF не изменяется
WCOL	6	Флаг конфликта записи. Поведение флага WCOL зависит от режима работы модуля SPI, который устанавливается битом ENH. 1 В нормальном режиме (ENH = 0) флаг WCOL устанавливается при попытке записи данных в регистр SPDR во время текущей передачи. При этом передача не прерывается, а запись данных блокируется. В этом режиме флаг WCOL сбрасывается аппаратно, одновременно со сбросом флага SPIF. 2 В режим буферизации передаваемых данных (ENH = 1) флаг WCOL устанавливается при попытке записи данных в регистр SPDR во время текущей передачи. При этом передача не прерывается, а данные записываются в буфер данных. По окончании передачи логика проверяет состояние флага WCOL, и если он установлен, загружает байт из буфера данных в сдвиговый передающий регистр и продолжает передачу. Одновременно с этим флаг WCOL аппаратно сбрасывается
LDEN	5	Разрешение загрузки в буфер данных. 1 В нормальном режиме (ENH = 0) бит LDEN не изменяет своего значения и остается равным нулю. 2 В режиме буферизации данных (ENH = 1) бит LDEN установлен в течение передачи первых четырех бит каждого передаваемого байта данных и сброшен во время передачи следующих четырех бит. Фактически бит LDEN определяет промежуток времени в течение передачи, когда желательно записать байт, который будет передан следующим по окончании текущей передачи

Окончание таблицы А.64

1	2	3	
DISSO	1	Независимое запрещение передачи данных. 1 В режиме мастера значение бита DISSO игнорируется. 2 В режиме ведомого бит DISSO контролирует выход данных. Пока бит DISSO остается равным нулю, ведомый передает и принимает данные. Установка бита DISSO блокирует передачу данных и переводит вывод MISO в состояние слабой логической единицы, при этом прием тактового сигнала и прием данных не блокируются	
ENH	0	Включение режима буферизации передаваемых данных	
		0	Модуль SPI работает в нормальном режиме. Передачи осуществляются по одному байту. Каждый следующий байт данных может быть записан в регистр SPDR и передан только по окончании текущей передачи
		1	Модуль SPI работает в режиме буферизации передаваемых данных. Байт, записываемый в регистр SPDR во время текущей передачи, попадает в буфер данных и загружается аппаратно в передающий регистр сразу же по окончании текущей передачи. В случае, если модуль SPI функционирует как мастер, то формируемый сигнал тактирования SCK в течение времени передачи всех байт не прерывается и его период остается стабильным. Таким образом, организуется непрерывная передача неограниченного числа байт
–	4–2	Зарезервировано	

Таблица А.65 – Регистр данных SPI

SPDR



Поле	Бит	Описание
DATA	7–0	Поле данных

Регистры контроллера LIN

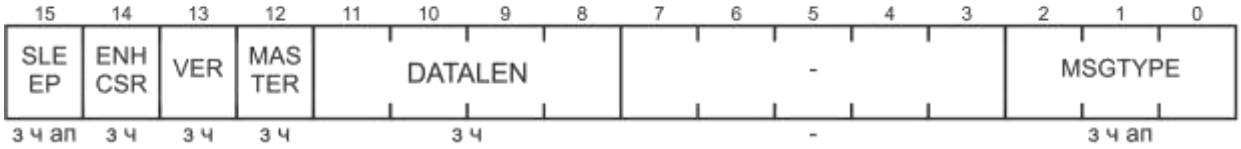
Таблица А.66 – Регистры данных

Мнемоническое обозначение	Название
LINDTX10	16-разрядные регистры данных для передачи 1F40h – 1F46h
LINDTX32	
LINDTX54	
LINDTX76	
LINDRX10	16-разрядные регистры принятых данных 1F48h – 1F4Eh
LINDRX32	
LINDRX54	
LINDRX76	

Таблица А.67 – Регистр управления LIN

LINCON

1F50h значение после Сброса: 0801h

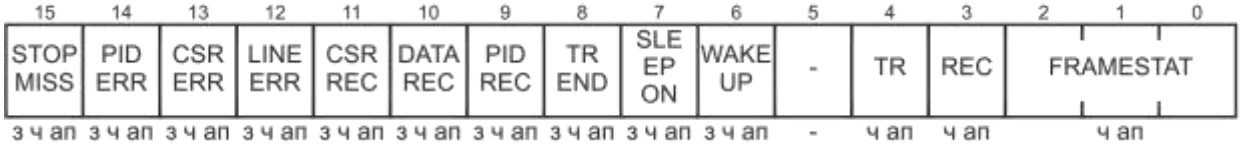


Поле	Бит	Описание	
SLEEP	15	Бит включения режима сна	
		0	Нормальный режим работы
		1	Включение режима сна
ENHCSR	14	Тип контрольной суммы	
		0	Обычная
		1	Расширенная
VER	13	Версия протокола	
		0	Версия 1.X
		1	Версия 2.X
MASTER	12	Бит типа устройства	
		0	Ведомый
		1	Мастер
DATALEN	11–8	Количество байт данных. Заполнять это поле необходимо только при использовании протокола версии 2.X. По умолчанию значение поля – 8h (восемь байт данных)	
MSGTYPE	2–0	Тип кадра сообщения	
		001	BUSWAIT (устройство находится в состоянии ожидания следующего кадра сообщения)
		010	HEADER
		011	HEADER+RESPONSE
		100	RESPONSE RECEIVE
		101	RESPONSE TRANSMIT
		111	WAKE
–	7–3	Зарезервировано	

Таблица А.68 – Регистр состояния LIN

LINSTAT

1F52h значение после Сброса: 0000h



Поле	Бит	Описание	
1	2	3	
STOPMISS	15	Флаг отсутствия стоп-бита. Устанавливается при отсутствии в поле стопового бита	
PIDERR	14	Флаг ошибки PID. Устанавливается при ошибке четности в PID	
CSRERR	13	Флаг ошибки контрольной суммы. Устанавливается, когда принятая сумма отличается от суммы, рассчитанной на основе принятых полей	
LINEERR	12	Флаг конфликта линии. Устанавливается, если при передаче рецессивного бита линия находится в доминантном состоянии	
CSRREC	11	Флаг получения контрольной суммы. Устанавливается при получении поля контрольной суммы	
DATAREC	10	Флаг получения данных. Устанавливается при получении поля данных	
PIDREC	9	Флаг получения PID. Устанавливается при получении PID	
TREND	8	Флаг окончания передачи. Устанавливается при отправке последнего символа текущей посылки	
SLEEPON	7	Флаг перехода в режим сна. Устанавливается, если принятый кадр сообщения является сигналом ко сну	
WAKEUP	6	Флаг выхода из режима сна. Устанавливается, когда во время сна принят сигнал пробуждения	
TR	4	Флаг активности передающей части интерфейса	
		0	Нет действий
		1	Устанавливается, когда происходит переход из IDLE в состояние передачи
REC	3	Бит активности приемной части интерфейса	
		0	Нет действий
		1	Устанавливается, когда происходит переход из IDLE в состояние приема. Тип кадра сообщения соответствует приему

Окончание таблицы А.68

1	2	3	
FRAMESTAT	2–0	Текущее состояние	
		001	Состояние IDLE – в этом состоянии может находиться только передатчик после окончания передачи текущего кадра сообщения)
		010	Состояние BREAK
		011	Состояние SYNC
		100	Состояние PID
		101	Состояние DATA
		110	Состояние CSUM
		111	Состояние посылки сигнала Wake
–	5	Зарезервировано	
Примечание – Все флаги маскируются битами регистра LININTEN и устанавливаются, если установлены соответствующие биты. Одновременно с установкой любого флага генерируется прерывание LININT. Все флаги (за исключением битов TR, REC и FRAMESTAT) должны сбрасываться программно записью единицы.			

Таблица А.69 – Регистр разрешения прерываний LIN

LININTEN

1F5Eh значение после Сброса: 0000h

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP MISS EN	PID ERR EN	CSR ERR EN	LINE ERR EN	CSR REC EN	DATA REC EN	PID REC EN	TR END EN	SLE EPON EN	WAKE UP EN	-	TR EN	REC EN	FRAMESTATEN		
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		

Биты 15–6, 4, 3 являются битами разрешения для соответствующих флагов регистра LINSTAT.

Поле FRAMESTATEN задает код режима, при входе в который будет генерироваться прерывание и выставляться соответствующий код в поле FRAMESTAT регистра LINSTAT

Таблица А.70 – Регистр настройки скорости передачи LIN

LINSPEED

1F56h значение после Сброса: 012Ch

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPEEDH								SPEEDL							
З Ч								З Ч							

SPEEDH и SPEEDL – старший и младший байты слова, которые задают скорость передачи блока LIN при функционировании в режиме мастера. Значение, которое записывается в регистр, вычисляется по формуле:

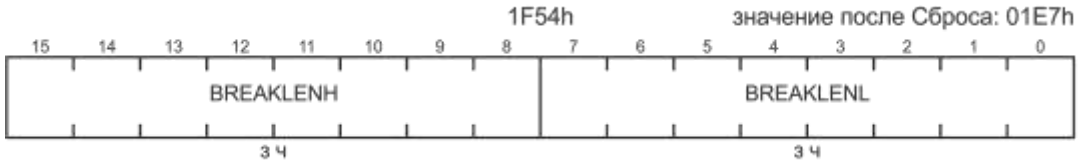
$$N = \frac{Fxtal}{4 \cdot Flin},$$

где Fxtal – тактовая частота микроконтроллера,
Flin – скорость передачи данных.

В режиме ведомого состояние регистра неважно.

Таблица А.71 – Регистр настройки длины поля BREAK

LINBREAK



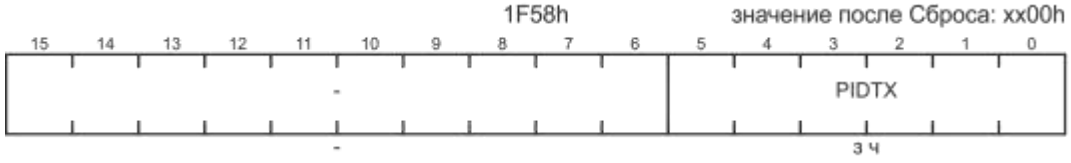
BREAKLENH и BREAKLENL – старший и младший байты слова, которое задает минимальную длину поля разрыва синхронизации, выраженную в периодах тактового сигнала. Длина вычисляется по формуле:

$$N = \frac{Fxtal}{2 \cdot Flin} \times \frac{Nbit}{16},$$

где Fxtal – тактовая частота микроконтроллера,
Flin – скорость передачи данных блоком LIN,
Nbit – минимальное количество бит в поле разрыва синхронизации

Таблица А.72 – Регистр передаваемого PID

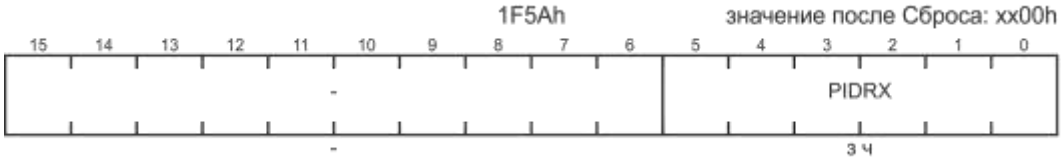
LINPIDTX



Поле	Бит	Описание
PIDTX	5–0	PID, который будет отправлен в следующем кадре данных
–	15–6	Зарезервировано

Таблица А.73 – Регистр принятого PID

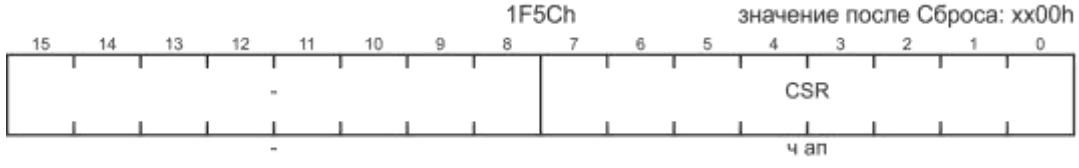
LINPIDRX



Поле	Бит	Описание
PIDRX	5–0	Принятый PID
–	15–6	Зарезервировано

Таблица А.74 – Регистр контрольной суммы LIN

LINCSR



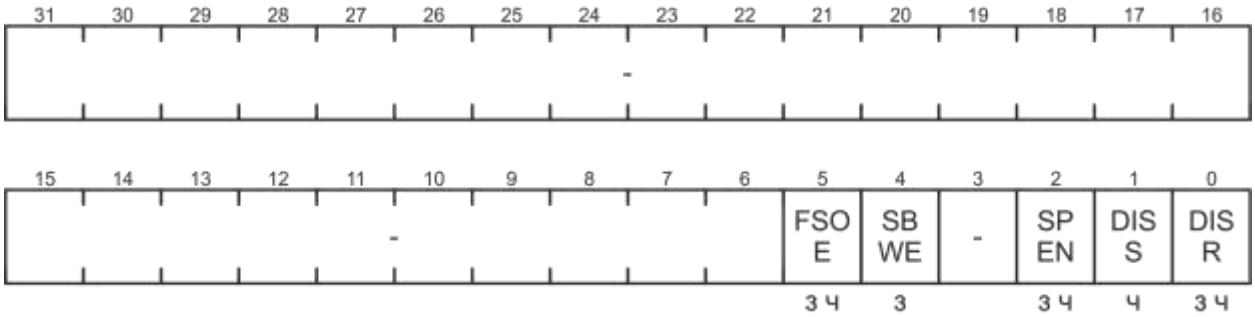
Поле	Бит	Описание
CSR	7–0	Значение принятой контрольной суммы
–	15–8	Зарезервировано

Регистры контроллера CAN

Контроллер CAN оперирует тремя группами регистров: регистры контроллера, регистры узлов и регистры объектов сообщений. Регистры контроллера представлены в таблицах А.75 – А.86. Регистры узлов с символом «x» в названии представлены в таблицах А.87 – А.93. Регистры объектов сообщений с символом «n» в названии представлены в таблицах А.95 – А.103.

Таблица А.75 – Регистр управления частотой

CLC 1A00h Сброс: 00000003h

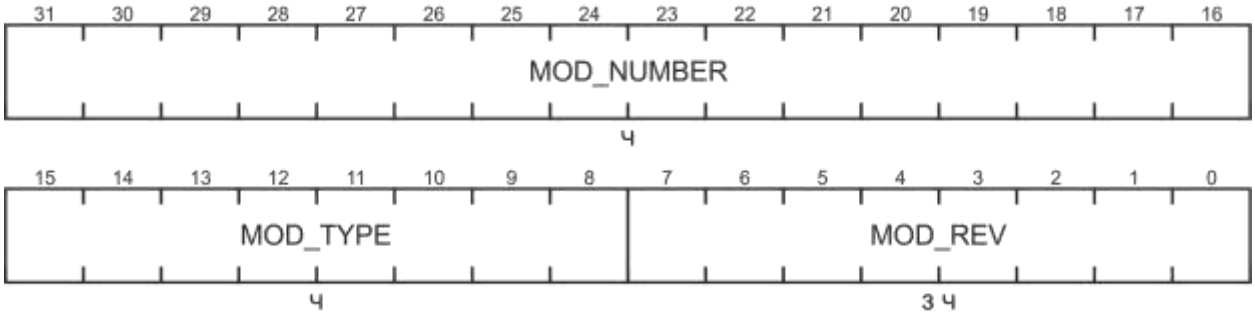


Поле	Бит	Описание	
FSOE	5	Бит активации быстрого выключения контроллера CAN для системы отладки. Не используется в данной версии микроконтроллера	
SBWE	4	Бит разрешения записи в биты FSOE и SPEN для системы отладки Не используется в данной версии микроконтроллера	
SPEN	2	Бит активации режима приостановки работы (Suspend) для системы отладки. Не используется в данной версии микроконтроллера	
DISS	1	Бит состояния контроллера CAN	
		0	Включен
		1	Выключен
DISR	0	Бит выключения контроллера CAN	
		0	Нет действий
		1	Запись единицы запускает механизм выключения
–	31–16, 15–6, 3	Зарезервировано	

Примечание – Когда контроллер CAN находится в выключенном состоянии, только регистр CLC доступен для записи и чтения, доступ к остальным регистрам невозможен.

Таблица А.76 – Регистр идентификации

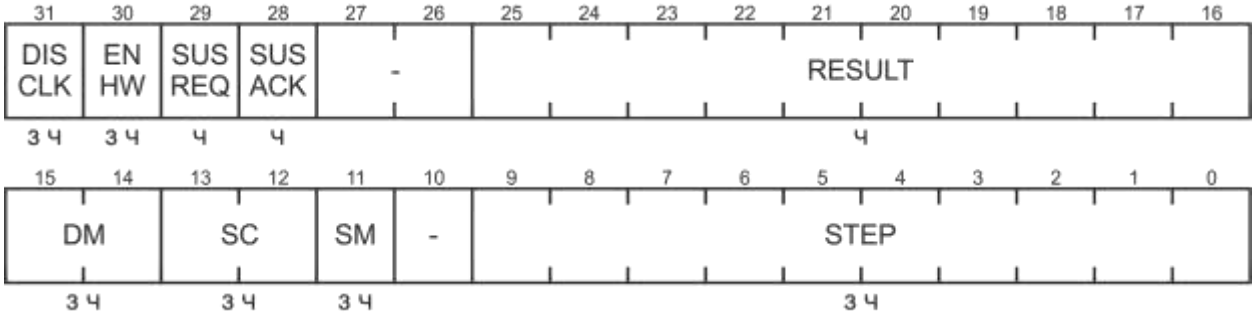
ID 1A08h Сброс: 002BC051h



Поле	Бит	Описание
MOD_NUMBER	31–16	Идентификационный номер контроллера CAN
MOD_TYPE	15–8	Разрядность контроллера CAN
MOD_REV	7–0	Число модификаций контроллера CAN

Таблица А.77 – Регистр делителя

FDR 1A0Ch Сброс: 00000000h



Поле	Бит	Описание	
1	2	3	
DISCLK	31	Бит запрета внутреннего тактирования	
		0	Генерирование сигнала Fcan разрешено
		1	Генерирование сигнала Fcan запрещено
ENHW	30	Бит контроля синхронизации. Этот бит аппаратно удерживается в сброшенном состоянии и не может быть установлен	
SUSREQ	29	Бит активации режима приостановки работы (Suspend). Не используется в данной версии микроконтроллера	
SUSACK	28	Индикатор режима Suspend. Не используется в данной версии микроконтроллера	
RESULT	25–16	Счетчик делителя частоты	
DM	15–14	Поле задания режима делителя частоты	
		00	Счетчик выключен. Синхросигнал Fout не генерируется. Сигнал сброса внешнего делителя в состоянии логической единицы
		01	Нормальный режим работы
		10	Режим дробного деления
		11	Счетчик выключен. Синхросигнал Fout не генерируется
SC	13–12	Поле задания конфигурации делителя частоты. Не используется в данной версии микроконтроллера	

Окончание таблицы А.77

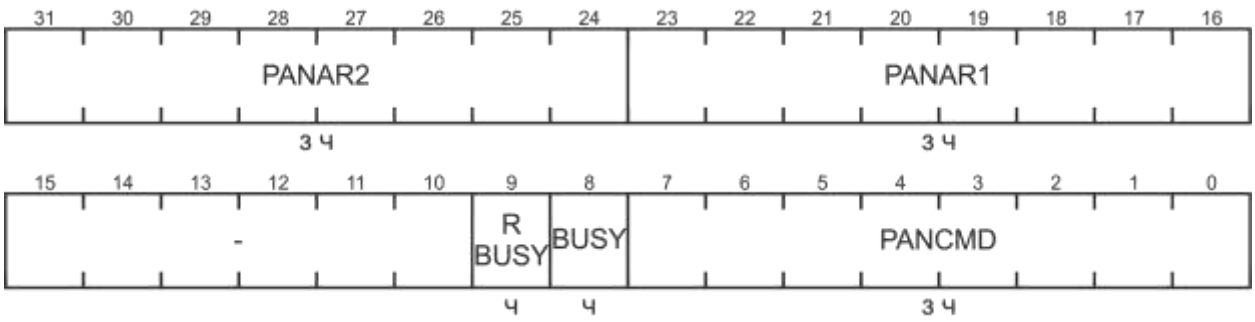
1	2	3
SM	11	Бит выбора режима перехода в режим Suspend. Не используется в данной версии микроконтроллера
STEP	9–0	Шаг делителя. Поле хранит значение, которое загружается в RESULT при переполнении счетчика делителя
–	27, 26, 10	Зарезервировано

Таблица А.78 – Варианты конфигурации делителя частоты

Поле DM	Состояние сигнала сброса внешнего делителя	Поле RESULT	Форми-рование сигнала Fout	Состояние/ режим делителя
00	1	Не меняется	Нет	Выключен
01	0	При активации режима загружается значением 3FFh. Далее периодически загружается значением из STEP	Да	Нормальный
10				Дробный
11		Не меняется	Нет	Выключен

Таблица А.79 – Регистр панели команд

PANCTR 1BC4h Сброс: 00000301h



Поле	Бит	Описание	
PANAR2	31–24	Панель аргумента 2 (см. таблицу А.80)	
PANAR1	23–16	Панель аргумента 1 (см. таблицу А.80)	
RBUSY	9	Флаг занятости панелей аргументов	
		0	Нет действий
		1	Выполняется команда списка, результат выполнения которой будет записан в PANAR1 и PANAR2
BUSY	8	Флаг занятости панелей аргументов	
		0	Панели готовы для записи
		1	Панели заняты – ожидают записи по окончании выполнения команды
PANCMD	7–0	Поле команды (см. таблицу А.80). После выполнения команды в это поле записывается 00h	
–	15–10	Зарезервировано	

Таблица А.80 – Коды команд работы со списками

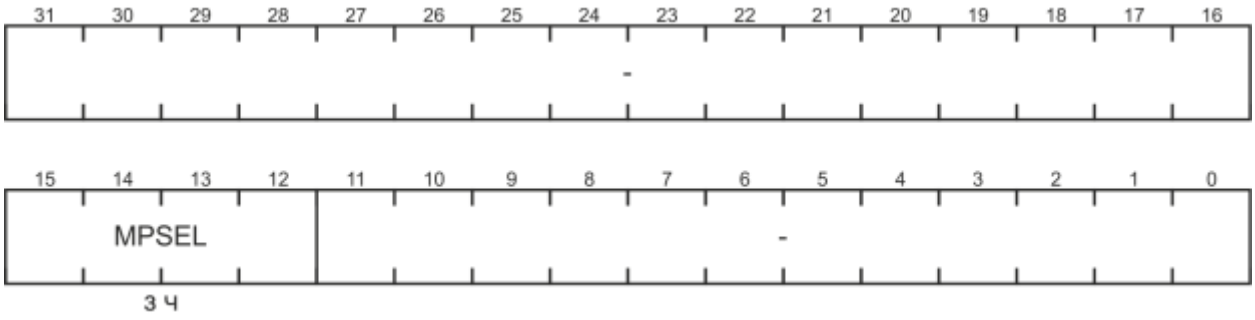
Код команды PAN CMD	Поле PANAR2	Поле PANAR1	Описание команды
1	2	3	4
00h	–	–	Нет операции. Никаких действий не выполняется
01h	Результат: бит 7 – ошибка, бит 6 – не определен	–	Инициализация списков. Запуск инициализации для очистки битовых полей CTRL и LIST всех объектов сообщений. Регистры LIST0 – LIST8 устанавливаются в свои значения после сброса. Это приводит к переносу всех объектов сообщений в список №0 (список нераспределенных объектов сообщений). Инициализация списков требует, чтобы биты INIT и CCE регистра NCR были установлены для обоих узлов. Бит 7 поля PANAR2 сигнализирует о результате операции: - 0 – Инициализация завершена успешно; - 1 – Инициализация не завершена, поскольку не все биты INIT и CCE были установлены. Команда инициализации списков автоматически запускается при каждом сбросе контроллера CAN, за исключением случая, когда все регистры объектов сообщений уже сброшены
02h	Аргумент: номер списка	Аргумент: номер объекта сообщения	Статическое занесение объекта сообщения в список. Объект сообщения переносится из текущего списка в список, указанный полем PANAR2 и добавляется в его конец. Эта команда также используется для дераспределения объекта сообщения, т.е. переноса его в список № 0 (если PANAR2 равно 00h)
03h	Аргумент: номер списка Результат: бит 7 – ошибка, бит 6– не определен	Результат: номер объекта сообщения	Динамическое занесение объекта сообщения в список. Первый объект сообщения списка №0 переносится в список, указанный полем PANAR2, и добавляется в его конец. Номер объекта сообщения возвращается полем PANAR1. Бит 7 поля PANAR2 сигнализирует о результате операции: - 0 – Операция выполнена; - 1 – Операция не выполнена – список №0 пуст
04h	Аргумент: номер объекта сообщения	Аргумент: текущий номер объекта сообщения	Перемещение по списку вверх. Перенос объекта сообщения с номером PANAR1 на одну позицию выше, чем расположен объект сообщения с номером PANAR2

Окончание таблицы А.80

1	2	3	4
05h	Аргумент: номер объекта сообщения Результат: бит 7 – ошибка, бит 6 – не определен	Результат: номер добавленного объекта сообщения	Динамическая вставка в список. Первый объект сообщения списка №0 вставляется на одну позицию выше, чем расположен объект сообщения с номером PANAR2. Номер добавленного объекта сообщения возвращается полем PANAR1. Бит 7 поля PANAR2 сигнализирует о результате операции: - 0 – Операция выполнена; - 1 – Операция не выполнена – список №0 пуст
06h	Аргумент: номер объекта сообщения	Аргумент: текущий номер объекта сообщения	Перемещение по списку вниз. Перенос объекта сообщения с номером PANAR1 на одну позицию ниже, чем расположен объект сообщения с номером PANAR2
07h	Аргумент: номер объекта сообщения Результат: бит 7– ошибка, бит 6 – не определен	Результат: номер добавленного объекта сообщения	Динамическая вставка в список. Первый объект сообщения списка №0 вставляется на одну позицию ниже, чем расположен объект сообщения с номером PANAR2. Номер добавленного объекта сообщения возвращается полем PANAR1. Бит 7 поля PANAR2 сигнализирует о результате операции: - 0 – Операция выполнена; - 1 – Операция не выполнена – список №0 пуст
08h – FFh	–	–	Зарезервировано

Таблица А.81 – Регистр управления

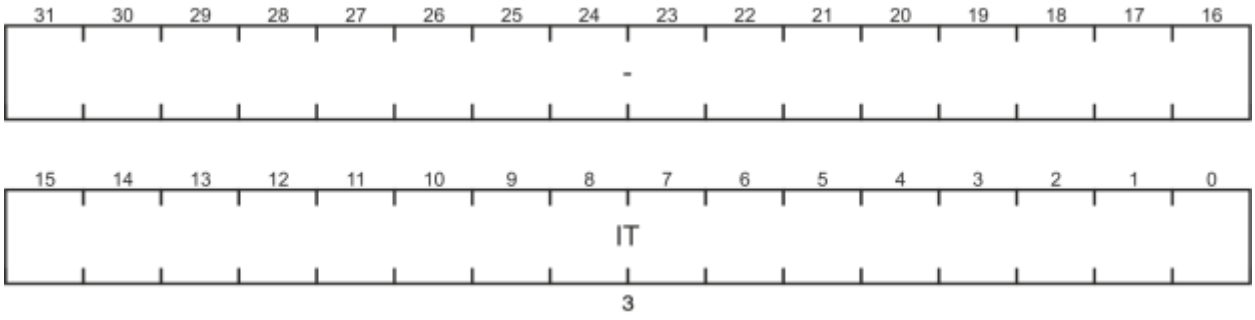
MCR 1BC8h Сброс: 00000000h



Поле	Бит	Описание
MPSEL	15–12	Поле задания позиции ждущего бита сообщения после приема/ передачи сообщения
–	31–16, 11–0	Зарезервировано

Таблица А.82 – Регистр прерываний

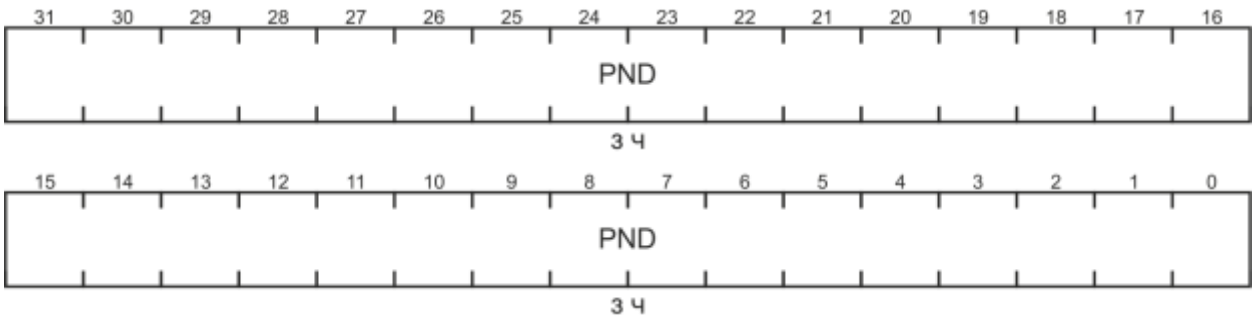
MITR 1BCCh Сброс: 00000000h



Поле	Бит	Описание
IT	15–0	Поле генератора прерываний. Каждый бит поля связан с одной из линий прерываний. Номера битов от 0 до 15 соответствуют номерам линий прерываний. Для того, чтобы сгенерировать одно или несколько прерываний, следует установить соответствующие биты. Установленные биты сбрасываются аппаратно
–	31–16	Зарезервировано

Таблица А.83 – Регистр ждущих прерываний

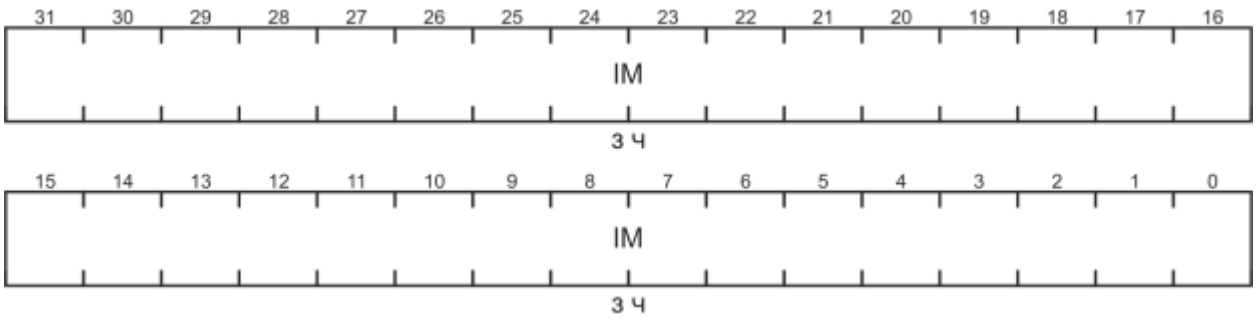
MSPND0 1B40h Сброс: 00000000h



Поле	Бит	Описание
PND	31–0	Поле ждущих битов сообщений. Каждому объекту сообщений выделяется два бита. Биты устанавливаются только аппаратно. Установленные биты сбрасываются аппаратно по окончании обслуживания запроса прерывания или могут быть сброшены в любой момент программно

Таблица А.84 – Регистр маски индекса сообщения

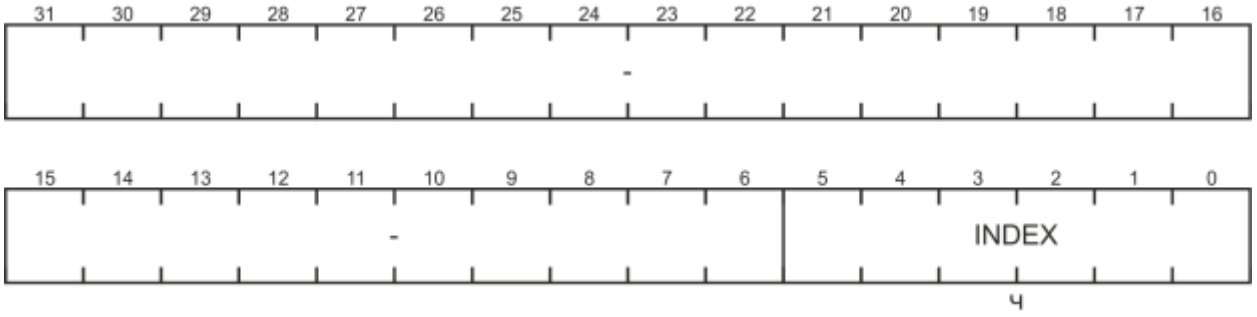
MSIMASK 1BC0h Сброс: 00000000h



Поле	Бит	Описание
IM	31–0	Маска для ждущих битов сообщений. Учитывается состояние только тех бит регистра MSPND, для которых в поле IM установлены соответствующие биты

Таблица А.85 – Регистр индекса сообщения

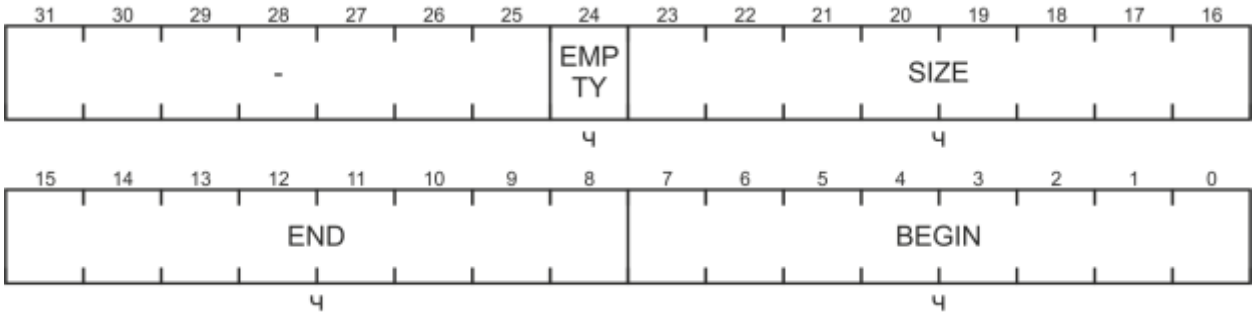
MSID0 1B80h Сброс: 00000020h



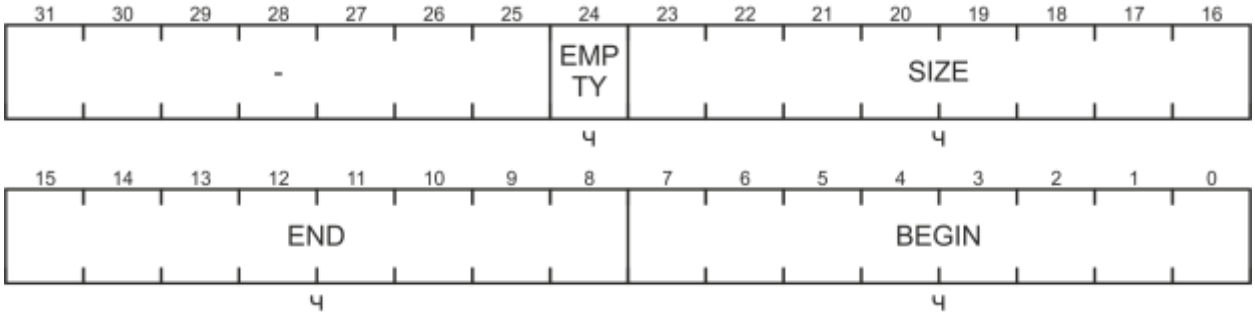
Поле	Бит	Описание
INDEX	5–0	Поле номера ждущего бита. Если в регистре MSPND есть установленные биты, которые не маскируются соответствующими битами регистра MSIMASK, то поле INDEX будет указывать на самый старший из них. Если в регистре MSPND нет установленных битов или они замаскированы, то в поле INDEX будет находиться значение 20h, указывающее на бит 31 регистра MSPND
–	31–6	Зарезервировано

Таблица А.86 – Регистр списка №0 и регистр свободного списка x

LIST0 1B00h Сброс: 007F7F00h



LISTx 1Baah Сброс: 01000000h

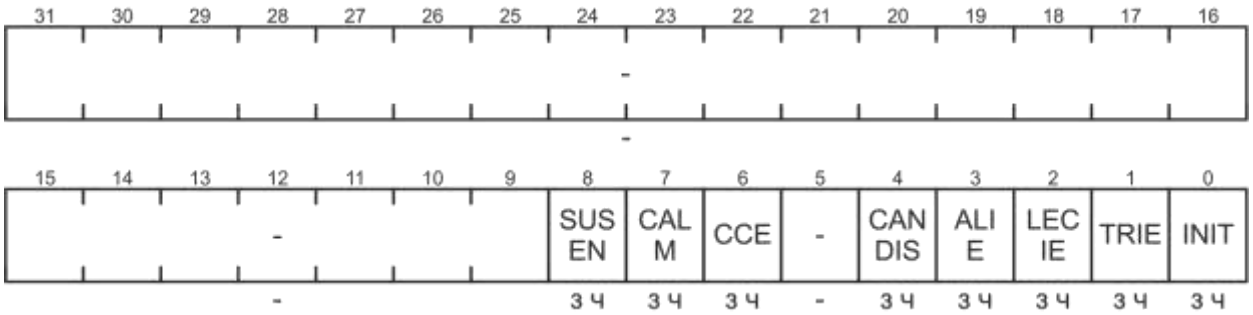


Поле	Бит	Описание	
EMPTY	24	Индикатор пустого списка	
		0	В списке есть как минимум один элемент
		1	Список пуст
SIZE	23–16	Размер списка. Количество элементов (объектов сообщений) в списке. Значение поля SIZE всегда на единицу меньше числа элементов. Если список пуст, SIZE = 00h	
END	15–8	Номер объекта сообщения, находящегося последним в списке. Поле может принимать значения от 00h до FFh, согласно количеству объектов сообщений (256)	
BEGIN	7–0	Номер объекта сообщения, находящегося первым в списке. Поле может принимать значения от 00h до FFh, согласно количеству объектов сообщений	
–	31–25	Зарезервировано	

Примечание – Приняты условные обозначения: x – порядковый номер списка от 1 до 7; aa – адрес регистра в таблице А.2.

Таблица А.87 – Регистр управления узла x

NCRx 1Caah Сброс: 00000001h



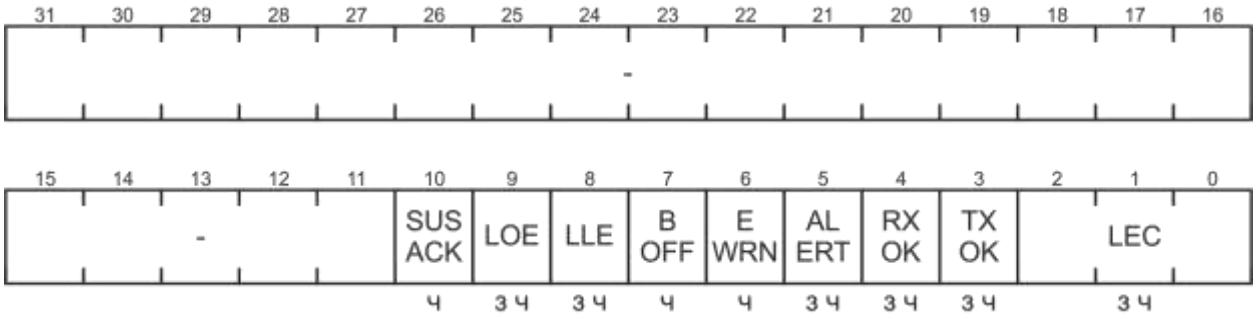
Поле	Бит	Описание	
1	2	3	
SUSEN	8	Бит разрешения режима приостановки работы узла. Не используется в данной версии микроконтроллера	
CALM	7	Бит включения режима анализа узла	
		0	Режим выключен
		1	Установка бита включает режим анализа узла. В этом режиме сообщения могут только приниматься, бит подтверждения не посылается после успешного приема сообщения, флаг активной ошибки посылается рецессивным вместо доминантного. На линии отправки сообщений поддерживается высокий уровень сигнала
		Бит может быть установлен только, если установлен бит INIT	
CCE	6	Бит разрешения изменения конфигурации узла. Управляет доступом к регистрам NBTRx, NPCRx и NECNTx	
		0	Только чтение
		1	Полный доступ
CANDIS	4	Бит выключения узла	
		0	Сброс бита включает узел
		1	Установка бита выключает узел. Сначала узел переходит в состояние «простоя» или «отключен от шины», далее аппаратно устанавливается бит INIT и, если разрешено, генерируется прерывание ALERT
ALIE	3	Бит разрешения прерывания ALERT от узла	
		0	Запрещено
		1	Разрешено
LECIE	2	Бит разрешения прерывания от узла при обнаружении кода последней ошибки	
		0	Запрещено
		1	Разрешено
TRIE	1	Бит разрешения прерывания от узла по окончании передачи/приема	
		0	Запрещено
		1	Разрешено

Окончание таблицы А.87

1	2	3	
INIT	0	Инициализация узла	
		0	Сброс бита разрешает участие узла в трафике CAN шины. Узел ожидает последовательность из 11 рецессивных бит на шине и включается в трафик. Если на момент сброса бита INIT узел находился в состоянии «отключен от шины», начинается процесс выхода из этого состояния в следующем порядке: получение 128 последовательностей бит (каждая из 11 рецессивных бит), выход из состояния «отключен от шины», включение в трафик
		1	Установка бита INIT прекращает участие узла в трафике. Все текущие передачи останавливаются, линии передач переходят в рецессивное состояние. Если на момент установки бита INIT узел находился в состоянии «отключен от шины», процесс выхода из этого состояния продолжается до его завершения. Далее узел остается неактивным до тех пор, пока установлен бит INIT
–	31–9, 5	Зарезервировано	
Примечание – Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.			

Таблица А.88 – Регистр состояния узла x

NSRx 1Caah Сброс: 00000000h



Поле	Бит	Описание	
1	2	3	
SUSACK	10	Индикатор перехода узла в режим приостановки. Не используется в данной версии микроконтроллера	
LOE	9	Флаг ошибки номера списка	
		0	Ошибок не обнаружено
		1	Обнаружена ошибка при фильтрации принимаемого сообщения. В регистре MOSTAT объекта сообщения обнаружен неверный номер списка
		Бит должен сбрасываться программно записью нуля	
LLE	8	Флаг ошибки списка	
		0	Ошибок не обнаружено
		1	Обнаружена ошибка при фильтрации принимаемого сообщения. Количество элементов списка, принадлежащего узлу, отличается от указанного в поле SIZE соответствующего регистра списка
		Бит должен сбрасываться программно записью нуля	

Продолжение таблицы А.88

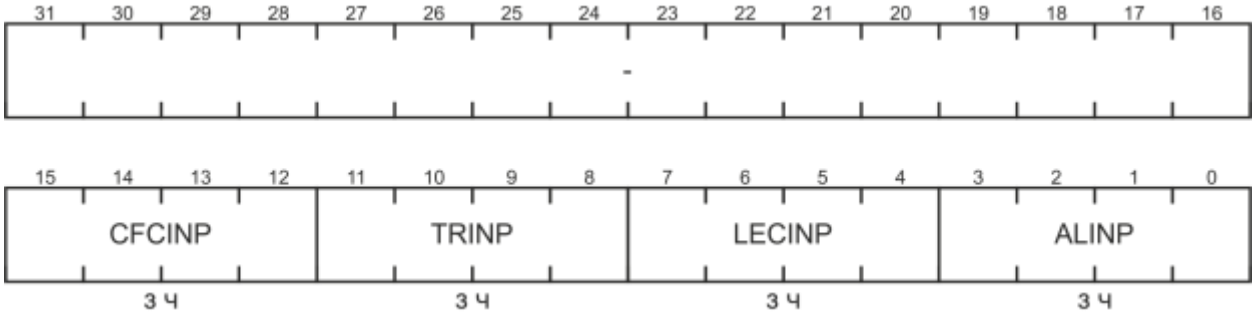
1	2	3	
BOFF	7	Флаг состояния «отключен от шины»	
		0	Узел не находится в состоянии «отключен от шины»
		1	Узел находится в состоянии «отключен от шины»
EWRN	6	Флаг критического количества ошибок	
		0	Лимит ошибок еще не достигнул
		1	По крайней мере, один из счетчиков ошибок (REC, TEC) достиг лимита ошибок, заданного полем EWRNLVL регистра NECNT узла
ALERT	5	Флаг предупреждения ALERT	
		0	Нет событий
		1	Произошло одно или несколько невзаимоисключающих событий: - модификация бита BOFF; - модификация/установка бита LOE; - установка бита LLE; - аппаратная установка бита INIT
		Бит должен сбрасываться программно записью нуля	
RXOK	4	Флаг успешного приема сообщения	
		0	Полученных сообщений нет
		1	Сообщение получено
		Бит должен сбрасываться программно записью нуля	
TXOK	3	Флаг успешной передачи сообщения	
		0	Переданных сообщений нет
		1	Сообщение передано без ошибок с получением подтверждения
		Бит должен сбрасываться программно записью нуля	
LEC	2–0	Код последней ошибки из обнаруженных ошибок работы узла	
		000	Ошибок нет
		001	Ошибка стаффинга (заполнения, STUFF ERROR). Может быть обнаружена во время передачи шестого бита из последовательности шести одинаковых бит в поле сообщения, которое должно быть кодировано методом разрядного заполнения (заключается в том, что после передачи пяти битов одинаковой полярности, шестой бит должен иметь противоположную полярность и вставляться передатчиком в поток данных автоматически, приемник пропускает этот бит)
		010	Ошибка формы (FORM ERROR). Обнаруживается, если: - в битовом поле фиксированного формата содержится количество битов, отличающееся от установленного; - на месте рецессивного бита находятся доминантный или наоборот. Исключение – для приемника доминантный бит в течение последнего бита поля «конец кадра» не интерпретируется как ошибка формы
		011	Ошибка подтверждения (ACKNOWLEGMENT ERROR). Обнаруживается передатчиком всякий раз, когда он не обнаруживает доминантный бит АСК в «области подтверждения»
		100	Разрядная ошибка или ошибка бита 1 (BIT 1 ERROR). Узел, который передает данные на шину, осуществляет мониторинг шины. Ошибка бита 1 имеет место, если при передаче рецессивного «1» бита (за исключением битов полей арбитража и подтверждения) на шине обнаруживается доминантный «0» бит

Окончание таблицы А.88

1	2	3	
LEC	2–0	101	Разрядная ошибка или ошибка бита 0 (BIT 0 ERROR). Ошибка возникает в случаях: - во время передачи сообщения (или бита подтверждения, флага активной ошибки, флага перезагрузки), узел передает доминантный бит «0», но на шине обнаруживается рецессивный «1»; - во время выхода из состояния «отключен от шины» при каждом обнаружении последовательности из 11 рецессивных битов. В этом случае, ЦП может использовать код 101 для отслеживания длительного простоя шины
		110	Ошибка циклического избыточного кода (CRC ERROR). Передатчик по установленному алгоритму вычисляет значение контрольной суммы (CRC) для передаваемых данных и вставляет ее в сообщение. Приемник, после получения данных, вычисляет CRC по тому же алгоритму, что и передатчик, и сравнивает вычисленное значение с принятым значением. В случае не совпадения фиксируется ошибка
		111	Код разрешения аппаратной записи в поле LEC
		После аппаратной записи в поле LEC значения кода, отличного от 111b, поле становится закрытым для записи, и далее центральный процессор не может изменить его состояние до тех пор, пока в это поле не будет программно записано значение 111b	
–	31–11	Зарезервировано	
Примечание – Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.			

Таблица А.89 – Регистр указателя прерываний узла x

NIPRx 1Caah Сброс: 00000000h



Поле	Бит	Описание
CFCINP	15–12	Указатель линии прерывания для прерывания при переполнении счетчика фреймов узла
TRINP	11–8	Указатель линии прерывания для прерывания по окончании передачи/приема сообщения
LECINP	7–4	Указатель линии прерывания для прерывания при записи кода последней ошибки
ALINP	3–0	Указатель линии прерывания для прерывания ALERT
–	31–16	Зарезервировано

Окончание таблицы А.89

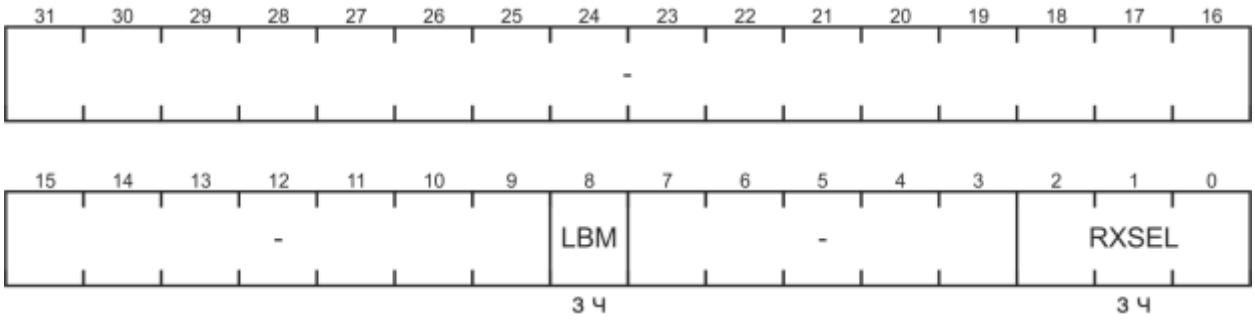
Примечания

1 Каждый из указателей позволяет задать номер одной из трех линий прерываний для каждого из четырех источников. Значение 00h соответствует нулевой линии прерываний, значение 01h – первой, 02h – третьей.

2 Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.

Таблица А.90 – Регистр управления портом узла x

NPCRx 1Caah Сброс: 00000000h

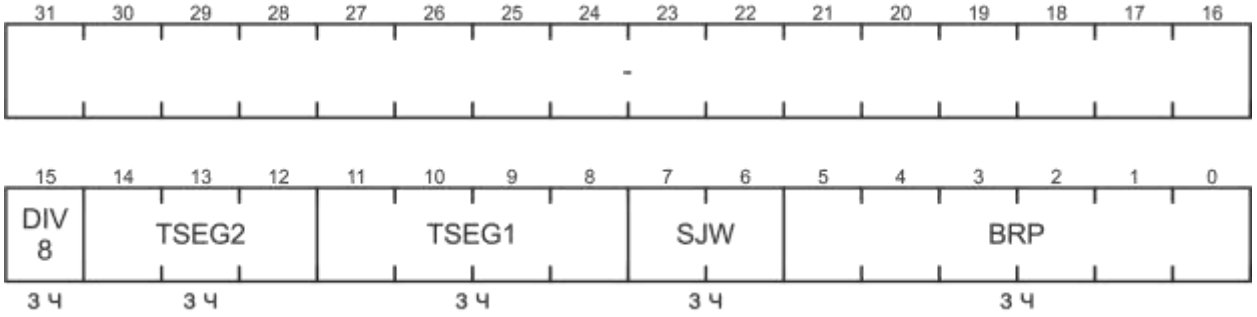


Поле	Бит	Описание	
LBM	8	Бит включения режима обратной петли (Loop-Back)	
		0	Режим выключен
		1	Включен режим обратной петли. В этом режиме узел подсоединяется к внутренней виртуальной CAN шине. Если для обоих узлов включен режим обратной петли, то они объединяются виртуальной CAN шиной и могут взаимодействовать друг с другом. При этом на внешних выводах узлов, соединенных с внешней физической CAN шиной, поддерживается рецессивный уровень сигнала, т.е. узлы не активны
RXSEL	2–0	Поле выбора вывода микроконтроллера для приема сообщений. В данной версии микроконтроллера значние поля не важно.	
–	31-9, 7-3	Зарезервировано	

Примечание – Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.

Таблица А.91 – Регистр синхронизации битов узла x

NBTRx 1Caah Сброс: 00000000h

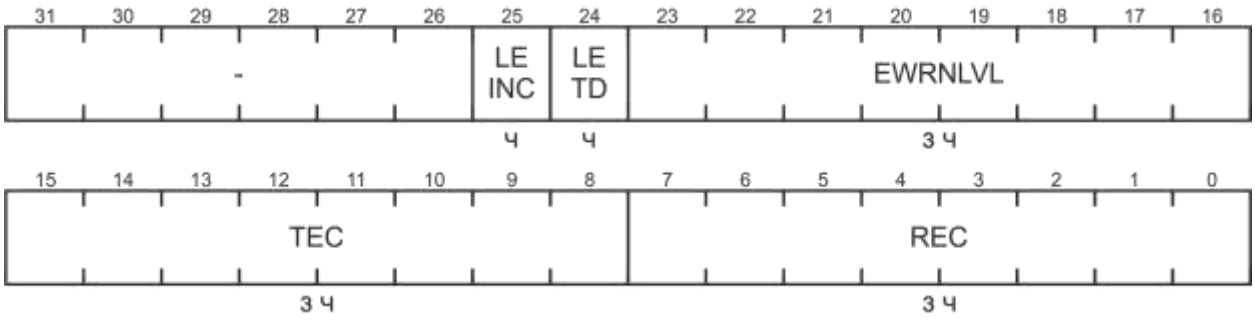


Поле	Бит	Описание	
DIV8	15	Делитель частоты на восемь	
		0	Длительность кванта времени (BRP + 1), тактов частоты
		1	Длительность кванта времени 8 × (BRP + 1), тактов частоты
TSEG2	14–12	Параметр 2. Временной промежуток от точки выборки до точки передачи, определяемый пользователем. Длительность сегмента равна tq × (TSEG2 + 1) и может быть уменьшена за счет ресинхронизации. Допустимые значения для TSEG1: от 01h до 07h	
TSEG1	11–8	Параметр 1. Временной промежуток от сегмента синхронизации до точки выборки, определяемый пользователем и включающий в себя сегмент распространения. Длительность равна tq × (TSEG1 + 1) и может быть увеличена за счет ресинхронизации. Допустимые значения для TSEG1: от 02h до 0Fh	
SJW	7–6	Ширина перехода ресинхронизации. Длительность равна tq × (SJW + 1)	
BRP	5–0	Предделитель скорости передачи. Если DIV8 = 0b, тогда длительность одного кванта времени равна (BRP + 1) тактам частоты. Если DIV8 = 1b, тогда длительность одного кванта времени равна 8×(BRP + 1) тактам частоты	
–	31–16	Зарезервировано	

Примечание – Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.

Таблица А.92 – Регистр счетчика ошибок узла x

NECNTx 1Caah Сброс: 00600000h

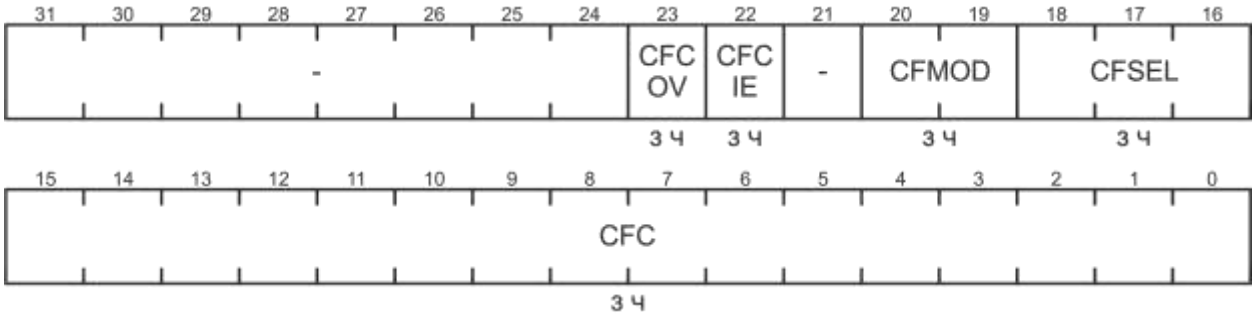


Поле	Бит	Описание	
LEINC	25	Индикатор инкрементирования при последней ошибке	
		0	Обнаруженная ошибка приводит к инкрементированию счетчика ошибок на единицу
		1	Обнаруженная ошибка приводит к инкрементированию счетчика ошибок на восемь
LETD	24	Флаг последней ошибки передачи	
		0	При приеме сообщения обнаружена ошибка, и произошло инкрементирование поля REC
		1	При передаче сообщения обнаружена ошибка, и произошло инкрементирование поля TEC
EWRNLVL	23–16	Поле задания лимита ошибок, по достижении которого выставляется флаг EWRN в регистре NSR (по умолчанию, количество ошибок – 96)	
TEC	15–8	Поле счетчика ошибок передачи сообщений	
REC	7–0	Поле счетчика ошибок приема сообщений	
–	31–26	Зарезервировано	

Примечание – Приняты условные обозначения: x – порядковый номер узла 0, 1; aa – адрес регистра в таблице А.2.

Таблица А.93 – Регистр счетчика сообщений узла x

NFCRx 1Caah Сброс: 00000000h



Поле	Бит	Описание	
CFCOV	23	Флаг переполнения счетчика сообщений	
		0	Счетчик не переполнен
		1	Счетчик переполнился. В режиме синхросчетчика этот флаг устанавливается при изменении поля CFC и, если установлен бит CFCIE, формируется прерывание
		Бит сбрасывается программно	
CFCIE	22	Бит разрешения прерывания от счетчика сообщений	
		0	Запрещено
		1	Разрешено
CFMOD	20–19	Поле задания режима работы счетчика сообщений	
		00	Счетчик сообщений. Инкрементируется после каждого успешного приема/передачи сообщения
		01–11	Зарезервировано. Не использовать!
CFSEL	18–16	Поле задания параметров выбранного режима счетчика сообщений (см. таблицу А.94)	
CFC	15–0	Поле счетчика сообщений. Хранит значение счетчика сообщений при CFMOD = 00b	
–	31–24, 21	Зарезервировано	

Примечание – Приняты условные обозначения: x – номер узла 0, 1; aa – адрес регистра в таблице А.2.

Таблица А.94 – Коды задания параметров режима счетчика сообщений

Счетчик сообщений (при CFMOD = 00b)	
Код в поле CFSEL	Действия
* * 1	Счетчик инкрементируется каждый раз при получении сообщения, не имеющего объекта сообщения
* 1 *	Счетчик инкрементируется каждый раз при получении сообщения, имеющего соответствующий объект сообщения
1 * *	Счетчик инкрементируется каждый раз при успешной отправке сообщения

Примечание – «*» указывает на то, что состояние этого бита поля CFSEL не важно для включения параметра режима. Все три параметра могут комбинироваться между собой (например, 110b или 101b).

Регистры объектов сообщений

Каждый из 16 объектов сообщений имеет набор из восьми регистров: MOARn, MODATAHn, MODATALn, MOAMRn, MOIPRn, MOFGPRn, MOFCRn и MOCTRn/MOSTATn (два регистра, обращение к которым производится по одному адресу – при записи данные попадают в MOCTRn, а при чтении возвращается значение из MOSTATn). Все наборы регистров идентичны, n – является номером объекта сообщения от 0 до 15. Все регистры имеют уникальные адреса. В таблице А.3 приведены адреса регистров, в зависимости от номера n объекта сообщения, которому они принадлежат.

Таблица А.95 – Регистр управления объектом сообщения n

MOCTRn

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
-				SET DIR	SET TXEN 1	SET TXEN 0	SET TXRQ	SET RXEN	SET RT SEL	SET MSG VAL	SET MSG LST	SET NEW DAT	SET RXUP D	SET TXPN D	SET RXPN D
				3	3	3	3	3	3	3	3	3	3	3	3

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-				RES DIR	RES TXEN 1	RES TXEN 0	RES TXRQ	RES RXEN	RES RT SEL	RES MSG VAL	RES MSG LST	RES NEW DAT	RES RXUP D	RES TXPN D	RES RXPN D
				3	3	3	3	3	3	3	3	3	3	3	3

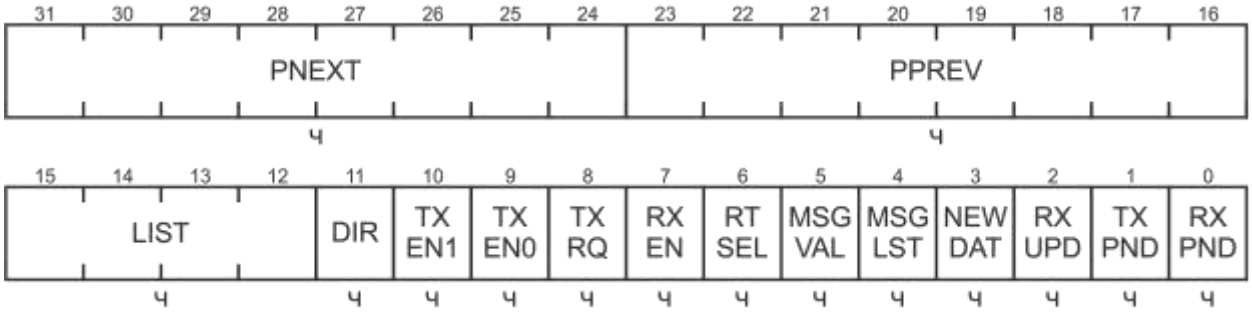
Биты регистра работают попарно. Комбинация состояний бит каждой пары оказывает влияние на один (соответствующий этой паре) бит регистра MOSTATn того же объекта сообщения. Так, пара SETDIR-RESDIR устанавливает и сбрасывает бит DIR, пара SETTXN1-RESTXN1 устанавливает и сбрасывает бит TXEN1 и т.д.

После записи старшего или младшего слова регистра MOCTRn аппаратная часть проверяет состояние бит каждой пары и, в зависимости от обнаруженной комбинации, выполняет соответствующее действие

Бит SET***	Бит RES***	Действие над битом ***
0	0	Нет
1	1	
1	0	Установка
0	1	Сброс
Биты 31–28 и 15–12 являются зарезервированными		

Таблица А.96 – Регистр состояния объекта сообщения n

MOSTATn



Поле	Бит	Описание	
1	2	3	
PNEXT	31–24	Указатель на следующий элемент списка. В поле находится номер объекта сообщения, расположенного выше по списку относительно текущего	
PPREV	23–16	Указатель на предыдущий элемент списка. В поле находится номер объекта сообщения, расположенного ниже по списку относительно текущего	
LIST	15–12	Номер списка, которому принадлежит объект сообщения n. Поле обновляется аппаратно при распределении/перераспределении объекта сообщения.	
DIR	11	Бит распределения	
		0	Объект приема. При установленном TXRQ формируется фрейм удаленного запроса с идентификатором объекта сообщения n и передается. Полученный в ответ фрейм данных с соответствующим идентификатором сохраняется в объекте сообщения n
		1	Объект передачи. При получении фрейма удаленного запроса с соответствующим идентификатором устанавливается флаг TXRQ, после чего в ответ передается фрейм данных, расположенный в объекте сообщения n. Передача фрейма данных может быть инициирована программной установкой бита TXRQ
TXEN1	10	Бит разрешения передачи фрейма	
		0	Запрещено
		1	Передача фрейма разрешена. Объект сообщения n может участвовать в передаче только, если установлены оба бита – TXEN1 и TXEN0. Контроллер CAN использует бит TXEN1 для выбора активного объекта передачи сообщения из FIFO
TXEN0	9	Бит разрешения передачи фрейма	
		0	Запрещено
		1	Передача фрейма разрешена. Объект сообщения n может участвовать в передаче только если установлены оба бита – TXEN0 и TXEN1. Контроллер CAN использует бит TXEN1 для выбора активного объекта передачи сообщения из FIFO. Можно программно очищать бит TXEN0 для запрета передачи сообщения, которое в настоящий момент формируется, или для запрета автоматической передачи в ответ на удаленный запрос

Продолжение таблицы А.96

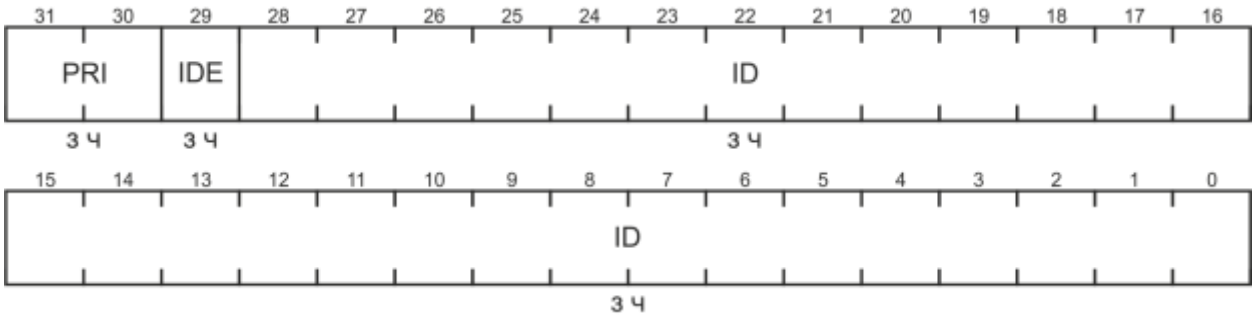
1	2	3	
TXRQ	8	Бит инициации передачи	
		0	Нет действий
		1	Установка бита инициирует передачу фрейма из объекта сообщения n. Инициация передачи фрейма возможна только в случае, если установлены биты TXRQ, TXEN0, TXEN1 и MSGVAL. Также бит TXRQ устанавливается аппаратно при получении фрейма удаленного запроса. Бит сбрасывается аппаратно при успешном завершении передачи и если при этом не был повторно программно установлен бит NEWDAT
RXEN	7	Бит разрешения приема	
		0	Запрещено
		1	Объект сообщения может принимать сообщения
		Состояние бита учитывается только при фильтрации принимаемых сообщений	
RTSEL	6	Индикатор возможности приема/передачи	
		0	Объект сообщения не может принимать/передавать сообщения
		1	Объект сообщения может принимать/передавать сообщения
		Прием фрейма. Бит RTSEL устанавливается аппаратно после того, как выбран объект сообщения n для сохранения только что принятого фрейма. Прежде, чем записать принятые данные в объект сообщения n, аппаратная часть проверяет состояние бита RTSEL. ЦПУ может сбрасывать этот бит, чтобы запретить запись принятого фрейма в объект сообщения n. Передача фрейма. Бит RTSEL устанавливается аппаратно после того, как выбран следующий объект сообщения n для передачи фрейма. Аппаратная часть перед началом передачи проверяет: установлен ли бит RTSEL и сброшен ли бит NEWDAT. Бит RTSEL должен оставаться установленным до окончания передачи. Проверка состояния бита RTSEL производится только при попытке изменения содержимого объекта сообщения n во избежание одновременного выполнения операций передачи фрейма и его изменения. Бит не участвует в фильтрации сообщений, и не сбрасывается аппаратно	
MSGVAL	5	Бит активности объекта сообщения n	
		0	Неактивен
		1	Активен
		Только те объекты сообщений, для которых установлен этот бит, могут использоваться для операций приема и передачи	
MSGLST	4	Бит потери сообщения	
		0	Ни одно сообщение не потеряно
		1	Принятое сообщение потеряно вследствие того, что контроллер CAN попытался установить бит NEWDAT по окончании приема сообщения при том, что флаг NEWDAT уже был установлен ранее после записи другого сообщения

Окончание таблицы А.96

1	2	3	
NEWDAT	3	Индикатор новых данных	
		0	С момента сброса бита NEWDAT никаких изменений объекта сообщений n не обнаружено
		1	Объект сообщения был изменен. Бит устанавливается аппаратно после того, как принятое сообщение было сохранено в объекте сообщения n. Бит сбрасывается аппаратно после начала передачи объекта сообщения n. Бит NEWDAT следует устанавливать программно после того, как новые данные для передачи будут сохранены в объекте сообщения n для предотвращения автоматического сброса бита TRXQ в конце текущей передачи
RXUPD	2	Индикатор изменений	
		0	Нет текущих изменений
		1	Идентификатор сообщения, поле длины данных DLC и данные в объекте сообщения изменяются
TXPND	1	Индикатор окончания передачи	
		0	Переданных сообщений нет
		1	Сообщение объекта сообщения n было успешно передано
RXPND	0	Индикатор окончания приема	
		0	Принятых сообщений нет
		1	Сообщение было успешно принято объектом сообщения n (напрямую или через шлюз). Бит должен сбрасываться программно

Таблица А.97 – Регистр арбитража объекта сообщения n

MOARn



Поле	Бит	Описание	
1	2	3	
PRI	31–30	Класс приоритета. Поле определяет один из четырех классов (0, 1, 2 и 3) приоритета объекта сообщения n. Нулевой класс устанавливает наивысший приоритет. Объекты сообщений с нулевым классом всегда выигрывают арбитраж при передаче и приеме сообщений. Фильтрация сообщений на основе идентификатора (маскируемого) и позиции в списке организуются только для объектов сообщений с равным приоритетом. Кроме этого, поле PRI определяет метод фильтрации	
		00	Зарезервировано

Окончание таблицы А.97

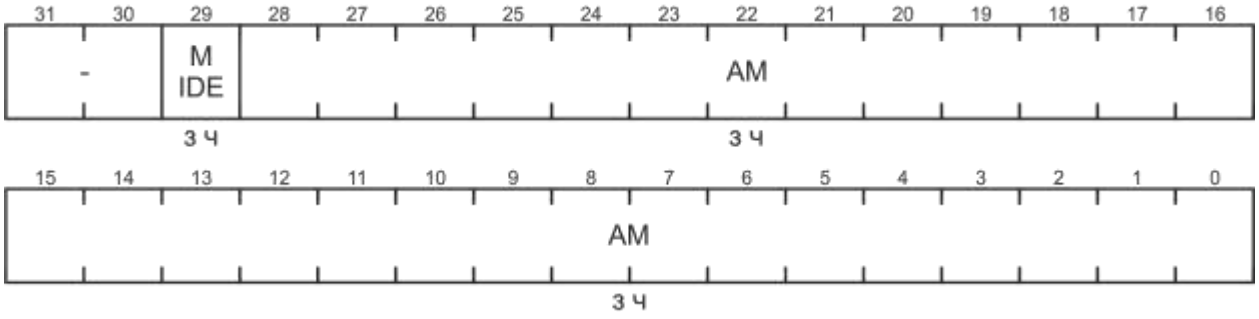
1	2	3	
PRI	31–30	01	Фильтрация в зависимости от положения объекта сообщения в списке. Объект сообщения n получает приоритет на передачу сообщения только в случае, если нет других объектов сообщений с установленными битами MSGVAL, TXEN0 и TXEN1, стоящих выше по списку
		10	Фильтрация в зависимости от значения идентификатора. Объект сообщения n получает приоритет на передачу сообщения только в случае, если в списке нет других объектов сообщений с «Идентификатор + IDE + DIR» более высокого приоритета (согласно правилам арбитража)
		11	Фильтрация в зависимости от положения объекта сообщения в списке (как при PRI = 01b)
IDE	29	Бит расширения идентификатора объекта сообщения n	
		0	Объект сообщения n оперирует с фреймами со стандартным 11-битным идентификатором
		1	Объект сообщения n оперирует с фреймами с расширенным 29-битным идентификатором
ID	28–0	Идентификатор объекта сообщения n. При оперировании с расширенными фреймами используются биты 28–0. При оперировании со стандартными фреймами используются биты 28–18, при этом состояние битов 17–0 не важно	

Таблица А.98 – Распределение приоритета между объектами сообщений согласно правилам арбитража

Установки для объектов сообщений 0 и 1, которые участвуют в арбитраже (приоритет объекта 0 выше приоритета объекта 1)	Пояснение
MOAR0[28:18] < MOAR1[28:18] (11-битный стандартный идентификатор объекта 0 меньше по числовому значению, чем 11-битный идентификатор объекта 1)	Стандартный фрейм с идентификатором, имеющим меньшее значение, обладает более высоким приоритетом
MOAR0[28:18] = MOAR1[28:18]. В регистре MOAR0 бит IDE = 0. В регистре MOAR1 бит IDE = 1.	При равенстве значений стандартных идентификаторов, стандартный фрейм имеет приоритет перед расширенным
MOAR0[28:18] = MOAR1[28:18]. Биты IDE обоих объектов сброшены. В регистре MOSTAT0 бит DIR = 1. В регистре MOSTAT1 бит DIR = 0.	При равенстве значений идентификаторов стандартный фрейм данных имеет приоритет перед стандартным фреймом удаленного запроса
MOAR0[28:0] = MOAR1[28:0] Биты IDE обоих объектов установлены. В регистре MOSTAT0 бит DIR = 1. В регистре MOSTAT1 бит DIR = 0.	При равенстве значений идентификаторов расширенный фрейм данных имеет приоритет перед расширенным фреймом удаленного запроса
MOAR0[28:0] < MOAR1[28:0] Биты IDE обоих объектов установлены. (29-битный идентификатор объекта 0 меньше по числовому значению, чем 29-битный идентификатор объекта 1)	Расширенный фрейм с идентификатором, имеющим меньшее значение, обладает более высоким приоритетом

Таблица А.99 – Регистр маски объекта сообщения n

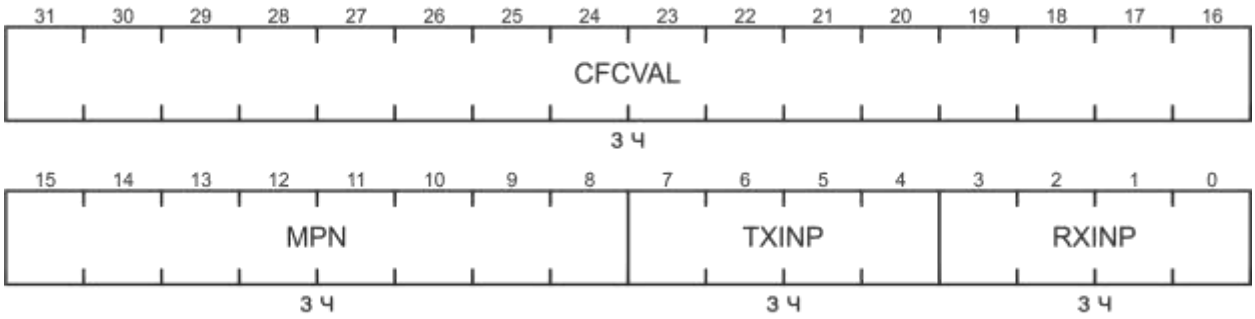
MOAMRn



Поле	Бит	Описание	
MIDE	29	Маска бита IDE сообщения	
		0	Объект сообщения n может принимать как стандартные, так и расширенные фреймы
		1	Объект сообщения n может принимать только те фреймы, у которых состояние бита IDE совпадает с его битом IDE
AM	28–0	Маска идентификатора. При приеме расширенного сообщения используется вся маска. При приеме стандартного сообщения используются биты 28–18, при этом состояние битов 17–0 не важно	
–	31, 30	Зарезервировано	

Таблица А.100 – Регистр указателя прерываний объекта сообщения n

MOIPRn



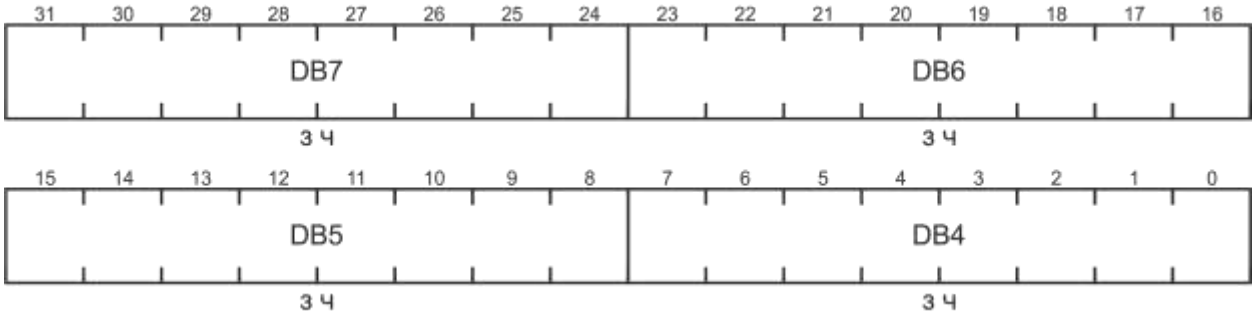
Поле	Бит	Описание
1	2	3
CFCVAL	31–16	Количество фреймов. Каждый раз после записи принятого сообщение в объект сообщения n или успешной передачи объекта сообщения n, значение счетчика фреймов CFC (регистр NFCRn) копируется в CFCVAL
MPN	15–8	Номер ждущего бита сообщения. Указывает позицию бита, соответствующего объекту сообщения n в регистре MSPNDx

Окончание таблицы А.100

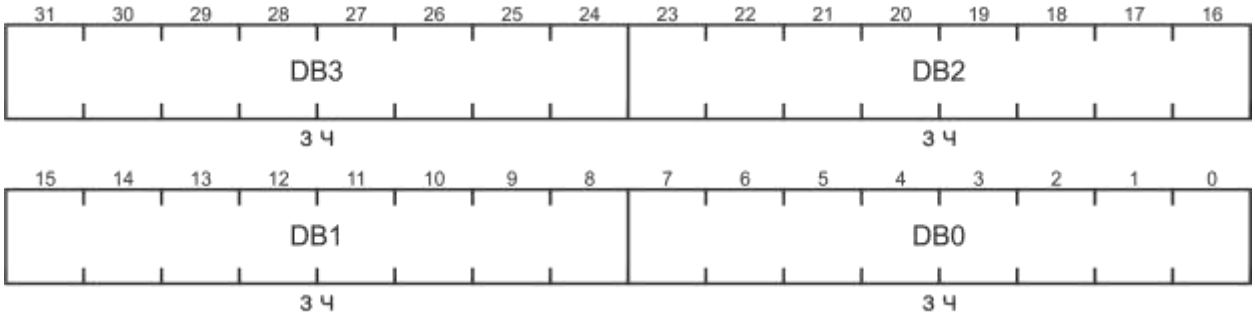
1	2	3
TXINP	7–4	Указатель линии прерываний для прерывания после передачи. Всего доступно три линии прерываний с номерами от 0 до 2. Значение 0000b, записанное в TXINP, выбирает нулевую линию прерываний, 0001b – первую, 0010b – вторую. Дополнительно бит TXINP используется для выбора позиции ждущего бита объекта сообщения n.
RXINP	3–0	Указатель линии прерываний для прерывания после приема. Всего доступно три линии прерываний с номерами от 0 до 2. Значение 0000b, записанное в RXINP, выбирает нулевую линию прерываний, 0001b – первую, 0010b – вторую. Дополнительно бит RXINP используется для выбора позиции ждущего бита объекта сообщения n

Таблица А.101 – Регистры данных объекта сообщения n

MODATAHn



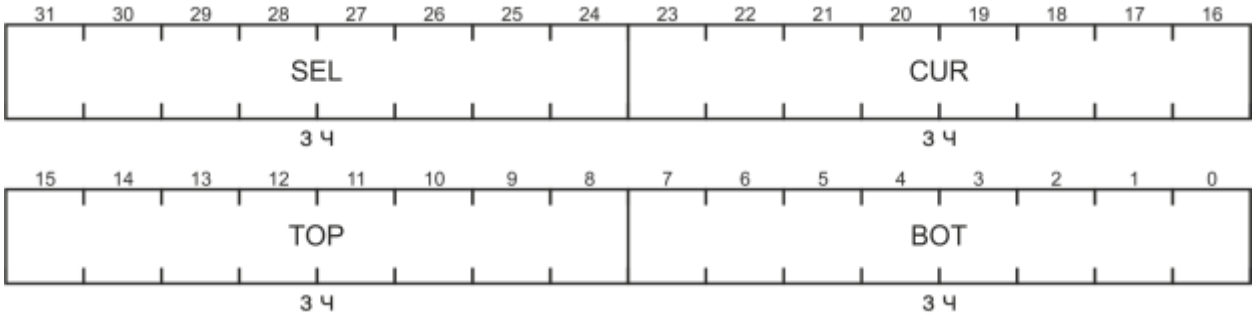
MODATALn



Поле	Бит	Описание
DB7	31–24	Седьмой байт данных
DB6	23–16	Шестой байт данных
DB5	15–8	Пятый байт данных
DB4	7–0	Четвертый байт данных
DB3	31–24	Третий байт данных
DB2	23–16	Второй байт данных
DB1	15–8	Первый байт данных
DB0	7–0	Нулевой байт данных

Таблица А.102 – Регистр указателя FIFO/шлюза объекта сообщения n

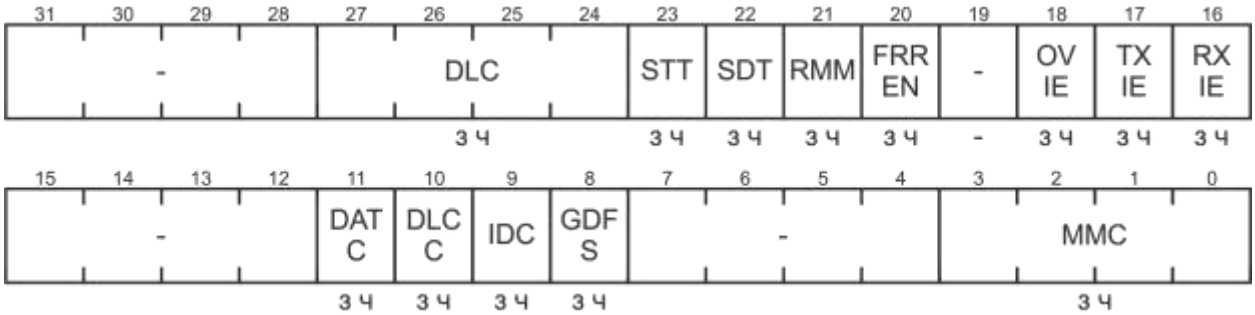
MOFGPRn



Поле	Бит	Описание
SEL	31–24	Указатель объекта сообщения. Второй (программный) указатель в дополнение к аппаратному указателю CUR при работе с FIFO. Поле SEL используется для общего мониторинга (генерирование прерываний FIFO)
CUR	23–16	Указатель на текущий объект в пределах FIFO или шлюза. После каждой операции FIFO или передачи через шлюз указатель CUR обновляется – в него заносится номер следующего объекта сообщения в списке (поле PNEXT регистра MOSTATn) – до тех пор, пока не будет достигнут верхний элемент FIFO (поле TOP), после чего CUR сбрасывается, и в него загружается номер нижнего элемента списка (из поля BOT)
TOP	15–8	Указатель верхнего элемента FIFO. В поле находится номер последнего элемента
BOT	7–0	Указатель нижнего элемента FIFO. В поле находится номер первого элемента

Таблица А.103 – Регистр управления функционированием объекта сообщения n

MOFCRn



Поле	Бит	Описание
1	2	3
DLC	27–24	Код длины данных. Показывает количество байт данных, находящихся в объекте сообщения. Диапазон – значение от 0 до 8. Если значение DLC больше 8, это автоматически указывает на 8 байт. Значение DLC полученного сообщения сохраняется таким, каким было получено

Продолжение таблицы А.103

1	2	3	
STT	23	Бит задания однократной пересылки данных	
		0	Нет действий
		1	Если бит установлен, тогда бит TXRQ сбрасывается после начала передачи объекта сообщения n. В связи с этим, в случае неудачной передачи, повторной передачи сообщения не будет
SDT	22	Бит задания однократного участия объекта сообщения n в пересылке	
		0	Нет действий
		1	Если бит установлен и объект сообщения n не является объектом FIFO, тогда бит MSGVAL сбрасывается после успешной пересылки данных (приема или передачи). В случае, если объект сообщения является объектом FIFO, то бит MSGVAL сбрасывается, когда указатель CUR (указатель на текущий объект) достигает значения SEL в регистре MOFGPRn
RMM	21	Бит включения удаленного мониторинга объекта передачи	
		0	Выключен. Идентификатор, бит IDE и поле DLC объекта сообщения n остаются без изменений до получения корректного фрейма удаленного запроса
		1	Включен. Идентификатор, бит IDE и поле DLC корректного фрейма удаленного запроса копируются в объект передачи n в порядке получения битов фрейма удаленного запроса монитора
		Состояние бита оказывает влияние только на объекты передач	
FRREN	20	Бит разрешения удаленного запроса. Определяет, будет ли устанавливаться бит TXRQ в объекте сообщения n или в другом объекте сообщения, на который указывает CUR	
		0	Бит TXRQ объекта сообщения n устанавливается после получения корректного фрейма удаленного запроса
		1	Бит TXRQ другого объекта сообщения (на который указывает CUR) устанавливается после получения им корректного фрейма удаленного запроса
OVIE	18	Бит разрешения прерывания по заполнению FIFO объекта сообщения n. Прерывание генерируется, когда указатель CUR (указатель на текущий объект) достигает значения SEL регистра MOFGPRn	
		0	Запрещено
		1	Разрешено
		Если объект сообщения n является объектом приема FIFO, то поле TXINP (регистр MOIPRn) указывает на одну из трех линий прерываний. Если объект сообщения n является объектом передачи FIFO, то поле RXINP (регистр MOIPRn) указывает на одну из трех линий прерываний. Для всех других режимов объекта сообщения состояние бита OVIE не важно	
TXIE	17	Бит разрешения прерывания по окончании передачи сообщения	
		0	Запрещено
		1	Разрешено. Прерывание генерируется, если сообщение из объекта сообщения n было успешно передано. Поле TXINP (регистр MOIPRn) указывает на одну из трех линий прерываний

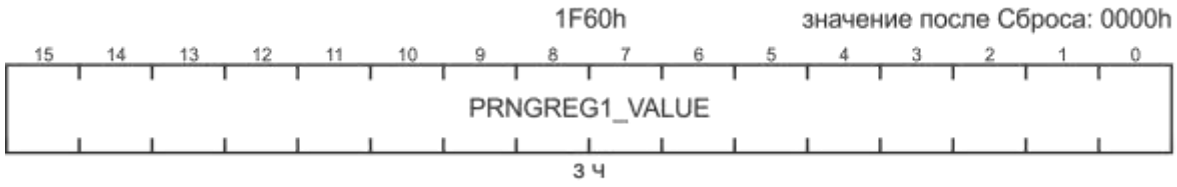
Окончание таблицы А.103

1	2	3	
RXIE	16	Бит разрешения прерывания по окончании приема сообщения	
		0	Запрещено
		1	Разрешено. Прерывание генерируется, если сообщение было успешно принято объектом сообщения n (напрямую или через шлюз). Поле RXINP (регистр MOIPRn) указывает на одну из 16 линий прерываний
DATC	11	Индикатор копирования данных	
		0	Данные не копируются
		1	Данные в регистрах MODATAHn и MODATALn объекта-источника шлюза (после сохранения принятого фрейма в источнике) копируются через шлюз в объект-приемник
		Бит DATC используется только объектом-источником в режиме шлюза. Во всех остальных случаях бит не функционирует	
DLCC	10	Индикатор копирования кода длины данных DLC	
		0	Код не копируется
		1	Код длины данных объекта-источника шлюза (после сохранения принятого фрейма в источнике) копируется через шлюз в объект-приемник
		Бит DLCC используется только объектом-источником в режиме шлюза. Во всех остальных случаях бит не функционирует	
IDC	9	Индикатор копирования идентификатора	
		0	Идентификатор не копируется
		1	Идентификатор объекта-источника шлюза (после сохранения принятого фрейма в источнике) копируется через шлюз в объект-приемник
		Бит IDC используется только объектом-источником в режиме шлюза. Во всех остальных случаях бит не функционирует	
GDFS	8	Индикатор отправки фрейма через шлюз	
		0	Состояние бита TXRQ объекта-приемника без изменений
		1	Установлен бит TXRQ объекта-приемника после внутренней передачи из объекта-источника
		Бит GDFS используется только объектом-источником в режиме шлюза. Во всех остальных случаях бит не функционирует	
MMC	3–0	Задание режима объекта сообщения n	
		0000	Объект сообщения
		0001	Объект-приемник FIFO
		0010	Объект-передатчик FIFO
		0011	Ведомый объект-передатчик FIFO
		0100	Объект-источник шлюза
		Остальные комбинации зарезервированы	
–	31–28, 19, 15–12, 7–4	Зарезервировано	
Примечание – Под корректным фреймом удаленного запроса подразумевается фрейм, идентификатор которого совпадает с идентификатором объекта сообщения.			

Регистры генератора ПСП

Таблица А.104 – Регистр ключа 0 генератора ПСП

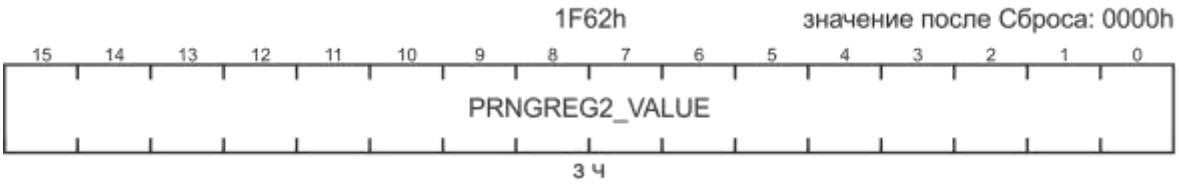
PRNGREG1



Поле	Бит	Описание
PRNGREG1_ VALUE	15–0	Значение ключа 0

Таблица А.105 – Регистр ключа 1 генератора ПСП

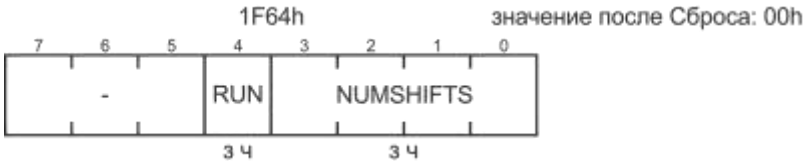
PRNGREG2



Поле	Бит	Описание
PRNGREG2_ VALUE	15–0	Значение ключа 1

Таблица А.106 – Регистр управления генератора ПСП

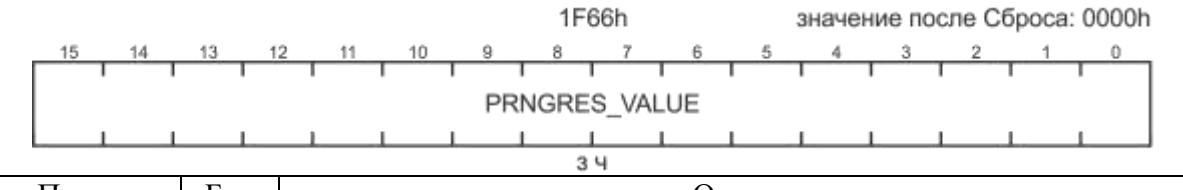
PRNGCON



Поле	Бит	Описание
RUN	4	Бит запуска кодирования. Запись единицы запускает преобразование. После выполнения заданного количества сдвигов бит аппаратно сбрасывается
NUMSHIFTS	3–0	Поле задания количества сдвигов. Следует помнить, что общее количество сдвигов, которые будут выполнены при кодировании равно NUMSHIFTS + 1
–	7–5	Зарезервировано

Таблица А.107 – Регистр результата генератора ПСП

PRNGRES

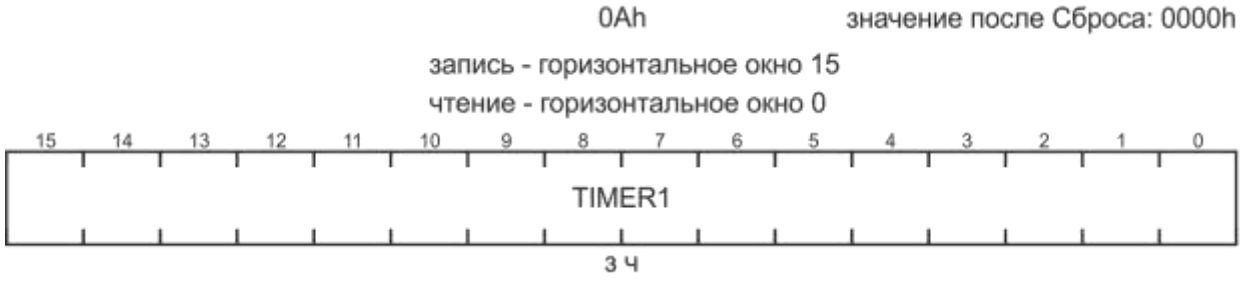


Поле	Бит	Описание
PRNGRES_ VALUE	15–0	Результат кодирования

Регистры модуля HSIO

Таблица А.108 – Регистр таймера 1

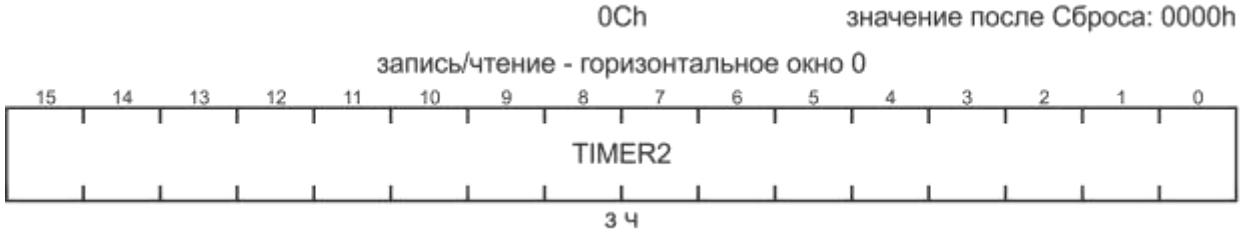
TIMER1



Поле	Бит	Описание
TIMER1	15–0	Поле счетчика таймера

Таблица А.109 – Регистр таймера 2

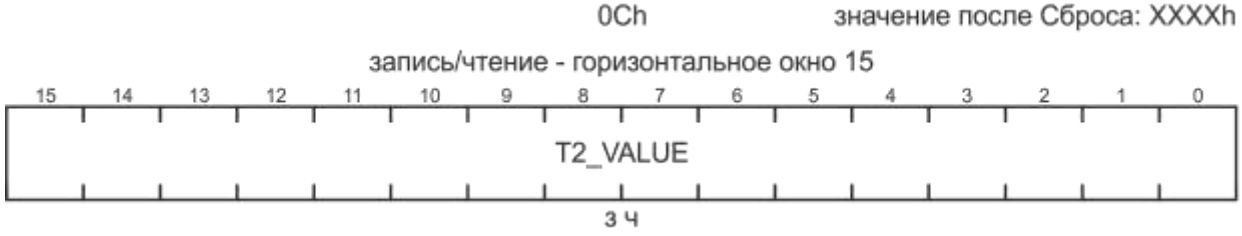
TIMER2



Поле	Бит	Описание
TIMER2	15–0	Поле счетчика таймера

Таблица А.110 – Регистр захвата таймера 2

T2CAPTURE



Поле	Бит	Описание
T2_VALUE	15–0	Захваченное значение таймера 2. Захват состояния счетчика таймера происходит после обнаружения положительного фронта сигнала на выводе P2.7 (T2CAP) микроконтроллера

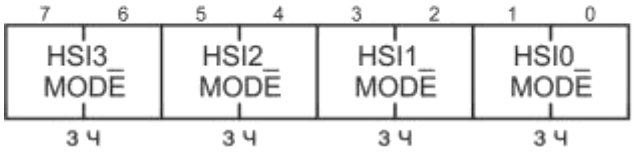
Таблица А.111 – Регистр режима HSI

HSI_MODE

03h значение после Сброса: FFh

запись - горизонтальное окно 0

чтение - горизонтальное окно 15



Поле	Бит	Описание
HSI3_MODE	7–6	Поле задания режима канала 3
HSI2_MODE	5–4	Поле задания режима канала 2
HSI1_MODE	3–2	Поле задания режима канала 1
HSI0_MODE	1–0	Поле задания режима канала 0

Таблица А.112 – Регистр состояния HSI

HSI_STATUS

06h значение после Сброса: X0X0X0X0b

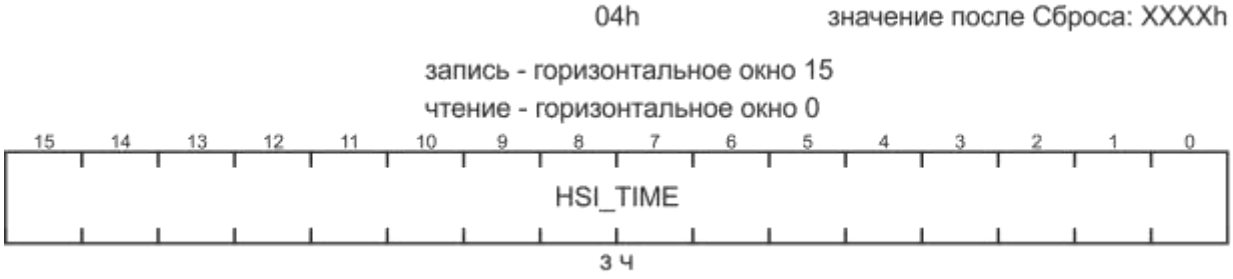
запись битов 0, 2, 4, 6 - горизонтальное окно 15

чтение - горизонтальное окно 0

Поле	Бит	Описание	
HSI3_STAT	7	Индикатор текущего состояния канала 3	
HSI3_EVENT	6	Индикатор события в канале 3	
		0	Нет события
		1	Обнаружено событие
HSI2_STAT	5	Индикатор текущего состояния канала 2	
HSI2_EVENT	4	Индикатор события в канале 2	
		0	Нет события
		1	Обнаружено событие
HSI1_STAT	3	Индикатор текущего состояния канала 1	
HSI1_EVENT	2	Индикатор события в канале 1	
		0	Нет события
		1	Обнаружено событие
HSI0_STAT	1	Индикатор текущего состояния канала 0	
HSI0_EVENT	0	Индикатор события в канале 0	
		0	Нет события
		1	Обнаружено событие

Таблица А.113 – Регистр времени HSI

HSI_TIME



Поле	Бит	Описание
HSI_TIME	15–0	Поле времени события

Таблица А.114 – Регистр управления HSO

HSO_COMMAND

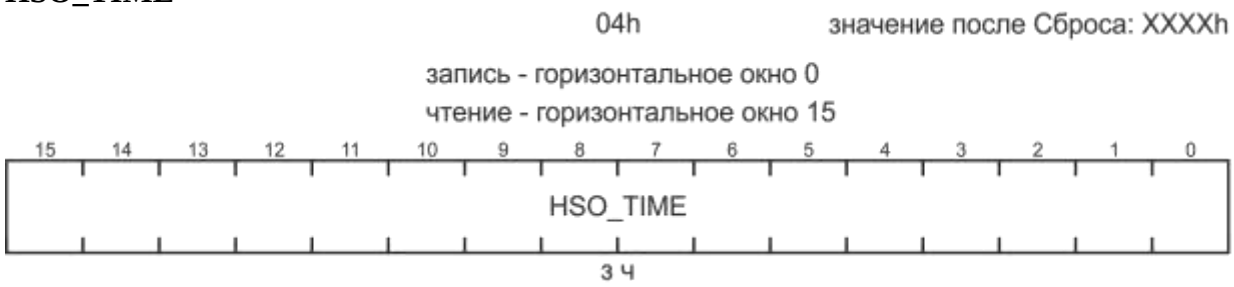
Поле	Бит	Описание	
CAM_LOCK	7	Бит фиксации команды в АЗУ. На работу бита влияет состояние бита LOCK_ENA регистра IOC2	
		0	Ячейка, в которой хранится команда, полностью очищается после выполнения команды
		1	Ячейка, в которой хранится команда, фиксируется, и команда будет выполняться каждый раз, когда запрограммированное значение времени будет совпадать со значением таймера
		Независимо от состояния бита CAM_LOCK, все (!) ячейки АЗУ очищаются, если устанавливается бит CAM_CLR регистра IOC2	
TIMER_SEL	6	Бит выбора таймера	
		0	Таймер 1
		1	Таймер 2
PIN_CMD	5	Бит выбора воздействия команды на вывод/выводы модуля HSO	
		0	Результатом выполнения команды является установка логического нуля на выходе/выходах
		1	Результатом выполнения команды является установка логической единицы на выходе/выходах
HSOINT_ENA	4	Бит разрешения прерывания при выполнении действия. Дополнительно прерывание маскируется битом HSO_MASK регистра INT_MASK	
		0	Запрещено
		1	Разрешено
CMD_TAG	3–0	Код команды. Определяет, какое действие будет активировано во время, заданное в HSO_TIME (см. таблицу А.116)	

Таблица А.115 – Коды команд и соответствующие им действия

CMD_TAG	Мнемоника команды	Действие
0h	HSO0	Переключение выхода HSO.0
1h	HSO1	Переключение выхода HSO.1
2h	HSO2	Переключение выхода HSO.2
3h	HSO3	Переключение выхода HSO.3
4h	HSO4	Переключение выхода HSO.4
5h	HSO5	Переключение выхода HSO.5
6h	HSO01	Переключение выхода HSO.0 и HSO.1
7h	HSO23	Переключение выхода HSO.2 и HSO.3
8h	SWT0	Программное прерывание 0
9h	SWT1	Программное прерывание 1
Ah	SWT2	Программное прерывание 2
Bh	SWT3	Программное прерывание 3
Ch	HSOALL	Переключение выходов HSO.0 – HSO.5
Dh	–	Зарезервировано
Eh	T2RST	Сброс таймера 2
Fh	A_D	Начало аналого-цифрового преобразования

Таблица А.116 – Регистр времени HSO

HSO_TIME

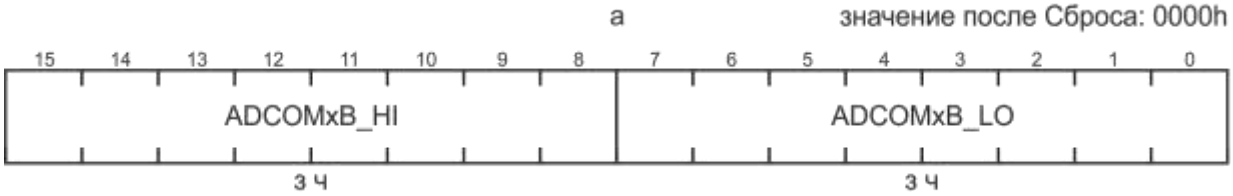


Поле	Бит	Описание
HSO_TIME	15–0	Поле задания времени выполнения команды

Регистры АЦП

Таблица А.117 – Регистр нижней границы цифрового компаратора x

ADCOMxB

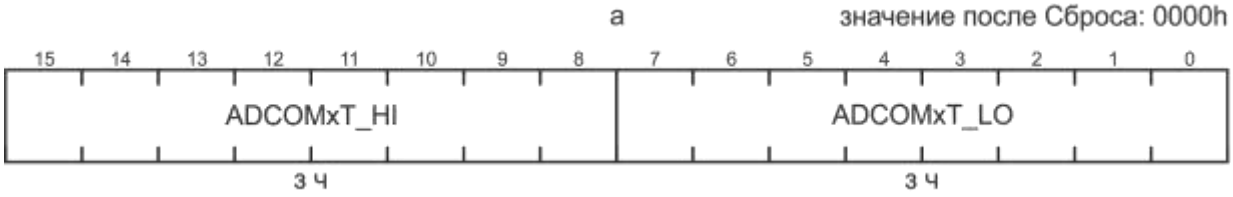


Поле	Бит	Описание
ADCOMxB_HI	15–8	Старший байт нижней границы срабатывания цифрового компаратора x
ADCOMxB_LO	7–0	Младший байт нижней границы срабатывания цифрового компаратора x

Примечание – Приняты условные обозначения: x – порядковый номер цифрового компаратора от 0 до 5; a – адрес регистра в таблице А.2.

Таблица А.118 – Регистр верхней границы цифрового компаратора x

ADCOMxT

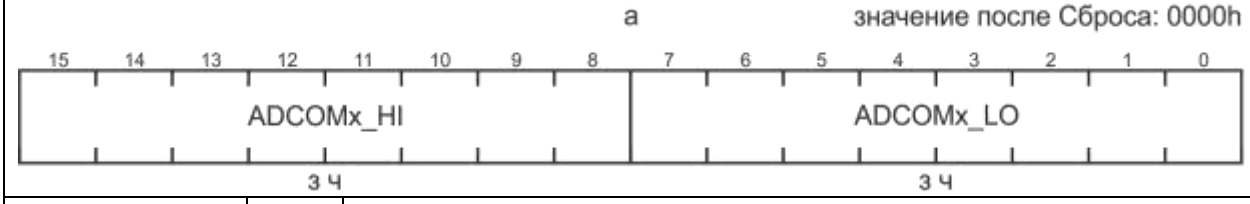


Поле	Бит	Описание
ADCOMxT_HI	15–8	Старший байт верхней границы срабатывания цифрового компаратора x
ADCOMxT_LO	7–0	Содержит младший байт верхней границы срабатывания цифрового компаратора 0

Примечание – Приняты условные обозначения: x – порядковый номер цифрового компаратора от 0 до 5; a – адрес регистра в таблице А.2.

Таблица А.119 – Регистр границы цифрового компаратора x

ADCOMx



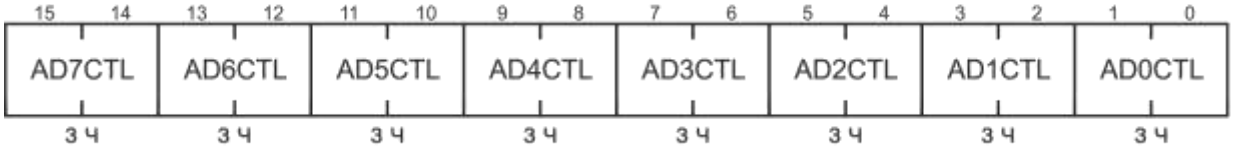
Поле	Бит	Описание
ADCOMx_HI	15–8	Старший байт границы цифрового компаратора x
ADCOMx_LO	7–0	Младший байт границы цифрового компаратора x

Примечание – Приняты условные обозначения: x – порядковый номер цифрового компаратора 6, 7; a – адрес регистра в таблице А.2.

Таблица А.120 – Регистр управления цифровыми компараторами

ADCOMC

1F7Eh значение после Сброса: 00h



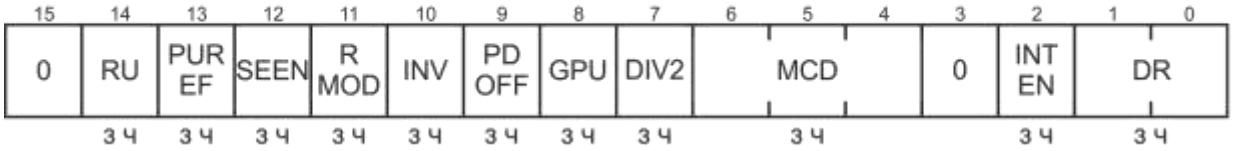
Поле	Бит	Описание	
ADxCTL	15–14, 13–12, 11–10, 9–8, 7–6, 5–4, 3–2, 1–0	Поле задания режима работы компаратора x	
		00	Срабатывание по выходу за нижнюю границу
		01	Срабатывание по выходу за верхнюю границу
		10	Срабатывание по выходу из диапазона
		11	Срабатывание по попаданию в диапазон

Примечание – Приняты условные обозначения: x – порядковый номер цифрового компаратора от 0 до 7.

Таблица А.121 – Регистр управления запуском преобразования

ADCR

1FC0h значение после Сброса: 3206h



Поле	Бит	Описание	
1	2	3	
RU	14	Бит включения подачи буферизированного опорного напряжения на вывод P0.7	
		0	Напряжение не подается
		1	Напряжение подается
PUREF	13	Бит включения опорного напряжения. Источник автоматически включается, если активирован любой из каналов.	
		0	Выключено
		1	Включено
SEEN	12	Бит включения дифференциального режима входов. Функционирует совместно с битами CHREx (x – номер канала от 0 до 7). Бит установлен по умолчанию	
		0	Не использовать!
		1	Режим прямого входа для канала x, чей бит CHREx установлен в регистре ADCRST. Канал x, чей бит CHREx сброшен, функционирует в режиме дифференциального входа
RMOD	11	Бит разрешения сброса аналоговых модуляторов	
		0	Сброс запрещен
		1	Сброс разрешен для канала x, чей бит CHREx установлен в регистре ADCRST

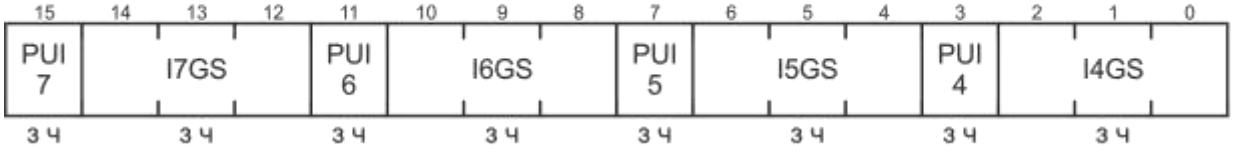
Окончание таблицы А.121

1	2	3	
INV	10	Бит разрешения инверсии каналов	
		0	Инверсия запрещена
		1	Инверсия разрешена для канала x, чей бит CHIx установлен в регистре ADCRST
PDOFF	9	Бит управления выключением опорного напряжения	
		0	Опорное напряжение не выключается
		1	Опорное напряжение выключается при входе в режим POWERDOWN или при запрещении тактового сигнала АЦП в регистре CLKCL
GPU	8	Бит запуска преобразования всех каналов непосредственно после установки	
		0	Нет действий
		1	Запуск преобразования
DIV2	7	Бит включения дополнительного делителя на два тактовой частоты	
		0	Выключен
		1	Включен
MCD	6–4	Поле задания делителя опорной тактовой частоты АЦП	
		0h	Делитель выключен
		1h	1/2
		2h	1/3
		3h	1/4
		4h	1/5
		5h –7h	Делитель выключен
INTEN	2	Бит разрешения прерывания по окончании преобразования	
		0	Запрещено
		1	Разрешено
DR	1–0	Поле задания частоты выборки	
		00	1/2048
		01	1/1024
		10	1/512
		11	1/256
0	15, 3	Зарезервировано. Всегда записывать «0»!	

Таблица А.122 – Регистр управления каналами АЦП

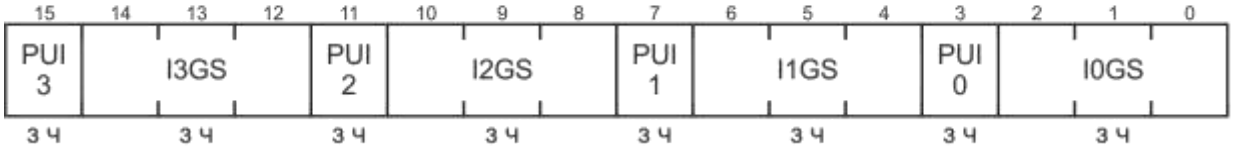
ADCCHCH

1FC4h значение после Сброса: 0000h



ADCCHCL

1FC2h значение после Сброса: 0000h



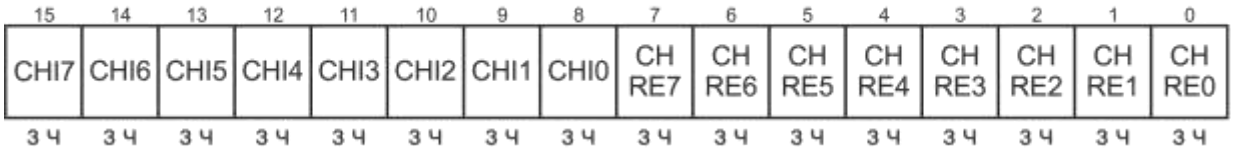
Поле	Бит	Описание	
PUIx	15, 11, 7, 3	Бит запуска преобразования канала x. Бит запуска непрерывного преобразования канала 0: «1» – старт преобразования; «0» – нет преобразования	
		0	Нет действий
		1	Запуск
IxGS	14–12, 10–8, 6–4, 2–0	Поле задания коэффициента усиления канала x (в дБ)	
		0h	0
		1h	6
		2h	12
		3h	18
		4h	20
		5h	26
		6h	32
		7h	38

Примечание – Приняты условные обозначения: x – номер канала от 0 до 3.

Таблица А.123 – Регистр управления сбросом каналов АЦП

ADCRST

1FC6h значение после Сброса: 00FFh



Установленный бит CHIx включает режим инверсии для канала x. Сброшенный бит CHIx выключает режим инверсии.

Установленный бит CHREx разрешает сброс и устанавливает режим прямого входа для канала x. Сброшенный бит CHREx запрещает сброс и устанавливает дифференциальный режим для канала x (только для каналов 0 – 3).

Примечания

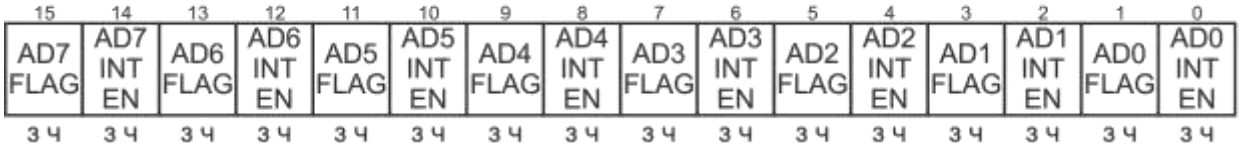
1 Не рекомендуется включать режим дифференциального включения входов для каналов 4 – 7, так как это приведет к неправильному функционированию АЦП.

2 Принятое обозначение x – номер канала от 0 до 7.

Таблица А.124 – Регистр управления прерываниями цифровых компараторов

ADCIEN

1F7Ch значение после Сброса: 0202h

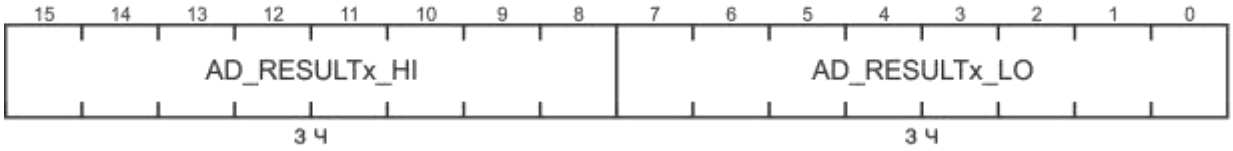


Поле	Бит	Описание
ADxFLAG	15, 13, 11, 9, 7, 5, 3, 1	Флаг срабатывания компаратора канала x
ADxINTEN	14, 12, 10, 8, 6, 4, 2, 0	Бит разрешения прерывания канала x
Примечание – Приняты условные обозначения: x – номер канала от 0 до 7.		

Таблица А.125 – Регистр результата канала x АЦП

ADCRESx

a значение после Сброса: 0000h



Поле	Бит	Описание
AD_RESULTx_HI	15–8	Старший байт результата преобразования канала x
AD_RESULTx_LO	7–0	Младший байт результата преобразования канала x
Примечание – Приняты условные обозначения: x – номер канала от 0 до 7; a – адрес регистра в таблице А.2.		

Регистры ЦАП, модуля ШИМ и сторожевого таймера

Таблица А.126 – Регистр управления ЦАП

DACVAL

Поле	Бит	Описание
SLEEPC	15	Бит управления переводом ЦАП в режим пониженного энергопотребления
		0 Управление переводом осуществляется битом SLEEP
		1 Перевод осуществляется автоматически при включении режима POWERDOWN или выключении тактового сигнала
SLEEP	14	Бит управления переводом ЦАП в режим пониженного энергопотребления. Функционирует совместно с битом SLEEPC
		0 Нет действий. ЦАП активен
		1 Установка бита переводит ЦАП в режим пониженного энергопотребления
DBx	Биты с 13 по 0	Биты, определяющие ток, протекающий через выводы ЦАП

Примечания
1 Для корректной работы блока значение в регистр необходимо записывать словом.
2 x – номер вывода от 0 до 13.

Таблица А.127 – Регистр длительности импульса канала x

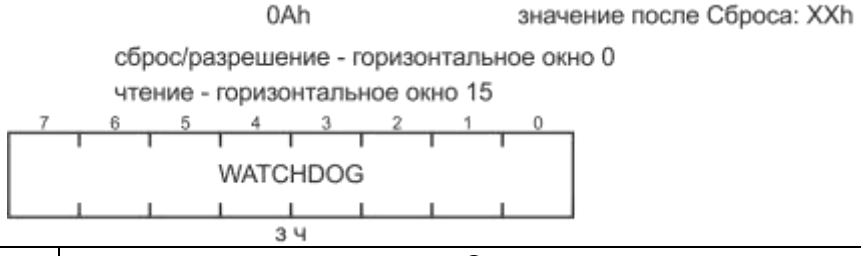
PWMx_CONTROL

Поле	Бит	Описание
IMP	7–0	Длительность импульса формируемого сигнала. Задается в количестве тактов (от 0 до 256)

Примечание – Адреса регистров см. в таблице А.2.

Таблица А.128 – Регистр сторожевого таймера

WATCHDOG



Поле	Бит	Описание
WATCHDOG	7–0	Старший байт счетчика сторожевого таймера

Приложение Б
(обязательное)

Система команд микроконтроллера

Система команд микроконтроллера включает в себя 111 команд, поддерживаемых стандартными трансляторами с языка ассемблера для микроконтроллеров семейства MCS-196, и команду программного прерывания TRAP, используемую в инструментальных средствах разработки. Команды сложения, вычитания, умножения, деления, логических операций, сравнения, загрузки, а также команды работы с битами и стеком поддерживают до четырех типов адресации.

Краткую информацию о системе команд можно получить из таблиц Б.1 и Б.2.

В таблице Б.3 представлена информация о командах микроконтроллера и вариантах их использования. Для каждой команды указано:

- мнемоническое название;
- название выполняемой операции и порядок следования операндов;
- состояние флагов регистра PSW, возникающее в результате выполнения команды;
- варианты использования, с указанием для каждого варианта его опкода, длины и времени выполнения.

В таблице Б.3 приняты обозначения:

DEST – операнд-приемник;
SRC – операнд-источник;
SRC1 – первый операнд-источник;
SRC2 – второй операнд-источник;
Ireg – регистр двойного слова (32 бита) во внутреннем регистровом файле;
wreg – регистр слова (16 бит) во внутреннем регистровом файле;
breg – регистр байта (8 бит) во внутреннем регистровом файле;
#data – 8/16-разрядная константа;
reg – 8/16-разрядный регистр во внутреннем регистровом файле, используемый для косвенной адресации;
boffset – 8-разрядная константа или 8-разрядный регистр во внутреннем регистровом файле. Используется при короткоиндексной адресации для задания смещения со знаком;
woffset – 16-разрядная константа или 16-разрядный регистр во внутреннем регистровом файле. Используется при длинноиндексной адресации для задания смещения со знаком;
cadd – 8/16/32-разрядный адрес в программном коде, указанный непосредственно, или как название метки перехода;
#count – 4-разрядная константа для задания количества сдвигов (от 0 до 15) при сдвиговых операциях.

Для представления влияния, оказываемого результатом выполнения команды на флаги регистра PSW, приняты обозначения:

√ – команда модифицирует флаг (устанавливает или сбрасывает, если требуется);
– – команда не модифицирует флаг;
↓ – команда сбрасывает флаг, если требуется, но не устанавливает;
↑ – команда устанавливает флаг, если требуется, но не сбрасывает;
1 – команда устанавливает флаг;
0 – команда сбрасывает флаг;
? – команда оставляет флаг в неопределенном состоянии.

Далее в настоящем приложении приводится дополнительная информация по всем командам микроконтроллера. В описании используются следующие обозначения:

aa – два младших бита опкода команды, определяющие используемый режим адресации:
00 – прямая регистровая;
01 – непосредственная;
10 – косвенная (в том числе с автоинкрементом);
11 – индексная (короткая и длинная);

DIreg – регистр двойного слова (32 бита) во внутреннем регистровом файле, использующийся в командах как операнд-приемник (DEST). Должен выравниваться по адресу, кратному четырем;

Dwreg – регистр слова (16 бит) во внутреннем регистровом файле, использующийся в командах как операнд-приемник (DEST). Должен выравниваться по адресу, кратному двум;

Dbreg – регистр байта (8 бит) во внутреннем регистровом файле, использующийся в командах как операнд-приемник (DEST);

SIreg – регистр двойного слова (32 бита) во внутреннем регистровом файле, использующийся в командах как операнд-источник (SRC). Должен выравниваться по адресу, кратному четырем;

Swreg – регистр слова (16 бит) во внутреннем регистровом файле, использующийся в командах как операнд-источник (SRC). Должен выравниваться по адресу, кратному двум;

Sbreg – регистр байта (8 бит) во внутреннем регистровом файле, использующийся в командах как операнд-источник (SRC);

wreg – регистр слова (16 бит) во внутреннем регистровом файле. Должен выравниваться по адресу, кратному двум. В командах с двумя и тремя операндами используется как операнд-источник;

breg – регистр байта (8 бит) во внутреннем регистровом файле. В командах с двумя и тремя операндами используется как операнд-источник;

waop – регистр слова (16 бит), адресуемый любым, допускаемым командой способом адресации, и всегда использующийся как операнд-источник (исключение: в командах ST и POP используется как приемник). Должен выравниваться по адресу, кратному двум;

baop – регистр байта (8 бит), адресуемый любым, допускаемым командой способом адресации, и всегда использующийся как операнд-источник (исключение: в команде STB используется как приемник);

cadd – адрес в программном коде, указанный непосредственно, например, JC 20C0h, или как название метки перехода, например, JC label;

disp – смещение, являющееся расстоянием между концом команды и адресом (меткой) перехода. Вычисляется аппаратно;

scount – количество сдвигов, указываемое в командах логического сдвига и командах арифметического сдвига. Количество сдвигов от 0 до 15 может быть задано непосредственно, например, SHR AX, #5, а количество сдвигов от 0 до 31 может быть задано как содержимое какого-либо регистра, в котором хранится значение от 00h до 1Fh, например, SHR AX, BX.

Таблица Б.1 – Команды микроконтроллера для работы со словами и байтами

Мнемоника команды, определяющей операцию над		Название операции
словами	байтами	
Арифметические команды		
ADD	ADDB	Сложение
ADDC	ADDCB	Сложение с переносом
CMP	CMPB	Сравнение
CMPL	–	Сравнение длинных слов
SUB	SUBB	Вычитание
SUBC	SUBCB	Вычитание с переносом
DIV	DIVB	Знаковое деление
DIVU	DIVUB	Беззнаковое деление
MUL	MULB	Знаковое умножение
MULU	MULUB	Беззнаковое умножение
Логические команды		
AND	ANDB	Умножение («И»)
OR	ORB	Сложение («ИЛИ»)
XOR	XORB	Сложение по модулю два (исключающее «ИЛИ»)
Команды работы с данными		
LD	LDB	Загрузка
–	LDBSE	Загрузка со знаковым расширением
–	LDBZE	Загрузка с беззнаковым расширением
ST	STB	Сохранение
CLR	CLRB	Очистка
INC	INCB	Инкремент
DEC	DECB	Декремент
NOT	NOTB	Инверсия
NEG	NEGB	Изменение знака
EXT	EXTB	Знаковое расширение
XCH	XCHB	Обмен
Блочные команды		
BMOV		Непрерываемое перемещение блоков данных
BMOVI		Прерываемое перемещение блоков данных
Команды сдвига		
NORML	–	Нормализация двойного слова
SHL	SHLB	Логический сдвиг влево
SHR	SHRB	Логический сдвиг вправо
SHLL	–	Логический сдвиг двойного слова влево
SHRL	–	Логический сдвиг двойного слова вправо
SHRA	SHRAB	Арифметический сдвиг вправо
SHRAL	–	Арифметический сдвиг двойного слова вправо

Таблица Б.2 – Команды микроконтроллера для работы с битами, стеком и специальные

Мнемоника команды	Название операции
1	2
Команды управления PTS	
DPTS	Запрет PTS

Окончание таблицы Б.2

1	2
EPTS	Разрешение PTS
Команды работы со стеком	
PUSH	Загрузка слова в стек
PUSHA	Загрузка всего в стек
PUSHF	Загрузка флагов в стек
POP	Чтение слова из стека
POPA	Чтение всего из стека
POPF	Чтение флагов из стека
Команды вызова	
LCALL	Длинный вызов
SCALL	Короткий вызов
RET	Возврат из подпрограммы
TRAP	Программное прерывание (не поддерживается транслятором с языка)
Команды безусловного и условного перехода	
BR	Косвенный переход
LJMP	Длинный вызов
SJMP	Короткий вызов
TIJMP	Косвенный табличный переход
DJNZ	Декремент байта и переход при отсутствии нуля
DJNZW	Декремент слова и переход при отсутствии нуля
JBC	Переход, если бит очищен
JBS	Переход, если бит установлен
JC	Переход, если C = 1
JNC	Переход, если C = 0
JLT	Переход, если N = 1
JGE	Переход, если N = 0
JE	Переход, если Z = 1
JNE	Переход, если Z = 0
JV	Переход, если V = 1
JNV	Переход, если V = 0
JVT	Переход, если VT = 1
JNVT	Переход, если VT = 0
JST	Переход, если ST = 1
JNST	Переход, если ST = 0
JGT	Переход, если N = 0 и Z = 0
JLE	Переход, если N = 1 или Z = 1
JH	Переход, если C = 1 и Z = 0
JNH	Переход, если C = 0 и Z = 1
Специальные команды	
DI	Запрещение всех прерываний
EI	Разрешение всех прерываний
CLRC	Очистка флага переноса
SETC	Установка флага переноса
CLRVT	Очистка дополнительного флага переполнения
IDLPD	Включение режима Idle или Powerdown
NOP	Нет операции
SKIP	Нет операции (двухбайтная)
RST	Сброс системы

Таблица Б.3 – Система команд микроконтроллера

Мнемо-ника	Операция и варианты использования команды	Опкод	Длина	Время выполнения*
1	2	3	4	5
ADD	**Сложение слов** (DEST, SRC)			Z N C V VT ST √ √ √ √ ↑ –
	ADD wreg, wreg	64h	3 байта	4 мт
	ADD wreg, #data	65h	4 байта	
	ADD wreg, [reg]	66h	3 байта	5/9 мт
	ADD wreg, [reg]+			6/10 мт
	ADD wreg, boffset[reg]	67h	4 байта	6/10 мт
	ADD wreg, woffset[reg]		5 байт	
ADD	**Сложение слов** (DEST, SRC1, SRC2)			Z N C V VT ST √ √ √ √ ↑ –
	ADD wreg, wreg, wreg	44h	4 байта	4 мт
	ADD wreg, wreg, #data,	45h	5 байт	
	ADD wreg, wreg, [reg]	46h	4 байта	5/9 мт
	ADD wreg, wreg, [reg]+			6/10 мт
	ADD wreg, wreg, boffset[reg]	47h	5 байт	6/10 мт
	ADD wreg, wreg, woffset[reg]		6 байт	
ADDB	**Сложение байт** (DEST, SRC)			Z N C V VT ST √ √ √ √ ↑ –
	ADDB breg, breg	74h	3 байта	4 мт
	ADDB breg, #data,	75h		
	ADDB breg, [reg]	76h		5/9 мт
	ADDB breg, [reg]+			6/10 мт
	ADDB breg, boffset[reg]	77h	4 байта	6/10 мт
	ADDB breg, woffset[reg]		5 байт	
ADDB	**Сложение байт** (DEST, SRC1, SRC2)			Z N C V VT ST √ √ √ √ ↑ –
	ADDB breg, breg, breg	54h	4 байта	4 мт
	ADDB breg, breg, #data,	55h		
	ADDB breg, breg, [reg]	56h		5/9 мт
	ADDB breg, breg, [reg]+			
	ADDB breg, breg, boffset[reg]	57h	5 байт	6/10 мт
	ADDB breg, breg, woffset[reg]		6 байт	
ADDC	**Сложение слов с переносом** (DEST, SRC)			Z N C V VT ST ↓ √ √ √ ↑ –
	ADDC wreg, wreg	A4h	3 байта	4 мт
	ADDC wreg, #data	A5h	4 байта	
	ADDC wreg, [reg]	A6h	3 байта	5/9 мт
	ADDC wreg, [reg]+			
	ADDC wreg, boffset[reg]	A7h	4 байта	6/10 мт
	ADDC wreg, woffset[reg]		5 байт	

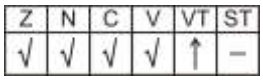
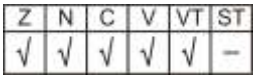
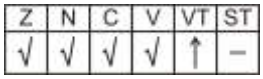
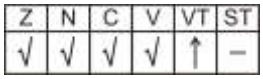
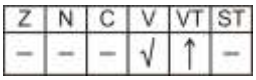
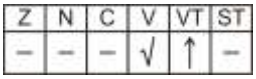
Продолжение таблицы Б.3

1	2	3	4	5
ADDCB	**Сложение байт с переносом** (DEST, SRC)		Z N C V VT ST ↓ √ √ √ ↑ –	
	ADDCB breg, breg	B4h	3 байта	4 мт
	ADDCB breg, #data,	B5h		
	ADDCB breg, [reg]	B6h		5/9 мт
	ADDCB breg, [reg]+			6/10 мт
	ADDCB breg, boffset[reg]	B7h	4 байта	6/10 мт
	ADDCB breg, woffset[reg]		5 байт	
AND	**Логическое «И» слов** (DEST, SRC)		Z N C V VT ST √ √ 0 0 – –	
	AND wreg, wreg	60h	3 байта	4 мт
	AND wreg, #data	61h	4 байта	
	AND wreg, [reg]	62h	3 байта	5/9 мт
	AND wreg, [reg]+			6/10 мт
	AND wreg, boffset[reg]	63h	4 байта	6/10 мт
	AND wreg, woffset[reg]		5 байт	
AND	**Логическое «И» слов** (DEST, SRC1, SRC2)		Z N C V VT ST √ √ 0 0 – –	
	AND wreg, wreg, wreg	40h	4 байта	4 мт
	AND wreg, wreg, #data,	41h	5 байт	
	AND wreg, wreg, [reg]	42h	4 байта	5/9 мт
	AND wreg, wreg, [reg]+			6/10 мт
	AND wreg, wreg, boffset[reg]	43h	5 байт	6/10 мт
	AND wreg, wreg, woffset[reg]		6 байт	
ANDB	**Логическое «И» байт** (DEST, SRC)		Z N C V VT ST √ √ 0 0 – –	
	ANDB breg, breg	70h	3 байта	4 мт
	ANDB breg, #data,	71h		
	ANDB breg, [reg]	72h		5/9 мт
	ANDB breg, [reg]+			6/10 мт
	ANDB breg, boffset[reg]	73h	4 байта	6/10 мт
	ANDB breg, woffset[reg]			
ANDB	**Логическое «И» байт** (DEST, SRC1, SRC2)		Z N C V VT ST √ √ 0 0 – –	
	ANDB breg, breg, breg	50h	4 байта	4 мт
	ANDB breg, breg, #data,	51h		
	ANDB breg, breg, [reg]	52h		5/9 мт
	ANDB breg, breg, [reg]+			6/10 мт
	ANDB breg, breg, boffset[reg]	53h	5 байт	6/10 мт
	ANDB breg, breg, woffset[reg]			

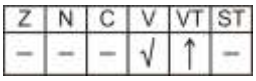
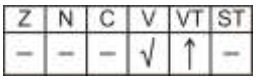
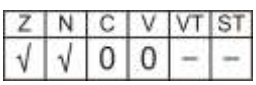
Продолжение таблицы Б.3

1	2	3	4	5
BMOV	**Непрерываемое перемещение блоков данных** (DEST, SRC)			
	BMOV Ireg, woffset[reg]	C1h	3 байта	11мт (Ex/In) 16 мт (Ex/Ex)
BMOVI	**Прерываемое перемещение блоков** (DEST, SRC)			
	BMOVI Ireg, woffset[reg]	CDh	3 байта	11мт (Ex/In) 16 мт (Ex/Ex)
BR	**Косвенный переход** (DEST)			
	BR [reg]	E3h	2 байта	2 мт
	BR [reg]+			
	BR boffset[reg]			
	BR woffset[reg]			
CLR	**Очистка слова** (DEST)		Z N C V VT ST 1 0 0 0 – –	
	CLR wreg	01h	–	2 мт
CLRB	**Очистка байта** (DEST)		Z N C V VT ST 1 0 0 0 – –	
	CLRB breg	11h	–	2 мт
CLRC	**Очистка флага переноса**		Z N C V VT ST – – 0 – – –	
	CLRC (прямая адресация)	F8h	1 байт	1 мт
CLRVT	**Очистка дополнительного флага переполнения**		Z N C V VT ST – – – – 0 –	
	CLRVT (прямая адресация)	FCh	1 байт	1 мт
CMP	**Сравнение слов** (DEST, SRC)		Z N C V VT ST √ √ √ √ ↑ –	
	CMP wreg, wreg	88h	3 байта	4 мт
	CMP wreg, #data	89h	4 байта	
	CMP wreg, [reg]	8Ah	3 байта	5/9 мт
	CMP wreg, [reg]+			6/10 мт
	CMP wreg, boffset[reg]	8Bh	4 байта	6/10 мт
	CMP wreg, woffset[reg]		5 байт	

Продолжение таблицы Б.3

1	2	3	4	5
CMPB	**Сравнение байт** (DEST, SRC)			
	CMPB breg, breg	98h	3 байта	4 мт
	CMPB breg, #data,	99h		
	CMPB breg, [reg]	9Ah		5/9 мт
	CMPB breg, [reg]+			6/10 мт
	CMPB breg, boffset[reg]	9Bh	4 байта	6/10 мт
	CMPB breg, woffset[reg]		5 байт	
CMPL	**Сравнение длинных слов** (DEST, SRC)			
	CMPL Ireg, Ireg	C5h	–	6 мт
DEC	**Декремент слова** (DEST)			
	DEC wreg	05h	–	3 мт
DECB	**Декремент байта** (DEST)			
	DEC breg	15h	–	3 мт
DI	**Запрещение всех прерываний**			
	DI	FAh	1 байт	1 мт
DIV	**Знаковое деление слов** (DEST, SRC)			
	DIV Ireg, wreg	FE8Ch	4 байта	8 мт
	DIV Ireg, #data	FE8Dh	5 байт	
	DIV Ireg, [reg]	FE8Eh	4 байта	9/13 мт
	DIV Ireg, [reg]+		4 байта	10/14 мт
	DIV Ireg, boffset[reg]	FE8Fh	5 байт	10/14 мт
	DIV Ireg, woffset[reg]		6 байт	
DIVB	**Знаковое деление байт** (DEST, SRC)			
	DIVB wreg, breg	FE9Ch	4 байта	6 мт
	DIVB wreg, #data	FE9Dh		
	DIVB wreg, [reg]	FE9Eh		7/11 мт
	DIVB wreg, [reg]+			8/12 мт
	DIVB wreg, boffset[reg]	FE9Fh	5 байт	8/12 мт
	DIVB wreg, woffset[reg]		6 байт	

Продолжение таблицы Б.3

1	2	3	4	5
DIVU	**Беззнаковое деление слов** (DEST, SRC)			
	DIVU Ireg, wreg	8Ch	3 байта	8 мт
	DIVU Ireg, #data	8Dh	4 байта	
	DIVU Ireg, [reg]	8Eh	3 байта	9/13 мт
	DIVU Ireg, [reg]+			10/14 мт
	DIVU Ireg, boffset[reg]	8Fh	4 байта	10/14 мт
	DIVU Ireg, woffset[reg]		5 байт	
DIVUB	**Беззнаковое деление байт** (DEST, SRC)			
	DIVUB wreg, breg	9Ch	3 байта	6 мт
	DIVUB wreg, #data	9Dh		
	DIVUB wreg, [reg]	9Eh		7/11 мт
	DIVUB wreg, [reg]+			8/12 мт
	DIVUB wreg, boffset[reg]	9Fh	4 байта	8/12 мт
	DIVUB wreg, woffset[reg]		5 байт	
DJNZ	**Декремент байта и переход при отсутствии нуля** (COUNT, ADDR)			
	DJNZ breg, cadd (короткоиндексная адресация)	E0h	3 байта	4 мт
DJNZW	**Декремент слова и переход при отсутствии нуля** (COUNT, ADDR)			
	DJNZW wreg, cadd (короткоиндексная адресация)	E1h	3 байта	4 мт
DPTS	**Запрет PTS**			
	DPTS	ECh	1 байт	1 мт
EI	**Разрешение всех прерываний**			
	EI	FBh	1 байт	1 мт
EPTS	**Разрешение PTS**			
	EPTS	EDh	1 байт	1 мт
EXT	**Знаковое расширение слова до двойного слова** (DEST)			
	EXT (прямая адресация)	06h	–	4 мт

Продолжение таблицы Б.3

1	2	3	4	5
EXTB	**Знаковое расширение байта до слова** (DEST)		Z N C V VT ST √ √ 0 0 – –	
	EXTB (прямая адресация)	16h	–	4 мт
IDLPD	**Включение режима Idle или Powerdown**		Если параметр (key) задан неверно, то флаги в PSW сбрасываются Z N C V VT ST 0 0 0 0 0 0	
	IDLPD #1 (включение режима Idle) IDLPD #2 (включение режима Powerdown) IDLPD #key (сброс микроконтроллера)	F6h	1 байт	5 мт
INC	**Инкремент слова** (DEST)		Z N C V VT ST √ √ √ √ ↑ 0	
	INC wreg	07h	–	3 мт
INCB	**Инкремент байта** (DEST)		Z N C V VT ST √ √ √ √ ↑ –	
	INC breg	17h	–	3 мт
JBC	**Переход, если бит очищен**			
	JBC breg, 0, cadd	30h	3 байт	2 мт
	JBC breg, 1, cadd	31h		
	JBC breg, 2, cadd	32h		
	JBC breg, 3, cadd	33h		
	JBC breg, 4, cadd	34h		
	JBC breg, 5, cadd	35h		
	JBC breg, 6, cadd	36h		
	JBC breg, 7, cadd (короткоиндексная адресация)	37h		
JBS	**Переход, если бит установлен**			
	JBS breg, 0, cadd	38h	3 байт	2 мт
	JBS breg, 1, cadd	39h		
	JBS breg, 2, cadd	3Ah		
	JBS breg, 3, cadd	3Bh		
	JBS breg, 4, cadd	3Ch		
	JBS breg, 5, cadd	3Dh		
	JBS breg, 6, cadd	3Eh		
	JBS breg, 7, cadd (короткоиндексная адресация)	3Fh		
JC	**Переход, если C = 1**			
	JC cadd (короткоиндексная адресация)	DBh	1 байт	1 мт

Продолжение таблицы Б.3

1	2	3	4	5
JE	**Переход, если Z = 1**			
	JE cadd (короткоиндексная адресация)	DFh	1 байт	1 мт
JGE	**Переход, если N = 0**			
	JGE cadd (короткоиндексная адресация)	D6h	1 байт	1 мт
JGT	**Переход, если N = 0 и Z = 0**			
	JGT cadd (короткоиндексная адресация)	D2h	1 байт	1 мт
JH	**Переход, если C = 1 и Z = 0**			
	JH cadd (короткоиндексная адресация)	D9h	1 байт	1 мт
JLE	**Переход, если N = 1 или Z = 1**			
	JLE cadd (короткоиндексная адресация)	DAh	1 байт	1 мт
JLT	**Переход, если N = 1**			
	JLT cadd (короткоиндексная адресация)	DEh	1 байт	1 мт
JNC	**Переход, если C = 0**			
	JNC cadd (короткоиндексная адресация)	D3h	1 байт	1 мт
JNE	**Переход, если Z = 0**			
	JNE cadd (короткоиндексная адресация)	D7h	1 байт	1 мт
JNH	**Переход, если C = 0 и Z = 1**			
	JNH cadd (короткоиндексная адресация)	D1h	1 байт	1 мт
JNST	**Переход, если ST = 0**			
	JNST cadd (короткоиндексная адресация)	D0h	1 байт	1 мт
JNV	**Переход, если V = 0**			
	JNV cadd (короткоиндексная адресация)	D5h	1 байт	1 мт
JNVT	**Переход, если VT = 0**			
	JNVT cadd (короткоиндексная адресация)	D4h	1 байт	1 мт

Z	N	C	V	VT	ST
–	–	–	–	0	–

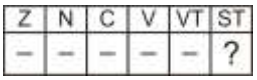
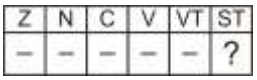
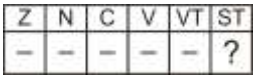
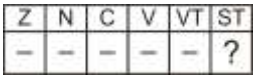
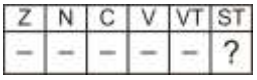
Продолжение таблицы Б.3

1	2	3	4	5
JST	**Переход, если ST = 1**			
	JST cadd (короткоиндексная адресация)	D8h	1 байт	1 мт
JV	**Переход, если V = 1**			
	JV cadd (короткоиндексная адресация)	DDh	1 байт	1 мт
JVT	**Переход, если VT = 1**		Z N C V VT ST – – – – 0 –	
	JVT cadd (короткоиндексная адресация)	DCh	1 байт	1 мт
LCALL	**Длинный вызов**			
	LCALL cadd (длинноиндексная адресация)	EFh	3 байта	3 (7) мт
LD	**Загрузка слова** (DEST, SRC)			
	LD wreg, wreg	A0h	3 байта	2 мт
	LD wreg, #data	A1h	4 байта	1 мт
	LD wreg, [reg]	A2h	3 байта	3/7 мт
	LD wreg, [reg]+			4/8 мт
	LD wreg, boffset[reg]	A3h	4 байта	4/8 мт
	LD wreg, woffset[reg]		5 байт	
LDB	**Загрузка байта** (DEST, SRC)			
	LDB breg, breg	B0h	3 байта	2 мт
	LDB breg, #data	B1h		1 мт
	LDB breg, [reg]	B2h		3/7 мт
	LDB breg, [reg]+			4/8 мт
	LDB breg, boffset[reg]	B3h	4 байта	4/8 мт
	LDB breg, woffset[reg]		5 байт	
LDBSE	**Загрузка байта со знаковым расширением** (DEST, SRC)			
	LDBSE wreg, breg	BCh	3 байта	2 мт
	LDBSE wreg, #data	BDh		1 мт
	LDBSE wreg, [reg]	BEh		3/7 мт
	LDBSE wreg, [reg]+			4/8 мт
	LDBSE wreg, boffset[reg]	BFh	4 байта	4/8 мт
	LDBSE wreg, woffset[reg]		5 байт	

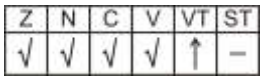
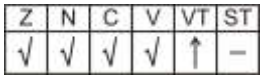
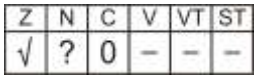
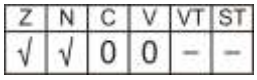
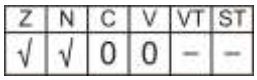
Продолжение таблицы Б.3

1	2	3	4	5
LDBZE	**Загрузка байта с беззнаковым расширением** (DEST, SRC)			
	LDBZE wreg, breg	ACh	3 байта	2 мт
	LDBZE wreg, #data	ADh		1 мт
	LDBZE wreg, [reg]	AEh		3/7 мт
	LDBZE wreg, [reg]+			4/8 мт
	LDBZE wreg, boffset[reg]	AFh	4 байта	4/8 мт
	LDBZE wreg, woffset[reg]		5 байт	
LJMP	**Длинный переход**	Z N C V VT ST – – – – – ?		
	LJMP cadd (длинноиндексная адресация)	E7h	2 байта	1 мт
MUL	**Знаковое умножение слов** (DEST, SRC)	Z N C V VT ST – – – – – ?		
	MUL Ireg, wreg	FE6Ch	4 байта	5 мт
	MUL Ireg, #data	FE6Dh	5 байт	
	MUL Ireg, [reg]	FE6Eh	4 байта	6/10 мт
	MUL Ireg, [reg]+			7/11 мт
	MUL Ireg, boffset[reg]	FE6Fh	5 байт	7/11 мт
	MUL Ireg, woffset[reg]		6 байт	
MUL	**Знаковое умножение слов** (DEST, SRC1, SRC2)	Z N C V VT ST – – – – – ?		
	MUL Ireg, wreg, wreg	FE4Ch	5 байт	5 мт
	MUL Ireg, wreg, #data	FE4Dh	6 байт	
	MUL Ireg, wreg, [reg]	FE4Eh	5 байт	6/10 мт
	MUL Ireg, wreg, [reg]+			7/11 мт
	MUL Ireg, wreg, boffset[reg]	FE4Fh	6 байт	7/11 мт
	MUL Ireg, wreg, woffset[reg]		7 байт	
MULB	**Знаковое умножение байт** (DEST, SRC)	Z N C V VT ST – – – – – ?		
	MULB wreg, breg	FE7Ch	4 байта	4 мт
	MULB wreg, #data	FE7Dh		
	MULB wreg, [reg]	FE7Eh		5/9 мт
	MULB wreg, [reg]+			6/10 мт
	MULB wreg, boffset[reg]	FE7Fh	5 байт	6/10 мт
	MULB wreg, woffset[reg]		6 байт	

Продолжение таблицы Б.3

1	2	3	4	5
MULB	**Знаковое умножение байт** (DEST, SRC1, SRC2) 			
	MULB wreg, breg, breg	FE5Ch	5 байт	4 мт
	MULB wreg, breg, #data	FE5Dh		
	MULB wreg, breg, [reg]	FE5Eh		5/9 мт
	MULB wreg, breg, [reg]+			6/10 мт
	MULB wreg, breg, boffset[reg]	FE5Fh	6 байт	6/10 мт
	MULB wreg, breg, woffset[reg]		7 байт	
MULU	**Беззнаковое умножение слов** (DEST, SRC) 			
	MULU Ireg, wreg	6Ch	3 байта	5 мт
	MULU Ireg, #data	6Dh	4 байта	
	MULU Ireg, [reg]	6Eh	3 байта	6/10 мт
	MULU Ireg, [reg]+			7/11 мт
	MULU Ireg, boffset[reg]	6Fh	4 байта	7/11 мт
	MULU Ireg, woffset[reg]		5 байт	
MULU	**Беззнаковое умножение слов** (DEST, SRC1, SRC2) 			
	MULU Ireg, wreg, breg	4Ch	4 байта	5 мт
	MULU Ireg, wreg, #data	4Dh	5 байт	
	MULU Ireg, wreg, [reg]	4Eh	4 байта	6/10 мт
	MULU Ireg, wreg, [reg]+			7/11 мт
	MULU Ireg, wreg, boffset[reg]	4Fh	5 байт	7/11 мт
	MULU Ireg, wreg, woffset[reg]		6 байт	
MULUB	**Беззнаковое умножение байт** (DEST, SRC) 			
	MULUB wreg, breg	7Ch	3 байта	4 мт
	MULUB wreg, #data	7Dh		
	MULUB wreg, [reg]	7Eh		5/9 мт
	MULUB wreg, [reg]+			6/10 мт
	MULUB wreg, boffset[reg]	7Fh	4 байта	6/10 мт
	MULUB wreg, woffset[reg]		5 байт	
MULUB	**Беззнаковое умножение байт** (DEST, SRC1, SRC2) 			
	MUL wreg, breg, breg	5Ch	4 байта	4 мт
	MUL wreg, breg, #data	5Dh		
	MUL wreg, breg, [reg]	5Eh		5/9 мт
	MUL wreg, breg, [reg]+			6/10 мт
	MUL wreg, breg, boffset[reg]	5Fh	5 байт	6/10 мт
	MUL wreg, breg, woffset[reg]		6 байт	

Продолжение таблицы Б.3

1	2	3	4	5
NEG	**Изменение знака слова**			
	NEG wreg	03h	–	3 мт
NEGB	**Изменение знака байта**			
	NEGB breg	13h	–	3 мт
NOP	**Нет операции**			
	NOP	FDh	1 байт	1 мт
NORML	**Нормализация двойного слова** (SRC, DEST)			
	NORML Ireg, breg	0Fh	3 байта	6 мт
NOT	**Инверсия слова**			
	NOT wreg	02h	–	3 мт
NOTB	**Инверсия байта**			
	NOTB breg	12h	–	3 мт
OR	**Логическое «ИЛИ» слов** (DEST, SRC)			
	OR wreg, wreg	80h	3 байта	4 мт
	OR wreg, #data	81h	4 байта	
	OR wreg, [reg]	82h	3 байта	5/9 мт
	OR wreg, [reg]+			6/10 мт
	OR wreg, boffset[reg]	83h	4 байта	6/10 мт
	OR wreg, woffset[reg]		5 байт	
ORB	**Логическое «ИЛИ» байт** (DEST, SRC)			
	ORB breg, breg	90h	3 байта	4 мт
	ORB breg, #data	91h		
	ORB breg, [reg]	92h		5/9 мт
	ORB breg, [reg]+			6/10 мт
	ORB breg, boffset[reg]	93h	4 байта	6/10 мт
	ORB breg, woffset[reg]		5 байт	

Продолжение таблицы Б.3

1	2	3	4	5
POP	**Чтение слова из стека**			
	POP wreg	CCh	2 байта	3 (7) мт
	POP [reg]	CEh	2 байта	8/13 (4/8) мт
	POP [reg]+			
	POP boffset[reg]	CFh	3 байта	
	POP woffset[reg]		4 байта	
POPA	**Чтение всего из стека**	Z N C V VT ST √ √ √ √ √ √		
	POPA	F5h	1 байт	3 (7) мт
POPF	**Чтение флагов из стека**	Z N C V VT ST √ √ √ √ √ √		
	POPF	F3h	1 байт	1 (5) мт
PUSH	**Загрузка слова в стек**			
	PUSH wreg	C8h	2 байта	2 (6) мт
	PUSH #data	C9h	3 байта	
	PUSH [reg]	CAh	2 байта	3/7 (7/12) мт
	PUSH [reg]+			4/8 (8/13) мт
	PUSH boffset[reg]	CBh	3 байта	4/8 (8/13) мт
	PUSH woffset[reg]		4 байта	
PUSHA	**Загрузка всего в стек**	Z N C V VT ST 0 0 0 0 0 0		
	PUSHA	F4h	1 байт	4 (8) мт
PUSHF	**Загрузка флагов в стек**	Z N C V VT ST 0 0 0 0 0 0		
	PUSHF	F2h	1 байт	2 (6) мт
RET	**Возврат из подпрограммы**			
	RET	F0h	1 байт	2 (4) мт
RST	**Сброс системы**			
	RST	FFh	1 байт	40 мт (включая выборку байта ССВ)

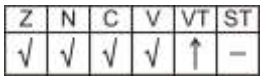
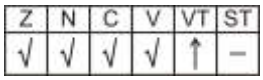
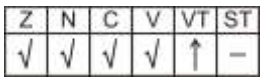
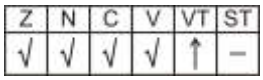
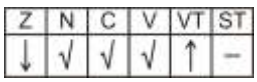
Продолжение таблицы Б.3

1	2	3	4	5
SCALL	**Короткий вызов**			
	SCALL cadd (короткоиндексная адресация)	28h – 2Fh	2 байта	3 (7) мт
SETC	**Установка флага переноса**	Z N C V VT ST – – 1 – – –		
	SETC	F9h	1 байт	1 мт
SHL	**Логический сдвиг слова влево**	Z N C V VT ST √ ? √ √ ↑ –		
	SHL wreg, #count SHL wreg, breg	09h	3 байта	4 мт (3 мт, если сдвига нет)
SHLB	**Логический сдвиг байта влево**	Z N C V VT ST √ ? √ √ ↑ –		
	SHLB breg, #count SHLB breg, breg	19h	3 байта	4 мт (3 мт, если сдвига нет)
SHLL	**Логический сдвиг двойного слова влево**	Z N C V VT ST √ ? √ √ ↑ –		
	SHLL lreg, #count SHLL lreg, breg	0Dh	3 байта	4 мт (3 мт, если сдвига нет)
SHR	**Логический сдвиг слова вправо**	Z N C V VT ST √ 0 √ 0 – √		
	SHR wreg, #count SHR wreg, breg	08h	3 байта	4 мт (3 мт, если сдвига нет)
SHRA	**Арифметический сдвиг слова вправо**	Z N C V VT ST √ √ √ 0 – √		
	SHRA wreg, #count SHRA wreg, breg	0Ah	3 байта	4 мт (3 мт, если сдвига нет)
SHRAB	**Арифметический сдвиг байта вправо**	Z N C V VT ST √ √ √ 0 – √		
	SHRAB breg, #count SHRAB breg, breg	1Ah	3 байта	4 мт (3 мт, если сдвига нет)

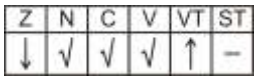
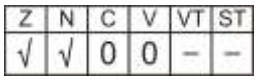
Продолжение таблицы Б.3

1	2	3	4	5
SHRAL	**Арифметический сдвиг двойного слова вправо**	Z N C V VT ST √ √ √ 0 – √		
	SHRAL Ireg, #count SHRAL Ireg, breg	0Eh	3 байта	6 мт (5 мт, если сдвига нет)
SHRB	**Логический сдвиг байта вправо**	Z N C V VT ST √ 0 √ 0 – √		
	SHRB breg, #count SHRB breg, breg	18h	3 байта	4 мт (3 мт, если сдвига нет)
SHRL	**Логический сдвиг двойного слова вправо**	Z N C V VT ST √ 0 √ 0 – √		
	SHRL Ireg, #count SHRL Ireg, breg	0Ch	3 байта	6 мт (5 мт, если сдвига нет)
SJMP	**Короткий переход**			
	SJMP cadd (короткоиндексная адресация)	20h – 27h	2 байта	1 мт
SKIP	**Нет операции (двухбайтная)**			
	SKIP	00h	2 байта	1 мт
ST	**Сохранение слова** (SRC, DEST)			
	ST wreg, wreg	C0h	3 байта	2 мт
	ST wreg, [reg]	C2h		3/7 мт
	ST wreg, [reg]+			4/8 мт
	ST wreg, boffset[reg]	C3h	4 байта	4/8 мт
	ST wreg, woffset[reg]		5 байт	
STB	**Сохранение байта** (SRC, DEST)			
	STB breg, breg	C4h	3 байта	2 мт
	STB breg, [reg]	C6h		3/7 мт
	STB breg, [reg]+			4/8 мт
	STB breg, boffset[reg]	C7h	4 байта	4/8 мт
	STB breg, woffset[reg]		5 байт	

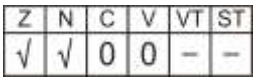
Продолжение таблицы Б.3

1	2	3	4	5
SUB	**Вычитание слова** (DEST, SRC) 			
	SUB wreg, wreg	68h	3 байта	4 мт
	SUB wreg, #data	69h	4 байта	
	SUB wreg, [reg]	6Ah	3 байта	5/9 мт
	SUB wreg, [reg]+			6/10 мт
	SUB wreg, boffset[reg]	6Bh	4 байта	6/10 мт
	SUB wreg, woffset[reg]		5 байт	
SUB	**Вычитание слов** (DEST, SRC1, SRC2) 			
	SUB wreg, wreg, wreg	48h	4 байта	4 мт
	SUB wreg, wreg, #data	49h	5 байт	
	SUB wreg, wreg, [reg]	4Ah	4 байта	5/9 мт
	SUB wreg, wreg, [reg]+			6/10 мт
	SUB wreg, wreg, boffset[reg]	4Bh	5 байт	6/10 мт
	SUB wreg, wreg, woffset[reg]		6 байт	
SUBB	**Вычитание байт** (DEST, SRC) 			
	SUBB breg, breg	78h	3 байта	4 мт
	SUBB breg, #data	79h		
	SUBB breg, [reg]	7Ah		5/9 мт
	SUBB breg, [reg]+			6/10 мт
	SUBB breg, boffset[reg]	7Bh	4 байта	6/10 мт
	SUBB breg, woffset[reg]		5 байт	
SUBB	**Вычитание байт** (DEST, SRC1, SRC2) 			
	SUBB breg, breg, breg	58h	4 байта	4 мт
	SUBB breg, breg, #data	59h		
	SUBB breg, breg, [reg]	5Ah		5/9 мт
	SUBB breg, breg, [reg]+			6/10 мт
	SUBB breg, breg, boffset[reg]	5Bh	5 байт	6/10 мт
	SUBB breg, breg, woffset[reg]		6 байт	
SUBC	**Вычитание слов с переносом** (DEST, SRC) 			
	SUBC wreg, wreg	A8h	3 байта	4 мт
	SUBC wreg, #data	A9h	4 байта	
	SUBC wreg, [reg]	AAh	3 байта	5/9 мт
	SUBC wreg, [reg]+			6/10 мт
	SUBC wreg, boffset[reg]	ABh	4 байта	6/10 мт
	SUBC wreg, woffset[reg]		5 байт	

Продолжение таблицы Б.3

1	2	3	4	5
SUBCB	**Вычитание байт с переносом** (DEST, SRC)			
	SUBCB breg, breg	B8h	4 байта	4 мт
	SUBCB breg, #data	B9h	3 байта	
	SUBCB breg, [reg]	BAh	3 байта	5/9 мт
	SUBCB breg, [reg]+			6/10 мт
	SUBCB breg, boffset[reg]	BBh	4 байта	6/10 мт
	SUBCB breg, woffset[reg]		5 байт	
TIJMP	**Косвенный табличный переход**			
	TIJMP wreg, [wreg], #mask	E2h	4 байта	9 мт (Ex/In) 14 мт (Ex/Ex)
TRAP	**Программное прерывание**			
	TRAP	F7h	1 байт	1 (5) мт
XCH	**Обмен слов** (DEST, SRC)			
	XCH wreg, wreg	04h	3 байта	4 мт
	XCH wreg, boffset[reg]	0Bh	4 байта	6/10 мт
	XCH wreg, woffset[reg]		5 байт	
XCHB	**Обмен байт** (DEST, SRC)			
	XCHB breg, breg	14h	3 байта	4 мт
	XCHB breg, boffset[reg]	1Bh	4 байта	6/10 мт
	XCHB breg, woffset[reg]		5 байт	
XOR	**Логическое исключающее «ИЛИ» слов** (DEST, SRC)			
	XOR wreg, wreg	84h	3 байта	4 мт
	XOR wreg, #data	85h	4 байта	
	XOR wreg, [reg]	86h	3 байта	5/9 мт
	XOR wreg, [reg]+			6/10 мт
	XOR wreg, boffset[reg]	87h	4 байта	6/10 мт
	XOR wreg, woffset[reg]		5 байт	

Окончание таблицы Б.3

1	2	3	4	5
XORB	**Логическое исключающее «ИЛИ» байт** (DEST, SRC)			
	XORB breg, breg	94h	3 байта	4 мт
	XORB breg, #data	95h		
	XORB breg, [reg]	96h		5/9 мт
	XORB breg, [reg]+			6/10 мт
	XORB breg, boffset[reg]	97h	4 байта	6/10 мт
	XORB breg, woffset[reg]		5 байт	

* Время выполнения команды указано в количестве машинных тактов (мт). Время выполнения команды с использованием косвенной и индексной адресации указано двумя цифрами, написанными через слеш. Первое число показывает время выполнения команды с использованием регистров специального назначения или регистров внутреннего ОЗУ с адресами 0000h – 07FFh. Второе число показывает время выполнения команды при работе с внешней памятью. Это время включает в себя время ожидания освобождения контроллера доступа к памяти команд (два машинных такта) и время одной выборки команды, так как приоритет отдается именно им (самая быстрая выборка – еще два машинных такта). Для некоторых команд (LCALL, POP, PUSH, RET и других, использующих стек) указано время выполнения при работе с внутренним стеком, а рядом в скобках – с внешним стеком.

Дополнительная информация о командах микроконтроллера

ADD Dwreg, waop
Dwreg, wreg, waop

Функция: **Сложение слов**

Код: (011001aa) (waop) (Dwreg)
(010001aa) (waop) (wreg) (Dwreg)

Описание: Сложение слова источника SRC и слова приемника DEST и сохранение суммы в приемнике.
Сложение слова первого источника SRC1 и слова второго источника SRC2 и сохранение суммы в приемнике DEST.

Алгоритм: (DEST) ← (DEST) + (SRC)
(DEST) ← (SRC1) + (SRC2)

ADDB Dbreg, baop
Dbreg, breg, baop

Функция: **Сложение байт**

Код: (011101aa) (baop) (Dbreg)
(010101aa) (baop) (breg) (Dbreg)

Описание: Сложение байта источника SRC и байта приемника DEST и сохранение суммы в приемнике.
Сложение байта первого источника SRC1 и байта второго источника SRC2 и сохранение суммы в приемнике DEST.

Алгоритм: (DEST) ← (DEST) + (SRC)
(DEST) ← (SRC1) + (SRC2)

ADDC Dwreg, waop

Функция: **Сложение слов с переносом**

Код: (101001aa) (waop) (Dwreg)

Описание: Сложение слова источника SRC, слова приемника DEST и флага переноса C и сохранение суммы в приемнике.

Алгоритм: (DEST) ← (DEST) + (SRC) + (C)

ADDCB Dbreg, baop

Функция: **Сложение слов с переносом**

Код: (101101aa) (baop) (Dbreg)

Описание: Сложение байта источника SRC, байта приемника DEST и флага переноса C и сохранение суммы в приемнике.

Алгоритм: (DEST) ← (DEST) + (SRC) + (C)

AND Dwreg, waop
Dwreg, wreg, waop

Функция: **Логическое «И» слов**

Код: (011000aa) (waop) (Dwreg)
(010000aa) (waop) (wreg) (Dwreg)

Описание: Логическое умножение слова источника SRC и слова приемника DEST и сохранение результата в приемнике.
Логическое умножение слова первого источника SRC1 и слова второго источника SRC2 и сохранение результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) AND (SRC)
(DEST) ← (SRC1) AND (SRC2)

ANDB Dbreg, baop
Dbreg, breg, baop

Функция: **Логическое «И» байт**

Код: (011100aa) (baop) (Dbreg)
(010100aa) (baop) (breg) (Dbreg)

Описание: Логическое умножение байта источника SRC и байта приемника DEST и сохранение результата в приемнике.
Логическое умножение байта первого источника SRC1 и байта второго источника SRC2 и сохранение результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) AND (SRC)
(DEST) ← (SRC1) AND (SRC2)

BMOV DIreg, wreg

Функция: **Непрерываемое перемещение блоков данных**

Код: (11000001) (wreg) (DIreg)

Описание: Перемещение блоков слов данных из одной зоны памяти в другую. Важно помнить, что никакое прерывание не будет обслужено до окончания выполнения команды BMOV, поэтому при пересылке больших массивов данных лучше использовать команду BMOVI. Адреса источника и приемника вычисляются при помощи режима косвенной адресации с автоинкрементом. Длинный регистр PTRS содержит указатели источника SRCPTR и приемника DSTPTR, которые хранятся в соседних регистрах слов. Блоки данных могут находиться в любом месте памяти, но не должны перекрываться. Регистр числа передач CNTREG определяет количество перемещений. Регистр CNTREG не декрементируется в процессе выполнения команды BMOV.

Алгоритм:

COUNT ← (CNTREG)
LOOP: SRCPTR ← (PTRS)
DSTPTR ← (PTRS+2)
(DSTPTR) ← (SRCPTR)
(PTRS) ← SRCPTR+2
(PTSR+2) ← DSTPTR+2
COUNT ← COUNT – 1
Если COUNT ≠ 0, тогда переход на LOOP

BMOVI DIreg, wreg

Функция: **Прерываемое перемещение блоков данных**

Код: (11001101) (wreg) (DIreg)

Описание: Перемещение блоков слов данных из одной зоны памяти в другую. Команда идентична команде BMOV, за исключением того, что она является прерываемой. Регистр числа передач CNTREG определяет количество перемещений. Регистр CNTREG изменяет свое значение только в том случае, если выполнение команды BMOVI прерывается. После обслуживания прерывания регистр CNTREG содержит число оставшихся пересылок, поэтому перед повторным вызовом команды BMOVI, этот регистр должен быть проинициализирован.

Алгоритм: Идентичен алгоритму команды BMOV

BR [Dwreg]

Функция: **Косвенный переход**

Код: (11100011) (Dwreg)

Описание: Загрузка в программный счетчик PC адреса, косвенно задаваемого операндом, и продолжение выполнения программы с этого адреса.

Алгоритм: PC ← (DEST)

CLR Dwreg

Функция: **Очистка слова**

Код: (00000001) (Dwreg)

Описание: Запись нулей в приемник DEST.

Алгоритм: (DEST) ← 0000h

CLRB Dbreg

Функция: **Очистка байта**

Код: (00010001) (Dbreg)

Описание: Запись нулей в младший байт приемника DEST.

Алгоритм: (DEST (младший байт)) ← 00h

CLRC

Функция: **Очистка флага переноса**

Код: (11111000)

Описание: Запись нуля в бит C регистра PSW.

Алгоритм: C ← 0b

CLRVT

Функция: **Очистка дополнительного флага переполнения**

Код: (11111100)

Описание: Запись нуля в бит VT регистра PSW.

Алгоритм: VT ← 0b

CMP Dwreg, waop

Функция: **Сравнение слов**

Код: (100010aa) (waop) (Dwreg)

Описание: Вычитание слова источника SRC из слова приемника DEST. Флаги в PSW устанавливаются, но значения операндов остаются прежними. Если возникает заем, то флаг переноса сбрасывается, в противном случае устанавливается.

Алгоритм: (DEST) – (SRC)

CMPB Dbreg, baop

Функция: **Сравнение байт**

Код: (100110aa) (baop) (Dbreg)

Описание: Вычитание байта источника SRC из байта приемника DEST. Флаги в PSW устанавливаются, но значения операндов остаются прежними. Если возникает заем, то флаг переноса сбрасывается, в противном случае устанавливается.

Алгоритм: (DEST) – (SRC)

CMPL DIreg, Ireg

Функция: **Сравнение длинных слов**

Код: (11000101) (Ireg) (DIreg)

Описание: Вычитание двойного слова источника SRC из двойного слова приемника DEST. Операнды определяются с использованием режима прямой адресации. Флаги в PSW устанавливаются, но значения операндов остаются прежними. Если возникает заем, то флаг переноса сбрасывается, в противном случае устанавливается.

Алгоритм: (DEST) – (SRC)

DEC Dwreg

Функция: **Декремент слова**

Код: (00000101) (Dwreg)

Описание: Уменьшение величины операнда на единицу.

Алгоритм: (DEST) ← (DEST) – 1

DECB Dbreg

Функция: **Декремент байта**

Код: (00010101) (Dbreg)

Описание: Уменьшение величины операнда на единицу.

Алгоритм: (DEST) ← (DEST) – 1

DI

Функция: **Запрещение всех прерываний**

Код: (11111010)

Описание: Запрещаются все прерывания. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: Бит I регистра PSW ← 0

DIV DIreg, waop

Функция: **Знаковое деление слов**

Код: (11111110) (100011aa) (waop) (DIreg)

Описание: Деление целого двойного слова приемника DEST на целое слово источника SRC, с использованием знаковой арифметики. Частное сохраняется в младшем слове (т. е. в слове с меньшим адресом) приемника, а остаток – в старшем слове.

Алгоритм: (DEST (младшее слово)) ← (DEST)/(SRC)
(DEST (старшее слово))← Остаток от (DEST)/(SRC)

DIVB Dwreg, baop

Функция: **Знаковое деление байт**

Код: (11111110) (100111aa) (baop) (Dwreg)

Описание: Деление целого слова приемника DEST на целый байт источника SRC, с использованием знаковой арифметики. Частное сохраняется в младшем байте (т. е. в байте с меньшим адресом) приемника, а остаток – в старшем байте.

Алгоритм: (DEST (младший байт)) ← (DEST)/(SRC)
(DEST (старший байт))← Остаток от (DEST)/(SRC)

DIVU DIreg, waop

Функция: **Беззнаковое деление слов**

Код: (100011aa) (waop) (DIreg)

Описание: Деление целого двойного слова приемника DEST на целое слово источника SRC, с использованием беззнаковой арифметики. Частное сохраняется в младшем слове (т. е. в слове с меньшим адресом) приемника, а остаток – в старшем слове.

Алгоритм: (DEST (младшее слово)) ← (DEST)/(SRC)
(DEST (старшее слово))← Остаток от (DEST)/(SRC)
Обе операции выполняются одновременно

DIVUB Dwreg, baop

Функция: **Беззнаковое деление байт**

Код: (100111aa) (baop) (Dwreg)

Описание: Деление целого слова приемника DEST на целый байт источника SRC, с использованием беззнаковой арифметики. Частное сохраняется в младшем байте (т. е. в байте с меньшим адресом) приемника, а остаток – в старшем байте.

Алгоритм: (DEST (младший байт)) ← (DEST)/(SRC)
(DEST (старший байт))← Остаток от (DEST)/(SRC)
Обе операции выполняются одновременно

DJNZ Dbreg, cadd

Функция: **Декремент байта и переход при отсутствии нуля**

Код: (11100000) (Dbreg) (disp)

Описание: Уменьшение значения байта на единицу и в случае неравенства результата нулю – переход на указанный адрес (метку). Если результат равен нулю, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды DJNZ и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: (COUNT) ← (COUNT) – 1
Если (COUNT) ≠ 0, тогда PC ← PC + disp

DJNZW Dwreg, cadd

Функция: **Декремент слова и переход при отсутствии нуля**

Код: (11100001) (Dwreg) (disp)

Описание: Уменьшение значения слова на единицу и, в случае неравенства результата нулю, переход на указанный адрес (метку). Если результат равен нулю, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: (COUNT) ← (COUNT) – 1
Если (COUNT) ≠ 0, тогда PC ← PC + disp

DPTS

Функция: **Запрет PTS**

Код: (11101100)

Описание: Блокирование сервера периферийных транзакций PTS.

Алгоритм: Бит PSE регистра PSW ← 0

EI

Функция: **Разрешение всех прерываний**

Код: (11111011)

Описание: Разрешаются все прерывания. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: Бит I регистра PSW ← 1

EPTS

Функция: **Разрешение PTS**

Код: (11101101)

Описание: Деблокирование сервера периферийных транзакций PTS.

Алгоритм: Бит PSE регистра PSW ← 1

EXT DIreg

Функция: **Знаковое расширение слова до двойного слова**

Код: (00000110) (DIreg)

Описание: Знаковое расширение младшего слова операнда до двойного слова.

Алгоритм: Если (DEST (младшее слово)) < 8000h, тогда
(DEST (старшее слово)) ← 0000h,
иначе
(DEST (старшее слово)) ← FFFFh

EXTB Dwreg

Функция: **Знаковое расширение байта до слова**

Код: (00010110) (Dwreg)

Описание: Знаковое расширение младшего байта операнда до слова

Алгоритм: Если (DEST (младший байт)) < 80h, тогда
(DEST (старший байт)) ← 00h,
иначе
(DEST (старший байт)) ← FFh

IDLPD #key

Функция: **Включение режима Idle или Powerdown**

Код: (11110110) (key)

Описание: Результат выполнения этой команды зависит от значения 8-битной величины операнда (key). Контроллер шины завершает цикл упреждающей выборки перед остановкой ЦПУ или сбросом. Следует помнить, что при неправильном задании величины операнда key, все флаги регистра PSW будут сброшены.

Алгоритм: Если (key) = 1, тогда вход в режим Idle,
иначе
если (key) = 2, тогда вход в режим Powerdown,
иначе
сброс микроконтроллера

INC Dwreg

Функция: **Инкремент слова**

Код: (00000111) (Dwreg)

Описание: Увеличение значения слова операнда на единицу.

Алгоритм: (DEST) ← (DEST) + 1

INCB Dbreg

Функция: **Инкремент байта**

Код: (00010111) (Dbreg)

Описание: Увеличение значения байта операнда на единицу.

Алгоритм: (DEST) ← (DEST) + 1

JBC Dbreg, bitno, cadd

Функция: **Переход, если бит очищен**

Код: (00110bbb) (Dbreg) (disp)

Описание: Тестирование бита с номером, задаваемым операндом (bitno) в указанном байте (Dbreg) и, в случае, если бит очищен, переход на указанный адрес (метку). Если бит установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127. Три младших бита (bbb) опкода команды – номер бита.

Алгоритм: Если бит = 0b, тогда PC ← PC + disp

JBS Dbreg, bitno, cadd

Функция: **Переход, если бит установлен**

Код: (00111bbb) (Dbreg) (disp)

Описание: Тестирование бита с номером, задаваемым операндом (bitno) в указанном байте (Dbreg) и, в случае если бит установлен, переход на указанный адрес (метку). Если бит очищен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127. Три младших бита (bbb) опкода команды – номер бита.

Алгоритм: Если бит = 1b, тогда PC ← PC + disp

JC cadd

Функция: **Переход, если C = 1**

Код: (11011011) (disp)

Описание: Тестирование флага переноса C в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг очищен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если C = 1b, тогда PC ← PC + disp

JE cadd

Функция: **Переход, если Z = 1**
(Переход, если равно)

Код: (11011111) (disp)

Описание: Тестирование флага нуля Z в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг очищен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если Z = 1b, тогда PC ← PC + disp

JGE cadd

Функция: **Переход, если N = 0**
(Переход, если число со знаком больше или равно)

Код: (11010110) (disp)

Описание: Тестирование флага отрицательного результата N в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если N = 0b, тогда PC ← PC + disp

JGT cadd

Функция: **Переход, если N = 0 и Z = 0**
(Переход, если число со знаком больше)

Код: (11010010) (disp)

Описание: Тестирование флагов отрицательного результата N и нуля Z в регистре PSW и, в случае, если оба флага сброшены, переход на указанный адрес (метку). Если хотя бы один из двух флагов установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если N = 0b и Z = 0b, тогда PC ← PC + disp

JH cadd

Функция: **Переход, если C = 1 и Z = 0**
(Переход, если беззнаковое число больше)

Код: (11011001) (disp)

Описание: Тестирование флагов переноса C и нуля Z в регистре PSW и, в случае, если флаг C установлен, а флаг Z сброшен, переход на указанный адрес (метку). В противном случае управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если C = 1b и Z = 0b, тогда PC ← PC + disp

JLE cadd

Функция: **Переход, если N = 1 и Z = 1**
(Переход, если число со знаком меньше или равно)

Код: (11011010) (disp)

Описание: Тестирование флагов отрицательного результата N и нуля Z в регистре PSW и, в случае, если оба флага установлены, переход на указанный адрес (метку). Если хотя бы один из двух флагов сброшен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если N = 1b и Z = 1b, тогда PC ← PC + disp

JLT cadd

Функция: **Переход, если N = 1**
(Переход, если число со знаком меньше)

Код: (11011110) (disp)

Описание: Тестирование флага отрицательного результата N в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг очищен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если N = 1b, тогда PC ← PC + disp

JNC cadd

Функция: **Переход, если C = 0**

Код: (11010011) (disp)

Описание: Тестирование флага переноса C в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если C = 0b, тогда PC ← PC + disp

JNE cadd

Функция: **Переход, если Z = 0**
(Переход, если не равно)

Код: (11010111) (disp)

Описание: Тестирование флага нуля Z в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если Z = 0b, тогда PC ← PC + disp

JNH cadd

Функция: **Переход, если C = 0 и Z = 1**
(Переход, если беззнаковое число не больше)

Код: (11010001) (disp)

Описание: Тестирование флагов переноса C и нуля Z в регистре PSW и, в случае если флаг C сброшен, а флаг Z установлен, переход на указанный адрес (метку). В противном случае управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если C = 0b и Z = 1b, тогда PC ← PC + disp

JNST cadd

Функция: **Переход, если ST = 0**

Код: (11010000) (disp)

Описание: Тестирование дополнительного флага переноса ST в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если ST = 0b, тогда PC ← PC + disp

JNV cadd

Функция: **Переход, если V = 0**

Код: (11010101) (disp)

Описание: Тестирование флага переполнения V в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если V = 0b, тогда PC ← PC + disp

JNVT cadd

Функция: **Переход, если VT = 0**

Код: (11010100) (disp)

Описание: Тестирование дополнительного флага переполнения VT в регистре PSW и, в случае, если флаг сброшен, переход на указанный адрес (метку). Если флаг установлен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если VT = 0b, тогда PC ← PC + disp

JST cadd

Функция: **Переход, если ST = 1**

Код: (11011000) (disp)

Описание: Тестирование дополнительного флага переноса ST в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг сброшен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если ST = 1b, тогда PC ← PC + disp

JV cadd

Функция: **Переход, если V = 1**

Код: (11011101) (disp)

Описание: Тестирование флага переполнения V в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг сброшен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если V = 1b, тогда PC ← PC + disp

JVT cadd

Функция: **Переход, если VT = 1**

Код: (11011100) (disp)

Описание: Тестирование дополнительного флага переполнения VT в регистре PSW и, в случае, если флаг установлен, переход на указанный адрес (метку). Если флаг сброшен, то управление передается следующей по порядку команде. Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение (disp) между концом команды и адресом метки перехода. Смещение может быть в диапазоне от –128 до +127.

Алгоритм: Если VT = 1b, тогда PC ← PC + disp

LCALL cadd

Функция: **Длинный вызов**

Код: (11101111) (disp-low) (disp-high)

Описание: Вызов подпрограммы, расположенной по указанному адресу (метке). Содержимое программного счетчика загружается в стек (адрес возврата из подпрограммы), после чего к значению программного счетчика PC добавляется смещение, задаваемое двумя байтами ((disp-low) (disp-high)), равное промежутку от конца команды до метки перехода. Операнд может иметь любой адрес во всем адресном пространстве. Подпрограмма должна заканчиваться командой RET.

Алгоритм: SP ← SP – 2
(SP) ← PC
PC ← PC + ((disp-high) (disp-low))

LD Dwreg, waop

Функция: **Загрузка слова**

Код: (101000aa) (waop) (Dwreg)

Описание: Загрузка значения слова источника SRC в слово приемника DEST.

Алгоритм: (DEST) ← (SRC)

LDB Dbreg, baop

Функция: **Загрузка байта**

Код: (101100aa) (baop) (Dbreg)

Описание: Загрузка значения байта источника SRC в байт приемника DEST.

Алгоритм: (DEST) ← (SRC)

LDBSE Dwreg, baop

Функция: **Загрузка байта со знаковым расширением**

Код: (101111aa) (baop) (Dwreg)

Описание: Знаковое расширение байта источника SRC и загрузка его в слово приемника DEST.

Алгоритм: (DEST (младший байт)) ← (SRC)
Если (SRC) < 80h, тогда
(DEST (старший байт)) ← 00h,
иначе
(DEST (старший байт)) ← FFh

LDBZE Dwreg, baop

Функция: **Загрузка байта с беззнаковым расширением**

Код: (101011aa) (baop) (Dwreg)

Описание: Беззнаковое расширение байта источника SRC и загрузка его в слово приемника DEST.

Алгоритм: (DEST (младший байт)) ← (SRC)
(DEST (старший байт)) ← 00h

LJMP cadd

Функция: **Длинный переход**

Код: (11100111) (disp-low) (disp-high)

Описание: Переход по указанному адресу (метке). Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение, задаваемое двумя байтами ((disp-low) (disp-high)), равное промежутку от конца команды до метки перехода. Операнд может иметь любой адрес во всем адресном пространстве.

Алгоритм: PC ← PC + ((disp-high)(disp-low))

MUL DIreg, waop
DIreg, wreg, waop

Функция: **Знаковое умножение слов**

Код: (11111110) (010111aa) (waop) (DIreg)
(11111110) (010011aa) (waop) (wreg) (DIreg)

Описание: Перемножение двойного слова приемника DEST и слова источника SRC с использованием знаковой арифметики и размещение двойного слова результата в приемнике DEST.
Перемножение слова первого источника SRC1 и слова второго источника SRC2 с использованием знаковой арифметики и размещение двойного слова результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) × (SRC)
(DEST) ← (SRC1) × (SRC2)

MULB Dwreg, baop
Dwreg, breg, baop

Функция: **Знаковое умножение байт**

Код: (11111110) (011111aa) (baop) (Dwreg)
(11111110) (010111aa) (baop) (breg) (Dwreg)

Описание: Перемножение слова приемника DEST и байта источника SRC с использованием знаковой арифметики и размещение слова результата в приемнике DEST.
Перемножение байта первого источника SRC1 и байта второго источника SRC2 с использованием знаковой арифметики и размещение слова результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) × (SRC)
(DEST) ← (SRC1) × (SRC2)

MULU DIreg, waop
DIreg, wreg, waop

Функция: **Беззнаковое умножение слов**

Код: (011011aa) (waop) (DIreg)
(010011aa) (waop) (wreg) (DIreg)

Описание: Перемножение двойного слова приемника DEST и слова источника SRC с использованием беззнаковой арифметики и размещение двойного слова результата в приемнике DEST.
Перемножение слова первого источника SRC1 и слова второго источника SRC2 с использованием беззнаковой арифметики и размещение двойного слова результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) × (SRC)
(DEST) ← (SRC1) × (SRC2)

MULUB Dwreg, baop
Dwreg, breg, baop

Функция: **Беззнаковое умножение байт**

Код: (011111aa) (baop) (Dwreg)
(010111aa) (baop) (breg) (Dwreg)

Описание: Перемножение слова приемника DEST и байта источника SRC с использованием беззнаковой арифметики и размещение слова результата в приемнике DEST.
Перемножение байта первого источника SRC1 и байта второго источника SRC2 с использованием беззнаковой арифметики и размещение слова результата в приемнике DEST.

Алгоритм: (DEST) ← (DEST) × (SRC)
(DEST) ← (SRC1) × (SRC2)

NEG Dwreg

Функция: **Изменение знака слова**

Код: (00000011) (Dwreg)

Описание: Изменение знака слова операнда на противоположный.

Алгоритм: (DEST) ← – (DEST)

NEGB Dbreg

Функция: **Изменение знака байта**

Код: (00010011) (Dbreg)

Описание: Изменение знака байта операнда на противоположный.

Алгоритм: (DEST) ← – (DEST)

NOP

Функция: **Нет операции**

Код: (11111101)

Описание: Пустая однобайтная команда. Управление передается следующей по порядку команде.

Алгоритм: PC ← PC + 1

NORML SIreg, Dbreg

Функция: **Нормализация двойного слова**

Код: (00001111) (Dbreg) (SIreg)

Описание: Сдвиг двойного слова источника SRC влево до тех пор, пока его старший значащий бит не окажется равным единице или пока не будет совершен 31 сдвиг. Если после 31 сдвига в старшем значащем бите остался ноль, то процесс сдвига прекращается и устанавливается флаг нуля Z в регистре PSW. В приемнике DEST сохраняется число совершенных сдвигов.

Алгоритм: (COUNT) ← 00h
Пока (старший бит SRC = 0b) и (COUNT)<31)
Выполнять
(SRC) ← (SRC) × 2
(COUNT) ← (COUNT) + 1
По окончании (DEST) ← (COUNT)

NOT Dwreg

Функция: **Инверсия слова**

Код: (00000010) (Dwreg)

Описание: Побитное инвертирование слова (каждая единица меняется на ноль, каждый ноль меняется на единицу).

Алгоритм: (DEST) ← NOT (DEST)

NOTB Dbreg

Функция: **Инверсия байта**

Код: (00010010) (Dbreg)

Описание: Побитное инвертирование байта (каждая единица меняется на ноль, каждый ноль меняется на единицу).

Алгоритм: (DEST) ← NOT (DEST)

OR Dwreg, waop

Функция: **Логическое «ИЛИ» слов**

Код: (100000aa) (waop) (Dwreg)

Описание: Логическое сложение слова источника SRC и слова приемника DEST и сохранение результата в приемнике.

Алгоритм: (DEST) ← (DEST) OR (SRC)

ORB Dbreg, baop

Функция: **Логическое «ИЛИ» байт**

Код: (100100aa) (baop) (Dbreg)

Описание: Логическое сложение байта источника SRC и байта приемника DEST и сохранение результата в приемнике.

Алгоритм: (DEST) ← (DEST) OR (SRC)

POP waop

Функция: **Чтение слова из стека**

Код: (110011aa) (waop)

Описание: Чтение слова из вершины стека и размещение его в приемнике DEST.

Алгоритм: (DEST) ← (SP)
SP ← SP + 2

POPA

Функция: **Чтение всего стека**

Код: (11110101)

Описание: Команда используется взамен команды POPF для поддержки восьми добавочных прерываний. Два слова читаются из стека и размещаются в сдвоенных регистрах INT_MASK1/WSR (первое слово) и PSW/INT_MASK (второе слово). Указатель стека SP инкрементируется на четыре. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: INT_MASK1/WSR ← (SP)
SP ← SP + 2
PSW/INT_MASK ← (SP)
SP ← SP + 2

POPF

Функция: **Чтение флагов из стека**

Код: (11110011)

Описание: Чтение слова из вершины стека и размещение его в регистре PSW. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: (PSW) ← (SP)
SP ← SP + 2

PUSH waop

Функция: **Загрузка слова в стек**

Код: (110010aa) (waop)

Описание: Загрузка значения операнда в стек.

Алгоритм: (SP) ← SP – 2
(SP) ← (DEST)

PUSHA

Функция: **Загрузка всего в стек**

Код: (11110100)

Описание: Команда используется взамен команды PUSHF для поддержки восьми добавочных прерываний. Два слова загружаются в стек из сдвоенных регистров PSW/INT_MASK (первое слово) и INT_MASK1/WSR (второе слово), после чего регистры PSW, INT_MASK и INT_MASK1 очищаются. Указатель стека SP декрементируется на четыре. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: SP ← SP – 2
(SP) ← PSW/INT_MASK
PSW/INT_MASK ← 0000h
SP ← SP – 2
(SP) ← INT_MASK1/WSR
INT_MASK1 ← 00h

PUSHF

Функция: **Загрузка флагов в стек**

Код: (11110010)

Описание: Загрузка значения сдвоенного регистра PSW/INT_MASK в вершину стека и очистка регистра PSW/INT_MASK. Все прерывания автоматически запрещаются. Запросы на прерывания не могут идти сразу после этой команды.

Алгоритм: SP ← SP – 2
(SP) ← PSW/INT_MASK
PSW/INT_MASK ← 0000h

RET

Функция: **Возврат из подпрограммы**

Код: (11110000)

Описание: Считывание значения адреса (адреса возврата) из вершины стека и размещение его в программном счетчике PC. Указатель стека SP инкрементируется на два.

Алгоритм: PC ← (SP)
SP ← SP + 2

RST

Функция: **Сброс системы**

Код: (11111111)

Описание: Очистка регистра PSW, загрузка в программный счетчик PC значения 2080h, загрузка в регистры SFR их начальных значений. В результате выполнения этой команды вывод RESET# микроконтроллера находится в состоянии логического нуля 16 машинных тактов.

Алгоритм: PSW ← 00h
PC ← 2080h
SFR ← начальное значение SFR
Вывод #RESET в состояние логического нуля

SCALL cadd

Функция: **Короткий вызов**

Код: (00101xxx) (disp)

Описание: Вызов подпрограммы, расположенной по указанному адресу (метке). Содержимое программного счетчика загружается в стек (адрес возврата из подпрограммы), после чего к значению программного счетчика PC добавляется смещение, задаваемое байтом (disp) и тремя младшими битами (xxx) кода команды, равное промежутку от конца команды до метки перехода. Смещение может быть в диапазоне от –1024 до +1023 включительно. Подпрограмма должна заканчиваться командой RET.

Алгоритм: SP ← SP – 2
(SP) ← PC
PC ← PC + ((xxx)(disp))

SETC

Функция: **Установка флага переноса**

Код: (11111001)

Описание: Запись единицы в бит C регистра PSW.

Алгоритм: C ← 1b

SHL Dwreg, scount

Функция: **Логический сдвиг слова влево**

Код: (00001001) (scount) (Dwreg)

Описание: Сдвиг слова приемника DEST влево столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный старший бит операнда DEST
        (DEST) ← (DEST) × 2
        Temp ← Temp – 1
```

SHLB Dbreg, scount

Функция: **Логический сдвиг байта влево**

Код: (00011001) (scount) (Dbreg)

Описание: Сдвиг байта приемника DEST влево столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный старший бит операнда DEST
        (DEST) ← (DEST) × 2
        Temp ← Temp – 1
```

SHLL DIreg, scount

Функция: **Логический сдвиг двойного слова влево**

Код: (00001101) (scount) (DIreg)

Описание: Сдвиг двойного слова приемника DEST влево столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.

Алгоритм:
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
    C ← Вытесненный старший бит операнда DEST
    (DEST) ← (DEST) × 2
    Temp ← Temp – 1

SHR Dwreg, scount

Функция: **Логический сдвиг слова вправо**

Код: (00001000) (scount) (Dwreg)

Описание: Сдвиг слова приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм:
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
    C ← Вытесненный младший бит операнда DEST
    (DEST) ← (DEST) / 2
    Temp ← Temp – 1

SHRA Dwreg, scount

Функция: **Арифметический сдвиг слова вправо**

Код: (00001010) (scount) (Dwreg)

Описание: Сдвиг слова приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями, если старший значащий бит исходного слова был ноль, и – единицами в противном случае. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный младший бит операнда DEST
        (DEST) ← (DEST) / 2
        Temp ← Temp – 1
```

SHRAB Dbreg, scount

Функция: **Арифметический сдвиг байта вправо**

Код: (00011010) (scount) (Dbreg)

Описание: Сдвиг байта приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями, если старший значащий бит исходного байта был ноль, и единицами в противном случае. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный младший бит операнда DEST
        (DEST) ← (DEST) / 2
        Temp ← Temp – 1
```

SHRAL Dwreg, scount

Функция: **Арифметический сдвиг двойного слова вправо**

Код: (00001110) (scount) (Dwreg)

Описание: Сдвиг двойного слова приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями, если старший значащий бит исходного двойного слова был ноль, и единицами в противном случае. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный младший бит операнда DEST
        (DEST) ← (DEST) / 2
        Temp ← Temp – 1
```

SHRB Dbreg, scount

Функция: **Логический сдвиг байта вправо**

Код: (00011000) (scount) (Dbreg)

Описание: Сдвиг байта приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм:
```
Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
        C ← Вытесненный младший бит операнда DEST
        (DEST) ← (DEST) / 2
        Temp ← Temp – 1
```

SHRL DIreg, scount

Функция: **Логический сдвиг двойного слова вправо**

Код: (00001100) (scount) (DIreg)

Описание: Сдвиг двойного слова приемника DEST вправо столько раз, сколько установлено операндом счетчиком (правый операнд). Освобождающиеся биты заполняются нулями. Последний вытесненный бит остается в переносе C. Количество сдвигов может быть от 0 до 15 и задано непосредственно или – от 0 до 31 и задано прямо, как содержимое какого-либо регистра.
В начале своего выполнения команда очищает дополнительный флаг переноса ST в регистре PSW. Если в течение сдвига в перенос C попадает единица, флаг ST устанавливается.

Алгоритм: Temp ← (COUNT)
Пока Temp ≠ 0
Выполнять
C ← Вытесненный младший бит операнда DEST
(DEST) ← (DEST) / 2
Temp ← Temp – 1

SJMP cadd

Функция: **Короткий переход**

Код: (00100xxx) (disp)

Описание: Переход по указанному адресу (метке). Для вычисления перехода к текущему значению программного счетчика PC добавляется смещение, задаваемое байтом (disp) и тремя младшими битами (xxx) кода команды, равное промежутку от конца команды до метки перехода. Смещение может быть в диапазоне от –1024 до +1023 включительно.

Алгоритм: PC ← PC + ((xxx)(disp))

SKIP breg

Функция: **Нет операции**

Код: (00000000) (breg)

Описание: Пустая двухбайтная команда. Управление передается следующей по порядку команде. Значение операнда не важно.

Алгоритм: PC ← PC + 2

ST Swreg, waop

Функция: **Сохранение слова**

Код: (110000aa) (waop) (Swreg)

Описание: Сохранение слова источника SRC в слове приемника DEST.

Алгоритм: (DEST) ← (SRC)

STB Sbreg, baop

Функция: **Сохранение байта**

Код: (110001aa) (baop) (Sbreg)

Описание: Сохранение байта источника SRC в байте приемника DEST.

Алгоритм: (DEST) ← (SRC)

SUB Dwreg, waop
Dwreg, wreg, waop

Функция: **Вычитание слов**

Код: (011010aa) (waop) (Dwreg)
(011010aa) (waop) (wreg) (Dwreg)

Описание: Вычитание слова источника SRC из слова приемника DEST и сохранение результата в приемнике. При заеме устанавливается флаг переноса C в регистре PSW.
Вычитание слова второго источника SRC2 из слова первого источника SRC1 и сохранение результата в приемнике DEST. При заеме устанавливается флаг переноса C в регистре PSW.

Алгоритм: (DEST) ← (DEST) – (SRC)
(DEST) ← (SRC1) – (SRC2)

SUBB Dbreg, baop
Dbreg, breg, baop

Функция: **Вычитание байт**

Код: (010110aa) (baop) (Dbreg)
(010110aa) (baop) (breg) (Dbreg)

Описание: Вычитание байта источника SRC из байта приемника DEST и сохранение результата в приемнике. При заеме устанавливается флаг переноса C в регистре PSW.
Вычитание байта второго источника SRC2 из байта первого источника SRC1 и сохранение результата в приемнике DEST. При заеме устанавливается флаг переноса C в регистре PSW.

Алгоритм: (DEST) ← (DEST) – (SRC)
(DEST) ← (SRC1) – (SRC2)

SUBC Dwreg, waop

Функция: **Вычитание слов с переносом**

Код: (101010aa) (waop) (Dwreg)

Описание: Вычитание слова источника SRC и флага переноса из слова приемника DEST и сохранение результата в приемнике. При заеме устанавливается флаг переноса C в регистре PSW.

Алгоритм: (DEST) ← (DEST) – (SRC) – (C)

SUBCB Dbreg, baop

Функция: **Вычитание байт с переносом**

Код: (101110aa) (baop) (Dbreg)

Описание: Вычитание байта источника SRC и флага переноса из байта приемника DEST и сохранение результата в приемнике. При заеме устанавливается флаг переноса C в регистре PSW.

Алгоритм: (DEST) ← (DEST) – (SRC) – (C)

TIJMP tbase, [index], #mask

Функция: **Косвенный табличный переход**

Код: (11100010) (index) (mask) (tbase)

Описание: Продолжение выполнения программы по адресу, выбранному из таблицы адресов переходов. Операнд (tbase) содержит 16-битный адрес начала таблицы переходов. Регистр этого адреса может располагаться в диапазоне адресов до 0FEh без работы с окнами и выше 0FFh при оконной адресации. Таблица адресов переходов может размещаться на странице памяти размером до 0FFh в любой свободной области, выровненной по границе слова.
Операнд ([index]) содержит 16-битный адрес байта, содержащий 7-битный индекс перехода. Регистр для хранения значения (index) может быть в диапазоне адресов до 0FEh без работы с окнами и выше 0FFh при оконной адресации.
Байтовый непосредственный операнд (#mask) содержит 7-битную маску, которая накладывается по «И» на значение (index) при расчете величины смещения OFFSET.
Адрес вектора перехода DEST X определяется путем добавления к адресу начала таблицы переходов удвоенного значения смещения.
Для задания таблицы адресов переходов следует использовать псевдокоманду ассемблера DCW. При этом автоматически выполняется выравнивание таблицы по границе слова.

Алгоритм: OFFSET ← (index) AND (mask)
DEST X ← tbase + (2×OFFSET)
PC ← (DEST X)

TRAP

Функция: **Программное прерывание**

Код: (11110111)

Описание: Выполнение этой команды эквивалентно вызову процедуры обслуживания прерывания по вектору 2010h и не зависит от состояния флага I в регистре PSW. Запросы на прерывания не могут идти сразу после этой команды.
Следует помнить, что команда TRAP не поддерживается транслятором с языка ассемблер. Она используется исключительно в инструментальных системах разработки, в частности в мониторах-отладчиках для реализации режимов выполнения пользовательских программ с точками останова или в пошаговом режиме. Попытка ввода этой команды на языке ассемблера приведет к сообщению транслятора о синтаксической ошибке.

Алгоритм: SP ← SP – 2
(SP) ← PC
PC ← (2010h)

XCH Dwreg, waop

Функция: **Обмен слов**

Код: (00000100) (waop) (Dwreg)
(00001011) (waop) (Dwreg)

Описание: Обмен словами между источником SRC и приемником DEST.

Алгоритм: (DEST) ↔ (SRC)

XCHB Dbreg, baop

Функция: **Обмен байт**

Код: (00010100) (baop) (Dbreg)
(00011011) (baop) (Dbreg)

Описание: Обмен байтами между источником SRC и приемником DEST.

Алгоритм: (DEST) ↔ (SRC)

XOR Dwreg, waop

Функция: **Логическое исключающее «ИЛИ» слов**

Код: (100001aa) (waop) (Dwreg)

Описание: Логическое сложение по модулю два слова источника SRC и слова приемника DEST и сохранение результата в приемнике.

Алгоритм: (DEST) ← (DEST) XOR (SRC)

XORB Dbreg, baop

Функция: **Логическое исключающее «ИЛИ» байт**

Код: (100101aa) (baop) (Dbreg)

Описание: Логическое сложение по модулю два байта источника SRC и байта приемника DEST и сохранение результата в приемнике.

Алгоритм: (DEST) ← (DEST) XOR (SRC)

Особенности системы команд

1 Опкоды 10h, 1Ch, 1Dh, 1Eh, 1Fh, E4h, E5h, E6h, E8h, E9h, EAh, Ebh, F1h являются зарезервированными и при обнаружении вызывают генерирование прерывания невыполнимого кода.

2 Опкод EEh является зарезервированным, но при обнаружении не вызывает генерирование прерывания невыполнимого кода.

3 Опкод FEh используется в качестве старшего байта двухбайтного опкода команд знакового деления DIV, DIVB и знакового умножения MUL, MULB.

4 Каждый сдвиг на один бит при выполнении команды логического сдвига влево (SHL и SHLL) идентичен беззнаковому арифметическому умножению на два. Каждый сдвиг на один бит при выполнении команды логического сдвига вправо (SHR, SHRL) идентичен беззнаковому арифметическому делению на два, а арифметического сдвига вправо (SHRA, SHRAL) – знаковому арифметическому делению на два.

Приложение В
(обязательное)

Коды состояний функционирования модуля I2C

В таблицах В.1 – В.11 представлена информация о соответствии кодов и операций.

Условные обозначения, принятые в таблицах:

[ADR,0], [ADR,1] – 8-разрядное значение, состоящее из 7-разрядного адреса ADR и бита направления передачи R/W#, значение которого «0» или «1» указывается непосредственно;

DAT – байт данных;

Код мастера – 8-разрядное значение 0000_1xxxb, где «xxx» – уникальный код каждого мастера в системе нескольких устройств;

«с ACK» – выражение, обозначающее, что после передачи адреса/байта в ответ на запрос подтверждения передачи (бит ACK) передатчик получает подтверждение передачи от ведомого (квитирование);

«с NACK» – выражение, обозначающее, что после передачи адреса/байта в ответ на запрос подтверждения передачи (бит ACK) передатчик получает неподтверждение передачи от ведомого (неквитирование);

X – бит может быть установленным (1b) или сброшенным (0b), в зависимости от режима работы, состояния и дальнейших действий модуля I2C.

Таблица В.1 – Исключительные состояния

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
00h	IDLE	–	–	–	–	–	Ожидать завершения текущей передачи байта
1Fh	Ошибка на шине	–	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)

Таблица В.2 – Режим FS мастера передатчика (дополнительно см. таблицу В.4)

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
1	2	3	4	5	6	7	8
01h	Старт	Код мастера	1	0	0	0	Передать код мастера и перейти в режим HS (0Ch/ 21h)
		[ADR,0]					Передать адрес ведомого (04h/ 05h)
02h	Повторный старт	[ADR,0]	1	0	0	0	Передать адрес ведомого (04h/ 05h)
		[ADR,1]					Передать адрес ведомого, после чего перейти в режим приемника (08h/ 09h)

Окончание таблицы В.2

1	2	3	4	5	6	7	8
03h	Потеря арбитража, мастер перешел в режим безадресного ведомого	–	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)
04h	Отправлен адрес ведомого с ACK	DAT	1	0	0	0	Передать байт данных (06h/ 07h)
		–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
05h	Отправлен адрес ведомого с NACK	–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
06h	Отправлен байт данных с ACK	DAT	1	0	0	0	Передать байт данных (06h/ 07h)
		–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
07h	Отправлен байт данных с NACK	–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)

Таблица В.3 – Режим FS мастера приемника

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
1	2	3	4	5	6	7	8
08h	Отправлен адрес ведомого с ACK	–	1	0	0	0	Получить байт данных, квитировать прием (0Ah)
			1	1	0	0	Получить байт данных, не квитировать прием (0Bh)
09h	Отправлен адрес ведомого с NACK	–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
0Ah	Принят байт данных и квити-рован	DAT	1	0	0	0	Получить байт данных, квитировать прием (0Ah)
			1	1	0	0	Получить байт данных, не квитировать прием (0Bh)

Окончание таблицы В.3

1	2	3	4	5	6	7	8
0Bh	Принят байт данных и не квити-рован	DAT	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)

Таблица В.4 – Режим FS мастера передатчика (дополнительно см. таблицу В2)

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
0Ch	Отправлен код мастера, обнаружена ошибка (ACK)	–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)

Таблица В.5 – Режим FS ведомого приемника (дополнительно см. таблицу В7)

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
10h	Принят адрес и квити-рован	–	1	0	0	0	Получить байт данных, квитировать прием (12h)
			1	1	0	0	Получить байт данных, не квитировать прием (13h)
11h	Принят адрес после потери арбитража и квити-рован	–	1	0	0	0	Получить байт данных, квитировать прием (12h)
			1	1	0	0	Получить байт данных, не квитировать прием (13h)
12h	Принят байт данных и квити-рован	DAT	1	0	0	0	Получить байт данных, квитировать прием (12h)
			1	1	0	0	Получить байт данных, не квитировать прием (13h)
13h	Принят байт данных и не квити-рован	DAT	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	0	0	1	Функционировать в режиме безадрес-ного ведомого, сделать старт после освобождения шины (00h, 01h)

Таблица В.6 – Режим FS ведомого передатчика

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
1	2	3	4	5	6	7	8
14h	Принят адрес и квити-рован	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (16h/17h)
15h	Принят адрес после потери арбитража и квити-рован	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (16h/17h)
16h	Отправлен байт данных с ACK	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (16h/17h)
17h	Отправлен байт данных с NACK	–	1	X	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	X	0	1	Функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)
18h	Принят адрес отклика и квити-рован	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (1Ah/1Bh)
19h	Принят адрес отклика после потери арбитража и квити-рован	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (1Ah/1Bh)
1Ah	Отправлен байт данных в ответ на получение адреса отклика с ACK	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (1Ah/1Bh)

Окончание таблицы В.6

1	2	3	4	5	6	7	8
1Bh	Отправлен байт данных в ответ на получение адреса отклика с NACK	–	1	X	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	X	0	1	Функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)

Таблица В.7 – Режим FS ведомого приемника (дополнительно см. таблицу В5)

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
1Ch	Стоп	–	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	0	0	1	Функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)
1Dh	Принят адрес общего вызова и квити-рован	–	1	0	0	0	Получить байт данных, квитировать прием (12h)
			1	1	0	0	Получить байт данных, не квитировать прием (13h)
1Eh	Принят адрес общего вызова после потери арбитража и квити-рован	–	1	0	0	0	Получить байт данных, квитировать прием (12h)
			1	1	0	0	Получить байт данных, не квитировать прием (13h)

Таблица В.8 – Режим HS мастера передатчика

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
21h	Успешно отправлен код мастера, мастер перешел в режим HS	–	1	0	0	1	Сделать повторный старт (22h)
22h	Повторный старт	[ADR,0]	1	0	0	0	Передать адрес ведомого (28h/29h)
		[ADR,1]					Передать адрес ведомого, после квитирования/не квитирования переключиться в режим мастера приемника (28h/29h)
23h	Потеря арбитража, мастер перешел в режим HS безадресного ведомого	–	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)
24h	Отправлен адрес ведомого с ACK	DAT	1	0	0	0	Передать байт данных (26h/27h)
		–	1	0	0	1	Сделать повторный старт (22h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
25h	Отправлен адрес ведомого с NACK	–	1	0	0	1	Сделать повторный старт (22h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
26h	Отправлен байт данных с ACK	DAT	1	0	0	0	Передать байт данных (26h/ 27h)
		–	1	0	0	1	Сделать повторный старт (22h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
27h	Отправлен байт данных с NACK	–	1	0	0	1	Сделать повторный старт (22h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)

Таблица В.9 – Режим HS мастера приемника

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
28h	Отправлен адрес ведомого с ACK	–	1	0	0	0	Получить байт данных, квитировать прием (2Ah)
			1	1	0	0	Получить байт данных, не квитировать прием (2Bh)
29h	Отправлен адрес ведомого с NACK	–	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)
2Ah	Принят байт данных и квити-рован	DAT	1	0	0	0	Получить байт данных, квитировать прием (2Ah)
			1	1	0	0	Получить байт данных, не квитировать прием (2Bh)
2Bh	Принят байт данных и не квити-рован	DAT	1	0	0	1	Сделать повторный старт (02h)
			1	0	1	0	Остановить передачу (00h)
			1	0	1	1	Остановить передачу, а затем сделать повторный старт (01h)

Таблица В.10 – Режим HS ведомого приемника

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
30h	Принят адрес и квити-рован	–	1	0	0	0	Получить байт данных, квитировать прием (32h)
			1	1	0	0	Получить байт данных, не квитировать прием (33h)
32h	Принят байт данных и квити-рован	DAT	1	0	0	0	Получить байт данных, квитировать прием (32h)
			1	1	0	0	Получить байт данных, не квитировать прием (33h)
33h	Принят байт данных и не квити-рован	DAT	1	0	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	0	0	1	Функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)

Таблица В.11 – Режим HS ведомого передатчика

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
34h	Принят адрес и квити-рован	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (36h/37h)
36h	Отправлен байт данных с ACK	DAT	1	X	0	0	Передать байт данных, квитировать/не квитировать (36h/37h)
37h	Отправлен байт данных с NACK	–	1	X	0	0	Функционировать в режиме безадресного ведомого (00h)
			1	X	0	1	Функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)

Приложение Г
(справочное)

Описание функций выводов микроконтроллера

В таблице Г.1 приводится перечень функций выводов микроконтроллера с подробным описанием их выполнения.

Пояснения к таблице Г.1:

- в графе «Включение функции» приведены сигналы, их состояния, а также состояния бит управления (регистр, которому принадлежит бит, указан в скобках), которые включают соответствующую функцию вывода микроконтроллера;
- в графе «Тип» указана конфигурация вывода, выполняющего включенную функцию:
- I – вход;
- O – выход;
- I/O – двунаправленный;
- QBD – квазидвунаправленный.

Все входы микроконтроллера, за исключением RESET#, являются фиксирующими входами.

Вход RESET# является уровнечувствительным входом. Во время режима POWERDOWN вход EXTINT используется как уровнечувствительный вход.

Таблица Г.1

Функция вывода	Включение функции	Тип	Описание
1	2	3	4
ACHx (x = 0 – 7)	Бит CHREx = 1 (ADCRST) и бит SEEN = 1 (ADCR)	I	ACH0 –ACH7 являются аналоговыми входами АЦП в режиме однополярных входов. Выводы ∩0V2 и ∩VCC2 должны быть подключены для функционирования АЦП
ACHNx (x = 0 – 3)	Бит CHREx = 0 (ADCRST) или бит SEEN = 0 (ADCR)	I	ACHN0–ACHN3 являются аналоговыми инверсными входами АЦП0 – АЦП3 в режиме дифференциального включения входов
ACHPx (x = 0 – 3)	Бит CHREx = 0 (ADCRST) или бит SEEN = 0 (ADCR)	I	ACHP0–ACHP3 являются аналоговыми прямыми входами АЦП0 – АЦП3 в режиме дифференциального включения входов
AD0–AD7 AD8–AD15	Шина доступа к внешним адресам	I/O	Системная шина адрес/данные. Выводы обеспечивают мультиплексированную шину адрес/данные. В течение адресной фазы шинного цикла адресные биты 0–15 выводятся на шину и могут быть зафиксированы с помощью ALE или ADV#. 8/16-битные данные передаются в течение фазы передачи данных

Продолжение таблицы Г.1

1	2	3	4
ADV#	Бит ALE = 0 (CCR)	O	Значение адреса. Выходной сигнал становится активным только во время доступа к внешней памяти. ADV# показывает, что значение данного адреса действительно находится на системной шине адрес/данные. Этот сигнал сохраняет низкий уровень до тех пор, пока шинный цикл продолжается и возвращается в высокий уровень, как только шинный цикл завершен. Внешняя защелка может использовать этот сигнал для выделения адреса с шины адрес/данные. Он также может использоваться с шифратором для выработки сигнала выборки кристалла внешней памяти
AINC#	EA# = 1 (режим програм-мирования)	I	Автоинкремент на единицу. В режиме управляемого программирования (SLAVE) этот входной сигнал с активным низким уровнем разрешает автоинкрементирование на единицу, что позволяет читать или записывать последовательные ячейки EEPROM без передачи адреса для каждого цикла чтения или записи
ALE	Бит ALE = 1 (CCR)	O	Разрешение функции адреса. Этот выходной сигнал становится активным только во время доступа к внешней памяти. ALE показывает, что значение адресной информации действительно находится на системной шине адрес/данные и сигнализирует о начале цикла шины. ALE отличается от ADV# тем, что не остается активным в течение всего шинного цикла. Внешняя защелка может использовать этот сигнал для выделения адреса с шины адрес/данные
BHE#	Бит WR = 1 (CCR)	O	Разрешение старшего байта. Это выходной сигнал с активным низким уровнем, устанавливающимся только при записи во внешнюю память слова или старшего байта. BHE# показывает, что данные передаются (записываются) через старшую половину системной шины адрес/данные. BHE# используется вместе с A0: BHE# A0 Записываемый байт 0 0 оба байта 0 1 только старший байт 1 0 только младший байт
BW	Бит BW0 = 1 (CCR)	I	Ширина шины. Если бит BW0 установлен, этот сигнал выбирает ширину шины во время внешнего доступа. Когда сигнал на входе BW высокий, выбирается 16-разрядная шина; когда низкий – 8-разрядная. Если BW0 сброшен, ширина шины всегда 8 бит, при этом состояние сигнала BW игнорируется

Продолжение таблицы Г.1

1	2	3	4
CAN_RX0	–	I	Вход 0 порта CAN
CAN_RX1	–	I	Вход 1 порта CAN
CAN_TX0	–	O	Выход 0 порта CAN
CAN_TX1	–	O	Выход 1 порта CAN
CLKOUT	Зарезервированный первый бит регистра IOC3. Всегда ноль	O	Тактовый выход (всегда разрешен). Выход внутреннего тактового генератора. Частота CLKOUT равна 1/2 частоты на входе XTAL1
CPVER*	EA# = 1 (режим программирования)	O	Общая верификация программы. Этот выходной сигнал позволяет обнаружить любую ошибку сравнения, которая имела место в режиме программирования. CPVER остается высоким до тех пор, пока не появится ошибка сравнения, в это время он становится низким. Если существует хотя бы одна ошибка, CPVER остается в низком уровне, пока устройство не выйдет из режима программирования. Когда сигнал CPVER имеет высокий уровень, это показывает, что все ячейки запрограммированы верно
EA#	–	I	Доступ к внешней памяти. Этот сигнал с активным низким уровнем разрешает доступ к памяти вне кристалла. Если уровень EA# высокий, то выбирается внутренняя EEPROM. Сигнал EA# фиксируется только по нарастающему фронту сигнала RESET#
EXTINT	Бит EXTINT_SRC = 0 (IOC1)	I	Внешнее прерывание. Вход EXTINT – динамический вход, однако, используется как чувствительный к уровню вход в режиме POWERDOWN. P2.2 всегда вырабатывает EXTINT1 прерывание (INT13). Когда EXTINT_SRC = 0, нарастающий фронт на выводе P2.2 также вырабатывает прерывание EXTINT (INT07). EXTINT и EXTINT1 должны удерживаться более чем два машинных такта, чтобы гарантировать, что они зафиксированы
EXTINT	Бит EXTINT_SRC = 1 (IOC1)	I	Внешнее прерывание. Нарастающий фронт на входе P0.7 вырабатывает внешнее прерывание EXTINT(INT07). EXTINT должен удерживаться более чем два машинных такта, чтобы гарантировать его захват
FSADJ	–	–	Вывод подстройки выходного тока ЦАП
HSI.0	Бит HSI0_ENA = 1 (IOC0)	I	Вход модуля высокоскоростного ввода. Вывод может также использоваться для выработки прерывания по нарастающему фронту (INT04). HSI.0 должен удерживаться более чем два машинных такта, чтобы гарантировать его захват

Продолжение таблицы Г.1

1	2	3	4
HSI.1	Бит HSI1_ENA = 1 (IOC0)	I	Вход модуля высокоскоростного ввода. Когда сброшены биты T2CLK_SRC и T2_ENA регистров IOC0 и IOC3, соответственно, вывод HSI.1 функционирует как источник синхросигнала таймера T2
HSI.2 HSI.3	Бит HSI2_ENA = 1 Бит HSI3_ENA = 1 (IOC0)	I	Вход модуля высокоскоростного ввода. Функции HSI и HSO каждого вывода могут быть одновременно активными, в этом случае вывод действует как выход, управляемый HSO
HSO.0 – HSO.3	– –	O	Выходы модуля высокоскоростного вывода
HSO.4	Бит HSO4ENA = 1 (IOC1)	O	Выход модуля высокоскоростного вывода Функции HSI и HSO могут быть одновременно активными, в этом случае вывод действует как выход, управляемый HSO
HSO.5	Бит HSO5ENA = 1 (IOC1)	O	Выход для модуля высокоскоростного вывода. Функции HSI и HSO могут быть одновременно активными, в этом случае вывод действует как выход, управляемый HSI
INST	–	O	Выборка команды. Сигнал действителен только в цикле чтения внешней памяти. Если уровень сигнала высокий, INST показывает, что выбирается команда, если уровень низкий – считываются данные. INST может использоваться в устройствах, которые требуют разделения байтов памяти для данных и команд
IOUTA	–	O	Токовый выход ЦАП
IOUTB	–	O	Комплементарный выход ЦАП
LIN_RX	–	I	Вход порта LIN
LIN_TX	–	O	Выход порта LIN
MISO	Установлен шестой бит регистра IOPORT1	I/O	Вход данных мастера, выход данных ведомого. Функционирует при разрешенной работе интерфейса SPI
MOSI	Установлен пятый бит регистра IOPORT1	I/O	Выход данных мастера, вход данных ведомого Функционирует при разрешенной работе интерфейса SPI
NMI	–	I	Немаскируемое прерывание. Положительный перепад вызывает немаскируемое прерывание через вектор, размещенный в ячейке 203Eh. NMI должен удерживаться более чем один машинный такт, чтобы гарантировать его захват. Когда NMI не используется, он должен быть подключен к нулю

Продолжение таблицы Г.1

1	2	3	4
P0.0 – P0.7	–	I	Порт 0 (работа только на ввод, регистр порта доступен только для чтения). 8-битный порт с высоким сопротивлением. P0.0 – P0.7 являются цифровыми входами. Возможно функционирование этих выводов одновременно как аналоговых, так и цифровых входов, но это не рекомендуется из-за того, что чтение порта 0 во время АЦ преобразования может привести к непредсказуемому результату преобразования
P1.0 – P1.2	–	QBD	Порт 1. 8-битный квазидвунаправленный порт ввода-вывода. При функционировании выводов P1.5 – P1.7 в качестве SPI-порта они являются стандартными выводами (неквазидвунаправленными)
P1.3 P1.4	Бит PWM1_SEL = 0 Бит PWM2_SEL = 0 (IOC3)	QBD	
P1.5 – P1.7	–	QBD	
P2.0	Бит TXD_SEL = 0 (IOC1)	O	Порт 2. Многофункциональный 8-битный порт
P2.1	–	I	
P2.2 – P2.4	–	I	
P2.5 P2.6	Бит PWM_SEL = 0 Бит TXD_SEL = 0 (IOC1)	O QDB	
P2.7	–	QDB	
P3.0 – P3.7 P4.0 – P4.7	EA# = 1	I/O	Порты 3 и 4. 8-битные двунаправленные порты ввода/вывода с выходами с открытым стоком. Эти выводы образуют мультиплексированную шину адрес/данные и имеют внутреннюю низкоомную нагрузку на UCC1. Порты 3 и 4 могут быть прочитаны и записаны только как слово. Во время режима программирования эти порты действуют как шина PBUS
PACT#*	EA# = 1 (режим програм-мирования)	O	Активное программирование. В режиме автоматического программирования низкий уровень сигнала PACT# показывает, что осуществляется программирование
PALE#*	EA# = 1 (режим програм-мирования)	I	Программирующий ALE. Когда PALE# установлен, команды и данные с портов 3 и 4 читаются в устройство

Продолжение таблицы Г.1

1	2	3	4
PBUS0– PBUS15	Шина доступа к внешним адресам	I/O	Шина для чтения и записи команд, адресов и данных. Порты 3 и 4 используются в режиме ввода-вывода с открытым стоком (не как системная шина)
PMODE0* PMODE1* PMODE2* PMODE3*	EA# = 1 (режим програм- мирования)	I	Выбор режима программирования. Определяют алгоритм программирования EEPROM, который будет выполняться. PMODE фиксируется после сброса устройства, когда EA# =1; PMODE должны быть стабильными, пока устройство работает
PROG#*	EA# = 1 (режим програм- мирования)	I	Начало программирования. Вход с активным низким уровнем имеет значение только во время подчиненного программирования (SLAVE). Когда установлен PROG#, это вызывает программирование данных с шины програм-мирования в EEPROM. Когда PROG# неактивен, программирующий импульс заканчивается
PVER*	EA# = 1 (режим програм- мирования)	O	Сравнение программы. В режиме программирования этот выходной сигнал с активным высоким уровнем показывает, что слово запрограммировано верно
PWM0	Бит PWM_SEL = 1 (IOC1)	O	Выход 0 широтно-импульсного модулятора Если PWM0 специально удерживается в высоком уровне в момент нарастающего фронта RESET#, то устройство входит в режим тестирования
PWM1 PWM2	Бит PWM1_SEL = 1 Бит PWM2_SEL = 1 (IOC3)	O O	Выходы 1 и 2 широтно-импульсного модулятора
RD#	–	O	Внешнее чтение. Выходной сигнал с активным низким уровнем, показывающий, что происходит чтение внешней памяти
READY	–	I	Вход готовности. Сигнал используется для удлинения цикла внешней памяти, выработавшей «состояния ожидания» для согласования с медленной памятью. Когда READY высокий, ЦПУ продолжает работать в нормальном режиме. Если READY принимает низкий уровень перед спадающим фронтом сигнала CLKOUT, контроллер памяти вводит циклы ожидания, пока в момент CLKOUT не будет высокого уровня на READY, или до тех пор, пока количество циклов ожиданий не будет равно количеству, запрограммированному битами IRC1–IRC0 регистра CCR, сигнал READY игнорируется для всей внутренней памяти. READY является активным во время выборки CCR

Продолжение таблицы Г.1

1	2	3	4
REFOUT	Бит RU = 1 (ADCR)	–	Буферизированный выход опорного напряжения АЦП. Возможен одновременный вывод опорного напряжения и включения преобразования восьмого АЦП в режиме прямого хода для калибровки
REFCAP	–	–	Вывод подключения внешнего источника опорного напряжения АЦП/Вывод подключения фильтрующего конденсатора источника опорного напряжения АЦП
REFIO	–	–	Вывод подключения внешнего источника опорного напряжения ЦАП/Вывод подключения фильтрующего конденсатора источника опорного напряжения ЦАП
RESET#	–	I/O	Вход сброса и выход с открытым стоком. Спадающий фронт сигнала RESET# инициирует процесс сброса. Когда RESET# устанавливается впервые, кристалл открывает транзистор с нагрузкой на #0V1, соединенный с выводом RESET, на 16 машинных тактов. Эта функция может также быть активизирована переполнением сторожевого таймера или выполнением команды RST. В режиме POWERDOWN процесс сброса вызывает переход контроллера в нормальный режим работы
RXD0	Бит REN = 1 (SP_CON0)	I/O	Последовательный вход данных порта USART0
RXD1	Бит REN = 1 (SP_CON1)	I/O	Последовательный вход данных порта USART1
SCK	Установлен седьмой бит регистра IOPORT1	I/O	Вход/выход тактовой частоты порта SPI
SCL	Бит ENABLE = 1 (SMBCTRL2)	I/O	Вход/выход тактовой частоты порта I2C. При активизации работы интерфейса I2C функционирует как вход или выход с открытым стоком
SDA	Бит ENABLE = 1 (SMBCTRL2)	I/O	Вход/выход данных порта I2C. При активизации работы интерфейса I2C функционирует как вход или выход с открытым стоком
SS#	Установлен второй бит регистра IOPORT1	I	Вход разрешения выбора ведомого порта SPI
T2CAP	–	I	Нарастающий фронт на P2.7 фиксирует значение таймера 2 в регистр T2CAPTURE и вырабатывает прерывание захвата таймера 2 (INT11). T2CAP должен удерживаться более чем два машинных такта, чтобы гарантировать захват

Продолжение таблицы Г.1

1	2	3	4
T2CLK	Бит T2CLK_SRC = 0 (IOC0) и бит T2_ENA = 0 (IOC3) Бит SOURCE = 0 (BOUD_RATEx)	I	Вход тактирования таймера 2 и вход генератора, задающего скорость передачи для последовательного порта
T2RST	Бит T2RST_ENA = 1 и бит T2RST_SRC = 0 (IOC0)	I	Сброс таймера 2. Нарастающий фронт на T2RST сбрасывает таймер 2
T2UPDN	Бит T2UD_ENA = 1 (IOC2)	I	Управление направлением счета таймера 2. Когда T2UPDN имеет высокий уровень, таймер 2 считает на уменьшение, когда T2UPDN имеет низкий уровень – на увеличение
TCK	–	I	Вход тактовой частоты JTAG
TDI	–	I	Вход данных JTAG
TDO	–	O	Выход данных JTAG
TMS	–	I	Вход выбора режима JTAG
TRST	–	I	Вход сброса JTAG
TXDx (x = 0,1)	Бит TXD_SEL = 1 (IOC1)	O	Выход последовательных данных порта USARTx
VPR	–	I	Вход в режим программирования/ Вход «старт с альтернативного адреса», а также вход «возврат из режима POWERDOWN»
WR#	Бит WR = 1 (CCR)	O	Внешняя запись. Выходной сигнал с активным низким уровнем, устанавливающийся при совершении записи во внешнюю память
WRH#	Бит WR = 0 (CCR)	O	Запись старшего байта. В 16-разрядном режиме шины WRH# устанавливается при записи старших байта или слова, в 8-разрядном режиме шины (BW0 = 0) – при записи старшего и младшего байт, а также слова
WRL#	Бит WR = 0 (CCR)	O	Запись младшего байта. В 16-разрядном режиме шины WRL# устанавливается при записи младшего байта или слова, в 8-разрядном режиме – при записи старшего и младшего байт, а также слова
#VCC1	–	–	Напряжение питания цифровой части устройства
∩VCC2	–	–	Опорное напряжение АЦП. Вывод ∩VCC2 является напряжением питания аналоговой части АЦП и логики, используемой для чтения порта 0. Для нормального функционирования АЦП и порта вывод ∩VCC2 должен быть подключен

Окончание таблицы Г.1

1	2	3	4
∩VCC3	–	–	Опорное напряжение ЦАП Вывод ∩VCC3 является напряжением питания для аналоговой части ЦАП. Для нормального функционирования ЦАП вывод ∩VCC3 должен быть подключен
#0V1	–	–	Общий вывод цифровой части микросхемы. Имеется несколько выводов #0V1, все они должны быть соединены
∩0V2	–	–	Аналоговая земля АЦП
∩0V3	–	–	Аналоговая земля ЦАП
XTAL1	–	I –	Вход тактового сигнала Вывод подключения кварцевого резонатора
XTAL2	–	–	Вывод подключения кварцевого резонатора

_____

* Функция реализуется только при использовании стандартной программы-монитора ОАО «НИИЭТ», записанной в ПЗУ.

В таблице Г.2 описаны состояния выводов во время и после сброса.

Таблица Г.2 – Состояние выводов во время и после сброса

Название вывода	Состояние вывода во время сброса	Состояние вывода после сброса
P0.0 – P0.7	Входы[1)]	Входы[1)]
P1.0 – P1.7	Слабый Pull-Up	Слабый Pull-Up
P2.0	Сильный Pull-Up	Выход
P2.1	Вход[2)]	Вход[2)]
P2.2	Вход[2)]	Вход[2)]
P2.3	Вход [2)]	Вход[2)]
P2.4	Вход[2)]	Вход[2)]
P2.5	Сильный Pull-Down	Выход
P2.6 – P2.7	Слабый Pull-Up	Слабый Pull-Up
P3.0 – P4.7	Слабый Pull-Up	Шина адрес/данные или порт ввода-вывода[3)]
HSI.0, HSI.1	Вход[2)]	Вход[2)]
HSI.2/HSO.4	Вход[2)]	Вход[2)]
HSI.3/HSO.5	Вход[2)]	Вход[2)]
HSO.0 – HSO.3	Слабый Pull-Down	Выход
ALE	Слабый Pull-Up	Выход
BHE#	Слабый Pull-Up	Выход
BW	Вход[2)]	Вход[2)]
CLKOUT	CLKOUT Выход	CLKOUT Выход
EA#	Вход[2)]	Вход[2)]
INST	Слабый Pull-Up	Выход
NMI	Слабый Pull-Down	Слабый Pull-Down
RD#	Слабый Pull-Up	Выход
READY	Вход[2)]	Вход[2)]
RESET#	Сильный Pull-Up	Сильный Pull-Up
WR#	Слабый Pull-Up	Выход

Окончание таблицы Г.2

[1] Эти выводы могут быть неподключенными. Однако, рекомендуется подключить их к высокому или низкому логическому уровню.

[2] Эти выводы должны быть подключены и не оставаться плавающими. Входное напряжение не должно превышать уровень напряжения питания микросхемы в режиме нормальной работы.

[3] Состояние этих выводов зависит от конфигурации устройства. Если шина адрес/данные активна, выводы действуют как шинные формирователи, иначе они действуют, как порт ввода/вывода с открытым стоком и являются неподключенными.

В таблице Г.3 приведены типовые максимальные значения токов для различных схем подтяжки до высокого (Pull-Up) и низкого (Pull-Down) уровней напряжения, подключаемых к выводам микроконтроллера во время и после сброса.

Таблица Г.3 – Описание состояния вывода

Состояние вывода	Значение вывода
Слабый Pull-Up	100 мкА
Сильный Pull-Up	1 мА
Слабый Pull-Down	200 мкА
Сильный Pull-Down	500 мкА
Высокий уровень	U_{OH}
Низкий уровень	U_{OL}

Приложение Д
(справочное)

Программно-аппаратный комплекс

Программно-аппаратный комплекс (ПАК) – это набор технических и программных средств, работающих совместно для выполнения одной или нескольких сходных задач.

ПАК инструментальных средств разработки и отладки программного обеспечения СБИС 1874ВЕ8Т предназначен для создания и отладки программ, создаваемых для выполнения средствами ИС, и состоит из двух частей:

- программной, включающей интегрированную среду проектирования CodeMaster-96, компиляторы, программную модель ядра, программное обеспечение аппаратного отладочного устройства, программу-монитор микроконвертера;
- аппаратной, включающей аппаратное отладочное устройство JEM-96, макетно-отладочную плату, аппаратные средства отладки целевого микроконтроллера.

Подробное описние ПАК и работы с ним приведено в документе «Микросхема интегральная 1874ВЕ8Т. Программно-аппаратный комплекс. Руководство» КФДЛ.424939.012 на МН.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	изме-ненных	заме-нен-ных	новых	анну-лиро-ванных				
-	-	-	все	-	383			29.12.14
1	1	-	-	-	-			29.12.14