МИКРОСХЕМА ИНТЕГРАЛЬНАЯ

1882ВМ1Т

Руководство пользователя

2013

Содержание

Введение

Интегральная микросхема 1882ВМ1Т представляет собой высокопроизводительный мультиинтерфейсный 8-разрядный периферийный сопроцессор. Такой сопроцессор включает в себя программируемый микроконтроллер с блоками энергонезависимой памяти и большое количество разнообразных интерфейсов (MIL-STD-1553B, LIN, I2C, UART, SPI). В сопроцессоре обеспечена еще и такая важная функция как встроенная аппаратная реализация алгоритмов защиты данных, основанных на алгоритме, описанном в ГОСТ 28147–89. ИС может применяться как для сопряжения между интерфейсами различных типов в сетях обмена информацией, так и для управления внешними периферийными устройствами (АЦП, ЦАП, карточки памяти и т.д.) по защищенным каналам связи.

КМОП микроконтроллер 1882ВМ1Т оснащен перепрограммируемой памятью программ типа Flash, которая может быть записана непосредственно в системе через механизм внутрисистемного программирования ISP (посредством модуля интерфейса SPI). Микроконтроллер совместим по системе команд и по функциональному назначению выводов с аналогичными устройствами семейства 80С51.

Микроконтроллер выпускается в 64-выводном корпусе типа 4203.64-1.

Помимо памяти программ типа Flash объемом 32 Кбайт, микроконтроллер имеет 4 Кбайт памяти данных типа EEPROM. Содержимое памяти типа Flash может быть защищено от несанкционированной записи/считывания. Кроме того, в состав микроконтроллера входят два указателя данных, блок 16-разрядной арифметики MDU, 48 программируемых линий ввода-вывода, три 16-разрядных таймера/счетчика событий, программируемый сторожевой таймер, контроллер прерываний с четырьмя уровнями приоритета, встроенный тактовый генератор. Микроконтроллер поддерживает протоколы последовательных интерфейсов SPI, I2C, LIN, ГОСТ Р 52070–2003 (аналог интерфейса MIL-STD-1553B), а также имеет универсальные асинхронные приемо-передатчики типа UART. Также имеется встроенный блок PCA. Частота работы микросхемы – до 24 МГц.

1 Назначение и основные технические характеристики микросхем

Микросхемы предназначены для применения во встроенных системах управления комплексами радиосвязи, в системах автоматизации технологических процессов, в системах автоматизированного управления электроприводом, перспективных системах радиосвязи шестого поколения, бортовых встроенных системах управления различных видов ВВСТ и телекоммуникационных средствах специального назначения и т. п.

1.1 Функциональные параметры

Микроконтроллер имеет следующие параметры:

- разрядность АЛУ, бит 8;
- память программ Flash, байт 32К;
- память данных EEPROM, байт 4К;
- ОЗУ данных, байт 256;
- адресуемая память, байт 64К;
- блок 16-разярдной арифметики MDU 1;
- таймеры 16-разрядные 3;
- массив таймеров-счетчиков (PCA) 1;
- блок кодирования по ГОСТ 28147–89 1;
- сторожевой таймер 1;
- 8-разрядные порты ввода-вывода 6;
- последовательный порт UART 2;
- последовательный интерфейс SPI 2;
- последовательный интерфейс I2C 1;
- последовательный интерфейс LIN 1;
- последовательный интерфейс ГОСТ Р 52070–2003 1.

1.2 Конструктивные характеристики

Кристалл микросхемы выполнен по КМОП технологии.

Микросхемы 1882ВМ1Т разработаны в корпусах 4203.64-1.

Условное графическое обозначение ИС приведено на рисунке 1.1. Функциональное назначение выводов приведено в таблице 1.1.

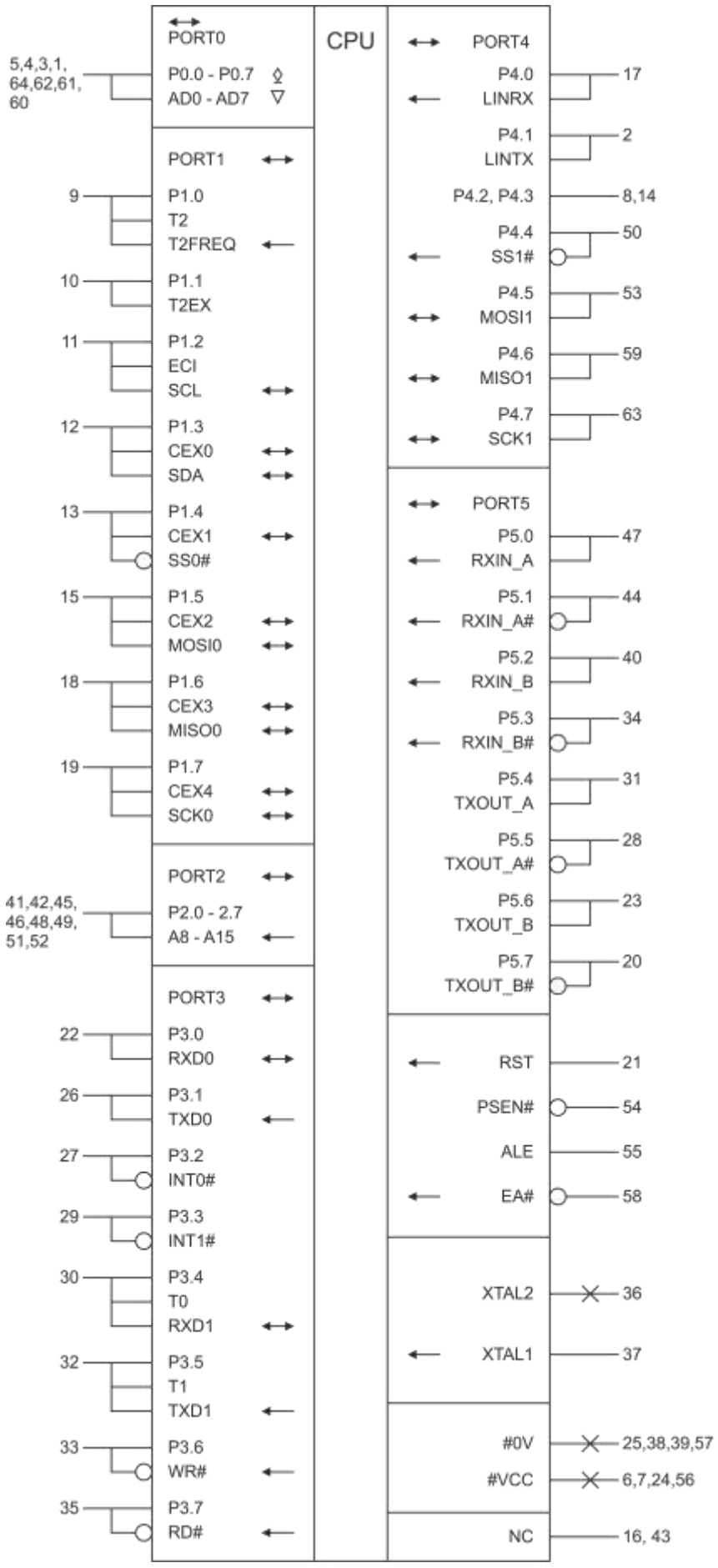


Рисунок 1.1 – Условное графическое обозначение ИС в корпусе 4203.64-1

Таблица 1.1 – Функциональное назначение выводов микросхемы

Обозначение		Номер вывода	Функциональное назначение вывода	Тип вывода
вывода	альтерна-тивной функции вывода			
1	2	3	4	5
P0.0 – P0.7	AD0 – AD7	5, 4, 3, 1, 64, 62, 61, 60	8-разрядный двунаправленный порт P0, 0 – 7 разряды Шина адреса/данных при работе с внешней памятью	I/O/2 I/O/Z
P1.0	T2 T2FREQ	9	Вход/выход порта 1, 0 разряд Вход таймера/счетчика 2 Выход частоты таймера/счетчика 2	I/O I O
P1.1	T2EX	10	Вход/выход порта 1, 1 разряд Вход триггера выборки-перезагрузки таймера/счетчика 2	I/O I
P1.2	ECI SCL	11	Вход/выход порта 1, 2 разряд Вход таймера массива программируемых счетчиков (PCA) Вход/выход тактовой частоты интерфейса I2C	I/O I I/O
P1.3	CEX0 SDA	12	Вход/выход порта 1, 3 разряд Вход/выход 0 канала PCA Вход/выход данных интерфейса I2C	I/O I/O I/O
P1.4	CEX1 SS0#	13	Вход/выход порта 1, 4 разряд Вход/выход 1 канала PCA Вход выбора ведомого SS# интерфейса SPI0	I/O I/O I
P1.5	CEX2 MOSI0	15	Вход/выход порта 1, 5 разряд Вход/выход 2 канала PCA Вход/выход MOSI интерфейса SPI0	I/O I/O I/O
P1.6	CEX3 MISO0	18	Вход/выход порта 1, 6 разряд Вход/выход 3 канала PCA Вход/выход MISO интерфейса SPI0	I/O I/O I/O
P1.7	CEX4 SCK0	19	Вход/выход порта 1, 7 разряд Вход/выход PCA, 4 канал Вход/выход SCK интерфейса SPI0	I/O I/O I/O
P2.0 – P2.7	A8 – A15	41, 42, 45 ,46, 48, 49, 51, 52	8-разрядный двунаправленный порт P2, 0 – 7 разряды Выход адреса в режиме работы с внешней памятью	I/O O
P3.0	RXD0	22	Вход/выход порта 3, 0 разряд Вход/выход последовательных данных UART0	I/O I/O
P3.1	TXD0	26	Вход/выход порта 3, 1 разряд Выход последовательных данных UART0	I/O O
P3.2	INT0#	27	Вход/выход порта 3, 2 разряд Вход внешнего прерывания 0	I/O I
P3.3	INT1#	29	Вход/выход порта 3, 3 разряд Вход внешнего прерывания 1	I/O I
P3.4	T0 RXD1	30	Вход/выход порта 3, 4 разряд Вход таймера/счетчика 0 Вход/выход последовательных данных UART1	I/O I I/O

Продолжение таблицы 1.1

1	2	3	4	5
P3.5	 T1 TXD1	32	Вход/выход порта 3, 5 разряд Вход таймера/счетчика 1 Выход последовательных данных UART1	I/O I O
P3.6	 WR#	33	Вход/выход порта 3, 6 разряд Выход стробирующего сигнала при записи во внешнюю память данных	I/O O
P3.7	 RD#	35	Вход/выход порта 3, 7 разряд Выход стробирующего сигнала при чтении из внешней памяти данных	I/O O
P4.0	 LINRX	17	Вход/выход порта 4, 0 разряд Вход данных интерфейса LIN	I/O I
P4.1	 LINTX	2	Вход/выход порта 4, 1 разряд Выход данных интерфейса LIN	I/O O
P4.2	–	8	Вход/выход порта 4, 2 разряд	I/O
P4.3	–	14	Вход/выход порта 4, 3 разряд	I/O
P4.4	 SS1#	50	Вход/выход порта 4, 4 разряд Вход выбора ведомого SS# интерфейса SPI1	I/O I
P4.5	 MOSI1	53	Вход/выход порта 4, 5 разряд Вход/выход MOSI интерфейса SPI1	I/O I/O
P4.6	 MISO1	59	Вход/выход порта 4, 6 разряд Вход/выход MISO интерфейса SPI1	I/O I/O
P4.7	 SCK1	63	Вход/выход порта 4, 7 разряд Вход/выход SCK интерфейса SPI1	I/O I/O
P5.0	 RXIN_A	47	Вход/выход порта 5, 0 разряд Входной сигнал канала А интерфейса ГОСТ Р 52070–2003	I/O I
P5.1	 RXIN_A#	44	Вход/выход порта 5, 1 разряд Входной сигнал канала А интерфейса ГОСТ Р 52070–2003. Инверсный.	I/O I
P5.2	 RXIN_B	40	Вход/выход порта 5, 2 разряд Входной сигнал канала В интерфейса ГОСТ Р 52070–2003	I/O I
P5.3	 RXIN_B#	34	Вход/выход порта 5, 3 разряд Входной сигнал канала В интерфейса ГОСТ Р 52070–2003. Инверсный.	I/O I
P5.4	 TXOUT_A	31	Вход/выход порта 5, 4 разряд Выходной сигнал канала А интерфейса ГОСТ Р 52070–2003	I/O O
P5.5	 TXOUT_A#	28	Вход/выход порта 5, 5 разряд Выходной сигнал канала А интерфейса ГОСТ Р 52070–2003. Инверсный.	I/O O
P5.6	 TXOUT_B	23	Вход/выход порта 5, 6 разряд Выходной сигнал канала В интерфейса ГОСТ Р 52070–2003	I/O O
P5.7	 TXOUT_B#	20	Вход/выход порта 5, 7 разряд Выходной сигнал канала В интерфейса ГОСТ Р 52070–2003. Инверсный.	I/O O
XTAL1	–	37	Вывод для подключения кварцевого резонатора Вход тактового сигнала	– I

Окончание таблицы 1.1

1	2	3	4	5
XTAL2	–	36	Вывод для подключения кварцевого резонатора	–
RST	–	21	Входной сигнал общего сброса	I
PSEN#	–	54	Выход разрешения программной памяти	O
ALE	–	55	Выход разрешения фиксации адреса	O
EA#	–	58	Вход блокировки работы с внутренней памятью программ	I
#VCC	–	6, 7, 24, 56	Выводы питания 3,3 В	–
#0V	–	25, 38, 39, 57	Общий вывод микросхемы	–
NC	–	16, 43	Неподключенные выводы	–

Примечания

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

2 Выводы портов P1, P3, P4, P5 микроконтроллера имеют схемы «pull-up», подтягивающие напряжение на выводах до высокого логического уровня сигнала.

3 Выводы порта P2 микроконтроллера имеют схемы «pull-up», которые аппаратно отключаются при работе микроконтроллера с внешней памятью.

1.3 Электрические характеристики

Электрические характеристики микросхем приведены в таблице 1.2.

Значения предельно допустимых электрических режимов эксплуатации микросхем в диапазоне рабочих температур приведены в таблице 1.3.

Таблица 1.2 – Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, ºС
		не менее	не более	
1	2	3	4	5
1 Выходное напряжение низкого уровня по выводам P0.0 – P0.7, P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, P4.0 – P4.7, P5.0 – P5.7, PSEN#, ALE, В, U_{CC} = 3,0 В, I_{OL} = 2,8 мА	U_{OL}	–	0,4	–60 ± 3 25 ± 10 125 ± 5
2 Выходное напряжение высокого уровня по выводам P0.0 – P0.7, P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, P4.0 – P4.7, P5.0 – P5.7, PSEN#, ALE, В, U_{CC} = 3,0 В, I_{OH} = –3,2 мА	U_{OH}	U_{CC}–0,9	–	
3 Ток утечки на входах без «pull-up» по выводам P0.0 – P0.7, EA#, мкА,[1)] U_{CC} = 3,6 В, 0 В < U_I < U_{CC}	I_{ILL}	–10	–	
	I_{ILH1}	–	10	
4 Ток утечки высокого уровня на входах «pull-up» по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, P4.0 – P4.7, P5.0 – P5.7, мкА,[1)] U_{CC} = 3,6 В, U_{IH} = 3,6 В	I_{ILH2}	–	10	

Окончание таблицы 1.2

1	2	3	4	5
5 Входной ток низкого уровня на входах «pull-up» по выводам P1.0 – P1.7, P2.0 – P2.7, P3.0 – P3.7, P4.0 – P4.7, P5.0 – P5.7, мкА, U_{CC} = 3,6 В, U_{IL} = 0 В	I_{IL1}	–200	–	
6 Входной ток по входу XTAL1, мкА, U_{CC} = 3,6 В, 0 В < U_I < U_{CC}	I_{IL2}	–40	–	
	I_{IH1}	–	40	
7 Входной ток высокого уровня по выводу сброса RST, мкА, U_{CC} = 3,6 В, U_{IH} = 3,6 В	I_{IH2}	20	800	–60 ± 3 25 ± 10 125 ± 5
8 Ток утечки высокого уровня на выходах «pull-up» по выводам ALE, PSEN#, мкА,[1)] U_{CC} = 3,6 В, U_{OH} = 3,6 В	I_{OLH}	–	10	
9 Динамический ток потребления в режиме сброса по выводу #VCC, мА, U_{CC} = 3,6 В, f_{CI} = 24 МГц	I_{OCC}	–	120	
10 Функциональный контроль U_{CC} = (3,0; 3,6) В, f_{CI} = 24 МГц	ФК	–	–	

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

Таблица 1.3 – Значения предельно допустимых электрических режимов эксплуатации в диапазоне рабочих температур от минус 60 до 125 °С

Наименование параметра режима, единица измерения	Буквенное обозначение параметра	Предельно допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
1 Напряжение питания, В	U_{CC}	3,0	3,6	–	6,0
2 Входное напряжение низкого уровня, В	U_{IL}	–0,5	0,8	–0,6	–
3 Входное напряжение высокого уровня по выводам, кроме XTAL1 и RST, В	U_{IH}	$0,2U_{CC}$+1,0	U_{CC}	–	U_{CC}+0,6
4 Входное напряжение высокого уровня по выводу XTAL1, В	U_{IHCI}	$0,7U_{CC}$	U_{CC}	–	U_{CC}+0,6
5 Входное напряжение высокого уровня по выводу RST, В	U_{IHRST}	$0,6U_{CC}$	U_{CC}	–	U_{CC}+0,6
6 Выходной ток высокого уровня, мА	I_{OH}	–3,2	–	–10	–
7 Выходной ток низкого уровня, мА	I_{OL}	–	2,8	–	10
8 Частота следования импульсов тактового сигнала, МГц	f_{CI}	0,001	24	–	–
9 Длительность фронта и спада тактового сигнала, нс	t_{LH} t_{HL}	–	5	–	–
10 Емкость нагрузки, пФ	C_L	–	100	–	200
Примечание – Время работы в одном из предельных режимов должно быть не более 5 с.					

2 Описание изделия

Микросхема представляет собой мультиинтерфейсный 8-разрядный сопроцессор на базе усовершенствованной архитектуры MCS-51 со встроенной аппаратной реализацией алгоритмов кодирования/декодирования информации.

2.1 Особенности микроконтроллера

Основными особенностями являются:

- 8-разрядное микроконтроллерное ядро, оптимизированное для приложений управления. Поддержка ускоренного режима работы ядра с уменьшенным временем выполнения команд;
- 32 Кбайт встроенной перепрограммируемой памяти программ типа Flash и 4 Кбайта встроенной памяти данных типа EEPROM с поддержкой как параллельного (с использованием обычного программатора), так и последовательного побайтного и пословного внутрисистемного программирования (ISP) посредством последовательного интерфейса SPI;
- возможность задания коэффициента деления внутренней тактовой частоты;
- три уровня защиты памяти программ;
- память ОЗУ (256 × 8) бит;
- контроллер прерываний с четырьмя уровнями приоритетов;
- возможность выполнения арифметических операций над 16-разрядными операндами благодаря наличию встроенного блока MDU;
- возможность кодирования/ декодирования данных на основе алгоритмов ГОСТ 28147–89;
- возможность работы на пониженной мощности в режиме холостого хода (Idle) и в режиме пониженного энергопотребления (Power Down) с выходом из режимов по прерыванию;
- поддержка большого количества протоколов обмена: UART, SPI, LIN, ГОСТ Р 52070–2003 (аналог MIL-STD-1553B).

Микросхема обеспечивает работу с внешней тактовой частотой до 24 МГц при включенном внутреннем делителе на два и до 12 МГц при выключенном делителе.

Режим холостого хода останавливает ЦПУ, при этом сохраняется работоспособность ОЗУ, таймеров/счетчиков, последовательных портов и системы прерываний. Режим пониженной мощности обеспечивает сохранность содержимого ОЗУ, однако, приостанавливает генератор, отключая все остальные функции, имеющиеся на кристалле, до следующего внешнего прерывания или аппаратного сброса.

2.2 Организация микроконтроллера

На рисунке 2.1 приведена структурная схема микроконтроллера 1882ВМ1Т.

Основным блоком микроконтроллера является 8-разрядное ядро, поддерживающее ускоренный режим работы с уменьшенным временем выполнения команд и включающее в себя регистровое АЛУ, регистр аккумулятора, регистр расширителя аккумулятора и регистр слова состояния процессора. В состав микроконтроллера также входит блок 16-разрядной арифметики или, другими словами, – блок умножения и деления, наличие которого позволяет использовать микроконтроллер в приложениях управления в реальном времени, которые требуют быстрых математических вычислений.

Память микроконтроллера представлена встроенной перепрограммируемой памятью программ типа Flash и встроенной памятью данных типа EEPROM с возможностью последовательного и параллельного программирования, а также ОЗУ.

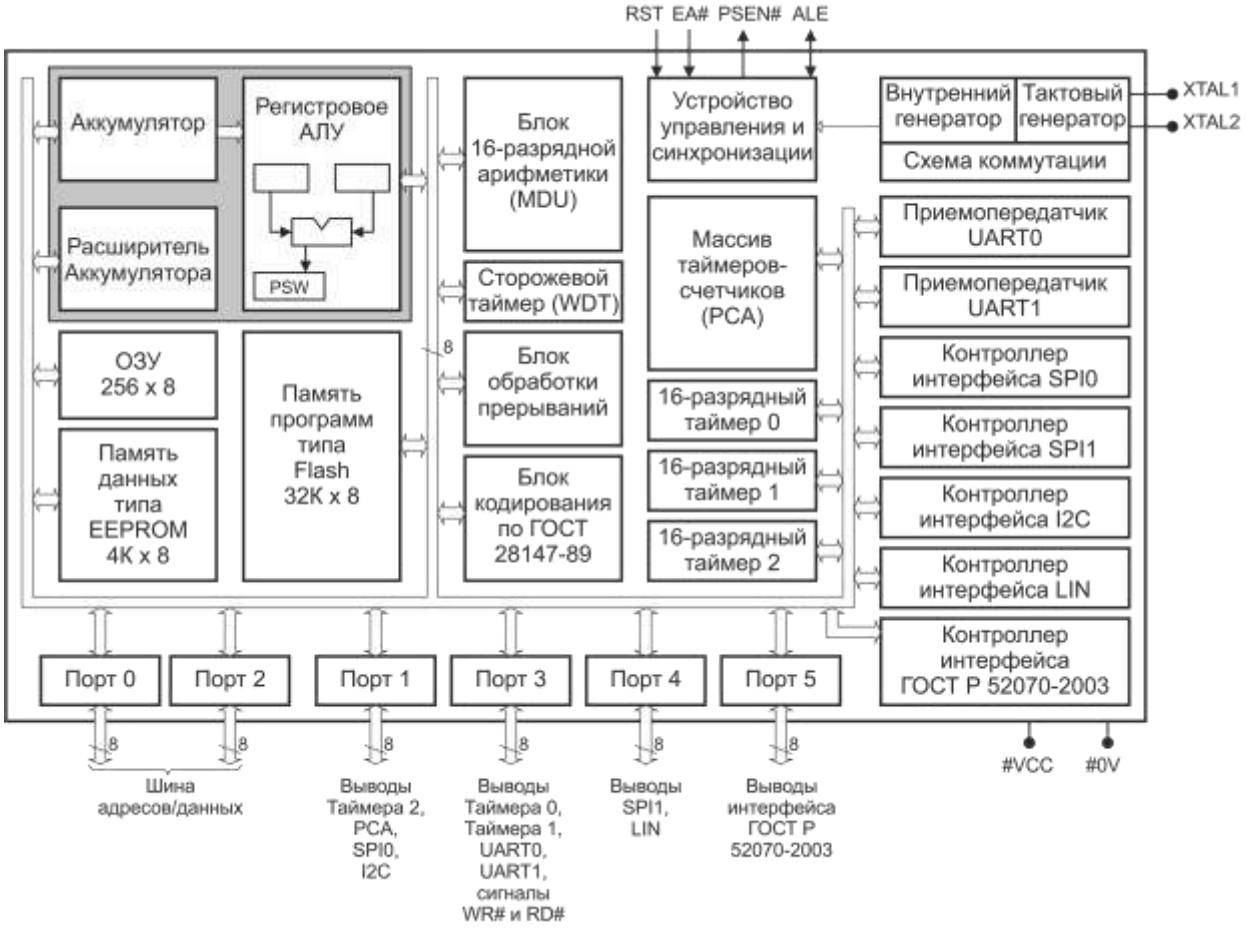


Рисунок 2.1 – Структурная схема 1882ВМ1Т

В составе микроконтроллера имеется блок обработки прерываний, сторожевой таймер, таймеры-счетчики, а также устройство управления и синхронизации и генератор тактового сигнала.

Периферия представлена приемопередатчиками типа UART, модулями последовательных интерфейсов типа I2C, SPI, LIN, последовательным интерфейсом ГОСТ Р 52070–2003 и шестью двунаправленными портами ввода/вывода.

Для подключения кварцевого резонатора к тактовому генератору используются выводы XTAL1 и XTAL2.

Для разрешения работы внешней памяти программ используется вывод PSEN#, который активируется только при обращении к внешнему ПЗУ.

Стробирование адреса внешней памяти осуществляется сигналом ALE с одноименного вывода.

Отключение внутренней памяти программ управляется сигналом EA#. Низкий уровень сигнала на этом входе заставляет микроконтроллер выполнять программу только из внешнего ПЗУ, игнорируя внутреннее.

Входом общего сброса является вывод RST.

Порт 0 (P0) и порт 2 (P2) являются 8-битными двунаправленными портами ввода-вывода информации. При работе с внешними ОЗУ и ПЗУ в режиме мультиплексирования по линиям порта 0 выдается младшая часть адреса, а по линиям порта 2 – старшая часть адреса (при 16-битовой адресации), после чего осуществляется передача или прием данных по порту 0.

Порт 1 (P1) является 8-битным двунаправленным портом ввода-вывода, каждый разряд которого может быть, независимо от других, запрограммирован как «вход», так и

как «выход». Выводы порта могут выполнять альтернативные функции, а именно, являться входами и выходами таймера 2, PCA и модулей интерфейсов I2C и SPI0.

Порт 3 (P3) является 8-битным двунаправленным портом, аналогичным порту 1. Выводы порта могут выполнять альтернативные функции, а именно, являться входами и выходами таймера 0, таймера 1, приемопередатчиков типа UART0 и UART1, а также выдавать сигналы управления внешней памятью WR# и RD# и принимать сигналы внешних прерываний.

Порт 4 (P4) является 8-битным двунаправленным портом, аналогичным порту 3. Выводы порта могут выполнять альтернативные функции, а именно, являться входами и выходами интерфейсов SPI1 и LIN.

Порт 5 (P5) является 8-битным двунаправленным портом, аналогичным порту 3. Выводы порта могут выполнять альтернативные функции – являться входами и выходами последовательного интерфейса ГОСТ Р 52070–2003.

Более подробное описание всех функциональных блоков микроконтроллера и управляющих регистров приводится в соответствующих разделах настоящего руководства.

2.3 Описание выводов XTAL1, XTAL2, PSEN#, ALE, EA#, RST

Выводы XTAL1 и XTAL2 являются выводами для подключения кварцевого резонатора. Вывод XTAL1 является входом для инвертирующего усилителя генератора и входом тактовых сигналов при работе в режиме внешней синхронизации. Вывод XTAL2 является выходом для инвертирующего усилителя генератора.

Вывод PSEN# является выходом сигнала разрешения внешней памяти программ (PSEN#), который выдается только при обращении к внешнему ПЗУ программ. Сигнал PSEN# с активным низким уровнем является сигналом чтения кода из внешней памяти программ. Когда микроконтроллер выполняет команду из внешней памяти, то PSEN# за каждый машинный цикл активизируется дважды, за исключением, когда PSEN# дважды пропускается во время каждого доступа к внешней памяти данных.

Вывод ALE является выходом сигнала разрешения защелкивания адреса (ALE). Сигнал ALE является выходным импульсом для защелкивания (по заднему фронту) младших разрядов адреса во время доступа к внешней памяти. Этот вывод является также входом программирующего импульса (PROG) во время программирования памяти программ типа Flash. В обычном режиме сигнал ALE выдается с постоянной частотой, равной 1/6 частоты тактового генератора, и может использоваться для внешнего задания времени или тактирования. Однако, следует иметь в виду, что один импульс ALE пропускается во время каждого доступа к внешней памяти данных.

По желанию, ALE можно отключить, записав единицу в бит DISALE регистра AUXR. В этом случае, сигнал ALE будет активен только во время выполнения команд MOVX и MOVC, в остальное время на выводе ALE будет удерживаться высокий уровень сигнала.

Вывод EA# является входным сигналом выбора режима работы с внутренней памятью программ или с внешней. Для разрешения доступа к внешней памяти программ с адресами, лежащими в диапазоне от 0000h до FFFFh, вывод EA# должен быть подсоединен к земле (0V). Для обращения к внутренней памяти программ вывод EA# должен быть подсоединен к питанию (#VCC). Во время параллельного (12-вольтового) программирования памяти программ типа Flash, на вывод EA# подается напряжение 12 В (VPP), формирующее разрешение программирования.

Вывод RST является входом сброса микроконтроллера. Наличие высокого уровня на выводе RST в течение, по крайней мере, двух машинных циклов, пока генератор продолжает работать, вызывает сброс устройства.

3 Порты ввода-вывода

Все порты микроконтроллера являются двунаправленными 8-разрядными портами ввода-вывода. Каждый из них содержит защелку вывода (регистр порта из области SFR), выходную цепь и входной буфер. Выводы портов, помимо ввода-вывода информации, могут выполнять альтернативные функции. Альтернативные функции портов включаются, если в соответствующие им регистры-защелки записаны единицы, а для порта 1 дополнительно установлены биты регистра P1SEL. Выходные цепи портов 0 (P0) и 2 (P2) и входной буфер порта 0 используются для доступа к внешней памяти (через порт 0 выдается младший байт адреса, мультиплексированный с байтом данных для записи или чтения, а через порт 2 – старший байт адреса). В остальных случаях через порт 2 выдается содержимое регистра P2. Во время программирования и верификации памяти программ типа Flash, через порт 0 передаются байты кодов (во время верификации необходимы внешние активные нагрузки). Порты 4 (P4) и 5 (P5) аналогичны порту 3. Управление состоянием выводов портов осуществляется посредством регистров IOPORT0, IOPORT1, IOPORT2, IOPORT3, IOPORT4 и IOPORT5.

3.1 Структура и функционирование портов

На рисунках 3.1 – 3.4 показаны функциональные схемы каналов с защелками выводов и буферами ввода-вывода каждого порта. Защелка вывода (один бита регистра порта) представляет собой D-триггер, тактируемый синхросигналом («запись в защелку») и захватывающий значение одного, соответствующего ему, разряда внутренней шины. С выхода Q триггера читается содержимое защелки (состояние бита регистра порта считывается по сигналу «чтение защелки»). С вывода порта считывается его состояние по сигналу «чтение вывода».

Как показано на рисунках 3.1 и 3.3, выходные цепи портов 0 и 2 переключаются сигналом «адрес» (с выхода защелки вывода) или «адрес/данные» в зависимости от сигнала «управление».

Во время работы с внешней памятью регистр P2 порта 2 остается без изменений, в то время как в регистр P0 порта 0 записываются единицы.

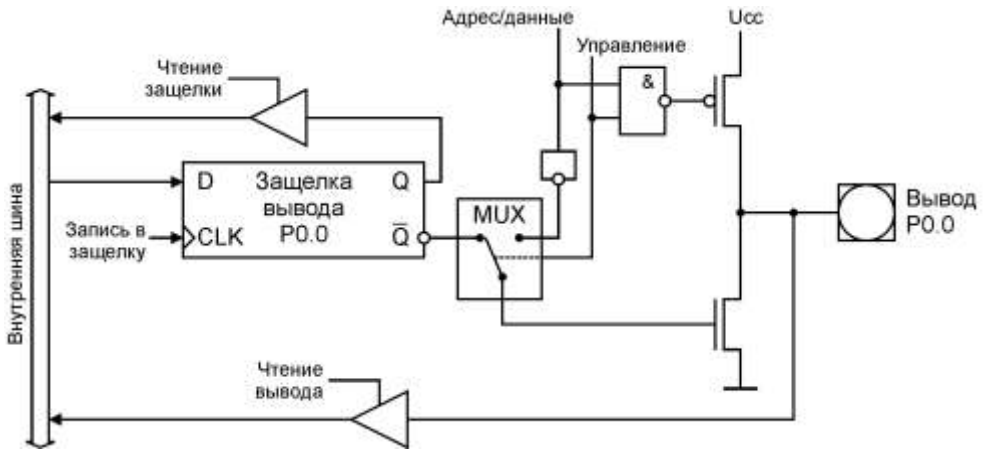


Рисунок 3.1 – Функциональная схема канала порта 0

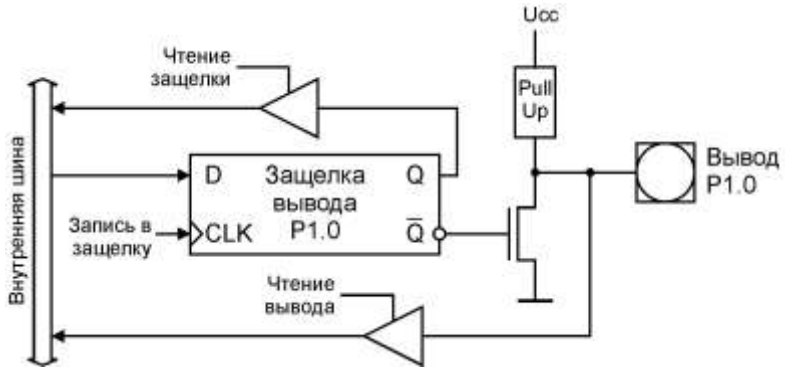


Рисунок 3.2 – Функциональная схема канала порта 1

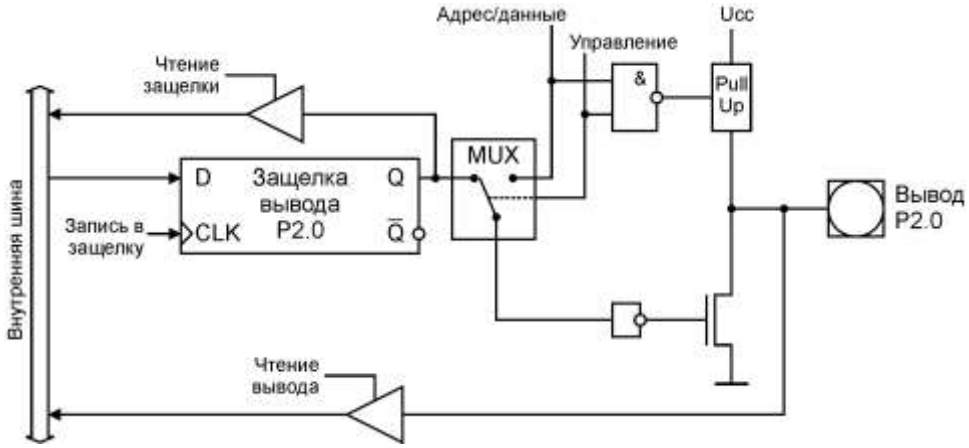


Рисунок 3.3 – Функциональная схема канала порта 2

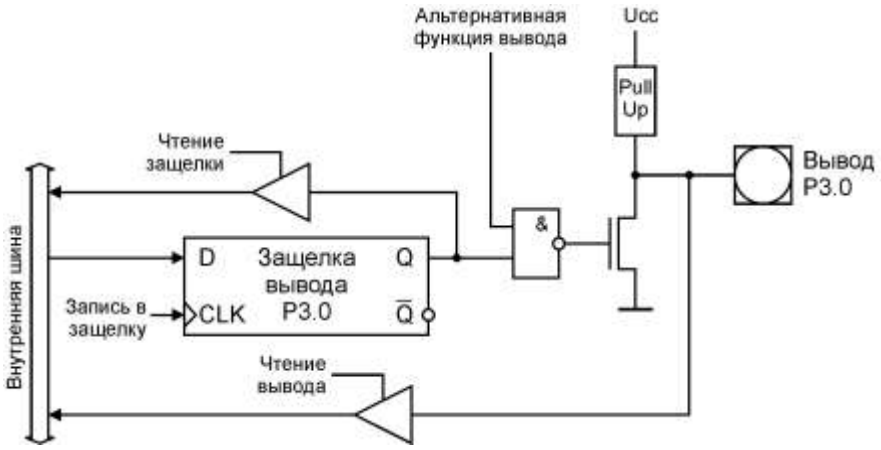


Рисунок 3.4 – Функциональная схема канала порта 3

Как видно из рисунка 3.4, если защелка вывода порта 3 содержит «1», то выходная цепь управляется сигналом «альтернативная функция вывода».

Порты 1, 2 и 3 имеют внутренние подтягивающие схемы pull-up (на рисунках 3.2 – 3.4 обозначены прямоугольниками с надписью «Pull Up»). Выводы порта 0, при использовании их как входов-выходов общего назначения (не при передаче адреса/данных при работе с внешней памятью), работают как выводы с открытым стоком.

Каждый канал может независимо работать как вход или выход (за исключением портов 0 и 2, которые не могут работать как порты ввода-вывода общего назначения в случае, если они используются для вывода адреса и данных при работе с внешней

памятью). Для работы канала на прием («вход») соответствующая защелка вывода должна содержать единицу, которая закрывает выходной транзистор. Отсюда следует, что на выводе порта (справедливо для портов 1, 2 и 3) будет удерживаться высокий уровень сигнала за счет подтягивающей схемы pull-up, который, однако, может быть сброшен внешним сигналом. Подтягивающая схема pull-up показана на рисунке 3.5.

Порт 0 не имеет подтягивающей схемы (см. рисунок 3.1). Подтягивающий транзистор в выходной цепи канала используется только, когда на вывод выдается единица во время обращения к внешней памяти. В остальных случаях транзистор закрыт. Следовательно, выводы порта 0, работающие как выходы, являются выводами с открытым стоком. Единица, записанная в защелку порта 0, закрывает оба транзистора выходной цепи и переводит вывод в состояние с плавающим потенциалом. В таком случае вывод может использоваться как вывод с третьим состоянием.

Структура и функционирование портов 4 и 5 аналогична структуре и функционированию порта 3.

После сброса все регистры портов содержат значение FFh.

Во время выполнения команд, изменяющих состояние регистров портов, новые значения защелкиваются в регистры в самом последнем цикле выполнения команды. На выводах микросхемы данные появляются только в начале выполнения следующей команды.

Каждый порт может быть нагружен восемью стандартными ТТЛ входами. При использовании в качестве обычного порта каждый из выводов должен быть подключен к шине питания через резистор номиналом 10 кОм (см. рисунок 3.6).

В случае порта 0, при использовании его в качестве шины адреса/данных, необходимость в резисторах отпадает.

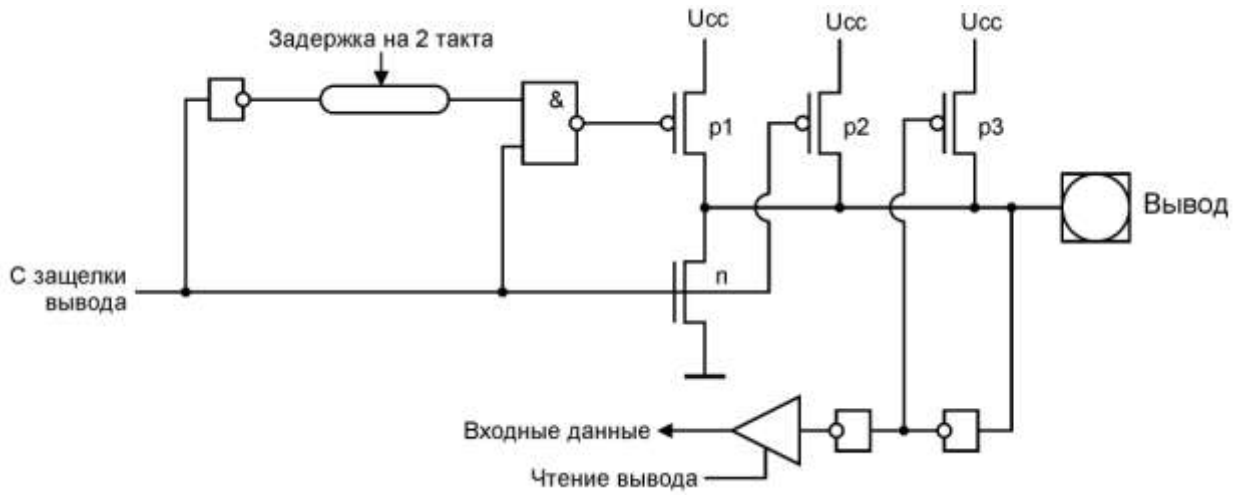


Рисунок 3.5 – Подтягивающая схема pull-up

Примечание – На рисунке 3.5 показан сигнал, приходящий с защелки вывода и закрывающий транзистор p0 (который задерживается при этом на два такта, и только по истечении этого времени транзистор p1 закрывается). В течение времени, пока транзистор p1 открыт, он поддерживает в открытом состоянии транзистор p3 через инвертор, формируя таким образом защелку, хранящую «1». Транзистор p2 тоже открыт.

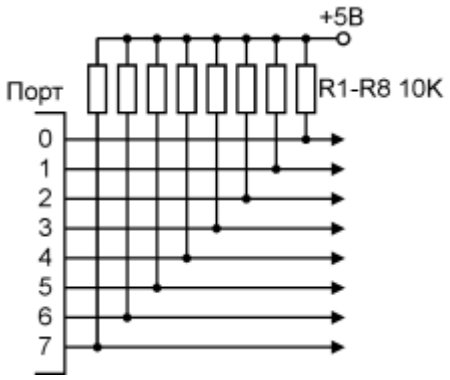


Рисунок 3.6 – Использование линий порта для ввода/вывода

Обращение к портам ввода-вывода возможно с использованием команд, оперирующих с байтом, отдельным битом и произвольной комбинацией бит. При этом в тех случаях, когда порт является одновременно операндом и местом назначения результата, автоматически реализуется специальный режим "чтение – модификация – запись". Этот режим обращения предполагает ввод сигналов не с внешних выводов порта, а из его регистра-защелки, что позволяет исключить неправильное считывание ранее выведенной информации. Этот механизм реализован в командах ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ, CLR, SETB, MOV.

4 ОЗУ и ПЗУ

Внутреннее ПЗУ микроконтроллера представлено памятью программ типа Flash объемом 32 Кбайт и памятью данных типа EEPROM объемом 4 Кбайта. Внутреннее ОЗУ имеет объем 256 байт. Все три памяти являются самостоятельными и независимыми друг от друга устройствами. Они физически и логически разделены, имеют различные механизмы адресации, работают под управлением различных сигналов и выполняют разные функции. В память программ типа Flash, кроме команд, могут записываться константы, управляющие слова инициализации, таблицы перекодировки входных и выходных переменных и т.п. Доступ к памяти программ типа Flash возможен только в режимах последовательного или параллельного программирования. Адресация памяти программ осуществляется как с использованием счетчика команд PC, так и с использованием специального двухбайтового регистра указателя данных DPTR.

Допускается подключение внешней памяти программ объемом до 64 Кбайт и памяти данных с таким же объемом, что обусловлено использованием 16-разрядной шины адреса.

4.1 ОЗУ

ОЗУ, доступное пользователю, имеет объем 256 байт. Область регистров специальных функций (регистры SFR) имеет оконный способ адресации. Каждое окно открывает область объемом 128 байт. Для переключения между окнами используется регистр WSR, расположенный по адресу FFh (см. рисунок 4.1).

Первые 32 байта (адреса с 00h по 1Fh) организованы в четыре банка регистров, каждый из которых состоит из 8 регистров (R0, R1, R2, R3, R4, R5, R6, R7). В любой момент времени программно доступен только один банк регистров, номер которого определяется битами RS0 и RS1 регистра PSW. Далее в пространстве с 20h до 2Fh расположены 16 регистров, допускающие побитную адресацию. 128 бит этой области адресуются по порядку с 00h до 7Fh – нулевому биту регистра; расположенному по адресу 20h соответствует номер 00h, а седьмому биту регистра, расположенному по адресу 2Fh – номер 7Fh. Для записи байта в эту область необходимо указывать адрес также, как для записи любого другого, не адресуемого побитно регистра. Для записи бита, например, командой SETB, следует указывать порядковый номер бита.

Так, например, записать число 10h в регистр с адресом 23h можно так:

```
MOV     23h, #10h,
```

а установить в единицу второй бит того же регистра – так:

```
setb        1Ah.
```

В адресном пространстве с 30h до 7Fh располагаются 80 регистров общего назначения. Регистры страницы, расположенные в адресном пространстве с 00h по 7Fh, могут адресоваться как прямо, так и косвенно.

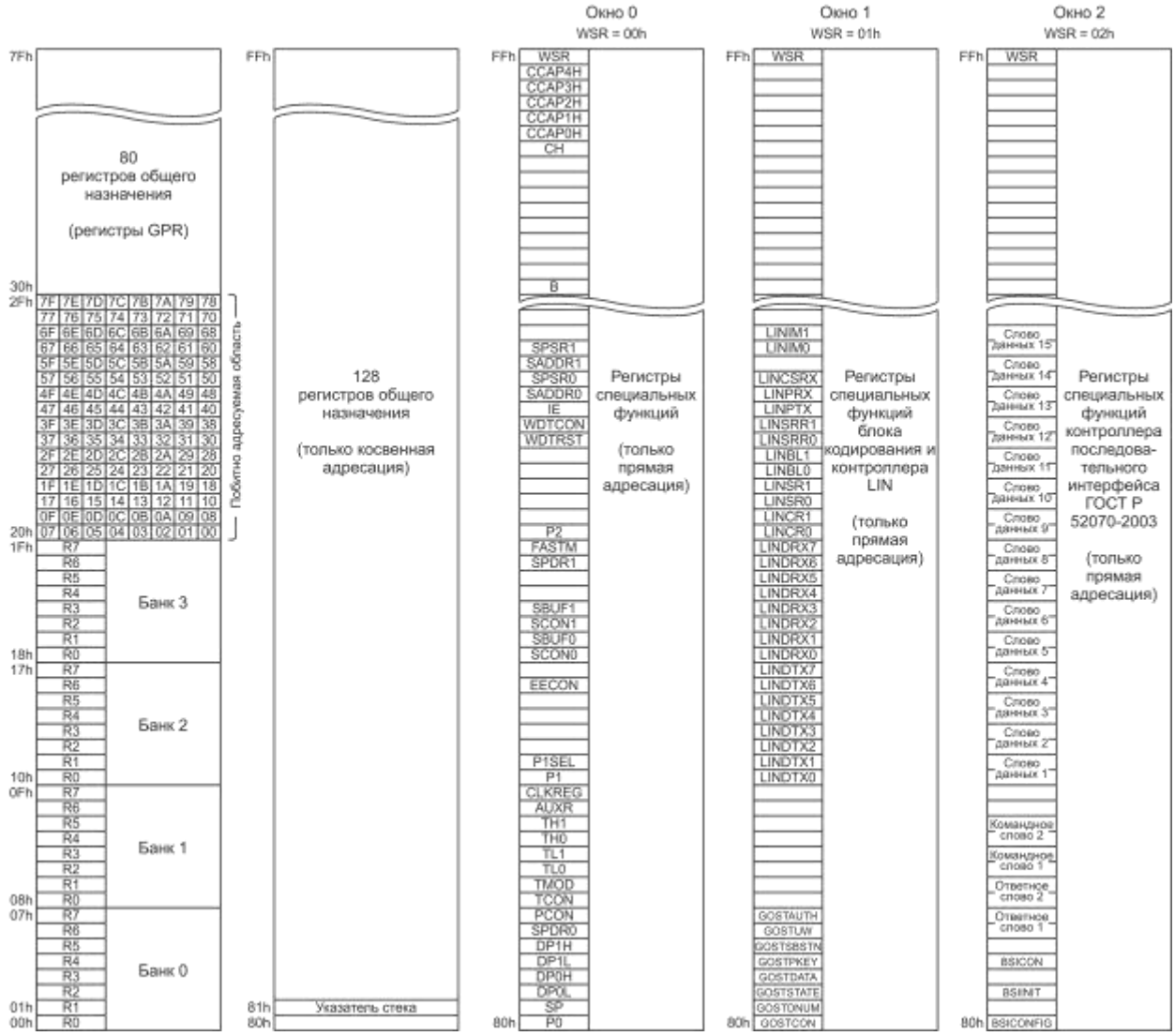


Рисунок 4.1 – Оконная организация ОЗУ

Вторая половина ОЗУ (адреса с 80h по FFh) – это 128 регистров общего назначения, адресуемые только косвенно. По адресу 81h располагается указатель стека.

Область регистров специальных функций (область SFR) – это три окна объемом 128 байт каждое. В каждый момент доступно только одно окно, заданное регистром WSR. В пределах окна доступны адреса от 80h до FFh. Регистр SFR могут адресоваться только прямо. Описание регистров приведено в приложении А.

Регистры специальных функций

Регистры предназначены для управления ходом вычислительных операций, а также отвечают за инициализацию, настройку и управление портами ввода/вывода, таймерами и другими блоками микроконтроллера, управляют обслуживанием прерываний. Некоторые регистры SFR допускают побитовую адресацию. При этом обращение к отдельным битам таких регистров возможно как с помощью обычных функций для работы с байтами, так и с помощью команд операций с битами. Например, в результате выполнения следующих команд будет установлен четвертый бит регистра TCON:

orl TCON, #10h или setb TCON.4.

На рисунке 4.2 показаны побитно адресуемые регистры SFR, с указанием их названий и адресов. Биты каждого из указанных регистров программно доступны и могут

устанавливаться и очищаться независимо. Каждый бит имеет название и адрес, которые могут использоваться для обращения к этим битам соответствующими командами. Например, чтобы установить четвертый бит порта 5 можно поступить так:

setb P5.4 или setb 0ECh

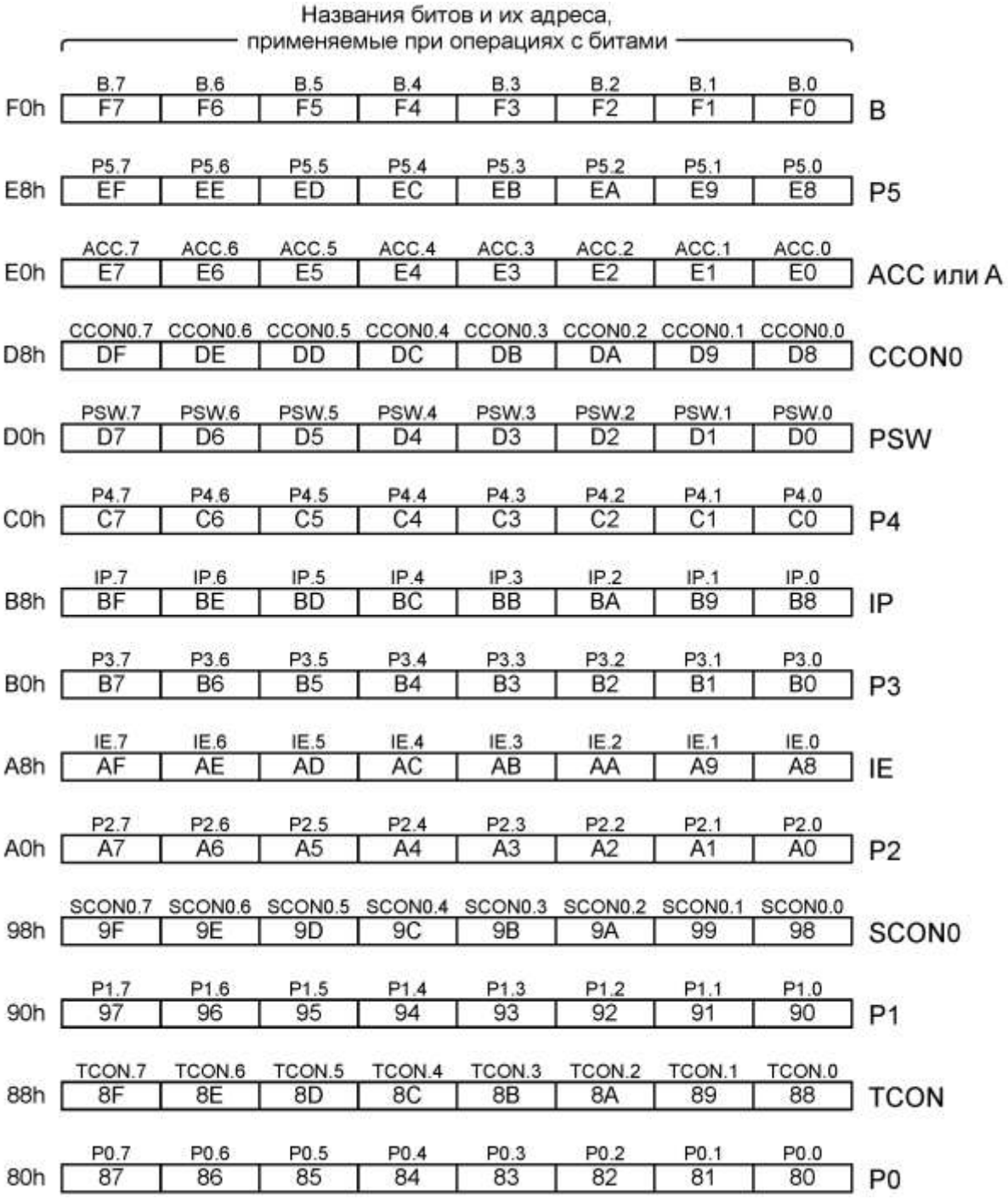


Рисунок 4.2 – Побитно адресуемые регистры области SFR

При обращении к битам аккумулятора следует использовать мнемонику «ACC». При обращении к аккумулятору как к байту, можно применять мнемонику «ACC» или «A».

Аккумулятор, регистр B, слово состояния процессора, указатель стека и регистр указателя данных играют важную роль при выполнении различных операций.

Аккумулятор (ACC или A) и расширитель аккумулятора (регистр B)

Аккумулятор является источником операнда и местом фиксации результата при выполнении большинства команд. Некоторые команды, например, сдвига, проверки на ноль и другие, могут быть выполнены только с использованием аккумулятора.

Расширитель аккумулятора – это вспомогательный регистр, являющий источником операнда и местом фиксации результата при операциях умножения и деления. При умножении регистр B является множителем и местом сохранения старшего байта произведения. При делении регистр B является делителем и местом сохранения остатка от деления.

Регистр слова состояния процессора (PSW)

Слово состояния процессора является регистром, биты которого являются флагами (признаками операций), отражающими работу АЛУ.

Регистр указателя стека (SP)

Указатель стека является 8-разрядным регистром, выполняющим функцию адресации данных, расположен по адресу 81h в ОЗУ.

Работа стека организована с применением косвенной адресации, т.е. в регистре SP размещается адрес ячейки начала стека (см. рисунок 4.3, а)). Указатель стека может адресовать любую область внутренней памяти данных ОЗУ микроконтроллера, причем в случае попадания адреса в диапазон 80h – FFh, обращение происходит к параллельному пространству SFR (см. рисунок 4.3, б)). При адресации содержимое указателя стека инкрементируется перед выполнением команд PUSH и CALL и декрементируется после выполнения команд POP и RET. В процессе инициализации микроконтроллера после сброса в указатель стека автоматически загружается значение 07h. Это значит, что если программно не переопределено содержимое указателя стека, то первый элемент данных в стеке будет располагаться в ячейке памяти с адресом 08h.

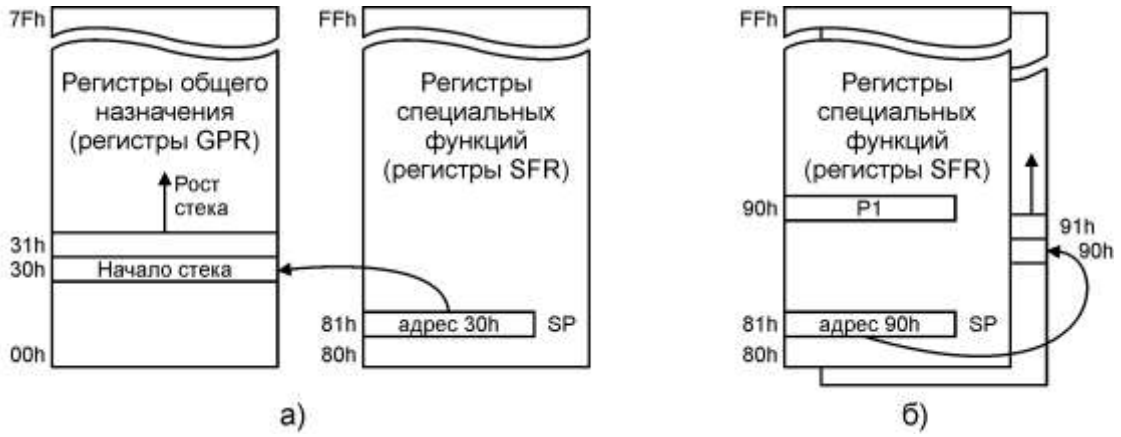


Рисунок 4.3 – Адресация указателем стека SP

Нужно отметить одну важную особенность – при загрузке данных в стек – адрес растет вверх, поэтому если в программе используются банки регистров 1, 2 или 3, то указатель стека следует проинициализировать адресом из неиспользуемой области памяти, например 30h, чтобы не перезаписать содержимое регистров одного из банков.

Регистр указателя данных (DPTR)

Регистр DPTR является условным 16-разрядным регистром, который обычно используется для фиксации 16-разрядного адреса в операциях с обращением к внешней и внутренней памяти данных. Под указателем данных DPTR подразумевается функционирование одного из двух банков регистров, в каждом по два регистра – DP0H, DP0L и DP1H, DP1L. Каждая пара регистров представляет собой 16-разрядный адрес (DP0H/DP1H – старший байт, DP0L/DP1L – младший байт), который используется при адресации. Выбор того или иного банка регистров осуществляется битом DPS регистра EECON.

Регистр может программироваться непосредственно:

```
MOV DPTR, #0105h
```

или побайтно, посредством двух составляющих его регистров:

```
MOV DP0H, #01h      ;(или DP1H, если установлен DPS в регистре EECON)
MOV DP0L, #05h      ;(или DP1L, если установлен DPS в регистре EECON).
```

Обращение к ячейкам ОЗУ возможно несколькими способами – непосредственной и косвенной адресацией. При непосредственной адресации адрес ячейки сам является операндом соответствующей команды.

Косвенная адресация реализуется при помощи любого из регистров R0, R1 выбранного банка регистров. Обращение к ячейке памяти происходит посредством выбранного регистра, в который предварительно был записан адрес ячейки.

Несколько примеров обращения к ОЗУ.

1 Использование команды MOV с непосредственной адресацией:

а) MOV 84h, #0FFh

Поскольку адрес ячейки, в которую производится прямая запись, попадает в диапазон 80h – FFh (область SFR), то будет выполнена запись значения FFh в регистр DP1L, расположенный по адресу 84h;

б)
```
MOV  FFh, #02h    ; Переключение на окно 2 (WSR = 02h)
MOV  84h, #0FFh.
```

После включения второго окна значение FFh будет записано в регистр BSICON.

2 Использование команды MOV с косвенной адресацией:

```
MOV  R0, #84h
MOV  @R0, #0FFh
```

Адрес ячейки, в которую производится запись, попадает в диапазон 80h – FFh (область SFR), но поскольку запись косвенная, то значение FFh будет записано в регистр, расположенный по адресу 84h в области регистров общего назначения, независимо от состояния регистра WSR.

4.2 ПЗУ

ПЗУ, доступное пользователю, представляет собой память данных типа EEPROM объемом 4 Кбайта и память программ типа Flash объемом 32 Кбайта.

Память данных

Для адресации памяти данных типа EEPROM (как внутренней, так и внешней) предусмотрены два 16-разрядных регистра, каждый из которых может функционировать как регистр указателя данных DPTR.

Первый регистр получается конкатенацией двух регистров DP0H (старший байт) и DP0L (младший байт) первого банка, второй – DP1H (старший байт) и DP1L (младший байт) второго банка. Выбор того или иного банка регистров осуществляется битом DPS регистра EECON.

Выбор внутренней памяти данных типа EEPROM или внешней памяти данных определяется битом EEMEN регистра EECON. Адреса внутренней памяти данных находятся в диапазоне 000h – FFFh. Для обращения к памяти используется команда MOVX.

Память данных позволяет записывать данные без предварительного стирания ячеек памяти. Память данных имеет 32-байтный буфер. При побайтной/постраничной записи данные всегда предварительно загружаются в буфер, откуда по команде записи переписываются в память данных.

Побайтное обращение

Записать в EECON значение «18h» («побайтная запись»).

Записать в DP0H и DP0L адрес байта.

По команде MOVX байт будет загружен в буфер, после чего сразу же будет инициирован процесс записи этого байта в память данных. Начало записи сопровождается

обнулением бита RDY/BSY# в регистре EECON. До тех пор, пока будет идти процесс записи, этот бит будет оставаться в нулевом состоянии. По окончании записи бит RDY/BSY# установится в единицу.

```
MOV EECON, #08h          ; режим побайтной записи

MOV DP0H, #01h           ; старший байт адреса
MOV DP0L, #05h           ; младший байт адреса

MOV A, #77h              ; записываемый байт
MOVX @DPTR, A            ; загрузка байта и инициирование записи

busy:                    ; ожидание окончания записи
        MOV A, EECON
        jnb ACC.1, busy
```

Для чтения, после указания адреса, используется команда MOVC:

```
MOV A, #0h
MOVC A, @A + DPTR
```

Постраничная запись

Позволяет записывать 32 байт за время, равное времени записи одного байта. Процесс заключается в последовательной загрузке байт в буфер временного хранения (с указанием адреса для каждого байта), с последующей записью содержимого буфера в память данных.

Запись в 32 байта страницы указывает на то, что все желаемые данные были загружены в буфер, и инициирует процесс записи буфера в указанную страницу памяти данных.

Для включения режима постраничной записи в регистр EECON следует записать значение 38h.

Пример заполнения 32 байт буфера значением «55h» в цикле с последующей записью.

В регистр DPTR записывается начальный адрес.

Далее число «55h» последовательно загружается в каждый байт буфера, начиная с нулевого. Как только регистр DP0L достигает значения, совпадающего с адресом 32 байта буфера, загрузка байта по этому адресу инициирует запись буфера в память данных.

Начало записи сопровождается обнулением бита RDY/BSY# в регистре EECON. До тех пор, пока будет идти процесс записи, этот бит будет оставаться в нулевом состоянии. По окончании записи установится бит RDY/BSY#. Этот бит используется для отслеживания момента завершения записи каждого байта.

По окончании записи, регистр DPTR будет проинкрементирован еще раз, и по условию сравнения его младшего регистра DP0L с числом 20h, цикл записи будет завершен.

```
        MOV DPTR, #0100h

write_loop:
        MOV A, #55h
        MOVX @DPTR, A

busy:                    ; ожидание окончания записи
        MOV A, EECON
```

```
jnb ACC.1, busy

inc DPTR

MOV A, DP0L
cjne A, #20h, write_loop
```

Память программ

Доступ к внутренней памяти программ типа Flash возможен только в режимах последовательного или параллельного программирования. Режимы описаны в разделе 20.

5 Внешняя память программ

К микроконтроллеру может быть одновременно подключено до 64 Кбайт внешней памяти данных и до 64 Кбайт внешней памяти программ (см. рисунок 5.1).

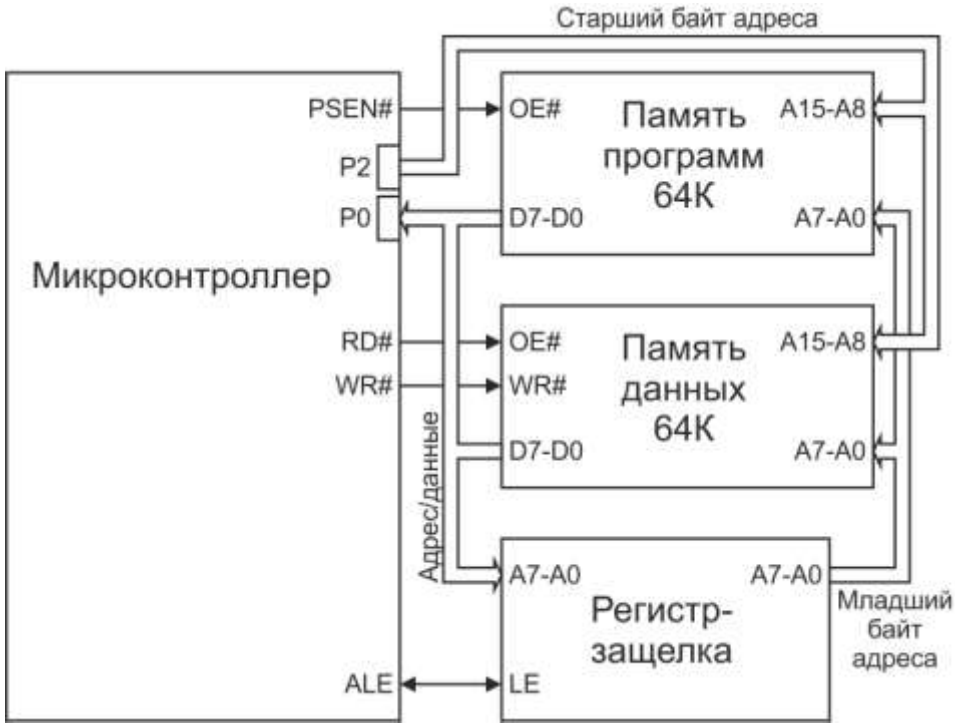


Рисунок 5.1 – Подключение внешней памяти

Объем обусловлен 16-разрядной шиной данных. Адресация внешней памяти осуществляется методом косвенной адресации при помощи регистров R0 и R1 (выбранного банка регистров заданной страницы ОЗУ) или при помощи регистра указателя данных DPTR. Чтение памяти программ стробируется сигналом PSEN#, чтение и запись памяти данных – сигналами RD# и WR# соответственно. Все три стробирующих сигнала имеют активный низкий уровень. Во время каждого доступа к внешней памяти пропускается один импульс сигнала ALE.

Стробирующим сигналом считывания памяти программ является сигнал PSEN#. За каждый машинный цикл сигнал PSEN# активизируется дважды. Временная диаграмма обращения к внешней памяти программ представлена на рисунке 5.2.

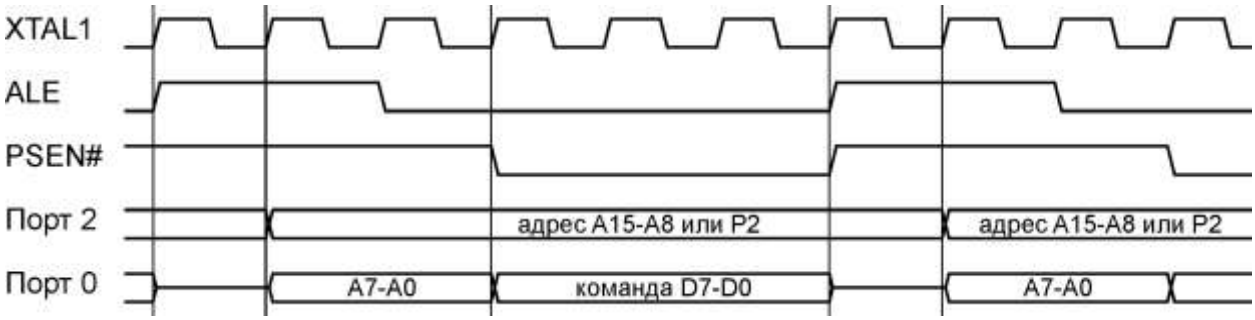


Рисунок 5.2 – Обращение к внешней памяти программ. Цикл чтения

В режиме, когда порт 0 работает на выдачу и прием адреса/данных, его выводы не являются выводами с открытым стоком и потому не требуют внешних подтягивающих резисторов. При чтении памяти программ через порт 0 выдается младший байт адреса и принимается байт данных, а через порт 2 – старший байт адреса, если используется 16-разрядная адресация или значение регистра P2. Для фиксации младшего байта адреса во внешнем регистре-защелке используется сигнал ALE с одноименного вывода ALE.

В момент перехода сигнала ALE в активный уровень порт 0 переводится в третье состояние. Через один такт XTAL1 после появления сигнала ALE на выводах порта 0 выставляется младший байт адреса (на рисунке 5.6 обозначен как «A7-A0»). На выводах порта 2 в этот же момент времени выставляется старший байт адреса («A15-A8») или, если используется 8-разрядная адресация, остается значение, хранящееся в регистре P2 («P2»).

По заднему фронту сигнала ALE младший байт адреса записывается в регистр-защелку. Таким образом, на выводах внешней памяти программ формируется полный адрес и далее память ожидает появление сигнала чтения, роль которого выполняет сигнал PSEN#.

Появление низкого (активного) уровня сигнала PSEN# является разрешением на выдачу данных (байта команды), и поэтому в этот момент времени на выводах порта 0 появляется значение прочитанного байта (на рисунке 5.2 обозначен как «команда D7-D0»). Переход сигнала PSEN# в неактивное состояние одновременно с сигналом ALE переводит порт 0 в третье состояние, что служит окончанием цикла чтения и (если это требуется) – началом следующего цикла чтения памяти программ.

Если используется 16-разрядная адресация, то старший байт адреса выводится через порт 2, где он сохраняется в течение всего цикла обращения к внешней памяти.

Внешняя память данных

Стробирующими сигналами записи и считывания памяти данных являются сигналы WR# и RD#. Временная диаграмма обращения к внешней памяти данных представлена на рисунке 5.3.

В режиме, когда порт 0 работает на выдачу и прием адреса/данных, его выводы не являются выводами с открытым стоком и потому не требуют внешних подтягивающих резисторов. При чтении памяти данных через порт 0 выдается младший байт адреса и принимается байт данных, а через порт 2 – старший байт адреса, если используется 16-разрядная адресация или значение регистра P2. Для фиксации младшего байта адреса во внешнем регистре-защелке используется сигнал ALE с одноименного вывода ALE.

В момент перехода сигнала ALE в активный уровень порт 0 переводится в третье состояние. Через один такт XTAL1 после появления сигнала ALE на выводах порта 0 выставляется младший байт адреса (на рисунке 5.3 обозначен как «A7-A0»). На выводах порта 2 в этот же момент времени выставляется старший байт адреса («A15-A8») или, если используется 8-разрядная адресация, остается значение, хранящееся в регистре P2 («P2»).

По заднему фронту сигнала ALE младший байт адреса записывается в регистр-защелку. Таким образом, на выводах внешней памяти данных формируется полный адрес и далее память ожидает появление сигнала записи WR# или чтения RD#.

Появление низкого (активного) уровня сигнала записи WR# является указанием для памяти подготовиться к записи байта данных, после чего через один такт сигнала XTAL1 на выводы порта 0 выставляется байт данных (на рисунке 5.3). Цикл записи обозначен как «D7-D0». В момент перехода сигнала записи в неактивный высокий уровень, данные достоверны и происходит их запись во внешнюю память данных. Сигнал WR# переходит в неактивное состояние на один такт сигнала XTAL1 раньше появления сигнала ALE. Появление сигнала ALE переводит порт 0 в третье состояние, что служит окончанием цикла записи и (если это требуется) – началом следующего цикла обращения к памяти данных.

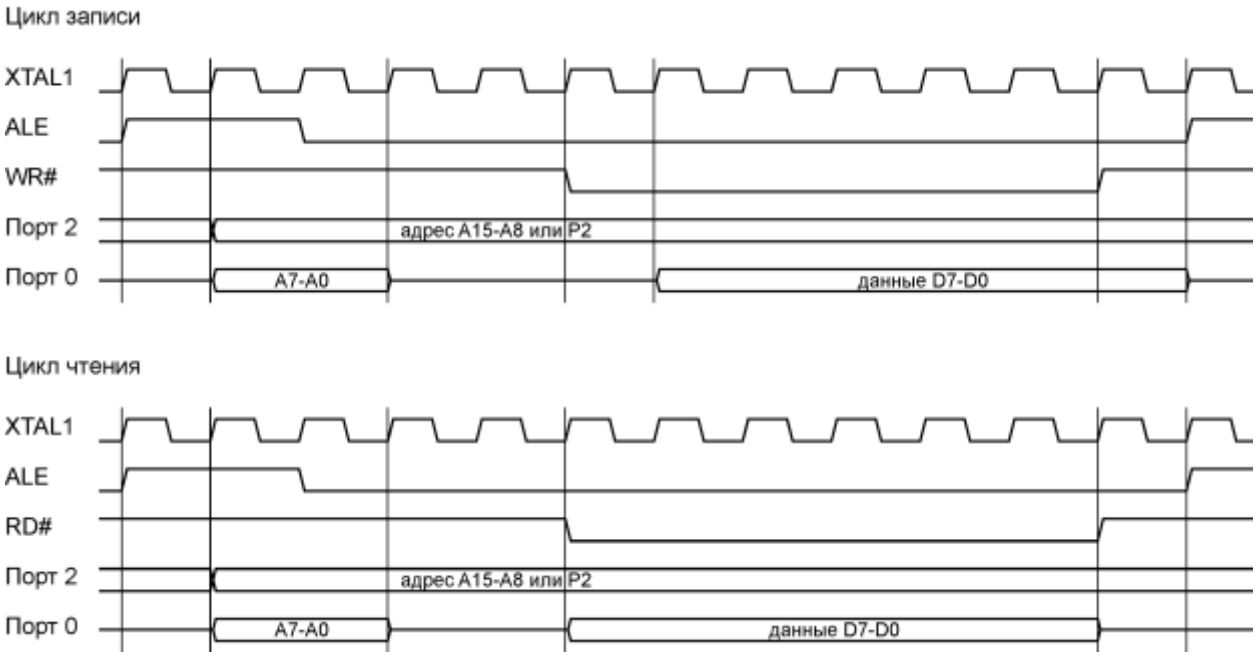


Рисунок 5.3 – Обращение к внешней памяти программ. Циклы чтения и записи

Появление низкого (активного) уровня сигнала чтения RD# является разрешением на выдачу байта данных, и поэтому в этот момент времени на выводах порта 0 появляется значение прочитанного байта (см. рисунок 5.3). Сигнал RD# переходит в неактивное состояние на один такт сигнала XTAL1 раньше появления сигнала ALE. Одновременно со снятием сигнала RD# линии порта 0 переводятся в третье состояние. Появление сигнала ALE служит окончанием цикла чтения и (если это требуется) – началом следующего цикла обращения к памяти данных.

Если используется 16-разрядная адресация, то старший байт адреса выводится через порт 2, где он сохраняется в течение всего цикла обращения к внешней памяти.

Примечание – Во время обращения к внешней памяти (как программ, так и данных) в регистр P0 аппаратно записывается значение FFh, в связи с чем хранимая в нем информация теряется. Это следует учитывать при использовании порта 0. Устройство порта 2 отличается от устройства порта 0, содержимое регистра P2 не теряется при выдаче адреса, что помимо прочего позволяет организовывать так называемый режим постраничного чтения/записи. Единственное ограничение на использование порта 2 таково, что при работе с внешней памятью его выводы не могут быть использованы для обычного ввода-вывода информации, а только для вывода адреса.

6 Устройство управления и синхронизации. Тактовый генератор

Выводы XTAL1 и XTAL2 являются, соответственно, входом и выходом инвертирующего усилителя, который используется в качестве генератора для устройства управления и синхронизации. К выводам XTAL1 и XTAL2 подключается кварцевый резонатор или внешний генератор (см. рисунок 6.1). При работе от внешнего генератора сигнал синхронизации должен быть подан на вход XTAL1 (вход XTAL2 при этом не используется).

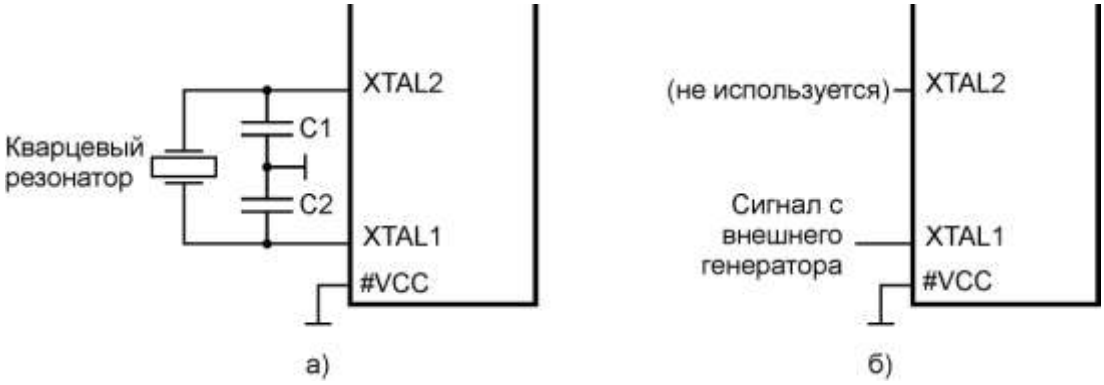


Рисунок 6.1 – Схемы подключения источника

Тактирование микроконтроллера

На рисунке 6.2 представлена схема тактирования микроконтроллера.

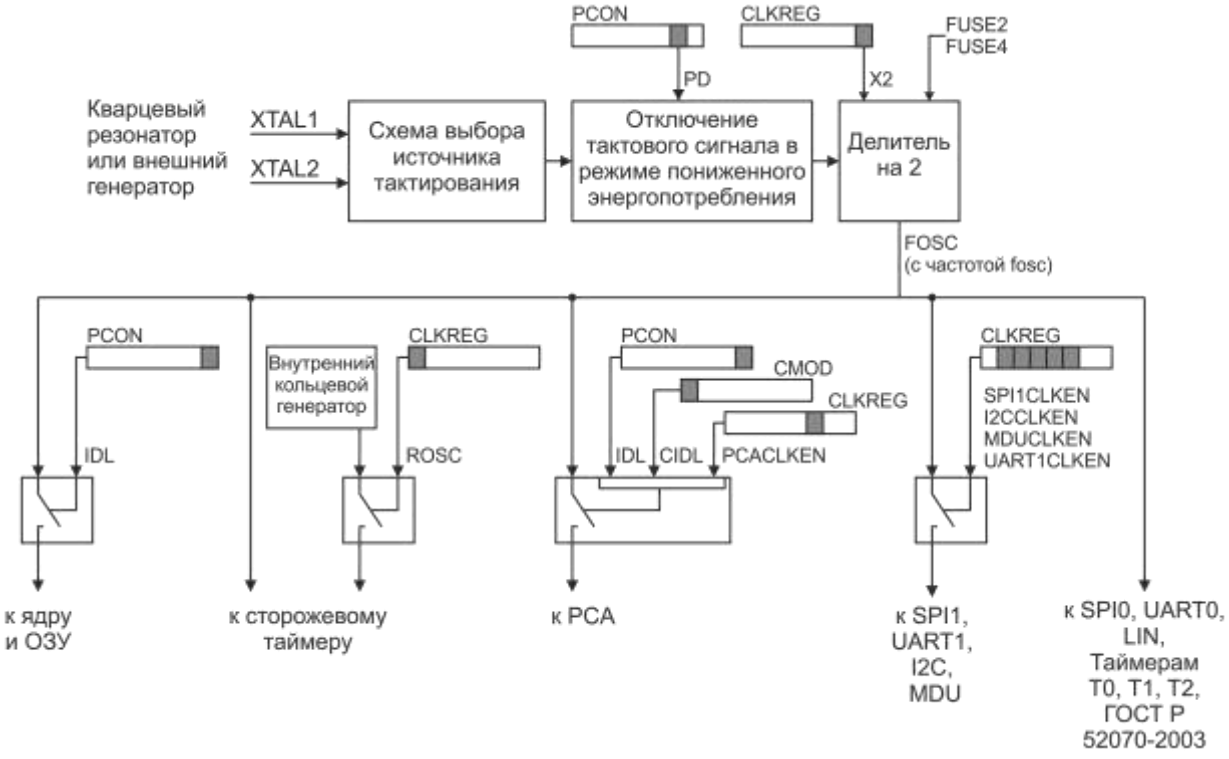


Рисунок 6.2 – Схема тактирования микроконтроллера

Микроконтроллер может работать с включенным делителем частоты (на два) на частотах (внешних) до 24 МГц. При выключенном делителе частоты максимальное значение внешней входной частоты не должно превышать значения 12 МГц.

Устройство управления микроконтроллером формирует машинный цикл фиксированной длительности (см. рисунок 6.3).

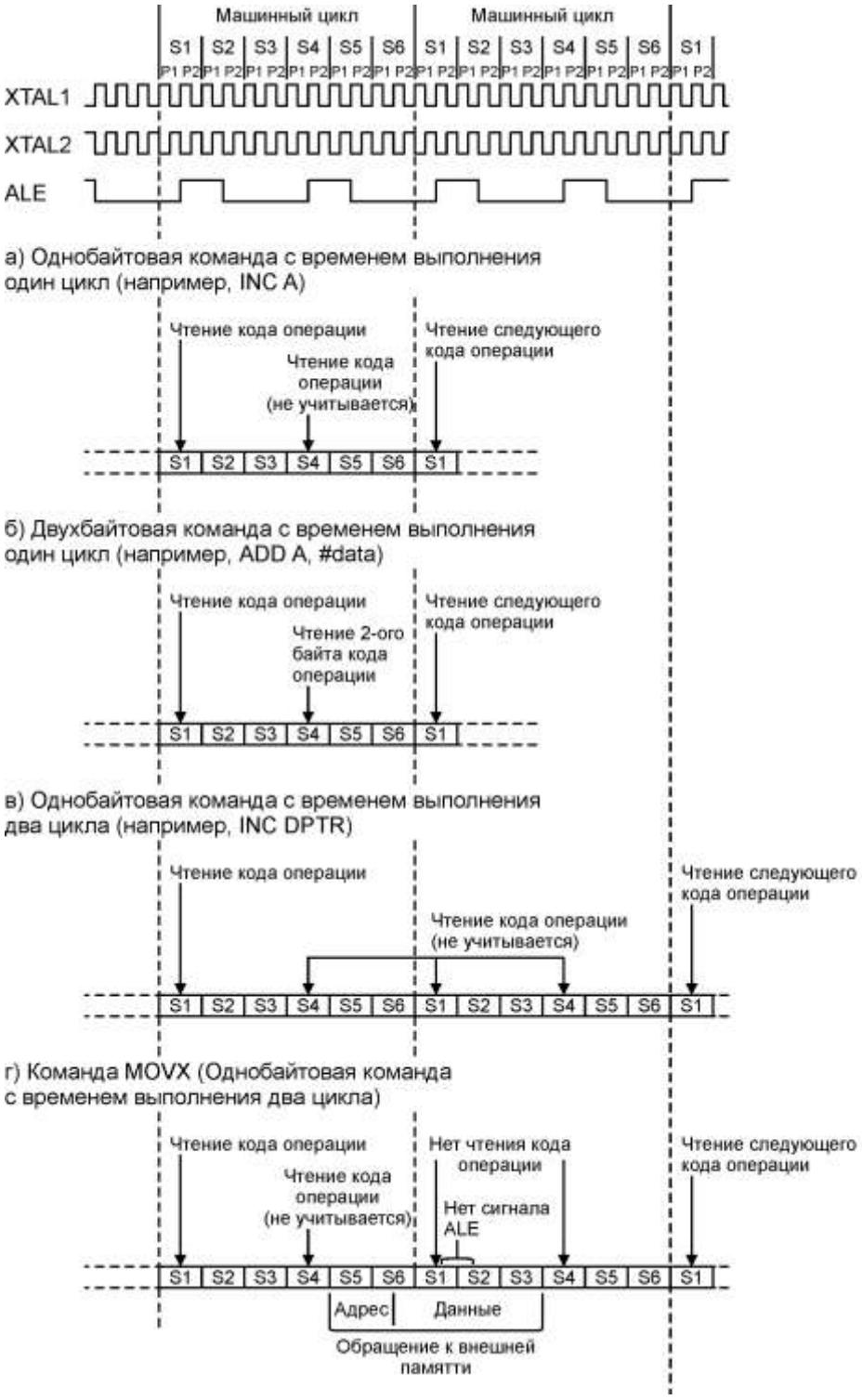


Рисунок 6.3 – Последовательность состояний микроконтроллера

Каждая пара фаз сигнала XTAL1 (P1 и P2) формирует, так называемое, состояние управляющего устройства (S1, S2, …, S6). Шесть состояний формируют машинный цикл. Как показано на рисунке 6.3, в течение каждого машинного цикла сигнал ALE формируется дважды. Выборка команд (чтение кода операции) происходит по сигналу ALE (за исключением операций записи и чтения внешней памяти с использованием

команды MOVX). В микроконтроллере реализована функция ускорения работы ядра, которая доступна через регистр FASTM

Включение ускорения ядра приводит к сокращению времени выполнения некоторых команд и, как следствие – к увеличению производительности микроконтроллера примерно на 20%. Информация о времени выполнения команд в режимах нормальной и ускоренной работы ядра доступна в приложении В настоящего руководства.

Источник синхронизации и режим встроенного делителя частоты можно установить при программировании микроконтроллера (битами конфигурации FUSE2 и FUSE4) или программно – соответствующими битами регистра CLKREG. Также соответствующими битами регистра CLCREG можно управлять синхронизацией блока MDU и периферийных блоков.

Возможность включения делителя частоты на два позволяет микроконтроллеру выполнять один машинный цикл за шесть периодов сигнала синхронизации вместо 12, что важно с точки зрения снижения электромагнитных помех. При включенном делителе максимальная внешняя тактовая частота не должна превышать 24 МГц, а при выключенном делителе – 12 МГц.

7 Блок 16-разрядной арифметики

Блок 16-разрядной арифметики или, другими словами, блок умножения и деления (далее просто блок MDU) обеспечивает быстрое умножение, деление, а также сдвиг и функцию нормализации, независимо и параллельно работе центрального процессора. Наличие этого блока позволяет использовать микроконтроллер в приложениях управления в реальном времени, которые требуют быстрых математических вычислений.

MDU использует в общей сложности 13 регистров (все находятся в области SFR):

MDUCON – регистр управления блока MDU;

MD0, MD1, MD2, MD3, MD4, MD5 – регистры данных;

MR0, MR1, MR2, MR3, MR4, MR5 – регистры результата.

Далее по тексту символ «x» заменяет порядковый номер регистра (от 0 до 5).

Функциональные возможности блока MDU указаны в таблице 7.1.

Таблица 7.1 – Операции блока MDU

<table>
<tr><th colspan="3">Операция</th><th>Результат</th><th>Остаток</th><th>Число циклов синхронизации, требуемых для операции</th></tr>
<tr><td rowspan="4">Деление</td><td rowspan="2">со знаком</td><td>32/16</td><td>32 бита</td><td>16 бит</td><td rowspan="7">2</td></tr>
<tr><td>16/16</td><td>16 бит</td><td>16 бит</td></tr>
<tr><td rowspan="2">без знака</td><td>32/16</td><td>32 бита</td><td>16 бит</td></tr>
<tr><td>16/16</td><td>16 бит</td><td>16 бит</td></tr>
<tr><td rowspan="2">Умножение</td><td>со знаком</td><td>16x16</td><td>32 бита</td><td rowspan="2">–</td></tr>
<tr><td>без знака</td><td>16x16</td><td>32 бита</td></tr>
<tr><td colspan="3">32-разрядная нормализация без знака</td><td>–</td><td>–</td></tr>
<tr><td colspan="3">32-разрядная операция арифметического /логического сдвига</td><td>–</td><td>–</td></tr>
</table>

Блок MDU может рассматриваться как специальный сопроцессор для умножения, деления, нормализации и сдвига. Его работа делится на три фазы:

Фаза первая – Загрузка регистров данных MDx.

В этой фазе операнды загружаются процессором в регистры MDx. Тип вычисления, которое должен выполнить блок MDU, выбирается путем записи 4-разрядного кода в битовое поле OPCODE регистра MDUCON.

Фаза вторая – Выполнение вычисления.

Эта фаза начинается после установки стартового бит START (регистр MDUCON), который фактически выполняет роль флага вычисления, поскольку аппаратно стирается по окончании вычисления. В течение этой фазы, MDU работает параллельно с центральным процессором микроконтроллера. Результат вычислений помещается в регистры результата MRx.

Фаза третья – Считывание результатов из регистров результата MRx.

В этой заключительной фазе центральный процессор микроконтроллера считывает результаты вычисления из регистров MRx. После выполнения следующего вычисления регистры результата будут перезаписаны новым значением.

Регистры

Регистры данных MDx и результата MRx используют одни и те же адреса, но так как регистры MRx не доступны для записи, то любая операция записи в регистр результата будет интерпретирована как запись в регистр данных. Для чтения регистров необходимо установить или очистить бит RSEL регистра MDUCON, чтобы указать, к каким регистрам идет обращение. По умолчанию считываются регистры результата MRx.

В зависимости от типа операции, регистры MDx и MRx выполняют различные функции. Например, в операции умножения регистры MD1 и MD0 составляют

16-разрядный множитель – старший байт умножаемого 16-разрядного числа записывается в регистр MD1, а младший – в регистр MD0. Для получения подробной информации о функциях регистров данных и результатах при различных операциях следует обратиться к таблицам 7.2 и 7.3.

Примечание – В таблицах 7.2 и 7.3 буква «H», стоящая в конце названия функции, указывает на то, что в регистре располагается старший байт 16-разрядного значения, а буква «L» – младший байт. Цветом выделены регистры, участвующие в 32-разрядных операциях. При этом «H» указывает на то, что в регистре располагается старший байт старшего слова, а «L» – младший байт младшего слова, «h» и «l» указывают на оставшиеся два байта 32-разрядного значения.

Таблица 7.2 – Функции регистров данных MDx

Регистры	Функции регистров в операциях			
	Умножение 16x16	Деление 32/16	Деление 16/16	Нормализация и сдвиг
MD0	Множитель1 L	Делимое L	Делимое L	Операнд L
MD1	Множитель1 H	Делимое l	Делимое H	Операнд l
MD2	–	Делимое h	–	Операнд h
MD3	–	Делимое H	–	Операнд H
MD4	Множитель 2 L	Делитель L	Делитель L	Управление*
MD5	Множитель 2 H	Делитель H	Делитель H	–

_____

* Функционирует как управляющий регистр.

Таблица 7.3 – Функции регистров результата MRx

Регистры	Функции регистров в операциях			
	Умножение 16x16	Деление 32/16	Деление 16/16	Нормализация и сдвиг
MR0	Произведение L	Частное L	Частное L	Результат L
MR1	Произведение l	Частное l	Частное H	Результат l
MR2	Произведение h	Частное h	–	Результат h
MR3	Произведение H	Частное H	–	Результат H
MR4	Множитель 2 L	Остаток L	Остаток L	Управление*
MR5	Множитель 2 H	Остаток H	Остаток H	–

_____

* Функционирует как управляющий регистр.

Операция умножения

Блок MDU поддерживает операцию умножения. Первый 16-разрядный множитель записывается в регистры MD0 (младший байт) и MD1 (старший байт), а второй – MD4 и MD5. Результат – 32-разрядное произведение – размещается в регистрах с MR0 (младший байт младшего слова) по MR3 (старший байт старшего слова). В регистрах MP4 и MR5 размещается второй множитель.

Операция деления

Блок MDU поддерживает операцию укороченного деления, определяемую стандартом ISO C99, и распространенную среди современных процессоров. Деление осуществляется по следующему правилу:

Если есть делимое D, делитель d, частное q и остаток от деления r (с условием $|r| < |d|$), то справедливо $D = q \times d + r$.

Такое деление округляет частное (q) до целого значения, а знак остатка от деления (r) всегда равен знаку делимого (D).

Сдвиг

Сдвиг проводится над 32-разрядной переменной, расположенной в регистрах с MD0 (младший байт младшего слова) по MD3 (старший байт старшего слова).

Во время логического сдвига нули сдвигаются в направлении старшего бита регистра MD3 (сдвиг влево) или младшего бита регистра MD0 (сдвиг вправо). Арифметический сдвиг влево идентичен логическому сдвигу влево, но во время арифметического сдвига вправо разряды со знаком сдвигаются в направлении старшего бита регистра MD3 (т.е. влево). Например, арифметический сдвиг вправо числа 0101b даст результат 0010b, а числа 1010b – результат 1101b.

Установить направление и количество сдвигов можно в регистре MD4.

Примечание – Блок MDU не показывает переполнение при операциях арифметического сдвига влево. Пользователю следует строить вычисления так, чтобы результат арифметического сдвига влево всегда был в пределах от 0 до 32.

Нормализация

Нормализация проводится над целочисленной 32-разрядной переменной без знака, расположенной в регистрах с MD0 (младший байт младшего слова) по MD3 (старший байт старшего слова). Функция нормализации предназначена для поддержки приложений, в которых используются арифметические операции над числами с плавающей запятой. Во время нормализации все нули перед первой значащей цифрой переменной удаляются операциями сдвига влево. Операция заканчивается, когда в самом старшем разряде появляется единица. Битовое поле SCTR регистра MR4 содержит число операций сдвига влево, которые были выполнены в ходе операции нормализации. Это число может использоваться как экспонента. Максимальное число сдвигов в операции нормализации – 31 ($2^5 - 1$).

Флаг занятости

Флаг занятости, роль которого играет бит START регистра MDUCON, показывает текущее состояние занятости блока MDU. После запуска флаг START будет оставаться установленным до окончания операции. По окончании вычислений флаг аппаратно будет сброшен. Запись в любые регистры блока MDU во время работы блока может привести к получению неправильного результата операции.

8 Таймеры-счетчики

В микроконтроллере имеется три 16-разрядных таймера/счетчика: T0, T1 и T2. Они могут быть использованы как в качестве таймеров, так в качестве счетчиков внешних событий.

При работе в качестве таймера, содержимое соответствующего таймера/счетчика инкрементируется в каждом машинном цикле – каждые 12 периодов сигнала XTAL1 (см. рисунок 8.1).

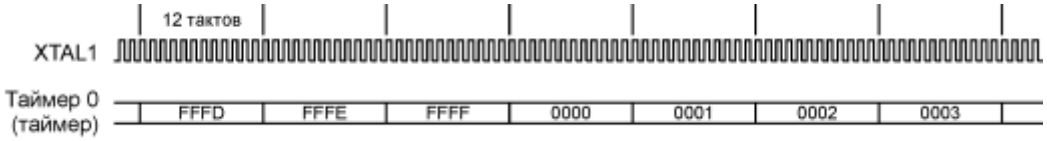


Рисунок 8.1 – Работа таймера/счетчика T0 в режиме таймера

При работе в качестве счетчика, содержимое таймера/счетчика инкрементируется каждый раз, когда на соответствующем входе микроконтроллера (выводы T0, T1) появляется фронт перепада входного сигнала из высокого уровня в низкий. Опрос выводов T0 и T1 происходит каждый машинный цикл, но следует помнить, что максимальная частота переключений сигналов на входах T0 и T1 не должна превышать 1/24 частоты сигнала XTAL1, т.е. минимальный период входного сигнала должен равняться двум машинным циклам. На рисунке 8.2 показан возможный вид входного сигнала на выводе T0.

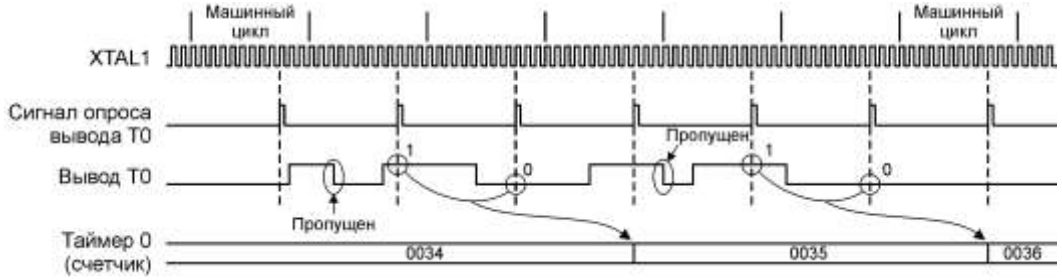


Рисунок 8.2 – Работа таймера/счетчика T0 в режиме счетчика

Инкрементирование таймера/счетчика T0 происходит только в том случае, если в одном машинном цикле был обнаружен высокий уровень входного сигнала, а в следующем – низкий. Инкрементирование счетчика происходит в машинном цикле, следующим за тем, в котором был обнаружен перепад уровня. На рисунке 8.2 символами «1» и «0» обозначены уровни сигнала, которые, следуя друг за другом, вызывают переключение счетчика (кривые стрелки). Перепады сигнала, которые не воспринимаются счетчиком или пропускаются и, как следствие, не влияют на него, подписаны словом «Пропущен».

Таймер/счетчик T2 способен работать и как таймер, и как счетчик событий.

Регистры T0, T1 и T2 образуются конкатенацией соответствующих пар регистров. TH0 и TL0 образуют T0, TH1 и TL1 – T1, TH2 и TL2 – T2.

8.1 Таймеры T0 и T1

Для каждого из таймеров/счетчиков T0 и T1 существует по четыре режима работы. Режимы 0, 1 и 2 таймеров/счетчиков идентичны. Режимы 3 таймеров/счетчиков имеют отличия. Далее, если функционирование таймеров/счетчиков идентично, в их названии, а также в названиях соответствующих им битов управления и флагов, будет использоваться символ «x» вместо «0» и «1».

Режим 0 таймера Tx

В этом режиме таймер/счетчик функционирует как 13-разрядный таймер, на вход которого подключен 5-битный предделитель частоты на 32. При переполнении и переходе из состояния «все единицы» в состояние «все нули» выставляется флаг прерывания таймера TxF в регистре TCON. Тактирование таймера разрешено, если соответствующий управляющий бит TxR установлен, либо управляющий бит TxGATE равен 0, либо на входе внешнего прерывания 1 (вывод INT1# микроконтроллера) поддерживается высокий уровень сигнала. Из этого следует, что при TxGATE = 1b, таймер может использоваться для измерения длительности импульсного сигнала, подаваемого на вывод INT1#.

Режим 1 таймера Tx

В этом режиме таймер/счетчик функционирует как 16-разрядный таймер. В остальном режим 1 аналогичен режиму 0.

Режим 2 таймера Tx

В этом режиме таймер/счетчик функционирует как 8-разрядный таймер с программируемой перезагрузкой. Непосредственно таймером является регистр TLx. Регистр THx хранит значение, которое загружается в таймер после его переполнения.

Таким образом, когда таймер достигает значения FFh, выставляется флаг переполнения TxF, после чего в таймер загружается значение из регистра THx. Содержимое регистра THx может быть изменено только программно, перезагрузка таймера на него не влияет.

Режим 3 таймера T0

В этом режиме регистры TH0 и TL0 функционируют как два независимых таймера, каждый из которых инкрементируется каждые 12 тактов сигнала XTAL1 (т.е. каждый машинный цикл). Работа регистра TL0 управляется битами T0C/T#, T0GATE, T0R регистра TCON, входным сигналом прерывания 0 (вывод INT0# микроконтроллера) и контролируется по флагу T0F. Регистр TH0, в отличие от регистра TL0, может функционировать только как таймер. Запуск и останов таймера управляется битом T1R, работа контролируется флагом T1F.

Режим 3 таймера T1

Таймер/счетчик T1 не функционирует и его содержимое остается без изменений.

8.2 Таймер T2

Таймер/счетчик T2 может работать в трех режимах: автоперезагрузки, захвата, генератора скорости передачи (в бодах).

В режиме таймера регистры TH2 и TL2 образуют 16-разрядный регистр, инкрементируемый в каждом машинном цикле, если это разрешено битом T2R регистра T2CON. Регистры RCAP2H и RCAP2L образуют 16-разрядный регистр захвата/перезагрузки RCAP2 таймера T2.

Работа таймера T2 в режиме автоперезагрузки

Таймер T2 функционирует как 16-разрядный счетчик. Направление счета (инкрементирование или декрементирование) задается битом DCEN регистра T2MOD и состоянием входа T2EX микроконтроллера.

Если бит DCEN = 0b (регистр T2MOD) и бит EXEN2 = 0b (регистр T2CON) таймер T2 только инкрементируется. По достижении значения FFFFh выставляется флаг переполнения T2F и в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Если бит DCEN = 0b и бит EXEN2 = 1b, таймер T2 только инкрементируется. Перезагрузка таймера происходит по достижении значения FFFFh или обнаружении фронта перепада уровня сигнала из высокого в низкий на входе T2EX микроконтроллера. По достижении значения FFFFh выставляется флаг переполнения T2F, а при обнаружении перехода «1» в «0» на входе T2EX выставляется флаг EXF2. В том и другом случае после выставления соответствующего флага в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Если бит DCEN = 1b и вход T2EX находится в состоянии логического нуля, то таймер T2 декрементируется до достижения значения 0000h, после чего выставляется флаг T2F и в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Если бит DCEN = 1b и вход T2EX находится в состоянии логической единицы, то таймер T2 инкрементируется до достижения значения FFFFh, после чего выставляется флаг T2F и в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Флаги T2F и EXA2 могут генерировать прерывания таймера T2.

При перезагрузке содержимое регистра RCAP2H помещается в регистр TH2, а содержимое регистра RCAP2L – в регистр TL2.

Работа таймера/счетчика T2 в режиме захвата

Если EXEN2 = 0b (регистр T2CON) таймер/счетчик T2 функционирует как 16-разрядный таймер или счетчик (в зависимости от состояния бита C/T2# регистра T2CON), при переполнении которого выставляется флаг T2F.

Если EXEN2 = 1b таймер/счетчик T2 функционирует как 16-разрядный таймер или счетчик, при переполнении которого выставляется флаг T2F. Отличием от предыдущего режима является то, что при появлении перепада из «1» в «0» на входе T2EX микроконтроллера, содержимое регистра T2 захватывается и помещается в регистр RCAP2 (т.е. TH2 в RCAP2H, а TL2 в RCAP2L) и выставляется флаг EXF2.

Работа счетчика T2

При работе в качестве счетчика содержимое таймера/счетчика инкрементируется каждый раз, когда на соответствующем входе микроконтроллера (вывод T2) появляется фронт перепада входного сигнала из высокого уровня в низкий. Опрос вывода T2 происходит каждый машинный цикл, но следует помнить, что максимальная частота переключений сигнала на входе T2 не должна превышать 1/24 частоты сигнала XTAL1, т.е. минимальный период входного сигнала должен равняться двум машинным циклам. Работа счетчика T2 в таком случае аналогична работе счетчика T0. Возможный вид входного сигнала для T2 будет аналогичен изображенному на рисунке 8.2.

Работа счетчика T2 в режиме автоперезагрузки

Таймер T2 функционирует как 16-разрядный счетчик. Направление счета (инкрементирование или декрементирование) задается битом DCEN регистра T2MOD и состоянием входа T2EX микроконтроллера.

Если бит DCEN = 0b (регистр T2MOD) и бит EXEN2 = 0b (регистр T2CON), таймер T2 только инкрементируется. По достижении значения FFFFh выставляется флаг переполнения T2F и в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Если бит DCEN = 0b и бит EXEN2 = 1b, таймер T2 только инкрементируется. Перезагрузка таймера происходит по достижении значения FFFFh или обнаружении фронта перепада уровня сигнала из высокого в низкий на входе T2EX микроконтроллера. По достижении значения FFFFh выставляется флаг переполнения T2F, а при обнаружении перехода «1» в «0» на входе T2EX выставляется флаг EXF2. В том и другом случае после выставления соответствующего флага в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Если бит DCEN = 1b и вход T2EX находится в состоянии логического нуля, то таймер T2 декрементируется до достижения значения, хранящегося в регистре RCAP2, после чего выставляется флаг T2F и в регистр T2 таймера загружается значение FFFFh.

Если бит DCEN = 1b и вход T2EX находится в состоянии логической единицы, то таймер T2 инкрементируется до достижения значения FFFFh, после чего выставляется флаг T2F и в регистр T2 таймера загружается 16-разрядное значение из регистра RCAP2.

Бит EXF2 переключается при переполнении или опустошении таймера. Этот бит может использоваться как бит 17 таймера-счетчика. В этом режиме EXF2 не является флагом прерывания. Только флаг T2F может генерировать прерывания таймера T2.

При перезагрузке содержимое регистра RCAP2H помещается в регистр TH2, а содержимое регистра RCAP2L – в регистр TL2.

Работа таймера/счетчика T2 в режиме тактирования передачи данных, осуществляемая приемопередатчиками типа UART

Режим включается установкой хотя бы одного из двух битов TCLK и RCLK регистра T2CON. В этом режиме таймер 2 функционирует как генератор тактового сигнала для приемопередатчиков типа UART (подробнее в разделе 11). Скорости передачи и приема информации могут отличаться. Это возможно, если использовать разные таймеры для тактирования передачи и приема.

При работе приемопередатчика в режиме 1 или 3 передача данных тактируется сигналом переполнения таймера 2 (см. рисунок 8.3).

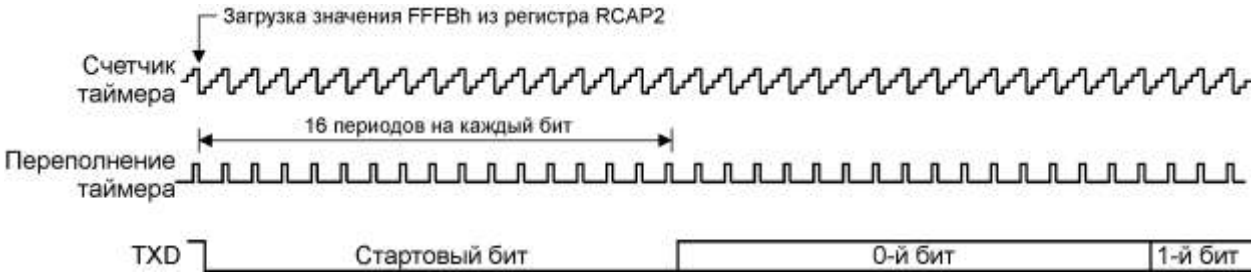


Рисунок 8.3 – Тактирование передачи

Скорость счета таймера зависит от входной частоты микроконтроллера на входе XTAL1 и состояния бита X2 регистра CLKREG (см. рисунок 6.2), который управляет делением входной частоты на два. После сброса делитель частоты на два включен (по умолчанию).

Переполнение регистра таймера формирует сигнал переполнения и инициирует загрузку в регистр таймера значения из регистра RCAP2, с которого таймер продолжает счет. На рисунке 8.3 загружаемое значение – FFFBh.

От начала стартового бита (появление отрицательного перепада уровня сигнала на выводе TXD – начало передачи) отсчитываются 16 периодов перезагрузки таймера 2, после чего начинается передача следующего бита.

Таким образом, время передачи одного бита можно вычислить так:

Время передачи бита = [Время от перезагрузки до переполнения таймера T2]×16.

С учетом всех вышеперечисленных особенностей скорость передачи в бодах определяется выражением:

$$\text{Bod} = \frac{\text{FXTAL1} \times 10^6}{16 \times [65535 - \text{RCAP2d}]} \times \frac{1}{2},$$

где Bod – скорость передачи информации, в бодах,

FXTAL1 – внешняя тактовая частота, МГц,

RCAP2d – значение, записанное в регистре RCAP2, представленное в десятичном формате.

Множитель 1/2 используется при включенном делителе входной частоты. В случае, если программно был установлен бит X2 регистра CLKREG, при расчете скорости множитель не используется.

Для примера рассчитаем скорость передачи при FXTAL1 = 10 МГц, RCAP2 = FFDFh (в десятичном формате – 65503) и включенном делителе частоты.

$$\text{Bod} = \frac{10 \times 10^6}{16 \times [65535 - 65503]} \times \frac{1}{2} = 9766.$$

Рассчитанная скорость передачи информации получилась равной 9766 бод, что примерно соответствует стандартной скорости 9600 бод.

Для расчета значения, которое следует записать в регистр RCAP2 при известной входной частоте и скорости передачи, следует воспользоваться формулой

$$\text{RCAP2d} = 65535 - \left[\frac{\text{FXTAL1} \times 10^6}{16 \times \text{Bod}}\right] \times \frac{1}{2},$$

в которой множитель 1/2 используется при включенном делителе входной частоты.

Для примера рассчитаем значение, которое следует записать в регистр RCAP2, чтобы получить скорость передачи 9600 бод при FXTAL1 = 10 МГц, и включенном делителе частоты.

$$\text{RCAP2d} = 65535 - \left[\frac{10 \times 10^6}{16 \times 9600}\right] \times \frac{1}{2} = 65502.$$

Получившееся значение 65502 после перевода в шестнадцатеричное значение будет равно FFDEh.

Перечень значений регистра RCAP2 для некоторых основных значений скорости передачи представлен в таблице 8.1.

Таблица 8.1 – Значение регистра RCAP2 для различных скоростей и внешних частот

Скорость передачи в бодах	Значения регистра RCAP2 для разных внешних тактовых частот	
	10 МГц	20 МГц
3200	FF9Eh	FF3Dh
6400	FFBFh	FF9Eh
9600	FFDFh	FFBFh
19200	FFECh	FFDFh
32800	FFF6h	FFECh
65600	FFFBh	FFF6h

В режиме синхронизации переполнение регистра T2 не выставляет флаг T2F (чтобы не вызвать прерывание). Еще одной особенностью является то, что если EXEN2 = 1b, то перепад из «1» в «0» на выводе T2EX устанавливает флаг EXF2, но не вызывает перезагрузки таймера значением из RCAP2, а значит вход EXF2 может использоваться как вход дополнительного внешнего прерывания.

В режиме синхронизации таймер/счетчик T2 может работать как в режиме таймера, так и в режиме счетчика, но как правило предпочтение отдается режиму таймера.

Таймер T2 в режиме генератора скорости передачи в бодах

В этом режиме таймер T2 инкрементируется в два раза быстрее, т.е. период переключения составляет половину машинного цикла – 6 тактов сигнала XTAL1 (в отличие от других режимов работы, где частота переключения равна 12 тактам XTAL1)

Переполнение таймера T2 устанавливает флаг T2F, после чего в таймер загружается значение из регистра RCAP2.

Запись в регистры TH2 и TL2 таймера T2 и чтение этих регистров во время работы таймера нежелательны, поскольку могут приводить как к некорректной работе счетчика таймера, так и к получению неточных данных.

Запись в регистры RCAP2H и RCAP2L во время работы таймера нежелательна, поскольку может приводить к некорректной перезагрузке счетчика таймера. Чтение регистров RCAP2H и RCAP2L разрешено всегда.

В связи с указанными выше особенностями работы таймера T2 в режиме генератора скорости передачи в бодах, запись и чтение регистров таймера TH2 и TL2, а также запись в регистры RCAP2H и RCAP2L следует производить только при остановленном таймере.

Режим генератора программируемого меандра

Вывод P1.0 микроконтроллера может быть запрограммирован как вывод T2FREQ, являющийся выходом генератора меандра (меандр – сигнал, представляющий собой последовательность прямоугольных импульсов со скважностью 2). При тактовой частоте 16 МГц таймер T2 может выполнять роль генератора меандра и формировать сигнал с частотой от 61 Гц до 4 МГц.

Переполнение таймера T2 вызывает загрузку в регистр таймера T2 значения из регистра RCAP2. Флаг T2F при этом не устанавливается (чтобы не вызвать прерывание).

Для включения режима следует записать «0» в бит C/T2# регистра T2CON и «1» – в бит T2OE регистра T2MOD. Запуском таймера управляет бит T2R.

Частота меандра FMDR может быть рассчитана по формуле:

$$\mathrm{FMDR} = \frac{\mathrm{FXTAL1}}{4 \times [\,65536 - \mathrm{RCAP2d}\,]},$$

где RCAP2d – значение, хранящееся в регистре RCAP2 в десятичном формате.

Перезагрузка таймера T2 без вызова прерывания делает режим генератора меандра схожим с режимом генератора скорости передачи в бодах. Отсюда следует, что таймер T2 может одновременно использоваться и как генератор скорости передачи в бодах, и как генератор меандра. Однако, следует помнить, что частоты передачи в бодах и частоты меандров нельзя определить независимо друг от друга.

9 Массив программируемых счетчиков

Наличие массива программируемых счетчиков (далее – блока PCA) является одной из особенностей микроконтроллера. Блок PCA – это блок ввода-вывода, предназначенный для выполнения различных операций счета и определения временных интервалов, в том числе при широтно-импульсной модуляции (ШИМ).

Блок PCA состоит из 16-разрядного таймера/счетчика TC и пяти модулей захвата/сравнения, в составе каждого из которых имеется 16-разряднй регистр. Каждый модуль имеет канал ввода/вывода.

Таймер/счетчик TC может функционировать как таймер, так и счетчик событий. 16-разрядный регистр TC таймера/счетчика TC состоит из двух регистров CH (старший байт) и CL (младший байт). Физически нельзя обратиться к регистру TC – это название введено только для удобства описания и подразумевает под собой пару регистров – CH и CL.

Управление работой и режимами таймера/счетчика TC осуществляется при помощи регистров CMOD и CCON. Режимы работы модулей захвата/сравнения определяются пятью регистрами – CCAPM0, CCAPM1, CCAPM2, CCAPM3, CCAPM4.

Выводы порта 1 со второго по седьмой являются входами/выходами (каналами) блока PCA для работы с внешними периферийными устройствами. Для конфигурирования выводов порта 1 как выводов блока PCA следует установить соответствующие биты в регистре P1SEL. После правильного конфигурирования выводы порта 1 будут выполнять функции, приведенные в таблице 9.1.

Таблица 9.1 – Функции выводов порта 1 при работе с блоком PCA

Номер вывода	Название вывода	Функция
P1.7	CEX4	Вход при захвате, выход при сравнении и ШИМ канала 4
P1.6	CEX3	Вход при захвате, выход при сравнении и ШИМ канала 3
P1.5	CEX2	Вход при захвате, выход при сравнении и ШИМ канала 2
P1.4	CEX1	Вход при захвате, выход при сравнении и ШИМ канала 1
P1.3	CEX0	Вход при захвате, выход при сравнении и ШИМ канала 0
P1.2	ECI	Вход таймера/счетчика TC

Блок PCA имеет лишь один вектор прерывания. Если бит ECCF (регистра CMOD) разрешения прерывания установлен, то выставление флага CF в регистре CCON сформирует запрос прерывания блока PCA.

Таймер/счетчик TC

Пара регистров CH и CL функционирует как 16-разрядный таймер/счетчик TC, тактируемый сигналом от одного из четырех источников (см. рисунок 9.1). Регистр CL (младший байт регистра TC) инкрементируется до переполнения, после чего в него загружается значение 00h. Через два периода (тактового сигнала, подаваемого на TC) после переполнения регистра CL инкрементируется регистр CH (старший байт регистра TC). После переполнения регистра CH выставляется флаг CF в регистре CCON и, если это разрешено, формируется запрос прерывания.

Источниками сигнала тактирования таймера/счетчика TC являются:

- сигнал FOSC (с частотой fosc, равной частоте сигнала XTAL1);
- сигнал переполнения таймера 0;
- внешний сигнал на выводе ECI микроконтроллера.

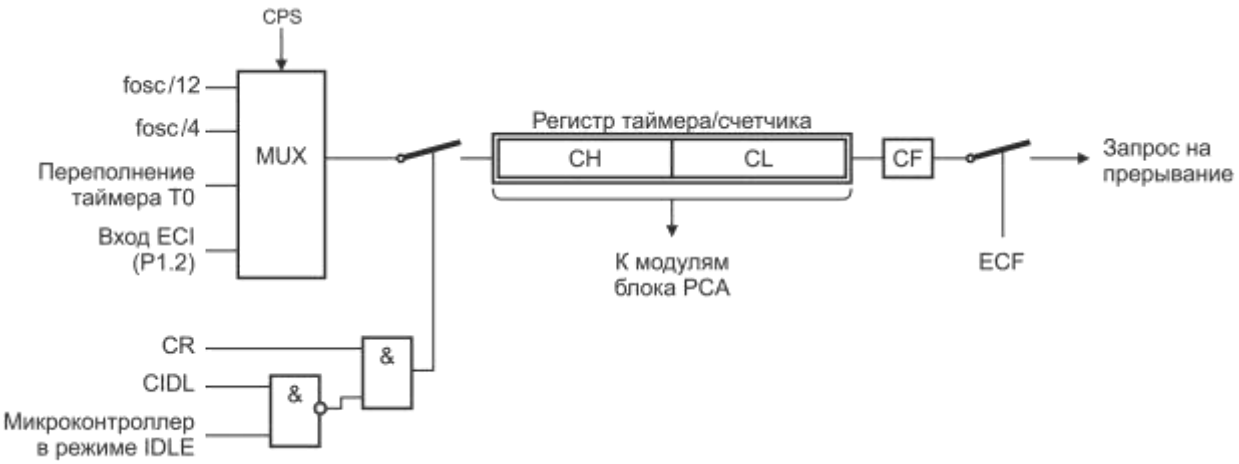


Рисунок 9.1 – Таймер/счетчик TC блока PCA и источники его сигнала тактирования

Сигнал XTAL1, используемый для тактирования таймера/счетчика TC, предварительно проходит через делитель частоты. Поэтому период синхросигнала, поступающего на таймер/счетчик TC, может быть равен машинному циклу или 1/3 машинного цикла.

Частоту сигнала тактирования таймера/счетчика TC можно программировать, если в качестве источника синхросигнала выступает таймер 0 со своим сигналом переполнения.

Если в качестве источника синхросигнала выбран вывод ECI микроконтроллера, то таймер/счетчик TC будет инкрементироваться каждый раз, когда на выводе будет обнаруживаться перепад сигнала из «1» в «0». За один машинный цикл вывод ECI опрашивается три раза, поэтому минимальная частота входного сигнала не должна превышать 1/8 сигнала XTAL1 и, соответственно, длительность входного импульса не должна быть меньше четырех периодов сигнала XTAL1.

За выбор того или иного источника синхронизации отвечает битовое поле CPS регистра CMOD.

Запуском и остановом таймера/счетчика TC управляет бит CR (регистр CCON). Бит CIDL управляет функционированием таймера/счетчика в режиме Idle.

Содержимое регистров CH и CL таймера/счетчика TC всегда может быть прочитано. Запись в эти регистры во время работы таймера/счетчика запрещена.

Модули захвата/сравнения блока PCA

В блоке PCA имеется пять модулей захвата/сравнения. Каждый модуль имеет пару регистров захвата/сравнения, компаратор, логику управления и схемы выбора сигналов.

Далее в настоящем разделе будет дано описание одного модуля захвата/сравнения, но поскольку работа модулей идентична, это описание распространяется и на остальные модули. При упоминании регистров модулей, битов управления и флагов вместо порядковых номеров (0, 1, 2, 3 и 4) будет использоваться символ «x».

В состав модуля входит 16-разрядный регистр захвата/сравнения (с условным названием CCAPx) канала x, образуемый регистрами CCAPxH (старший байт) и CCAPxL (младший байт) (см. таблицу 9.6). Регистр CCAPx может функционировать в трех режимах:

- захвата, когда в нем сохраняется время или значение счетчика, при котором произошло внешнее событие

- сравнения, когда в нем сохраняется время или значение счетчика, при котором должно произойти установленное действие;

- ШИМ, когда младший байт регистра управляет шириной выходного сигнала.

В свою очередь, каждый модуль блока PCA может быть сконфигурирован необходимым образом и работать в одном из трех режимов.

Режимы модуля блока PCA:

- захват 16-разрядного значения таймера/счетчика TC, включая три подрежима:

- по положительному фронту (перепад из «0» в «1») на входе CEXx;
- по отрицательному фронту (перепад из «1» в «0») на входе CEXx;
- по обоим фронтам сигнала на входе CEXx;
- сравнение содержимого регистра CCAPx со значением регистра таймера/счетчика TC, включая четыре подрежима:
- 16-разрядного программируемого таймера;
- 16-разрядного скоростного вывода;
- сторожевого таймера (только для модуля 4);
- 8-разрядного формирователя ШИМ.

Для конфигурирования модуля используются комбинации битов в регистре CCAPMx. Возможные комбинации приведены в таблицах 9.7 и 9.8.

Для работы модулей захвата/сравнения необходима работа таймера/счетчика TC.

Для запрещения работы (отключения) любого модуля его следует перевести в режим «Модуль отключен», записав в регистр CCON значение 00h. При наличии события (захват, срабатывание таймера, скоростной вывод) устанавливается флаг CCFx в регистре CCON и если разрешено, формируется запрос на прерывания блока PCA.

Регистры CCAPxH и CCAPxL всегда доступны для записи и чтения.

Режимы захвата

Режимы захвата дают возможность измерять длительность импульсов, циклов, разность фаз сигналов на соответствующем входе блока PCA. В зависимости от выбранного режима, вход CEXx опрашивается для обнаружения положительного и/или отрицательного перепада сигнала. Как только перепад сигнала на входе CEXx совпадает с ожидаемым, происходит захват значения таймера/счетчика TC и помещение его в регистр CCAPx, после чего выставляется флаг CCFx в регистр CCON. Далее, если это разрешено, формируется запрос прерывания модуля PCA. В целом, схема функционирования модуля x в режимах захвата соответствует представленной на рисунке 9.2.

Примечание – Флаг CCFx не очищается аппаратно, а только программно. Значение регистра CCAPx перезаписывается каждый раз, когда возникает запрограммированное событие (появляется ожидаемый фронт).

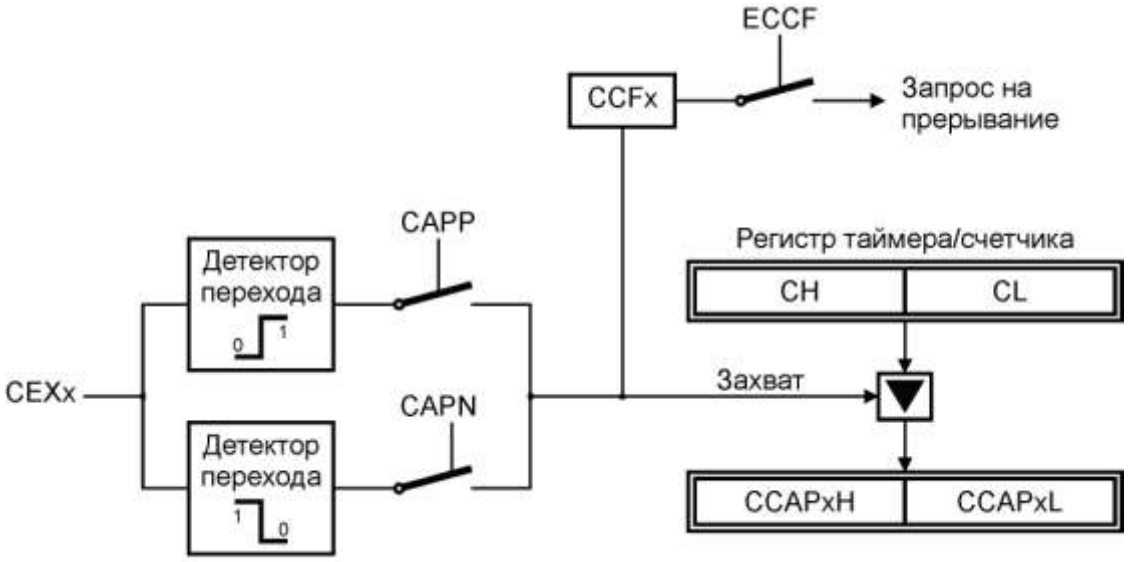


Рисунок 9.2 – Схема функционирования модуль x для режимов захвата

Следует помнить, что схема модуля может отслеживать импульсный сигнал, частота которого не превышает частоту тактирования таймера/счетчика TC. В случае нарушения

этого условия, некоторые (или большинство) фронты перепадов уровня входного сигнала не будут обнаружены модулем.

Режимы сравнения блока PCA

Функция сравнения обеспечивает четыре режима:

- 16-разрядного таймера;
- скоростного вывода;
- сторожевого таймера;
- ШИМ.

В первых трех режимах модуль постоянно сравнивает содержимое таймера/счетчика со значением, загруженным предварительно в его регистр CCAPx. В режиме ШИМ модуль постоянно сравнивает содержимое регистра CL таймера/счетчика TC со значением регистра CCAPxL. Сравнение производится три раза за цикл обмена, т. е. с наибольшей возможной частотой (fosc/4).

Функция сравнения для модуля выбирается установкой бита ECOM в регистре CCAPMx. Последовательность работы в режиме сравнения:

- выбирается режим работы модуля;
- выбирается входной сигнал для таймера/счетчика;
- загружается значение в регистр CCAPx;
- устанавливается бит управления запуском таймера/счетчика TC;
- после прерывания программно очищается флаг события CCFx.

Режим программируемого 16-разрядного таймера

В этом режиме сравнивается текущее значение таймера/счетчика TC со значением регистра CCAPx, который был загружен заранее (см. рисунок 9.3). При совпадении устанавливается флаг события CCFx в регистре CCON. Флаг следует сбрасывать программно при обработке прерывания. Также, при обслуживании прерывания можно загрузить новое значение в регистр CCAPx.

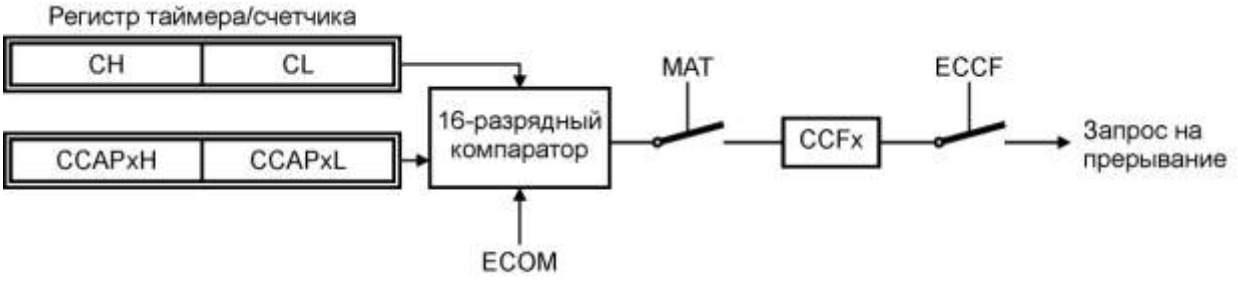


Рисунок 9.3 – Блок PCA в режиме 16-разрядного таймера

Примечание – При записи в регистр CCAPxL очищается бит ECOM, что запрещает выполнять функцию сравнения. При записи в регистр CCAPxH бит ECOM устанавливается, что разрешает функцию сравнения. Этот механизм является защитой от ложных срабатываний. Поэтому в процессе загрузки регистров рекомендуется вначале записывать данные в регистр CCAPxL, а затем в регистр CCAPxH.

Режим скоростного вывода

Рисунок 9.4 – Блок PCA в режиме скоростного вывода

В режиме скоростного вывода совпадение значений таймера/счетчика TC и значения регистра CCAPx вызывает смену уровня сигнала на выходе CEXx и установку флага CCFx (см. рисунок 9.4). Это обеспечивает более высокую точность, чем при программном переключении сигнала, т. к. переключение осуществляется до обслуживания запроса прерывания. Таким образом, интервал времени, связанный с обслуживанием прерывания, не вклинивается в диаграмму формирования выходного сигнала. Программно задавая уровень сигнала на выходе CEXx, возможно задать тип перепада уровня сигнала – из низкого в высокий или наоборот. Флаг CCFx при обработке прерывания следует сбрасывать программно. Если в процессе обслуживания прерывания не было записано новое значение в регистр CCAPx, то следующее совпадение произойдет через полный цикл таймера/счетчика TC.

Режим сторожевого таймера (только для модуля 4)

В режиме сторожевого таймера при совпадении значения таймера/счетчика TC со значением регистра CCAP4 (т.е. по истечении установленного времени), осуществляется сброс и инициализация микроконтроллера (см. рисунок 9.5), что является стандартным приемом выхода из зависаний программы управления.

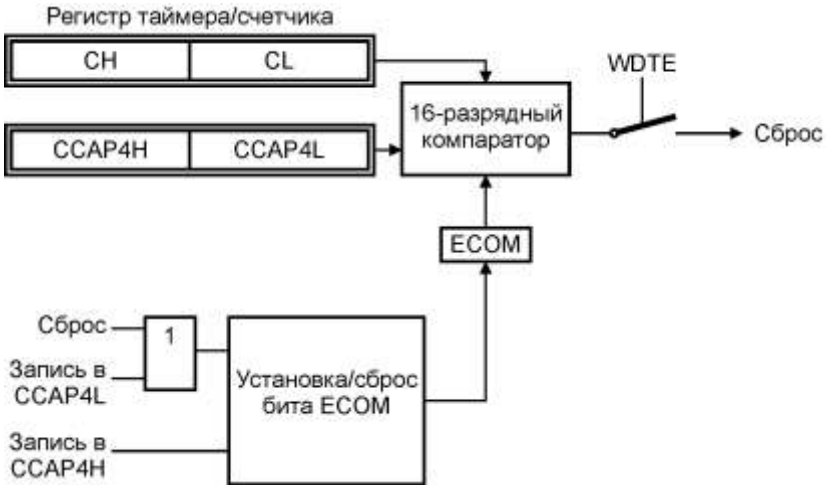


Рисунок 9.5 – Модуль 4 блока PCA в режиме сторожевого таймера

Чтобы перевести модуль 4 в режим сторожевого таймера необходимо:

- установить биты ECOM4 и MAT4 регистра CCAPM4;
- установить бит WDTE регистра CMOD;
- выбрать нужный вход для таймера/счетчика CT с помощью битового поля CPS регистра CMOD;

- загрузить 16-разрядное значение для сравнения в регистр CCAP4 и 16-разрядное начальное значение в таймер/счетчик ТС (можно использовать значение 0000h, определенное по сбросу). Разность между этими числами, умноженная на частоту входных импульсов PCA, определяет интервал времени, на который взведен сторожевой таймер.

Предотвратить сброс контроллера от сторожевого таймера блока PCA можно следующими способами:

1) периодически менять сравниваемое значение в регистре CCAP4, чтобы избежать совпадения;

2) периодически менять значение в таймере/счетчике ТС, чтобы избежать совпадения;

3) очищать бит WDTE до момента совпадения, а затем вновь устанавливать его.

Примечание – Вариант 2) не рекомендуется применять, если работают другие модули блока PCA, поскольку все они взаимодействуют с таймером/счетчиком ТС.

Режим широтно-импульсной модуляции

Все пять модулей блока PCA могут быть запрограммированы на режим широтно-импульсной модуляции ШИМ (рисунок 9.6). При этом на выходах CEXx выдаются модулированные сигналы, ширина импульсов которых определяется 8-разрядным разрешением. Это позволяет преобразовывать цифровой код в аналоговый сигнал при помощи простой внешней схемы (например, интегрирующей цепочки).

В режиме ШИМ младший байт (регистр CL) таймера/счетчика ТС постоянно сравнивается с содержимым регистра CCAPxL. Если значение в CL меньше значения в CCAPxL, то на выходе CEXx удерживается низкий уровень сигнала. При совпадении значений уровень сигнала на выходе становится высоким и удерживается до переполнения регистра CL таймера/счетчика ТС. После обнуления регистра CL уровень сигнала на выходе CEXx становится низким, а в регистр CCAPxL загружается значение из регистра CCAPxH и начинается новый цикл счета.

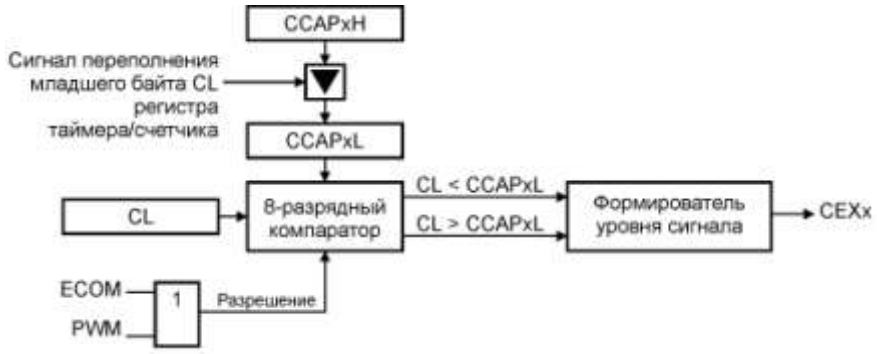


Рисунок 9.6 – Блок PCA в режиме широтно-импульсного модулятора

Число в регистре CCAPxL определяет ширину импульса в текущем цикле, а число в регистре CCAPxH определяет ширину импульса в следующем цикле. При CCAPxL = 00h (0) ширина импульса составляет 100 %, а при CCAPxL = E1h (225) она равна 0,4 %. Частота сигнала на выходе модуля CCXx равна частоте сигнала на входе таймера/счетчика ТС, деленной на 256. Самая высокая частота формируется при входной частоте, равной fosc/4. Так, если входной сигнал имеет частоту fosc = 12 МГц, то частота сигнала на выходе будет 11,7 кГц.

Для перевода модуля x в режим ШИМ нужно:

- установить биты ECOM и PWM регистра CCAPMx;
- определить вход посредством комбинации в битовом поле CPS регистра CMOD;
- загрузить 8-разрядные числа в регистры CCAPxL и CCAPxH;
- установить бит CR управления запуском таймера/счетчика в регистре CCON.

10 Сторожевой таймер

В состав микроконтроллера входит аппаратный сторожевой таймер (WDT), основное назначение которого – сброс микроконтроллера в случае «зависания» выполняемой программы.

Если работа сторожевого таймера разрешена, то он инкрементируется с частотой, заданной сигналом тактирования FOSC. Переполнение таймера вызывает аппаратный сброс микроконтроллера. Период работы сторожевого таймера определяется значением, хранящимся в битовом поле PS регистра WDTCON и может составлять от 16 до 2048 мс. Сброс WDT происходит при общем сбросе микроконтроллера, а также при переходе в режим пониженной мощности PowerDown.

Во избежание срабатывания сторожевого таймера в программную последовательность команд вставляются строки, периодически обнуляющие счетчик сторожевого таймера. В таком случае WDT сработает только в при «зависания» программы.

11 Приемопередатчики UART0 и UART1

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

Условные названия приемопередатчиков – UART0 и UART1. Далее символы «0» и «1», указывающие на принадлежность регистров, выводов и др., к тому или иному приемопередатчику будут заменены символом «x», что будет указывать на то, что написанное применимо как к UART0, так и к UART1 вследствие их полной идентичности.

В состав приемопередатчика входят принимающий и передающий сдвиговые регистры (недоступные программно), а также специальный буферный регистр SBUFx для записи данных для передачи и чтения полученных данных.

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

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

Функция автоматического распознавания адреса позволяет на аппаратном уровне распознать адреса в последовательном битовом потоке.

Последовательный порт UARTx может работать в четырех различных режимах.

Режим 0 (посылка 8 бит)

В этом режиме 8 бит данных передаются (младшим битом вперед) и принимаются через внешний вывод RxDx. Через внешний вывод TxDx выдаются синхросигналы, которые тактируют передачу/прием каждого бита. Частота синхросигнала передачи фиксированная и равна fosc/12. На рисунке 11.1 показана временная диаграмма передачи байта.

После записи байта данных в регистр SBUFx на аппаратном уровне происходит перезапись этого байта в передающий сдвиговый регистр, запись «1» в бит 9 сдвигового регистра и запуск блока управления передачей. Длительность времени от момента записи в регистр SBUFx до начала передачи составляет один машинный цикл. С каждым тактом синхросигнала передачи содержимое сдвигового регистра сдвигается вправо (в сторону младших битов) и поступает на вывод RxDx. В освобождающиеся биты передающего сдвигового регистра записываются нули. По окончании передачи устанавливается флаг TI в регистре SCONx.

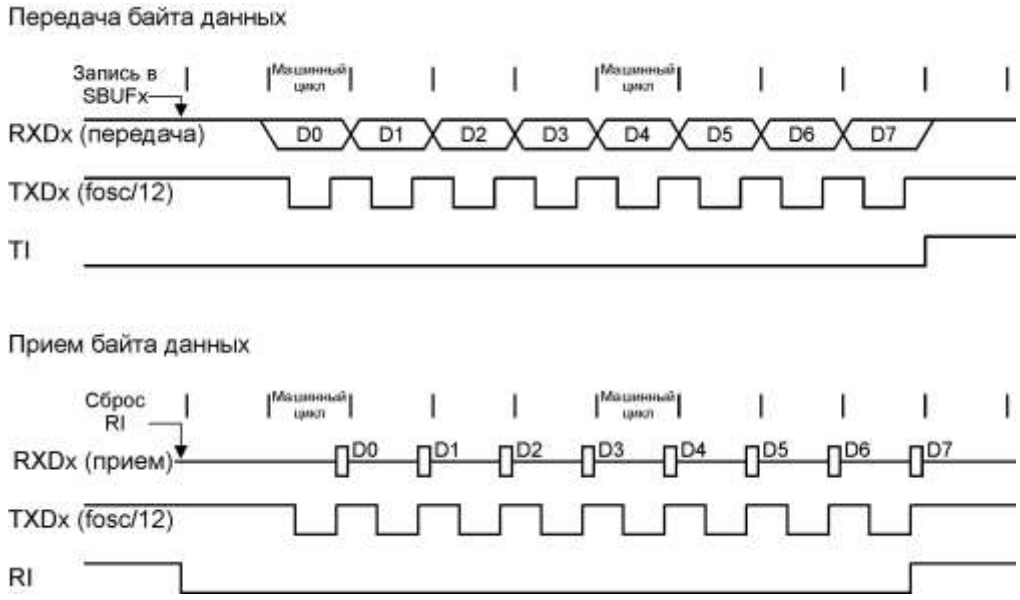


Рисунок 11.1 – Временная диаграмма передачи байта в режиме 0

Прием данных инициализируется установкой бита REN разрешения приема. При этом флаг RI окончания приема в регистре SCONx должен быть сброшен. Прием осуществляется через вывод RxDx по сигналу тактирования, который приходит на вывод TxDx.

Если прием данных инициализирован, а флаг RI не сброшен, то приемник начнет принимать (считывать) данные с вывода RxDx только после сброса флага RI и по истечении одного машинного цикла сигнала тактирования (см. рисунок 11.1).

По окончании приема выставляется флаг RI, и принятые данные переносятся из сдвигового регистра приема в регистр SBUFx и могут быть прочитаны программно.

Режим 1 (посылка 10 бит)

В этом режиме данные передаются через вывод TxDx или принимаются с вывода RxDx. Одна посылка состоит из 10 бит информации: старт-бита, 8 бит данных и стоп-бита. При приеме стоп-бит записывается в бит RB8 регистра SCONx. Временная диаграмма передачи показана на рисунке 11.2.

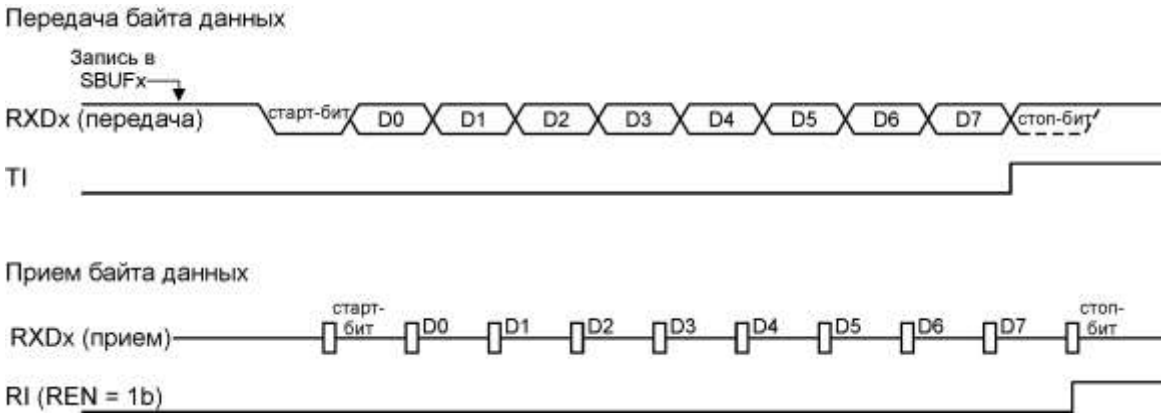


Рисунок 11.2 – Временная диаграмма передачи байта в режиме 1

После записи байта данных в регистр SBUFx, на аппаратном уровне происходит перезапись этого байта в передающий сдвиговый регистр, запись «1» в бит 9 сдвигового регистра и запуск блока управления передачей. Фактически, передача начнется в следующем машинном цикле в момент переключения внутреннего делителя входной

частоты на 16. С каждым тактом содержимое сдвигового регистра сдвигается вправо (в сторону младших битов) и поступает на вывод RxDx. В освобождающиеся биты передающего сдвигового регистра записываются нули.

В режиме 1 тактовый сигнал не передается, поэтому для согласованной работы устройств (соединенных интерфейсом UART), они должны быть настроены на одну скорость передачи. Скорость передачи может быть задана битами RCLK и TCLK регистра T2CON.

По тактовым сигналам с выхода делителя на 16 на вывод TxDx сначала подается старт-бит, а затем биты данных и далее – стоп-бит. По окончании передачи устанавливается флаг TI в регистре SCONx.

Прием начинается при обнаружении перепада уровня сигнала из высокого в низкий на входе RxDx. Для этого под управлением внутреннего счетчика вход RxDx опрашивается 16 раз за период удержания бита (задается заранее). Как только перепад уровня на входе RxDx обнаруживается, внутренний счетчик сбрасывается и перезапускается для выравнивания со счетчиком передающего устройства для синхронизации приема-передачи бит.

Как было сказано, внутренний счетчик переключается 16 раз за время удержания одного бита на линии RxDx. Каждое седьмое, восьмое и девятое переключения счетчика сопровождается опросом вывода RxDx. Таким образом, производится три замера уровня сигнала и ,согласно принципу «два из трех», достоверным считается значение, которое было получено по меньшей мере в двух из трех замеров. Этот механизм обеспечивает подавление ложных (сбойных) бит. Особенно это важно при отслеживании старт-бита, поскольку неправильное его обнаружение приведет к рассинхронизации устройств и, как следствие, некорректному приему данных.

По окончании приема выставляется флаг RI.

Режим 2 (посылка 11 бит)

В этом режиме данные передаются через вывод TxDx или принимаются с вывода RxDx. Одна посылка состоит из 11 бит информации: старт-бита, 8 бит данных, программируемого бита 9 данных и стоп-бита. При передаче бит 9 данных (бит TB8 в регистре SCONx) может быть задан как «0» или «1» или же являться битом контроля четности (бит паритета P регистра PSW, скопированный в TB8). При приеме бит 9 данных записывается в RB8 регистра SCONx, а стоп-бит игнорируется. Скорость передачи программируется и может быть равна fosc/32 или fosc/64. Временная диаграмма передачи показана на рисунке 11.3.

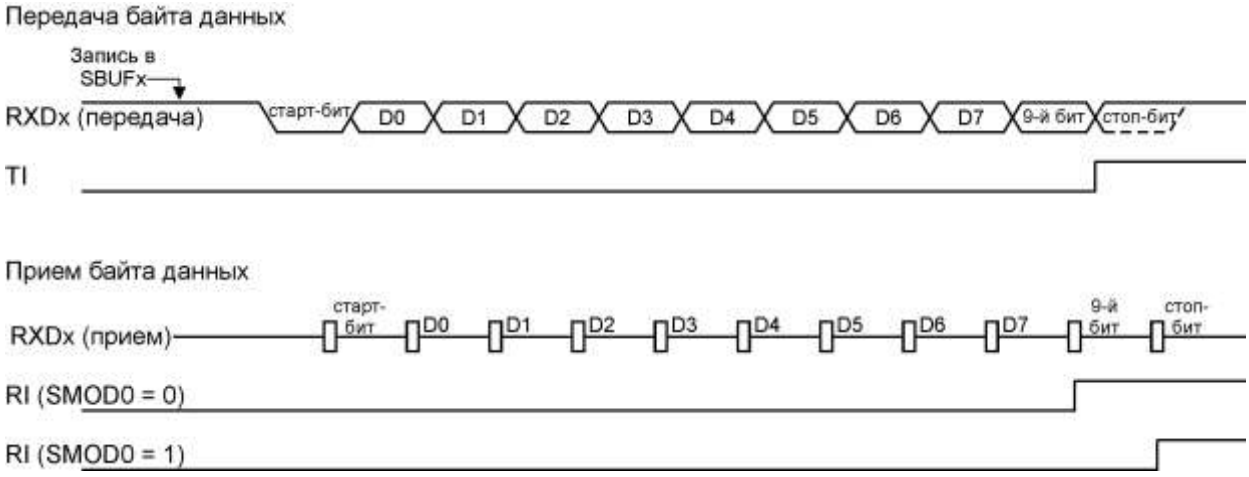


Рисунок 11.3 – Временная диаграмма передачи байта в режиме 2

Режим 3 (посылка 11 бит)

Режим полностью идентичен режиму 2, за исключением того, что скорость приема/передачи может программироваться также, как в режиме 1, т.е. может быть задана битами RCLK и TCLK регистра T2CON.

Примечание – Во всех четырех режимах передача запускается командой записи данных в регистр SBUFx. Прием начинается при обнаружении перепада уровня сигнала из верхнего в нижний если:

- в режиме 0 бит REN = 1b и бит RI = 0b;
- в режимах 1, 2, 3 бит REN = 1b.

Регистр управления и статуса SCONx

Управление режимом работы блока UARTx осуществляется посредством регистра SCONx.

Скорость приема/передачи

Скорость приема/передачи, другими словами, частота работы приемопередатчика может быть как постоянной, так и переменной, в зависимости от режима работы. В режиме 0 частота зависит только от входной тактовой частоты блока UARTx и равняется fosc/12. За один машинный цикл последовательный порт передает/принимает один бит информации. В режимах 1, 2 и 3 частота зависит от состояния бита SMOD1 регистра PCON и битов RCLK и TCLK регистра T2CON.

Дополнительные функции приемопередатчиков

В приемопередатчиках UART0 и UART1, входящих в состав микроконтроллера, помимо их классических функций, дополнительно реализованы еще две:

- обнаружение ошибок кадрирования;
- автоматическое распознавание адреса.

Обнаружение ошибок кадрирования

Каждый блок UARTx имеет устройство детектирования ошибок кадрирования. Это устройство позволяет проверять правильность стоп-битов в режимах 1, 2 и 3. Некорректный стоп-бит может возникать в результате шума на линии передачи или из-за одновременной передачи информации двумя разными устройствами.

Для включения устройства детектирования ошибок кадрирования следует установить бит SMOD0 в регистре PCON.

Если в конце передачи не обнаруживается корректный стоп-бит, устанавливается флаг FE в регистре SCONx. На рисунке 11.4 показано, когда в результате срабатывания детектора ошибок кадрирования выставляется флаг FE (в режимах 1, 2 и 3). Для обнаружения ошибок этот флаг может быть проверен программно после каждого приема (об окончании приема свидетельствует флаг RI). Флаг FE не сбрасывается аппаратно. Будучи однажды установленным, бит FE может быть сброшен только программно.

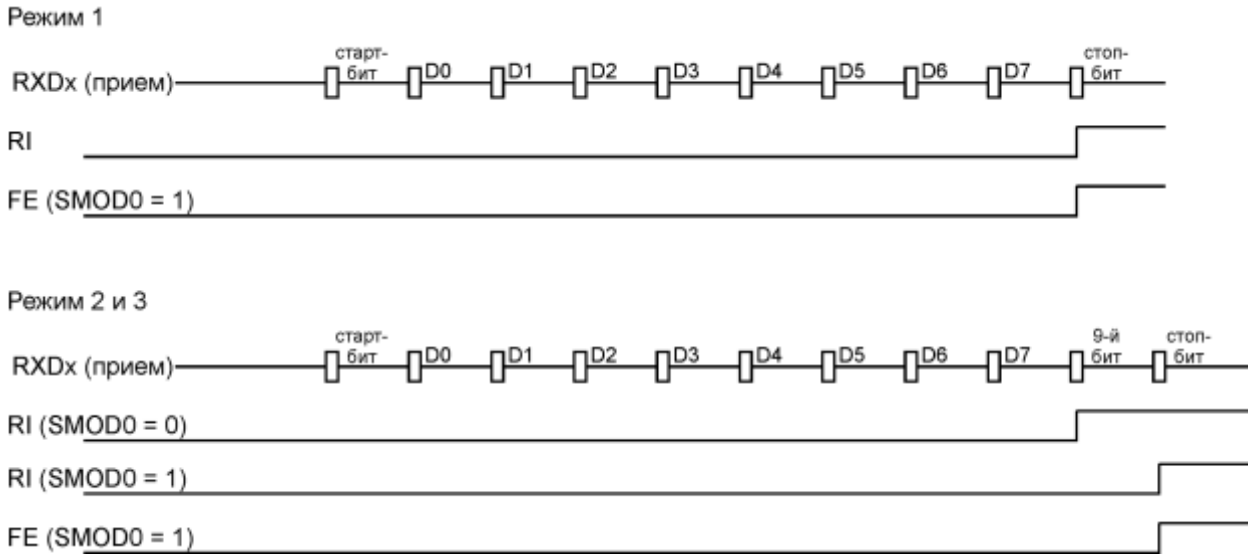


Рисунок 11.4 – Выставление флага FE при приеме некорректного стоп-бита

Автоматическое распознавание адреса

Аппаратно реализованное автоматическое распознавание адреса является важной функцией, позволяющей уменьшить время, затрачиваемое центральным процессором на работу с блоком UARTx. В системах с несколькими устройствами, где существует адресация, наличие блока автоматического распознавания адреса высвобождает процессор от лишних прерываний, поскольку флаг RI выставляется аппаратно лишь в случае, если устройство приняло свой или широковещательный адрес. Эта функция, как правило, требуется в режимах, где пересылаются 9-битные данные (для описываемых в настоящем руководстве блоков UARTx это режимы 2 и 3).

В общем случае, функционирование блока UARTx в системе с несколькими устройствами происходит следующим образом. Когда ведущий контроллер хочет передать посылку одному или нескольким ведомым, он посылает адресный байт, который идентифицирует адресата. Следует помнить, что в адресном байте бит 9 всегда равен «1», а в байте данных – «0».

Примечание – Если блок UARTx является ведомым, его бит SM2 в регистре SCONx должен быть установлен в «1».

Устройство, обнаружившее свой адрес, выставляет флаг RI, сообщая таким образом центральному процессору о том, что далее именно оно будет взаимодействовать с ведущим.

Каждое ведомое устройство имеет индивидуальный адрес, который задается регистром SADDRx и маску адреса с «незначащими» битами (биты, в которые записаны нули), задаваемую регистром SADENx. Наличие маски с «незначащими» битами позволяет ведомому устройству более гибко встраиваться в систему с несколькими устройствами.

После получения адресного байта от ведущего устройства ведомый сравнивает полученный байт с результатом логического «И» регистров SADDRx и SADENx (с учетом «незначащих» бит) и в случае совпадения выставляет флаг RI. Отсюда следует, что для адресации устройства по его индивидуальному адресу (хранится в регистре SADDRx) в регистр SADENx следует записывать маску «1111 1111», т.е. число FFh.

Далее в таблице 11.1 приводятся примеры адресации для одного ведомого и группы ведомых устройств с пояснениями (в примерах условно считается, что все ведомые – это приемопередатчики с регистрами SADDR и SADEN, идентичные блоку UARTx с регистрами SADDRx и SADENx). Указано состояние регистров адреса и маски и значение адресного байта, при получении которого тот или иной ведомый будет считаться

адресованным (т.е. ведомый распознает принятый адресный байт как «свой» адрес).

Таблица 11.1 – Примеры адресации

№ примера	Состояние регистров и адресный байт		Пояснение
1	SADDR SADEN Адрес*	0101 0110b 1111 1100b 0101 01xxb	Пример для ведомого устройства, условно находящегося в системе с несколькими устройствами. Адрес ведомого «0101 0110» задается регистром SADER0 и маской «1111 1100». Согласно маске, биты 0 и 1 адреса закрыты нулями, т.е. являются «незначащими». Логическое «И» регистров дает в результате значение адреса «0101 01xx». Таким образом, чтобы ведомый распознал полученный адрес как «свой», адрес должен быть «0101 01xx», где на месте «xx» могут находиться любые значения
2	Ведомый A: SADDR SADEN Адрес* Ведомый B: SADDR SADEN Адрес* Ведомый C: SADDR SADEN Адрес*	 1111 0001b 1111 1010b 1111 0x0xb 1111 0011b 1111 1001b 1111 0xx1b 1111 0011b 1111 1101b 1111 00x1b	Пример для системы с тремя ведомыми устройствами – A, B и C и условным ведущим. Для каждого из трех ведомых приводится состояние его регистров адреса и маски и значение адресного байта. Варианты адресации: а) для ведомого A бит 0 является «незначащим», а для ведомых B и C, бит 0 равен «1». Тогда для адресации только ведомого A ведущий должен отправить адрес, где бит 0 равен «0» (т.е. 1111 0000b); б) для ведомого A бит 1 равен «1», а для ведомых B и C бит 1 является «незначащим». Тогда для адресации ведомых B и C (за исключением ведомого A) ведущий должен отправить адрес, где биты 0,1 равны «1» (т.е. 1111 0011b); в) для адресации всех трех ведомых ведущий должен отправить адрес, в котором бит 0 равен «1», а биты 1 и 2 равны «0» (т.е. 1111 0001b)
Примечание – Символом «*» отмечен адресный байт, получив который, ведомый определяет, что к нему идет обращение ведущего устройства. Символами «x» обозначены биты, состояние которых не важно при распознавании адреса тем или иным ведомым.			

Одним из вариантов адресации является широковещательный адрес или адрес общего вызова. Полученный адресный байт будет считаться адресом общего вызова, если он совпадет с результатом логического «ИЛИ» регистров SADDRx и SADENx с учетом «незначащих» битов. К примеру, если:

SADDR0 0101 0110b,
SADEN0 1111 1100b,
Полученный адресный байт 1111 111xb,

то полученный адресный байт «1111 111x» будет распознан как широковещательный, поскольку логическое «И» регистров адреса и маски с ним совпадает (значение 0 разряда адресного байта не важно).

Использование «незначащих» битов значительно расширяет возможности использования адресов общего вызова, тем не менее, в большинстве систем в качестве широковещательного адреса используется значение FFh.

В таблице 11.2 приводятся примеры для группы ведомых устройств с пояснениями. Указано состояние регистров адреса и маски и значение адресного байта, при получении которого тот или иной ведомый будет считаться адресованным (т.е. ведомый распознает принятый адресный байт как адрес общего вызова).

Таблица 11.2 – Примеры формирования широковещательного адреса

№ примера	Состояние регистров и адресный байт	Пояснение
1	Ведомый A: SADDR 1111 0001b SADEN 1111 1010b Адрес* 1111 1x11b Ведомый B: SADDR 1111 0011b SADEN 1111 1001b Адрес* 1111 1x11b Ведомый C: SADDR 1111 0010b SADEN 1111 1101b Адрес* 1111 1111b	Для ведомых A и B бит 2 является «незначащим», а для ведомого C –бит 2 равен «1». Для адресации всех ведомых ведущий должен отправить широковещательный адрес «1111 1111». Для адресации ведомых A и B (за исключением C), ведущий должен отправить широковещательный адрес «1111 1011»
Примечание – Символом «*» отмечен адресный байт, получив который, ведомый «поймет», что к нему идет обращение ведущего устройства. Символами «x» обозначены биты, состояние которых не важно при распознавании адреса тем или иным ведомым.		

После сброса в регистрах SADDRx и SADENx находятся нули, т.е. все биты являются «незначащими». Это позволяет обращаться к любому из блоков UARTx без адресации, что дает совместимость с ранее выпущенными интерфейсами, которые не поддерживали автоматическое распознавание адреса.

12 Контроллеры интерфейсов SPI0 и SPI1

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.

В состав микроконтроллера входят два идентичных модуля, реализующих поддержку интерфейса SPI. Названия модулей – SPI0 и SPI1. Далее символы «0» и «1» будут заменены символом «x». Написанное выше одинаково применимо к каждому из двух модулей.

Модуль SPIx 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 SPI0 используется для последовательного внутрисхемного программирования микроконтроллера.

Модуль SPI0 имеет четыре вывода SCK0, MISO0, MOSI0 и SS0#, которые объединены логически по «И» с выводами микроконтроллера P1.7, P1.6, P1.5 и P1.4 соответственно. Модуль SPI1 имеет выводы SCK1, MISO1, MOSI1 и SS1#, которые объединены логически по «И» с выводами микроконтроллера P4.7, P4.6, P4.5 и P4.4 соответственно. Когда модули выключены, они не влияют на работу выводов. В свою очередь, для нормального функционирования модулей следует записывать единицы в биты регистров IOPORT1 и IOPORT4, которые управляют указанными выводами микроконтроллера. Выводы P1.4 и P4.4 являются входами выбора устройства. Когда модуль SPI0/SPI1 работает в режиме ведомого, подача низкого уровня сигнала на вход P1.4/P4.4 активирует модуль. Как правило, управление уровнем сигнала на входе SSx# осуществляется внешним мастером, который активирует ведомого для обмена данными и отключает его в остальное время. Тем не менее, установить низкий уровень сигнала на входе P1.4/P4.4 можно записью нуля в соответствующий бит регистра IOPORT1/IOPORT4. В режиме мастера состояние вывода P1.4/P4.4 не оказывает влияния на работу модуля.

Модуль SPIx 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 SPIx 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 12.1 показана структурная схема взаимодействия блоков модуля SPIx.

Синхронизирующим сигналом модуля SPIx является сигнал Fspi, поступающий от коммутатора частоты синхронизации (см. рисунок 9.1). 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.

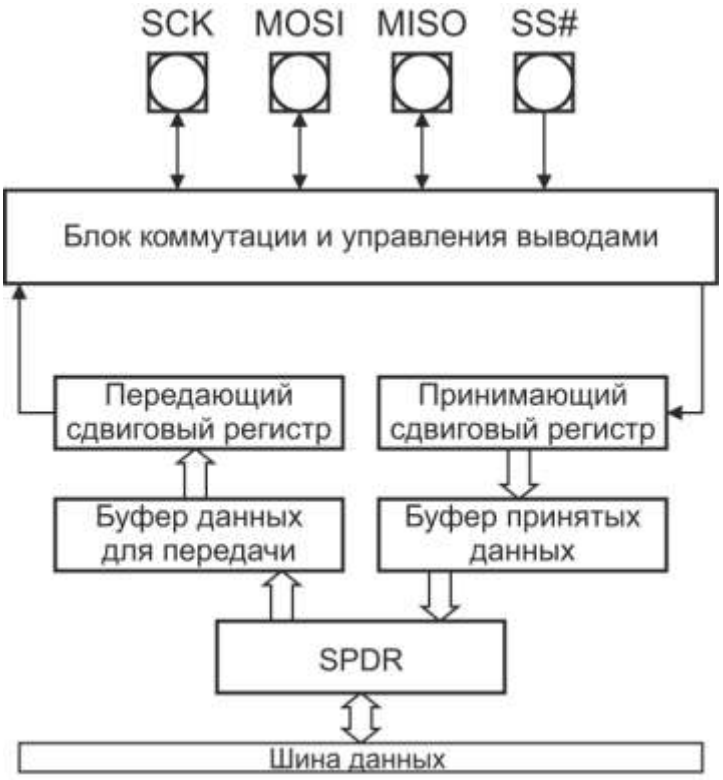


Рисунок 12.1 – Структурная схема взаимодействия блоков модуля SPI

12.1 Управляющие регистры

Управление и контроль состояния модуля SPIx, загрузка и чтение полученных данных осуществляются посредством трех регистров SPCRx, SPSRx и SPDRx.

Регистр SPCR

По умолчанию, модуль SPIx выключен. Регистр управления SPCRx 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 SPIx. 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 SPIx 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 12.2).

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

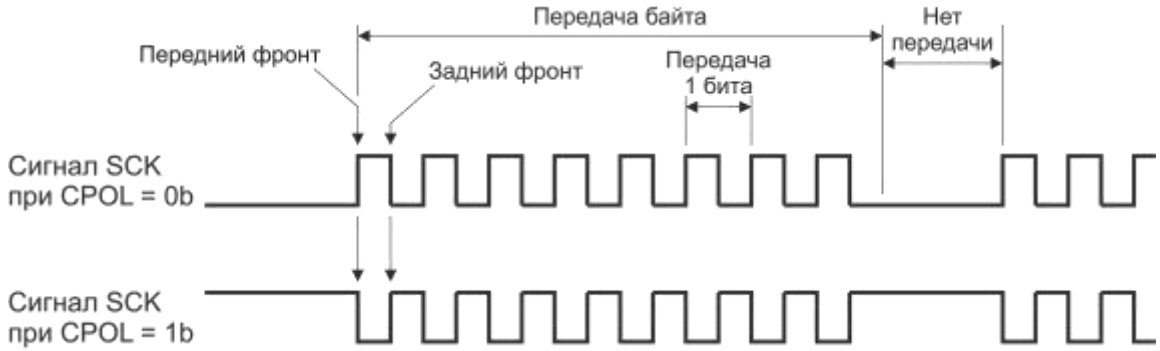


Рисунок 12.2 – Сигнал синхронизации SCK при разных состояниях бита CPOL

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 12.3).

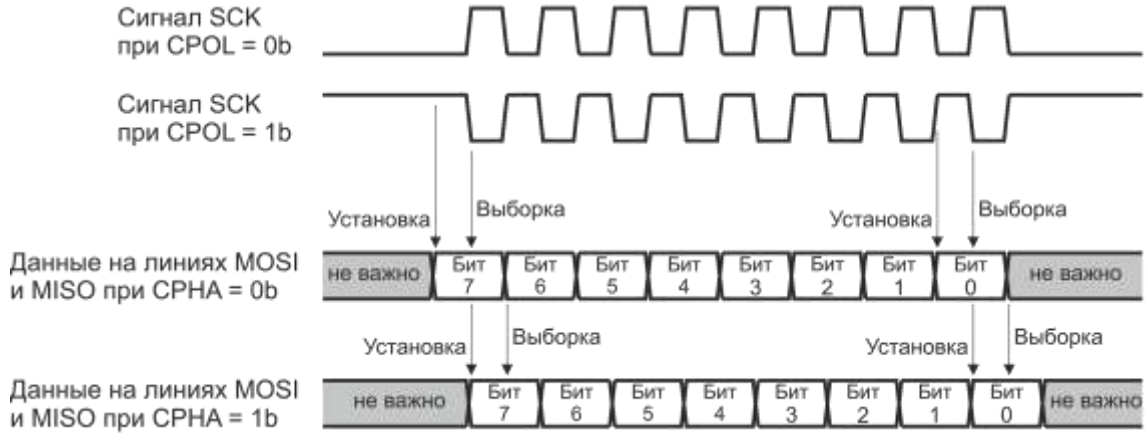


Рисунок 12.3 – Временные диаграммы передач данных в разных режимах

По умолчанию данные передаются старшим битом вперед. Установка и считывание данных выполняется мастером и ведомым одновременно (поэтому на рисунке 12.3 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 12.1).

Таблица 12.1 – 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 SPIx 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. 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 SPIx 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 SPIx 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.

12.2 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 и 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.

13 Контроллер интерфейса I2C

Контроллер интерфейса 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 контроллера:

- 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.

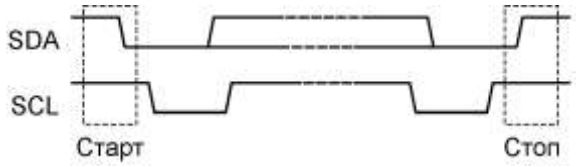


Рисунок 13.2 – Состояния старта и стопа

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).

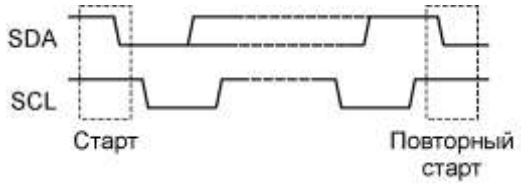


Рисунок 13.3 – Состояние повторного старта

Арбитраж

Арбитраж выполняется в момент времени, когда на линии SCL находится «1». Два устройства могут сгенерировать стартовое состояние в одно и тоже 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.

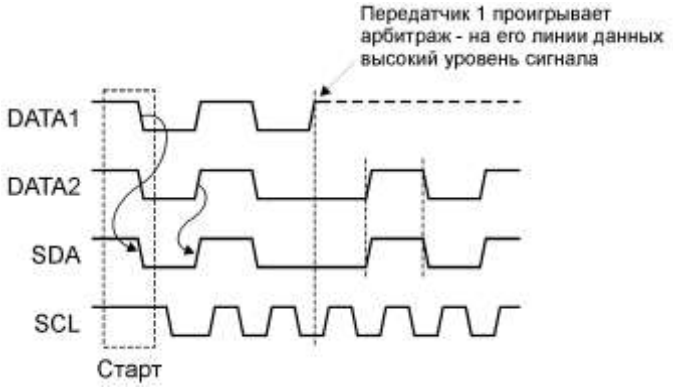


Рисунок 13.4 – Арбитраж на линии SDA

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).

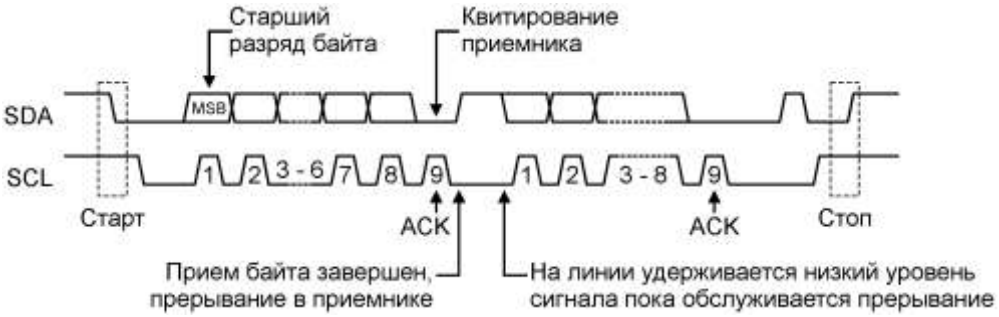


Рисунок 13.6 – Передача данных

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.

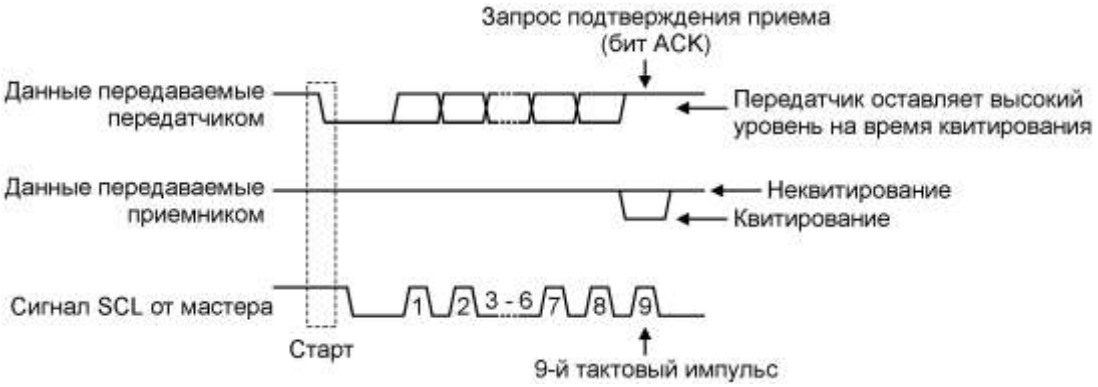


Рисунок 13.7 – Квитирование и неквитирование бита подтверждения ACK

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 желает не отвечать на запрос подтверждения и не подтверждать 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 не квитировать 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).

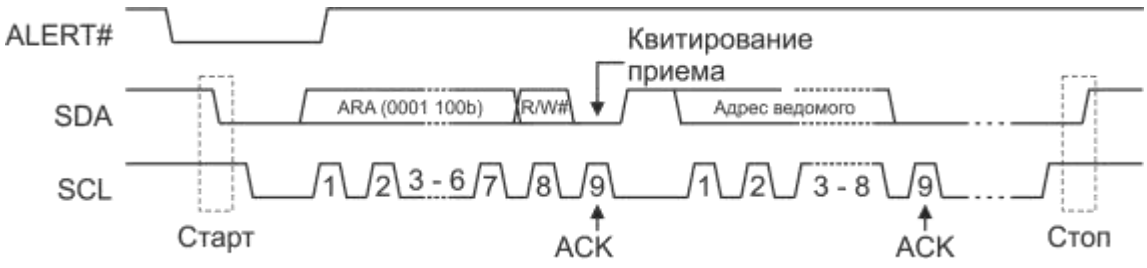


Рисунок 13.9 – Передача адреса отклика на сигнал предупреждения

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 (бит 8 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 руководстве 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 1024 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, который 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).

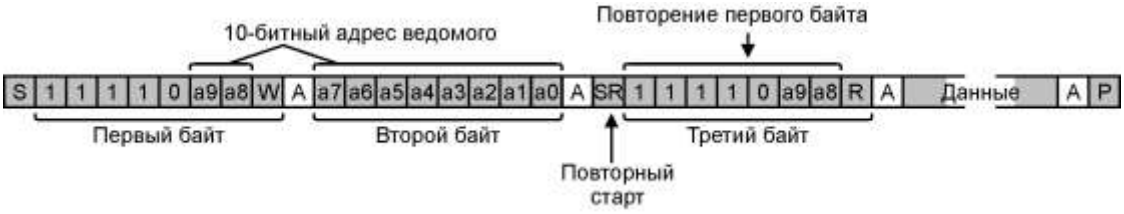


Рисунок 13.11 – Получение данных от ведомого с 10-битным адресом

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, показывающие 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;
- SMBSDA. Осуществляет последовательный прием/передачу данных модуля интерфейса I2C;
- SMBADDR. Содержит собственный адрес модуля интерфейса I2C.

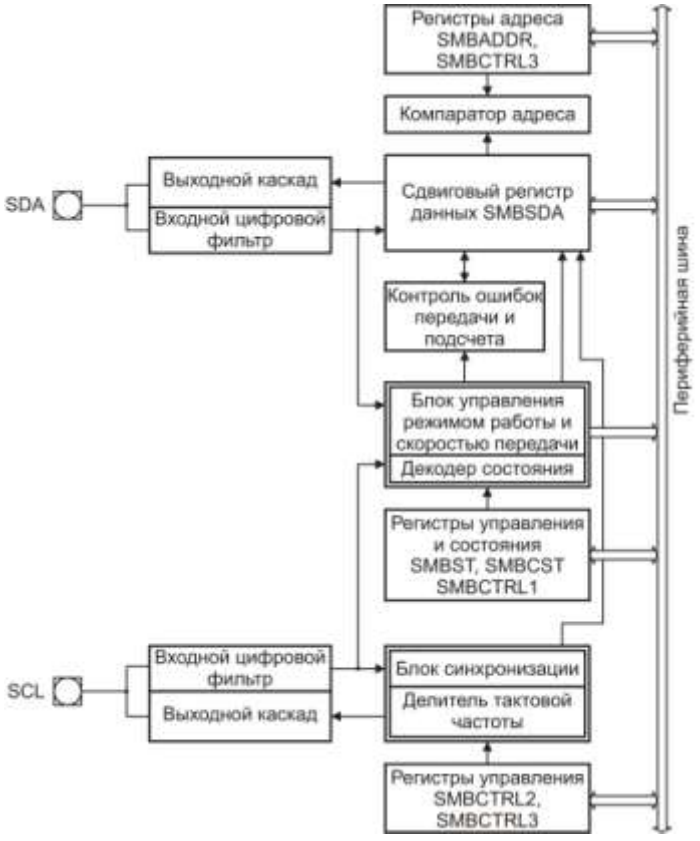


Рисунок 13.12 – Структурная схема модуля I2C

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).

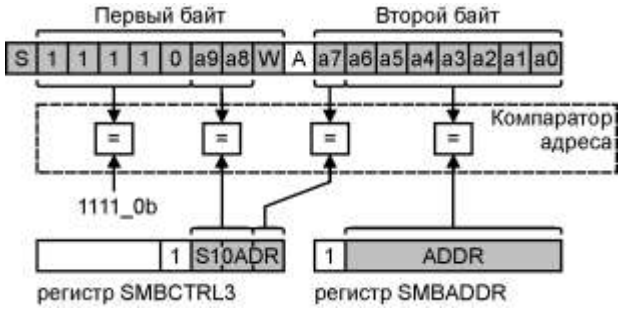


Рисунок 13.13 – Компаратор адреса в режиме 10-битной адресации

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 шести бит определяется значением битового поля 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.

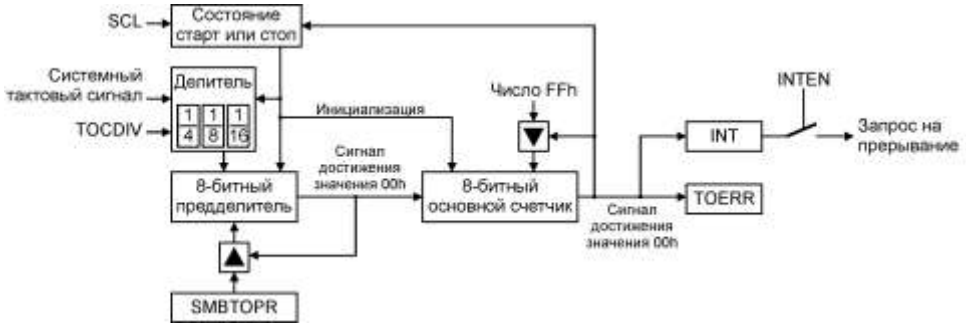


Рисунок 13.14 – Функциональная схема счетчика времени ожидания

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:

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

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 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 65 кГц до 2,35 МГц (при XTAL1 = 33 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 710 кГц до 7,3 МГц (при XTAL1 = 33 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, поддерживающие 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 состояние 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 SMBCTRL2 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 не нулевое значение в битовое поле TOCDIV регистра SMBCST.

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

Функционирование

Контроллер может работать в режиме мастера или ведомого. Также он может функционировать как передатчик или приемник. В целом, 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 13.10 и 13.11 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, hex	Мнемоника	Описание состояния на момент записи кода в поле MODE регистра SMBST	ACK/ NACK
Общий		00	IDLE	IDLE, нет доступной валидной информации о статусе	–
Мастер в режиме FS	–	01	STDONE	Сформировано состояние старта	–
		02	RSDONE	Сформировано состояние повторного старта	–
		03	IDLARL	Потеря арбитража, переход в режим безадресного ведомого	–
	Передача	04	MTADPA	Отправлен адрес ведомого	ACK
		05	MTADNA	Отправлен адрес ведомого	NACK
		06	MTDAPA	Отправлен байт данных	ACK
		07	MTDANA	Отправлен байт данных	NACK
	Прием	08	MRADPA	Отправлен адрес ведомого	ACK
		09	MRADNA	Отправлен адрес ведомого	NACK
		0A	MRDAPA	Принят байт данных	ACK
		0B	MRDANA	Принят байт данных	NACK
	–	0C	MTMCER	Отправлен код мастера, обнаружена ошибка	ACK
Ведомый в режиме FS	Прием	10	SRADPA	Принят адрес	ACK
		11	SRAAPA	Принят адрес после потери арбитража	ACK
		12	SRDAPA	Принят байт данных	ACK
		13	SRDANA	Принят байт данных	NACK
	Передача	14	STADPA	Принят адрес	ACK
		15	STAAPA	Принят адрес после потери арбитража	ACK
		16	STDAPA	Отправлен байт данных	ACK
		17	STDANA	Отправлен байт данных	NACK
	Передача адреса отклика	18	SATADP	Принят адрес отклика на предупреждение	ACK
		19	SATAAP	Принят адрес отклика на предупреждение после потери арбитража	ACK
		1A	SATDAP	Отправлены данные в ответ на получение адреса отклика	ACK
		1B	SATDAN	Отправлены данные в ответ на получение адреса отклика	NACK
	–	1C	SSTOP	Обнаружено состояние останова ведомого	–
		1D	SGADPA	Принят адрес общего вызова	ACK
		1E	SDAAPA	Принят адрес общего вызова после потери арбитража	ACK
Общий		1F	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, hex	Мнемоника	Описание состояния на момент записи кода в поле MODE регистра SMBST	ACK/ NACK
Мастер в режиме HS	–	21	HMTMCOK	Код мастера передан успешно, переход в режим HS	–
		22	HRSDONE	Сформировано состояние повторного старта	–
		23	HIDLARL	Потеря арбитража, переход в режим HS безадресного ведомого	–
	Передача	24	HMTADPA	Отправлен адрес ведомого	ACK
		25	HMTADNA	Отправлен адрес ведомого	NACK
		26	HMTDAPA	Отправлен байт данных	ACK
		27	HMTDANA	Отправлен байт данных	NACK
	Прием	28	HMRADPA	Отправлен адрес ведомого	ACK
		29	HMRADNA	Отправлен адрес ведомого	NACK
		2A	HMRDAPA	Принят байт данных	ACK
		2B	HMRDANA	Принят байт данных	NACK
Ведомый в режиме HS	Прием	30	HSRADPA	Принят адрес	ACK
		32	HSRDAPA	Принят байт данных	ACK
		33	HSRDANA	Принят байт данных	NACK
	Передача	34	HSTADPA	Принят адрес	ACK
		36	HSTDAPA	Отправлен байт данных	ACK
		37	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 7– 1).

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.

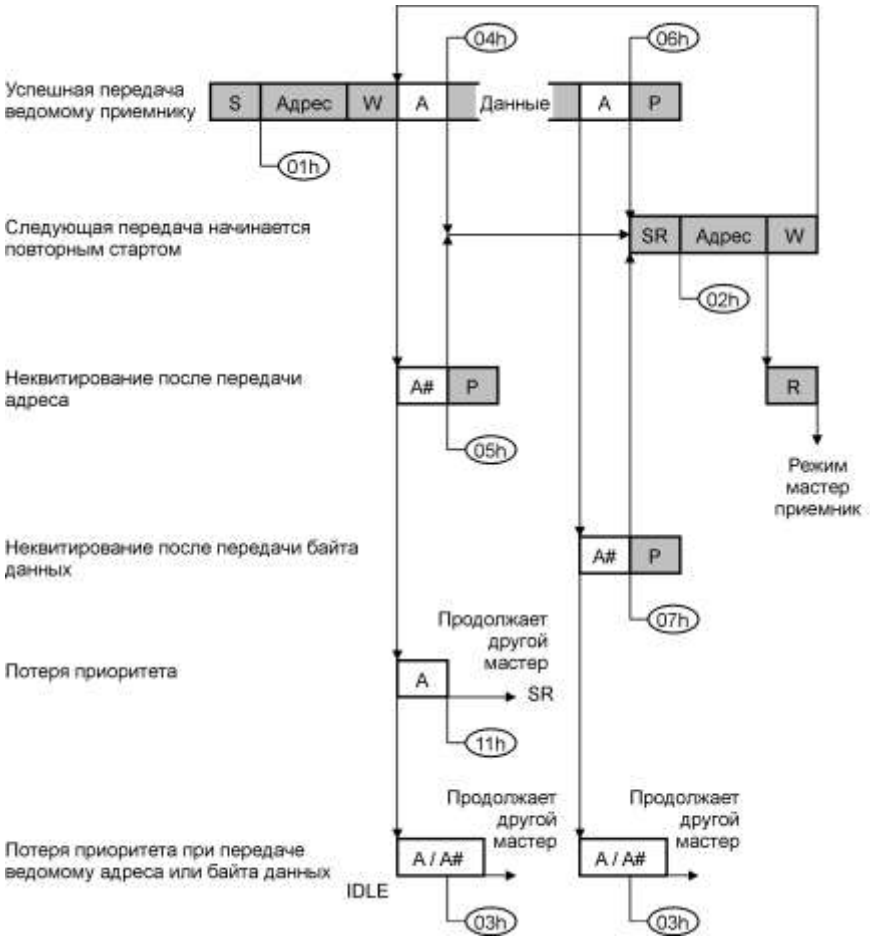


Рисунок 13.16 – Режим FS мастера передатчика

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 единицей в бит CLRST.

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

Дополнительно можно обратиться к приложению В.

На рисунке 13.17 представлено графическое пояснение к описанию режима.

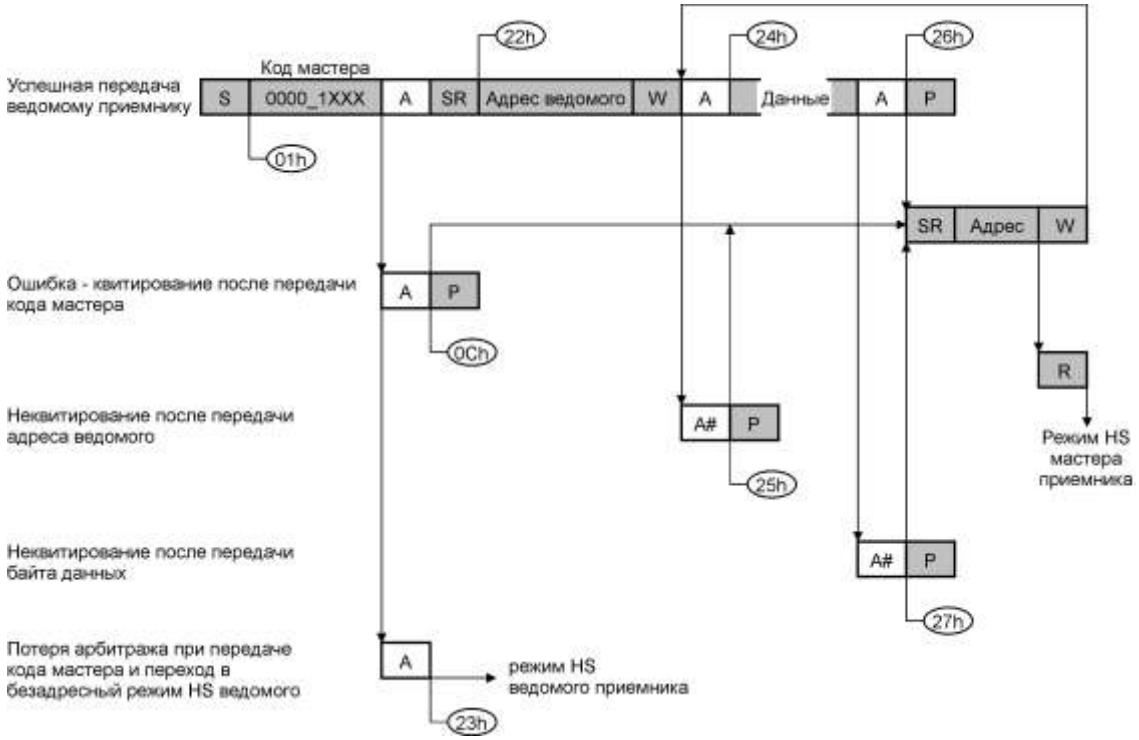


Рисунок 13.17 – Режим HS мастера передатчика

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 то же 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 контролером 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 то же время, если требуется отправка байта CRC, следует установить бит PECNEXT в регистре SMBCST. После сброса флага INT будет

принят последний байт данных и не квитирован. 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 не нулевой, то установится флаг ошибки PECFAULT в регистре SMBCST.

Дополнительно можно обратиться к приложению В.

На рисунке 13.18 представлено графическое пояснение к описанию режима.

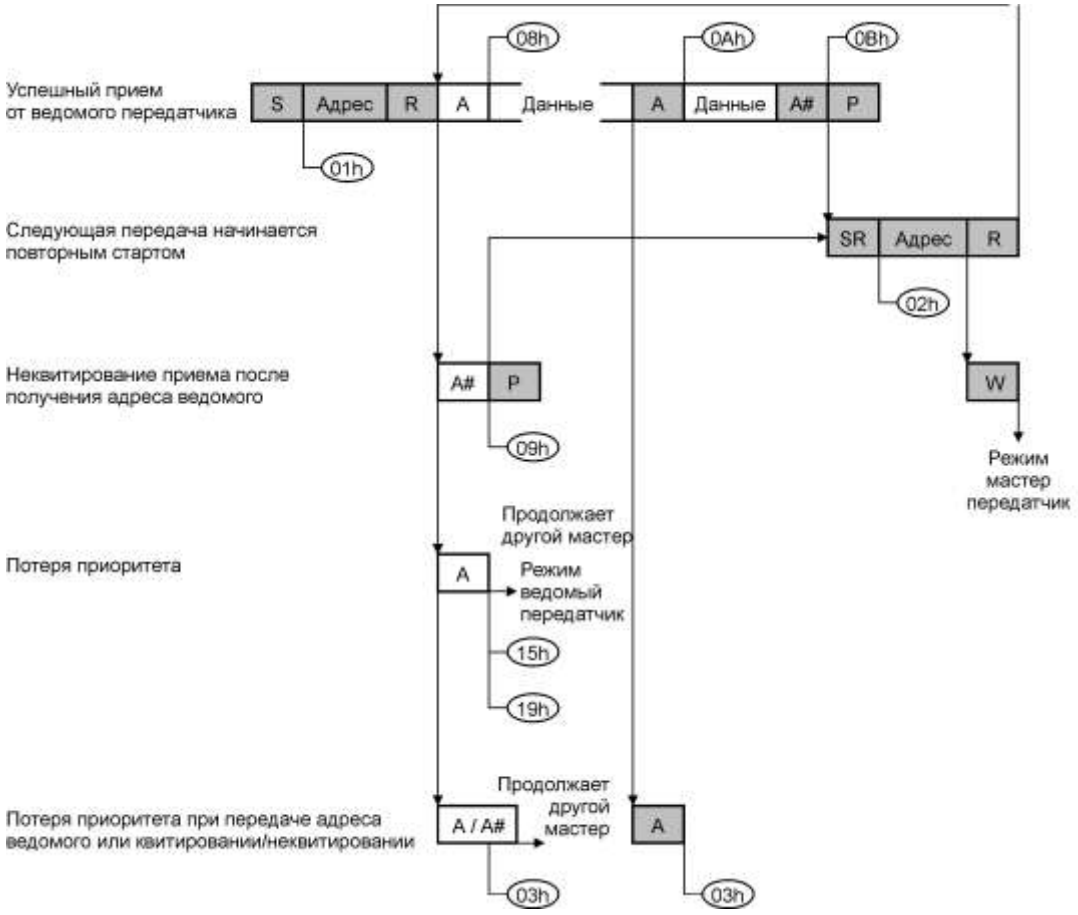


Рисунок 13.18 – Режим FS мастера приемника

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.

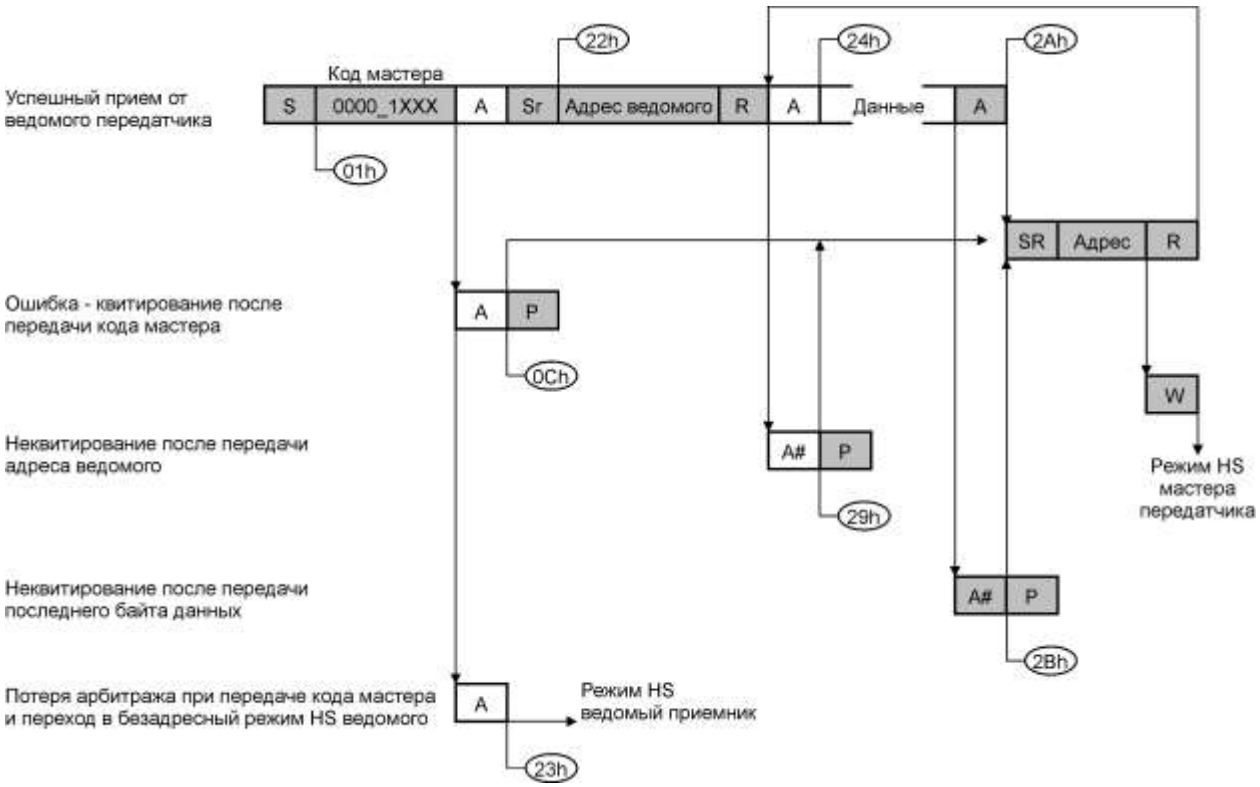


Рисунок 13.19 – Режим HS мастера приемника

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.

Механизм распознавания адреса изложен ранее, в теме «Регистры адреса и компаратор адреса» данного подраздела и показан на рисунке 13.13.

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.

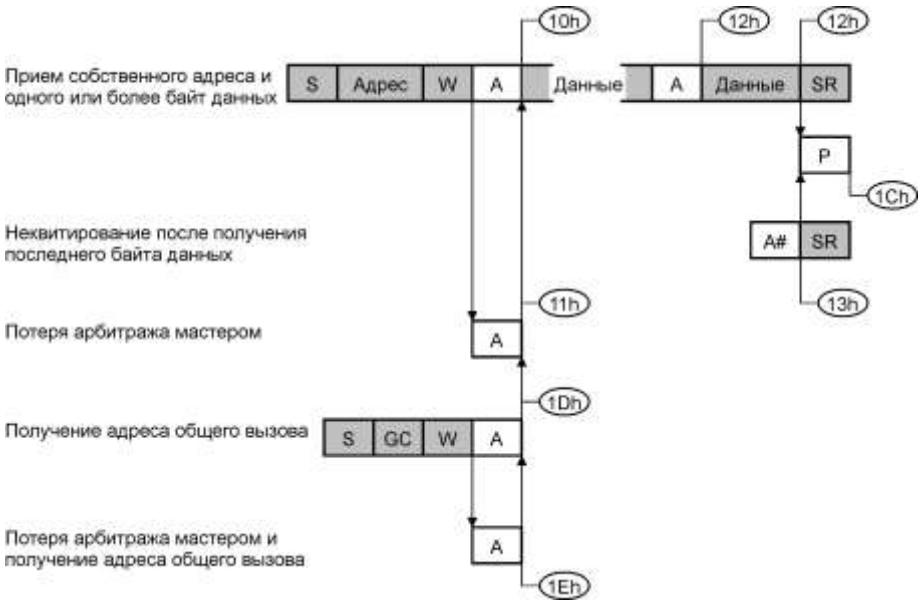


Рисунок 13.20 – Режим FS ведомого приемника

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. режим FS ведомого приемника, описанный ранее).

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 в теме «Режим FS ведомого приемника» данного подраздела.

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, 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.

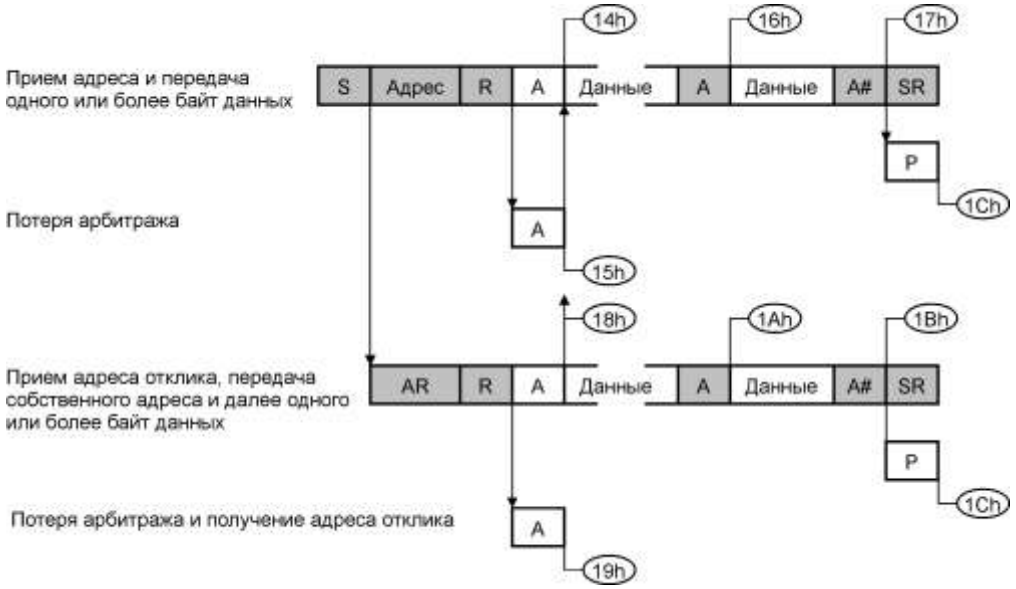


Рисунок 13.22 – Режим FS ведомого передатчика

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.

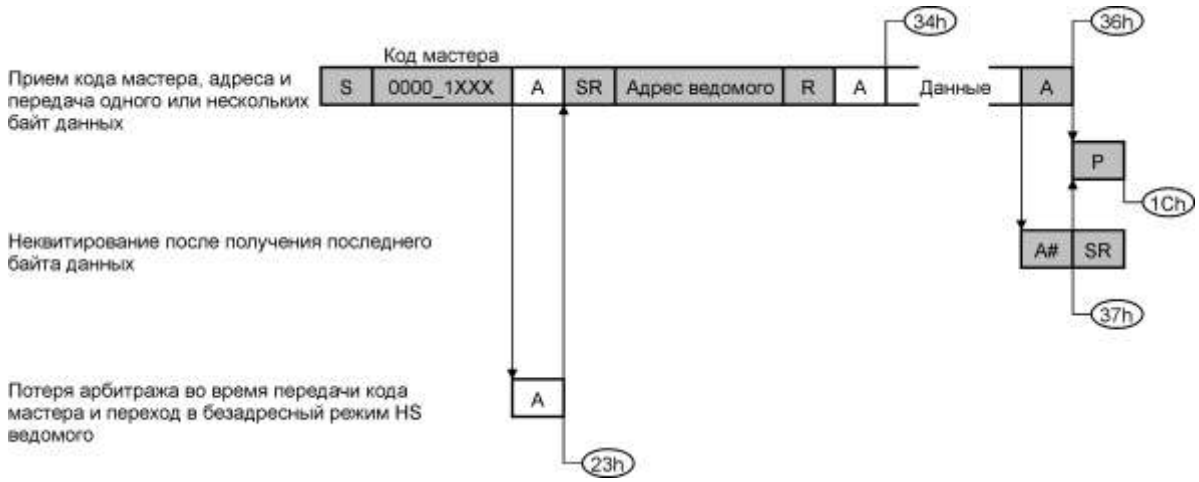


Рисунок 13.23 – Режим HS ведомого передатчика

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 не обнаруженные ошибочные стартовые или стоповые состояния, сформировавшиеся в течение приема ведомым данных. Если считать, что причиной зависания явился именно 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 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 14.1).

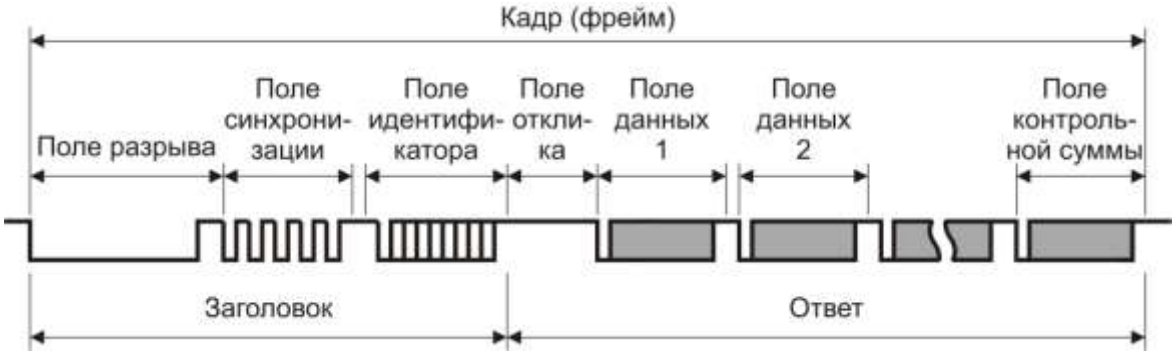


Рисунок 14.1 – Кадр сообщения

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.

14.1 Байтовые поля

Каждое байтовое поле содержит 10 бит – старт бит, 8 бит данных, стоп (см. рисунок 14.2). 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 14.2 – 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 14.3).

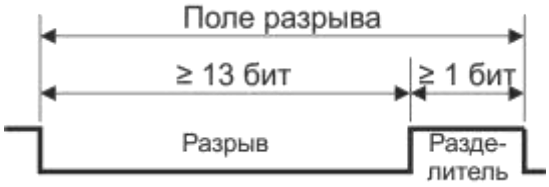


Рисунок 14.3 – Поле разрыва синхронизации

Поле синхронизации содержит сигналы, необходимые для синхронизации задающих генераторов ведомых устройств сети. Поле синхронизации является байтовым полем, имеющем значение 55h (см. рисунок 14.4).

Рисунок 14.4 – 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 14.5). Таким образом, происходит деление 64 идентификационных номеров на четыре комплекта, каждый из которых содержит 16 идентификаторов.

Рисунок 14.5 – Идентификационное поле в версии 1.X

В соответствии с протоколом 1.X количество полей данных в кадре данных определяется по таблице 14.1.

Таблица 14.1 – 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 14.6). 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 LINCONH.

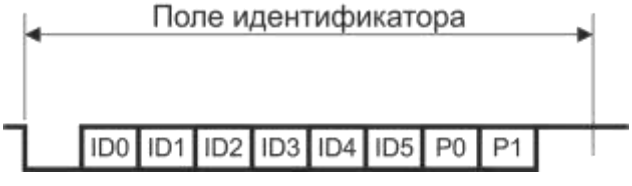


Рисунок 14.6 – Идентификационное поле в версии 2.X

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 14.7).

Рисунок 14.7 – Поле данных

Последним полем в кадре сообщения является поле контрольной суммы (см. рисунок 14.8).

Рисунок 14.8 – 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.

14.2 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 14.9).

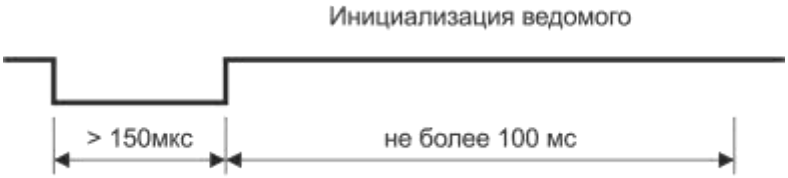


Рисунок 14.9 – Запрос выхода из сна

14.3 Регистры управления и контроля

Все регистры контроллера LIN доступны в первом окне области памяти регистров SFR, т.е. для обращения к ним следует записывать в регистр WSR значение 01h.

Контроллеру LIN принадлежат 16 регистров данных. Восемь регистров, расположенные по адресам 90h – 97h, отведены под восемь байт данных, которые будут передаваться в кадре сообщения.

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

Восемь регистров с адресами 98h – 9Fh отведены под восемь байт данных, которые будут получены. Количество принимаемых байт определяется полем PID кадра сообщения или полем DATALEN.

Регистр управления состоит из двух регистров – LINCONH и LINCONL. Старший байт 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 14.2).

Таблица 14.2 – Тип кадра сообщения и соответствующие действия устройства

Тип кадра сообщения, заданный полем MSGTYPE	Действия, в зависимости от типа устройства	
	Мастер	Ведомый
BUSWAIT	Не активен	
HEADER	Передача	Прием
HEADER+RESPONCE		
RESPONSE RECEIVE	Прием	Прием
RESPONSE TRANSMIT	Не влияет	Передача ответа
WAKE	Передача запроса на пробуждение	
Примечание – Примененные в таблице типы означают: HEADER – в кадре передаются поля разрыва, синхронизации и идентификатора; RESPONSE – в кадре передаются поля данных и контрольной суммы.		

Регистр состояния состоит из двух регистров LINSTATH и LINSTATL, в которых находятся биты, несущие информацию о действиях контроллера LIN, его текущем состоянии и ошибках передачи. Флаги регистров LINSTATH и LINSTATL маскируются битами регистров разрешений прерываний LININTENH и LININTENL. По умолчанию состояние регистров разрешений прерываний 00h 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 LINSTATH или LINSTATL, 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 LINSTATL.

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 состоит из двух регистров – LINBREAKH и LINBREAKL и используется, если устройство в сети является ведомым. В регистрах 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 длинной 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 LINSPEEDH записывается значение 01h, а в регистр LINSPEEDL записывается значение 96h.

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

Регистр настройки скорости передачи состоит из двух регистров – LINSPEEDH и LINSPEEDL и 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 скорости LIN 20 Кб/с рассчитанное значение равно 12Ch.

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 LINCSR контрольной суммы используется для считывания полученного значения контрольной суммы. Обновление регистра происходит одновременно с

установкой флага CSRREC. Если значения контрольной суммы, вычисленное на основе полученных данных, не совпадает с принятой контрольной суммой, то устанавливается флаг CSRERR.

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 – 33 МГц, скорость LIN – 20 Кбит/с.

```
MOV    LINSPEEDL,#9CH        ; Flin = 20 кГц, 33 МГц
MOV    LINSPEEDH,#01H
MOV    LINCONH ,#74H         ; Мастер, протокол 2.X, 4 байта данных
MOV    LINDTX0,#0AAH         ; 1 байт
MOV    LINDTX1,#55H          ; 2 байт
MOV    LINDTX2,#0CCH         ; 3 байт
MOV    LINDTX3,#33H          ; 4 байт
MOV    LINPIDTX ,#37H        ; Передаваемый PID
MOV    LINCONL ,#03H         ; HEADER+RESPONSE
```

15 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 8 байт) 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 и состояния CDCON, CDSTATE и CDDATNUM, 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. 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 побайтно по адресу, указанному в таблице 15.1. Если данные (побайтно) поступают в порядке D0, D1, D2 и т.д., то их размещение в регистре CDDATA будет соответствовать указанному на рисунке 15.1.

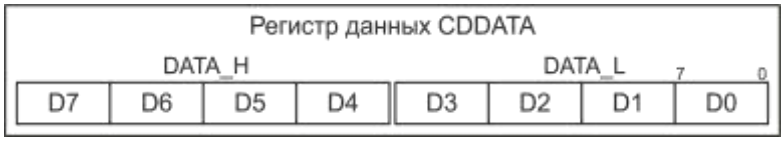


Рисунок 15.1 – Структура регистра CDDATA

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 восьмикратное

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 побайтно. Если данные (побайтно) поступают в порядке K0, K1, K2 и т.д., то их размещение в регистре CDKEY будет соответствовать указанному на рисунке 15.2. Регистр CDKEY доступен только для записи. На рисунке 15.3 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 15.2 – Структура регистра CDKEY

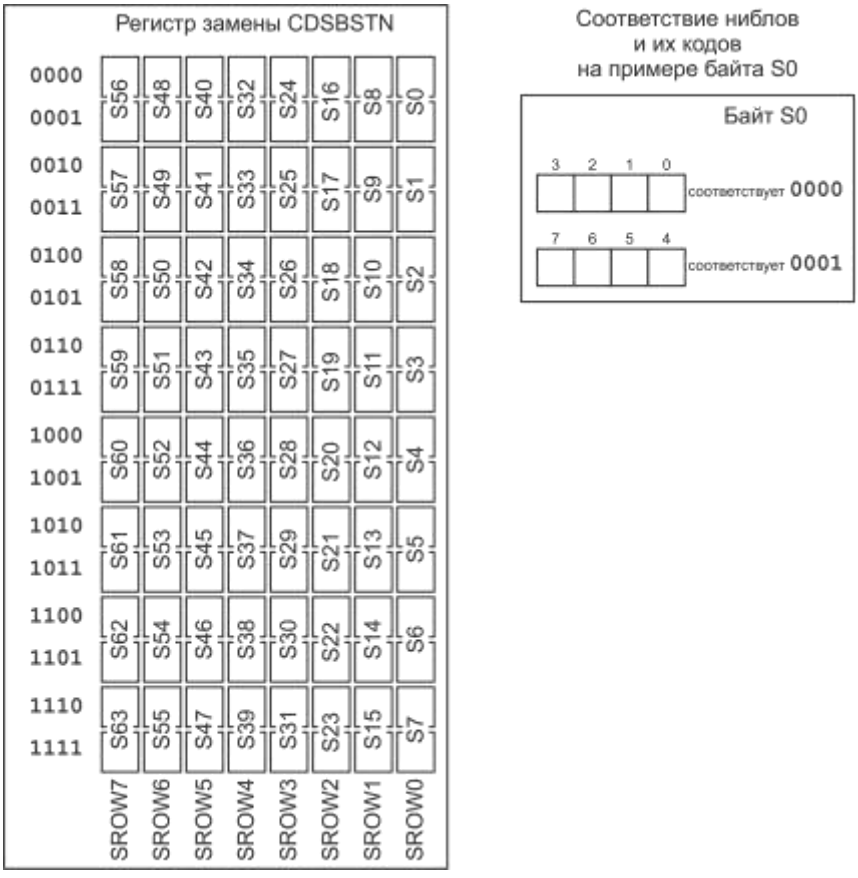


Рисунок 15.3 – Структура регистра CDSBSTN

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 побайтно. Если данные (побайтно) поступают в порядке S0, S1, S2 и т.д., то их размещение в регистре CDSBSTN будет соответствовать указанному на рисунке 15.3. 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 15.3, соответствует таблице подстановки, применяемой в процессе кодирования/декодирования, с восемью столбцами и 16 строками. Каждому полубайту соответствует код (двоичный порядковый номер).

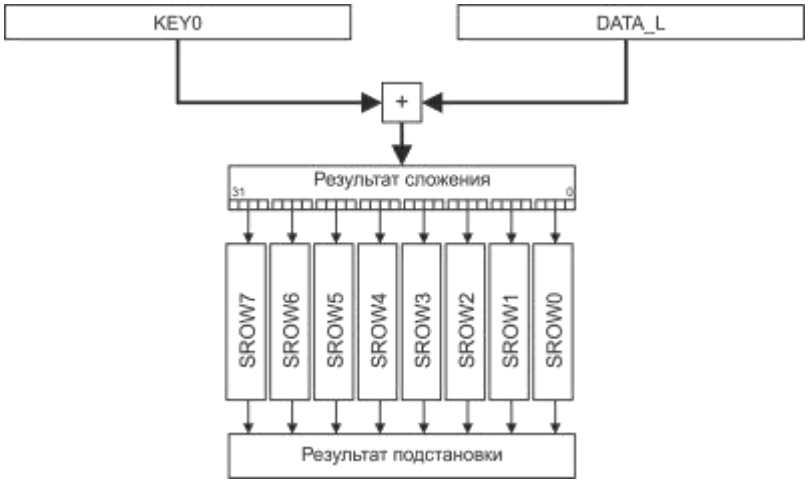


Рисунок 15.4 – Частичная функциональная схема алгоритма режима простой замены

Согласно алгоритму кодирования/декодирования, двойное слово обрабатываемых данных разбивается на восемь полубайт, каждому из которых соответствует один из столбцов (см. рисунок 15.4). 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 побайтно. Если данные (побайтно) поступают в порядке UW0, UW1, UW2 и т.д., то их размещение в регистре CDUW будет соответствовать указанному на рисунке 15.5. Регистр CDUW доступен только для записи. В пределах регистра данные разбиваются на два 32-разрядных подблока с названиями UW_H и UW_L.

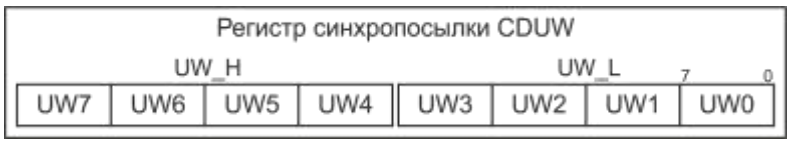


Рисунок 15.5 – Структура регистра CDUW

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 15.6.

Рисунок 15.6 – Структура регистра CDAUTH

Регистр является 64-битным и доступен только для чтения. Для получения имитовставки требуется восьмикратное 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 15.1 – 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	Состояние после сброса
CDDATA	64	83h (WSR = 01h)	8 (запись/чтение)	00h
CDKEY	256	84h (WSR = 01h)	32 (только запись)	00h
CDSBSTN	512	85h (WSR = 01h)	64 (только запись)	00h
CDUW	64	86h (WSR = 01h)	8 (только запись)	00h
CDAUTH	64	87h (WSR = 01h)	8 (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 вычисление 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 32 байта данных;
- в регистр замены CDSBSTN записываются 64 байта данных;
- если предполагается использование одного из режимов гаммирования, то в регистр синхропосылки CDUW записываются восемь байт данных;
- в регистр данных CDDATA записываются восемь байт данных, подлежащих кодированию/декодированию;
- если известно количество блоков данных и если предполагается использование регистра CDDATNUM, то соответствующее значение должно быть записано в регистр;
- в регистре управления CDCON 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 15.1. Если число записанных/прочитанных байт 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.

16 Модуль магистрального последовательного интерфейса ГОСТ Р 52070–2003

Модуль представляет собой устройство, поддерживающее обмен данными с другими устройствами (контроллерами) посредством магистрального последовательного интерфейса (ГОСТ Р 52070–2003). Совместимость со стандартом MIL-STD-1553B.

На физическом уровне интерфейс представляет собой последовательную шину данных (экранированная витая пара), к которой подключены устройства. Допустимыми устройствами являются: контроллер шины, монитор шины и оконечные устройства (удаленные терминалы). Контроллер шины является ведущим устройством. Он единственный инициирует любой обмен информацией и контролирует работу сети. Контроллер шины может обращаться к любому из оконечных устройств (максимальное количество 31), каждому из которых присвоен уникальный 5-разрядный адрес. Монитор шины – пассивное устройство, подключенное к шине данных для отслеживания и записи передаваемой по шине информации. Для получения подробной информации о протоколе следует обратиться к ГОСТ Р 52070–2003.

Модуль магистрального последовательного интерфейса имеет два канала A и B с выводами TXOUT_A, TXOUT_A#, RXIN_A, RXIN_A#, TXOUT_B, TXOUT_B#, RXIN_B, RXIN_B#. Каналы полностью идентичны. Модуль может функционировать в одном из трех режимов:

- контроллер шины (КШ);
- оконечное устройство (ОУ);
- монитор шины (МШ).

Установка режима, а также других дополнительных параметров работы осуществляется посредством регистра конфигурации BSICONF1. Для каждого выбранного режима работы необходимо правильно настроить делитель частоты для получения частоты передачи 1 МГц (требование протокола). Значение делителя, которое следует записать в поле DIV регистра BSICONF1 определяется по формуле:

$$DIV = fosc/4,$$

где fosc – тактовая частота микроконтроллера в МГц.

Примечание – Для нормальной работы модуля магистрального последовательного интерфейса частоту сигнала Fosc желательно выбирать кратной четырем.

Один из трех режимов работы модуля задается полем MODE регистра BSICONF1.

16.1 Контроллер шины

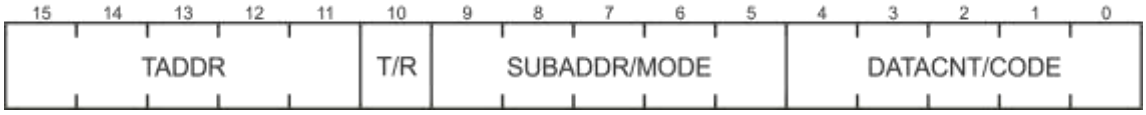
Устройство является контроллером шины по умолчанию. Контроллер шины инициирует любой обмен информацией в сети и контролирует работу сети. Контроллер шины может обращаться к любому из оконечных устройств по его адресу, посылать и принимать одиночные сообщения. Одно сообщение может состоять из одного или более слов.

Используются следующие типы слов:

- командное (см. таблицу 16.1);
- данных (см. таблицу 16.3);
- ответное (см. таблицу 16.4);

Командное слово позволяет указать устройства на шине, между которыми нужно произвести передачу слова/слов данных, а также направление передачи. Ответное слово передается оконечным устройством и несет информацию о состоянии устройства и корректности передачи.

Таблица 16.1 – Командное слово (КС)



TADDR	15-11	Адрес оконечного устройства. В поле записывается адрес оконечного устройства, к которому обращается контроллер шины. Значение 1Fh является групповым адресом и служит для обращения ко всем оконечным устройствам одновременно.	
T/R	10	Направление передачи данных	
		1	Передача
		0	Прием
SUBADDR/ MODE	9-5	Подадрес/Управление	
		00h или 1Fh	Режим «Управление». Значение в поле DATACNT/CODE является кодом команды управления (см. таблицу 16.2)
		01h – 1Eh	Режим «Подадрес». В поле находится адрес подчиненного устройства (абонента), подключенного непосредственно к выбранному оконечному устройству. Значение в поле DATACNT/CODE указывает количество слов данных для передачи/приема.
DATACNT/ CODE	4-0	Количество данных/Код команды управления. Режим работы битового поля устанавливается полем SUBADDR/MODE	

Команды управления

Команды управления передаются в поле DATACNT/CODE командного слова контроллера шины и применяются только для управления оконечными устройствами (не для обмена данными).

Команды управления с кодами 00h – 08h применяются без слов данных, а команды управления с кодами 10h – 15h применяются с одним словом данных.

Возможность использования той или иной команды управления в групповых сообщениях и состояние бита T/R указаны в таблице 16.2.

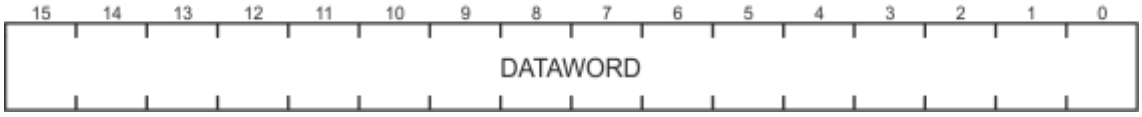
Таблица 16.2 – Команды управления, передаваемые в командном слове

Код команды	Название команды управления	Бит T/R	Возможность применения	
			в групповых сообщениях	со словом данных
1	2	3	4	5
00h	Принять управление интерфейсом	1	–	–
01h	Синхронизация	1	+	
02h	Передать ответное слово*	1	–	
03h	Начать самоконтроль	1	+	
04h	Блокировать передатчик*	1	+	

Окончание таблицы 16.2

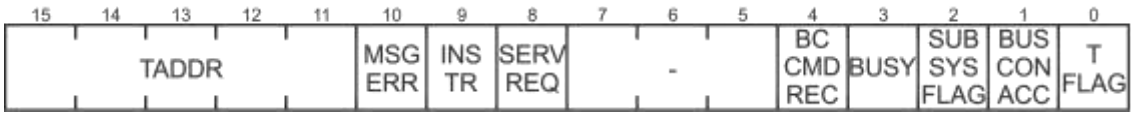
1	2	3	4	5
05h	Разблокировать передатчик*	1	+	–
06h	Блокировать признак неисправности оконечного устройства*			
07h	Разблокировать признак неисправности оконечного устройства *			
08h	Установить оконечное устройство в исходное состояние			
10h	Передать векторное слово*	1	–	+
11h	Синхронизация со словом данных	0	+	
12h	Передать последнюю команду*	1	–	
13h	Передать слово встроенной системы контроля (ВСО) оконечного устройства к контроллеру шины			
14h	Блокировать выбранный передатчик	0	+	
15h	Разблокировать выбранный передатчик			
09h-0Fh, 16h-1Fh	Зарезервировано. Не использовать!			
Примечание – В случае получения управляющей команды, отмеченной знаком «*», оконечное устройство не формирует запрос на прерывание, а только аппаратно выполняет предписанное командой действие.				

Таблица 16.3 – Слово данных (СД)



Поле	Бит	Тип	Описание
DATAWORD	15-0	Запись Чтение	Данные

Таблица 16.4 – Ответное слово (ОС)



Поле	Бит	Тип	Описание
1	2	3	4
TADDR	15-11	Аппаратное влияние	Собственный адрес оконечного устройства
MSGERR	10	Аппаратное влияние	Ошибка в сообщении. Установленный бит указывает на то, что оконечное устройство не смогло осуществить корректный прием
INSTR	9	Аппаратное влияние	Бит распознавания (всегда ноль)
SERV REQ	8	Аппаратное влияние	Запрос на обслуживание Устанавливается посредством бита SERVREQ регистра BSICON в режиме оконечного устройства

Окончание таблицы 16.4

1	2	3	4
BCCMD REC	4	Аппаратное влияние	Готовность работы с групповыми сообщениями. Устанавливается посредством бита BCMSGEN регистра BSICON в режиме оконечного устройства
BUSY	3	Аппаратное влияние	Абонент занят. Устанавливается посредством бита ABUSY регистра BSICON в режиме оконечного устройства. Указывает на то, что подчиненное устройство (абонент) оконечного устройства занят
SUBSYS FLAG	2	Аппаратное влияние	Неисправность абонента. Устанавливается посредством бита AERR регистра BSICON в режиме оконечного устройства. Указывает на ошибки в работе подчиненного устройства (абонента) оконечного устройства
BUSCON ACC	1	Аппаратное влияние	Бит подтверждения принятия управления. Устанавливается аппаратно в случае, если оконечное устройство не было занято и получило команду управления «Принять управление интерфейсом» с кодом 00h
TFLAG	0	Аппаратное влияние	Неисправность оконечного устройства. Устанавливается посредством бита TERR регистра BSICON в режиме оконечного устройства. Указывает на ошибки в работе оконечного устройства.
–	7–5	–	Зарезервировано
Примечание – Ответное слово не передается после приема группового сообщения и в случае обнаружения ошибки в принятых данных.			

Форматы сообщений

В зависимости от того, какие слова передаются в пределах сообщения, сообщение может иметь тот или иной формат. Форматы делятся на две группы – форматы основных сообщений и форматы групповых сообщений (по ГОСТ Р 52070–2003).

Форматы основных сообщений

Применяются для передачи информации, предназначенной одному из оконечных устройств с получением от него соответствующего ответа. Описания форматов приведено в таблице 16.5.

Таблица 16.5 – Форматы основных сообщений

Название формата	Алгоритм передачи	Состав сообщения
1	2	3
Формат 1	Передача данных от контроллера шины к оконечному устройству	Передача: - командное слово; - последовательно все слова данных (до 32 слов). Прием: - ответное слово.
Формат 2	Передача данных от оконечного устройства к контроллеру шины	Передача: - командное слово. Прием: - ответное слово; - слова данных (количество согласно запросу)

Окончание таблицы 16.5

1	2	3
Формат 3	Передача данных от оконечного устройства к оконечному устройству	Передача: - командное слово для приемника; - командное слово для передатчика. Прием: - ответное слово от передатчика; - слова данных (количество согласно запросу); - ответное слово от приемника
Формат 4	Передача команды управления от контроллера шины к оконечному устройству	Передача: - командное слово. Прием: - ответное слово
Формат 5	Передача команды управления от контроллера шины к оконечному устройству и получение от него слова данных	Передача - командное слово. Прием - ответное слово; - слово данных
Формат 6	Передача команды управления и одного слова данных от контроллера шину к оконечному устройству	Передача - командное слово; - слово данных. Прием: - ответное слово

Форматы групповых сообщений

Применяются для передачи информации, предназначенной нескольким оконечным устройствам без получения от них ответных слов. Групповые сообщения всегда начинаются с передачи команды общего вызова с кодом 1Fh в командном слове. Каждое оконечное устройство, которое может принять команду общего вызова, после ее обнаружения устанавливает соответствующий флаг в регистре, где хранится ответное слово, но не передает ответное слово.

Описания форматов приведено в таблице 16.6.

Таблица 16.6 – Форматы групповых сообщений

Название формата	Алгоритм передачи	Состав сообщения
1	2	3
Формат 7	Передача данных (в групповом сообщении) от контроллера шины к оконечным устройствам	Передача: - командное слово; - последовательно все слова данных (до 32 слов)
Формат 8	Передача данных (в групповом сообщении) от оконечного устройства к оконечному устройству	Передача: - командное слово для приемника; - командное слово для передатчика. Прием: - ответное слово от передатчика; - слова данных (количество согласно запросу)

Окончание таблицы 16.6

1	2	3
Формат 9	Передача групповой команды управления от контроллера шины к оконечным устройствам	Передача: - командное слово
Формат 10	Передача групповой команды управления и одного слова данных от контроллера шины к оконечным устройствам	Передача: - командное слово; - слово данных

Инициализация контроллера шины

1 В поле MODE записать 00b (BSICONF1).

2 В регистр BSICMD1_0 записать младший байт командного слова.

3 В регистр BSICMD1_1 записать старший байт командного слова.

В случае, если сообщение имеет формат 3 или 8, дополнительно записать младший и старший байты второго командного слова в регистры BSICMD2_0 и BSICMD2_1 соответственно.

4 Если предполагается передача, то записать в регистры BSIDATAx_0 и BSIDATAx_1 соответственно младший и старший байты слова для передачи («x» является номером слова данных и меняется от 1 до 32, поэтому слова будут передаваться в том же порядке).

В случае приема в регистры будут записаны принятые данные.

Примечание – Символ «x» в названии регистров является номером слова данных.

5 Установить бит START (BSICONF2).

Далее начинается обмен данными. Количество и тип передаваемых и принимаемых слов зависит от заданного формата сообщения.

Полученное ответное слово сохраняется в регистрах BSISW1_1 (старший байт) и BSISW1_0 (младший байт). Второе ответное слово (в случае, если был задан формат 3) сохраняется в регистрах BSISW2_1 (старший байт) и BSISW2_0 (младший байт).

По окончании обмена бит START сбрасывается, формируется запрос на прерывание и устройство переходит в режим ожидания следующего старта.

Если в процессе обмена возникают ошибки,или в ответном слове возвращаются не нулевые биты, то в зависимости от состояния битов регистра BSICONT возможны следующие действия:

- если разрешены повтор (значение RPT > 000b) и переход на альтернативную линию (установлен бит ALTCHNLEN), то обмен данными будет повторяться заданное число раз, причем каждый раз канал будет меняться на альтернативный;

- если повтор разрешен, а перехода на альтернативную линию нет, то обмен будет повторяться только в канале, который задан битом CHNLNUM;

- если повтор запрещен, то происходит остановка по ошибке. Бит START сбрасывается, устанавливаются биты DWERR и INT регистра BSIINTSTAT, формируется запрос на прерывание и устройство переходит в режим ожидания.

Примечание – Остановка по ошибке происходит также, если все повторы кончаются ошибкой.

Прерывания

В режиме контроллера шины возможны три источника прерываний:

- по окончании успешной передачи сообщения;

- по ошибке, если время непрерывной передачи превысило 800 мкс (при этом установится флаг GENINCHNL);

- по ошибке (при этом установится флаг DWERR).

Примечание – Модулю магистрального последовательного интерфейса выделен один вектор прерывания, поэтому для определения источника прерывания следует считывать состояние регистра BSIINTSTAT.

16.2 Оконечное устройство

Оконечное устройство выполняет прием и дешифрирование командных слов, выполняет команды управления, формирует и выдает через выбранный основной/ резервный канал ответные слова, отслеживает ошибки в сообщениях. Каждому оконечному устройству присваивается адрес в диапазоне от 00h до 1Eh.

Если устройство функционировало как контроллер или монитор шины, то для переключения в режим оконечного устройства нужно записать значение 01b в поле MODE (BSICONF1).

Инициализация оконечного устройства

1 В регистре BSICONF2 задать адрес, возможность работы с групповым адресом, использовать ли десятый принятый бит для идентификации командного и ответного слов и установить бит START.

Примечание – При установленном бите START смена адреса устройства не возможна.

После того, как будет установлен бит START, устройство переходит в режим ожидания команды от контроллера шины.

Полученное достоверное командное слово сохраняется в регистрах BSICMD1_1 (старший байт) и BSICMD1_0 (младший байт), затем выполняются действия, предписанные полученной командой, и формируется запрос на прерывание.

Если получены данные, то они сохраняются в регистрах BSIDATAx_0 и BSIDATAx_1 («x» меняется от 1 до 32 и полученные слова сохраняются в том же порядке). Для передачи данных последние должны быть предварительно записаны в эти регистры.

Слово данных, получаемое с командой управления, сохраняется в регистрах BSIDATA1_1 и BSIDATA1_0. Для передачи слова данных в ответ на команду управления, это слово должно быть предварительно записано в указанные регистры.

При приеме команд управления, выполнение которых требует вмешательства центрального процессора, в регистре BSIINTSTAT устанавливается флаг CTRLCMD и формируется запрос на прерывание.

Для передачи ответного слова используются данные, хранящиеся в регистре BSISW1_0.

Прерывания

В режиме оконечного устройства возможны три источника прерываний:

- по окончании выполнения действий, предписанных полученной командой управления;
- по ошибке, если время непрерывной передачи превысило 800 мкс (при этом установится флаг GENINCHNL);
- если обнаружена ошибка длины команды, паритета или кода Манчестер II слова данных (при этом установится флаг DWERR).

Примечание – Для определения источника прерывания следует считывать состояние регистра BSIINTSTAT.

16.3 Монитор шины

Монитор шины непрерывно следит за работой двух каналов (основного и резервного), фиксирует начало сообщения, определяет его формат, сохраняет принятые слова в соответствующих регистрах и формирует запросы на прерывания. Флаги регистра BSIINTSTAT информируют об окончании сообщения или истечении времени, необходимого для сообщения в данном формате.

Переключение в режим монитора шины приводит к сбросу внутреннего 32-разрядного таймера. Также таймер можно сбросить, установив бит RESET в регистре BSICONT.

После принятия достоверного командного слова в регистры BSITIMER3, BSITIMER2, BSITIMER1, BSITIMER0 записываются значения таймера от старшего байта к младшему соответственно. Далее определяется формат сообщения, и полученные данные записываются в регистрах:

- командное слово – в BSICMD1_1 и BSICMD1_0;
- второе командное слово (только для форматов 3 и 8) – в BSICMD2_1 и BSICMD2_0;
- данные – в BSIDATAx_1 и BSIDATAx_0 («x» изменяется от 1 до 32 и слова сохраняются в том же порядке);
- ответное слово – в регистры BSISW1_1 и BSISW1_0;
- второе ответное слово (только для формата 3) – в регистры BSISW2_1 и BSISW2_0.

Если ожидаемое слово принято с ошибкой, то в соответствующие регистры будут записаны значения FFh, а в регистре BSIINTSTAT установятся флаги ошибок.

Следует помнить, что монитор шины отличает командное слово от ответного по десятому принятому биту, поэтому в командном слове десятый бит должен быть равен единице, а в ответном – нулю.

17 Система прерываний

Микроконтроллер содержит расширенный программно-аппаратный контроллер прерываний с четырьмя уровнями приоритета. Каждый источник прерываний может отдельно программироваться на один из четырех уровней путем установки соответствующих битов в регистрах IP и IPH. На каждом уровне происходит аппаратное ранжирование прерываний.

Микроконтроллер имеет семь векторов прерываний (указаны в порядке уменьшения приоритета):

- внешнего прерывания INT0;
- прерывания таймера T0;
- внешнего прерывания INT1;
- прерывания таймера T1;
- прерывания последовательного порта (SP);
- прерывания таймера T2;
- прерывания блока PCA.

Все флаги прерываний находятся в регистрах модулей/блоков, которым они соответствуют.

На рисунке 17.1, а) на примере прерывания INT0 показан механизм попадания этого прерывания на один из четырех уровней, в зависимости от комбинации битов PX0H (старший) и PX0 (младший) регистров IPH и IP соответственно. Аналогично программно можно задать уровень для каждого из семи векторов прерывания. Четвертый уровень имеет наивысший приоритет, первый уровень – низший. Поэтому прерывания, например, третьего уровня всегда имеют приоритет над прерываниями первого уровня.

Поскольку на один уровень приоритета могут попадать одновременно более одного вектора прерывания, в пределах каждого уровня организовано аппаратное ранжирование. Так, если на один из уровней одновременно попадут два вектора прерываний, например, от таймера T0 и таймера T2, то приоритет будет у T0, поскольку оно имеет аппаратный приоритет над прерыванием от таймера T2. Распределение приоритетов прерываний в пределах одного уровня показано на рисунке 17.1, б). Все четыре уровня имеют идентичную структуру.

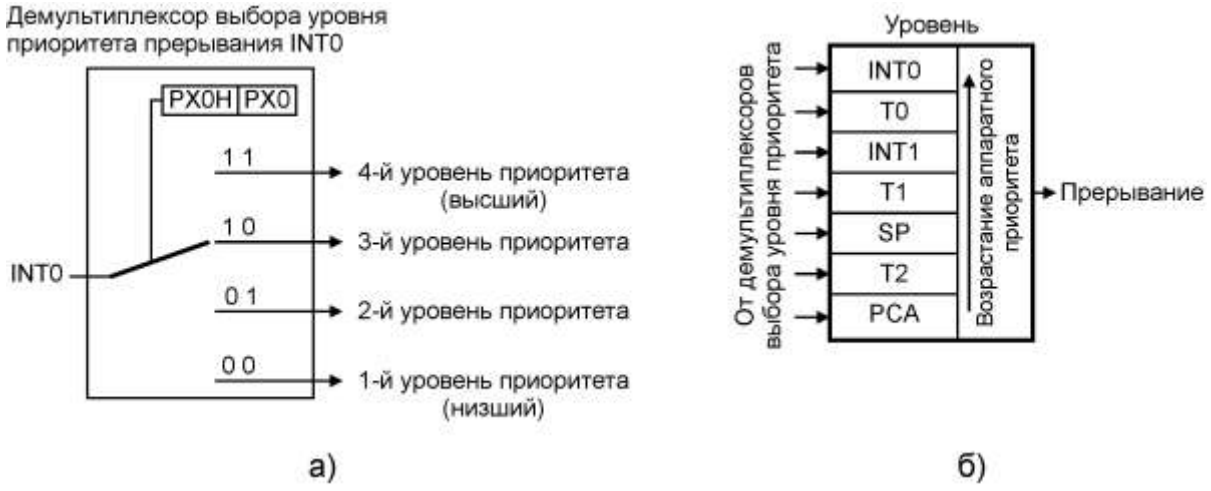


Рисунок 17.1 – Механизмы ранжирования векторов прерываний

Прерывание от каждого источника может быть индивидуально разрешено или запрещено соответствующим ему битом регистра IE. Запретом всех прерываний управляет бит EA регистра IE.

Прерывание с низким приоритетом может быть прервано прерыванием с более высоким приоритетом. Адреса векторов прерываний приведены в таблице 17.1

Таблица 17.1 – Адреса векторов прерываний

Прерывание	Источник/ флаг прерывания	Адрес вектора
Системный сброс	RST или POR, или BOD	0000h
Внешнее прерывание INT0	IE0	0003h
Переполнение счетчика таймера T0	T0F	000Bh
Внешнее прерывание INT1, прерывание ГОСТ 52070–2003 и LIN	IE1, BSIINT, LININT	0013h
Переполнение счетчика таймера T1	T1F	001Bh
Прерывание последовательных портов UART0, UART1, SPI0, SPI1 и I2C	RI, TI, SPIF, INT	0023h
Переполнение счетчика таймера T2	T2F	002Bh
Прерывание блока PCA	CF	0033h
Примечание – Примененные обозначения адресов векторов прерываний: RST – сброс микроконтроллера; POR – «холодный» сброс, вызванный снижением напряжения питания; BOD – «горячий» сброс, инициируемый устройством обнаружения снижения уровня мощности. RST и POR влияют на флаг POF регистра PCON, который вызывает прерывание. BOD не оказывает влияния на флаг POF.		

Внешние прерывания INT0 и INT1 могут быть вызваны как низким уровнем сигналов, так и отрицательными перепадами уровней сигналов на входах INT0# и INT1# микроконтроллера, в зависимости от состояния управляющих битов IT0 и IT1 регистра TCON. Обнаружение заданного состояния на входе INT0#/INT1# вызывает установку флага IE0/IE1, который инициируют вызов соответствующей подпрограммы обработки прерывания. Сброс флага выполняется аппаратно только в том случае, если прерывание было вызвано фронтом отрицательного перепада сигнала. В противном случае сбросом флага должна управлять соответствующая подпрограмма.

Флаги T0F и T1F запросов прерываний от таймеров T0 и T1 сбрасываются автоматически при передаче управления подпрограмме обработки прерываний.

Прерывание таймера T2 формируется логическим «ИЛИ» флагов T2F и EXF2 регистра T2CON. Подпрограмма обслуживания прерывания должна самостоятельно определять, какой из этих флагов установлен. Оба этих флага сбрасываются только программно.

Прерывание последовательного порта формируется логическим «ИЛИ» состояний флагов прерываний (в скобках указан модуль/блок, которому принадлежит флаг):

- RI, TI (UART0, UART1);
- SPIF (SPI0, SPI1);
- INT (I2C).

При возникновении прерывания последовательного порта источник прерывания должен определяться программно. Флаги запросов прерываний от различных последовательных портов устанавливаются аппаратно, но сбрасываются только программным путем. Прерывание последовательного порта может быть сгенерировано не

только аппаратно, но и программно, поскольку флаги всех последовательных модулей/блоков программно доступны

Флаги прерываний опрашиваются в каждом машинном цикле. Приоритетность прерываний определяется в следующем машинном цикле. Аппаратный вызов (LCALL) соответствующей подпрограммы обработки прерывания происходит только в том случае, если он не заблокирован одним из следующих условий:

- в текущий момент обслуживается запрос прерывания равного или более высокого приоритета;
- текущий машинный цикл не является последним в цикле выполняемой команды;
- выполняется команда RETI или любая другая команда, связанная с обращением к регистрам IE и IP.

Если флаг прерывания был установлен, но по одному из перечисленных выше условий не получил обслуживания, а к моменту окончания блокировки уже был сброшен, то запрос прерывания теряется.

По аппаратно сформированному коду команды LCALL содержимое счетчика команд PC помещается в стек, а в счетчик команд загружается адрес вектора соответствующей подпрограммы обработки прерывания. По адресу вектора должна быть расположена команда JMP безусловного перехода к начальному адресу подпрограммы обслуживания прерывания. Подпрограмма обслуживания, в случае необходимости, должна начинаться командой PUSH, которая запишет в стек состояние регистра PSW, аккумулятора, расширителя аккумулятора, указателя данных и др. Подпрограмма должна заканчиваться командой POP для восстановления данных из стека. Подпрограмма обслуживания прерывания должна завершаться командой RETI, по которой в счетчик команд из стека перезагружается адрес возврата в основную программу. Команда RET также как и RETI возвращает управление прерванной основной программе, но при этом не снимает блокировку прерываний, что приводит к необходимости иметь программный механизм анализа окончания процедуры обслуживания данного прерывания.

18 Сброс микроконтроллера

Сброс осуществляется подачей высокого уровня сигнала на вывод RST микроконтроллера и удержанием высокого уровня этого сигнала, по крайней мере, в течение двух машинных циклов (24 периода тактового сигнала с частотой fosc) при работающем генераторе. В ответ на это центральный процессор вырабатывает внутренний сигнал сброса.

Внешний сигнал сброса асинхронен по отношении к внутреннему тактовому сигналу, и вывод RST опрашивается каждый машинный цикл. До тех пор, пока на нем удерживается высокий уровень сигнала, выводы ALE и PSEN# превращаются во входы, и потенциал на них медленно возрастает.

После появления низкого уровня сигнала на выводе RST, сброс микроконтроллера заканчивается. В течение двух машинных циклов от момента окончания сброса синхронизируются сигналы ALE и PSEN#. Следует помнить, что подача сигнала низкого уровня на выводы ALE и PSEN# в момент сброса может перевести микроконтроллер в неопределенное состояние.

Во время сброса, во все регистры области SFR записывается исходная информация. Сброс не оказывает влияния на состояние ячеек внутреннего ОЗУ, однако, следует учитывать, что их состояние после включения питающего напряжения не определено.

Микроконтроллер имеет в своем составе схему сброса при подаче питающего напряжения. Для обеспечения ее устойчивой работы рекомендуется подключение вывода RST к положительному полюсу Ucc источника питания 3,3 В через конденсатор емкостью порядка 1 мкФ. Кроме того, должна быть исключена возможность попадания помех и выбросов на вывод RST. Пока не запустится внутренний генератор и не выполнится описанный выше алгоритм сброса, выводы портов микроконтроллера будут находиться в неопределенном состоянии.

Состояние выводов микроконтроллера во время и после сброса представлено в таблице 18.1.

Таблица 18.1 – Состояние выводов микроконтроллера во время и после сброса

<table>
<tr><td rowspan="2">Выводы</td><td rowspan="2">Номер</td><td rowspan="2">Состояние во время сброса</td><td colspan="2">Состояние после сброса</td></tr>
<tr><td>EA#=0</td><td>EA#=1</td></tr>
<tr><td>P0.0 – P0.7</td><td>5, 4, 3, 1, 64, 62, 61, 60</td><td>Выключено (высокий импеданс)</td><td>Адрес/данные</td><td>Выходы с открытым стоком</td></tr>
<tr><td>P1.0 – P1.7</td><td>9, 10, 11, 12, 13, 15, 18, 19</td><td>pull-up (I_{IL1}) [1)]</td><td colspan="2">pull-up (I_{IL1})</td></tr>
<tr><td>P2.0 – P2.7</td><td>41, 42, 45 ,46, 48, 49, 51, 52</td><td>pull-up (I_{IL1})</td><td>Активные уровни ($U_{OL,}$ U_{OH})</td><td>pull-up (I_{IL1})</td></tr>
<tr><td>P3.0 – P3.7</td><td>22, 26, 27, 29, 30, 32, 33, 35</td><td>pull-up (I_{IL1})</td><td colspan="2">pull-up (I_{IL1})</td></tr>
<tr><td>P4.0 – P4.7</td><td>17, 2, 8, 14, 50, 53, 59, 63</td><td>pull-up (I_{IL1})</td><td colspan="2">pull-up (I_{IL1})</td></tr>
<tr><td>P5.0 – P5.7</td><td>47, 44, 40 ,34, 31, 28, 23, 20</td><td>pull-up (I_{IL1})</td><td colspan="2">pull-up (I_{IL1})</td></tr>
<tr><td>PSEN#</td><td>54</td><td>pull-up (I_{OZL}) [2)]</td><td colspan="2">Активные уровни ($U_{OL,}$ U_{OH})</td></tr>
<tr><td>ALE</td><td>55</td><td>pull-up (I_{OZL})</td><td colspan="2">Активные уровни ($U_{OL,}$ U_{OH})</td></tr>
<tr><td colspan="5">[1)] См. параметр 5 в таблице 1.2.
[2)] См. параметр 8 в таблице 1.2.</td></tr>
</table>

19 Режимы работы микроконтроллера с пониженным энергопотреблением

Энергопотребление является одним из основных параметров микроконтроллера. Микроконтроллер 1882ВМ1Т имеет два режима работы с пониженным энергопотреблением – режим IDLE (холостой ход) и режим PowerDown (отключение). Управление режимами осуществляется с помощью регистра PCON.

Режим IDLE

Включается программно установкой бита IDL регистр PCON. В режиме IDLE центральный процессор переходит в спящее состояние, но при этом все имеющиеся периферийные устройства остаются активными. Содержимое встроенного ОЗУ и всех регистров специальных функций остается неизменным. Допускается программное отключение режима мониторинга ОЗУ.

Режим IDLE может быть прерван любым прерыванием или сбросом. При этом обычно возобновляет исполнение программы с того места, с которого микроконтроллер перешел в режим IDLE. В случае, когда режим IDLE заканчивается сбросом, команда, следующая за командой, вызывающей режим IDLE, не должна вести запись в порт или во внешнюю память. Состояние выводов во время режим IDLE указано в таблице 19.1.

Таблица 19.1 – Состояние выводов во время режимов IDLE и PowerDown

<table>
<tr><td rowspan="2">Режим</td><td rowspan="2">Память программ</td><td colspan="2">Состояние выводов</td><td colspan="2">Порты</td></tr>
<tr><td>ALE</td><td>PSEN#</td><td>Порт 0</td><td>Остальные порты</td></tr>
<tr><td rowspan="2">IDLE</td><td>Внутренняя</td><td rowspan="2">1</td><td rowspan="2">1</td><td>Данные</td><td rowspan="4">Данные</td></tr>
<tr><td>Внешняя</td><td>«Плавающее» состояние</td></tr>
<tr><td rowspan="2">PowerDown</td><td>Внутренняя</td><td rowspan="2">0</td><td rowspan="2">0</td><td>Данные</td></tr>
<tr><td>Внешняя</td><td>«Плавающее» состояние</td></tr>
</table>

Режим PowerDown

Включается программно установкой бита PD регистр PCON. В режиме PowerDown останавливается тактовый генератор, прекращается тактирование центрального процессора, последовательного порта, таймеров/счетчиков и схемы прерываний. Встроенное ОЗУ и регистры специальных функций сохраняют свои значения. Состояние выводов во время режима PowerDown указано в таблице 19.1

Выход из режима PowerDown осуществляется аппаратным сбросом или внешним прерыванием.

Чтобы обеспечить правильный выход из режима, сброс или внешнее прерывание не должны подаваться раньше, чем восстановится нормальный уровень рабочего напряжения, и должны удерживаться достаточно долго, чтобы генератор перестартовал и стабилизировался (около 10 мс).

Сброс микроконтроллера переводит регистры области SFR в их исходное состояние. Содержимое встроенного ОЗУ остается без изменений.

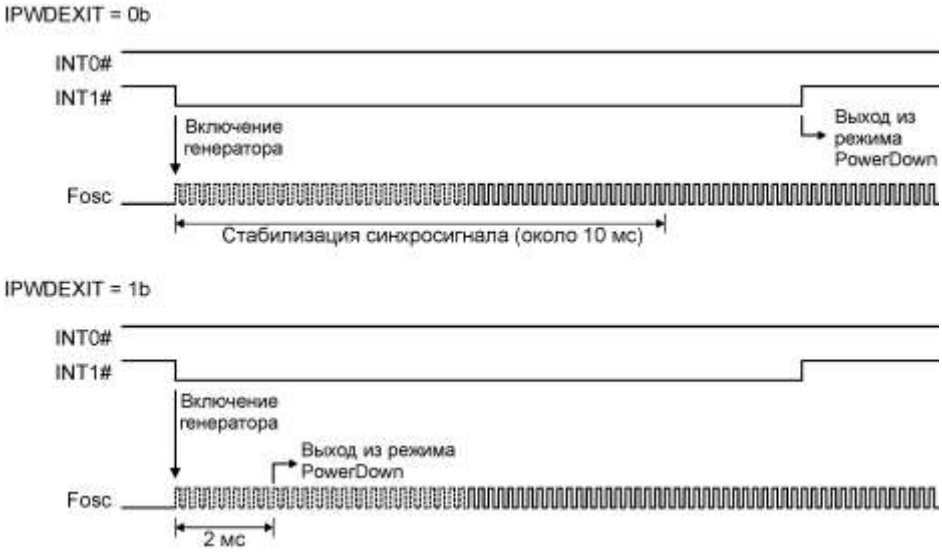


Рисунок 19.1 – Выход из режима PowerDown

Использование внешних прерываний для выхода из режима позволяет сохранять состояние регистров области SFR (кроме бита PD регистра PCON) и содержимое встроенного ОЗУ.

Появление низкого уровня сигнала на любом из входов INT0# (вывод P3.2) или INT1# (вывод P3.3) вызовет запуск внутреннего генератора (см. рисунок 19.1). Низкий уровень сигнала должен удерживаться в течение времени, которое требуется для стабилизации формирования внутреннего синхросигнала (около 10 мс). После этого, если не установлен бит IPWDEXIT регистра AUXR, микроконтроллер будет ожидать появление фронта положительного перепада сигнала, по которому будет осуществлен выход из режима PowerDown. Если бит IPWDEXIT установлен, то выход из режима автоматически осуществляется через 2 мс после появления низкого уровня сигнала на любом из входов INT0# и INT1# (см. рисунок 19.1).

Примечание – Режим PowerDown не должен повторно включаться, как минимум, в течение 4 мс.

Обслуживание прерывания, вызвавшего выход из режима PowerDown, должно заканчиваться командой RETI. После чего выполнение прерванной программы начнется с команды, которая следует за командой, включившей режим PowerDown.

20 Программирование микроконтроллера

Микроконтроллер имеет 32 Кбайта памяти программ типа Flash и 4 Кбайта памяти данных типа EEPROM, которые поддерживают параллельное и последовательное программирование постранично и побайтно без предварительного стирания.

Доступное адресное пространство памяти программ 0000h – 7FFFh, памяти данных – 0000h – 0FFFh. Начальное состояние ячеек памяти на момент поставки – FFh.

Параллельное и последовательное программирование осуществляются с помощью программатора, при этом последовательное программирование позволяет перепрограммировать устройство внутри пользовательской системы (посредством интрефейса SPI).

Программирование памяти программ и памяти данных независимое. Память программ допускает как постраничное программирование по 64 байта, так побайтное при одном и том же времени записи – 5 мс.

Память данных допускает постраничное программирование по 32 байта и побайтное при одном и том же времени записи – 5 мс.

Для защиты информации, хранящейся в памяти микроконтроллера от случайного или преднамеренного считывания или стирания, предусмотрены специальные биты защиты (Fuse-биты и Lock-биты).

20.1 Параллельное программирование

Схема включения микроконтроллера при параллельном программировании показана на рисунке 20.1

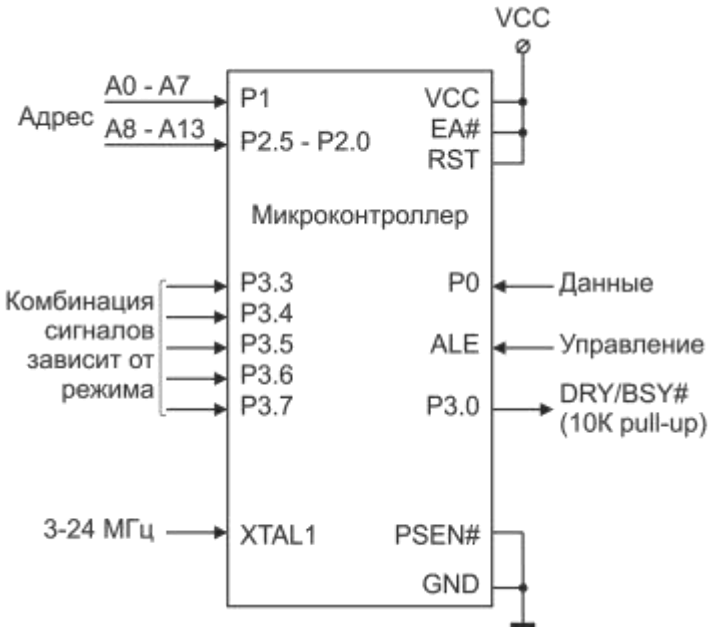


Рисунок 20.1 – Схема включения при параллельном программировании

Программирование микроконтроллера и чтение записанных данных допускается только в случае, если это не запрещено битами защиты.

Управление программированием осуществляется посредством выводов P3.7 – P3.3 порта 3 и вывода ALE. Адреса передаются через порт P1 и выводы P2.5 – P2.0 порта P2, данные – через порт P0.

Для индикации программирования (программируется байт) используется вывод RDY/BSY# (P3.0) с подтягивающим резистором (pull-up). Вывод переходит в состояние нуля после перехода ALE в состояние единицы. По окончании записи байта P3.0 снова переходит состояние единицы.

Таблица 20.1 – Режимы программирования

Режим		RST	PSEN#	ALE	EA#	Выводы порта 3					Данные	Адрес
						7	6	5	4	3		
Код режима параллельного программирования		H	H	H	–	–	–	–	–	–	–	–
Стирание кристалла		H	L	1,0 мкс	12В	L	L	H	L	H	–	–
Запись страницы памяти программ		H	L	1,0 мкс	12В	H	H	H	H	L	DI	ADDR
Чтение страницы памяти программ		H	L	H	12В	H	H	H	L	L	DO	ADDR
Запись страницы памяти данных		H	L	1,0 мкс	12В	H	H	L	H	L	DI	ADDR
Чтение страницы памяти данных		H	L	H	12В	H	H	L	L	L	DO	ADDR
Запись lock-битов	L1	H	L	1,0 мкс	12В	L	H	H	L	H	D0	–
	L2										D1	–
	L3										D2	–
Чтение lock-битов	L1	H	L	H	12В	L	L	H	H	H	D0	–
	L2										D1	–
	L3										D2	–
Запись страницы пользователя		H	L	1,0 мкс	12В	H	H	H	L	H	DI	0–3Fh
Чтение страницы пользователя		H	L	H	12В	H	L	H	L	L	DO	0–3Fh
Чтение сигнатуры		H	L	H	12В	L	L	H	L	L	DO	0–3Fh
Запись fuse-битов	F1	H	L	1,0 мкс	12В	H	L	H	H	L	D0	–
	F2										D1	–
	F3										D2	–
	F4										D3	–
Чтение fuse-битов	F1	H	L	H	12В	H	L	H	H	H	D0	–
	F2										D1	–
	F3										D2	–
	F4										D3	–

Примечание – Принятые обозначения:
ADDR – адрес байта данных;
DI – записываемый байт данных;
DO – считываемый байт данных;
D3, D2, D1, D0 – младшие биты байта данных передающиеся через выводы P0.3, P0.2 P0.1 и P0.0 соответственно.
H – высокий логический уровень сигнала на выводе;
L – низкий логический уровень сигнала на выводе.

Последовательность действий при параллельном постраничном программировании.

1 Подключить питание и общий вывод к выводам #VCC и GND соответственно. Установить единицу на выводе RST. Подключить источник синхронизации с частотой от 3 до 24 МГц к выводу XTAL1 и выждать не менее 10 мс.

2 Установить ноль на выводе PSEN#, на выводах ALE, EA# и всех других – единицу.

3 Увеличить напряжение на выводах EA#, #VCC до 12 В для разрешения программирования. Подключить к выводу P3.0 подтягивающий резистор (10 кОм).

4 Подать комбинацию сигналов на выводы P3.7 – P3.3 для выбора одного из режимов программирования (см. таблицу 20.1).

5 Подать байт адреса на выводы P1.7 – P1.0 и P2.5 – P2.0 и байт данных на выводы P0.7 – P0.0.

6 Подать импульс ALE для инициации записи.

7 Повторить шаги 5-6, изменяя адрес и данные для загрузки одной страницы (64 байта или 32 байта). Интервал между загружаемыми байтами не должен превышать 150 мкс.

8 После загрузки последнего байта страницы необходимо ожидать 5 мс (время записи страницы) или контролировать состояние вывода RDY/BSY# – пока он не переключится в единицу.

9 Для верификации последнего байта только что записанной страницы подать ноль на вывод P3.4 и прочитать записанные данные на выводах P0.7 – P0.0.

10 Завершение работы. Перевести в состояние высокого импеданса шины адреса и данных, отключить подтягивающий резистор вывода P3.0, установить ноль на выводе XTAL1, после чего установить ноль на выводах RST и EA#. Отключить питание с вывода #VCC.

При считывании сигнатур микроконтроллера следует выполнить те же операции, что и при чтении адресов 30h и 31h, за исключением того, что выводы P3.6 и P3.7 должны быть в состоянии нуля. Ответные значения для 30h и 31h – 1E и 73, соответственно.

20.2 Последовательное программирование

Схема включения микроконтроллера при последовательном программировании показана на рисунке 20.2.

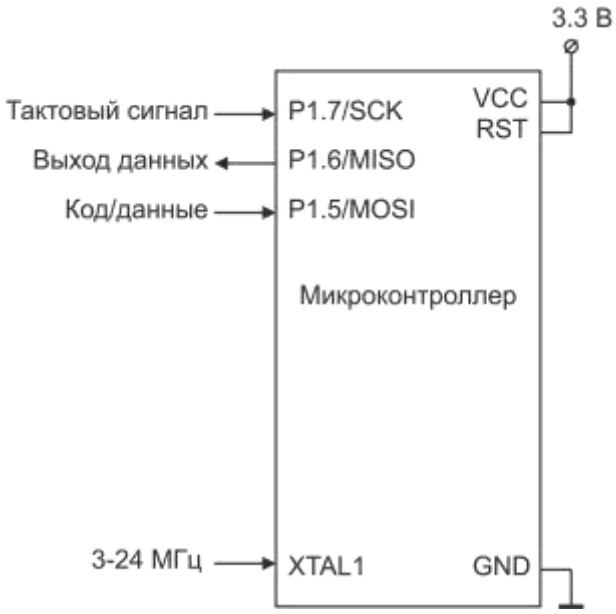


Рисунок 20.2 – Схема включения при последовательном программировании

Программирование микроконтроллера и чтение записанных данных допускается только в случае, если это не запрещено битами защиты.

Программирование осуществляется посредством выводов P1.7 – P1.3 порта 1 по протоколу интерфейса SPI. Сигнал тактирования передачи SCK подается на вывод P1.7, адреса и данные для записи подаются на вывод P1.5 (MOSI), а считываемые данные передаются через вывод P1.6 (MISO). В отсутствие передачи на линии SCK удерживается

низкий уровень сигнала. Установка данных происходит по переднему фронту сигнала SCK, выборка – по заднему (см. рисунок 12.3 для CPOL = 0 и CPHA = 1).

Программатор и микроконтроллер могут иметь независимые источники питания, но общие выводы должны быть объединены. Длина соединительных проводов не должна превышать 30 см.

Последовательность действий при последовательном постраничном программировании.

1 Подключить питание и общий вывод к выводам #VCC и GND соответственно. Установить единицу на выводе RST. Подключить источник синхронизации с частотой от 3 до 24 МГц к выводу XTAL1 и выждать не менее 10 мс (при этом на выводе P1.7 должен удерживаться ноль).

2. Начать программирование с передачи четырех байт разрешения программирования ACh-53h-00h-00h (см. таблицу 20.2). Частота тактового сигнала SCK, подаваемого на вывод P1.7, не должна превышать 16 кГц.

3 Последовательно передать код команды записи, начальный адрес страницы и 64/32 байта страницы данных. После загрузки последнего байта страницы необходимо ожидать 5 мс (время записи страницы).

4 Для верификации записанных данных следует передать код команды чтения и адрес, после чего принять запрашиваемые данные.

5 Завершение программирования. Чтобы выйти из режима программирования и перевести микроконтроллер в нормальный режим работы, достаточно подать ноль на вывод RST.

6 Завершение работы. Установить ноль на выводе XTAL1, после чего установить ноль на выводе RST. Отключить питание с вывода #VCC.

Таблица 20.2 – Команды последовательного программирования

Операция	Формат команды операции
1	2
Разрешение программирования	1010 1100 | 0 1 0 1 0 0 1 1 | X X X X X X X X | X X X X X X X X
Стирание кристалла	1010 1100 | 1 0 0 X X X X X | X X X X X X X X | X X X X X X X X
Запись байта в память программ	0100 0000 | X X A13 A12 A11 A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0
Чтение байта из памяти программ	0010 0000 | X X A13 A12 A11 A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0
Запись страницы памяти программ	0101 0000 | X X A13 A12 A11 A10 A9 A8 | A7 A6 0 0 0 0 0 0 | Байт 0, ..., байт 63
Чтение страницы памяти программ	0011 0000 | X X A13 A12 A11 A10 A9 A8 | A7 A6 0 0 0 0 0 0 | Байт 0, ..., байт 63
Запись байта в память данных	1100 0000 | X X X X X A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0

Окончание таблицы 20.2

1	2
Чтение байта из памяти данных	1010 0000 | X X X X X A10 A9 A8 | A7 A6 A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0
Запись страницы памяти данных	1101 0000 | X X X X X A10 A9 A8 | A7 A6 0 0 0 0 0 0 | Байт 0, ..., байт 31
Чтение страницы памяти данных	1011 0000 | X X X X X A10 A9 A8 | A7 A6 0 0 0 0 0 0 | Байт 0, ..., байт 31
Запись fuse-битов	1010 1100 | 0 0 0 1 F4 F3 F2 F1 | X X X X X X X X | X X X X X X X X
Чтение fuse-битов	0010 0001 | X X X X X X X X | X X X X X X X X | X X X X F4 F3 F2 F1
Запись lock-битов	0010 0100 | 1 1 1 0 0 L3 L2 L1 | X X X X X X X X | X X X X X X X X
Чтение lock-битов	0010 0001 | X X X X X X X X | X X X X X X X X | X X X X X L3 L2 L1
Запись байта в страницу пользователя	0100 0010 | X X X X X X X X | X X A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0
Чтение байта из страницы пользователя	0010 0010 | X X X X X X X X | X X A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0
Запись страницы пользователя	0101 0010 | X X X X X X X X | X X X X X X X X | Байт 0, ..., байт 63
Чтение страницы пользователя	0011 0010 | X X X X X X X X | X X X X X X X X | Байт 0, ..., байт 63
Чтение сигнатуры	0010 1000 | X X X X X X X X | X X A5 A4 A3 A2 A1 A0 | D7 D6 D5 D4 D3 D2 D1 D0

Биты защиты

Биты защиты fuse-биты и lock-биты – это специальные биты, доступ к которым возможен только в режиме программирования микроконтроллера. Запись и чтение fuse-битов и lock-битов производится независимо, посредством соответствующих кодов (см. таблицы 20.1 и 20.2).

Fuse-биты используются для предотвращения случайного стирания страницы пользователя (64 байта) и запрета последовательного программирования. Кроме этого,

посредством fuse-битов можно выбирать внешний источник синхронизации микроконтроллера (кварцевый резонатор или генератор частоты) и включать/выключать делитель на частоты синхронизации.

Все четыре fuse-бита программируются одновременно, но при этом состояние каждого из них задается независимо. Биты, переданные нулями, будут считаться установленными, единицами – неустановленными. При считывании четырех битов на позициях установленных бит находятся нули. Состояние битов на момент поставки микросхемы всегда 1111b.

Примечание – Fuse-биты всегда доступны для перепрограммирования. Исключение составляет только бит F1, который после установки может быть сброшен только при параллельном программировании.

Назначения fuse-битов приведены в таблице 20.3.

Таблица 20.3 – Назначения fuse-битов защиты

Fuse-бит	Состояние бита при программировании/ считывании	Назначение
F1	0	Запрет последовательного программирования. Бит может быть установлен как при последовательном, так и при параллельном программировании, но сброшен только при параллельном
	1	Разрешение последовательного программирования
F2	0	Делитель на два входной частоты синхронизации микроконтроллера выключен
	1	Делитель на два входной частоты включен
F3	0	Запрет программирования и стирания страницы пользователя. Доступно только чтение
	1	Страница пользователя доступна для программирования
F4	0	Источником внешней синхронизации является внешний генератор
	1	Источником внешней синхронизации является кварцевый резонатор

Lock-биты используются для запрещения доступа к памяти программ и памяти данных как частичного (запрет записи), так и полного (запрет записи и считывания).

Все lock-биты программируются одновременно, но в отличие от fuse-битов допустимыми являются только три комбинации их состояний. Запрограммированные lock-биты доступны для повторного программирования только после выполнения команды «Стирание кристалла», которая инициирует очистку памяти программ, памяти данных и сброс lock-битов в состояние 111b (начальное состояние при поставке микросхемы). Назначения lock-битов приведены в таблице 20.4.

Примечание – Доступ к странице пользователя (запись, стирание) возможен только, если не установлен fuse-бит F3.

Таблица 20.4 – Назначения lock-битов защиты

Комбинация битов			Влияние комбинации битов на доступ к памяти программ и памяти данных
L3	L2	L1	
1	1	1	Разрешен полный доступ – запись, чтение, стирание
1	1	0	Запись запрещена
1	0	0	Чтение и запись запрещены
0	0	0	Чтение и запись запрещены. Запрещено обращение к внешней памяти

21 Система команд микроконтроллера

Система команд МК включает в свой состав 111 основных команд. Длина команд составляет 1, 2 или 3 байта. 94 % команд – одно- или двухбайтовые. Все команды выполняются за один или два машинных цикла (0,5 или 1,0 мкс при тактовой частоте 24 МГц соответственно), исключение составляют лишь команды умножения и деления, которые выполняются за четыре машинных цикла (2,0 мкс).

Для получения полной информации о системе команд микроконтроллера следует обратиться к приложению Б.

Заключение

В руководстве КФДЛ.431281.035 приведено описание архитектуры, функционального построения, системы команд и особенностей применения микроконтроллера 1882ВМ1Т. Микроконтроллер имеет в своем составе 8-разрядное микропроцессорное ядро MCS-51, 32 Кбайт Flash памяти программ, 4 Кбайт EEPROM данных, порты интерфейсов UART, SPI, I2C, LIN, ГОСТ Р 52070-2003, блок кодирования по ГОСТ 28147-89, массив таймеров-счетчиков (PCA), блок 16-разрядной арифметики MDU.

Настоящее руководство может служить практическим пособием по применению микроконтроллеров 1882ВМ1Т для разработчиков систем на их основе.

Приложение А
(обязательное)

Список регистров микроконтроллера

В таблицах А.1 – А.3 представлены списки регистров микроконтроллера. Далее в таблицах А.4 – А.81 представлены форматы регистров и информация о функциональном назначении их бит.

Таблица А.1 – Регистры SFR нулевого окна (WSR = 00h)

Адрес	Название регистра	Назначение	Сброс
1	2	3	4
80h	P0*	Регистр порта 0	FFh
81h	SP	Указатель стека	07h
82h	DP0L	Младший байт регистра основного указателя данных DPTR0	00h
83h	DP0H	Старший байт регистра основного указателя данных DPTR0	00h
84h	DP1L	Младший байт регистра основного указателя данных DPTR1	00h
85h	DP1H	Старший байт регистра основного указателя данных DPTR1	00h
86h	SPDR0	Регистр данных интерфейса SPI0	00h
87h	PCON	Регистр управления энергопотреблением	00x00000b
88h	TCON*	Регистр управления и состояния таймеров T0 и T1	00h
89h	TMOD	Регистр режимов таймеров T0 и T1	00h
8Ah	TL0	Младший байт регистра таймера T0	00h
8Bh	TL1	Младший байт регистра таймера T1	00h
8Ch	TH0	Старший байт регистра таймера T0	00h
8Dh	TH1	Старший байт регистра таймера T1	00h
8Eh	AUXR	Управление режимом ALE и выходом из прерывания	xxxxxxx0b
8Fh	CLKREG	Регистр управления частотой внутренней синхронизации	x1111100b
90h	P1*	Регистр порта 1	FFh
91h	P1SEL	Регистр управления альтернативными функциями порта 1	FFh
92h – 95h	–	Зарезервировано	–
96h	EECON	Регистр управления памятью данных EEPROM	02h
97h	–	Зарезервировано	–
98h	SCON0*	Регистр управления и статуса порта UART0	00h
99h	SBUF0	Регистр данных порта UART0	xxh
9Ah	SCON1	Регистр управления и статуса порта UART1	00h
9Bh	SBUF1	Регистр данных порта UART1	xxh
9Ch, 9Dh	–	Зарезервировано	–
9Eh	SPDR1	Регистр данных интерфейса SPI1	00h
9Fh	FASTM	Регистр управления ускоренным режимом работы ядра	00h

Продолжение таблицы А.1

1	2	3	4
A0h	P2*	Регистр порта 2	00h
A1h – A5h	–	Зарезервировано	–
A6h	WDTRST	Регистр аппаратного режима сторожевого таймера	xxh
A7h	WDTCON	Регистр управления сторожевого таймера	00h
A8h	IE*	Регистр разрешения прерываний	0x000000b
A9h	SADDR0	Регистр заданного адреса порта UART0	00h
AAh	SPSR0	Регистр состояния интерфейса SPI0	00xxxxxxb
ABh	SADDR1	Регистр заданного адреса порта UART1	00h
ACh	SPSR1	Регистр состояния интерфейса SPI1	00xxxxxxb
ADh – AFh	–	Зарезервировано	–
B0h	P3*	Регистр порта 3	FFh
B1h– B6h	–	Зарезервировано	–
B7h	IPH	Регистр высоких приоритетов прерываний	xx000000b
B8h	IP*	Регистр приоритетов прерываний	xx000000b
B9h	SADEN0	Регистр маски адреса порта UART0	00h
Bah	–	Зарезервировано	–
BBh	SADEN1	Регистр маски адреса порта UART1	00h
BCh	–	Зарезервировано	–
BDh	–	Зарезервировано	–
BEh	RAMST0	Младший байт регистра счетчика ошибок ОЗУ	xxh
BFh	RAMST1	Старший байт регистра счетчика ошибок ОЗУ	xxh
C0h	P4*	Регистр порта 4	FFh
C1h	MDUCON	Регистр управления блока MDU	00h
C2h	MD0/MR0	Регистр данных 0/результата 0	00h
C3h	MD1/MR1	Регистр данных 1/результата 1	00h
C4h	MD2/MR2	Регистр данных 2/результата 2	00h
C5h	MD3/MR3	Регистр данных 3/результата 3	00h
C6h	MD4/MR4	Регистр данных 4/результата 4	00h
C7h	MD5/MR5	Регистр данных 5/результата 5	00h
C8h	T2CON	Регистр управления таймера T2	00h
C9h	T2MOD	Регистр режима таймера T2	xxxxxx00b
CAh	RCAP2L	Младший байт регистра захвата/перезагрузки таймера T2	00h
CBh	RCAP2H	Старший байт регистра захвата/перезагрузки таймера T2	00h
CCh	TL2	Младший байт регистра таймера T2	00h
CDh	TH2	Старший байт регистра таймера T2	00h
CEh – CFh	–	Зарезервировано	–
D0h	PSW*	Регистр слова состояния процессора	00h
D1h – D4h	–	Зарезервировано	–
D5h	SPCR0	Регистр управления интерфейса SPI0	04h
D6h	SPCR1	Регистр управления интерфейса SPI1	04h
D7h	–	Зарезервировано	00h

Продолжение таблицы А.1

1	2	3	4
D8h	CCON*	Регистр управления таймера/счетчика блока PCA	00h
D9h	CMOD	Регистр режима таймера/счетчика блока PCA	00h
DAh	CCAPM0	Регистр режима работы модуля захвата/сравнения канала 0 блока PCA	00h
DBh	CCAPM1	Регистр режима работы модуля захвата/сравнения канала 1 блока PCA	00h
DCh	CCAPM2	Регистр режима работы модуля захвата/сравнения канала 2 блока PCA	00h
DDh	CCAPM3	Регистр режима работы модуля захвата/сравнения канала 3 блока PCA	00h
DEh	CCAPM4	Регистр режима работы модуля захвата/сравнения канала 4 блока PCA	00h
DFh	–	Зарезервировано	–
E0h	ACC*	Аккумулятор	00h
E1h	SMBSDA	Регистр сдвига данных интерфейса I2C	xxh
E2h	SMBST	Регистр статуса интерфейса I2C	00h
E3h	SMBCST	Регистр управления и статуса интерфейса I2C	00h
E4h	SMBCTRL1	Регистр 1 управления интерфейса I2C	00h
E5h	SMBADDR	Регистр собственного адреса интерфейса I2C	00h
E6h	SMBCTRL2	Регистр 2 управления интерфейса I2C	00h
E7h	SMBTOPR	Регистр прескалера времени ожидания интерфейса I2C	00h
E8h	P5*	Регистр порта 5	FFh
E9h	CL	Младший байт регистра таймера/счетчика блока PCA	00h
EAh	CCAP0L	Младший байт регистра захвата/сравнения канала 0 блока PCA	00h
EBh	CCAP1L	Младший байт регистра захвата/сравнения канала 1 блока PCA	00h
ECh	CCAP2L	Младший байт регистра захвата/сравнения канала 2 блока PCA	00h
EDh	CCAP3L	Младший байт регистра захвата/сравнения канала 3 блока PCA	00h
EEh	CCAP4L	Младший байт регистра захвата/сравнения канала 4 блока PCA	00h
EFh	SMBCTRL3	Регистр 3 управления интерфейса I2C	00h
F0h	B*	Регистр расширителя аккумулятора	00h
F1h – F8h	–	Зарезервировано	–
F9h	CH	Старший байт регистра таймера/счетчика блока PCA	00h
FAh	CCAP0H	Старший байт регистра захвата/сравнения канала 0 блока PCA	00h
FBh	CCAP1H	Старший байт регистра захвата/сравнения канала 1 блока PCA	00h
FCh	CCAP2H	Старший байт регистра захвата/сравнения канала 2 блока PCA	00h
FDh	CCAP3H	Старший байт регистра захвата/сравнения канала 3 блока PCA	00h
FEh	CCAP4H	Старший байт регистра захвата/сравнения канала 4 блока PCA	00h

Окончание таблицы А.1

1	2	3	4
FFh	WSR	Регистр выбора окна области SFR	00h
Примечание – символом «*» отмечены регистры, допускающие адресацию своих отдельных бит при выполнении команд, оперирующих с битами.			

Таблица А.2 – Регистры SFR первого окна (WSR = 01h)

Адрес	Название регистра	Назначение	Сброс
1	2	3	4
80h	CDCON	Регистр управления ГОСТ 28147–89	00h
81h	CDDATNUM	Регистр числа блоков данных ГОСТ 28147–89	00h
82h	CDSTATE	Регистр состояния ГОСТ 28147–89	00h
83h	CDDATA	Регистр данных ГОСТ 28147–89	00h
84h	CDKEY	Регистр ключа ГОСТ 28147–89	00h
85h	CDSBSTN	Регистр замены ГОСТ28147–89	00h
86h	CDUW	Регистр синхропосылки ГОСТ 28147–89	00h
87h	CDAUTH	Регистр имитовставки ГОСТ 28147–89	00h
88h – 8Fh	–	Зарезервировано	–
90h	LINDTX0	Регистр 0 данных для передачи LIN	00h
91h	LINDTX1	Регистр 1 данных для передачи LIN	00h
92h	LINDTX2	Регистр 2 данных для передачи LIN	00h
93h	LINDTX3	Регистр 3 данных для передачи LIN	00h
94h	LINDTX4	Регистр 4 данных для передачи LIN	00h
95h	LINDTX5	Регистр 5 данных для передачи LIN	00h
96h	LINDTX6	Регистр 6 данных для передачи LIN	00h
97h	LINDTX7	Регистр 7 данных для передачи LIN	00h
98h	LINDRX0	Регистр 0 принятых данных LIN	00h
99h	LINDRX1	Регистр 1 принятых данных LIN	00h
9Ah	LINDRX2	Регистр 2 принятых данных LIN	00h
9Bh	LINDRX3	Регистр 3 принятых данных LIN	00h
9Ch	LINDRX4	Регистр 4 принятых данных LIN	00h
9Dh	LINDRX5	Регистр 5 принятых данных LIN	00h
9Eh	LINDRX6	Регистр 6 принятых данных LIN	00h
9Fh	LINDRX7	Регистр 7 принятых данных LIN	00h
A0h	LINCONL	Младший байт регистра управления LIN	01h
A1h	LINCONH	Старший байт регистра управления LIN	08h
A2h	LINSTATL	Младший байт регистра состояния LIN	00h
A3h	LINSTATH	Старший байт регистра состояния LIN	00h
A4h	LINBREAKL	Младший байт регистра настройки длины поля BREAK	9Eh
A5h	LINBREAKH	Старший байт регистра настройки длины поля BREAK	02h
A6h	LINSPEEDL	Младший байт регистра настройки скорости LIN	9Ch
A7h	LINSPEEDH	Старший байт регистра настройки скорости LIN	01h
A8h	LINPIDTX	Регистр передаваемого PID	00h
A9h	LINPIDRX	Регистр принятого PID	00h
AAh	LINCSR	Регистр контрольной суммы LIN	00h
ABh	–	Зарезервировано	–

Окончание таблицы А.2

1	2	3	4
ACh	LININTENL	Младший байт регистра разрешения прерывания LIN	00h
ADh	LININTENH	Старший байт регистра разрешения прерываний LIN	00h
AEh–FEh	–	Зарезервировано	–
FFh	WSR	Регистр выбора окна области SFR	00h

Таблица А.3 – Регистры SFR второго окна (WSR = 02h)

Адрес	Название регистра	Назначение	Сброс
1	2	3	4
80h	BSICONF1	Регистр конфигурации ГОСТ Р 52070–2003	05h
81h	–	Зарезервировано	
82h	BSICONF2	Дополнительный регистр конфигурации ГОСТ Р 52070–2003	00h
83h	–	Зарезервировано	
84h	BSICON	Регистр управления ГОСТ Р 52070–2003	00h
85h	–	Зарезервировано	
86h	BSISW1_0	Младший регистр первого ответного слова	02h
87h	BSISW1_1	Старший регистр первого ответного слова	00h
88h	BSISW2_0	Младший регистр второго ответного слова	00h
89h	BSISW2_1	Старший регистр второго ответного слова	00h
8Ah	BSICMD1_0	Младший регистр первого командного слова	20h
8Bh	BSICMD1_1	Старший регистр первого командного слова	00h
8Ch	BSICMD2_0	Младший регистр второго командного слова	20h
8Dh	BSICMD2_1	Старший регистр второго командного слова	00h
8Eh, 8Fh	–	Зарезервировано	–
90h	BSIDAT1L	Младший регистр 1 слова данных	00h
91h	BSIDAT1H	Старший регистр 1 слова данных	00h
92h	BSIDAT2L	Младший регистр 2 слова данных	00h
93h	BSIDAT2H	Старший регистр 2 слова данных	00h
94h	BSIDAT3L	Младший регистр 3 слова данных	00h
95h	BSIDAT3H	Старший регистр 3 слова данных	00h
96h	BSIDAT4L	Младший регистр 4 слова данных	00h
97h	BSIDAT4H	Старший регистр 4 слова данных	00h
98h	BSIDAT5L	Младший регистр 5 слова данных	00h
99h	BSIDAT5H	Старший регистр 5 слова данных	00h
9Ah	BSIDAT6L	Младший регистр 6 слова данных	00h
9Bh	BSIDAT6H	Старший регистр 6 слова данных	00h
9Ch	BSIDAT7L	Младший регистр 7 слова данных	00h
9Dh	BSIDAT7H	Старший регистр 7 слова данных	00h
9Eh	BSIDAT8L	Младший регистр 8 слова данных	00h
9Fh	BSIDAT8H	Старший регистр 8 слова данных	00h
A0h	BSIDAT9L	Младший регистр 9 слова данных	00h
A1h	BSIDAT9H	Старший регистр 9 слова данных	00h
A2h	BSIDAT10L	Младший регистр 10 слова данных	00h
A3h	BSIDAT10H	Старший регистр 10 слова данных	00h
A4h	BSIDAT11L	Младший регистр 11 слова данных	00h

Продолжение таблицы А.3

1	2	3	4
A5h	BSIDAT11H	Старший регистр 11 слова данных	00h
A6h	BSIDAT12L	Младший регистр 12 слова данных	00h
A7h	BSIDAT12H	Старший регистр 12 слова данных	00h
A8h	BSIDAT13L	Младший регистр 13 слова данных	00h
A9h	BSIDAT13H	Старший регистр 13 слова данных	00h
AAh	BSIDAT14L	Младший регистр 14 слова данных	00h
ABh	BSIDAT14H	Старший регистр 14 слова данных	00h
ACh	BSIDAT15L	Младший регистр 15 слова данных	00h
ADh	BSIDAT15H	Старший регистр 15 слова данных	00h
AEh	BSIDAT16L	Младший регистр 16 слова данных	00h
AFh	BSIDAT16H	Старший регистр 16 слова данных	00h
B0h	BSIDAT17L	Младший регистр 17 слова данных	00h
B1h	BSIDAT17H	Старший регистр 17 слова данных	00h
B2h	BSIDAT18L	Младший регистр 18 слова данных	00h
B3h	BSIDAT18H	Старший регистр 18 слова данных	00h
B4h	BSIDAT19L	Младший регистр 19 слова данных	00h
B5h	BSIDAT19H	Старший регистр 19 слова данных	00h
B6h	BSIDAT20L	Младший регистр 20 слова данных	00h
B7h	BSIDAT20H	Старший регистр 20 слова данных	00h
B8h	BSIDAT21L	Младший регистр 21 слова данных	00h
B9h	BSIDAT21H	Старший регистр 21 слова данных	00h
BAh	BSIDAT22L	Младший регистр 22 слова данных	00h
BBh	BSIDAT22H	Старший регистр 22 слова данных	00h
BCh	BSIDAT23L	Младший регистр 23 слова данных	00h
BDh	BSIDAT23H	Старший регистр 23 слова данных	00h
BEh	BSIDAT24L	Младший регистр 24 слова данных	00h
BFh	BSIDAT24H	Старший регистр 24 слова данных	00h
C0h	BSIDAT25L	Младший регистр 25 слова данных	00h
C1h	BSIDAT25H	Старший регистр 25 слова данных	00h
C2h	BSIDAT26L	Младший регистр 26 слова данных	00h
C3h	BSIDAT26H	Старший регистр 26 слова данных	00h
C4h	BSIDAT27L	Младший регистр 27 слова данных	00h
C5h	BSIDAT27H	Старший регистр 27 слова данных	00h
C6h	BSIDAT28L	Младший регистр 28 слова данных	00h
C7h	BSIDAT28H	Старший регистр 28 слова данных	00h
C8h	BSIDAT29L	Младший регистр 29 слова данных	00h
C9h	BSIDAT29H	Старший регистр 29 слова данных	00h
CAh	BSIDAT30L	Младший регистр 30 слова данных	00h
CBh	BSIDAT30H	Старший регистр 30 слова данных	00h
CCh	BSIDAT31L	Младший регистр 31 слова данных	00h
CDh	BSIDAT31H	Старший регистр 31 слова данных	00h
CEh	BSIDAT32L	Младший регистр 32 слова данных	00h
CFh	BSIDAT32H	Старший регистр 32 слова данных	00h
D0h	BSIINTSTAT	Регистр состояния прерываний ГОСТ Р 52070–2003	00h
D1h	–	Зарезервировано	–
D2h	BSITIMER1	Регистр 0 байта таймера ГОСТ Р 52070–2003	00h
D3h	BSITIMER2	Регистр 1 байта таймера ГОСТ Р 52070–2003	00h

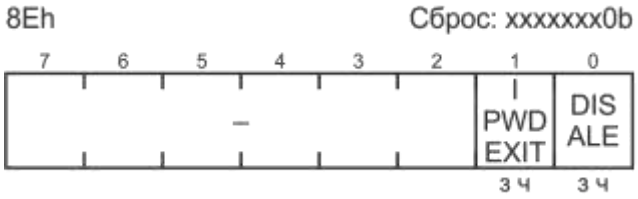
Окончание таблицы А.3

1	2	3	4
D4h	BSITIMER3	Регистр 2 байта таймера ГОСТ Р 52070–2003	00h
D5h	BSITIMER4	Регистр 3 байта таймера ГОСТ Р 52070–2003	00h
D6h-FEh	–	Зарезервировано	–
FFh	WSR	Регистр выбора окна области SFR	00h

Специальные регистры микроконтроллера

Таблица А.4 – Регистр управления режимом ALE и выходом из прерывания

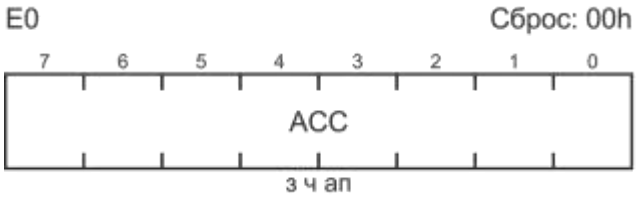
AUXR



Поле	Бит	Описание	
IPWDEXIT	1	Выход из режима пониженного энергопотребления по прерыванию	
		0	Выход по прерыванию осуществляется после интервала времени, отсчитываемого от заднего фронта сигнала прерывания (2 мс)
		1	Обеспечивается выход по прерыванию из режима пониженной мощности (по приходу переднего фронта сигнала прерывания)
DISALE	0	Запрет ALE	
		0	Частота сигнала на выходе ALE равна 1/6 частоты синхронизации (при выполнении команды MOVX один импульс ALE отсутствует)
		1	ALE активен только во время команды MOVX или MOVC
–	7-2	Зарезервировано	

Таблица А.5 – Аккумулятор

ACC



Поле	Бит	Описание
ACC	7–0	Байт данных

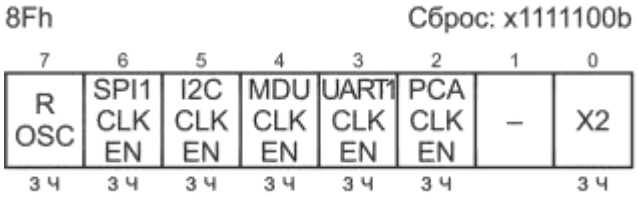
Таблица А.6 – Регистр расширителя аккумулятора

B

Поле	Бит	Описание
ACCE	7–0	Байт данных

Таблица А.7 – Регистр управления частотой внутренней синхронизации

CLKREG



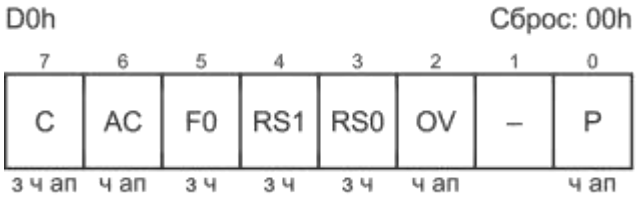
Поле	Бит	Описание	
1	2	3	
ROSC	7	Бит включения внутреннего кольцевого генератора	
		0	Источник тактового сигнала внешний (кварцевый резонатор или генератор)
		1	Источник тактового сигнала внутренний кольцевой генератор
SPI1CLKEN	6	Бит управления тактовым сигналом интерфейса SPI1	
		0	Тактовый сигнал не подается. Чтение регистров интерфейса возможно, запись в регистры запрещена
		1	Тактовый сигнал подается
I2CCLKEN	5	Бит управления тактовым сигналом интерфейса I2C	
		0	Тактовый сигнал не подается. Чтение регистров интерфейса возможно, запись в регистры запрещена
		1	Тактовый сигнал подается
MDUCLKEN	4	Бит управления тактовым сигналом блока MDU	
		0	Тактовый сигнал не подается. Чтение регистров блока возможно, запись в регистры запрещена
		1	Тактовый сигнал подается
UART1CLKEN	3	Бит управления тактовым сигналом блока UART1	
		0	Тактовый сигнал не подается. Чтение регистров интерфейса возможно, запись в регистры запрещена
		1	Тактовый сигнал подается
PCACLKEN	2	Бит управления тактовым сигналом блока PCA	
		0	Тактовый сигнал не подается. Чтение регистров интерфейса возможно, запись в регистры запрещена
		1	Тактовый сигнал подается

Окончание таблицы А.7

1	2	3	
X2	0	Бит выключения делителя. Позволяет пользователю выбирать частоту синхронизации исходя из соображений снижения электромагнитных помех.	
		0	Частота синхронизации с вывода XTAL1 делится на два и используется в качестве системной частоты
		1	Частота синхронизации с вывода XTAL1 не изменяется и является системной частотой с максимальным значением – 16 МГц
–	1	Зарезервировано	

Таблица А.8 – Слово состояния процессора

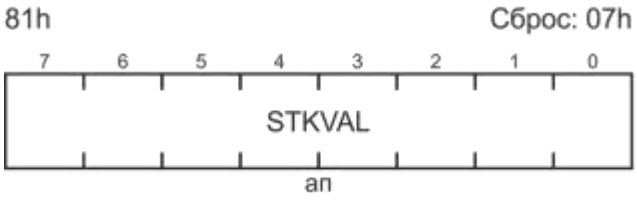
PSW



Поле	Бит	Описание			
C	7	Флаг переноса. Устанавливается и сбрасывается как аппаратно, так и программным путем			
AC	6	Флаг вспомогательного переноса. Устанавливается и сбрасывается только аппаратными средствами при выполнении команд сложения и вычитания и сигнализирует о переносе или заеме в бите 3 аккумулятора (ACC.3)			
F0	5	Флаг пользователя. Может быть установлен, сброшен или прочитан программой пользователя			
RS1, RS0	4, 3	Биты выбора используемого банка регистров. Могут быть изменены программным путем			
		RS1	RS0	Банк	Границы адресов ОЗУ
		0	0	0	00h – 07h
		0	1	1	08h – 0Fh
		1	0	2	10h – 17h
		1	1	3	18h – 1Fh
OV	2	Флаг переполнения. Устанавливается и сбрасывается аппаратно при выполнении арифметических команд			
P	0	Флаг паритета. Устанавливается и сбрасывается аппаратно в каждом цикле команды и фиксирует нечетное/четное число единичных бит в аккумуляторе			
–	1	Зарезервировано			

Таблица А.9 – Указатель стека

SP

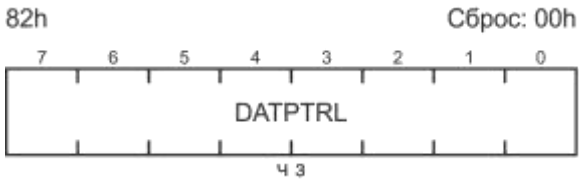


Поле	Бит	Описание
STKVAL	7–0	Адрес начала стека

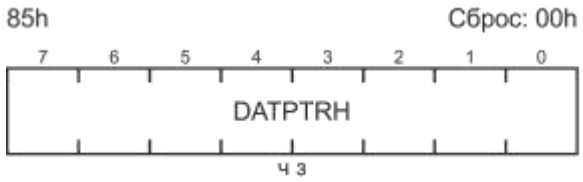
Таблица А.10 – Регистры основного указателя данных DPTR0 и дополнительного DPTR1

DP0H

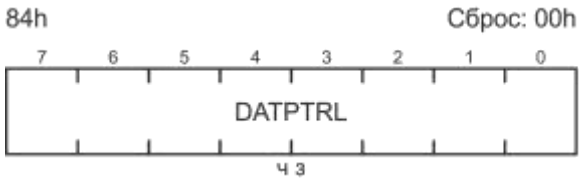
DP0L



DP1H



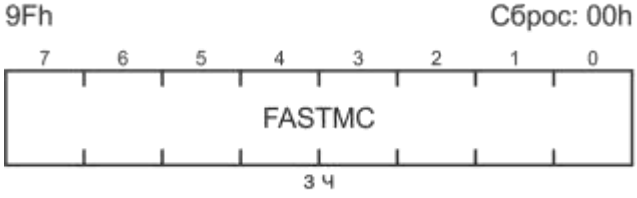
DP1L



Поле	Бит	Описание
DATPTRH	7–0	Старший байт адреса
DATPTRL	7–0	Младший байт адреса

Таблица А.11 – Регистр управления ускорением ядра

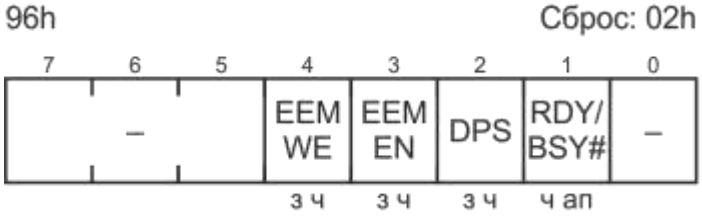
FASTM



Поле	Бит	Описание	
FASTMC	7–0	Поле кода ускорения ядра	
		05h	Механизм ускорения ядра включен
		00h	Механизм ускорения ядра выключен
		Остальные значения зарезервированы и запись любого из них не будет иметь результата	

Таблица А.12 – Регистр управления памятью данных EEPROM

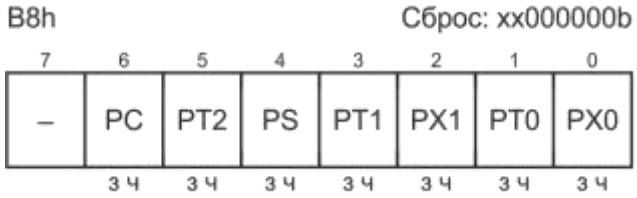
EECON



Поле	Бит	Описание	
EEMWE	4	Бит разрешения записи в память данных EEPROM. Для записи байта в EEPROM необходимо: - установить бит EEMWE; - подать команду записи MOVX; - по окончании записи программно сбросить бит EEMWE. Контроль записи осуществляется битом RDY/BSY#	
EEMEN	3	Бит разрешения доступа к встроенной памяти данных EEPROM	
		0	Команда MOVX получает доступ к внешней памяти данных по адресу, указанному в DPTR (адрес ≥ 2 К)
		1	Команда MOVX получает доступ к внутренней памяти данных EEPROM по адресу, указанному в DPTR. При этом адрес должен быть < 2К. В случае, если значение адреса ≥ 2К, действие команды MOVX будет аппаратно перенаправлено на внешнюю память данных
DPS	2	Выбор банка регистров указателя данных	
		0	Первый банк регистров – DP0H и DP0L
		1	Второй банк регистров – DP1H и DP1L
RDY/BSY#	1	Флаг обращения к памяти данных EEPROM. Выставляется и сбрасывается аппаратно	
		0	Выполнение команды MOVX завершено, и идет цикл записи/стирания памяти
		1	Цикл записи/стирания памяти завершен
–	7–5, 0	Зарезервировано	

Таблица А.13 – Регистр приоритетов прерываний

IP



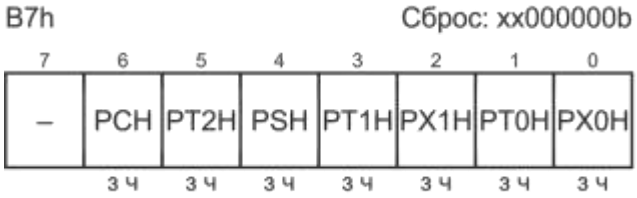
Поле	Бит	Описание	
1	2	3	
PC	6	Приоритет прерывания блока PCA	
		0	Низкий
		1	Высокий

Окончание таблицы А.13

1	2	3	
PT2	5	Приоритет прерывания таймера T2	
		0	Низкий
		1	Высокий
PS	4	Приоритет прерывания последовательного порта	
		0	Низкий
		1	Высокий
PT1	3	Приоритет прерывания таймера T1	
		0	Низкий
		1	Высокий
PX1	2	Приоритет внешнего прерывания INT1	
		0	Низкий
		1	Высокий
PT0	1	Приоритет прерывания таймера T0	
		0	Низкий
		1	Высокий
PX0	0	Приоритет внешнего прерывания INT0	
		0	Низкий
		1	Высокий
–	7	Зарезервировано	

Таблица А.14 – Регистр высоких приоритетов прерываний

IPH



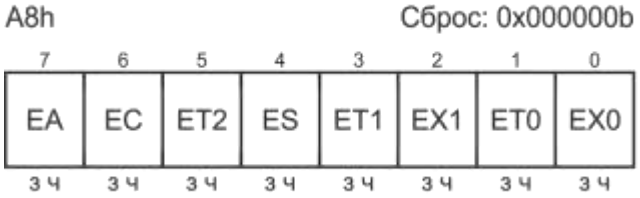
Поле	Бит	Описание	
1	2	3	
PCH	6	Приоритет прерывания блока PCA	
		0	Низкий
		1	Высокий
PT2H	5	Приоритет прерывания таймера T2	
		0	Низкий
		1	Высокий
PSH	4	Приоритет прерывания последовательного порта	
		0	Низкий
		1	Высокий
PT1H	3	Приоритет прерывания таймера T1	
		0	Низкий
		1	Высокий
PX1H	2	Приоритет внешнего прерывания INT1	
		0	Низкий
		1	Высокий

Окончание таблицы А.14

1	2	3	
PT0H	1	Приоритет прерывания таймера T0	
		0	Низкий
		1	Высокий
PX0H	0	Приоритет внешнего прерывания INT0	
		0	Низкий
		1	Высокий
–	7	Зарезервировано	

Таблица А.15 – Регистр разрешения прерываний

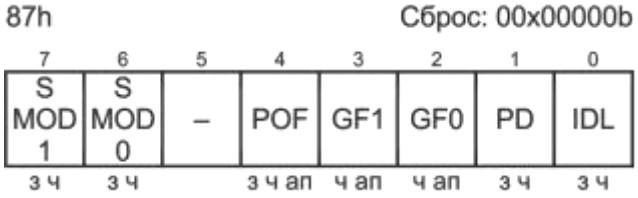
IE



Поле	Бит	Описание	
1	2	3	
EA	7	Бит запрета всех прерываний	
		0	Запрещены
		1	Разрешены. Прерывание каждого источника разрешается/запрещается соответствующим битом
EC	6	Бит разрешения прерывания блока PCA	
		0	Запрещено
		1	Разрешено
ET2	5	Бит разрешения прерываний таймера T2	
		0	Запрещено
		1	Разрешено
ES	4	Бит разрешения прерываний последовательного порта	
		0	Запрещено
		1	Разрешено
ET1	3	Бит разрешения прерывания таймера T1	
		0	Запрещено
		1	Разрешено
EX1	2	Бит разрешения внешнего прерывания INT1	
		0	Запрещено
		1	Разрешено
ET0	1	Бит разрешения прерывания таймера T0	
		0	Запрещено
		1	Разрешено
EX0	0	Бит разрешения внешнего прерывания INT0	
		0	Запрещено
		1	Разрешено

Таблица А.16 – Регистр управления энергопотреблением

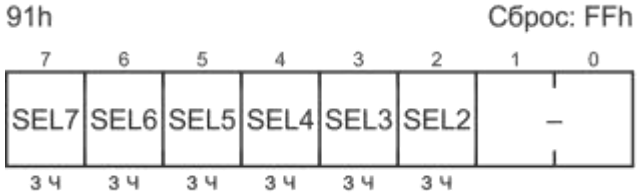
PCON



Поле	Бит	Описание	
SMOD1	7	Бит включения удвоенной скорости передачи	
		0	В режимах 0, 1, 2, 3 номинальная скорость передачи
		1	В режиме 0 скорость остается номинальной. В режимах 1, 2, 3 скорость передачи в два раза больше номинальной
SMOD0	6	Бит разрешения флага ошибки кадрирования	
		0	Бит SM0/FE регистра SCONx (x = 0, 1) является битом режима 0, т.е. SM0
		1	Бит SM0/FE регистра SCONx (x = 0, 1) является флагом ошибки кадрирования, т.е. FE
		Примечание – При каждом обнаружении ошибки кадрирования бит SM0/FE будет выставляться, т.е. работать как флаг FE, независимо от состояния бита SMOD0	
POF	4	Флаг отключения мощности. Флаг устанавливается во время возрастания мощности (повышения напряжения от 0 до 5 В), т.е. во время «холодной» перезагрузки. «Горячие» перезагрузки не оказывают влияния на состояние флага. Может устанавливаться и сбрасываться программно	
GF1	3	Флаг 1 общего назначения	
GF0	2	Флаг 0 общего назначения	
PD	1	Бит включения режима пониженного энергопотребления	
		0	Режим выключен
		1	Режим включен
IDL	0	Бит включения режима холостого хода	
		0	Режим выключен
		1	Режим включен
–	5	Зарезервировано	
Примечание – При одновременной установке битов PD и IDL бит PD имеет приоритет, поэтому будет включен режим PowerDown.			

Таблица А.17 – Регистр управления альтернативными функциями порта P1

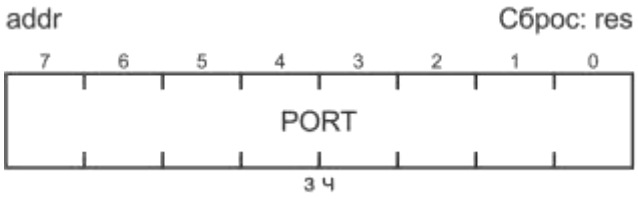
P1SEL



Поле	Бит	Описание	
SEL7	7	Бит выбора альтернативной функции вывода 7 порта 1	
		0	CEX4
		1	SCK0
SEL6	6	Бит выбора альтернативной функции вывода 6 порта 1	
		0	CEX3
		1	MISO0
SEL5	5	Бит выбора альтернативной функции вывода 5 порта 1	
		0	CEX2
		1	MOSI0
SEL4	4	Бит выбора альтернативной функции вывода 4 порта 1	
		0	CEX1
		1	SS0#
SEL3	3	Бит выбора альтернативной функции вывода 3 порта 1	
		0	CEX0
		1	SDA
SEL2	2	Бит выбора альтернативной функции вывода 2 порта 1	
		0	ECI
		1	SCL
–	1–0	Зарезервировано	

Таблица А.18 – Регистр порта (x – номер порта)

Px



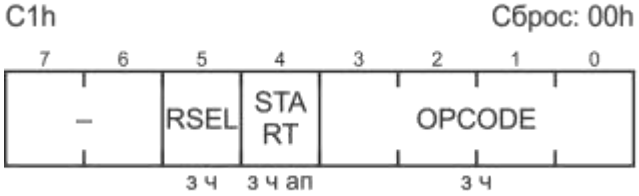
x	0	1	2	3	4	5
Регистр	P0	P1	P2	P3	P4	P5
addr	80h	90h	A0h	B0h	C0h	E8h
res	FFh	FFh	00h	FFh	FFh	FFh

Поле	Бит	Описание
PORT	7–0	Байт состояния порта

Регистры блока 16-разрядной арифметики

Таблица А.19 – Регистр управления

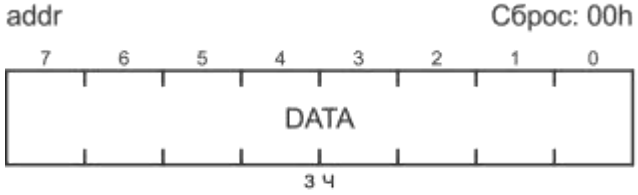
MDUCON



Поле	Бит	Описание	
RSEL	5	Бит выбора источника чтения	
		0	Регистры MRx
		1	Регистры MDx
START	4	Бит запуска вычисления. Установка этого бита запускает вычисление. Очистка бита происходит аппаратно в конце вычисления, поэтому он может использоваться как флаг	
OPCODE	3–0	Поле задания кода операции	
		0000	16-разрядное умножение без знака
		0001	16/16-разрядное деление без знака
		0010	32/16-разрядное деление без знака
		0011	32-разрядные логические сдвиги L/R
		0100	16-разрядное умножение со знаком
		0101	16/16-разрядное деление со знаком
		0110	32/16-разрядное деление со знаком
		0111	32-разрядный арифметический сдвиг L/R
		1000	32-разрядная нормализация
		Остальные значения зарезервированы. Запись их не будет иметь никакого результата	
–	7 – 6	Зарезервировано	

Таблица А.20 – Регистр данных (x – номер регистра)

MDx

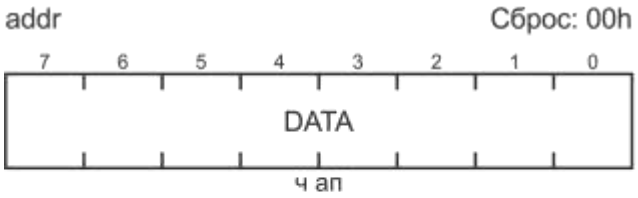


x	0	1	2	3	4	5
Регистр	MD0	MD1	MD2	MD3	MD4	MD5
addr	C2h	C3h	C4h	C5h	C6h	C7h

Поле	Бит	Описание
DATA	7–0	Операнд

Таблица А.21 – Регистр результата (x – номер регистра)

MRx

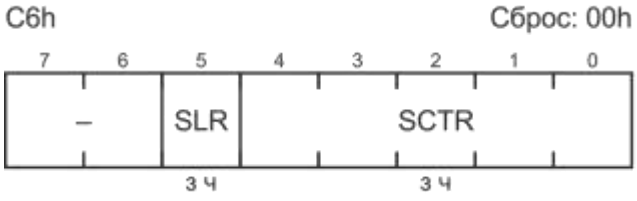


x	0	1	2	3	4	5
Регистр	MR0	MR1	MR2	MR3	MR4	MR5
addr	C2h	C3h	C4h	C5h	C6h	C7h

Поле	Бит	Описание
DATA	7–0	Результат операции

Таблица А.22 – Регистр управления сдвигом (альтернативная функция)

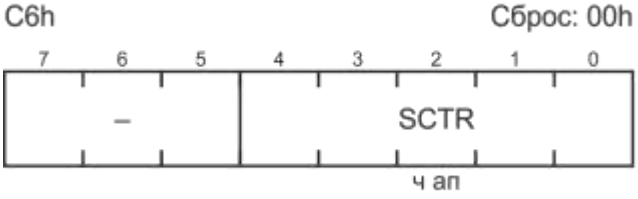
MD4



Поле	Бит	Описание	
SLR	5	Бит выбора направления сдвига	
		0	Сдвиг влево
		1	Сдвиг вправо
SCTR	4–0	Поле счетчика количества сдвигов. Определяет количество сдвигов, которые будут выполнены во время операции сдвига	
–	7–6	Зарезервировано	

Таблица А.23 – Регистр количества сдвигов (альтернативная функция)

MR4

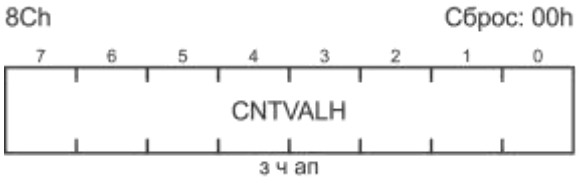


Поле	Бит	Описание
SCTR	4–0	Поле счетчика количества сдвигов. Определяют количество сдвигов, которые были выполнены во время операции нормализации
–	7–5	Зарезервировано

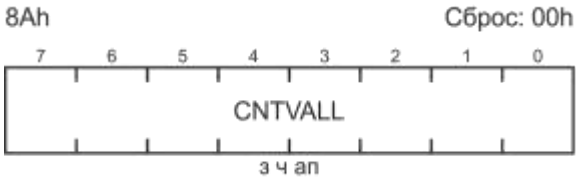
Таймеры-счетчики

Таблица А.24 – Регистры таймеров Т0, Т1 и Т2

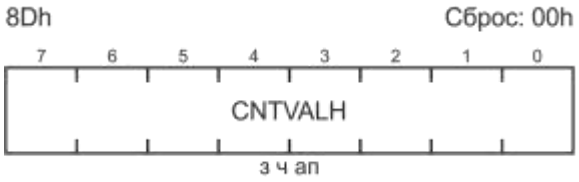
TH0



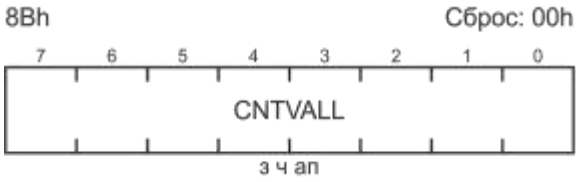
TL0



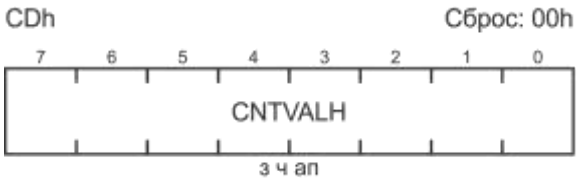
TH1



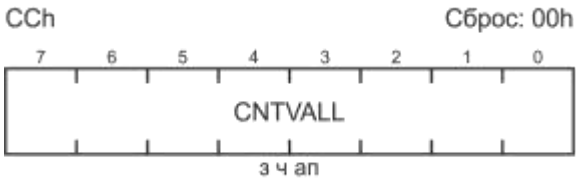
TL1



TH2



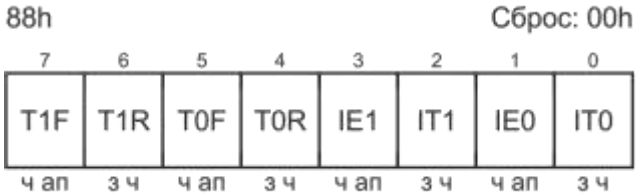
TL2



Поле	Бит	Описание
CNTVALH	7–0	Старший байт счетчика
CNTVALL	7–0	Младший байт счетчика

Таблица А.25 – Регистр управления и состояния таймеров Т0 и Т1

TCON



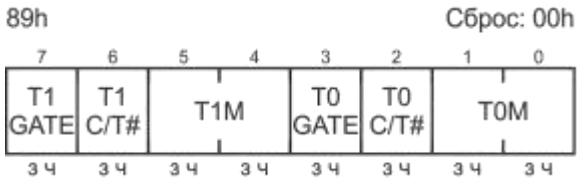
Поле	Бит	Описание	
1	2	3	
T1F	7	Флаг переполнения таймера 1. Устанавливается аппаратно при переполнении таймера/счетчика Т1. Сбрасывается аппаратно при обслуживании прерывания	
T1R	6	Бит запуска/останова таймера Т1	
		0	Остановлен
		1	Работает
T0F	5	Флаг переполнения таймера 0. Устанавливается аппаратно при переполнении таймера/счетчика Т0. Сбрасывается аппаратно при обслуживании прерывания	

Окончание таблицы А.25

1	2	3	
T0R	4	Бит запуска/останова таймера Т0	
		0	Остановлен
		1	Работает
IE1	3	Флаг фронта прерывания по событию таймера Т1. Устанавливается аппаратно, когда детектируется фронт внешнего сигнала на входе INT1# (P3.3). Сбрасывается аппаратно при обслуживании прерывания	
IT1	2	Бит управления типом внешнего прерывания INT1. Устанавливается/сбрасывается программно для спецификации запроса INT1# (срез/низкий уровень)	
		0	Прерывание, если на входе INT1# низкий уровень сигнала
		1	Прерывание, если на входе INT1# обнаруживается фронт перехода уровня сигнала из верхнего в нижний
IE0	1	Флаг фронта прерывания по событию таймера Т0. Устанавливается аппаратно, когда детектируется фронт внешнего сигнала на входе INT0# (P3.2). Сбрасывается аппаратно при обслуживании прерывания	
IT0	0	Бит управления типом внешнего прерывания INT0. Устанавливается/сбрасывается программно для спецификации запроса INT0# (срез/низкий уровень)	
		0	Прерывание, если на входе INT0# низкий уровень сигнала
		1	Прерывание, если на входе INT0# обнаруживается фронт перехода уровня сигнала из верхнего в нижний

Таблица А.26 – Регистр режимов таймеров Т0 и Т1

TMOD



Поле	Бит	Описание	
1	2	3	
T1GATE	7	Управление блокировкой таймера Т1	
		0	Блокировка снята, если установлен бит управления T1R (регистр TCON)
		1	Блокировка снята, если установлен бит T1R (регистр TCON) и на входе INT1# (P3.3) поддерживается высокий уровень сигнала
T1C/T#	6	Бит конфигурации таймера Т1	
		0	Таймер, тактируемый внутренним сигналом синхронизации
		1	Счетчик, тактируемый сигналом со входа Т1 (P3.5)

Окончание таблицы А.26

1	2	3	
T1M	5–4	Поле выбора режима работы таймера T1	
		00	Регистр TL1 работает как 5-битный предделитель
		01	Регистры TH1 и TL1 включены последовательно, образуя, таким образом, 16-битный таймер/счетчик
		10	Регистр TL1 работает как 8-битный автоперезагружаемый таймер/счетчик. TH1 хранит значение, которое загружается в TL1 каждый раз по его переполнению
		11	Таймер T1 выключен
T0GATE	3	Управление блокировкой таймера T0	
		0	Блокировка снята, если установлен бит управления T0R (регистр TCON)
		1	Блокировка снята, если установлен бит T0R (регистр TCON) и на входе INT0# (P3.2) поддерживается высокий уровень сигнала
T0C/T#	2	Бит конфигурации таймера T0	
		0	Таймер, тактируемый внутренним сигналом синхронизации
		1	Счетчик, тактируемый сигналом со входа T0 (P3.4)
T0M	1–0	Поле выбора режима работы таймера T0	
		00	Регистр TL0 работает как 5-битный предделитель
		01	Регистры TH0 и TL0 включены последовательно, образуя таким образом 16-битный таймер/счетчик
		10	Регистр TL0 работает как 8-битный автоперезагружаемый таймер/счетчик. TH0 хранит значение, которое загружается в TL0 каждый раз по его переполнению
		11	Регистр TL0 работает как 8-битный таймер/счетчик. Регистр TH0 работает как 8-битный таймер. Режимы обоих регистров определяются управляющими регистрами таймера T0

Таблица А.27 – Регистр управления таймера T2

T2CON

C8h Сброс: 00h

7	6	5	4	3	2	1	0
T2F	EXF2	RCLK	TCLK	EX EN2	T2R	C/ T2#	CP/ RL2#
з ч ап	з ч ап	з ч	з ч	з ч	з ч	з ч	з ч

Поле	Бит	Описание
1	2	3
T2F	7	Флаг переполнения таймера T2. Устанавливается (если RCLK = TCLK = 0b) при переполнении регистра таймера T2. Флаг сбрасывается программно

Окончание таблицы А.27

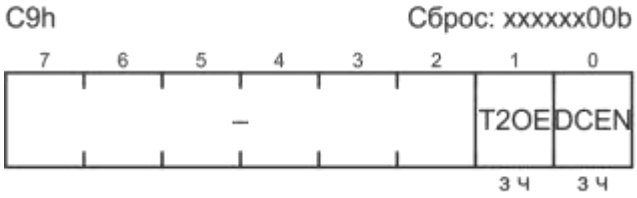
1	2	3	
EXF2	6	Внешний флаг таймера Т2. Устанавливается (если EXEN2 = 1b), когда захват/ перезагрузка таймера Т2 происходит в ответ на появление фронта отрицательного перепада уровня сигнала на входе Т2EX. Если прерывание таймера Т2 разрешено, то выставление флага EXF2 вызовет подпрограмму обработки прерывания таймера Т2. Если таймер Т2 работает в режиме счетчика с двунаправленным счетом (бит DCEN = 1b), выставление флага EXF2 не вызывает прерывания	
RCLK*	5	Бит выбора таймера для приемника. Задает таймер, последовательность импульсов переполнения которого будет использоваться как сигнал синхронизации приемника последовательного порта UARTx (режимы 1 и 3)	
		0	Таймер Т1
		1	Таймер Т2
TCLK*	4	Бит выбора таймера для передатчика. Задает таймер, последовательность импульсов переполнения которого будет использоваться как сигнал синхронизации передатчика последовательного порта UARTx (режимы 1 и 3)	
		0	Таймер Т1
		1	Таймер Т2
EXEN2	3	Бит разрешения тактирования таймера Т2 внешним сигналом со входа Т2EX	
		0	Запрещено. События на входе Т2EX не влияют на состояние таймера
		1	Разрешено. В ответ на появление фронта отрицательного перепада уровня сигнала на входе Т2EX происходит захват/ перезагрузка таймера Т2 (если RCKL = TCKL = 0b)
T2R*	2	Бит запуска/останова таймера Т2	
		0	Остановлен
		1	Работает
C/T2#	1	Бит конфигурации таймера Т2	
		0	Таймер
		1	Счетчик внешних событий. Подсчет количества фронтов отрицательных перепадов уровня сигнала на входе P1.0
CP/RL2#*	0	Бит выбора режима работы таймера Т2	
		0	Если EXEN2 = 1b и RCKL = TCKL = 0b, то переполнение собственного регистра таймера или появление фронта отрицательного перепада уровня сигнала на входе Т2EX приводит к перезагрузка таймера Т2. Если EXEN2 = 0b или RCKL = 1b, или TCKL = 1b, то перезагрузка таймера Т2 инициируется только переполнением регистра таймера
		1	Если EXEN2 = 1b, то появление фронта отрицательного перепада уровня сигнала на входе Т2EX приводит к захвату таймера Т2
Примечание – Четыре бита, отмеченные символом «*», совместно образуют комбинации сигналов, которые задают тот или иной режим работы таймер/счетчика Т2. Описание этих комбинаций и соответствующих режимов приводится в таблице А.28.			

Таблица А.28 – Комбинации битов регистра T2CON и режимы работы T2

RCLK	TCLK	CP/RL2#	T2R	Режим
0	0	0	1	16-битный таймер/счетчик с перезагрузкой
0	0	1	1	16-битный таймер/счетчик с защелкиванием информации (захват)
1	0	Не важно	1	Генератор импульсов
0	1			
1	1			
Не важно			0	Выключен

Таблица А.29 – Регистр режима таймера T2

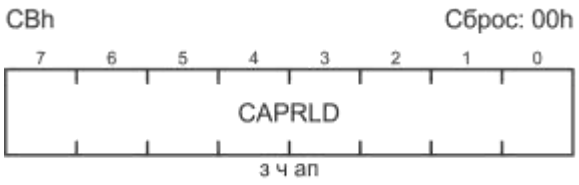
T2MOD



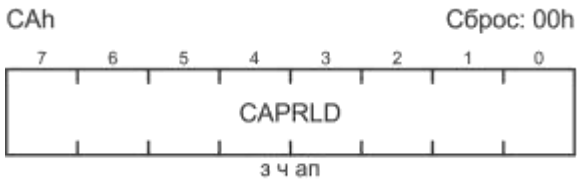
Поле	Бит	Описание	
T2OE	1	Бит разрешения передачи выходного сигнала таймера T2	
		0	Запрещено
		1	Разрешено
DCEN	0	Режим работы таймера T2 в конфигурации счетчика	
		0	Однонаправленный счетчик. Только инкремент
		1	Двунаправленный счетчик. Инкремент (если на выводе микроконтроллера T2EX удерживается высокий уровень сигнала) и декремент (если на выводе T2EX – низкий уровень сигнала)
–	7–2	Зарезервировано	

Таблица А.30 – Регистры захвата/перезагрузки таймера T2

RCAP2H



RCAP2L

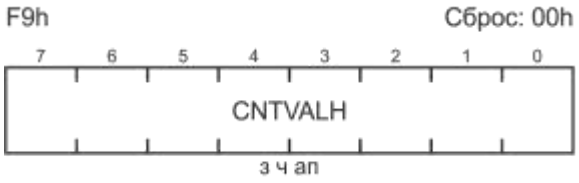


Поле	Бит	Описание
CAPRLD	7–0	Значения перезагрузки старшего и младшего байт таймера T2

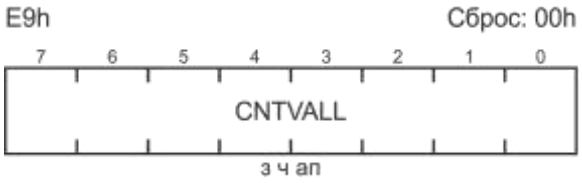
Регистры массива программируемых счетчиков (PCA)

Таблица А.31 – Регистры таймера-счетчика

CH



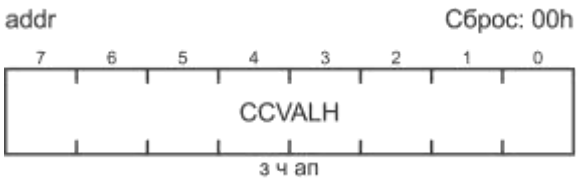
CL



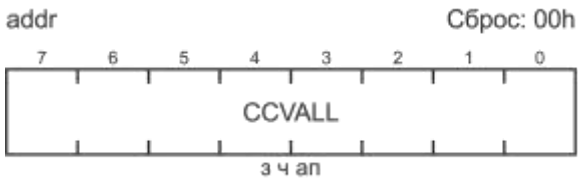
Поле	Бит	Описание
CNTVALH	7–0	Старший байт счетчика
CNTVALL	7–0	Младший байт счетчика

Таблица А.32 – Регистры захвата/сравнения канала (x – номер модуля)

CCAPxH



CCAPxL

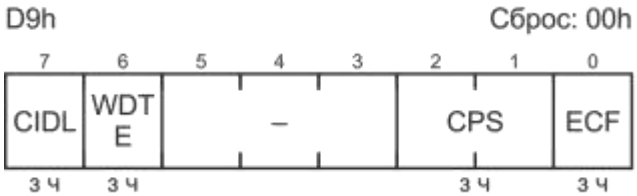


x	0		1		2		3		4	
Регистр	CCAP0H	CCAP0L	CCAP1H	CCAP1L	CCAP2H	CCAP2L	CCAP3H	CCAP3L	CCAP4H	CCAP4L
addr	FAh	EAh	FBh	EBh	FCh	ECh	FDh	EDh	FEh	EEh

Поле	Бит	Описание
CCVALH	7–0	Старший байт значения захвата/сравнения
CCVALL	7–0	Младший байт значения захвата/сравнения

Таблица А.33 – Регистр режима таймера-счетчика

CMOD



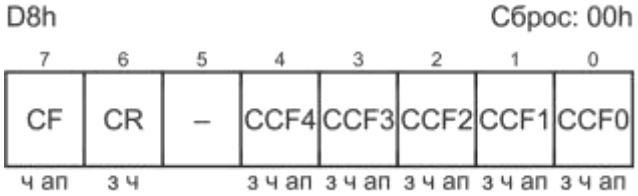
Поле	Бит	Описание	
1	2	3	
CIDL	7	Бит разрешения функционирования блока PCA в режиме Idle	
		0	Разрешено
		1	Запрещено
WDTE	6	Бит разрешения функции сторожевого таймера модуля 4	
		0	Запрещено
		1	Разрешено

Окончание таблицы А.33

1	2	3	
CPS	2–1	Поле выбора источника синхросигнала	
		00	fosc/12
		01	fosc/4
		10	Сигнал переполнения таймера T0
		11	Вход ECI (P1.2)
ECF	0	Бит разрешения прерывания блока PCA	
		0	Запрещено
		1	Разрешено
–	5 – 3	Зарезервировано	

Таблица А.34 – Регистр управления таймера-счетчика

CCON

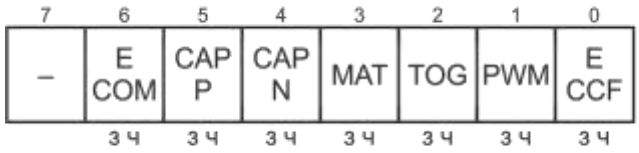


Поле	Бит	Описание	
CF	7	Флаг переполнения таймера/счетчика PCA. Вызывает прерывание, если установлен флаг ECF регистра CMOD	
CR	6	Бит включения таймера/счетчика PCA	
		0	Остановлен
		1	Работает
CCF4	4	Флаг прерывания модуля 4	
CCF3	3	Флаг прерывания модуля 3	
CCF2	2	Флаг прерывания модуля 2	
CCF1	1	Флаг прерывания модуля 1	
CCF0	0	Флаг прерывания модуля 0	
–	5	Зарезервировано	

Таблица А.35 – Регистр режима работы модуля захвата/сравнения канала (x – номер модуля)

CCAPMx

addr Сброс: 00h



x	0	1	2	3	4
Регистр	CCAPM0	CCAPM1	CCAPM2	CCAPM3	CCAPM4
addr	DAh	DBh	DCh	DDh	DEh

Поле	Бит	Описание	
ECOM	6	Бит разрешения выполнения функции сравнения	
		0	Запрещено
		1	Разрешено
CAPP	5	Бит разрешения сравнения по положительному фронту	
		0	Запрещено
		1	Разрешено
CAPN	4	Бит разрешения сравнения по отрицательному фронту	
		0	Запрещено
		1	Разрешено
MAT	3	Бит разрешения флага прерывания	
		0	Запрещено
		1	При совпадении значений таймера-счетчика и регистра модуля выставляется соответствующий флаг CCFx в регистре CCON
TOG	2	Бит разрешения переключения вывода канала	
		0	Запрещено
		1	При совпадении значений таймера-счетчика и регистра модуля на выходе модуля CEXx переключается уровень выходного сигнала
PWM	1	Бит включения режима ШИМ	
		0	Режим ШИМ выключен
		1	Режим ШИМ включен, модулированный сигнал подается на выход CEXx
ECCF	0	Бит разрешения прерывания по флагу CCFx регистра CCON	
		0	Запрещено
		1	Разрешено
–	7	Зарезервировано	

Примечание – Допустимые комбинации битов регистра и соответствующие им режимы работы модуля x блока PCA представлены в таблицах А.36 и А.37

Таблица А.36 – Состояния битов регистра CCAPMx и соответствующие режимы захвата

ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	Режим захвата канала x
–	1	0	0	0	0	–	Захват содержимого регистра таймера/счетчика ТС в ответ на появление фронта перепада сигнала из «0» в «1» на входе CEXx
–	0	1	0	0	0	–	Захват содержимого регистра таймера/счетчика ТС в ответ на появление фронта перепада сигнала из «1» в «0» на входе CEXx
–	1	1	0	0	0	–	Захват содержимого регистра таймера/счетчика ТС в ответ на появление любого фронта перепада сигнала на входе CEXx

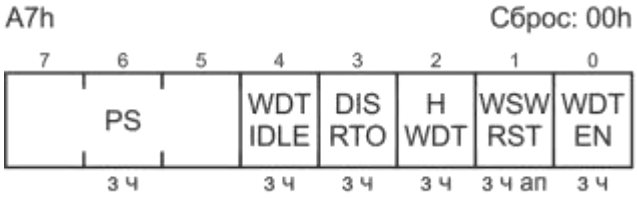
Таблица А.37 – Состояния битов регистра CCAPMx и соответствующие режимы сравнения

ECOM	CAPP	CAPN	MAT	TOG	PWM	ECCF	Режим сравнения канала x
1	0	0	1	0	0	–	16-разрядный таймер
1	0	0	1	1	0	–	Скоростной вывод
1	0	0	0	0	1	0	Широтно-импульсный модулятор
1	0	0	1	–	0	–	Сторожевой таймер (только для модуля 4)
0	0	0	0	0	0	0	Модуль x выключен

Регистры сторожевого таймера (WDT)

Таблица А.38 – Регистр управления сторожевого таймера

WDTCON



Поле	Бит	Описание	
1	2	3	
PS	7–5	Поле задания периода отключения сторожевого таймера	
		000	16 мс (минимальный номинальный период)
		001	32 мс
		010	64 мс
		011	128 мс
		100	256 мс
		101	512 мс
		110	1024 мс
		111	2048 мс (максимальный номинальный период)
WDTIDLE	4	Бит выключения сторожевого таймера в режиме IDLE	
		0	Работает, когда микроконтроллер находится в режиме IDLE
		1	Приостанавливает работу, пока микроконтроллер находится в режим IDLE
DISRTO	3	Бит управления доступом сторожевого таймера к выводу RST	
		0	После окончания счета сторожевого таймера и сброса микроконтроллера на выводе RST устанавливается высокий уровень сигнала
		1	После окончания счета сторожевого таймера происходит сброс микроконтроллера. Вывод RST работает как вход
HWDT	2	Бит выбора режима аппаратного управления	
		0	Сторожевой таймер может быть запущен/остановлен установкой/сбросом бита WDTEN
		1	Сторожевой таймер запускается записью 1Eh/E1h в регистр WDTRST, после чего сторожевой таймер не может быть остановлен, за исключением холодного или горячего сброса. Для предотвращения сброса микроконтроллера по переполнению сторожевого таймера следует постоянно записывать 1Eh/E1h в регистр WDTRST

Окончание таблицы А.38

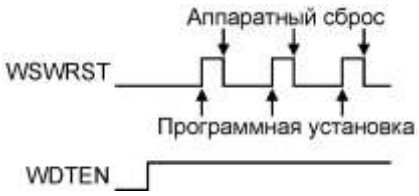
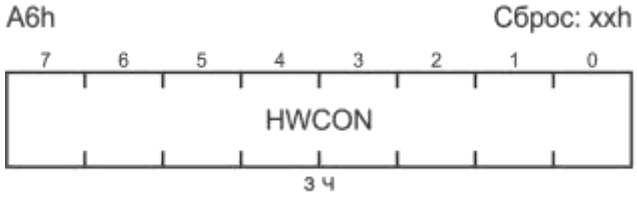
1	2	3
WSWRST	1	Бит программного сброса. Если бит HWDT сброшен (сторожевой таймер находится в режиме программного управления), программная установка бита WSWRST приводит к сбросу сторожевого таймера. После установки этот бит сбрасывается аппаратно в течение машинного цикла. В случае, если установлен бит HWDT, то установка бита WSWRST не будет влиять на работу сторожевого таймера и аппаратного сброса установленного бита не произойдет 
WDTEN	0	Бит выбора режима программного управления. Если бит HWDT сброшен, то сторожевой таймер находится в режиме программного управления. Запуск/останов сторожевого таймера осуществляется установкой/очисткой бита WDTEN. Этот бит не сбрасывает сторожевой таймер, а только запускает/останавливает счетчик. Если бит HWDT установлен, то бит WDTEN доступен только для чтения и показывает состояние сторожевого таймера – запущен («1») или остановлен («0»)

Таблица А.39 – Регистр аппаратного режима сторожевого таймера

WDTRST



Поле	Бит	Описание
HWCON	7–0	Байт запуска сторожевого таймера. Рекомендуемое значение – 1Eh/E1h . Это значение следует постоянно записывать в битовое поле HWCON по переполнению сторожевого таймера для предотвращения сброса микроконтроллера

Регистры приемо-передатчиков UART0 и UART1

Таблица А.40 – Регистр данных (номер порта)

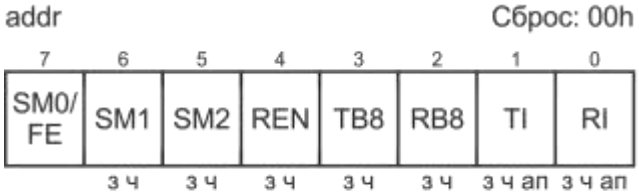
SBUFx

x	0	1
Регистр	SBUF0	SBUF1
addr	99h	9Bh

Поле	Бит	Описание
DATA	7 – 0	Данные

Таблица А.41 – Регистр управления и статуса порта x

SCONx



x	0	1
Регистр	SCON0	SCON1
addr	98h	9Ah

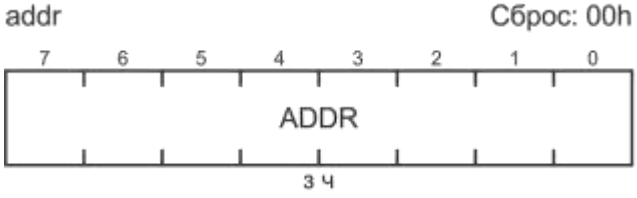
Поле	Бит	Описание				
1	2	3				
SM0/ FE	7	Бит режима 0/ Флаг ошибки кадрирования. Бит SM0. Для доступа к этому биту необходимо в регистре PCON установить бит SMOD0 равным «0». Бит FE. При обнаружении ошибок кадрирования в сообщениях определяются отсутствующие стоп-биты, и устанавливается флаг FE (для этого необходимо предварительно в регистре PCON установить бит SMOD0 равным «1»). Флаг FE должен сбрасываться программно				
SM1	6	Бит режима 1				
		Режим	SM0	SM1	Описание	Скорость передачи
		0	0	0	Сдвиговый регистр	f/12
		1	0	1	8-бит UART	переменная
		2	1	0	9-бит UART	f/64 или f/32
		3	1	1	9-бит UART	переменная

Окончание таблицы А.41

1	2	3
SM2	5	Бит режима 2. В режиме 0 бит SM2 должен быть равным «0». В режиме 1, если SM2 = 1b, флаг RI не устанавливается до получения достоверного стоп-бита после приема байта адреса. В режимах 2 и 3, если SM2 = 1b, флаг RI устанавливается только в том случае, если бит 9 данных равен «1» (это указывает на то, что полученный байт является адресом).
REN	4	Бит разрешения последовательного приема
		0 – Прием запрещен
		1 – Прием разрешен
TB8	3	Девятый передаваемый бит данных. Бит, который будет передан девятым в посылке в режимах 2 и 3
RB8	2	Девятый принятый бит данных. В режиме 0 – не используется. В режиме 1 – стоп-бит (если SM2 = 0b). В режимах 2 и 3 – бит данных, который был получен девятым
TI	1	Флаг окончания передачи. В режиме 0 – устанавливается аппаратно после передачи восьмого бита. В режимах 1, 2, 3 – устанавливается аппаратно в начале передачи стоп-бита. Сбрасывается программно
RI	0	Флаг окончания приема. В режиме 0 – устанавливается аппаратно после приема восьмого бита. В режимах 1, 2, 3 – устанавливается аппаратно в течение приема стоп-бита. Сбрасывается программно

Таблица А.42 – Регистр заданного адреса порта x

SADDRx

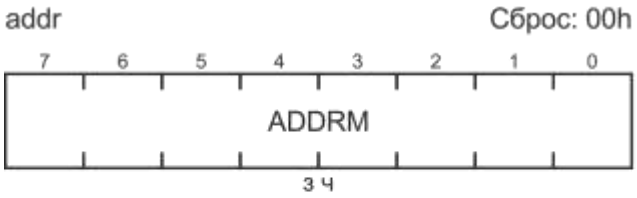


x	0	1
Регистр	SADDR0	SADDR1
addr	A9h	ABh

Поле	Бит	Описание
ADDR	7 – 0	Заданный адрес

Таблица А.43 – Регистр маски адреса порта x

SADENx



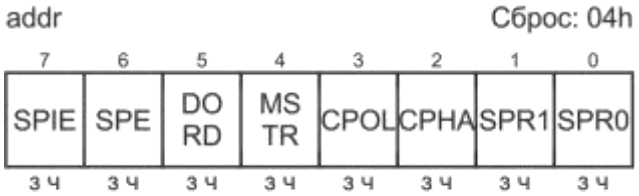
x	0	1
Регистр	SADEN0	SADEN1
addr	B9h	BBh

Поле	Бит	Описание
ADDRM	7–0	Маска заданного адреса

Регистры контроллеров интерфейса SPI

Таблица А.44 – Регистр управления контроллера SPIx (x – номер модуля)

SPCRx

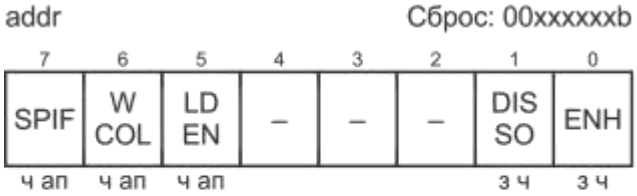


x	0	1
Регистр	SPCR0	SPCR1
addr	D5h	D6h

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	fosc/4	fosc/2
		0	1	fosc/16	fosc/8
		1	0	fosc/64	fosc/32
		1	1	fosc/128	fosc/64

Таблица А.45 – Регистр состояния контроллера SPIx (x – номер модуля)

SPSRx



x	0	1
Регистр	SPSR0	SPSR1
addr	AAh	ACh

Поле	Бит	Описание
1	2	3
SPIF	7	Флаг прерывания модуля SPI. Устанавливается 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. 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. 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.45

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.46 – Регистр данных контроллера SPIx (x – номер модуля)

SPDRx

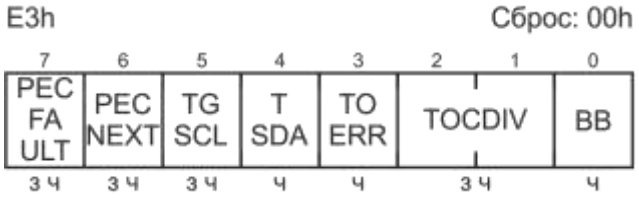
x	0	1
Регистр	SPDR0	SPDR1
addr	86h	9Eh

Поле	Бит	Описание
DATA	7–0	Данные

Регистры контроллера интерфейса I2C

Таблица А.47 – Регистр управления и состояния I2C

SMBCST



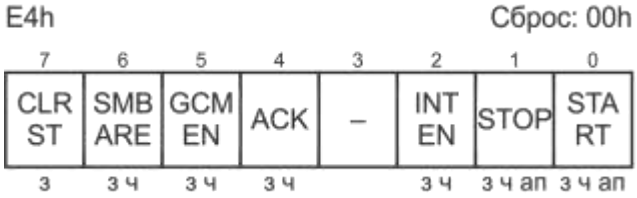
Поле	Бит	Описание
1	2	3
PECFAULT	7	Флаг ошибки. Устанавливается в случае, если после расчета контрольной суммы для пакета данных и сравнения ее с полученной, значение во внутреннем регистре ошибок не нулевое
PECNEXT	6	Бит управления отправкой байта контрольной суммы. Установка бита указывает на то, что 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.47

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.48 – Регистр 1 управления I2C

SMBCTRL1



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	
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 содержит значение, которое передается в течение цикла отклика на запрос передатчика подтвердить прием. Передача нуля по окончании передачи байта означает, что данные успешно получены. Передача единицы означает, что приемник не может продолжать работу по каким - либо причинам. Бит ACK очищается аппаратно по окончании цикла отклика	

Окончание таблицы А.48

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	

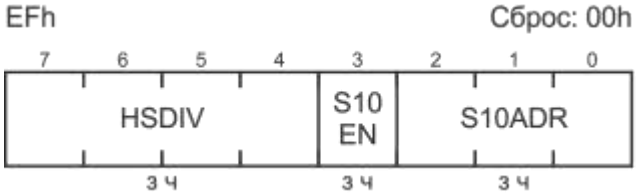
Таблица А.49 – Регистр 2 управления I2C

SMBCTRL2

Поле	Бит	Описание	
SCLFRQ	7–1	Поле выбора частоты fscl сигнала на выводе SCL в режиме мастера. Длительности высокого(TSCLH) и низкого(TSCLL) уровней сигнала SCL зависят тактовой частоты fosc модуля I2C и рассчитываются по формулам: TSCLH = 2 × SCLFRQ × (1/fosc), TSCLL = 2 × SCLFRQ × (1/fosc). Таким образом, частота сигнала на выводе SCL равна: fscl = 1/(TSCLH + TSCLL). 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.50 – Регистр 3 управления I2C

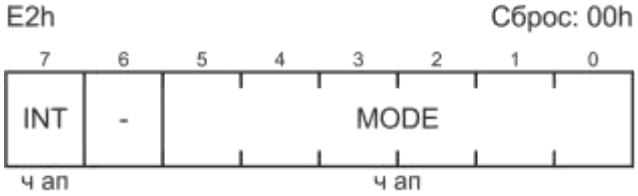
SMBCTRL3



Поле	Бит	Описание	
HSDIV	7–4	Поле выбора частоты fscl сигнала на выводе SCL в режиме HS мастера. Длительности высокого(THSCLH) и низкого(THSCLL) уровней сигнала на выводе SCL зависят от тактовой частоты fosc модуля I2C и рассчитываются по формулам: THSCLH = HSDIV × (1/fosc), THSCLL = 2 × HSDIV × (1/fosc). Таким образом, частота сигнала на выводе SCL равна: fscl = 1/(THSCLH + THSCLL). 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 HSDIV 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]	

Таблица А.51 – Регистр состояния I2C

SMBST



Поле	Бит	Описание
INT	7	Флаг прерывания. Устанавливается после девятого 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); - неквитированая 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.52 – Регистр собственного адреса I2C

SMBADDR

Поле	Бит	Описание	
SAEN	7	Бит разрешение 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 семь 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#)	

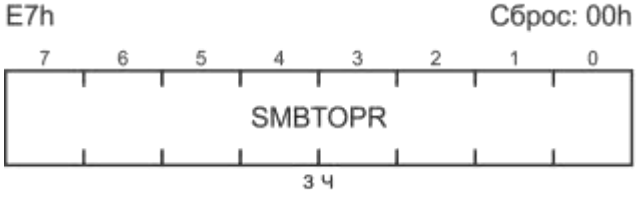
Таблица А.53 – Сдвиговый регистр данных I2C

SMBSDA

Поле	Бит	Описание
DATA	7–0	Поле данных

Таблица А.54 – Регистр загрузки предделителя

SMBTOPR



Поле	Бит	Описание
SMBTOPR	7–0	Значение перезагрузки предделителя

Регистры контроллера интерфейса LIN

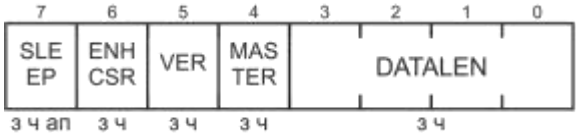
Таблица А.55 – Регистры данных

Мнемоническое название	Назначение	Адрес (WSR = 01h)	Состояние после сброса
LINDTX0	Регистры данных для передачи	90h	00h
LINDTX1		91h	
LINDTX2		92h	
LINDTX3		93h	
LINDTX4		94h	
LINDTX5		95h	
LINDTX6		96h	
LINDTX7		97h	
LINDRX0	Регистры принятых данных	98h	
LINDRX1		99h	
LINDRX2		9Ah	
LINDRX3		9Bh	
LINDRX4		9Ch	
LINDRX5		9Dh	
LINDRX6		9Eh	
LINDRX7		9Fh	

Таблица А.56 – Старший и младший байты регистра управления LIN

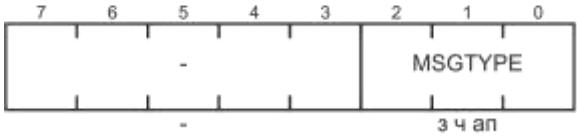
LINCONH

A1h (WSR = 01h) Сброс: 08h



LINCONL

A0h (WSR = 01h) Сброс: 01h



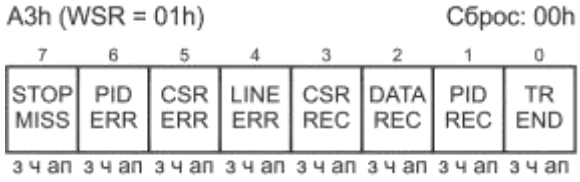
Поле	Бит	Описание	
1	2	3	
SLEEP	7	Бит включения режима сна	
		0	Нормальный режим работы
		1	Включение режима сна
ENHCSR	6	Тип контрольной суммы	
		0	Обычная
		1	Расширенная
VER	5	Версия протокола	
		0	Версия 1.X
		1	Версия 2.X
MASTER	4	Бит типа устройства	
		0	Ведомый
		1	Мастер
DATALEN	3–0	Количество байт данных. Заполнять это поле необходимо только при использовании протокола версии 2.X. По умолчанию значение поля – 8h (восемь байт данных)	

Окончание таблицы А.56

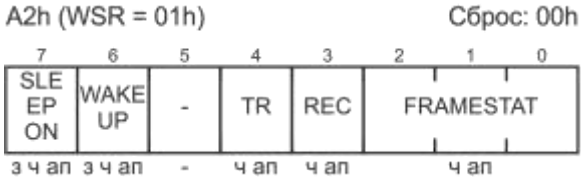
1	2	3	
MSGTYPE	2–0	Тип кадра сообщения	
		001	BUSWAIT (устройство находится в состоянии ожидания следующего кадра сообщения)
		010	HEADER
		011	HEADER+RESPONSE
		100	RESPONSE RECEIVE
		101	RESPONSE TRANSMIT
		111	WAKE
–	7–3	Зарезервировано	

Таблица А.57 – Старший и младший байты регистра состояния LIN

LINSTATH



LINSTATL



Поле	Бит	Описание
1	2	3
STOPMISS	7	Флаг отсутствия стоп-бита. Устанавливается при отсутствии в поле стопового бита
PIDERR	6	Флаг ошибки PID. Устанавливается при ошибке четности в PID
CSRERR	5	Флаг ошибки контрольной суммы. Устанавливается, когда принятая сумма отличается от суммы, рассчитанной на основе принятых полей
LINEERR	4	Флаг конфликта линии. Устанавливается, если при передаче рецессивного бита линия находится в доминантном состоянии
CSRREC	3	Флаг получения контрольной суммы. Устанавливается при получении поля контрольной суммы
DATAREC	2	Флаг получения данных. Устанавливается при получении поля данных
PIDREC	1	Флаг получения PID. Устанавливается при получении PID
TREND	0	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

Окончание таблицы А.57

1	2	3	
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
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 регистров LININTENH и LININTENL 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.

Таблица А.58 – Старший и младший байты регистра разрешения прерываний LIN

LININTENH

ADh (WSR = 01h) Сброс: 00h

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
З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч	З Ч

LININTENL

ACh (WSR = 01h) Сброс: 00h

7	6	5	4	3	2	1	0
SLE EPON EN	WAKE UP EN	-	TR EN	REC EN	FRAMESTATEN		
З Ч	З Ч	-	З Ч	З Ч	З Ч		

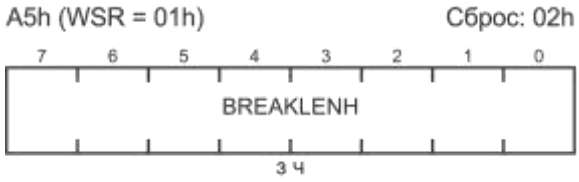
Описание

Биты регистра LININTENH и биты с седьмого по третий регистра LININTENL являются битами разрешения для соответствующих флагов регистров LINSTATH и LINSTATL и прерываний.

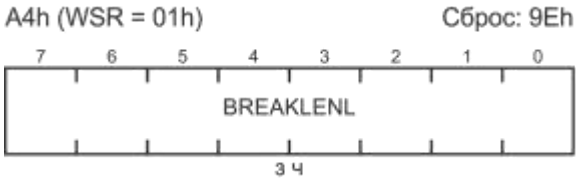
Поле FRAMESTATEN задает код режима, при входе в который будет генерироваться прерывание и выставляться соответствующий код в поле FRAMESTAT регистра LINSTATL

Таблица А.59 – Старший и младший байты регистра настройки длины поля BREAK

LINBREAKH



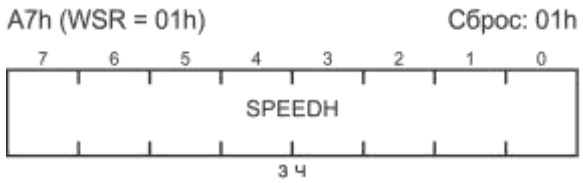
LINBREAKL



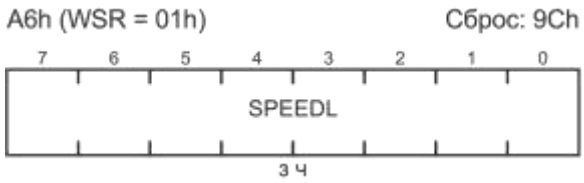
Поле	Бит	Описание
BREAKLENH	7–0	Старший и младший байты слова, которое задает минимальную длину поля разрыва синхронизации, выраженную в периодах тактового сигнала. Длина вычисляется по формуле: $N = \frac{Fxtal}{2 \cdot Flin} \times \frac{Nbit}{16}$, где Fxtal – тактовая частота микроконтроллера, Flin – скорость передачи данных блоком LIN, Nbit – минимальное количество бит в поле разрыва синхронизации
BREAKLENL	7–0	

Таблица А.60 – Старший и младший байты регистра настройки скорости LIN

LINSPEEDH



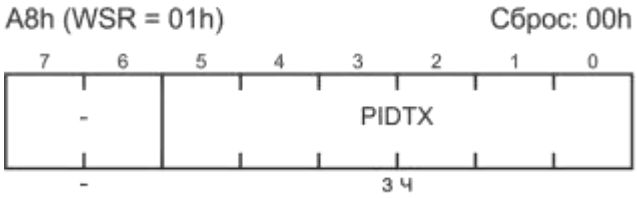
LINSPEEDL



Поле	Бит	Описание
SPEEDH	7–0	Старший и младший байты слова, которое задают скорость передачи блока LIN при функционировании в режиме мастера. В режиме ведомого состояние регистров не важно. Значение, которое записывается в регистры, вычисляется по формуле: $N = \frac{Fxtal}{4 \cdot Flin}$ где Fxtal – тактовая частота микроконтроллера, Flin – скорость передачи данных блоком LIN
SPEEDL	7–0	

Таблица А.61 – Регистр передаваемого PID

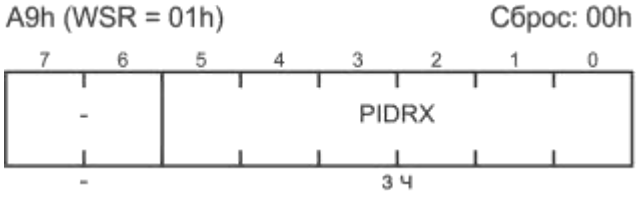
LINPIDTX



Поле	Бит	Описание
PIDTX	5–0	PID, который будет отправлен в следующем кадре данных
–	7, 6	Зарезервировано

Таблица А.62 – Регистр принятого PID

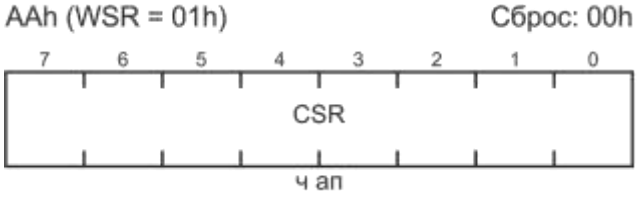
LINPIDRX



Поле	Бит	Описание
PIDRX	5–0	Принятый PID
–	7, 6	Зарезервировано

Таблица А.63 – Регистр контрольной суммы LIN

LINCSR

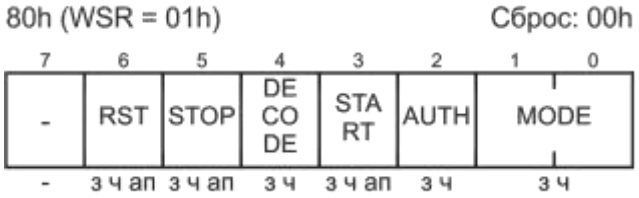


Поле	Бит	Описание
CSR	7–0	Значение принятой контрольной суммы

Регистры блока кодирования по ГОСТ 28147–89

Таблица А.64 – Регистр управления

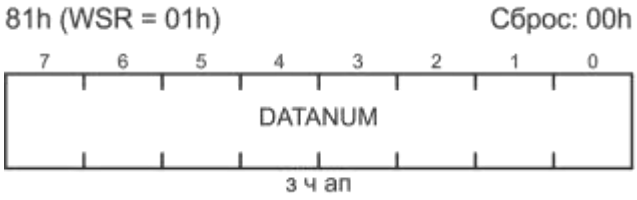
CDCON



Поле	Бит	Описание	
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	
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
–	7	Зарезервировано	

Таблица А.65 – Регистр числа блоков данных

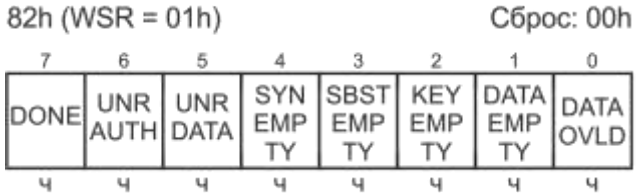
CDDATNUM



Поле	Бит	Описание
DATANUM	7–0	Число 64-разрядных блоков открытых данных, которые подлежат кодированию/декодированию

Таблица А.66 – 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
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

Окончание таблицы А.66

1	2	3
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

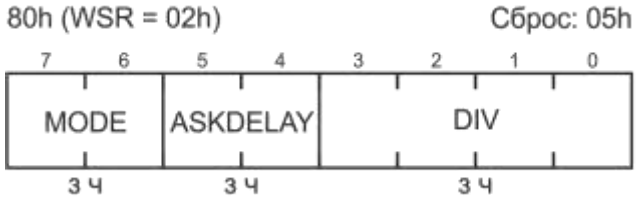
Таблица А.67 – Адреса доступа к 64/256/512-битным регистрам блока кодирования

Мнемоническое название регистра	Разрядность, бит	Адрес доступа к регистру		Обязательное количество последовательных циклов обращения к регистру	Состояние после сброса
CDDATA	64	83h	WSR = 01h	8 (запись/чтение)	00h
CDKEY	256	84h		32 (только запись)	00h
CDSBSTN	512	85h		64 (только запись)	00h
CDUW	64	86h		8 (только запись)	00h
CDAUTH	64	87h		8 (только чтение)	00h

Регистры интерфейса ГОСТ Р 52070–2003

Таблица А.68 – Регистр конфигурации

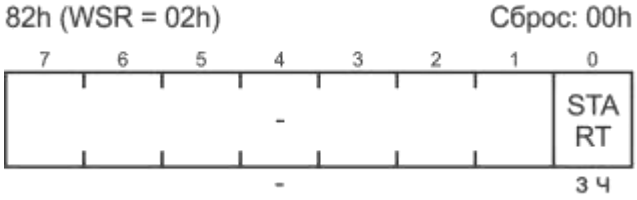
BSICONF1



Поле	Бит	Описание	
MODE	7–6	Режимы работы	
		00	Контроллер шины (КШ)
		01	Оконечное устройство (ОУ)
		10	Зарезервировано
		11	Монитор шины (МШ)
ASKDELAY	5–4	Контроль паузы до ответного слова	
		00	14 мкс
		01	28 мкс
		10	56 мкс
		11	112 мкс
DIV	3–0	Делитель частоты. Предназначен для согласования частоты синхронизации микроконтроллера и частоты передачи информации, которая является стандартной и равна 1 МГц. Значение вычисляется по формуле: DIV = fosc/4, где fosc – тактовая частота микроконтроллера на входе XTAL1 в МГц (желательно, чтобы частота fosc была кратна четырем). Полученное значение делителя следует переводить в шестнадцатеричный формат и записывать в поле DIV	

Таблица А.69 – Дополнительный регистр конфигурации (в режиме контроллера шины)

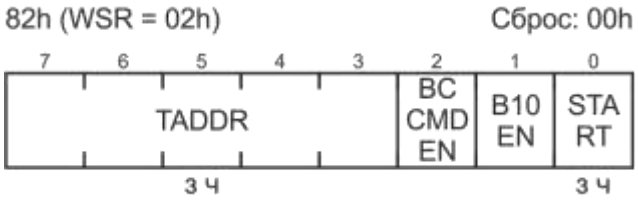
BSICONF2 (КШ)



Поле	Бит	Описание
START	0	Запуск передачи. Очищается аппаратно при остановке любым способом и при изменении режима работы в CONFIG1
–	7–1	Зарезервировано

Таблица А.70 – Дополнительный регистр конфигурации (в режиме оконечного устройства)

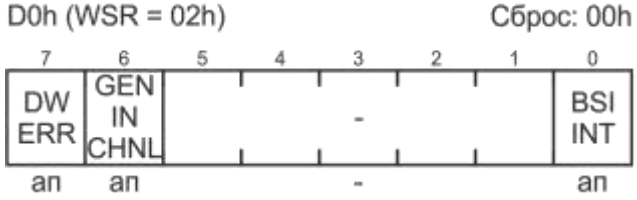
BSICONF2 (УТ)



Поле	Бит	Описание	
TADDR	7–3	Адрес оконечного устройства	
BCCMDEN	2	Разрешение работать с групповым адресом	
		0	Игнорирование адреса 1Fh
		1	Распознавание адреса 1Fh как группового
B10EN	1	Разрешение использования бита 10 как признака слов КС или ОС	
		0	Десятый бит не контролируется
		1	Принятое слово считается командным, если имеет нужный синхроимпульс и десятый принятый бит равен единице
START	0	Включение оконечного устройства. Сбрасывается при изменении режима работы (изменение MODE)	

Таблица А.71 – Регистр состояния прерываний (в режиме контроллера шины)

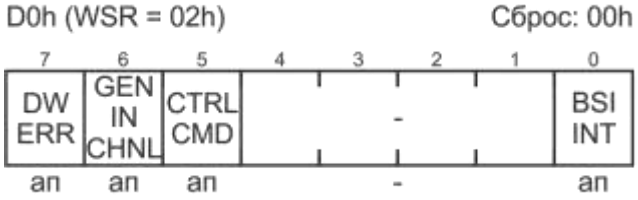
BSIINTSTAT (КШ)



Поле	Бит	Описание
DWERR	7	Ошибка в слове данных
GENINCHNL	6	Генерация в канале
BSIINT	0	Запрос прерывания
–	5–1	Зарезервировано

Таблица А.72 – Регистр состояния прерываний (в режиме оконечного устройства)

BSIINTSTAT (УТ)



Поле	Бит	Описание
1	2	3
DWERR	7	Ошибка в слове данных
GENINCHNL	6	Генерация в канале

Окончание таблицы А.72

1	2	3
CTRLCMD	5	Команда управления
BSIINT	0	Запрос прерывания
–	4–1	Зарезервировано

Таблица А.73 – Регистр состояния прерываний (в режиме монитора шины)

BSIINTSTAT (МШ)

D0h (WSR = 02h) Сброс: 00h

7	6	5	4	3	2	1	0
CHNLN	CODEERR	WLENERR	PARERR	SWMISS	TYPEERR	TRBREAK	BSIINT
an	an	an	an	an	an	an	an

Поле	Бит	Описание
CHNLN	7	Номер канала
CODEERR	6	Ошибка кода
WLENERR	5	Ошибка длины слова
PARERR	4	Ошибка четности
SWMISS	3	Отсутствие ответного слова
TYPEERR	2	Ошибка типа синхроимпульса. Устанавливается, если вместо ожидаемого слова данных получено ответное слово или наоборот
TRBREAK	1	Прерывание передачи
BSIINT	0	Запрос прерывания

Таблица А.74 – Регистр управления

BSICON

84h (WSR = 02h) Сброс: 00h

7	6	5	4	3	2	1	0
CHNLNUM	FRAME	ALTCHNLEN	RPT			-	RESET
3 ч	3 ч	3 ч	3 ч			-	3

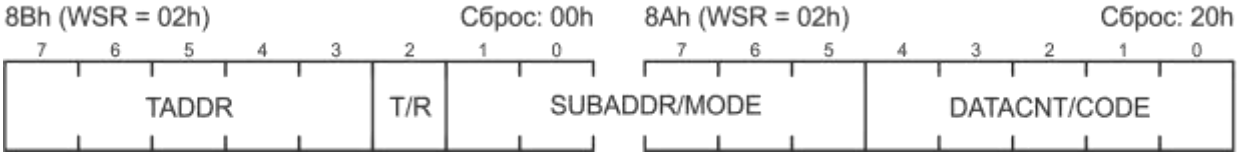
Поле	Бит	Описание	
1	2	3	
CHNLNUM	7	Номер канала	
		0	Основной
		1	Резервный
FRAME	6	Указатель формата сообщения	
		0	Любой формат, кроме форматов 3 и 8
		1	Формат 3 или формат 8
ALTCHNLEN	5	Бит переключения канала при ошибке	
		0	Канал передачи/приема не меняется при повторной передаче
		1	Канал передачи/приема меняется после каждого повторения

Окончание таблицы А.74

1	2	3	
RPT	4–2	Поле количества повторений. Если во время передачи была обнаружена ошибка, сообщение может быть передано повторно	
		000	Повтор запрещен
		001	1
		010	2
		011	4
		100	8
		101	16
		110	31
		111	Повтор передачи до тех пор, пока сообщение не будет передано без ошибок (не будет ошибок в словах данных и установленных битов ошибок в ответном слове)
RESET	0	Сброс. Бит сбрасывается аппаратно	
–	1	Зарезервировано	

Таблица А.75 – Регистры командного слова (первого командного слова для форматов 3 и 8)

BSICMD1_1 **BSICMD1_0**



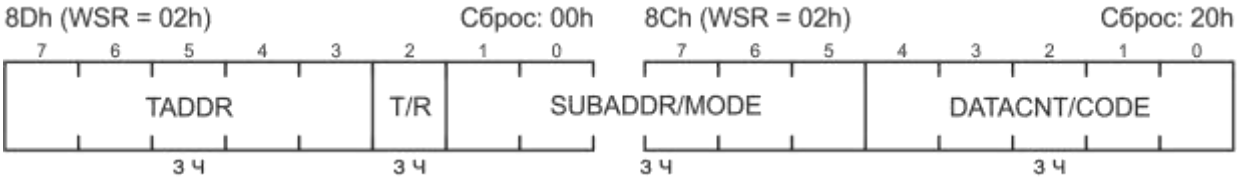
Поле	Бит	Описание	
1	2	3	
TADDR	7–3	Адрес оконечного устройства. В поле записывается адрес оконечного устройства, к которому обращается контроллер шины. Значение 1Fh является групповым адресом и служит для обращения ко всем оконечным устройствам одновременно.	
T/R	2	Направление передачи данных	
		0	Прием
		1	Передача
SUBADDR/MODE	1, 0, 7–5	Подадрес/Управление	
		00h или 1Fh	Режим «Управление». Если в поле записано 00h или 1Fh, то значение в поле DATACNT/CODE является кодом команды управления
		01h–1Eh	Режим «Подадрес». В поле находится адрес подчиненного устройства (абонента), подключенного непосредственно к выбранному оконечному устройству. В этом случае в поле DATACNT/CODE указывается количество слов данных для передачи/приема

Окончание таблицы А.75

1	2	3
DATACNT/CODE	4–0	Количество данных/Код команды управления. Режим работы битового поля устанавливается полем SUBADDR/MODE. Если DATACNT/CODE = 000h, то будут передаваться/приниматься 32 слова данных.

Таблица А.76 – Регистр второго командного слова (только для форматов 3 и 8)

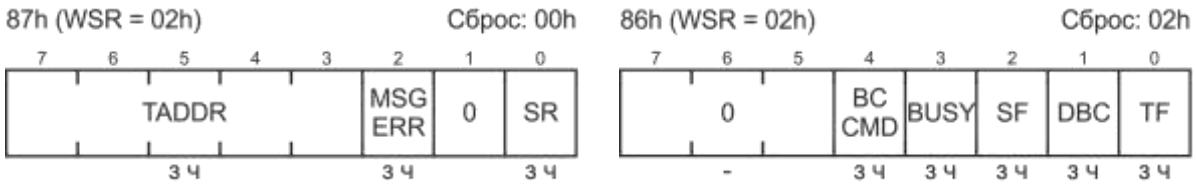
BSICMD2_1 **BSICMD2_0**



Примечание – Назначение битов идентично назначению битов регистра первого командного слова (см. таблицу А.75)

Таблица А.77 – Регистр ответного слова (первого ответного слова для формата 3)

BSISW1_1 (КШ и МШ) **BSISW1_0 (КШ и МШ)**



Поле	Бит	Описание
1	2	3
TADDR	7–3	Собственный адрес оконечного устройства
MSGERR	2	Ошибка в сообщении. Установленный бит указывает на то, что оконечное устройство не смогло осуществить корректный прием
0	1	Бит распознавания (всегда ноль)
SR	0	Запрос на обслуживание
0	7–5	Нулевые биты
BCCMD	4	Индикатор группового сообщения
BUSY	3	Абонент занят
SF	2	Неисправность абонента. Указывает на ошибки в работе подчиненного устройства (абонента) оконечного устройства.
DBC	1	Бит подтверждения принятия управления. Устанавливается аппаратно в случае, если оконечное устройство не было занято и получило команду управления «Принять управление интерфейсом» с кодом 00h

Окончание таблицы А.77

1	2	3
TF	0	Неисправность оконечного устройства. Указывает на ошибки в работе оконечного устройства
Примечание – Ответное слово не передается после приема группового сообщения и в случае обнаружения ошибки в принятых данных.		

Таблица А.78 – Регистры ответного слова (первого ответного слова для формата 3)

BSISW1_1 (УТ)

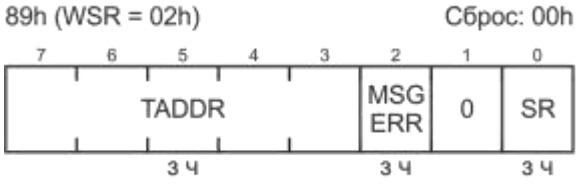
Не используется

BSISW1_0 (УТ)

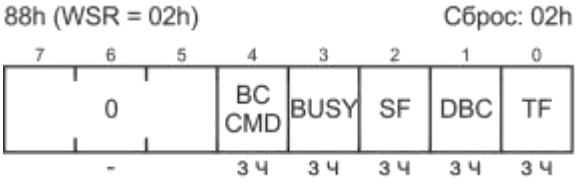
Поле	Бит	Описание
SR	7	Запрос на обслуживание
0	6–4	Нулевые биты
BUSY	3	Абонент занят
SF	2	Неисправность абонента
DBC	1	Бит подтверждения принятия управления
TF	0	Неисправность оконечного устройства
Примечание – Все биты регистра программируются пользователем.		

Таблица А.79 – Регистры второго ответного слова (только для формата 3)

BSISW2_1 (КШ и МШ)

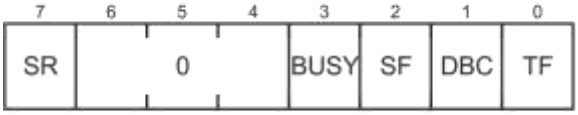


BSISW2_0 (КШ и МШ)



Примечание – Назначение битов идентично назначению битов регистра первого ответного слова (см. таблицу А.78).

Таблица А.80 – Регистр второго ответного слова (только для формата 3)

BSISW2_1 (УТ)	**BSISW2_0 (УТ)**
Не используется	88h (WSR = 02h) Сброс: 02h  Все биты регистра устанавливаются пользователем

Примечание – Назначение битов идентично назначению битов регистра первого ответного слова (см. таблицу А.78).

Приложение Б
(обязательное)

Система команд микроконтроллера

Система команд микроконтроллера предоставляет большие возможности обработки данных, обеспечивает реализацию логических и арифметических вычислений, а также управление в режиме реального времени.

Основные функциональные блоки микроконтроллера (ПЗУ, ОЗУ, регистры специальных функций, АЛУ и внешние шины) имеют 8-разрядную организацию. 16-разрядные данные используются регистром указателя данных DPTR и счетчиком команд PC. Регистр указателя данных может использоваться как 16-разрядный регистр или как два 8-разрядных регистра области SFR – DPH и DPL. Счетчик команд PC всегда используется как 16-разрядный регистр.

Система команд включает в себя 111 команд, которые условно можно разделить на пять групп:

- арифметические операции;
- логические операции;
- операции перемещения;
- операции с битами;
- программные переходы.

Типы адресации операндов команд:

- регистровая;
- прямая;
- косвенно-регистровая;
- непосредственная;
- косвенно-регистровая по сумме базового и индексного регистров.

Методы адресации

Регистровая адресация используется для обращения к восьми рабочим регистрам (R0 – R7) выбранного банка рабочих регистров. Эти же регистры могут быть выбраны с помощью прямой адресации и косвенно-регистровой адресации как обычные ячейки внутреннего ОЗУ данных. Регистровая адресация используется для обращения к регистрам A, B, AB (сдвоенный), DPTR и к флагу переноса C. Использование регистровой адресации позволяет получать двухбайтовый эквивалент трехбайтовых команд прямой адресации.

Прямая побайтная адресация используется для обращения к ячейкам внутреннего ОЗУ данных и к регистрам специального назначения. Прямая побитовая адресация используется для обращения к отдельно адресуемым 128 битам, расположенным в ячейках с адресами от 20h до 2Fh и к отдельно адресуемым битам регистров области SFR. Старший бит байта кода прямого адреса выбирает одну из двух групп отдельно адресуемых битов, расположенных в ОЗУ или регистрах специального назначения (подробнее в разделе 4).

Косвенно-регистровая адресация используется для обращения к ячейкам внутреннего ОЗУ данных. В качестве регистров-указателей используются регистры R0 и R1 выбранного банка регистров. В командах PUSH и POP используется содержимое указателя стека SP. Косвенно-регистровая адресация используется также для обращения к внешней памяти данных. В этом случае с помощью регистров R0 и R1 (выбранного банка регистров) выбирается ячейка из блока внешней памяти данных. Номер блока предварительно задается содержимым порта P2. 16-разрядный указатель данных (DPTR) может быть использован для обращения к любой ячейке адресного пространства внешней памяти данных объемом до 64 Кбайт.

Непосредственная адресация позволяет выбрать из адресного пространства памяти программ константы, явно указанные в команде.

Косвенно-регистровая адресация по сумме базового и индексного регистров (содержимое аккумулятора A) упрощает просмотр таблиц, зашитых в памяти программ. Любой байт из таблицы может быть выбран по адресу, определяемому суммой содержимого DPTR или PC и содержимого A.

Обозначения и символы, используемые в описании команд, приведены в таблице Б.1

Таблица Б.1 – Обозначения и символы, используемые в описании команд

Обозначение, символ	Назначение
1	2
A	Аккумулятор
Rn	Регистр R0 – R7 выбранного банка регистров
n	Номер регистра, принимающий значения от 0 до 7, а при кодировании команд соответственно – от 000b до 111b
direct dir	Прямо адресуемая ячейка памяти данных внутреннего ОЗУ (000h – 07Fh) или области SFR (080h – 0FFh)
@Ri	Косвенно адресуемая ячейка памяти данных внутреннего ОЗУ (000h – 07Fh), адрес которой находится в регистре R0 или R1
i	Номер регистра, принимающий значения 0 или 1, а при кодировании команд соответственно – 0b и 1b
#data	8-битная константа
#data16	16-битная константа
addr16	16-битный адрес. Используется командами LCALL и LJMP. Переход в пределах 64К памяти программ
addr11	11-битный адрес. Используется командами LCALL и LJMP. Переход в пределах 2К памяти программ
rel	Знаковый байт смещения (метка). Используется командой SJMP и командами переходов с условиями. Диапазон смещения от –128 до +127 байт
Bit	Прямо адресуемый бит в памяти данных внутреннего ОЗУ (000h – 07Fh) или области SFR (080h – 0FFh)
<src-byte> <src-bit>	Операнд-источник
<dest-byte> <dest-bit>	Операнд-приемник
<base-reg>	16-разрядный регистр (DPTR или PC)
<byte>	8-разрядный регистр (Rn или direct)

В таблице Б.2 показано влияние команд на флаги переноса и переполнения регистра PSW.

Таблица Б.2 – Команды и флаги регистра PSW, на которые они влияют

Команда	Флаги			Команда	Флаги		
	C	OV	AC		C	OV	AC
ADD	*	*	*	CLR C	0		
ADDC	*	*	*	CPL C	*		
SUBB	*	*	*	ANL C, bit	*		
MUL	0	*		ANL C, /bit	*		
DIV	0	*		ORL C, bit	*		
DA	*			ORL C, /bit	*		
RRC	*			MOV C, bit	*		
RLC	*			CJNE	*		
SETB C	1						

Примечания

1 Флаги C (флаг переноса), OV (флаг переполнения) и AC (флаг вспомогательного переноса) являются битами регистра PSW.

2 Символ «0» указывает на то, что в результате выполнения команды флаг будет сброшен.

3 Символ «1» указывает на то, что в результате выполнения команды флаг будет установлен.

4 Символ «*» указывает на то, что в результате выполнения команды флаг может быть как установлен, так и сброшен.

Перечень команд со всеми возможными комбинациями операндов приведен в таблицах Б.3 – Б.7. В колонках с названием «Периодов XTAL» для некоторых команд время выполнения указано двумя значениями, разделенными косой чертой. Первое значение – это время выполнения команды в режиме нормальной работы ядра, а второе – время выполнения команды в режиме ускоренной работы ядра, который включается после записи значения 05h в регистр FASTM. Выключается режим записью 00h в тот же регистр.

Таблица Б.3 – Перечень команд арифметических операций

Мнемоника		Описание	Кол-во байт	Периодов XTAL
ADD	A, Rn	Сложение регистра с аккумулятором	1	12/6
ADD	A, direct	Сложение ячейки памяти с аккумулятором	2	12
ADD	A, @Ri	Сложение косвенно адресуемой ячейки памяти с аккумулятором	1	12
ADD	A, #data	Сложение константы с аккумулятором	2	12
ADDC	A, Rn	Сложение регистра с аккумулятором с переносом	1	12/6
ADDC	A, direct	Сложение ячейки памяти с аккумулятором с переносом	2	12
ADDC	A, @Ri	Сложение косвенно адресуемой ячейки памяти с аккумулятором с переносом	1	12
ADDC	A, #data	Сложение константы с аккумулятором с переносом	2	12
SUBB	A, Rn	Вычитание регистра из аккумулятора с заемом	1	12/6
SUBB	A, direct	Вычитание ячейки памяти из аккумулятора с заемом	2	12
SUBB	A, @Ri	Вычитание косвенно адресуемой ячейки памяти из аккумулятора с заемом	1	12
SUBB	A, #data	Вычитание константы из аккумулятора	2	12
INC	A	Инкремент аккумулятора	1	12/6
INC	Rn	Инкремент регистра	1	12/6
INC	direct	Инкремент ячейки памяти	2	12
INC	@Ri	Инкремент косвенно адресуемой ячейки памяти	1	12
DEC	A	Декремент аккумулятора	1	12/6
DEC	Rn	Декремент регистра	1	12/6
DEC	direct	Декремент ячейки памяти	2	12
DEC	@Ri	Декремент косвенно адресуемой ячейки памяти	1	12
INC	DPTR	Инкремент регистра указателя данных	1	24/6
MUL	AB	Умножение A и B	1	48/12
DIV	AB	Деление A на B	1	48/12
DA	A	Десятичная коррекция аккумулятора	1	12

Таблица Б.4 – Перечень команд логических операций

Мнемоника		Описание	Кол-во байт	Периодов XTAL
ANL	A, Rn	Логическое «И» аккумулятора с регистром	1	12
ANL	A, direct	Логическое «И» аккумулятора с ячейкой памяти	2	12
ANL	A, @Ri	Логическое «И» аккумулятора с косвенно адресуемой ячейкой памяти	1	12
ANL	A, #data	Логическое «И» аккумулятора с константой	2	12
ANL	direct, A	Логическое «И» ячейки памяти с аккумулятором	2	12
ANL	direct, #data	Логическое «И» ячейки памяти с константой	3	24/18
ORL	A, Rn	Логическое «ИЛИ» аккумулятора с регистром	1	12
ORL	A, direct	Логическое «ИЛИ» аккумулятора с ячейкой памяти	2	12
ORL	A, @Ri	Логическое «ИЛИ» аккумулятора с косвенно адресуемой ячейкой памяти	1	12
ORL	A, #data	Логическое «ИЛИ» аккумулятора с константой	2	12
ORL	direct, A	Логическое «ИЛИ» ячейки памяти с аккумулятором	2	12
ORL	direct, #data	Логическое «ИЛИ» ячейки памяти с константой	3	24/18
XRL	A, Rn	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» аккумулятора с регистром	1	12
XRL	A, direct	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» аккумулятора с ячейкой памяти	2	12
XRL	A, @Ri	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» аккумулятора с косвенно адресуемой ячейкой памяти	1	12
XRL	A, #data	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» аккумулятора с константой	2	12
XRL	direct, A	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» ячейки памяти с аккумулятором	2	12
XRL	direct, #data	Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» ячейки памяти с константой	3	24/18
CLR	A	Очистка аккумулятора	1	12/6
CPL	A	Инверсия аккумулятора	1	12/6
RL	A	Сдвиг содержимого аккумулятора влево	1	12/6
RLC	A	Сдвиг содержимого аккумулятора влево через перенос	1	12/6
RR	A	Сдвиг содержимого аккумулятора вправо	1	12/6
RRC	A	Сдвиг содержимого аккумулятора вправо через перенос	1	12/6
SWAP	A	Обмен тетрадами внутри аккумулятора	1	12/6

Таблица Б.5 – Перечень команд операций перемещения

Мнемоника		Описание	Кол-во байт	Периодов XTAL
MOV	A, Rn	Копирование регистра в аккумулятор	1	12/6
MOV	A, direct	Копирование ячейки памяти в аккумулятор	2	12
MOV	A, @Ri	Копирование косвенно адресуемой ячейки памяти в аккумулятор	1	12/6
MOV	A, #data	Занесение константы в аккумулятор	2	12
MOV	Rn, A	Копирование аккумулятора в регистр	1	12/6
MOV	Rn, direct	Копирование ячейки памяти в регистр	2	24/18
MOV	Rn, #data	Занесение константы в регистр	2	12
MOV	direct, A	Копирование аккумулятора в ячейку памяти	2	12
MOV	direct, Rn	Копирование регистра в ячейку памяти	2	24/12
MOV	direct, direct	Копирование ячейки памяти в ячейку памяти	3	24/18
MOV	direct, @Ri	Копирование косвенно адресуемой ячейки памяти в ячейку памяти	2	24/12
MOV	direct, #data	Занесение константы в ячейку памяти	3	24/18
MOV	@Ri, A	Копирование аккумулятора в косвенно адресуемую ячейку памяти	1	12
MOV	@Ri, direct	Копирование ячейки памяти в косвенно адресуемую ячейку памяти	2	24
MOV	@Ri, #data	Занесение константы в косвенно адресуемую ячейку памяти	2	12
MOV	DPTR, #data16	Занесение 16-разрядной константы в регистр указателя данных	3	24/18
MOVC	A, @A+DPTR	Копирование байта из памяти программ (с адресом DPTR+A) в аккумулятор	1	24/18
MOVC	A, @A+PC	Копирование байта из памяти программ (с адресом PC+A) в аккумулятор	1	24/18
MOVX	A, @Ri	Копирование байта из памяти данных (с 8-битным адресом) в аккумулятор	1	24
MOVX	A, @DPTR	Копирование байта из памяти данных (с 16-битным адресом) в аккумулятор	1	24
MOVX	@Ri, A	Копирование аккумулятора в память данных (8-битный адрес)	1	24
MOVX	@DPTR, A	Копирование аккумулятора в память данных (16-битный адресом)	1	24
PUSH	direct	Запись в стек	2	24/18
POP	direct	Чтение из стека	2	24/12
XCH	A, Rn	Обмен регистра с аккумулятором	1	12
XCH	A, direct	Обмен ячейки памяти с аккумулятором	2	12
XCH	A, @Ri	Обмен косвенно адресуемой ячейки памяти с аккумулятором	1	12
XCHD	A, @Ri	Обмен младшей тетрады косвенно адресуемой ячейки памяти с младшей тетрадой аккумулятора	1	12

Таблица Б.6 – Перечень команд операций с битами

Мнемоника		Описание	Кол-во байт	Периодов XTAL
CLR	C	Очистка переноса	1	12/6
CLR	bit	Очистка бита	2	12
SETB	C	Установка переноса	1	12/6
SETB	bit	Установка бита	2	12
CPL	C	Инверсия переноса	1	12/6
CPL	bit	Инверсия бита	2	12
ANL	C, bit	Логическое «И» бита и переноса	2	24/12
ANL	C, /bit	Логическое «И» инверсии бита и переноса	2	24/12
ORL	C, bit	Логическое «ИЛИ» бита и переноса	2	24/12
ORL	C, /bit	Логическое «ИЛИ» инверсии бита и переноса	2	24/12
MOV	C, bit	Копирование бита в перенос	2	12
MOV	bit, C	Копирование переноса в бит	2	24/12
JC	rel	Переход, если перенос установлен	2	24/18
JNC	rel	Переход, если перенос не установлен	2	24/18
JB	bit, rel	Переход, если бит установлен	3	24
JNB	bit, rel	Переход, если бит не установлен	3	24
JBC	bit, rel	Переход, если бит установлен и очистка бита	3	24

Таблица Б.7 – Перечень команд программных переходов

Мнемоника		Описание	Кол-во байт	Периодов XTAL
ACALL	addr11	Абсолютный вызов подпрограммы	2	24
LCALL	addr16	Длинный вызов подпрограммы	3	24
RET		Возврат из подпрограммы	1	24
RETI		Возврат из прерывания	1	24
AJMP	addr11	Абсолютный переход	2	24/18
LJMP	addr16	Длинный переход	3	24
SJMP	rel	Короткий переход	2	24/18
JMP	@A+DPTR	Косвенный переход	1	24/12
JZ	Rel	Переход, если аккумулятор равен нулю	2	24/18
JNZ	Rel	Переход, если аккумулятор не равен нулю	2	24/18
CJNE	A, direct, rel	Сравнение ячейки памяти с аккумулятором и переход, если не равно	3	24
CJNE	A, #data, rel	Сравнение константы с аккумулятором и переход, если не равно	3	24
CJNE	Rn, #data, rel	Сравнение константы с регистром и переход, если не равно	3	24
CJNE	@Ri, #data, rel	Сравнение константы с косвенно адресуемой ячейкой памяти и переход, если не равно	3	24
DJNZ	Rn, rel	Декремент регистра и переход, если нет равенства нулю	2	24/18
DJNZ	direct, rel	Декремент ячейки памяти и переход, если нет равенства нулю	3	24
NOP		Нет операции	1	12

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	**NOP**	**AJMP** addr11 [2B,2C]	**LJMP** addr16 [3B,2C]	**RR** A	**INC** A	**INC** dir [2B]	**INC** @R0	**INC** @R1	**INC** R0	**INC** R1	**INC** R2	**INC** R3	**INC** R4	**INC** R5	**INC** R6	**INC** R7
1	**JBC** bit,rel [3B,2C]	**ACALL** addr11 [2B,2C]	**LCALL** addr16 [3B,2C]	**RRC** A	**DEC** A	**DEC** dir [2B]	**DEC** @R0	**DEC** @R1	**DEC** R0	**DEC RI** R1	**DEC** R2	**DEC** R3	**DEC** R4	**DEC** R5	**DEC** R6	**DEC** R7
2	**JB** bit,rel [3B,2C]	**AJMP** addr11 [2B,2C]	**RET** [2C]	**RL** A	**ADD** A,#data [2B]	**ADD** dir [2B]	**ADD** A,@R0	**ADD** A,@R1	**ADD** A,R0	**ADD** A,R1	**ADD** A,R2	**ADD** A,R3	**ADD** A,R4	**ADD** A,R5	**ADD** A,R6	**ADD** A,R7
3	**JNB** bit,rel [3B,2C]	**ACALL** addr11 [2B,2C]	**RETI** [2C]	**RLC** A	**ADDC** A,#data [2B]	**ADDC** dir [2B]	**ADDC** A,@R0	**ADDC** A,@R1	**ADDC** A,R0	**ADDC** A,R1	**ADDC** A,R2	**ADDC** A,R3	**ADDC** A,R4	**ADDC** A,R5	**ADDC** A,R6	**ADDC** A,R7
4	**JC** rel [2B,2C]	**AJMP** addr11 [2B,2C]	**ORL** dir,A [2B]	**ORL** dir,#data [3B,2C]	**ORL** A,#data [2B]	**ORL** dir [2B]	**ORL** A,@R0	**ORL** A,@R1	**ORL** A,R0	**ORL** A,R1	**ORL** A,R2	**ORL A,** A,R3	**ORL** A,R4	**ORL** A,R5	**ORL** A,R6	**ORL** A,R7
5	**JNC** rel [2B,2C]	**ACALL** addr11 [2B,2C]	**ANL** dir,A [2B]	**ANL** dir,#data [3B,2C]	**ANL** A,#data [2B]	**ANL** dir [2B]	**XRL** dir [2B]	**MOV** dir,#data [3B,2C]	**ANL** A,R0	**ANL** A,R1	**ANL** A,R2	**ANL** A,R3	**ANL** A,R4	**ANL** A,R5	**ANL** A,R6	**ANL** A,R7
6	**JZ** rel [2B,2C]	**AJMP** addr11 [2B,2C]	**XRL** dir,A [2B]	**XRL** dir,#data [3B,2C]	**XRL** A,#data [2B]	**ANL** A,@R0	**XRL** A,@R0	**MOV** @R0, #data [2B]	**XRL** A,R0	**XRL** A,R1	**XRL** A,R2	**XRL** A,R3	**XRL** A,R4	**XRL** A,R5	**XRL** A,R6	**XRL** A,R7
7	**JNZ** rel [2B,2C]	**ACALL** addr11 [2B,2C]	**ORL** C,bit [2B,2C]	**JMP** @A+ DPTR [2C]	**MOV** A,#data [2B]	**ANL** A,@R1	**XRL** A,@R1	**MOV** @R1, #data [2B]	**MOV** R0, #data [2B]	**MOV** R1, #data [2B]	**MOV** R2, #data [2B]	**MOV** R3, #data [2B]	**MOV** R4, #data [2B]	**MOV** R5, #data [2B]	**MOV** R6, #data [2B]	**MOV** R7, #data [2B]
8	**SJMP** rel [2B,2C]	**AJMP** addr11 [2B,2C]	**ANL** C,bit [2B,2C]	**MOVC** A,@A+ PC [2C]	**DIV** AB [2B,4C]	**MOV** dir,dir [3B,2C]	**MOV** dir,@R0 [2B,2C]	**MOV** dir,@R1 [2B,2C]	**MOV** dir,R0 [2B,2C]	**MOV** dir,R1 [2B,2C]	**MOV** dir,R2 [2B,2C]	**MOV** dir,R3 [2B,2C]	**MOV** dir,R4 [2B,2C]	**MOV** dir,R5 [2B,2C]	**MOV** dir,R6 [2B,2C]	**MOV** dir,R7 [2B,2C]
9	**MOV** DPTR, #data16 [3B,2C]	**ACALL** addr11 [2B,2C]	**MOV** bit,C [2B,2C]	**MOVC** A,@A+ DPTR [2C]	**SUBB** A,#data [2B,4C]	**SUBB** A,dir [2B]	**SUBB** A,@R0	**SUBB** A,@R1	**SUBB** A,R0	**SUBB** A,R1	**SUBB** A,R2	**SUBB** A,R3	**SUBB** A,R4	**SUBB** A,R5	**SUBB** A,R6	**SUBB** A,R7
A	**ORL** C,/bit [2B,2C]	**AJMP** addr11 [2B,2C]	**MOV** C,bit [2B]	**INC** DPTR [2C]	**MUL** AB [4C]		**MOV** @R0,dir [2B,2C]	**MOV** @R1,dir [2B,2C]	**MOV** R0,dir [2B,2C]	**MOV** R1,dir [2B,2C]	**MOV** R2,dir [2B,2C]	**MOV** R3,dir [2B,2C]	**MOV** R4,dir [2B,2C]	**MOV** R5,dir [2B,2C]	**MOV** R6,dir [2B,2C]	**MOV** R7,dir [2B,2C]
B	**ANL** C,/bit [2B,2C]	**ACALL** addr11 [2B,2C]	**CPL** bit [2B]	**CPL** C	**CJNE** A, #data, rel [3B,2C]	**CJNE** A, dir, rel [3B,2C]	**CJNE** @R0, #data, rel [3B,2C]	**CJNE** @R1, #data, rel [3B,2C]	**CJNE** R0, #data, rel [3B,2C]	**CJNE** R1, #data, rel [3B,2C]	**CJNE** R2, #data, rel [3B,2C]	**CJNE** R3, #data, rel [3B,2C]	**CJNE** R4, #data, rel [3B,2C]	**CJNE** R5, #data, rel [3B,2C]	**CJNE** R6, #data, rel [3B,2C]	**CJNE** R7, #data, rel [3B,2C]
C	**PUSH** dir [2B,2C]	**AJMP** addr11 [2B,2C]	**CLR** bit [2B]	**CLR** C	**SWAP** A	**XCH** A,dir [2B]	**XCH** A,@R0	**XCH** A,@R1	**XCH** A,R0	**XCH** A,R1	**XCH** A,R2	**XCH** A,R3	**XCH** A,R4	**XCH** A,R5	**XCH** A,R6	**XCH** A,R7
D	**POP** dir [2B,2C]	**ACALL** addr11 [2B,2C]	**SETB** bit [2B]	**SETB** C	**DA** A	**DJNZ** dir,rel [3B,2C]	**XCHD** A,@R0	**XCHD** A,@R1	**DJNZ** R0,rel [2B,2C]	**DJNZ** R1,rel [2B,2C]	**DJNZ** R2,rel [2B,2C]	**DJNZ** R3,rel [2B,2C]	**DJNZ** R4,rel [2B,2C]	**DJNZ** R5,rel [2B,2C]	**DJNZ** R6,rel [2B,2C]	**DJNZ** R7,rel [2B,2C]
E	**MOVX** A,@DPTR [2C]	**AJMP** addr11 [2B,2C]	**MOVX** A,@R0 [2C]	**MOVX** A,@Ri [2C]	**CLR** A	**MOV** A,dir [2B]	**MOV** A,@R0	**MOV** A,@R1	**MOV** A,R0	**MOV** A,R1	**MOV** A,R2	**MOV** A,R3	**MOV** A,R4	**MOV** A,R5	**MOV** A,R6	**MOV** A,R7
F	**MOVX** @DPTR,A [2C]	**ACALL** addr11 [2B,2C]	**MOVX** @R0,A [2C]	**MOVX** @R1,A [2C]	**CPL** A	**MOV** dir,A [2B]	**MOV** @R0,A	**MOV** @R,A	**MOV** R0,A	**MOV** R1,A	**MOV** R2,A	**MOV** R3,A	**MOV** R4,A	**MOV** R5,A	**MOV** R6. A	**MOV** R7,A

Рисунок Б.1 – Сводная таблица кодов команд с мнемоникой команд

На рисунке Б.1 представлена сводная таблица кодов команд операций с мнемоникой команд. Символ [2B] или [3B] указывает на длину команды, равную двум или трем байтам соответственно, символ [2C] или [4C] указывает на длительность выполнения команды, равную двум или четырем циклам соответственно. Для однобайтовых команд и времени выполнения, равного одному циклу, указания длины и времени отсутствуют.

Для того, чтобы найти по известному коду команду, следует сначала найти строку под номером, совпадающим с шестнадцатеричным значением первого байта кода, а затем – столбец с номером, совпадающим с шестнадцатеричным значением второго байта кода. Так, например, команда, имеющая код 73h – это JMP с операндом @A+DPTR.

Описание команд

ACALL addr11

Функция: Абсолютный вызов подпрограммы

Код:

a10	a9	a8	1	0	0	0	1	a7	a6	a5	a4	a3	a2	a1	a0

Описание: Вызов подпрограммы, размещенной по указанному адресу (addr11 – метка размещения подпрограммы). При этом счетчик команд увеличивается на два для получения адреса следующей команды, после чего полученное 16-разрядное значение PC помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на два. Адрес перехода (страницы) получается с помощью конкатенации старших бит (7, 6 и 5 биты Кода) увеличенного содержимого счетчика команд и младшего байта команды (второй байт Кода).

Байт: 2
Циклов: 2

Алгоритм: ACALL
(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP)) ← (PC7 – 0)
(SP) ← (SP) + 1
((SP)) ← (PC15 –8)
(PC_{10-0}) ← адрес страницы

ADD A, <src-byte>

Функция: Сложение операнда-источника с аккумулятором.

Описание: Команда прибавляет к содержимому операнда-источника содержимое аккумулятора и заносит результат в аккумулятор.

При появлении переноса из бита 7 аккумулятора флаг переноса C устанавливается, в противном случае – сбрасывается. При сложении целых чисел без знака флаг переноса C выполняет роль флага переполнения.

При появлении переноса из бита 3 аккумулятора флаг дополнительного переноса AC устанавливается, в противном случае – сбрасывается.

При появлении переноса из одного из двух бит аккумулятора – седьмого или шестого – флаг переполнения OV устанавливается. При одновременном появлении переносов из седьмого и шестого бит флаг OV сбрасывается. При сложении целых чисел со знаком флаг OV указывает на:

- отрицательный результат суммирования разнознаковых чисел;
- положительный результат суммирования отрицательных чисел.

В качестве операнда-источника могут выступать:

- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data).

ADD A, Rn

Функция: Сложение регистра с аккумулятором

Код:

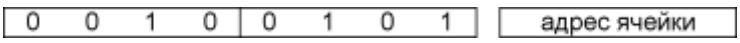
Байт: 1
Циклов: 1

Алгоритм: ADD
(A) ← (A) + (Rn)

ADD A, direct

Функция: Сложение ячейки памяти с аккумулятором

Код:



Байт: 2
Циклов: 1

Алгоритм: ADD
(A) ← (A) + (direct)

ADD A, @Ri

Функция: Сложение косвенно адресуемой ячейки памяти с аккумулятором

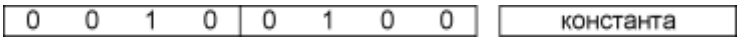
Код: 

Байт: 1
Циклов: 1

Алгоритм: ADD
(A) ← (A) + ((Ri))

ADD A, #data

Функция: Сложение константы с аккумулятором

Код: 

Байт: 2
Циклов: 1

Алгоритм: ADD
(A) ← (A) + #data

ADDC A, <src-byte>

Функция: Сложение операнда-источника с аккумулятором с переносом
Описание: Команда прибавляет к содержимому операнда-источника содержимое аккумулятора и заносит результат в аккумулятор.
При появлении переноса из бита 7 аккумулятора флаг переноса C устанавливается, в противном случае – сбрасывается. При сложении целых чисел без знака флаг переноса C выполняет роль флага переполнения.
При появлении переноса из бита 3 аккумулятора флаг дополнительного переноса AC устанавливается, в противном случае – сбрасывается.
При появлении переноса из одного из двух бит аккумулятора – седьмого или шестого – флаг переполнения OV устанавливается. При одновременном появлении переносов из седьмого и шестого бит флаг OV сбрасывается. При сложении целых чисел со знаком флаг OV указывает на:
- отрицательный результат суммирования разнознаковых чисел;
- положительный результат суммирования отрицательных чисел.

В качестве операнда-источника могут выступать:
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data).

ADDC A, Rn

Функция: Сложение регистра с аккумулятором

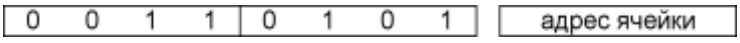
Код: 

Байт: 1
Циклов: 1

Алгоритм: ADDC
(A) ← (A) + (C) + (Rn)

ADDC A, direct

Функция: Сложение ячейки памяти с аккумулятором

Код: 

Байт: 2
Циклов: 1

Алгоритм: ADDC
(A) ← (A) + (C) + (direct)

ADDC A, @Ri

Функция: Сложение косвенно адресуемой ячейки памяти с аккумулятором

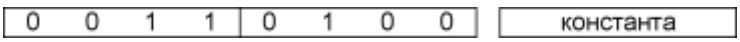
Код: 

Байт: 1
Циклов: 1

Алгоритм: ADDC
(A) ← (A) + (C) + ((Ri))

ADDC A, #data

Функция: Сложение константы с аккумулятором

Код: 

Байт: 2
Циклов: 1

Алгоритм: ADDC
(A) ← (A) + (C) + #data

AJMP addr11

Функция: Абсолютный переход

Код:

a10	a9	a8	0	0	0	0	1	a7	a6	a5	a4	a3	a2	a1	a0

Описание: Передает управление команде, расположенной по указанному адресу, который получается конкатенацией пяти старших бит счетчика команд PC (после увеличения его на два), битов 7, 6 и 5 Кода команды и младшего байта Кода. Адрес перехода должен находиться внутри страницы памяти программы объемом 2К, определяемой пятью старшими битами счетчика команд.

Байт: 2
Циклов: 2

Алгоритм: AJMP
(PC) ← (PC) + 2
(PC_{10-0}) ← адрес страницы

ANL <dest-byte>, <src-byte>

Функция: Логическое «И» операнда-приемника с операндом-источником
Описание: Побитно логически складывает операнды и размещает результат в операнде-приемнике.

Примечание – Когда команда ANL применяется для изменения состояния порта, значение, которое будет использоваться в качестве операнда, берется с выходной защелки порта, а не с выводов порта.
В качестве операндов могут выступать:
- аккумулятор (A);
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data)
- бит (bit).

ANL A, Rn

Функция: Логическое «И» аккумулятора с регистром

Код:

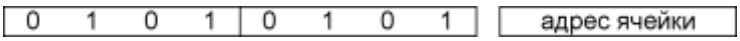
0	1	0	1	1	n

Байт: 1
Циклов: 1

Алгоритм: ANL
(A) ← (A) Λ (Rn)

ANL A, direct

Функция: Логическое «И» ячейки памяти с аккумулятором

Код: 

Байт: 2
Циклов: 1

Алгоритм: ANL
(A) ← (A) Λ (direct)

ANL A, @Ri

Функция: Логическое «И» косвенно адресуемой ячейки памяти с аккумулятором

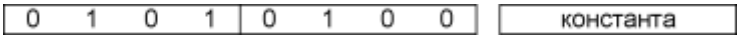
Код: 

Байт: 1
Циклов: 1

Алгоритм: ANL
(A) ← (A) Λ ((Ri))

ANL A, #data

Функция: Логическое «И» константы с аккумулятором

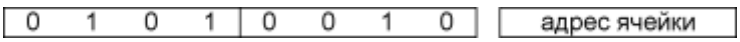
Код: 

Байт: 2
Циклов: 1

Алгоритм: ANL
(A) ← (A) Λ #data

ANL direct, A

Функция: Логическое «И» ячейки памяти с аккумулятором

Код: 

Байт: 2
Циклов: 1

Алгоритм: ANL
(direct) ← (direct) Λ (A)

ANL direct, #data

Функция: Логическое «И» ячейки памяти с константой

Код:

Байт: 3
Циклов: 2

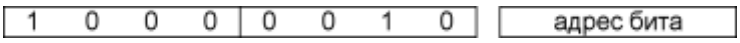
Алгоритм: ANL
(direct) ← (direct) Λ #data

ANL C, <src-bit>

Функция: Логическое «И» бита-источника и переноса
Описание: Логически складывает прямо адресуемый бит-источник и перенос и сбрасывает флаг переноса C в случае, если бит-источник равен нулю. Если бит-источник равен единице, состояние флага C остается без изменений.
Косая черта («/»), используемая в мнемонике перед битовым операндом-источником указывает на то, что команда ANL выполняет логическое сложение инверсного значения прямо адресуемого бита-источника и переноса. При этом сам бит-источник остается без изменений.

ANL C, bit

Код:

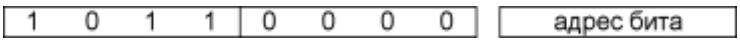


Байт: 2
Циклов: 2

Алгоритм: ANL
(C) ← (C) Λ (bit)

ANL C, /bit

Код:



Байт: 2
Циклов: 2

Алгоритм: ANL
(C) ← (C) Λ ¯|(bit)

CJNE <dest-byte>, <src-byte>, rel

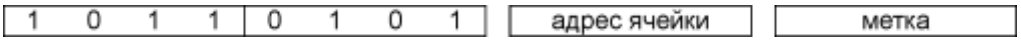
Функция: Сравнение операндов и переход на метку, если не равно

Описание: Сравнивает значение операнда-источника с операндом-приемником и в случае несовпадения выполняет переход по адресу, на который указывает метка. Адрес перехода вычисляется при помощи сложения знакового значения, указанного в последнем байте команды, с содержимым счетчика команд после увеличения его на три.
Если беззнаковое значение байта-приемника меньше, беззнакового значения байта-источника, то устанавливается флаг переноса C.
В случае, если значения операндов равны, флаг переноса C сбрасывается.
Команда не изменяет состояния байта-приемника и байта-источника.

В качестве операндов могут выступать:
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data).

CJNE A, direct, rel

Код:



Байт: 3
Циклов: 2

Алгоритм:
(PC) ← (PC) + 3
Если (A) ≠ (direct)
Тогда
    (PC) ← (PC) + смещение
Если (A) < (direct)
Тогда
    (C) ← 1
Иначе
    (C) ← 0

CJNE A, #data, rel

Код:

Байт: 3
Циклов: 2

Алгоритм: (PC) ← (PC) + 3
Если (A) ≠ data
Тогда
(PC) ← (PC) + смещение
Если (A) < data
Тогда
(C) ← 1
Иначе
(C) ← 0

CJNE Rn, #data, rel

Код:

Байт: 3
Циклов: 2

Алгоритм: (PC) ← (PC) + 3
Если (Rn) ≠ data
Тогда
(PC) ← (PC) + смещение
Если (Rn) < data
Тогда
(C) ← 1
Иначе
(C) ← 0

CJNE @Ri, #data, rel

Код:

Байт: 3
Циклов: 2

Алгоритм: (PC) ← (PC) + 3
Если ((Ri)) ≠ data
Тогда
(PC) ← (PC) + смещение
Если ((Ri)) < data
Тогда
(C) ← 1
Иначе
(C) ← 0

CLR A

Функция: Очистка аккумулятора

Код:

Описание: Очищает аккумулятор, записывая во все его биты «0»

Байт: 1
Циклов: 1

Алгоритм: CLR
(A) ← 0

CLR bit

Функция: Очистка бита
Описание: Очищает указанный бит, записывая в него «0»

В качестве операнда могут выступать:
- флаг переноса C;
- прямо адресуемый бит (bit).

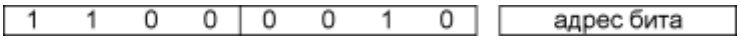
CLR C

Код: 

Байт: 1
Циклов: 1

Алгоритм: CLR
(C) ← 0

CLR bit

Код: 

Байт: 2
Циклов: 1

Алгоритм: CLR
(bit) ← 0

CPL A

Функция: Инверсия аккумулятора

Код: 

Описание: Побитно инвертирует значение, хранящееся в аккумуляторе

Байт: 1
Циклов: 1

Алгоритм: CPL
(A) ← ¯|(A)

CPL bit

Функция: Инверсия бита
Описание: Инвертирует значение указанного бита

В качестве операнда могут выступать:
- флаг переноса C;
- прямо адресуемый бит (bit).

CPL C

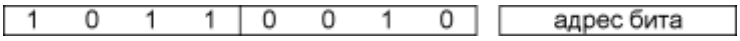
Код:

Байт: 1
Циклов: 1

Алгоритм: CPL
(C) ← ¯|(C)

CPL bit

Код:



Байт: 2
Циклов: 1

Алгоритм: CPL
(bit) ← ¯|(bit)

DA A

Функция: Десятичная коррекция аккумулятора

Код:

Описание: Корректирует 8-разрядное значение, находящееся в аккумуляторе, которое в свою очередь является результатом сложения (команды ADD и ADDC) двух переменных двоично-десятичного формата.

Если значение младшей тетрады аккумулятора превышает девять (т.е. в аккумуляторе находится число из диапазона от 0Ah до FFh) или если установлен флаг AC, то к аккумулятору прибавляется число 6, преобразуя, таким образом, формат младшей тетрады аккумулятора в двоично-десятичный. Если в результате этого сложения возникает перенос из младшей тетрады в старшую, и этот перенос проходит через все биты старшей тетрады, то устанавливается флаг переноса C, в противном случае флаг переноса C остается без изменений. Далее, если значение старшей тетрады аккумулятора превышает девять (т.е. в аккумуляторе находится число из диапазона от A0h до FFh) или если установлен флаг переноса C, то к старшей тетраде аккумулятора прибавляется число 6, преобразуя, таким образом, формат тетрады в двоично-десятичный. Если в результате такого сложения возникает перенос, то устанавливается флаг переноса C, в противном случае флаг переноса C остается без изменений. Таким образом, флаг переноса C является индикатором того, что сумма двух исходных двоично-десятичных чисел превышает значение 100, позволяя провести сложение чисел в десятичном формате с высокой точностью.

Фактически, для десятичной коррекции используются четыре числа – 00h, 06h, 60h и 66h, – одно из которых, в зависимости от значения, находящегося в аккумуляторе и состояния флага переноса C, складывается с аккумулятором. Следует помнить, что команда DA A не может конвертировать шестнадцатеричное число в двоично-десятичное и не может применяться при вычитании десятичных значений.

Пример:
Нужно сложить два десятичных числа: 56 и 67. Фактически 56 + 67 = 123. Рассмотрим алгоритм вычисления.
В аккумуляторе записано число 56h, являющееся двоично-десятичным представлением десятичного числа 56, в регистре R3 записано число 67h, являющееся двоично-десятичным представлением десятичного числа 67.
Порядок команд следующий:

```
ADD   A, R3
DA    A
```

В результате выполнения команды ADD в аккумулятор запишется значение BDh, являющееся суммой шестнадцатеричных чисел 56h и 67h.
В результате выполнения команды DA в аккумулятор запишется значение 23h (фактически это двоично-десятичное представление десятичного числа 23), являющееся суммой шестнадцатеричных чисел BDh и 66h (число 66h добавляется, поскольку значение каждой из тетрад аккумулятора превышает 9). Также будет выставлен флаг переноса C, указывающий на то, что сумма складываемых чисел (56 и 67) превышает 100. Таким образом, 100 + 23 = 123

Байт: 1
Циклов: 1

Алгоритм: DA
- содержимое аккумулятора является двоично-десятичным числом
Если $[[(A_{3-0}) > 9] \vee [(AC) = 1]]$
Тогда
$(A_{3-0}) \leftarrow (A_{3-0}) + 6$
И
Если $[[(A_{7-4}) > 9] \vee [(C) = 1]]$
Тогда
$(A_{7-4}) \leftarrow (A_{7-4}) + 6$

DEC byte

Функция: Декремент операнда
Описание: Уменьшает значение операнда на 1. В случае, если операнд был равен 00h, его значение становится 0FFh.

В качестве операнда могут выступать:
- аккумулятор (A);
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri)

DEC A

Функция: Декремент аккумулятора

Код:

Байт: 1
Циклов: 1

Алгоритм: DEC
(A) ← (A) – 1

DEC Rn

Функция: Декремент регистра

Код:

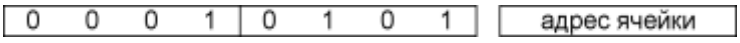
Байт: 1
Циклов: 1

Алгоритм: DEC
(Rn) ← (Rn) – 1

DEC direct

Функция: Декремент ячейки памяти

Код:



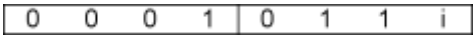
Байт: 2
Циклов: 1

Алгоритм: DEC
(direct) ← (direct) – 1

DEC @Ri

Функция: Декремент косвенно адресуемой ячейки памяти

Код:



Байт: 1
Циклов: 1

Алгоритм: DEC
((Ri)) ← ((Ri)) – 1

DIV AB

Функция: Деление

Код:

1	0	0	0	0	1	0	0

Описание: Беззнаковое деление 8-разрядного числа, находящегося в аккумуляторе A на 8-разрядное число, находящееся в регистре B с занесением целой части результата деления в аккумулятор, а остатка в регистр B со сбросом флагов переполнения OV и переноса C.
Флаг дополнительного переноса AC не сбрасывается, если (A) < (B).

В случае, если в регистре B изначально находится значение 00h, то после деления в аккумуляторе и регистре B будут находиться случайные значения. При этом будет выставлен флаг переполнения OV.

Пример:
В аккумуляторе находится значение 251 (FBh или 11111011b) и в регистре B – значение 18 (12h или 00010010b). Порядок команд:

```
DIV   AB
```

После выполнения команды деления в аккумуляторе будет значение 13 (0Dh или 00001101b), а в регистре B значение 17 (11h или 00010001b), поскольку 251 = (13 × 18) + 17. Флаги C и OV сброшены.

Байт: 1
Циклов: 4

Алгоритм: DIV
$(A)_{15\text{-}8} \leftarrow (A)/(B)$
$(B)_{7\text{-}0}$

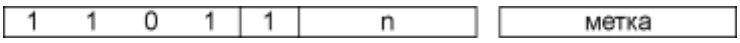
DJNZ <byte>, <rel-addr>

Функция: Декремент операнда и переход на метку, если операнд не равен нулю
Описание: Уменьшает значение операнда на единицу, сравнивает полученный результат с нулем и в случае неравенства осуществляет переход на указанную метку. Если значение операнда было 00h, то после вычитания оно станет равно FFh. Адрес перехода вычисляется сложением знакового смещения, указанного в последнем байте команды, с содержимым счетчика команд, увеличенным на длину команды DJNZ.

В качестве операнда могут выступать:
- регистр (Rn);
- ячейка памяти (direct).

DJNZ Rn, rel

Код:

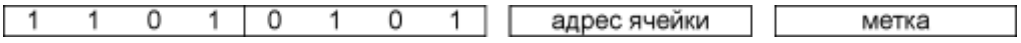


Байт: 2
Циклов: 2

Алгоритм: DJNZ
(PC) ← (PC) + 2
(Rn) ← (Rn) – 1
Если (Rn) > 0 or (Rn) < 0
Тогда
(PC) ← (PC) + rel

DJNZ direct, rel

Код:



Байт: 3
Циклов: 2

Алгоритм: DJNZ
(PC) ← (PC) + 2
(direct) ← (direct) – 1
Если (direct) > 0 or (direct) < 0
Тогда
(PC) ← (PC) + rel

INC <byte>

Функция: Инкремент
Описание: Увеличивает значение операнда на единицу. Если значение операнда было FFh, то после увеличения оно станет равно 00h.
В качестве операнда могут выступать:
- аккумулятор
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri)

INC A

Функция: Инкремент аккумулятора

Код: 

Байт: 1
Циклов: 1

Алгоритм: INC
(A) ← (A) + 1

INC Rn

Функция: Инкремент регистра

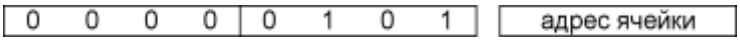
Код: 

Байт: 1
Циклов: 1

Алгоритм: INC
(Rn) ← (Rn) + 1

INC direct

Функция: Инкремент ячейки памяти

Код: 

Байт: 2
Циклов: 1

Алгоритм: INC
(direct) ← (direct) + 1

INC @Ri

Функция: Инкремент косвенно адресуемой ячейки памяти

Код: 

Байт: 1
Циклов: 1

Алгоритм: INC
((Ri)) ← ((Ri)) + 1

INC DPTR

Функция: Инкремент регистра указателя данных

Код:

1	0	1	0	0	0	1	1

Описание: Увеличивает значение 16-разрядного регистра указателя данных на единицу. Регистр DP0H/DP1H старшего байта указателя данных и регистр DP0L/DP1L младшего байта указателя данных функционируют как один 16-разрядный регистр-счетчик, поэтому переполнение (достижение значения FFh и переключение в 00h) регистра DP0L/DP1L вызовет увеличение на единицу значения регистра DP0H/DP1H.

Байт: 1
Циклов: 2

Алгоритм: INC
(DPTR) ← (DPTR) + 1

JB bit, rel

Функция: Переход, если бит установлен

Код:

0	0	1	0	0	0	0	0	адрес бита	метка

Описание: Проверяет указанный бит на равенство его значения единице. Если бит установлен («1»), то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из третьего байта команды к содержимому счетчика команд после прибавления к нему 3.

Байт: 3
Циклов: 2

Алгоритм: JB
(PC) ← (PC) + 3
Если (bit) = 1
Тогда
(PC) ← (PC) + rel

JBC bit, rel

Функция: Переход, если бит установлен и очистка бита

Код:

Описание: Проверяет указанный бит на равенство его значения единице. Если бит установлен («1»), то бит очищается и осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из третьего байта команды к содержимому счетчика команд после прибавления к нему 3.

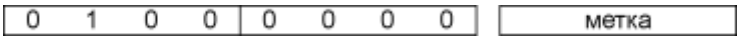
Байт: 3
Циклов: 2

Алгоритм: JBC
(PC) ← (PC) + 3
Если (bit) = 1
Тогда
(bit) ← 0
(PC) ← (PC) + rel

JC rel

Функция: Переход, если перенос установлен

Код:



Описание: Проверяет флаг переноса C на равенство его значения единице. Если флаг установлен («1»), то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из второго байта команды к содержимому счетчика команд после прибавления к нему 2.

Байт: 2
Циклов: 2

Алгоритм: JC
(PC) ← (PC) + 2
Если (C) = 1
Тогда
(PC) ← (PC) + rel

JMP @A+DPTR

Функция: Косвенный переход

Код:

Описание: Складывает беззнаковое содержимое аккумулятора с 16-разрядным указателем данных и загружает полученный результат (адрес) в счетчик команд (PC) для выборки следующей команды. Содержимое аккумулятора и указателя данных не изменяется.

Байт: 1

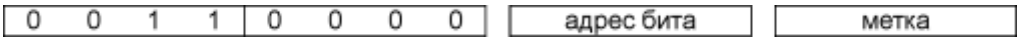
Циклов: 2

Алгоритм: JMP
(PC) ← (A) + (DPTR)

JNB bit, rel

Функция: Переход, если бит не установлен

Код:



Описание: Проверяет указанный бит на равенство его значения нулю. Если бит не установлен («0»), то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из третьего байта команды к содержимому счетчика команд после прибавления к нему 3.

Байт: 3

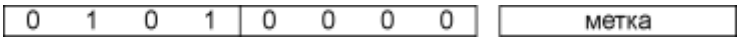
Циклов: 2

Алгоритм: JNB
(PC) ← (PC) + 3
Если (bit) = 0
Тогда
(PC) ← (PC) + rel

JNC rel

Функция: Переход, если перенос не установлен

Код:



Описание: Проверяет флаг переноса C на равенство его значения нулю. Если флаг не установлен («0»), то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из второго байта команды к содержимому счетчика команд после прибавления к нему 2.

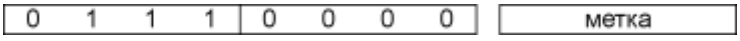
Байт: 2
Циклов: 2

Алгоритм: JNC
(PC) ← (PC) + 2
Если (C) = 0
Тогда
(PC) ← (PC) + rel

JNZ rel

Функция: Переход, если аккумулятор не равен нулю

Код:



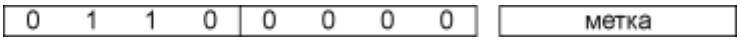
Описание: Проверяет содержимое регистра аккумулятора на равенство его значения нулю. Если хотя бы один бит регистра аккумулятора равен единице, то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из второго байта команды к содержимому счетчика команд после прибавления к нему 2.
Содержимое аккумулятора остается без изменения.

Байт: 2
Циклов: 2

Алгоритм: JNZ
(PC) ← (PC) + 2
Если (A) ≠ 0
Тогда
(PC) ← (PC) + rel

JZ rel

Функция: Переход, если аккумулятор равен нулю

Код: 

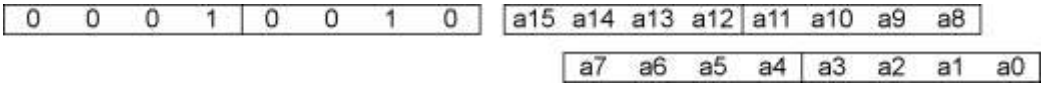
Описание: Проверяет содержимое регистра аккумулятора на равенство его значения нулю. Если все биты регистра аккумулятора равны нулю, то осуществляется переход на указанную метку. Адрес ветвления вычисляется с помощью прибавления относительного знакового смещения из второго байта команды к содержимому счетчика команд после прибавления к нему 2.
Содержимое аккумулятора остается без изменения.

Байт: 2
Циклов: 2

Алгоритм: JZ
(PC) ← (PC) + 2
Если (A) = 0
Тогда
(PC) ← (PC) + rel

LCALL addr16

Функция: Длинный вызов подпрограммы

Код: 

Описание: Вызов подпрограммы, размещенной по указанному адресу (addr16 – метка размещения подпрограммы). Счетчик команд увеличивается на 3 для получения адреса следующей команды, после чего 16-битовое значение счетчика команд помещается в стек (сначала следует младший байт), и содержимое указателя стека также увеличивается на два. Вызов подпрограммы осуществляется в пределах 64 Кбайт адресного пространства памяти программ.

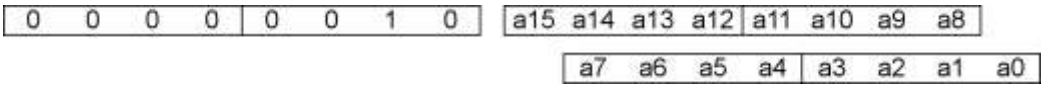
Байт: 3
Циклов: 2

Алгоритм: LCALL
(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP)) ← (PC_{7-0})
(SP) ← (SP) + 1
((SP)) ← (PC_{15-8})
(PC) ← $адрес_{15-0}$

LJMP addr16

Функция: Длинный переход

Код:



Описание: Длинный переход по указанному адресу (на метку addr16). В старший байт счетчика команд помещается второй байт Кода, а в младший – третий байт Кода. После загрузки адреса осуществляется переход в пределах 64 Кбайт адресного пространства памяти программ.

Байт: 3
Циклов: 2

Алгоритм: LJMP
(PC) ← адрес$_{15\text{-}0}$

MOV <dest-byte>, <src-byte>

Функция: Копирование операнда-источника в операнд-приемник.
Описание: Копирует содержимое операнда-источника в операнд-приемник, без изменения содержимого операнда-источника.

В качестве операндов могут выступать:
- аккумулятор (A);
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data).

MOV A, Rn

Функция: Копирование регистра в аккумулятор

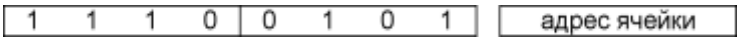
Код:

Байт: 1
Циклов: 1

Алгоритм: MOV
(A) ← (Rn)

MOV A, direct

Функция: Копирование ячейки памяти в аккумулятор

Код: 

Байт: 2
Циклов: 1

Алгоритм: MOV
(A) ← (direct)

MOV A, @Ri

Функция: Копирование косвенно адресуемой ячейки памяти в аккумулятор

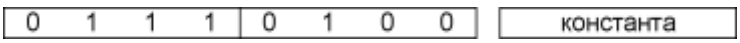
Код: 

Байт: 1
Циклов: 1

Алгоритм: MOV
(A) ← ((Ri))

MOV A, data

Функция: Занесение константы в аккумулятор

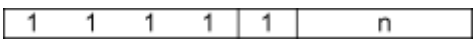
Код: 

Байт: 2
Циклов: 1

Алгоритм: MOV
(A) ← #data

MOV Rn, A

Функция: Копирование аккумулятора в регистр

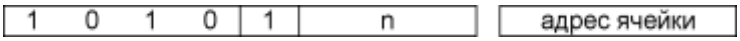
Код: 

Байт: 1
Циклов: 1

Алгоритм: MOV
(Rn) ← (A)

MOV Rn, direct

Функция: Копирование ячейки памяти в регистр

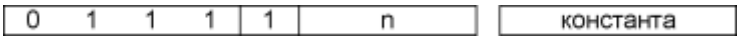
Код: 

Байт: 2
Циклов: 2

Алгоритм: MOV
(Rn) ← (direct)

MOV Rn, #data

Функция: Занесение константы в регистр

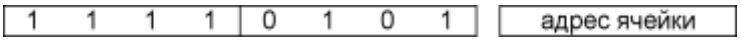
Код: 

Байт: 2
Циклов: 1

Алгоритм: MOV
(Rn) ← #data

MOV direct, A

Функция: Копирование аккумулятора в ячейку памяти

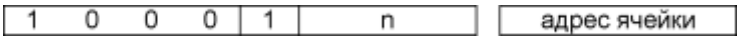
Код: 

Байт: 2
Циклов: 1

Алгоритм: MOV
(direct) ← (A)

MOV direct, Rn

Функция: Копирование регистра в ячейку памяти

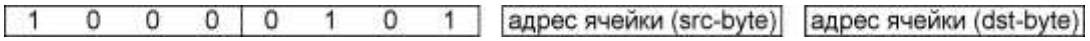
Код: 

Байт: 2
Циклов: 2

Алгоритм: MOV
(direct) ← (Rn)

MOV direct, direct

Функция: Копирование ячейки памяти в ячейку памяти

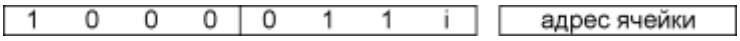
Код: 

Байт: 3
Циклов: 2

Алгоритм: MOV
(direct) ← (direct)

MOV direct, @Ri

Функция: Копирование косвенно адресуемой ячейки памяти в ячейку памяти

Код: 

Байт: 2
Циклов: 2

Алгоритм: MOV
(direct) ← ((Ri))

MOV direct, #data

Функция: Занесение константы в ячейку памяти

Код: 

Байт: 3
Циклов: 2

Алгоритм: MOV
(direct) ← #data

MOV @Ri, A

Функция: Копирование аккумулятора в косвенно адресуемую ячейку памяти

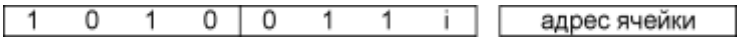
Код: 

Байт: 1
Циклов: 1

Алгоритм: MOV
((Ri)) ← (A)

MOV @Ri, direct

Функция: Копирование ячейки памяти в косвенно адресуемую ячейку памяти

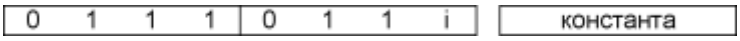
Код: 

Байт: 2
Циклов: 2

Алгоритм: MOV
((Ri)) ← (direct)

MOV @Ri, #data

Функция: Занесение константы в косвенно адресуемую ячейку памяти

Код: 

Байт: 2
Циклов: 1

Алгоритм: MOV
((Ri)) ← #data

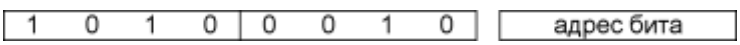
MOV <dest-bit>, <src-bit>

Функция: Копирование бита-источника в бит-приемник
Описание: Копирует прямо адресуемый бит-источник в прямо адресуемый бит-приемник, без изменения содержимого бита-источника.

В качестве операнда могут выступать:
- бит переноса (C);
- прямо адресуемый бит (bit).

MOV C, bit

Функция: Копирование бита в перенос

Код: 

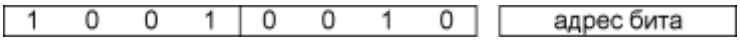
Байт: 2
Циклов: 1

Алгоритм: MOV
(C) ← (bit)

MOV bit, C

Функция: Копирование переноса в бит

Код:



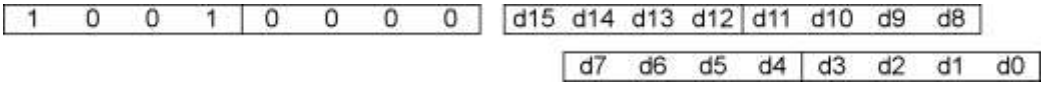
Байт: 2
Циклов: 2

Алгоритм: MOV
(bit) ← (C)

MOV DPTR, #data16

Функция: Занесение 16-разрядной константы в регистр указателя данных

Код:



Описание: Заносит 16-разрядную константу в регистр указателя данных DPTR. В старший байт DP0H/DP1H заносится второй байт Кода, а в младший байт DP0L/DP1L – третий байт Кода

Байт: 3
Циклов: 2

Алгоритм: MOV
(DPTR) ← #data$_{15\text{-}0}$
DPH ← #data$_{15\text{-}8}$
DPL ← #data$_{7\text{-}0}$

MOVC A, @A+<base-reg>

Функция: Копирование байта из памяти программ в аккумулятор

Описание: Копирует байт кода или константу из памяти программ в аккумулятор. Адрес копируемого байта получается сложением беззнакового значения аккумулятора и 16-разрядного значения операнда <base-reg>. Если в качестве операнда <base-reg> выступает счетчик команд PC, то его значение сначала увеличивается до значения адреса следующей команды (т.е. инкрементируется на единицу), а затем складывается с аккумулятором.

В качестве операндов могут выступать:
- аккумулятор (A);
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);

- константа (#data)

MOVC A, @A+DPTR

Функция: Копирование байта из памяти программ (с адресом DPTR+A) в аккумулятор

Код:

1	0	0	1	0	0	1	1

Байт: 1
Циклов: 2

Алгоритм: MOVC
(A) ← ((A) + (DPTR))

MOVC A, @A+PC

Функция: Копирование байта из памяти программ (с адресом PC+A) в аккумулятор

Код:

1	0	0	0	0	0	1	1

Байт: 1
Циклов: 2

Алгоритм: MOVC
(PC) ← (PC) + 1
(A) ← ((A) + (PC))

MOVX <dest-byte>, <src-byte>

Функция: Копирование байта из памяти данных в аккумулятор
Описание: Копирует байт из памяти данных в аккумулятор. В зависимости от адресации (8- или 16-разрядная) используется один из двух типов команды MOVX.

Первый тип – 8-разрядная адресация. Регистр R0 или R1 выбранного банка регистров содержит адрес, мультиплексируемый данными порта 0. Для памяти небольшого объема достаточно использовать 8-разрядный адрес. Если объем памяти превышает 256 байт, то для расширения адреса можно использовать любые выводы порта, значения которых можно задать командой, предшествующей команде MOVX.

Второй тип – 16-разрядная адресация. Через порт 2 выводится старший байт адреса (содержимое регистра DP0H/DP1H), а через порт 0 мультиплексно выводится младший байт адреса (содержимое регистра DP0L/DP1L) и данные. Регистр P2 порта 2 хранит записанное ранее значение в то время, как выходные буферы порта 2 содержат значение регистра старшего байта

указателя данных DP0H/DP1H. Подобная организация позволяет быстро и легко адресовать любую ячейку памяти данных в пределах 64 Кбайт, без использования дополнительных команд.

MOVX A, @Ri

Функция: Копирование байта из памяти данных (с 8-разрядным адресом) в аккумулятор

Код:

1	1	1	0	0	0	1	i

Байт: 1
Циклов: 2

Алгоритм: MOVX
(A) ← ((Ri))

MOVX A, @DPTR

Функция: Копирование байта из памяти данных (с 16-разрядным адресом) в аккумулятор

Код:

1	1	1	0	0	0	0	0

Байт: 1
Циклов: 2

Алгоритм: MOVX
(A) ← (DPTR)

MOVX @Ri, A

Функция: Копирование аккумулятора в память данных (с 8-разрядным адресом)

Код:

1	1	1	1	0	0	1	i

Байт: 1
Циклов: 2

Алгоритм: MOVX
((Ri)) ← (A)

MOVX @DPTR, A

Функция: Копирование аккумулятора в память данных (с 16-разрядным адресом)

Код:

1	1	1	1	0	0	0	0

Байт: 1
Циклов: 2

Алгоритм: MOVX
(DPTR) ← (A)

MUL AB

Функция: Умножение A на B

Код:

1	0	1	0	0	1	0	0

Описание: Умножает беззнаковое 8-разрядное содержимое аккумулятора на содержимое регистр B. Младший байт 16-разрядного результата помещается в аккумулятор, а старший байт – в регистр B. Если произведение превысило значение 255 (FFh), устанавливается флаг переполнения OV, в противном случае флаг сбрасывается. Флаг переноса C всегда сбрасывается

Байт: 1
Циклов: 4

Алгоритм: MUL
$(A)_{7\text{-}0} \leftarrow (A)\times(B)$
$(B)_{15\text{-}8}$

NOP

Функция: Нет операции

Код:

0	0	0	0	0	0	0	0

Описание: Никакие действия не выполняются. Счетчик команд инкрементируется на единицу, и управление переходит к следующей команде
Байт: 1
Циклов: 1

Алгоритм: NOP
(PC) ← (PC) + 1

ORL <dest-byte>, <src-byte>

Функция: Логическое «ИЛИ» операнда-приемника с операндом-источником
Описание: Побитно логически умножает операнды и размещает результат в операнде-приемнике.

Примечание – Когда команда ORL применяется для изменения состояния порта, значение, которое будет использоваться в качестве операнда, берется с выходной защелки порта, а не с выводов порта.

В качестве операндов могут выступать:
- аккумулятор (A);
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data)
- бит (bit).

ORL A, Rn

Функция: Логическое «ИЛИ» аккумулятора с регистром

Код:

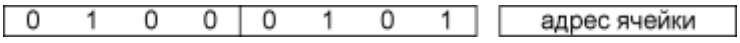
Байт: 1
Циклов: 1

Алгоритм: ORL
(A) ← (A) V (Rn)

ORL A, direct

Функция: Логическое «ИЛИ» ячейки памяти с аккумулятором

Код:



Байт: 2
Циклов: 1

Алгоритм: ORL
(A) ← (A) V (direct)

ORL A, @Ri

Функция: Логическое «ИЛИ» косвенно адресуемой ячейки памяти с аккумулятором

Код:

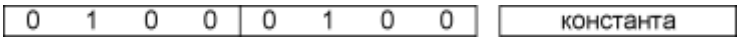
Байт: 1
Циклов: 1

Алгоритм: ORL
(A) ← (A) V ((Ri))

ORL A, #data

Функция: Логическое «ИЛИ» константы с аккумулятором

Код:



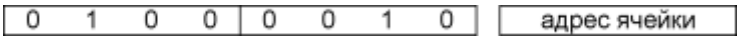
Байт: 2
Циклов: 1

Алгоритм: ORL
(A) ← (A) V #data

ORL direct, A

Функция: Логическое «ИЛИ» ячейки памяти с аккумулятором

Код:



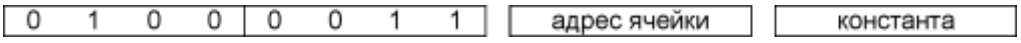
Байт: 2
Циклов: 1

Алгоритм: ORL
(direct) ← (direct) V (A)

ORL direct, #data

Функция: Логическое «ИЛИ» ячейки памяти с константой

Код:



Байт: 3
Циклов: 2

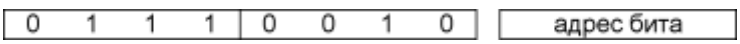
Алгоритм: ORL
(direct) ← (direct) V #data

ORL C, <src-bit>

Функция: Логическое «ИЛИ» бита-источника и переноса
Описание: Логически умножает прямо адресуемый бит-источник и перенос и устанавливает флаг переноса C в случае, если результат равен единице. В остальных случаях состояние флага C остается без изменений.
Косая черта («/»), используемая в мнемонике перед битовым операндом-источником, указывает на то, что команда ORL выполняет логическое сложение инверсного значения прямо адресуемого бита-источника и переноса. При этом сам бит-источник остается без изменений

ORL C, bit

Код:

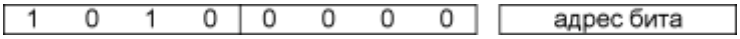


Байт: 2
Циклов: 2

Алгоритм: ORL
(C) ← (C) V (bit)

ORL C, /bit

Код:



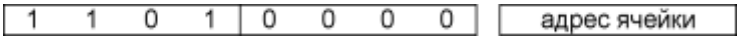
Байт: 2
Циклов: 2

Алгоритм: ORL
(C) ← (C) V ¯|(bit)

POP direct

Функция: Чтение из стека

Код:



Описание: Копирует содержимое ячейки внутренней ОЗУ, адресуемой указателем стека в ячейку памяти, адрес которой берется из второго байта Кода. Далее указатель стека декрементируется на единицу

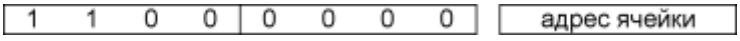
Байт: 2
Циклов: 2

Алгоритм: POP
(direct) ← ((SP))
(SP) ← (SP) – 1

PUSH direct

Функция: Запись в стек

Код:



Описание: Инкрементирует указатель стека на единицу. Далее ячейка памяти, адрес которой указан во втором байте Кода, копируется в ячейку внутренней ОЗУ, адресуемую указателем стека

Байт: 2
Циклов: 2

Алгоритм: PUSH
(SP) ← (SP) + 1

((SP)) ← (direct)

RET

Функция: Возврат из подпрограммы

Код:

0	0	1	0	0	0	1	0

Описание: Последовательно считывает и записывает в счетчик команд PC старший и младший байты адреса, с которого далее продолжается выполнение программы, а также последовательно декрементирует указатель стека SP на два.

Байт: 2
Циклов: 2

Алгоритм: RET
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$
$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RETI

Функция: Возврат из прерывания

Код:

0	0	1	1	0	0	1	0

Описание: Последовательно считывает и записывает в счетчик команд PC старший и младший байты адреса, с которого далее продолжается выполнение программы, а также последовательно декрементирует указатель стека SP на два. Переводит логику прерываний в состояние, в котором разрешаются прерывания с таким же уровнем приоритета. Регистр PSW аппаратно не переводится в состояние, в котором он находился до прерывания. Выполнение программы продолжается с команды, идущей непосредственно за той, которая вызвала прерывание. Если во время выполнения команды RETI возник запрос на прерывание с меньшим или таким же приоритетом, то сначала будет завершено выполнение команды RETI, а затем обслужен запрос

Байт: 1
Циклов: 2

Алгоритм: RETI
$(PC_{15-8}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

$(PC_{7-0}) \leftarrow ((SP))$
$(SP) \leftarrow (SP) - 1$

RL A

Функция: Сдвиг содержимого аккумулятора влево

Код:

0	0	1	0	0	0	1	1

Описание: Сдвигает влево на одну позицию все восемь бит аккумулятора, при этом седьмой бит перебрасывается в нулевой

Байт: 1
Циклов: 1

Алгоритм: RL
$(A_{n+1}) \leftarrow (A_n)$, где n = 0 – 6
$(A_0) \leftarrow (A_7)$

RLC A

Функция: Сдвиг содержимого аккумулятора влево через перенос

Код:

0	0	1	1	0	0	1	1

Описание: Сдвигает влево на одну позицию все восемь бит аккумулятора, при этом седьмой бит перебрасывается в перенос, а перенос – в нулевой бит

Байт: 1
Циклов: 1

Алгоритм: RLC
$(A_{n+1}) \leftarrow (A_n)$, где n = 0 – 6
$(A_0) \leftarrow (C)$
$(C) \leftarrow (A_7)$

RR A

Функция: Сдвиг содержимого аккумулятора вправо

Код:

0	0	0	0	0	0	1	1

Описание: Сдвигает вправо на одну позицию все восемь бит аккумулятора, при этом нулевой бит перебрасывается в седьмой

Байт: 1
Циклов: 1

Алгоритм: RR
$(A_n) \leftarrow (A_{n+1})$, где n = 0 – 6
$(A_7) \leftarrow (A_0)$

RRC A

Функция: Сдвиг содержимого аккумулятора вправо через перенос

Код:

0	0	0	1	0	0	1	1

Описание: Сдвигает вправо на одну позицию все восемь бит аккумулятора, при этом нулевой бит перебрасывается в перенос, а перенос – в седьмой бит.

Байт: 1
Циклов: 1

Алгоритм: RLC
$(A_n) \leftarrow (A_{n+1})$, где n = 0 – 6
$(A_7) \leftarrow (C)$
$(C) \leftarrow (A_0)$

SETB <bit>

Функция: Установка бита
Описание: Устанавливает указанный бит в единицу. Команда оперирует только с прямо адресуемыми битами и с переносом

SETB C

Код:

1	1	0	1	0	0	1	1

Байт: 1
Циклов: 1

Алгоритм: SETB
$(C) \leftarrow 1$

SETB bit

Код:

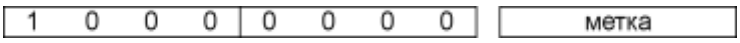
1	1	0	1	0	0	1	0	адрес бита

Байт: 2
Циклов: 1

Алгоритм: SETB
(bit) ← 1

SJMP rel

Функция: Короткий переход

Код: 

Описание: Переход по указанному адресу (на метку rel). Счетчик команд инкрементируется на два и к его значению добавляется знаковое значение из второго байта Кода. Диапазон перехода от –128 байт до 127 байт относительно расположения команды SJMP.

Байт: 2
Циклов: 2

Алгоритм: SJMP
(PC) ← (PC) + 2
(PC) ← (PC) + rel

SUBB A, <src-byte>

Функция: Вычитание операнда-источника из аккумулятора с заемом

Описание: Вычитает содержимое операнда-источника и переноса (если требуется) из содержимого аккумулятора и заносит результат в аккумулятор. При необходимости заема для бита 7 аккумулятора, флаг переноса C устанавливается, в противном случае – сбрасывается. Если флаг переноса C был установлен до начала выполнения команды SUBB, (что указывает на то, что заем был необходим на предыдущем шаге вычитания), то перенос вычитается из аккумулятора вместе с операндом-источником.
При необходимости заема для бита 3 аккумулятора, флаг вспомогательного переноса AC устанавливается, в противном случае – сбрасывается.
При необходимости заема для одного из двух бит аккумулятора – 7 или 6 – устанавливается флаг переполнения OV.
При вычитании целых чисел со знаком флаг OV указывает на:
- отрицательный результат при вычитании отрицательного числа из положительного;
- положительный результат при вычитании положительного числа из отрицательного.

В качестве операнда-источника могут выступать:
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data).

SUBB A, Rn

Функция: Вычитание регистра из аккумулятора с заемом

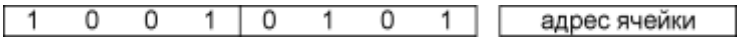
Код: 

Байт: 1
Циклов: 1

Алгоритм: SUBB
(A) ← (A) – (C) – (Rn)

SUBB A, direct

Функция: Вычитание ячейки памяти из аккумулятора с заемом

Код: 

Байт: 2
Циклов: 1

Алгоритм: SUBB
(A) ← (A) – (C) – (direct)

SUBB A, @Ri

Функция: Вычитание косвенно адресуемой ячейки памяти из аккумулятора с заемом

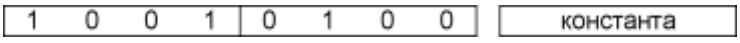
Код: 

Байт: 1
Циклов: 1

Алгоритм: SUBB
(A) ← (A) – (C) – ((Ri))

SUBB A, #data

Функция: Вычитание константы из аккумулятора с заемом

Код: 

Байт: 2
Циклов: 1

Алгоритм: SUBB
(A) ← (A) – (C) – #data

SWAP A

Функция: Обмен тетрадами внутри аккумулятора

Код:

1	1	0	0	0	1	0	0

Описание: Меняет местами старшую и младшую тетрады аккумулятора

Байт: 1
Циклов: 1

Алгоритм: SWAP
(A_{3-0}) D (A_{7-4})

XCH A, <byte>

Функция: Обмен операнда с аккумулятором
Описание: Загружает в аккумулятор содержимое байта-операнда и в то же время записывает в байт-операнд содержимое аккумулятора, обменивая, таким образом, их содержимое

В качестве байта-операнда могут выступать:
- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri)

XCH A, Rn

Функция: Обмен регистра с аккумулятором.

Код:

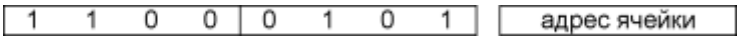
1	1	0	0	1	n

Байт: 1
Циклов: 1

Алгоритм: XCH
(A) D (Rn)

XCH A, direct

Функция: Обмен ячейки памяти с аккумулятором
Код:



Байт: 2
Циклов: 1

Алгоритм: XCH
(A) D (direct)

XCH A, @Ri

Функция: Обмен косвенно адресуемой ячейки памяти с аккумулятором

Код: 

Байт: 1
Циклов: 1

Алгоритм: XCH
(A) D ((Ri))

XCHD A, @Ri

Функция: Обмен тетрадами косвенно адресуемой ячейки памяти и аккумулятора

Код: 

Описание: Меняет местами младшую тетраду аккумулятора (как правило, определяющую шестнадцатеричный или двоично-десятичный формат числа) и младшую тетраду косвенно адресуемой ячейки памяти. Старшие тетрады остаются без изменений

Байт: 1
Циклов: 1

Алгоритм: XCHD
(A_{3-0}) D ((Ri_{7-4}))

XRL <dest-byte>, <src-byte>

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» операнда-приемника с операндом-источником

Описание: Побитно складывает по модулю два операнда и размещает результат в операнде-приемнике

Примечание – Когда команда XRL применяется для изменения состояния порта, значение, которое будет использоваться в качестве операнда, берется с выходной защелки порта, а не с выводов порта

В качестве операндов могут выступать:
- аккумулятор (A);

- регистр (Rn);
- ячейка памяти (direct);
- косвенно адресуемая ячейка памяти (@Ri);
- константа (#data)
- бит (bit)

XRL A, Rn

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» аккумулятора с регистром

Код:

0	1	1	0	1	n

Байт: 1
Циклов: 1

Алгоритм: XRL
(A) ← (A) $\underline{V}$ (Rn)

XRL A, direct

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» ячейки памяти с аккумулятором

Код:

0	1	1	0	0	1	0	1	адрес ячейки

Байт: 2
Циклов: 1

Алгоритм: XRL
(A) ← (A) $\underline{V}$ (direct)

XRL A, @Ri

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» косвенно адресуемой ячейки памяти с аккумулятором

Код:

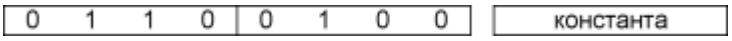
0	1	1	0	0	1	1	i

Байт: 1
Циклов: 1

Алгоритм: XRL
(A) ← (A) $\underline{V}$ ((Ri))

XRL A, #data

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» константы с аккумулятором
Код:

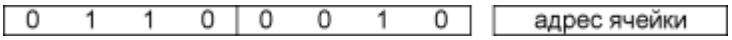


Байт: 2
Циклов: 1

Алгоритм: XRL
(A) ← (A) V #data

XRL direct, A

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» ячейки памяти с аккумулятором

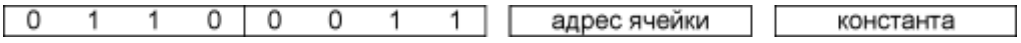
Код: 

Байт: 2
Циклов: 1

Алгоритм: XRL
(direct) ← (direct) V (A)

XRL direct, #data

Функция: Логическое «ИСКЛЮЧАЮЩЕЕ ИЛИ» ячейки памяти с константой

Код: 

Байт: 3
Циклов: 2

Алгоритм: XRL
(direct) ← (direct) V #data

Приложение В
(обязательное)

Коды состояний контроллера интерфейса 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 [ADR,0]	1	0	0	0	- передать код мастера и перейти в режим HS (0Ch/ 21h); - передать адрес ведомого (04h/ 05h)
02h	Повторный старт	[ADR,0] [ADR,1]	1	0	0	0	- передать адрес ведомого (04h/ 05h); - 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 1 1	0 0 0	0 1 1	1 0 1	- сделать повторный старт (02h); - остановить передачу (00h); - остановить передачу, а затем сделать повторный старт (01h)

Таблица В.4 – Режим FS мастера передатчика (дополнительно см. таблицу В.2)

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 1 1	0 0 0	0 1 1	1 0 1	- сделать повторный старт (02h); - остановить передачу (00h); - остановить передачу, а затем сделать повторный старт (01h)

Таблица В.5 – Режим FS ведомого приемника (дополнительно см. таблицу В.7)

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 1	0 1	0 0	0 0	- получить байт данных, квитировать прием (12h); - получить байт данных, не квитировать прием (13h)
11h	Принят адрес после потери арбитража и квити-рован	–	1 1	0 1	0 0	0 0	- получить байт данных, квитировать прием (12h); - получить байт данных, не квитировать прием (13h)
12h	Принят байт данных и квити-рован	DAT	1 1	0 1	0 0	0 0	- получить байт данных, квитировать прием (12h); - получить байт данных, не квитировать прием (13h)
13h	Принят байт данных и не квити-рован	DAT	1 1	0 0	0 0	0 1	- функционировать в режиме безадресного ведомого (00h); - функционировать в режиме безадресного 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 1	X X	0 0	0 1	- функционировать в режиме безадресного ведомого (00h); - 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 1	X X	0 0	0 1	- функционировать в режиме безадресного ведомого (00h); - функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)

Таблица В.7 – Режим FS ведомого приемника (дополнительно см. таблицу В.5)

Код	Описание состояния	Регистр SMBSDA	Биты регистра SMBCTRL1				Возможные дальнейшие действия и коды результатов их выполнения
			CLRST	ACK	STOP	START	
1Ch	Стоп	–	1 1	0 0	0 0	0 1	- функционировать в режиме безадресного ведомого (00h); - функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)
1Dh	Принят адрес общего вызова и квити-рован	–	1 1	0 1	0 0	0 0	- получить байт данных, квитировать прием (12h); - получить байт данных, не квитировать прием (13h)
1Eh	Принят адрес общего вызова после потери арбитража и квити-рован	–	1 1	0 1	0 0	0 0	- получить байт данных, квитировать прием (12h); - 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] [ADR,1]	1	0	0	0	- передать адрес ведомого (28h/ 29h); - 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 1	X X	0 0	0 1	- функционировать в режиме безадресного ведомого (00h); - функционировать в режиме безадресного ведомого, сделать старт после освобождения шины (00h, 01h)

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Подп.	Дата
	измененных	замененных	новых	аннулированных				
-	-	-	-	-	230			09.12.13
1	1	-	-	-	-			24.12.13
2	-	8, 9, 105	-	-	-			22.01.14
3	-	5	-	-	-			09.04.14
4	-	8	-	-	-			17.04.14
5	-	8, 9, 69	-	-	-			05.05.14
6	-	145	-	-	-			15.12.14
7	-	5, 7, 94	-	-	-			27.04.15
8	-	117	-	-	-			13.05.15