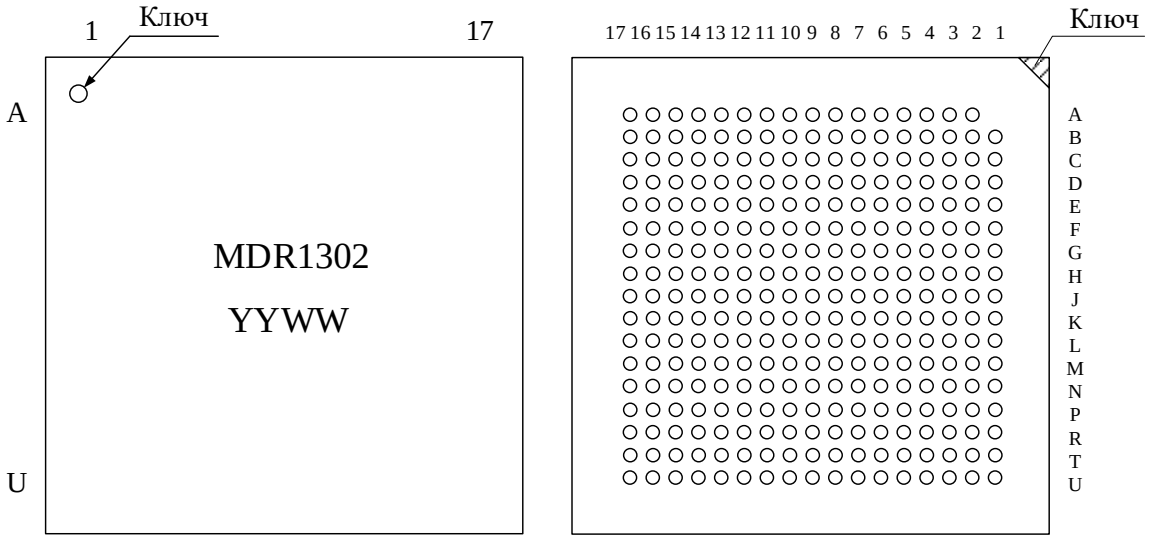
Микросхема высокопроизводительного процессора цифровой обработки сигнала с суперскалярной архитектурой К1967ВН04BG



Лицевая сторона Обратная сторона

YY – год выпуска
WW – неделя выпуска

Основные параметры микросхемы:

- Два последовательных порта;
- Разрядность внешней шины данных 32 бит;
- Разрядность внешней шины адреса 22 бит;
- Напряжение питания ввода/вывода U_{CCIO} от 3,0 до 3,6 В;
- Динамический ток потребления схем ввода/вывода I_{OCC} не более 800 мА;
- Тактовая частота процессора 230 МГц;
- Рабочий диапазон температур от минус 40 °С до плюс 85 °С.

Тип корпуса:

- 288-выводной пластмассовый корпус BGA288.

Масса микросхем не более 2 г.

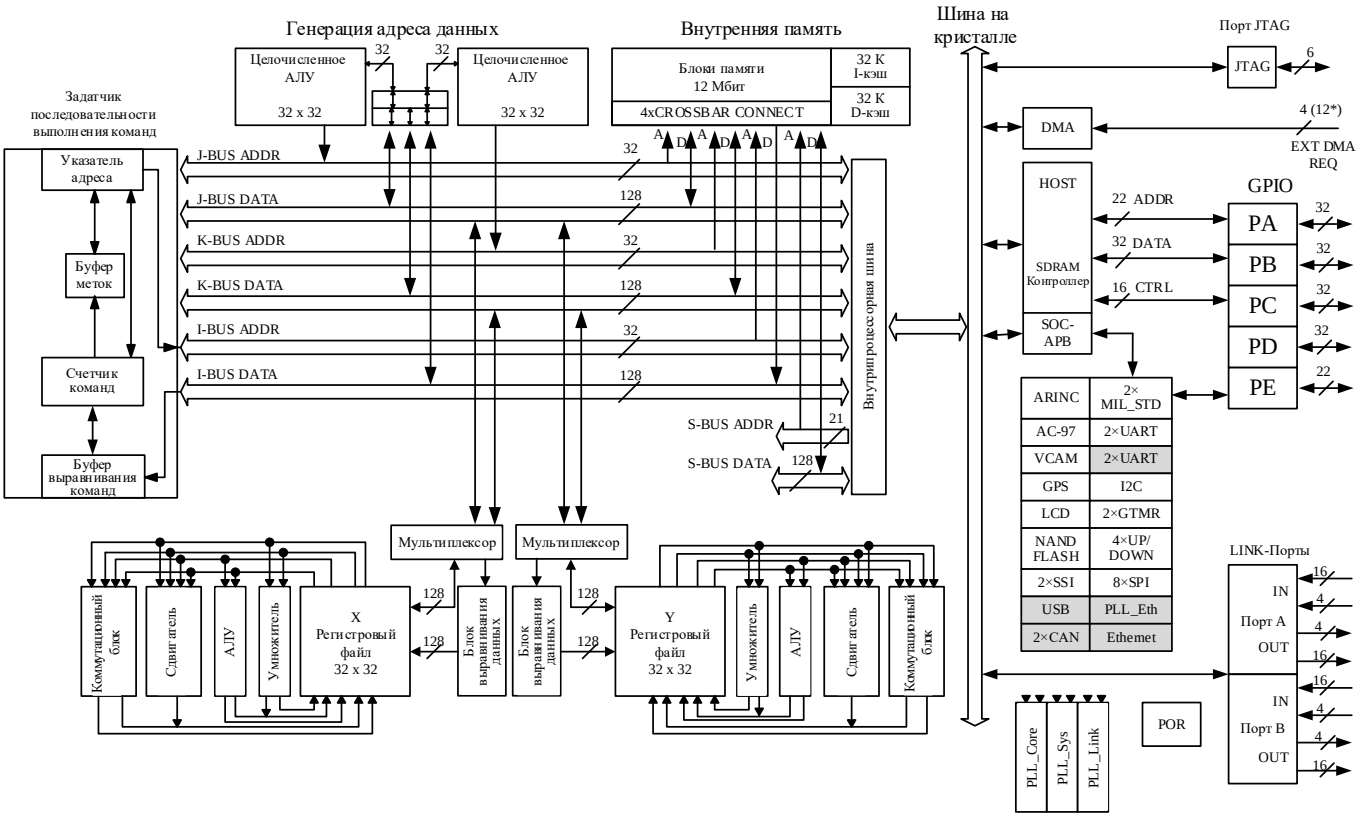
Общее описание и области применения микросхемы

Микросхемы интегральные К1967ВН04BG (далее – микросхемы) предназначены для использования в аппаратуре в качестве процессора цифровой обработки сигналов с ОЗУ 12 Мбит и тактовой частотой 230 МГц.

Важно: спецификация действительна совместно с документом К1967ВН04BG Errata Notice.

Содержание

1 Структурная блок-схема микросхемы



– Блоки, добавленные в микросхемах с ревизии 3

* Для микросхем с ревизии 3.

Используемые сокращения:

HOST – интерфейс доступа внешнего ведущего устройства;

POR – сброс по включению питания;

АЛУ – арифметико-логическое устройство;

ПДП – контроллер прямого доступа в память.

Рисунок 1 – Структурная блок-схема микросхемы

2 Условное графическое обозначение

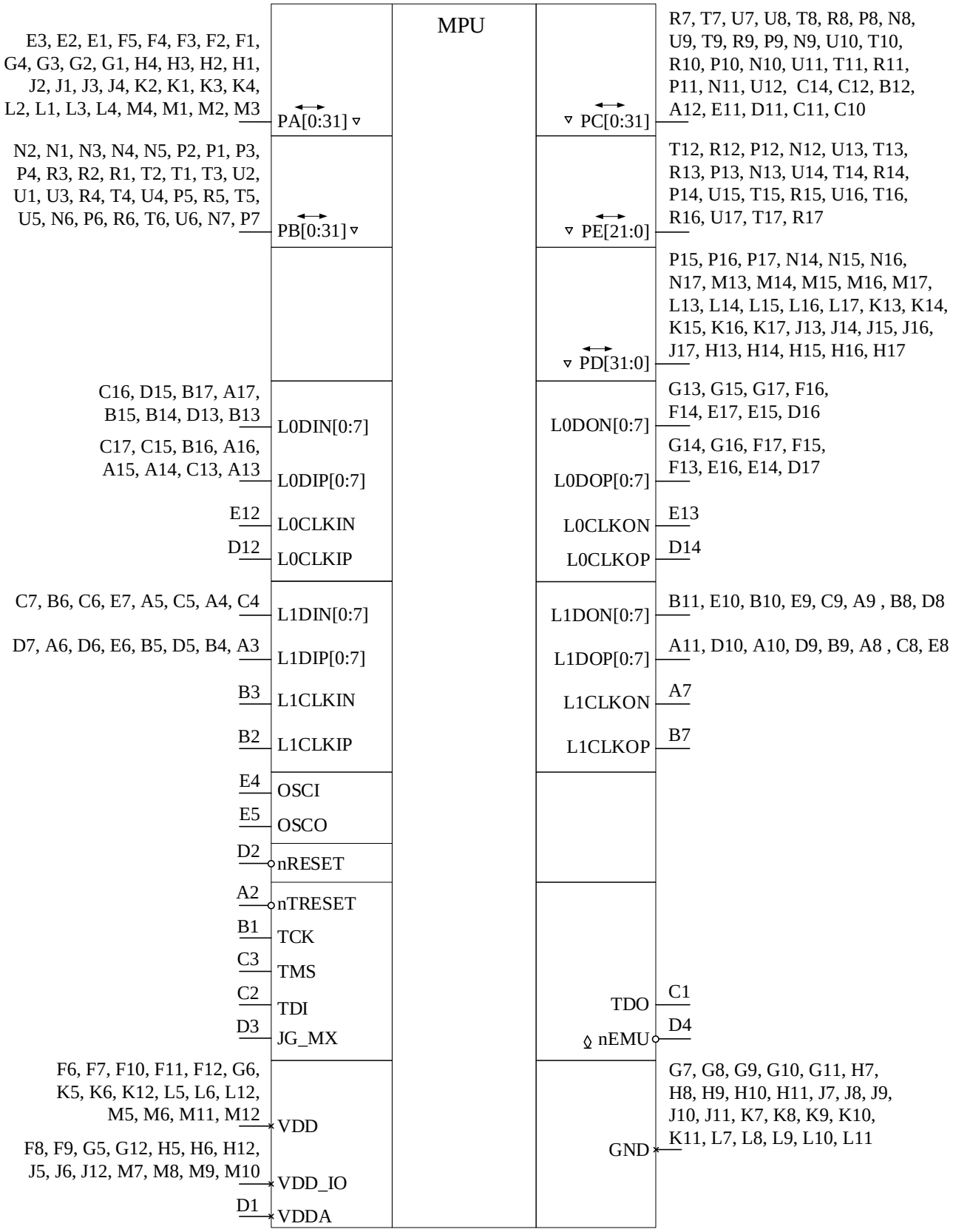


Рисунок 2 – Условное графическое обозначение

3 Описание выводов

Таблица 1 – Описание выводов

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
A2	nTRESET	-	I (pu)	Интерфейс JTAG. Сброс
A3	L1DIP[7]	-	I	LINK-порт 1. Вход данных (прямой). Бит 7
A4	L1DIN[6]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 6
A5	L1DIN[4]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 4
A6	L1DIP[1]	-	I	LINK-порт 1. Вход данных (прямой). Бит 1
A7	L1CLKON	-	O	LINK-порт 1. Выход синхросигнала (инверсный)
A8	L1DOP[5]	-	O	LINK-порт 1. Выход данных (прямой). Бит 5
A9	L1DON[5]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 5
A10	L1DOP[2]	-	O	LINK-порт 1. Выход данных (прямой). Бит 2
A11	L1DOP[0]	-	O	LINK-порт 1. Выход данных (прямой). Бит 0
A12	PC[27]	L0BCMPI	IO (ppd)	Порт С общего назначения. LINK-порт 0. Вход разрешения передачи
		GPS0_SIGN		Интерфейс GPS0. Знак
A13	L0DIP[7]	-	I	LINK-порт 0. Вход данных (прямой). Бит 7
A14	L0DIP[5]	-	I	LINK-порт 0. Вход данных (прямой). Бит 5
A15	L0DIP[4]	-	I	LINK-порт 0. Вход данных (прямой). Бит 4
A16	L0DIP[3]	-	I	LINK-порт 0. Вход данных (прямой). Бит 3
A17	L0DIN[3]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 3
B1	TCK	-	I (pu)	Интерфейс JTAG. Синхросигнал
B2	L1CLKIP	-	I	LINK-порт 1. Вход синхросигнала (прямой)
B3	L1CLKIN	-	I	LINK-порт 1. Вход синхросигнала (инверсный)
B4	L1DIP[6]	-	I	LINK-порт 1. Вход данных (прямой). Бит 6
B5	L1DIP[4]	-	I	LINK-порт 1. Вход данных (прямой). Бит 4
B6	L1DIN[1]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 1
B7	L1CLKOP	-	O	LINK-порт 1. Выход синхросигнала (прямой)
B8	L1DON[6]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 6
B9	L1DOP[4]	-	O	LINK-порт 1. Выход данных (прямой). Бит 4
B10	L1DON[2]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 2
B11	L1DON[0]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 0
B12	PC[26]	L0BCMPO	IO (ppu)	Порт С общего назначения. LINK-порт 0. Выход окончания блока
B13	L0DIN[7]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 7
B14	L0DIN[5]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 5
B15	L0DIN[4]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 4
B16	L0DIP[2]	-	I	LINK-порт 0. Вход данных (прямой). Бит 2
B17	L0DIN[2]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 2
C1	TDO	-	O	Интерфейс JTAG. Выход данных
C2	TDI	-	I (pu)	Интерфейс JTAG. Вход данных
C3	TMS	-	I (pu)	Интерфейс JTAG. Выбор режима

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
C4	L1DIN[7]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 7
C5	L1DIN[5]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 5
C6	L1DIN[2]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 2
C7	L1DIN[0]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 0
C8	L1DOP[6]	-	O	LINK-порт 1. Выход данных (прямой). Бит 6
C9	L1DON[4]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 4
C10	PC[31]	L1BCMPI	IO (ppd)	Порт С общего назначения LINK-порт 1. Вход разрешения передачи
		GPS1_SIGN		Интерфейс GPS1. Знак
C11	PC[30]	L1BCMPO	IO (ppu)	Порт С общего назначения. LINK-порт1. Выход окончания блока
C12	PC[25]	L0ACKI	IO (ppd)	Порт С общего назначения. LINK-порт 0. Вход разрешения передачи
		GPS0_MAG		Интерфейс GPS0. Амплитуда
C13	L0DIP[6]	-	I	LINK-порт 0. Вход данных (прямой). Бит 6
C14	PC[24]	L0ACKO	IO (ppu)	Порт С общего назначения. LINK-порт 0. Выход разрешения передачи
		GPS0_CLKO		Интерфейс GPS0. Выходной синхросигнал
C15	L0DIP[1]	-	I	LINK-порт 0. Вход данных (прямой). Бит 1
C16	L0DIN[0]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 0
C17	L0DIP[0]	-	I	LINK-порт 0. Вход данных (прямой). Бит 0
D1	VDDA	-	PWR	Питание аналоговых блоков
D2	nRESET	-	I	Общий сброс (активный низкий уровень)
D3	JG_MX	-	I (pd)	Выбор блока JTAG
D4	nEMU	-	O	Индикатор режима отладки
D5	L1DIP[5]	-	I	LINK-порт 1. Вход данных (прямой). Бит 5
D6	L1DIP[2]	-	I	LINK-порт 1. Вход данных (прямой). Бит 2
D7	L1DIP[0]	-	I	LINK-порт 1. Вход данных (прямой). Бит 0
D8	L1DON[7]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 7
D9	L1DOP[3]	-	O	LINK-порт 1. Выход данных (прямой). Бит 3
D10	L1DOP[1]	-	O	LINK-порт 1. Выход данных (прямой). Бит 1
D11	PC[29]	L1ACKI	IO (ppd)	Порт С общего назначения. LINK-порт 1. Вход разрешения передачи
		GPS1_MAG		Интерфейс GPS1. Амплитуда
		CAN1_RX		Интерфейс CAN1. Приемник
D12	L0CLKIP	-	I	LINK-порт 0. Вход синхросигнала (прямой)
D13	L0DIN[6]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 6
D14	L0CLKOP	-	O	LINK-порт 0. Выход синхросигнала (прямой)
D15	L0DIN[1]	-	I	LINK-порт 0. Вход данных (инверсный). Бит 1
D16	L0DON[7]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 7
D17	L0DOP[7]	-	O	LINK-порт 0. Выход данных (прямой). Бит 7
E1	PA[2]	U1_TXD	IO (ppu)	Порт А общего назначения. Интерфейс UART1. Выход передатчика
		NF_RE		Интерфейс Nand Flash. Строб чтения

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
E2	PA[1]	U0_RXD	IO (ppu)	Порт А общего назначения. Интерфейс UART0. Вход приемника
E3	PA[0]	U0_TXD	IO (ppu)	Порт А общего назначения. Интерфейс UART0. Выход передатчика
E4	OSCI	-	I	Вход тактовой частоты, вывод подключения резонатора
E5	OSCO	ADDR	O	Выход тактовой частоты, вывод подключения резонатора. Шина адреса внешнего интерфейса
E6	L1DIP[3]	-	I	LINK-порт 1. Вход данных (прямой). Бит 3
E7	L1DIN[3]	-	I	LINK-порт 1. Вход данных (инверсный). Бит 3
E8	L1DOP[7]	-	O	LINK-порт 1. Выход данных (прямой). Бит 7
E9	L1DON[3]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 3
E10	L1DON[1]	-	O	LINK-порт 1. Выход данных (инверсный). Бит 1
E11	PC[28]	L1ACKO	IO (ppu)	Порт С общего назначения. LINK-порт 1. Выход разрешения передачи
		GPS1_CLKO		Интерфейс GPS1. Выходной синхросигнал
		CAN1_TX		Интерфейс CAN1. Передатчик
E12	L0CLKIN	-	I	LINK-порт 0. Вход синхросигнала (инверсный)
E13	L0CLKON	-	O	LINK-порт 0. Выход синхросигнала (инверсный)
E14	L0DOP[6]	-	O	LINK-порт 0. Выход данных (прямой). Бит 6
E15	L0DON[6]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 6
E16	L0DOP[5]	-	O	LINK-порт 0. Выход данных (прямой). Бит 5
E17	L0DON[5]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 5
F1	PA[7]	SPI0_CS[0]	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Выбор SPI устройства 0
F2	PA[6]	SPI0_DI	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Вход данных
		NF_D[1]		Интерфейс Nand Flash Бит данных 1
F3	PA[5]	SPI0_DO	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Выход данных
		NF_D[0]		Интерфейс Nand Flash Бит данных 0
F4	PA[4]	SPI0_CLK	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Синхросигнал
		NF_ALE		Интерфейс Nand Flash. Строб адреса
F5	PA[3]	U1_RXD	IO (ppu)	Порт А общего назначения. Интерфейс UART1. Вход приемника
		NF_WE		Интерфейс Nand Flash. Строб записи
F6	VDD	-	PWR	Питание ядра
F7	VDD	-	PWR	Питание ядра
F8	VDD_IO	-	PWR	Питание контактных площадок
F9	VDD_IO	-	PWR	Питание контактных площадок

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
F10 - F12	VDD	-	PWR	Питание ядра
F13	L0DOP[4]	-	O	LINK-порт 0. Выход данных (прямой). Бит 4
F14	L0DON[4]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 4
F15	L0DOP[3]	-	O	LINK-порт 0. Выход данных (прямой). Бит 3
F16	L0DON[3]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 3
F17	L0DOP[2]	-	O	LINK-порт 0. Выход данных (прямой). Бит 2
G1	PA[11]	SPI0_CS[4]	IO (ppu)	Порт А общего назначения. Интерфейс SPI. Выбор SPI устройства 4
		NF_D[4]		Интерфейс NAND Flash. Бит данных 4
G2	PA[10]	SPI0_CS[3]	IO (ppu)	Порт А общего назначения. Интерфейс SPI. Выбор SPI устройства 3
		NF_D[3]		Интерфейс NAND Flash. Бит данных 3
G3	PA[9]	SPI0_CS[2]	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Выбор SPI устройства 2
		NF_D[2]		Интерфейс NAND Flash. Бит данных 2
G4	PA[8]	SPI0_CS[1]	IO (ppu)	Порт А общего назначения. Интерфейс SPI0. Выбор SPI устройства 1
		NF_CS[1]		Интерфейс Nand Flash. Выборка модуля 1
G5	VDD_IO	-	PWR	Питание площадок ввода/вывода
G6	VDD	-	PWR	Питание ядра
G7	GND	-	GND	Общий
G8	GND	-	GND	Общий
G9	GND	-	GND	Общий
G10	GND	-	GND	Общий
G11	GND	-	GND	Общий
G12	VDD_IO	-	PWR	Питание контактных площадок
G13	L0DON[0]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 0
G14	L0DOP[0]	-	O	LINK-порт 0. Выход данных (прямой). Бит 0
G15	L0DON[1]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 1
G16	L0DOP[1]	-	O	LINK-порт 0. Выход данных (прямой). Бит 1
G17	L0DON[2]	-	O	LINK-порт 0. Выход данных (инверсный). Бит 2
H1	PA[15]	SSI0_TXD	IO (ppu)	Порт А общего назначения. Интерфейс SSI0. Данные передатчика
		AC97_0_SDO		Интерфейс AC97_0. Выходные данные
		SPI1_DI		Интерфейс SPI1. Вход данных
		NF_D[7]		Интерфейс NAND Flash. Бит данных 7
H2	PA[14]	SSI0_TFS	IO (ppu)	Порт А общего назначения. Интерфейс SSI0. Начало кадра или выбор канала левый-правый
		AC97_0_RST		Интерфейс AC97_0. Сброс
		SPI1_DO		Интерфейс SPI1. Выход данных
		NF_D[6]		Интерфейс NAND Flash. Бит данных 6

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
H3	PA[13]	SSI0_TCLK	IO (ppu)	Порт A общего назначения. Интерфейс SSI0. Синхросигнал передатчика
		AC97_0_CLK		Интерфейс AC97_0. Синхросигнал
		SPI1_CLK		Интерфейс SPI1. Синхросигнал
		NF_CLE		Интерфейс NAND Flash. Строб команды
H4	PA[12]	SPI0_CS[5]	IO (ppu)	Порт A общего назначения. Интерфейс SPI. Выбор SPI устройства 5
		NF_D[5]		Интерфейс NAND Flash. Бит данных 5
H5	VDD_IO	-	PWR	Питание площадок ввода/вывода
H6	VDD_IO	-	PWR	Питание площадок ввода/вывода
H7	GND	-	GND	Общий
H8	GND	-	GND	Общий
H9	GND	-	GND	Общий
H10	GND	-	GND	Общий
H11	GND	-	GND	Общий
H12	VDD_IO	-	PWR	Питание контактных площадок
H13	PD[4]	DATA[4]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 4
H14	PD[3]	DATA[3]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 3
H15	PD[2]	DATA[2]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 2
H16	PD[1]	DATA[1]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 1
H17	PD[0]	DATA[0]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 0
J1	PA[17]	SSI0_RFS	IO (ppu)	Порт A общего назначения. Интерфейс SSI0. Начало кадра или выбор канала левый-правый
		AC97_0_SYNC		Интерфейс AC97_0. Синхронизация фрейма
		NF_CS[0]		Интерфейс NAND Flash. Выборка модуля 0
J2	PA[16]	SSI0_RCLK	IO (ppu)	Порт A общего назначения. Интерфейс SSI0. Синхросигнал приемника
		SPI1_CS		Интерфейс SPI1. Выбор SPI устройства
J3	PA[18]	SSI0_RXD	IO (ppu)	Порт A общего назначения. Интерфейс SSI0. Данные приемника
		AC97_0_SDI		Интерфейс AC97_0. Входные данные
		NF_RDY		Интерфейс NAND Flash. Вход готовности
J4	PA[19]	SSI1_TCLK	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Синхросигнал передатчика
		AC97_1_CLK		Интерфейс AC97_1. Синхросигнал
		RXC/REF_CLK		Интерфейс RGMII. Вход синхросигнала / Интерфейс RMII. Вход. Опорный синхросигнал 50 МГц

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
J5	VDD_IO	-	PWR	Питание площадок ввода/вывода
J6	VDD_IO	-	PWR	Питание площадок ввода/вывода
J7	GND	-	GND	Общий
J8	GND	-	GND	Общий
J9	GND	-	GND	Общий
J10	GND	-	GND	Общий
J11	GND	-	GND	Общий
J12	VDD_IO	-	PWR	Питание площадок ввода/вывода
J13	PD[9]	DATA[9]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 9
J14	PD[8]	DATA[8]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 8
J15	PD[7]	DATA[7]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 7
J16	PD[6]	DATA[6]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 6
J17	PD[5]	DATA[5]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 5
K1	PA[21]	SSI1_TXD	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Данные передатчика
		AC97_1_SDO		Интерфейс AC97_1. Выходные данные
		RX[0]		Интерфейсы RGMII, RMII. Вход данных, бит 0
K2	PA[20]	SSI1_TFS	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Начало кадра или выбор канала левый-правый
		AC97_1_RST		Интерфейс AC97_1. Сброс
		RXCTL		Интерфейс RGMII. Вход служебных сигналов
K3	PA[22]	SSI1_RCLK	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Синхросигнал приемника
		GTMR0 CH0o		Таймер 0. Выход 0 ШИМ (+)
		RX[1]		Интерфейсы RGMII, RMII. Вход данных, бит 1
K4	PA[23]	SSI1_RFS	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Начало кадра или выбор канала левый-правый
		AC97_1_SYNC		Интерфейс AC97_1. Синхронизация фрейма
		GTMR0 nCH0o		Таймер 0. Выход 0 ШИМ (-)
		RX[2]/RX_ER		Интерфейс RGMII. Вход данных, бит 2 / Интерфейс RMII. Ошибка приема
K5	VDD	-	PWR	Питание ядра
K6	VDD	-	PWR	Питание ядра
K7	GND	-	GND	Общий
K8	GND	-	GND	Общий
K9	GND	-	GND	Общий

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
K10, K11	GND	-	GND	Общий
K12	VDD	-	PWR	Питание ядра
K13	PD[14]	DATA[14]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 14
K14	PD[13]	DATA[13]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 13
K15	PD[12]	DATA[12]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 12
K16	PD[11]	DATA[11]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 11
K17	PD[10]	DATA[10]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 10
L1	PA[25]	VC_CLK	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Синхросигнал
		GTMR0 nCH1o		Таймер 0. Выход 1 ШИМ (-)
		TXC		Интерфейс RGMII. Выход синхросигнала
L2	PA[24]	SSI1_RXD	IO (ppu)	Порт A общего назначения. Интерфейс SSI1. Данные приемника
		AC97_1_SDI		Интерфейс AC97_1. Входные данные
		GTMR0 CH1o		Таймер 0. Выход 1 ШИМ (+)
		RX[3]/CRS_DV		Интерфейс RGMII. Вход данных, бит 3 / Интерфейс RMII. Служебная информация. Carrier Sense/Receive data valid
L3	PA[26]	VC_VSYNC	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Вход вертикальной развертки
		MIL1_OU1P		Интерфейс МКПД0. Выход основного канала (прямой)
		GTMR0 CH2o		Таймер 0. Выход 2 ШИМ (+)
		TXCTL		Интерфейс RGMII. Выход служебных данных
L4	PA[27]	VC_HSYNC	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Вход горизонтальной развертки видеоинтерфейса
		MIL0_OU1N		Интерфейс МКПД0. Выход основного канала (инверсный)
		GTMR0 nCH2o		Таймер 0. Выход 2 ШИМ (-)
		TX[0]		Интерфейсы RGMII, RMII. Выход данных, бит 0
L5	VDD	-	PWR	Питание ядра
L6	VDD	-	PWR	Питание ядра
L7	GND	-	GND	Общий
L8	GND	-	GND	Общий
L9	GND	-	GND	Общий

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
L10, L11	GND	-	GND	Общий
L12	VDD	-	PWR	Питание ядра
L13	PD[19]	DATA[19]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 19
		NF_D[3]		Интерфейс NAND Flash. Бит данных 3
		MIL1_OU1N		Интерфейс МКПД1. Выход основного канала (–)
L14	PD[18]	DATA[18]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 18
		NF_D[2]		Интерфейс NAND Flash. Бит данных 2
		MIL1_OU1P		Интерфейс МКПД1. Выход основного канала (прямой)
L15	PD[17]	DATA[17]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 17
		NF_D[1]		Интерфейс NAND Flash. Бит данных 1
		MIL1_IN1N		Интерфейс МКПД1. Вход основного канала (–)
L16	PD[16]	DATA[16]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 16
		NF_D[0]		Интерфейс NAND Flash. Бит данных 0
		MIL1_IN1P		Интерфейс МКПД1. Вход основного канала (прямой)
L17	PD[15]	DATA[15]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 15
M1	PA[29]	VC_DATA[1]	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Бит 1 данных
		MIL0_OU2P		Интерфейс МКПД0. Выход резервного канала (прямой)
		GTMR0 nCH3o		Таймер 0. Выход 3 ШИМ (-)
		TX[2]/TX_EN		Интерфейс RGMII. Выход данных, бит 2 / Интерфейс RMII. Разрешение передачи
M2	PA[30]	VC_DATA[2]	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Бит 2 данных
		MIL0_OU2N		Интерфейс МКПД0. Выход резервного канала (инверсный)
		GTMR0 BRK		Таймер 0. Блокировка выходов
		TX[3]		Интерфейс RGMII. Выход данных, бит 3
M3	PA[31]	VC_DATA[3]	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Бит 3 данных
		MIL0_OU2X		Интерфейс МКПД0. Разрешение передачи резервного канала
		GTMR0 ETR		Таймер 0. Универсальный регистратор событий

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
M4	PA[28]	VC_DATA[0]	IO (ppu)	Порт A общего назначения. Интерфейс видеокамеры. Бит 0 данных
		MIL0_OU1X		Интерфейс МКПД0. Разрешение передачи основного канала
		GTMR0 CH3o		Таймер 0. Выход 3 ШИМ (+)
		TX[1]		Интерфейсы RGMII, RMII. Выход данных, бит 1
M5	VDD	-	PWR	Питание ядра
M6	VDD	-	PWR	Питание ядра
M7	VDD_IO	-	PWR	Питание площадок ввода/вывода
M8	VDD_IO	-	PWR	Питание контактных площадок
M9, M10	VDD_IO	-	PWR	Питание контактных площадок
M11, M12	VDD	-	PWR	Питание ядра
M13	PD[24]	DATA[24]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 24
		NF_CLE		Интерфейс NAND Flash. Строб записи
		MIL1_OU2N		Интерфейс МКПД1. Выход резервного канала (инверсный)
M14	PD[23]	DATA[23]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 23
		NF_D[7]		Интерфейс NAND Flash. Бит данных 7
		MIL1_OU2P		Интерфейс МКПД1. Выход резервного канала (прямой)
M15	PD[22]	DATA[22]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 22
		NF_D[6]		Интерфейс NAND Flash. Бит данных 6
		MIL1_IN2N		Интерфейс МКПД1. Вход резервного канала (–)
M16	PD[21]	DATA[21]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 21
		NF_D[5]		Интерфейс NAND Flash. Бит данных 5
		MIL1_IN2P		Интерфейс МКПД1. Вход резервного канала (прямой)
M17	PD[20]	DATA[20]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 20
		NF_D[4]		Интерфейс NAND Flash. Бит данных 4
		MIL1_OU1X		Интерфейс МКПД1. Разрешение передачи основного канала
N1	PB[1]	VC_DATA[5]	IO (ppu)	Порт B общего назначения. Интерфейс видеокамеры. Бит 5 данных
		MIL0_IN1N		Интерфейс МКПД0. Вход основного канала (инверсный)
		SPI2_DO		Интерфейс SPI2. Выход данных
		GTMR0 CH1i		Таймер 0. Вход 1 захвата

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
N2	PB[0]	VC_DATA[4]	IO (ppu)	Порт B общего назначения. Интерфейс видеокамеры. Бит 4 данных
		MIL0_IN1P		Интерфейс МКПД0. Вход основного канала (прямой)
		SPI2_CLK		Интерфейс SPI2. Синхросигнал
		GTMR0 CH0i		Таймер 0. Вход 0 захвата
N3	PB[2]	VC_DATA[6]	IO (ppu)	Порт B общего назначения. Интерфейс видеокамеры. Бит 6 данных
		MIL0_IN2P		Интерфейс МКПД0. Вход резервного канала (прямой)
		SPI2_DI		Интерфейс SPI2. Вход данных
		GTMR0 CH2i		Таймер 0. Вход 2 захвата
N4	PB[3]	VC_DATA[7]	IO (ppu)	Порт B общего назначения. Интерфейс видеокамеры. Бит 7 данных
		MIL0_IN2N		Интерфейс МКПД0. Вход резервного канала (инверсный)
		SPI2_CS		Интерфейс SPI2. Выбор SPI устройства
		GTMR0 CH3i		Таймер 0. Вход 3 захвата
N5	PB[4]	LC_B[0]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 0
		AR_IN1P		Интерфейс ARINC. Вход приемника 1 (прямой)
		nDMAR[0]		Запрос канала 0 контроллера ПДП
N6	PB[25]	LC_T[3]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Дополнительный выход 3
		nIRQ[3]		Вход прерывания 3
		nDMAR[7]		Запрос канала 7 контроллера ПДП
N7	PB[30]	LC_CLK	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Синхросигнал
		AR_OU4P		Интерфейс ARINC. Выход передатчика 4 (прямой)
		nDMAR[11]		Запрос канала 11 контроллера ПДП
		U3_RXD		Интерфейс UART3. Вход приемника
N8	PC[7]	SCLK	O	Порт C общего назначения. Синхросигнал внешнего интерфейса
N9	PC[12]	MSSD[3]	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Активация модуля 3
		I2C_SCL		Интерфейс I2C. SCL
		MDC		Интерфейс RMII, RGMII. Синхросигнал
N10	PC[17]	SD_A10	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Бит 10 адреса
N11	PC[22]	nWR	IO (ppu)	Порт C общего назначения. Интерфейс статической памяти. Строб записи

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
N12	PE[18]	ADDR[18]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 18
N13	PE[13]	ADDR[13]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 13
N14	PD[28]	DATA[28]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 28
		NF_CS[0]		Интерфейс NAND Flash. Выборка модуля 0
N15	PD[27]	DATA[27]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 27
		NF_WE		Интерфейс NAND Flash. Подсветка команды
N16	PD[26]	DATA[26]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 26
		NF_RE		Интерфейс NAND Flash. Подсветка адреса
N17	PD[25]	DATA[25]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 25
		NF_ALE		Интерфейс NAND Flash. Строб чтения
		MIL1_OU2X		Интерфейс МКПД1. Разрешение передачи резервного канала
P1	PB[6]	LC_B[2]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 2
		AR_IN2P		Интерфейс ARINC. Вход приемника 2 (прямой)
		nDMAR[2]		Запрос канала 2 контроллера ПДП
P2	PB[5]	LC_B[1]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 1
		AR_IN1N		Интерфейс ARINC. Вход приемника 1 (инверсный)
		nDMAR[1]		Запрос канала 1 контроллера ПДП
P3	PB[7]	LC_B[3]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 3
		AR_IN2N		Интерфейс ARINC. Вход приемника 2 (инверсный)
		nDMAR[3]		Запрос канала 3 контроллера ПДП
P4	PB[8]	LC_B[4]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 4
		AR_IN3P		Интерфейс ARINC. Вход приемника 3 (прямой)
		GTMR1 CH0o		Таймер 1. Выход 0 ШИМ (+)
P5	PB[21]	LC_R[5]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 5
		AR_OU1N		Интерфейс ARINC. Выход передатчика 1 (инверсный)
		GTMR1 CH3i		Таймер 1. Вход 3 захвата
		USB_D[7]		Интерфейс USB. Выход данных, бит 7

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
P6	PB[26]	LC_PWM	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Выход ШИМ для управления яркостью ЖКИ
		AR_OU2P		Интерфейс ARINC. Выход передатчика 2 (прямой)
		nDMAR[2]		Запрос канала 2 контроллера ПДП
P7	PB[31]	nDMAR3	IO (ppu)	Порт B общего назначения. Запрос канала 3 контроллера ПДП
		AR_OU4N		Интерфейс ARINC. Выход передатчика 4 (инверсный)
P8	PC[6]	BOOT[2]	IO (ppu)	Порт C общего назначения. Выбор режима загрузки
		HDO		Хост интерфейс. Выход данных
P9	PC[11]	MSSD[2]	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Активация модуля 2
		I2C_SDA		Интерфейс I2C. SDA
		MDIO		Интерфейс RMII, RGMII. Линия данных
P10	PC[16]	SD_DQM	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Маска
P11	PC[21]	nRD	IO (ppu)	Порт C общего назначения. Интерфейс статической памяти. Строб чтения
P12	PE[19]	ADDR[19]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 19
P13	PE[14]	ADDR[14]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 14
P14	PE[9]	ADDR[9]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 9
P15	PD[31]	DATA[31]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 31
		NF_RDY		Интерфейс NAND Flash. Вход готовности
P16	PD[30]	DATA[30]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 30
		NF_CS[2]		Интерфейс NAND Flash. Выборка модуля 2
P17	PD[29]	DATA[29]	IO (ppu)	Порт D общего назначения. Шина данных внешнего интерфейса. Бит 29
		NF_CS[1]		Интерфейс NAND Flash. Выборка модуля 1
R1	PB[11]	LC_G[1]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 1
		AR_IN4N		Интерфейс ARINC. Вход приемника 4 (инверсный)
		GTMR1 nCH1o		Таймер 1. Выход 1 ШИМ (-)
		USB_DIR		Интерфейс USB. Направление передачи

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
R2	PB[10]	LC_G[0]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 0
		AR_IN4P		Интерфейс ARINC. Вход приемника 4 (прямой)
		GTMR1 CH1o		Таймер 1. Выход 1 ШИМ (+)
		USB_CLK		Интерфейс USB. Синхросигнал
R3	PB[9]	LC_B[5]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит синего цвета 5
		AR_IN3N		Интерфейс ARINC. Вход приемника 3 (инверсный)
		GTMR1 nCH0o		Таймер 1. Выход 0 ШИМ (-)
R4	PB[18]	LC_R[2]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 2
		AR_IN8P		Интерфейс ARINC. Вход приемника 8 (прямой)
		GTMR1 CH0i		Таймер 1. Вход 0 захвата
		USB_D[4]		Интерфейс USB. Выход данных, бит 4
R5	PB[22]	LC_T[0]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Дополнительный выход 0
		nIRQ[0]		Вход прерывания 0
		nDMAR[4]		Запрос канала 4 контроллера ПДП
R6	PB[27]	LC_DRDY	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Сигнал готовности данных для ЖКИ
		AR_OU2N		Интерфейс ARINC. Выход передатчика 2 (инверсный)
		nDMAR[8]		Запрос канала 8 контроллера ПДП
		U2_TXD		Интерфейс UART2. Выход передатчика. Для коммутации этого вывода необходимо разрешить альтернативную функцию вывода PB[27] и работу блока UART2 в регистре CFG8
R7	PC[0]	FLAG[0]	IO (ppu)	Порт C общего назначения. Управление флагом 0
		CAN0_TX		Интерфейс CAN0. Передатчик
R8	PC[5]	BOOT[1]	IO (ppu)	Порт C общего назначения. Выбор режима загрузки
		HDI		Хост интерфейс. Вход данных
R9	PC[10]	MSSD[1]	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Активация модуля 1
R10	PC[15]	SD_nWE	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Признак записи
R11	PC[20]	nBMS	O	Порт C общего назначения. Интерфейс статической памяти. Выбор внешней памяти начальной загрузки

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
R12	PE[20]	ADDR[20]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 20
R13	PE[15]	ADDR[15]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 15
R14	PE[10]	ADDR[10]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 10
R15	PE[6]	ADDR[6]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 6
R16	PE[3]	ADDR[3]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 3
R17	PE[0]	ADDR[0]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 0
T1	PB[13]	LC_G[3]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 3
		AR_IN5N		Интерфейс ARINC. Вход приемника 5 (инверсный)
		GTMR1 nCH2o		Таймер 1. Выход 2 ШИМ (-)
		USB_STP		Интерфейс USB. Остановка передачи
T2	PB[12]	LC_G[2]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 2
		AR_IN5P		Интерфейс ARINC. Вход приемника 5 (прямой)
		GTMR1 CH2o		Таймер 1. Выход 2 ШИМ (+)
		USB_NXT		Интерфейс USB. Готовность приема
T3	PB[14]	LC_G[4]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 4
		AR_IN6P		Интерфейс ARINC. Вход приемника 6 (прямой)
		GTMR1 CH3o		Таймер 1. Выход 3 ШИМ (+)
		USB_D[0]		Интерфейс USB. Выход данных, бит 0
T4	PB[19]	LC_R[3]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 3
		AR_IN8N		Интерфейс ARINC. Вход приемника 8 (инверсный)
		GTMR1 CH1i		Таймер 1. Вход 1 захвата
		USB_D[5]		Интерфейс USB. Выход данных, бит 5
T5	PB[23]	LC_T[1]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Дополнительный выход 1
		nIRQ[1]		Вход прерывания 1
		nDMAR[5]		Запрос канала 5 контроллера ПДП
T6	PB[28]	LC_VSYNC	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Сигнал вертикальной синхронизации ЖКИ
		AR_OU3P		Интерфейс ARINC. Выход передатчика 3 (прямой)
		nDMAR[9]		Запрос канала 9 контроллера ПДП
		U2_RXD		Интерфейс UART2. Вход приемника

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
T7	PC[1]	FLAG[1]	IO (ppu)	Порт C общего назначения. Управление флагом 1
		CAN0_RX		Интерфейс CAN0. Приемник
T8	PC[4]	BOOT[0]	IO (ppu)	Порт C общего назначения. Выбор режима загрузки
		HCLK		Хост интерфейс. Синхросигнал
T9	PC[9]	MSSD[0]	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Активация модуля 0
T10	PC[14]	SD_nRAS	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Выбор строки
T11	PC[19]	nMS[0]	O	Порт C общего назначения. Интерфейс статической памяти. Выбор типа 0
T12	PE[21]	ADDR[21]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 21
T13	PE[16]	ADDR[16]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 16
T14	PE[11]	ADDR[11]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 11
T15	PE[7]	ADDR[7]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 7
T16	PE[4]	ADDR[4]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 4
T17	PE[1]	ADDR[1]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 1
U1	PB[16]	LC_R[0]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 0
		AR_IN7P		Интерфейс ARINC. Вход приемника 7 (прямой)
		nDMAR[1]		Запрос канала 1 контроллера ПДП
		GTMR1 BRK		Таймер 1. Блокировка выходов
		USB_D[2]		Интерфейс USB. Выход данных, бит 2
U2	PB[15]	LC_G[5]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит зеленого цвета 5
		AR_IN6N		Интерфейс ARINC. Вход приемника 6 (инверсный)
		GTMR1 nCH3o		Таймер 1. Выход 3 ШИМ (-)
		USB_D[1]		Интерфейс USB. Выход данных, бит 1
U3	PB[17]	LC_R[1]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 1
		AR_IN7N		Интерфейс ARINC. Вход приемника 7 (инверсный)
		GTMR1 ETR		Таймер 1. Универсальный регистратор событий
		USB_D[3]		Интерфейс USB. Выход данных, бит 3

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение
U4	PB[20]	LC_R[4]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Бит красного цвета 4
		AR_OU1P		Интерфейс ARINC. Выход передатчика 1 (прямой)
		GTMR1 CH2i		Таймер 1. Вход 2 захвата
		USB_D[6]		Интерфейс USB. Выход данных, бит 6
U5	PB[24]	LC_T[2]	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Дополнительный выход 2
		nIRQ[2]		Вход прерывания 2
		nDMAR[6]		Запрос канала 6 контроллера ПДП
U6	PB[29]	LC_HSYNC	IO (ppu)	Порт B общего назначения. Интерфейс ЖКИ. Сигнал горизонтальной синхронизации ЖКИ
		AR_OU3N		Интерфейс ARINC. Выход передатчика 3 (инверсный)
		nDMAR[10]		Запрос канала 10 контроллера ПДП
		U3_TXD		Интерфейс UART3. Выход передатчика Для коммутации этого вывода необходимо разрешить альтернативную функцию вывода PB[29] и работу блока UART3 в регистре CFG8
U7	PC[2]	FLAG[2]	IO (ppu)	Порт C общего назначения. Управление флагом 2
U8	PC[3]	FLAG[3]	IO (ppu)	Порт C общего назначения. Управление флагом 3
U9	PC[8]	SD_CKE	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Разрешение синхронизации
U10	PC[13]	SD_nCAS	IO (ppu)	Порт C общего назначения. Интерфейс SDRAM. Выбор колонки
U11	PC[18]	nMS[1]	O	Порт C общего назначения. Интерфейс статической памяти. Выбор типа 1
U12	PC[23]	ACK	IO (ppu)	Порт C общего назначения. Интерфейс статической памяти. Сигнал готовности
U13	PE[17]	ADDR[17]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 17
U14	PE[12]	ADDR[12]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 12
U15	PE[8]	ADDR[8]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 8
U16	PE[5]	ADDR[5]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 5
U17	PE[2]	ADDR[2]	O	Порт E общего назначения. Шина адреса внешнего интерфейса. Бит 2

Номер вывода	Обозначение вывода	Дополнительное назначение вывода[1]	Тип вывода	Функциональное назначение

Примечания

1 При использовании альтернативной функции вывод порта GPIO переходит под управление одного из интерфейсов процессора, работа которого в данный момент разрешена. Дополнительные назначения выводов указаны в таблице в порядке возрастания приоритета: сверху указано назначение с наименьшим приоритетом. В мультиплексировании участвуют только дополнительные назначения, используемые интерфейсом как выход;

2 Обозначение типов выводов:

I – вход;

O – выход;

IO – вход/выход;

PWR – питание;

GND – общий;

pu – резистор доопределения до питания;

ppu – резистор доопределения до питания, включаемый программно. По сбросу включен;

pd – резистор доопределения до нуля;

ppd – резистор доопределения до нуля, включаемый программно. По сбросу включен.

3 Цветом выделены функции микросхемы с ревизии 3

4 Указания по применению и эксплуатации

Материал шариковых выводов Sn63/Pb37.

Требования к выполнению технологических операций пайки см. ГОСТ Р 56427-2015.

Выполнение класса задач типа чтения данных из одного порта и их трансляцию в другие порты обеспечивается при тактовой частоте процессора $f_C \leq 200$ МГц. Устойчивая работа микросхем при реализации более сложных алгоритмов обеспечивается при $f_C \leq 230$ МГц.

Типовая схема включения микросхем приведена на рисунке 170.

При ремонте аппаратуры и измерении электрических параметров микросхем замену микросхем необходимо проводить только при отключенных источниках питания.

Запрещается подведение каких-либо электрических сигналов (в том числе шин питание, общий) к выходам микросхем, не используемым согласно таблице 1.

Порядок подачи и снятия напряжения питания и входных сигналов на микросхемы:

– подача (включение микросхем) – общий, питание, входные сигналы или одновременно;

– снятие (выключение микросхем) – одновременно или в обратном порядке.

5 Архитектура микросхемы

Процессор состоит их трех архитектурных частей:

– ядро процессора (рисунок 3), где исполняются команды;

– внутренняя память (рисунок 4), где хранятся данные;

– периферийные устройства (рисунок 5), которые осуществляют операции обмена с внешними устройствами.

В процессоре можно выделить следующие элементы:

– два вычислительных модуля: X и Y, каждый из которых содержит умножитель, ALU, CLU, сдвиговое устройство и регистровый файл объемом в 32 слова;

– два блока целочисленных ALU: J и K, каждый из которых содержит 32-битное целочисленное ALU, а также регистровый файл объемом в 32 слова;

– устройство управления (Sequencer), управляющее ходом исполнения программы и содержащее буфер выравнивания команд (instruction alignment buffer - IAB) и буфер целевых адресов перехода (branch target buffer – BTB);

– три 128-битные шины, обеспечивающие возможность высокоскоростного обмена между внутренней памятью и другими компонентами ядра процессора (вычислительными блоками, блоками целочисленных ALU, устройством управления и SOC-интерфейсом);

– 128-битная шина, обеспечивающая возможность высокоскоростного обмена между внутренней памятью и периферийными устройствами внешнего ввода/вывода (DMA, внешним портом и LINK-портами);

– SOC-интерфейс, обеспечивающий связь между внутренними шинами ядра и шиной периферийных устройств;

– периферийные устройства: интерфейс внешнего порта, контроллер SDRAM, конвейерный интерфейс со статической организацией конвейера, двенадцать каналов DMA, два порта LVDS-линков (с двумя каналами DMA каждый) и др.;

– 12 Мбит внутренней памяти, организованной как шесть блоков по 2 Мбит, каждый из которых содержит 64 К 32-битных слов;

– средства поддержки отладки;

– тестовый порт JTAG.

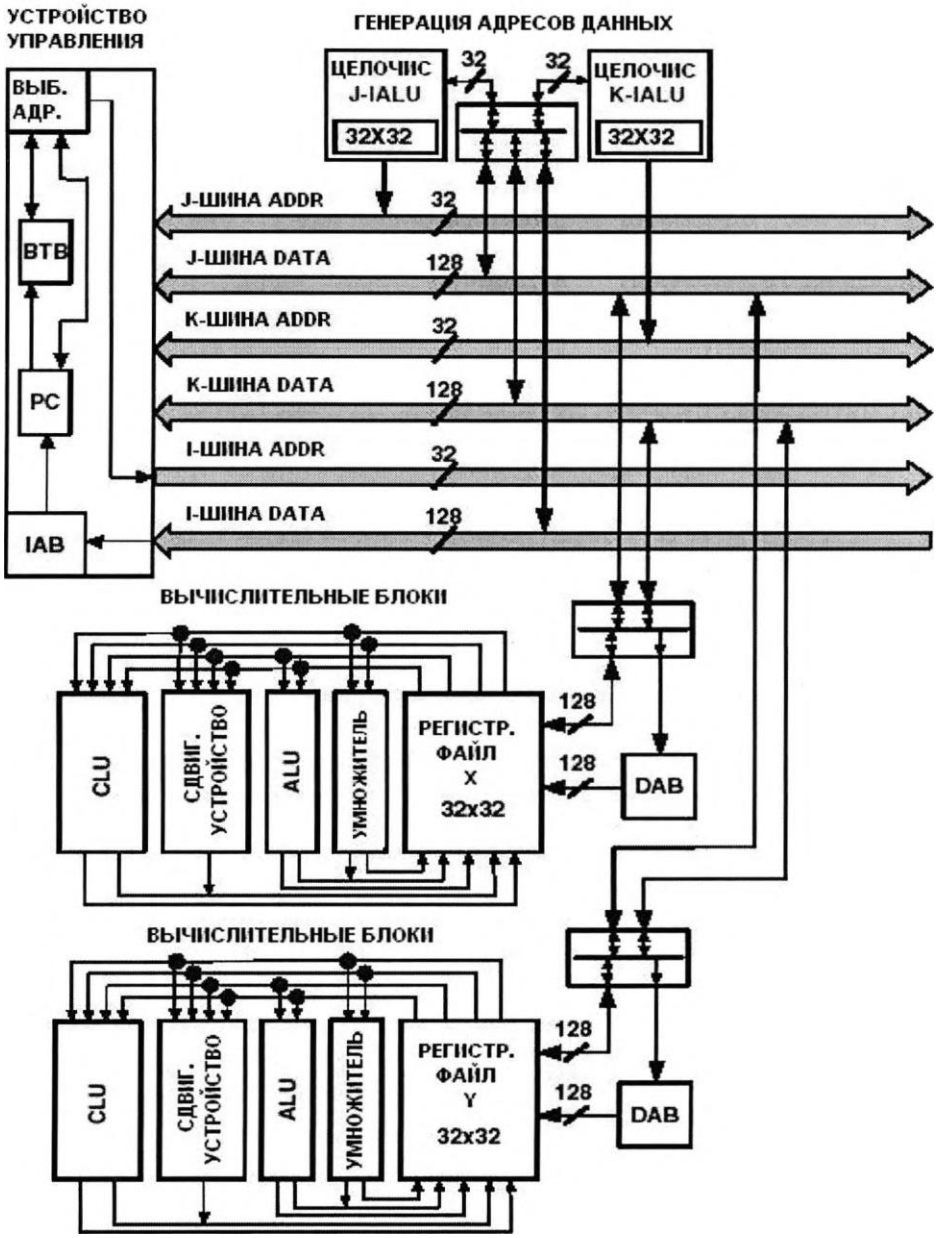


Рисунок 3 – Схема ядра процессора

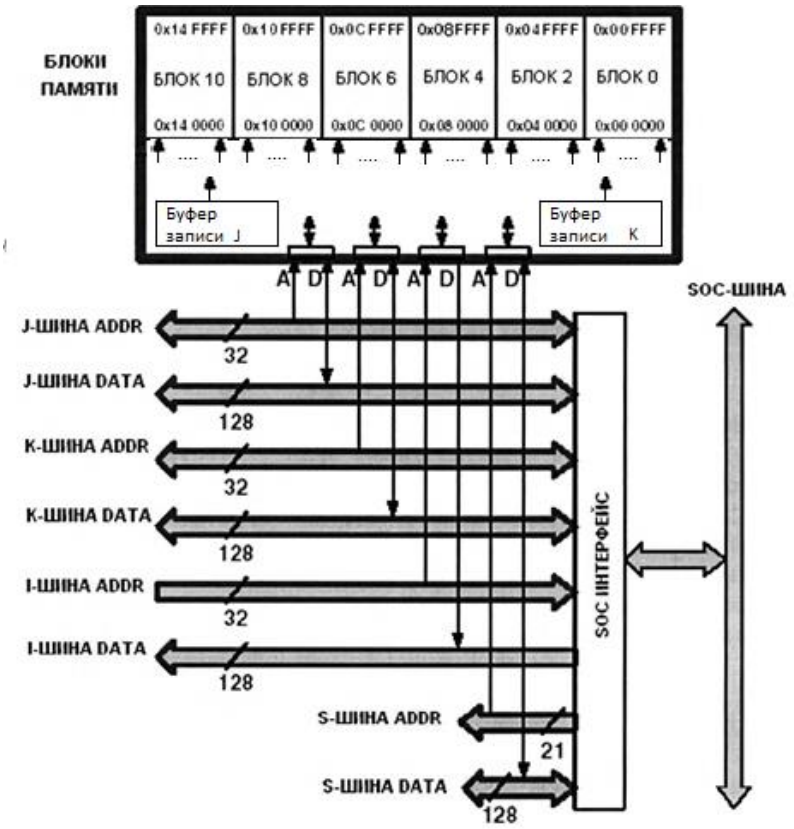
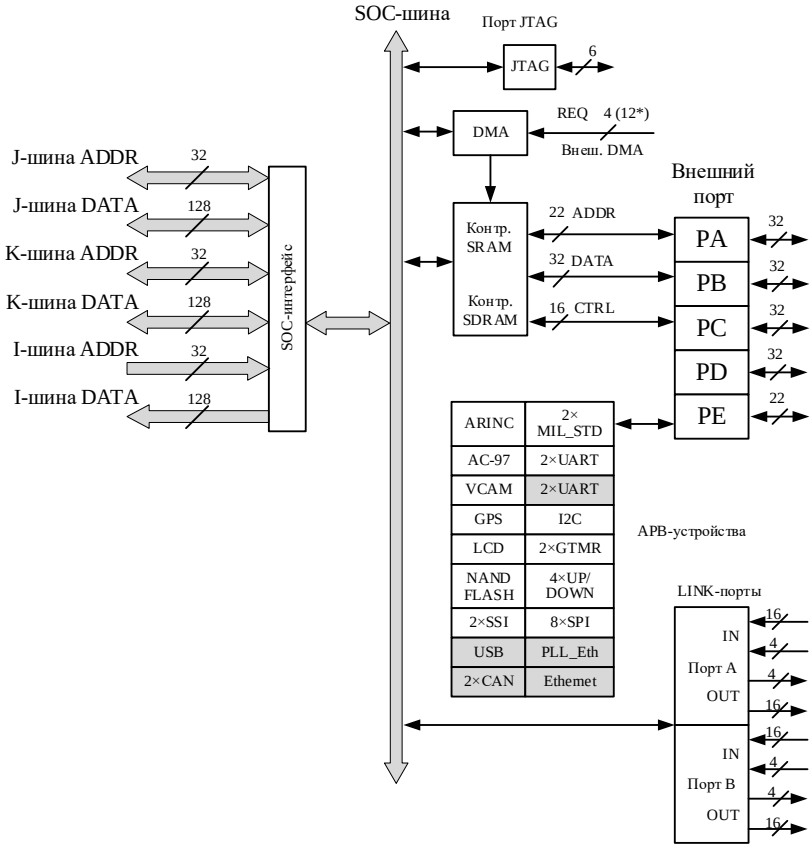


Рисунок 4 – Подключение внутренней памяти к ядру процессора



– Блоки, добавленные в микросхемах с ревизии 3

* Для микросхем с ревизии 3.

Рисунок 5 – Подключение периферийных устройств к ядру процессора

Внутренняя память объемом 12 Мбит разбита на шесть блоков памяти по 64 К слов. Каждая из четырех пар внутренних шин «адрес/данные» подсоединена ко всем шести блокам памяти через коммутационную матрицу. Шесть блоков памяти поддерживают до четырех обращений в каждом такте, причем каждый блок памяти может выполнить 128-битное обращение за такт.

Внешний порт поддерживает ширину шины данных 16 или 32 бита. Высокая пропускная способность ввода/вывода сочетается с высокой скоростью работы ядра. Для достижения высокой тактовой частоты процессор использует конвейерную внешнюю шину для синхронной статической памяти (SSRAM) и для синхронной динамической памяти (SDRAM).

Высокую пропускную способность передачи данных из точки в точку поддерживают два порта LVDS-линков. Каждый LINK-порт обеспечивает полнодуплексную связь.

Наличие разнообразных контроллеров периферийных устройств позволяет подключать:

- внешнюю NAND флэш-память;
- последовательную флэш-память с интерфейсом SPI;
- внешние аудио кодеки с интерфейсом SSI или AC97;
- внешние устройства, поддерживающие интерфейсы ARINC и MIL_STD;
- внешнюю LCD-панель произвольного разрешения;
- внешнюю видеокамеру;
- внешние ЦАП\АЦП.

5.1 Ядро процессора

В ядре процессора можно выделить пять самостоятельных функциональных модуля (рисунок 3):

– Модули X и У – вычислительные модули (устройства), в которых происходит основная обработка данных, и они в свою очередь включают в себя различные устройства обработки данных;

– Модули J и K – целочисленные АЛУ, которые также являются адресными генераторами (или адресными АЛУ), т.к. только они имеют возможность формирования адреса для доступа к памяти. Однако кроме адресных функций эти модули способны выполнять функции и по обработке данных. При этом данные (как и адреса) могут иметь только тип Integer, что соответствует 32-разрядному целому числу;

– Модуль S – устройство управления, которое формирует адреса команд, управляет потоком команд, а также управляет работой всего конвейера ядра.

Модули X и У являются симметричными: все, что делает модуль X, может делать и модуль У. Каждый модуль обрабатывает свой поток команд, при этом модули X и У можно рассматривать, как объединенный модуль, управляемый одной командой.

Модули J и K также симметричны, и каждый из них выполняет свой поток команд.

Устройство управления (модуль S) также имеет способность выполнять несколько потоков команд определенных типов. Среди них основными можно назвать команды переходов и вызовов.

Модульная архитектура процессора с независимыми потоками команд позволяет организовать высокопараллельные вычисления. Так в процессе интенсивных вычислений одно или оба целочисленных ALU вычисляют или генерируют адреса для выборки до двух операндов размером в квадрослово из двух блоков памяти, в то время как устройство управления одновременно извлекает следующую четверку команд из третьего блока памяти. Параллельно вычислительные устройства могут обрабатывать ранее считанные операнды, а устройство управления подготавливать переход.

Пока ядро процессора занято вышеописанными действиями, каналы DMA могут в фоновом режиме обновлять содержимое внутренней памяти квадрословами данных из внешнего порта или из LINK-портов.

Вычислительное ядро процессора достигает исключительно высокой производительности при цифровой обработке сигналов благодаря использованию следующих особенностей:

- вычислительного конвейера;
- пары вычислительных устройств;
- исполнения до четырех команд за такт;
- выборки/записи до восьми слов памяти за такт.

Два идентичных вычислительных устройства (X и У) выполняют арифметические операции как с плавающей, так и с фиксированной точкой. Эти устройства выполняют до шести операций с плавающей точкой или до 24 операций с фиксированной точкой за такт. Модули J и K выполняют SISD-обработку, модули X и У могут выполнять SISD- и SIMD-обработку. При этом модули X и У могут управляться одной общей командой. Модули X и У имеют сложную структуру и включают в себя разнообразные вычислительные блоки.

5.1.1 Вычислительные модули

Ядро процессора содержит два вычислительных устройства, называемых *вычислительными модулями.* Каждый вычислительный модуль содержит регистровый файл и четыре независимых вычислительных блока: ALU, CLU, умножитель и сдвиговое устройство.

Вычислительные блоки способны обрабатывать данные в нескольких форматах представления с фиксированной и плавающей точкой (рисунок 6):

- Форматы данных с фиксированной точкой:
- 64-битное длинное слово (Long Long);
- 32-битное обычное слово (Integer);
- 32-битное комплексное слово (16-битная действительная и 16-битная мнимая части);
- 16-битное короткое слово (Short);
- 8-битное однобайтное слово (Char).

Для коротких слов арифметики с фиксированной точкой учетверенные операции над данными, выровненными на границу квадрослова, обеспечивают быструю обработку вектора данных. Байтовые операции поддерживаются также для данных, выровненных на границу октослова (т.е. 256-битного слова).

– Форматы данных с плавающей точкой:
– 32-битное обычное слово (float);
– 64-битное двойное слово (double);
– 40-битное расширенное слово.

Операции с плавающей точкой выполняются с одинарной, двойной и расширенной точностью. Формат обычного слова с плавающей точкой соответствует стандартному IEEE-формату, а 40-битное число формата с увеличенной точностью размещается в двойном слове (64 бита), занимая дополнительно к 32 битам восемь наименее значащих битов (Least Significant Bits – LSBs) мантиссы для достижения большей точности.

Каждый вычислительный модуль имеет многопортовый регистровый файл, содержащий регистры общего назначения для обмена данными между вычислительными блоками и шинами данных, а также для хранения промежуточных результатов. Ко всем этим регистрам можно обращаться как к обычному, двойному или квадрорегистру. Все операции вычислительных модулей исполняются на двух стадиях конвейера. Результат операции доступен только на второй стадии, поэтому если следующая команда вычислительного модуля использует результат текущей команды как операнд-источник, всегда возникает такт простоя процессора (пузырь).

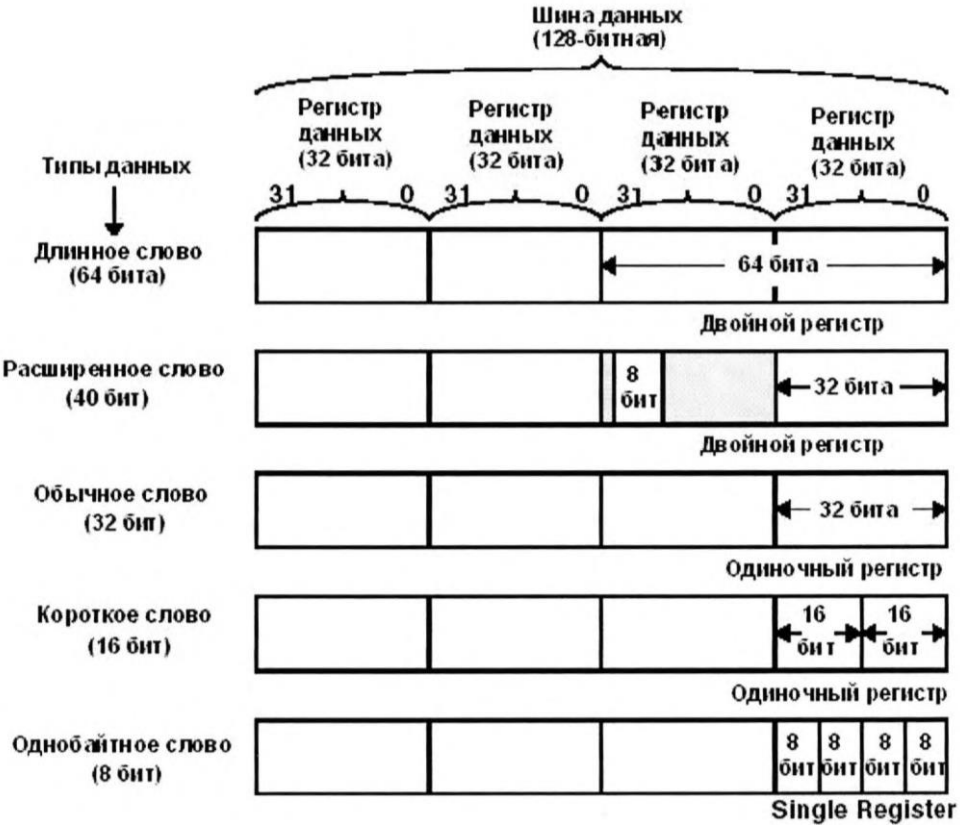


Рисунок 6 – Определения форматов слов

5.1.1.1 *Арифметико-логическое устройство (ALU)*

ALU выполняет арифметические операции над данными с фиксированной и с плавающей точкой, а также логические операции над данными с фиксированной точкой. Источником и приемником большинства операций ALU является регистровый файл вычислительного блока. Операнды ALU могут иметь ширину 32 или 64 бита, что соответствует одиночным операциям над типами Int, Float, Double, Long Long. Для типов меньшей разрядности (Short, Char) поддерживается векторная обработка. Очевидно, что количество элементов в векторах для типа Int равно двум, для типа Short – двум или четырем, а для типа Char – четырем или восьми. Учитывая, что вычислительные модули X и Y могут работать как один сдвоенный модуль, длина обрабатываемых векторов может быть увеличена в два раза, т.е. за такт процессор способен обрабатывать четыре типа Int, восемь типов Short или 16 типов Char. Все это соответствует максимальной ширине шины памяти в 128 бит.

5.1.1.2 *Коммуникационно-логическое устройство (CLU)*

Каждый вычислительный модуль содержит вычислительное устройство специального назначения, называемое *коммуникационно-логическим устройством (CLU)*. Команды CLU предназначены для поддержки различных алгоритмов, используемых в коммуникационных приложениях, а именно алгоритмов:

– Декодирования по Витерби;

– Декодирования турбо кода;

– Сжатия [частотной полосы сигнала] (despreading) для систем множественного доступа с кодовым разделением (CDMA);

– Кросс-корреляций, используемых для отыскания пути.

Особенностью данного блока является собственный 32-словный набор регистров общего назначения. Таким образом, команды CLU могут использовать операнды из РОН вычислительного модуля и из РОН собственного блока.

5.1.1.3 *Умножитель с накоплением (Умножитель)*

Блок умножителя выполняет умножение чисел с фиксированной или плавающей точкой, а также операции умножения с накоплением для чисел с фиксированной точкой. Умножитель поддерживает несколько типов данных для форматов с фиксированной и плавающей точкой. Для плавающей точки – это стандартные форматы одинарной и двойной точности, а также формат с расширенной точностью. Источником и приемником для большинства операций является регистровый файл вычислительного блока. Умножитель также поддерживает операции умножения над комплексными числами, представленными в виде пары 16-разрядных коротких слов упакованными в одно 32-разрядное слово. Младшие значащие биты этого слова представляют собой действительную часть комплексного числа, а старшие значащие биты – мнимую часть. Команды умножения с накоплением используют в качестве приемника специальные регистры-аккумуляторы.

5.1.1.4 *Побитное сдвиговое устройство (сдвиговое устройство)*

Сдвиговое устройство выполняет логические и арифметические сдвиги, манипуляции с битами, заполнение полей и извлечение полей. Сдвиговое устройство выполняет операции с одним 64-битным, одним или двумя 32-битным, двумя или четырьмя 16-битными, а также с четырьмя или восемью восьмибитными операндами с фиксированной точкой. Операции сдвигового устройства включают в себя:

– сдвиги и циклические сдвиги влево и вправо;

– операции манипуляции с битами, включая установку бита, очистку бита, переключение бита и проверку,

– операции манипуляции с битовыми полями, включая извлечение полей, заполнение полей с использованием регистра BFOTMP (внутреннего регистра сдвигового устройства);

– операции с однобитным FIFO для поддержки потоков битов с полями переменной длины;

– поддержка операций преобразования фиксированной и плавающей точки (такими, как извлечение порядка, определение числа ведущих единиц и нулей).

5.1.2 Целочисленное арифметико-логическое устройство (IALU)

Целочисленные ALU могут исполнять стандартные автономные операции ALU с регистровыми файлами IALU, операции загрузки/сохранения регистров и пересылки из регистра в регистр, а также формировать адреса при обмене данными между памятью и регистрами. Процессор имеет пару IALU (J- IALU и K-IALU), что позволяет формировать адреса одновременно для двух параллельно исполняемых транзакций по 128 бит. Наличие целочисленных ALU позволяет вычислительным операциям исполняться с максимальной эффективностью, поскольку вычислительные устройства могут быть заняты исключительно обработкой данных.

Каждое IALU имеет многопортовый 32-словный регистровый файл. Любая вычислительная операция IALU исполняется за один такт. При генерации адресов целочисленные ALU поддерживают предмодификацию регистров без их обновления и постмодификацию с обновлением. Кольцевые буферы реализованы аппаратно. Целочисленные ALU поддерживают следующие типы команд:

– обычные команды IALU;

– команды пересылки данных [из регистра в регистр];

– команды загрузки константы в регистр;

– команды загрузки/сохранения данных с модификацией значением регистра;

– команды загрузки/сохранения данных с модификацией непосредственным значением.

При косвенной адресации (команды с модификацией) значение одного из регистров регистрового файла может быть модифицировано значением другого регистра файла или непосредственным 8- или 32-битным значением, как до (предмодификация), так и после (постмодификация) выполнения обращения к памяти. При адресации кольцевых буферов каждому из четырех первых регистров IALU может

быть сопоставлено значение длины для выполнения автоматической адресации по ее модулю внутри такого буфера. Границами кольцевых буферов могут служить любые адреса памяти. Кольцевые буферы делают возможной эффективную реализацию линий задержки и других структур данных, используемых обычно в цифровых фильтрах и преобразованиях Фурье. Переход адресного указателя через границу кольцевого буфера обрабатывается процессором автоматически, что позволяет сократить накладные расходы и упростить программную реализацию алгоритмов.

Целочисленные ALU поддерживают также бит-реверсивную адресацию, ориентированную на алгоритм БПФ. Бит-реверсивная адресация реализуется с использованием сложения с обратным переносом, подобного обычному сложению, но у которого бит переноса берется из старших битов и направляется в младшие.

IALU обеспечивают гибкость в пересылках данных обычными, двойными или квадрословами. Каждая команда может исполняться с производительностью одна на такт. Обычно задержки, обусловленные зависимостью между командами IALU, отсутствуют, но, если таковые имеются, их длительность может достигать трех тактов. Задержки могут появляться, если текущая команда производит загрузку данных из памяти в регистр IALU или пересылку регистра в регистр IALU, а следующая команда использует загружаемый регистр в качестве операнда-источника. В таком случае возникает три такта задержки до готовности операнда. Необходимо отметить, что IALU обладает практически такими же вычислительными возможностями по обработке данных типа Integer, как и вычислительный модуль. Наиболее проблемным местом является только отсутствие в IALU операции умножения.

5.1.3 Устройство управления потоком команд

Устройство управления (УУ) предоставляет адреса для выборки команд из памяти. УУ совместно с IALU позволяет вычислительным операциям выполняться с максимальной эффективностью. Эффективность ветвления обеспечивается устройством управления за счет использования буфера целевых адресов перехода (BTB), позволяющего сократить задержки на исполнение условных и безусловных переходов. УУ выполняет две задачи:

– декодирование извлеченных из памяти команд (выделение команд из полей командной строки) и отсылка их на соответствующие исполнительные устройства (вычислительные блоки, целочисленные ALU или УУ);

– управление ходом исполнения программы.

Команды устройства управления подразделяются на два типа:

– *команды управления ходом исполнения программы* – используются для смены текущего адреса исполнения (перехода) и для условного исполнения индивидуальных команд;

– *команды непосредственного расширения* – используются для расширения числовых полей, указываемых в непосредственных операндах для УУ и IALU.

Команды управления ходом исполнения программы подразделяются, в свою очередь, на две категории:

– Явные переходы и вызовы, основанные на непосредственно указываемом в коде команды адресном операнде. Например, по команде 'if <cond> jump 100;' всегда осуществляется переход по адресу 100, если только вычисленное значение выражения <cond> есть «истина».

– Косвенные переходы, базирующиеся на адресе, предоставляемом регистром. (Команды, используемые для указания на условное исполнение командной строки, представляют собой подкатегорию косвенных переходов). Например, 'if <cond> cjmp;' является переходом по адресу, содержащемуся в регистре CJMP.

Примечание – Команды управления ходом исполнения должны располагаться в первом поле командной строки. Непосредственными расширениями могут обладать команды IALU и УУ (в последнем случае только команды управления ходом исполнения).

Команды управления ходом исполнения программы не задаются программистом явно, а порождаются автоматически в случае, если размер непосредственных данных, использованных в командах, достаточно велик. Программист должен поместить команду, которая требует непосредственного расширения, в первое поле командной строки и оставить свободное поле в этой строке (задействовав только три поля) с тем, чтобы ассемблер мог поместить непосредственное расширение во второе поле командной строки. В одной командной строке может присутствовать только одно непосредственное расширение.

Процессор достигает своей высокой скорости исполнения за счет использования девяти стадий конвейера. Каждая стадия соответствует одному такту частоты процессора. Упрощенная структура конвейера приведена на рисунке 7.

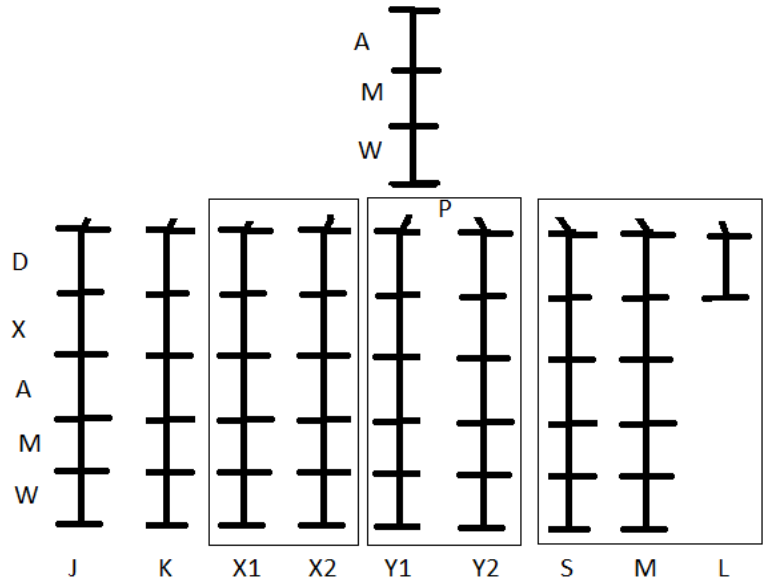


Рисунок 7 – Структура конвейера процессора

На рисунке горизонтальным линиям конвейера соответствуют регистры для хранения информации между стадиями конвейера, а вертикальные линии – задержке обрабатывающей логики. Самая верхняя горизонтальная линия соответствует регистру IP – указателю на текущую команду устройства управления. IP содержит адрес, который отправляется в подсистему памяти для выборки команды.

Стадия адреса (стадия A) соответствует логике декодирования адреса, а также логике арбитража доступа к памяти.

Следующая стадия M соответствует чтению данных из накопителя в буфер памяти.

На стадии W происходит передача команды из буфера памяти в буфер выравнивания команд IAB. Шина команд имеет ширину 128 бит, что соответствует выборке четырех команд за такт.

На стадии P одна линия конвейера *чтения* команд разветвляется на девять линий (ветвей) конвейера *исполнения* команд.

Из рисунка видно, что модули J и K имеют по одной линии конвейера, модули X и Y – по две линии конвейера каждый, а модуль управления включает три ветви S, M, L. Ветвь L короткая, т.к. она обрабатывает команды-расширители константного поля, и вся обработка этой команды сводится к определению, какой линии конвейера соответствует расширение и передачи данных на эту линию. Для модулей обработки J, K, X, Y последняя горизонтальная линия конвейера соответствует РОН, в который записывается результат операции. Название стадий конвейера обработки данных D, X, A, M, W выбрано исходя из логики выполнения команды загрузки из памяти модулей J и K.

- D – декодирование команды;
- X – исполнение команды (вычисление адреса доступа);
- A – передача адреса в подсистему памяти;
- M – чтение данных из накопителя в буфер;
- W – передача данных из буфера памяти в регистр ядра.

В зависимости от типа команды операции, выполняемые на данных стадиях конвейера, будут различны. Так для модулей X и Y наиболее важными являются стадии A, M, W. На стадии A происходит декодирование команды, на стадии M - первый такт исполнения и на стадии W – второй (заключительный такт исполнения).

Одной из задач УУ является отслеживание зависимостей по операндам во всех линиях конвейера и приостановка конвейера в случае, если операнды не готовы. Все линии конвейера продвигаются синхронно. При этом максимум 6 линий из 9-ти могут исполнять одновременно команды.

Для УУ основной линией конвейера является линия S, на которой происходит обработка команд переходов. Для такого глубоко конвейеризированного процессора скорость выполнения переходов является одним из ключевых факторов эффективности исполнения программы. Переходы могут быть безусловными и условными. Условные переходы могут быть двух типов: переходы, анализируемые на стадии X конвейера и на стадии W. Последние (на W) соответствуют анализу флагов операций в модулях X и Y. Можно видеть, что в случае выполнения перехода на стадии X процессор потеряет пять тактов, прежде чем следующая команда достигнет стадии X. Для ветвлений на стадии W таких тактов будет уже восемь. Для уменьшения потерь, связанных с переходами, УУ содержит буфер целевых адресов перехода (BTB) и имеет механизм статического прогнозирования переходов (выполняется пользователем). Если переход помечен как прогнозируемый, он выполняется на стадии D линии S конвейера. Информация о

переходе заносится в строку BTB. При первом выполнении данного перехода теряются четыре такта, однако при последующих его выполнениях потери будут равны нулю.

Еще одной задачей УУ является обработка запросов прерываний.

Процессор имеет внешние линии запроса прерываний общего назначения, а также может обрабатывать прерывания от внутренних периферийных устройств. Есть возможность программным способом обеспечить обработку вложенных прерываний. Прерывания имеют низкую латентность и не могут сбросить с конвейера текущие команды, уже начавшие свое исполнение. Прерывание векторизуется непосредственно по заданному пользователем адресу в регистровом файле таблицы векторов прерываний.

Запрос прерывания поступает в УУ и, если прерывания разрешены, стадия P конвейера прекращает поставлять команды на линии обработки команд, а УУ помещает в регистр IP адрес вектора прерывания (сбрасывая содержимое конвейера чтения команд) и запускает чтение нового потока.

5.1.4 Управление ядром процессора

Существует несколько режимов работы процессора, а также средств управления его работой. В данном разделе описываются режимы работы, которые не управляются посредством выполнения команд. Эти средства управления включают управление:

- доменами синхронизации;
- режимом работы;
- режимом загрузки.

5.1.4.1 *Режимы работы*

Процессор может работать в одном из трех режимов:

- пользовательский;
- супервизор;
- эмулятор.

В режиме пользователя и супервизора все команды выполняются стандартным образом (т.е. путем чтения из памяти). Отличие режима пользователя от режима супервизора заключается только в ограничении доступа к отдельным компонентам системы для пользователя. Режим также влияет на прием и обработку исключений. Приоритеты режимов от наименьшего приоритета к наивысшему: пользовательский, супервизор, эмулятор.

Пользовательский режим

Работа в пользовательском режиме используется для алгоритмов и управляющего кода, который не требует манипуляций с системными ресурсами. Многие системные ресурсы недоступны в пользовательском режиме. Если процессор пытается получить доступы к таким ресурсам, возникает исключение. Ядро ОС работает в режиме супервизора, но пользовательский код ограничен пользовательским режимом.

Следующие регистры доступны ядру программы в пользовательском режиме:

- группы регистров от 0x00 до 0x09 – регистры вычислительных модулей;

– группы регистров от 0x0C до 0x0F – регистры целочисленных АЛУ;

– регистры УУ – CJMP, регистры счетчика циклов LC0, LC1 и регистр статичного флага (SFREG).

Все другие регистры не могут быть записаны ядром программы в пользовательском режиме. Попытка записи в защищенный регистр вызовет исключение.

Режим супервизора

Режим супервизора позволяет программе получить доступ ко всем ресурсам процессора. Процессор работает в режиме супервизора при выполнении одного из условий:

– бит NMOD в SQCTL установлен;

– процедура прерывания запущена.

При аппаратном или программном сбросе процессор переходит в состояние ожидания прерывания. При получении прерывания он переходит в состояние исполнения программы в режиме супервизора.

Если бит NMOD в SQCTL сброшен, процессор входит в пользовательский режим после выхода из прерывания, если только прерывание не является вложенным.

Режим эмулятора

Режим эмулятора используется при управлении процессором отладочным инструментом через порт JTAG. Процессор входит в режим эмулятора при генерации специального исключения. Исключение эмулятора генерируется при одном из событий:

– команда EMUTRAP;

– срабатывание точки наблюдения для входа в эмулятор;

– фронт/срез TMS при установленном бите TEME в EMUCTL.

Исключения эмулятора имеют высочайший приоритет среди исключений и прерываний.

Пока процессор находится в режиме эмулятора, единственным источником команд для него является регистр EMUIR. Регистр EMUIR загружается через порт тестового доступа JTAG. При вхождении в режим эмулятора, внешний контроллер JTAG должен быть включен.

После работы в режиме эмулятора, единственным способом выхода из данного режима является выполнение команды возврата из прерывания (RTI).

В режиме эмулятора, доступ к регистрам отладки (группа регистров 0x1B) может быть получен путем выполнения команд пересылки «регистр-регистр» или командой непосредственной загрузки данных. Эти регистры не могут быть загружены из памяти.

BTB, счетчик циклов, мониторы производительности и буфер трассировки неактивны в режиме эмулятора. Кроме команды RTI (np), все управляющие команды (переход, вызов, RDS, условные переходы, и т.д.) и команда IDLE не должны быть использованы в режиме эмулятора. Команда RTI может быть использована только безусловно.

5.1.4.2 *Режимы старта*

После сброса процессор начинает работу с выполнения специальной загрузочной программы, записанной во внутреннее ПЗУ. Данная программа осуществляет анализ внешних конфигурационных выводов и выполняет заданную последовательность действий. Подробное описание программы начального старта будет приведено в разделе 32 «Начальный старт процессора».

6 Память, регистры и шины

Глава содержит описание карты памяти и регистров. Для информации по использованию регистров для конфигурирования процессорной периферии необходимо смотреть соответствующие разделы данной спецификации, описывающие периферийные модули. Для получения информации об использовании регистров для вычислений и памяти для загрузки и хранения необходимо обращаться к руководству по программированию ТСКЯ.431281.002РП.

Процессор имеет шесть внутренних блоков памяти, как показано на карте памяти на рисунке 8. Каждый блок памяти состоит из 2 МБит пространства памяти и сконфигурирован как 64 Кслова шириной по 32 бита. Существует четыре отдельные 128-битные внутренние шины, каждая из которых может получать доступ к блокам памяти. Процессор ввода-вывода (DMA) имеет собственную внутреннюю шину и не «соревнуется» с ядром за использование внутренней шины. Таким образом, за один цикл может осуществляться до четырех 128-битных передач внутри ядра (две передачи данных, одна команда и одна передача интерфейса ввода-вывода).

Процессор имеет 32-битные адресные шины, обеспечивающие адресное пространство до четырех гигаслов. Процессор поддерживает словную адресацию памяти, а также возможен режим байтовой адресации (см. ниже). Данная глава определяет карту памяти для процессора и показывает, где в пространстве памяти определяется каждый элемент. Зоны пространства памяти составлены из следующих областей:

– пространство внешней памяти – область для стандартной адресации внешней памяти, включая SDRAM, банк 0 (MS0), банк 1 (MS1)

– пространство внутренней памяти – область для стандартной внутренней адресации.

Для быстродействия процессора большое значение имеет конфигурация банков внутренней памяти. Шесть банков внутренней памяти обеспечивают возможность одновременного параллельного доступа к ним в течение одного такта ядра. Такой уровень быстродействия можно достичь только, если процессор будет, например, производить чтение команд из банка 0, чтение данных по шине J из банка 1, чтение данных по шине K из банка 2, чтение-запись по шине S внешним мастером банка 3, отложенная запись по шине J в банк 4 и отложенная запись по шине K в банк 5.

Общая карта памяти показана на рисунке 8.

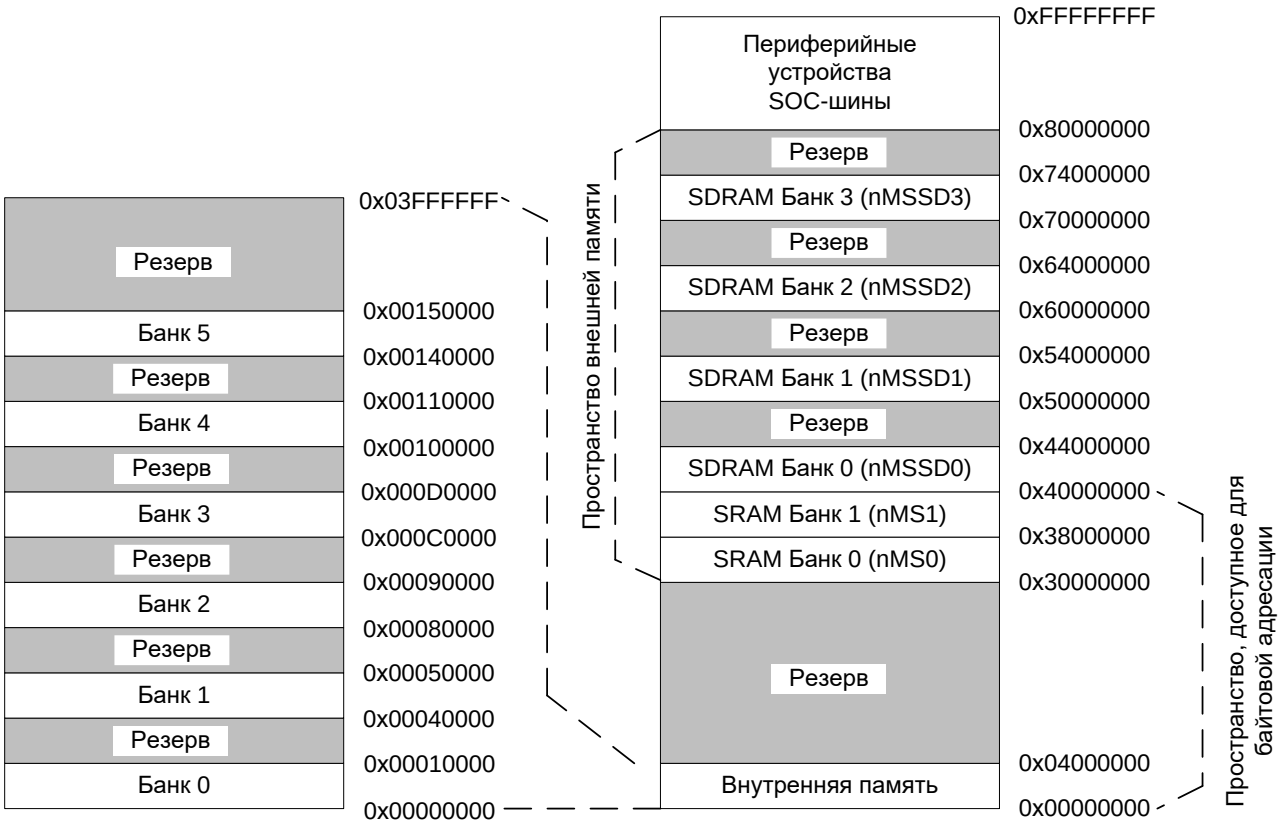


Рисунок 8 – Базовая карта памяти

Процессор поддерживает возможность байтовой адресации не только внутренней, но и внешней памяти. В связи с тем, что перед доступом к памяти адрес байта предварительно сдвигается вправо на два разряда, максимально адресуемое адресное пространство 32-разрядных слов ограничено диапазоном адресов от 0 до 0x3FFF_FFFF. Невозможен доступ к данным с помощью указателя на байт для области динамической памяти, так как для адреса слова (начиная с 0x4000_0000 и выше) для хранения адреса байта недостаточно 32 разрядов.

В связи с этим регистр управления внешним интерфейсом SYSCON дополнен новой функцией бита 27. Если данный бит установить в «1», то доступ к динамической памяти становится возможным с помощью двух адресных областей

Карта памяти процессора с включенной дополнительной областью для доступа к динамической памяти приведена на рисунке 9.

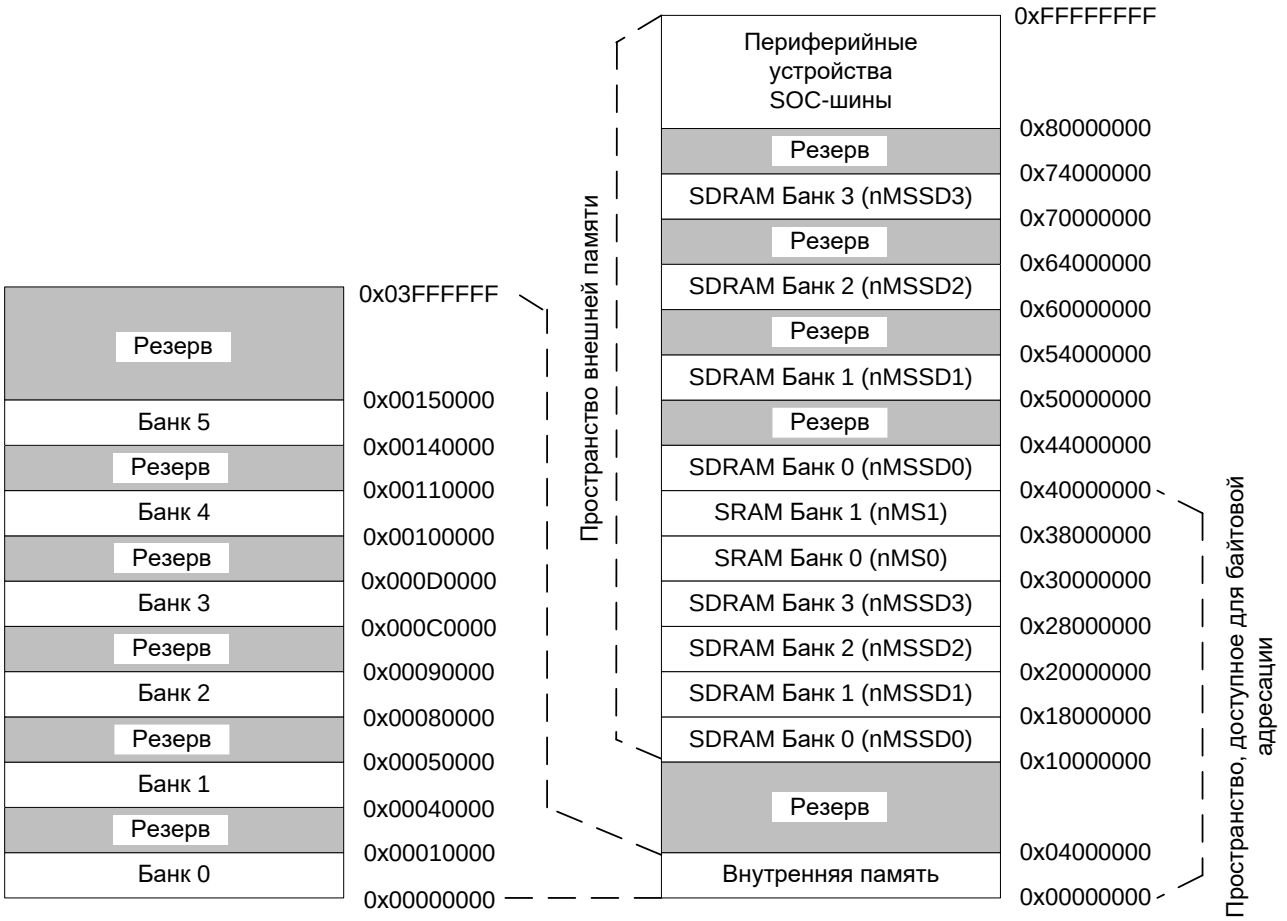


Рисунок 9 – Карта памяти с включенной дополнительной областью

Если для работы прикладной программы необходимо использование внешней динамической памяти, и выполняется обработка данных, требующая их байтовой адресации, в качестве адресного пространства динамической памяти должен быть выбран диапазон адресов (адрес слова) от 0x1000_0000 до 0x2FFF_FFFF. Это соответствует диапазону байтовой адресации от 0x4000_0000 до 0xBFFF_FFFF. Использование дополнительного адресного пространства упрощает преобразование указателей. Установка бита SYSCON[27] не исключает использование базового адресного пространства динамической памяти, т.е. доступ к одному и тому же слову динамической памяти можно сделать как по адресу 0x4000_0000, так и по адресу 0x1000_0000.

Область периферийных устройств имеет только словную адресацию и не поддерживает доступ к отдельным байтам или коротким.

6.1 Пространство периферийных устройств шины SOC

Пространство периферийных устройств шины SOC используется процессором и контроллером DMA для адресации внутренних периферийных устройств, подключенных к шине SOC. Если 31-й бит адреса равен «1», возможен доступ к данному адресному пространству. При этом доступ к регистрам некоторых периферийных устройств может поддерживаться непосредственно по имени (номеру) из команды процессора. Однако это возможно только к регистрам устройств, совместимых с микросхемой К1967ВН028.

Карта памяти периферийных устройств приведена таблице 2.

Таблица 2 – Карта памяти периферийных устройств с режимами доступа

Периферийное устройство	Базовый адрес	CPU	DMA	HOST
Интерфейс Ethernet (буферная память) (для микросхем с ревизии 3)	0x0400_9000	RW	R*W	W
Интерфейс USB (для микросхем с ревизии 3)	0x0480_9000	RW	R*W	W
Контроллер DMA	0x8000_0000 0x8000_0020 0x8000_0040 0x8000_0060	RW	W	W
Хост-интерфейс	0x8000_005c	R	-	-
Интерфейс внешней шины EBIU	0x8000_0080	RW	W	W
Порты связи	0x8000_00a0 0x8000_00e0	RW	W	W
Интерфейс UART0	0x8000_0100	RW	R*W	W
Интерфейс UART1	0x8000_0120	RW	R*W	W
Интерфейс SPI0	0x8000_0140	RW	R*W	W
Интерфейс ЖКИ	0x8000_0180	RW	R*W	W
Интерфейс видеокамеры	0x8000_01a0	RW	R*W	W
Блок управления синхронизацией и энергопотреблением	0x8000_01c0	RW	R*W	W
Часы реального времени RTC	0x8000_01e0	RW	R*W	W
Интерфейс I2S0	0x8000_0200	RW	R*W	W
Интерфейс I2S1	0x8000_0220	RW	R*W	W
Интерфейс NAND Flash	0x8000_0240	RW	R*W	W
Модуль цифровой обработки 0	0x8000_0280	RW	R*W	W
Модуль цифровой обработки 1	0x8000_0260	RW	R*W	W
Модуль цифровой обработки 2	0x8000_02a0	RW	R*W	W
Модуль цифровой обработки 3	0x8000_02c0	RW	R*W	W
Интерфейс I2C	0x8000_02e0	RW	R*W	W
Контроллер прерываний и таймеры общего назначения	0x8000_0300 0x8000_0320 0x8000_0340	RW	W	W
Интерфейс SPI1	0x8000_0360	RW	R*W	W
Интерфейс SPI2	0x8000_0380	RW	R*W	W
Порт JTAG	0x8000_03a0	RW	W	W
Регистры AutoDMA	0x8000_03e0	W	W	-
Таймер 0 с функцией Захвата/ШИМ	0x8000_0400	RW	R*W	W
Таймер 1 с функцией Захвата/ШИМ	0x8000_0440	RW	R*W	W
Интерфейс UART2 (для микросхем с ревизии 3)	0x8000_0500	RW	R*W	W
Интерфейс UART3 (для микросхем с ревизии 3)	0x8000_0520	RW	R*W	W
Интерфейс ARINC (приемники)	0x8000_2000	RW	R*W	W
Интерфейс ARINC (передатчики)	0x8000_3000	RW	R*W	W
Цифровой коррелятор	0x8000_5000	RW	R*W	W
Интерфейс МКПД0	0x8000_6000	RW	R*W	W

Периферийное устройство	Базовый адрес	CPU	DMA	HOST
Интерфейс МКПД1	0x8000_7000	RW	R*W	W
Интерфейс Ethernet (регистры) (для микросхем с ревизии 3)	0x8000_A000	RW	R*W	W
Интерфейс CAN0 (для микросхем с ревизии 3)	0x8000_C000	RW	R*W	W
Интерфейс CAN1 (для микросхем с ревизии 3)	0x8000_E000	RW	R*W	W

Примечания

1 Обозначения в таблице:

RW – чтение и запись;

R – чтение;

W – запись;

R*W – чтение и запись, но операция чтения регистров периферийных устройств доступна только для каналов DMA 0, 1, 2, 3 и 8, 9, 10, 11

6.2 Пространство банка внешней памяти

Пространство банка внешней памяти относится к внешней памяти и устройствам ввода-вывода (SDRAM, SRAM, периферия ввода-вывода и другие стандартные устройства памяти). Внешний интерфейс поддерживает разбиение общего банка внешней памяти на отдельные банки и ставит им в соответствие специальные сигналы выборки и другие управляющие сигналы. Имеется набор банков для SDRAM, доступ к которым производится через контакты выбора внешней памяти MSSD0, MSSD1, MSSD2 и MSSD3. Доступ к этим банкам осуществляется по протоколу SDRAM. Другой набор банков определен контактами выбора внешней памяти MS0 и MS1, где протокол доступа конфигурируется пользователем как конвейерный или протокол медленного устройства, и параметры этих банков определяются регистром SYSCON. Адрес внешней памяти разделяется на поля, как показано в таблице 3.

Таблица 3 – Пространство банка внешней памяти

Биты	Имя	Определение
ADDR25–0	Address	Биты, которые осуществляют выбор данных внутри адресного пространства определенного банка
ADDR30–26	MS/SD	Выбор типа банка внешней памяти: 01100 – банк 0 (MS0); 01110 – банк 1 (MS1); 10000 – SDRAM банк 0 (MSSD0); 10001 – зарезервировано; 10100 – SDRAM банк 1 (MSSD1); 10101 – зарезервировано; 11000 – SDRAM банк 2 (MSSD2); 11001 – зарезервировано; 11100 – SDRAM банк 3 (MSSD3); 11101 – зарезервировано
ADDR31		Значение бита всегда будет равно нулю

6.3 Внутреннее адресное пространство

Внутреннее адресное пространство (см. рисунок 8) соответствует собственному адресному пространству процессора. Оно используется для передач внутри процессора, т.е. для доступа к блокам внутренней памяти.

6.4 Универсальные регистры

К универсальным регистрам относятся все регистры вычислительного ядра. Универсальные регистры не отображаются в карте памяти, т.е. доступ к ним по адресу памяти невозможен. Доступ к регистрам возможен только при выполнении команд. Все регистры разбиты по группам. В одной группе находится 32 регистра. При явном доступе к регистру в коде команды указывается номер группы и номер регистра в группе. Регистры распределены по группам в соответствии с их отношением к различным модулям процессора. Например, регистры общего назначения (РОН) данных (XR31–0, YR31–0, или XYR31–0) вычислительных блоков находятся в одной группе регистров, тогда как РОН данных для целочисленного АЛУ (J30–0 или K30–0) – в других. Термин «универсальный регистр» указывает на следующие особенности.

Во-первых, содержимое регистра может быть загружено из памяти, сохранится в памяти или передаваться в\из других универсальных регистров процессора.

Во-вторых, регистр может быть доступен в одинарный, двойной и счетверенный (квадрорегистр). Ряд регистров может быть доступен только как одинарные.

Загрузка, сохранение и пересылка содержимого универсальных регистров доступна внутри процессора с использованием имени регистра в командах процессора, а не по адресу в памяти. Имена регистров содержатся в таблице групп регистров. Команда загрузки регистра читает данные из памяти процессора и помещает их в регистр. Команда сохранения регистра берет данные из регистра и помещает их в память процессора. Команда пересылки копирует данные из одного регистра в другой. Имеются также команды загрузки константы в регистр, когда значение операнда берется из кода команды и помещается в регистр. Следующий пример показывает доступы загрузки, сохранения и передачи одного регистра.

```
XR0 = 0x76543210 ;; /* загружает в XR0  конкретные 32-бит  данные */
XR4 = [J31 + 0x43] ;; /* загружает в XR4 данные из ячейки памяти с адресом 0x43 */
[J0 + J4] = YR0 ;; /* сохраняет YR0 в память */
XR7 = SQSTAT ;; /* передает содержимое SQSTAT в XR7 */
```

При возможности выполнения процессором до четырех команд, одновременно могут быть выполнены:

- две команды загрузки регистров из памяти
- две команды сохранения в память
- две команды пересылки.
- две загрузки регистров константами.

Все эти типы команд выполняются в целочисленных J и K модулях.

Для одновременного доступа к двум или четырем регистрам, образующим длинные слова и квадрослова, предусмотрена специальная техника образования нового имени регистра. При этом изменение имени регистра указывает на диапазон регистров, а не на новый регистр. Например, имя сдвоенного регистра XR1:0 указывает регистр R1 и R0 в вычислительном блоке X, а счетверенное имя регистра J31:28 указывает регистры J31, J30, J29, и J28 в J-IALU. Обязательно выравнивание границ адресов сдвоенных и счетверенных регистров. Для сдвоенного регистра его младший номер должен быть кратен двум (например, R1:0, R3:2, R27:26), а для счетверенного кратен 4 (например, R3:0, R7:4, R31:28). Следующий пример кода показывает загрузку, сохранение и передачу сдвоенного и счетверенного регистра.

```
YR5:4 = L [J31 +  0xf0] ;; /* загрузка YR5:4 из памяти */
XR3:0 = Q [K4 += K5] ;; /* загрузка XR3:0 из памяти */
L [J0 + J2] = YR31:30 ;; /* сохранение YR31:30 в память */
Q [K31 + K28] = YR7:4 ;; /* сохранение YR7:4 в память */
DCS0 = YR11:8 ;; /* передача YR11:8 в DCS0 (DP, DY, DX, DI) */
```

Обычно цикл загрузки или сохранения регистра в памяти происходит между регистром и словом памяти одинакового размера – 32, 64 и 128 бит. В этом случае операция называется «нормальный доступ чтения\записи».

Для использования преимуществ обработки «одна команда – много данных» (SIMD) архитектура шины процессора поддерживает типы доступа, в которых содержимое одного слова памяти может быть загружено в два регистра (широковещательное чтение), и тип доступа, в котором несколько регистров могут быть загружены\сохранены с различными данными, используя одно слово в памяти (доступ объединенного чтения или записи).

Пространство регистров состоит их 64 регистровых групп с количеством регистров в группе равным 32. Группы регистров определены в диапазоне 0x3F–0 (63–0). Группы 0x1F–0 (31-0) доступны для любых команд передачи данных (загрузка константой, пересылка регистра, загрузка и сохранение в памяти). Эти группы относятся к вычислительному ядру процессора.

Группы 0x3F–0x20 доступны только командам пересылки регистров и соответствуют регистрам некоторых периферийных устройств SOC-шины. Доступ к этим регистрам возможен как с помощью команд пересылки, так и с помощью адреса в пространстве адресов периферийных устройств.

Группы регистров вычислительного ядра:

- 0x00–0x09 вычислительные модули X и У;
- 0x0C–0x0F целочисленные J и K АЛУ;
- 0x0A,0x1B модуль отладки;
- 0x1A устройство управления;
- 0x1E – 0x1F устройство защиты памяти.

Группы регистров периферийных устройств.

- 0x20 – 0x23 контроллер DMA;
- 0x24 контроллер внешнего порта;

– 0x25 – 0x27 порты связи;
– 0x38, 0x39, и 0x3A контроллер прерываний;
– 0x3D модуль JTAG.

Прочие группы зарезервированы и не могут быть доступны для приложений, т.к. они могут вызвать непредсказуемую реакцию процессора.

Существует прямая связь между адресом универсального регистра в карте памяти и группой, в которой он находится. Важно знать номер группы, которой принадлежит регистр, т.к. они могут иметь различные ограничения доступа. Для определения номера группы регистров см. рисунок 10.

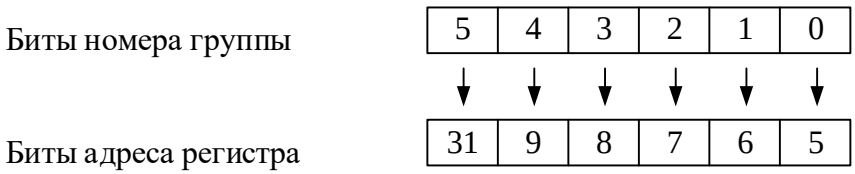


Рисунок 10 – Номер группы регистров в зависимости от адреса регистра

Важно обратить внимание на номер регистра в плане его (регистра) расположения в процессоре: в ядре процессора (внутренние универсальные регистры) или в периферийном модуле SOC-шины, т.к. регистры имеют различные ограничения доступа. Некоторые из регистров не используют все 32 бита. Неиспользованные биты зарезервированы. При записи в регистр с резервными битами резервные биты должны быть записаны нулевым значением. При чтении регистра с резервными битами резервные биты могут иметь любое значение.

Ниже приведен перечень групп регистров вычислительного ядра. Описание групп регистров периферийных модулей приведено в главах, посвященных конкретным периферийным устройствам.

6.5 Группы регистров вычислительных модулей

Файл регистров вычислительного модуля содержит 32 регистра общего назначения (РОН). Существует два подобных файла, по одному в каждом вычислительном модуле (X и Y). Однако, фактически имея две группы по 32 регистра, регистры вычислительных модулей используют 10 номеров групп (с нулевого по девятый) для своей адресации. Можно сказать, что каждый регистр имеет собственное имя и несколько псевдонимов. Использование подобных псевдонимов удобно в командах пересылки регистров, а также в командах чтения/записи при работе с памятью. С помощью псевдонима происходит дополнительное кодирование выполняемой с регистром операции.

Ниже указаны виды доступов к регистрам вычислительного блока в соответствии с их распределением в память (таблицы 4 – 8). Номер регистра образуется конкатенацией номера группы (старшие шесть бит) и номера регистра в группе (младшие пять бит).

Таблица 4 – Группа регистров вычислительного блока X

Имя	Номер	Значение по умолчанию	Имя	Номер	Значение по умолчанию
XR0	0x 0000	Не определено	XR16	0x0010	Не определено
XR1	0x 0001	Не определено	XR17	0x 0011	Не определено
XR2	0x 0002	Не определено	XR18	0x 0012	Не определено
XR3	0x 0003	Не определено	XR19	0x 0013	Не определено
XR4	0x 0004	Не определено	XR20	0x 0014	Не определено
XR5	0x 0005	Не определено	XR21	0x 0015	Не определено
XR6	0x 0006	Не определено	XR22	0x 0016	Не определено
XR7	0x 0007	Не определено	XR23	0x 0017	Не определено
XR8	0x 0008	Не определено	XR24	0x 0018	Не определено
XR9	0x 0009	Не определено	XR25	0x 0019	Не определено
XR10	0x 000A	Не определено	XR26	0x 001A	Не определено
XR11	0x 000B	Не определено	XR27	0x 001B	Не определено
XR12	0x 000C	Не определено	XR28	0x 001C	Не определено
XR13	0x 000D	Не определено	XR29	0x 001D	Не определено
XR14	0x 000E	Не определено	XR30	0x 001E	Не определено
XR15	0x 000F	Не определено	XR31	0x 001F	Не определено

Таблица 5 – Группа регистров вычислительного блока Y

Имя	Номер	Значение по умолчанию	Имя	Номер	Значение по умолчанию
YR0	0x 0040	Не определено	YR16	0x 0050	Не определено
YR1	0x 0041	Не определено	YR17	0x 0051	Не определено
YR2	0x 0042	Не определено	YR18	0x 0052	Не определено
YR3	0x 0043	Не определено	YR19	0x 0053	Не определено
YR4	0x 0044	Не определено	YR20	0x 0054	Не определено
YR5	0x 0045	Не определено	YR21	0x 0055	Не определено
YR6	0x 0046	Не определено	YR22	0x 0056	Не определено
YR7	0x 0047	Не определено	YR23	0x 0057	Не определено
YR8	0x 0048	Не определено	YR24	0x 0058	Не определено
YR9	0x 0049	Не определено	YR25	0x 0059	Не определено
YR10	0x 004A	Не определено	YR26	0x 005A	Не определено
YR11	0x 004B	Не определено	YR27	0x 005B	Не определено
YR12	0x 004C	Не определено	YR28	0x 005C	Не определено
YR13	0x 004D	Не определено	YR29	0x 005D	Не определено
YR14	0x 004E	Не определено	YR30	0x 005E	Не определено
YR15	0x 004F	Не определено	YR31	0x 005F	Не определено

Таблица 6 – Объединенная группа регистров вычислительного модуля XY

Имя	Адрес	Значение по умолчанию	Имя	Адрес	Значение по умолчанию
XYR0	0x 0080	Не определено	XYR16	0x 0090	Не определено
XYR1	0x 0081	Не определено	XYR17	0x 0091	Не определено
XYR2	0x 0082	Не определено	XYR18	0x 0092	Не определено
XYR3	0x 0083	Не определено	XYR19	0x 0093	Не определено
XYR4	0x 0084	Не определено	XYR20	0x 0094	Не определено
XYR5	0x 0085	Не определено	XYR21	0x 0095	Не определено
XYR6	0x 0086	Не определено	XYR22	0x 0096	Не определено
XYR7	0x 0087	Не определено	XYR23	0x 0097	Не определено
XYR8	0x 0088	Не определено	XYR24	0x 0098	Не определено
XYR9	0x 0089	Не определено	XYR25	0x 0099	Не определено
XYR10	0x 008A	Не определено	XYR26	0x 009A	Не определено
XYR11	0x 008B	Не определено	XYR27	0x 009B	Не определено
XYR12	0x 008C	Не определено	XYR28	0x 009C	Не определено
XYR13	0x 008D	Не определено	XYR29	0x 009D	Не определено
XYR14	0x 008E	Не определено	XYR30	0x 009E	Не определено
XYR15	0x 008F	Не определено	XYR31	0x 009F	Не определено

Таблица 7 – Объединенная группа регистров вычислительного модуля YX

Имя	Адрес	Значение по умолчанию	Имя	Адрес	Значение по умолчанию
YXR0	0x 00C0	Не определено	YXR16	0x 00D0	Не определено
YXR1	0x 00C1	Не определено	YXR17	0x 00D1	Не определено
YXR2	0x 00C2	Не определено	YXR18	0x 00D2	Не определено
YXR3	0x 00C3	Не определено	YXR19	0x 00D3	Не определено
YXR4	0x 00C4	Не определено	YXR20	0x 00D4	Не определено
YXR5	0x 00C5	Не определено	YXR21	0x 00D5	Не определено
YXR6	0x 00C6	Не определено	YXR22	0x 00D6	Не определено
YXR7	0x 00C7	Не определено	YXR23	0x 00D7	Не определено
YXR8	0x 00C8	Не определено	YXR24	0x 00D8	Не определено
YXR9	0x 00C9	Не определено	YXR25	0x 00D9	Не определено
YXR10	0x 00CA	Не определено	YXR26	0x 00DA	Не определено
YXR11	0x 00CB	Не определено	YXR27	0x 00DB	Не определено
YXR12	0x 00CC	Не определено	YXR28	0x 00DC	Не определено
YXR13	0x 00CD	Не определено	YXR29	0x 00DD	Не определено
YXR14	0x 00CE	Не определено	YXR30	0x 00DE	Не определено
YXR15	0x 00CF	Не определено	YXR31	0x 00DF	Не определено

Таблица 8 – Группа регистров широковещательного доступа вычислительного модуля XY

Имя	Адрес	Значение по умолчанию	Имя	Адрес	Значение по умолчанию
XYR0	0x 0100	Не определено	XYR16	0x 0110	Не определено
XYR1	0x 0101	Не определено	XYR17	0x 0111	Не определено
XYR2	0x 0102	Не определено	XYR18	0x 0112	Не определено
XYR3	0x 0103	Не определено	XYR19	0x 0113	Не определено
XYR4	0x 0104	Не определено	XYR20	0x 0114	Не определено
XYR5	0x 0105	Не определено	XYR21	0x 0115	Не определено
XYR6	0x 0106	Не определено	XYR22	0x 0116	Не определено
XYR7	0x 0107	Не определено	XYR23	0x 0117	Не определено
XYR8	0x 0108	Не определено	XYR24	0x 0118	Не определено
XYR9	0x 0109	Не определено	XYR25	0x 0119	Не определено
XYR10	0x 010A	Не определено	XYR26	0x 011A	Не определено
XYR11	0x 010B	Не определено	XYR27	0x 011B	Не определено
XYR12	0x 010C	Не определено	XYR28	0x 011C	Не определено
XYR13	0x 010D	Не определено	XYR29	0x 011D	Не определено
XYR14	0x 010E	Не определено	XYR30	0x 011E	Не определено
XYR15	0x 010F	Не определено	XYR31	0x 011F	Не определено

Отметим, что дополнительно имеются номера групп, которые используются командами альтернативного доступа с использованием буфера DAB и кольцевых буферов.

6.6 Регистры вычислительного модуля без номера

Есть несколько регистров в вычислительных модулях, которые не являются универсальными, поэтому не доступны посредством их адресации с помощью номера группы и номера в группе. Эти регистры доступны только при выполнении специальных команд, которые передают данные между ними и РОН вычислительного модуля.

6.6.1 Регистры статуса вычислительных модулей (XSTAT/YSTAT)

Регистры XSTAT и YSTAT – 32-битные регистры вычислительных модулей, которые хранят состояние флагов модулей. Каждый флаг регистра обновляется при выполнении соответствующей команды. Вычислительный модуль состоит из вычислительных блоков: ALU, умножитель, сдвигатель, CLU. Каждый из этих блоков формирует специальные флаги-признаки результата операции. Регистр состояний хранит данные флаги. Флаги имеют разный тип. Большинство флагов регистра изменяет свое значение при выполнении соответствующих команд, однако есть специальные флаги (sticky), которые после их установки в первое значение уже невозможно сбросить. Сброс возможен только командой загрузки регистра. Stiky биты очень удобны, когда нужно в конце алгоритма проверить были ли аномальные ситуации при выполнении вычислений на протяжении всего алгоритма. В отличие от них другие флаги нужно проверять (если

необходимо) сразу после выполнения операции. Подробное описание бит регистров X/YSTAT (значение после сброса = 0x0000 0000) приведено в таблице 9.

Таблица 9 – Регистр состояния вычислительного модуля

Бит	Имя	Назначение
0	AZ	ALU. Признак нулевого результата операции
1	AN	ALU. Признак отрицательного результата операции
2	AV	ALU. Признак переполнения результата операции
3	AC	ALU. Признак переноса
4	MZ	Умножитель. Признак нулевого результата операции
5	MN	Умножитель. Признак отрицательного результата операции
6	MV	Умножитель. Признак переполнения результата операции
7	MU	Умножитель. Признак исчезновения (underflow) результата операции
8	SZ	Сдвигатель. Признак нулевого результата операции
9	SN	Сдвигатель. Признак отрицательного результата операции
10	BF0	Сдвигатель. Флаг блока операций с плавающей запятой
11	BF1	Сдвигатель. Флаг блока операций с плавающей запятой
12	AI	Ошибочная (Invalid) операция АЛУ с плавающей запятой
13	MI	Ошибочная (Invalid) операция умножителя с плавающей запятой
14	TROV	CLU. Переполнение операции trellis
15	TRSOV	CLU. Переполнение операции trellis. Stiky бит
16…19	-	
20	UEN	Разрешение (если 1) генерации исключительной ситуации при возникновении флага исчезновения (underflow) результата
21	OEN	Разрешение (если 1) генерации исключительной ситуации при возникновении флага переполнения(overflow) результата
22	IVEN	Разрешение (если 1) генерации исключительной ситуации при возникновении флага ошибочноuj (invalid) результата при операциях с плавающей запятой
23	-	
Stiky биты		
24	AUS	Underflow в АЛУ с плавающей запятой
25	AVS	Переполнение в АЛУ с плавающей запятой
26	AOS	Переполнение в АЛУ с фиксированной точкой
27	AIS	Ошибочный результат в АЛУ с плавающей запятой
28	MUS	Underflow в умножителе с плавающей запятой
29	MVS	Переполнение в умножителе с плавающей запятой
30	MOS	Переполнение в умножителе с фиксированной точкой
31	MIS	Ошибочный результат в умножителе с плавающей запятой

6.6.2 Регистры АЛУ

Регистры параллельных результатов (PR0 и PR1) – 32-битные регистры, используемые для специальных команд суммирования.

6.6.3 Регистры умножителя

Регистры результатов умножителя (MR3–0 и MR4) используются для накопления при выполнении операций умножения с накоплением, т.е. выполняют функции аккумуляторов.

6.6.4 Регистры сдвигателя

Регистр BFOTMP – 64-битный регистр для команды PUTBITS, используемой для организации потока бит.

6.6.5 Регистры блока CLU

Модуль CLU использует дополнительный набор регистров для выполнения своих команд. Это регистры данных (TR31–0), регистры истории (THR3–0) и регистр управления (CMCTL). Каждый регистр имеют ширину 32 бита.

6.7 Группы регистров целочисленного АЛУ

Каждое целочисленное АЛУ имеет две группы универсальных регистров:

– первая группа – файл регистров общего назначения, включающий 32 слова;

– вторая группа – регистры для указания параметров кольцевой буферизации.

Группы регистров целочисленного АЛУ показаны в таблицах 10 – 13.

Файл регистров кольцевого буфера содержит следующие регистры:

3 ÷ 0 – кольцевого буфера JB3–0 и KB3–0;

7 ÷ 4 – длина кольцевого буфера JL3–0 и KL3–0;

26 ÷ 8 – резерв;

27 – регистр указателя стека режима пользователя (если включено разрешение на его использование);

31 ÷ 28 – резерв.

Таблица 10 – Группа регистров целочисленного АЛУ типа J (J-IALU)

Имя	Номер	Значение по умолчанию	Имя	Номер	Значение по умолчанию
J0	0x 0180	Не определено	J16	0x 0190	Не определено
J1	0x 0181	Не определено	J17	0x 0191	Не определено
J2	0x 0182	Не определено	J18	0x 0192	Не определено
J3	0x 0183	Не определено	J19	0x 0193	Не определено
J4	0x 0184	Не определено	J20	0x 0194	Не определено
J5	0x 0185	Не определено	J21	0x 0195	Не определено
J6	0x 0186	Не определено	J22	0x 0196	Не определено
J7	0x 0187	Не определено	J23	0x 0197	Не определено
J8	0x 0188	Не определено	J24	0x 0198	Не определено
J9	0x 0189	Не определено	J25	0x 0199	Не определено
J10	0x 018A	Не определено	J26	0x 019A	Не определено
J11	0x 018B	Не определено	J27	0x 019B	Не определено

Имя	Номер	Значение по умолчанию	Имя	Номер	Значение по умолчанию
J12	0x 018C	Не определено	J28	0x 019C	Не определено
J13	0x 018D	Не определено	J29	0x 019D	Не определено
J14	0x 018E	Не определено	J30	0x 019E	Не определено
J15	0x 018F	Не определено	J31	0x 019F	0x0000 0000

Таблица 11 – Группа регистров целочисленного АЛУ типа K (K-IALU)

Имя	Номер	Значение по умолчанию	Имя	Номер	Значение по умолчанию
K0	0x 01A0	Не определено	K16	0x 01B0	Не определено
K1	0x 01A1	Не определено	K17	0x 01B1	Не определено
K2	0x 01A2	Не определено	K18	0x 01B2	Не определено
K3	0x 01A3	Не определено	K19	0x 01B3	Не определено
K4	0x 01A4	Не определено	K20	0x 01B4	Не определено
K5	0x 01A5	Не определено	K21	0x 01B5	Не определено
K6	0x 01A6	Не определено	K22	0x 01B6	Не определено
K7	0x 01A7	Не определено	K23	0x 01B7	Не определено
K8	0x 01A8	Не определено	K24	0x 01B8	Не определено
K9	0x 01A9	Не определено	K25	0x 01B9	Не определено
K10	0x 01AA	Не определено	K26	0x 01BA	Не определено
K11	0x 01AB	Не определено	K27	0x 01BB	Не определено
K12	0x01AC	Не определено	K28	0x 01BC	Не определено
K13	0x01AD	Не определено	K29	0x 01BD	Не определено
K14	0x 01AE	Не определено	K30	0x 01BE	Не определено
K15	0x 01AF	Не определено	K31	0x 01BF	0x0000 0000

Таблица 12 – Группа регистров циклического буфера целочисленного АЛУ типа J (J-IALU)

Имя	Номер	Значение по умолчанию
JB0	0x 01C0	Не определено
JB1	0x 01C1	Не определено
JB2	0x 01C2	Не определено
JB3	0x 01C3	Не определено
JL0	0x 01C4	Не определено
JL1	0x 01C5	Не определено
JL2	0x 01C6	Не определено
JL3	0x 01C7	Не определено
резервные	0x 01C8– 0x 01DA	Не определено
JUSP	0x 01DB	Не определено
резервные	0x 01DC– 0x 01DF	Не определено

Таблица 13 – Группа регистров циклического буфера K-IALU

Имя	Номер	Значение по умолчанию
KB0	0x 01E0	Не определено
KB1	0x 01E1	Не определено
KB2	0x 01E2	Не определено
KB3	0x 01E3	Не определено
KL0	0x 01E4	Не определено
KL1	0x 01E5	Не определено
KL2	0x 01E6	Не определено
KL3	0x 01E7	Не определено
резервные	0x 01E8– 0x 01FA	Не определено
KUSP	0x 01FB	Не определено
резервные	0x 01FC– 0x 01FF	Не определено

6.7.1 Регистры статуса целочисленного АЛУ (J31/JSTAT и K31/KSTAT)

Регистры статуса JSTAT (для J-IALU) и KSTAT (для K-IALU) хранят состояния флагов и обновляются в результате выполнения различных команд целочисленного АЛУ. При записи возможно обращение к регистру JSTAT как J31. При использовании в качестве операнда в арифметических, логических и функциональных операциях J31 рассматривается как ноль. Регистры J31/JSTAT (K31/KSTAT) (значение сброса = 0x0000 0000) описаны в таблице 14.

Таблица 14 – Регистр состояния целочисленного АЛУ

Бит	Имя	Назначение
0	Z	Признак нулевого результата операции
1	N	Признак отрицательного результата операции
2	V	Признак переполнения результата операции
3	C	Признак переноса
4	-	Резервные. Всегда 0

6.8 Группы регистров устройства управления

Группа содержит 32 регистра, однако используются не все регистры. Часть регистров зарезервирована. Все регистры группы выполняют специальные функции. Особенностью данной группы является то, что регистры доступны только как однословные.

Таблица 15 – Группа регистров устройства управления

Имя	Описание	Номер	Значение по умолчанию
CJMP	Вычисленный адрес перехода	0x0340	Не определено
-	-	0x0341	-
RETI	Возврат из прерывания	0x0342	Не определено
RETIB	Псевдоним RETI, для вложенных прерываний	0x0343	Не определено

Имя	Описание	Номер	Значение по умолчанию
RETS	Возврат из исключительной ситуации	0x0344	Не определено
DBGE	Возврат из эмуляции	0x0345	Не определено
-	-	0x0346–0x0347	-
LC0	Счетчик циклов 0	0x0348	Не определено
LC1	Счетчик циклов 1	0x0349	Не определено
-	-	0x034A–0x034B	-
XPAGE	Регистр управления дополнительной областью BTB (для микросхем с ревизии 3)	0x 034C	0
-	-	0x034D–0x034F	
IVSW	Адрес-вектор для программных исключений	0x 0350	Не определено
-	-	0x0351–0x0353	-
FLAGREG	Регистр управления флагами процессора	0x0354	0x0000 0000
FLAGREGST	Установка бит регистра флага	0x0355	Не определено
FLAGREGCL	Сброс бит регистра флага	0x0356	Не определено
-	-	0x0357	-
SQCTL	Регистр управления	0x0358	0x0000 0204
SQCTLST	Установка бит регистра управления	0x0359	Не определено
SQCTLCL	Сброс бит регистра управления	0x035A	Не определено
SQSTAT	Регистр статуса, только чтение	0x035B	0x0000 FF04
SFREG	регистр специальных статических флагов	0x035C	0x0000 0000
-	-	0x035D	-
EXT_FUN	Включение расширенных функций	0x035E	-
-	-	0x035F	-

6.8.1 Регистр XPAGE (для микросхем с ревизии 3)

Таблица 16 – Биты регистра управления XPAGE

Бит	Имя	Назначение
20:0	-	Не используются. При чтении всегда равны 0
31:21	XPAGE	Базовый адрес области памяти (старшие 11 бит), кэшируемой в буфере BTB

По умолчанию в BTB кэшируются только адреса переходов с опцией (P) для области виртуальных адресов диапазона 0-0x1FFFFF. По сути это можно назвать нулевой страницей размером 8 Мбайт. Дополнительную страницу размером 8 Мбайт можно задать с помощью регистра XPAGE. При записи в регистр младшие 21 бит игнорируются. Таким образом, страница должна быть выровнена по границе 8 Мбайт.

6.8.2 Регистр управления флагом (FLAGREG)

Регистр FLAGREG – 32-битный регистр, который контролирует направление контакта флага (вход или выход) и обеспечивает значение на контакте (1 или 0), когда контакт сконфигурирован как выход. Регистр FLAGREG (значение сброса = 0x0000 0000) описывается в таблице 17.

Таблица 17 – Биты регистра управления флагом

Бит	Имя	Назначение
3:0	FLAGx_EN	Направление флага: 0 – прием; 1 – выдача
7:4	FLAGx_OUT	Значение флага при выдаче
31–8	-	Не используются и всегда равны нулю

Контакт FLAG0 использует биты 0 и 4, FLAG1 использует биты 1 и 5, FLAG2 использует биты 2 и 6, FLAG3 использует биты 3 и 7.

6.8.3 Регистр управления SQCTL

Устройство управления последовательностью команд контролируется значением регистра SQCTL. После сброса значение регистра равно 0x0000_0204. Подробное описание всех бит регистра приведено в таблице 18.

Таблица 18 – Биты регистра управления SQCTL

Бит	Имя	Назначение
0, 1	-	
2	GIE	Глобальное разрешение прерываний: 1 – прерывания разрешены; 0 – запрещены
3	SWIE	Разрешение генерации исключительных ситуаций (программных прерываний): 1 – разрешено; 0 – запрещено
4-6	-	
7	KUSP	Выбор указателя стека в режиме супервизора (NMOD=1) привключенном режиме работы с раздельными указателями (USP=1): 0 – используется регистр KSP; 1 – используется регистр USP
8	DBGEN	Разрешение отладки: 1 – разрешено; 0 – запрещено
9	NMOD	Режим работы: 0 – пользователь; 1 – супервизор

Бит	Имя	Назначение
10	TRCBEN	Включение буфера трасс: 0 – буфер трасс выключен и хранит старое значение; 1 – включен и следит за исполнением переходов
11	TRCBEXEN	Разрешение генерации исключительной ситуации от буфера трасс: 0 – запрещено; 1 – разрешено
12	DF_IEEE	Разрешение использования команд обработки данных с плавающей запятой двойной точности: 0 – используется расширенная точность; 1 – код команды расширенной точности соответствует коду команды двойной точности
13	BYTE_ON	Разрешение использования кодов команды TRAP 32-63 в качестве кодов префикса для создания новых команд: 0 – команды TRAP 32-63 используются для генерации исключений; 1 – команды TRAP 32-63 используются в качестве префикса
14	USP	Разрешение использования двух указателей стеков (регистр JK27): 0 – регистр JK27 один для пользователя и супервизора; 1 – для пользователя и супервизора используются разные регистры
31-15	-	Зарезервированы и не используются. При чтении всегда равны нулю

Биты 12, 13, 14 регистра управления соответствуют дополнительным возможностям процессора, и их включение должно быть дополнительно разрешено (см. бит EXT_MODE регистра SQSTAT, а также регистр EXT_FUN).

6.8.4 Регистр управления (SQCTLST). Установка бит

Бит SQCTLST является псевдонимом для SQCTL. При записи по данному адресу «1» в любом бите записываемых данных устанавливает соответствующий бит в SQCTL, тогда как «0» в записываемых данных не меняет значение бита.

6.8.5 Регистр управления (SQCTLCL). Сброс бит

Бит SQCTLCL является псевдонимом для записи в SQCTL. При записи по данному адресу «0» в любом бите записываемых данных сбрасывает соответствующий бит в SQCTL, тогда как «1» в записываемых данных не меняет значение бита.

6.8.6 Регистр статических флагов (SFREG)

Статические флаги используются как статические копии условий. При желании сохранить значение условия до того, как другая команда поменяет его значение, можно скопировать значение в регистр SFREG и использовать его позже как условие. Все статические флаги условий группируются в регистре SFREG. Регистр SFREG после сброса равен нулю и описание его разрядов приведено в таблице 19.

Таблица 19 – Биты регистра статических флагов

Бит	Имя	Назначение
0	GSCF0	Статический флаг целочисленных АЛУ
1	GSCF1	Статический флаг целочисленных АЛУ
2	XSCF0	Статический флаг вычислительного модуля X
3	XSCF1	Статический флаг вычислительного модуля X
4	YSCF0	Статический флаг вычислительного модуля У
5	YSCF1	Статический флаг вычислительного модуля У
31-6	-	Не используются и всегда равны нулю

6.8.7 Регистр включения расширенных функций (EXT_FUN)

Регистр состоит из одного бита, значение которого может быть прочитано в регистре SQSTAT, бит 30 – EXT_MODE. Этот бит по сбросу устанавливается в 1(все расширенные функции включены) и меняет свое значение на противоположное всякий раз после выполнения команды 0x8B5E1A3C. Этот код суть команда пересылки из регистра EXT_FUN в этот же регистр EXT_FUN (EXT_FUN=EXT_FUN).

6.8.8 Регистр статуса устройства управления (SQSTAT)

Данный регистр доступен только для чтения и содержит информацию о текущем статусе устройства управления. Подробное описание разрядов регистра приведено в таблице 20.

Таблица 20 – Биты регистра SQSTAT

Бит	Имя	Назначение
1:0	MODE	Текущий режим работы процессора: 00 – пользователь; 01 – супервизор; 11 – эмулятор. Другие состояния невозможны
2	IDLE	Процессор в состоянии ожидания прерывания (если 1)
7:3	SPVCMD	Значение параметра последней выполненной команды TRAP
11:8	EXCAUSE	Код последней исключительной ситуации: 0000 – команда TRAP; 0001 – точка наблюдения или буфер трасс; 0010 – команда плавающей запятой; 0011 – ошибочная линия команд; 0100 – невыровненный доступ; 0101 – попытка записи защищенного регистра; 0110 – переполнения счетчика статистики; 0111 – чтение из области 0x0C00_0000 – 0x0FFF_FFFF; 1000 – x; 1001 – ошибочная ситуация в модуле H264; 1010 – x;

Бит	Имя	Назначение
		1011 – x; 1100 – x; 1101 – x; 1110 – x; 1111 – не было ошибок с момента сброса
15:12	EMCAUSE	Код ситуации, вызвавшей переход в режим эмулятора: 1111 – не было перехода в режим эмулятора после сброса; 0000 – команда EMUTRAP; 0001 – запрос JTAG; 0010 – сработала точка наблюдения. Все другие коды зарезервированы и не могут присутствовать
16	FLAG0	Состояние внешнего контакта FLAG0
17	FLAG1	Состояние внешнего контакта FLAG1
18	FLAG2	Состояние внешнего контакта FLAG2
19	FLAG3	Состояние внешнего контакта FLAG3
20	EXE_ISR	Флаг устанавливается в 1 при: – переходе процессора в состояние обработки прерывания; – записи данных в регистр RETIB. Установленный флаг запрещает прерывания. Флаг сбрасывается: – командой выхода из прерывания RTI; – командой RDS (если нет обработки исключительной ситуации); – чтением регистра RETIB. Сброшенный флаг означает возможность обработки нового прерывания процессором
21	EXE_SWI	Признак того, что процессор находится в состоянии обработки исключительной ситуации (когда равен 1)
22	EMUL	Признак того, что процессор находится в режиме эмулятора (когда равен 1)
23	ISR_MODE	Флаг устанавливается в 1 при переходе процессора в состояние обработки прерывания. Флаг сбрасывается: – командой выхода из прерывания RTI; – командой RSD (если нет обработки исключиткльной ситуации). Установленный флаг означает состояние обработки прерывания в режиме супервизора
24	BTBEN	Дублирование бита 28
25	-	BTBLK
26, 27	-	
28	BTBEN	Буфер предсказания переходов включен (если 1) или выключен (если 0)
29	-	BTBLK
30	EXT_MODE	Флаг включения расширенных операций процессора. После аппаратного сброса значение 1 (расширенные операции разрешены)
31	I_LOCK	Флаг блокировки прерываний (1 – прерывания блокированы)

6.9 Группа регистров устройства защиты памяти

Процессор имеет несколько режимов работы и особенностью режима пользователя является то, что для него ограничен доступ к некоторым аппаратным ресурсам. Однако внутренняя и внешняя память одинаково доступны как в режиме супервизора, так и в режиме пользователя. В ряде приложений желательно защитить код или данные системы от разрушения некорректным поведением пользовательской программы. Для этого в процессоре имеется группа регистров, с помощью которой можно описать отдельные регионы внутренней памяти и способы доступа к ним. Также эта группа регистров содержит регистры управления кэш-памятью. Все регистры данной группы доступны только как однословные. Запись к регистрам разрешена, только если включен режим использования расширенных функций процессора (бит регистра EXT_FUN).

Подробное описание регистров приведено в разделе 7 «Архитектура кэш-памяти процессора».

Таблица 21 – Группа регистров устройства защиты памяти

Имя	Описание	Номер	Значение по умолчанию
Группа 0x1E			
MS0_C	Регистр управления кэшированием данных MS0	0x03C0	0
MS0_WT	Регистр управления сквозной записью MS0	0x03C1	0
MS0_CI	Регистр управления кэшированием команд MS0	0x03C2	0
		0x03C3 – 0x03C7	
MS1_C	Регистр управления кэшированием данных MS1	0x03C8	0
MS1_WT	Регистр управления сквозной записью MS1	0x03C9	0
MS1_CI	Регистр управления кэшированием команд MS1	0x03CA	0
		0x03CB – 0x03CF	
SDR_C	Регистр управления кэшированием данных SDRAM	0x03D0	0
SDR_WT	Регистр управления сквозной записью SDRAM	0x03D1	0
SDR_CI	Регистр управления кэшированием команд SDRAM	0x03D2	0
		0x03D3 –0x03DF	
Группа 0x1F			
PU0	Регистр защиты 0	0x03E0	Не определено
PU1	Регистр защиты 1	0x03E1	Не определено
PU2	Регистр защиты 2	0x03E2	Не определено
PU3	Регистр защиты 3	0x03E3	Не определено
PU4	Регистр защиты 4	0x03E4	Не определено
PU5	Регистр защиты 5	0x03E5	Не определено

Имя	Описание	Номер	Значение по умолчанию
PU6	Регистр защиты 6	0x03E6	Не определено
PU7	Регистр защиты 7	0x03E7	Не определено
PU_SR	Регистр состояния	0x03E8	0
-	-	0x03E9 – 0x03FB	0
PU_CR	Регистр управления	0x03FC	0
IDC_CR	Управление кэш-памятью	0x03FD	0
-	-	0x03FE – 0x03FF	0

6.9.1 Регистры защиты (PUx)

Каждый регистр защиты позволяет описать определенную область памяти и способы доступа к ней. Работа регистра защиты включается соответствующим битом в регистре управления. Подробное описание бит регистров защиты приведено в таблице 22.

Таблица 22 – Регистр PU

Бит	Имя	Назначение
10:0	STA	Начальный адрес модуля памяти. Соответствует физическому адресу памяти образуемому как {11'b0,START_A[10:0],10'b0}
11	-	
22:12	ENDA	Конечный адрес модуля памяти. Соответствует физическому адресу памяти образуемому как {11'b0,END_A[10:0],10'b11_1111_1111}
23	-	
25:24	JK_AP	Режим доступа к модулю со стороны шин J и K: 00 – полный доступ; 01 – супервизор чтение и запись, пользователь чтение; 10 – всем только чтение; 11 – супервизор только чтение
27:26	I_AP	Режим доступа к модулю со стороны шины I: 00 – полный доступ; 01 – только супервизор; 10 – доступ запрещен; 11 – доступ запрещен
29:28	H_AP	Режим доступа к модулю со стороны шины S (хост, DMA): 00 – чтение и запись; 01 – только чтение; 10 – доступ запрещен; 11 – доступ запрещен
31:30	-	Резерв

Поля STA и ENDA используются для сравнения со значениями адресных шин J, K, I, S. При этом в сравнении участвуют только биты 20:10 указанных шин. Проверка выполняется только при доступе к внутренней памяти. Так для K шины доступа попадание в некоторый модуль означает истинность следующего выражения:

Hit_PU = (K[20:10]>=STA) && (K[20:10]<=ENDA).

Минимальный размер модуля равен странице из 1 К слов. Модуль может рассматриваться как множество из 1 К страниц.

При нарушении прав доступа к модулю памяти вырабатывается исключительная ситуация. Исключение составляет контроль доступа со стороны шины S. Нарушение прав доступа будет вызывать блокировку записи и установку флага ошибки. Исключительная ситуация вырабатываться не будет. Необходимо отметить, что исключительная ситуация всегда форсируется после выполнения команды. Поэтому в случае срабатывания защиты по доступу со стороны шины I, процессор выполнит одну линию команд из защищенной области, прежде чем перейдет на обработку исключительной ситуации. Разобраться какое событие вызвало срабатывание исключительной ситуации можно анализируя регистр состояния устройства управления, а также регистр состояния модуля защиты.

6.9.2 Регистр состояния (PU_SR)

Регистр хранит информацию о событии, которое вызвало срабатывание модуля защиты. После сброса значение регистра равно нулю. Подробное описание бит регистра состояния приведено в разделе 7 «Архитектура кэш-памяти процессора».

6.9.3 Регистр управления (PU_CR)

Регистр осуществляет включение в работу всех функций модуля защиты, а также функций управления кэш-памятью. После сброса значение регистра равно нулю. Подробное описание бит регистра управления приведено в разделе 7 «Архитектура кэш-памяти процессора».

6.9.4 Регистр управления кэшированием данных

Регистр используется для задания атрибута кэшируемости для отдельных страниц банков внешней памяти. Как известно процессор может адресовать три банка внешней памяти: MS0, MS1, SDRAM. При этом модули MS0, MS1 имеют деление на страницы по 128 К слов, а модуль SDRAM имеет деление на страницы по 512 К слов. Каждый бит регистра соответствует странице памяти. Так нулевой бит – нулевой странице, первый – первой и т. д. Итого можно определить до 32•128 = 4 М слов памяти модулей MS0, MS1 и до 16 М слов динамической памяти.

6.9.5 Регистр управления сквозной записью

Аналогично управлению кэшируемости страниц, имеется возможность для каждой страницы указать бит стратегии записи при попадании в кэш: «0» – запись только в кэш, «1» – запись в кэш и во внешнюю память.

6.9.6 Регистр управления кэшированием команд

Аналогично управлению кэшируемостью страниц для данных имеется возможность для каждой страницы указать бит кэшируемости при обращении за командами: «0» – не кэшируется, «1» – кэшируется.

Более подробно о функциях регистров управления кэшами см. приведено в разделе 7 «Архитектура кэш-памяти процессора».

6.10 Группы регистров отладки

Группы отладки описываются в таблице 23. Доступ к регистрам отладки возможет только как к отдельным словам. К регистрам отладки может быть применена только команда пересылки «регистр-регистр» или командам немедленной загрузки данных. Эти регистры не могут быть загружены из памяти или сохранены напрямую в память. После каждой записи в регистры группы отладки происходит перезапуск конвейера процессора.

Таблица 23 – Группа регистров отладки

Имя	Описание	Адрес	Значение по умолчанию
WP0CTL	Управление точкой наблюдения 0	0x0360	0x0000 0000
WP1CTL	Управление точкой наблюдения 1	0x0361	0x0000 0000
WP2CTL	Управление точкой наблюдения 2	0x0362	0x0000 0000
-	-	0x0363	-
WP0STAT	Состояние точки наблюдения 0; (RO)	0x0364	0x0000 0000
WP1STAT	Состояние точки наблюдения 1; (RO)	0x0365	0x0000 0000
WP2STAT	Состояние точки наблюдения 2; (RO)	0x0366	0x0000 0000
-	-	0x0367	-
WP0L	Нижний адрес точки наблюдения 0	0x0368	Не определено
WP0H	Верхний адрес точки наблюдения 0	0x0369	Не определено
WP1L	Нижний адрес точки наблюдения 1	0x036A	Не определено
WP1H	Верхний адрес точки наблюдения 1	0x036B	Не определено
WP2L	Нижний адрес точки наблюдения 2	0x036C	Не определено
WP2H	Верхний адрес точки наблюдения 2	0x036D	Не определено
WPDR	Регистр данных точки наблюдения 1	0x036E	0
WPMR	Регистр маски точки наблюдения 1.	0x036F	0
CCNT0	Нижняя часть счетчика циклов	0x0370	0
CCNT1	Верхняя часть счетчика циклов	0x0371	0
PRFM	Маска монитора производительности	0x0372	0x0000 0000
PRFCNT	Счетчик монитора производительности	0x0373	0x0000 0000
TRCBMASK	Маска буфера трассировки (RO)	0x0374	0x0000 0000
TRCBPTR	Указатель буфер трассировки (RO)	0x0375	0x0000 0000
-	-	0x0376 – 0x037B	-
TRCBVAL	Допустимый буфер трассировки (RO)	0x 037C	0x0000 0000
-	-	0x037D – 0x037F	-
TRCB31–0	Буфер трассировки (RO)	0x0140 – 0x015F	0x0000 0000

Точка наблюдения представляет собой механизм описания условия, при выполнении которого модуль отладки формирует запрос к процессору на обработку исключительной ситуации или на переход в режим эмуляции.

6.10.1 Регистр управления точкой наблюдения (WPxCTL)

У каждой из трех точек наблюдения есть регистр управления (WP0CTL, WP1CTL и WP2CTL), который используется для определения действий точки наблюдения. Значением сброса регистров WPxCTL является 0x0000 0000. Каждая из точек наблюдения осуществляет мониторинг определенных шин процессорного ядра: точка 0 – шину I, точка 1 – шины J и K, точка 2 – шину S.

Подробное описание разрядов регистра WP0CTL приведено в таблице 24.

Таблица 24 – Регистр WP0CTL

Бит	Имя	Назначение
1:0	OPMODE	Режим работы: 00 – выключена; 01 – включена. Анализ совпадения адреса; 10 – включена. Анализ попадания в интервал; 11 – включена. Анализ выхода вне интервала
3:2	EXTYPE	Действия при выполнении условия: 00 – нет действий; 01 – программное исключение; 10 – переход в режим эмулятора; 11 – резерв
4	SSTP	Режим пошагового выполнения команд: 1 – включен; 0 – выключен
15:5	-	Зарезервировано
31:16	WP0CNT	Счетчик количества срабатываний точки наблюдения до генерации исключительной ситуации

Подробное описание разрядов регистра WP1CTL приведено в таблице 25.

Таблица 25 – Регистр WP1CTL

Бит	Имя	Назначение
1:0	OPMODE	Режим работы: 00 – выключена; 01 – включена. Анализ совпадения адреса; 10 – включена. Анализ попадания в интервал; 11 – включена. Анализ выхода вне интервала
3:2	EXTYPE	Действия при выполнении условия: 00 – нет действий; 01 – программное исключение; 10 – переход в режим эмулятора; 11 – резерв
4	READ	Мониторинг циклов чтения: 1 – включен; 0 – выключен

Бит	Имя	Назначение
5	WRITE	Мониторинг циклов записи: 1 – включен; 0 – выключен
6	Jbus	Мониторинг шины J: 1 – включен; 0 – выключен
7	Kbus	Мониторинг шины K: 1 – включен; 0 – выключен
15:8	-	Зарезервировано
31:16	WP1CNT	Счетчик количества срабатываний точки наблюдения до генерации исключительной ситуации

Подробное описание разрядов регистра WP2CTL приведено в таблице 26.

Таблица 26 – Регистр WP2CTL

Бит	Имя	Назначение
1:0	OPMODE	Режим работы: 00 – выключена; 01 – включена. Анализ совпадения адреса; 10 – включена. Анализ попадания в интервал; 11 – включена. Анализ выхода вне интервала
3:2	EXTYPE	Действия при выполнении условия: 00 – нет действий; 01 – программное исключение; 10 – переход в режим эмулятора; 11 – резерв
4	READ	Мониторинг циклов чтения: 1 – включен; 0 – выключен
5	WRITE	Мониторинг циклов записи: 1 – включен; 0 – выключен
15:6	-	Зарезервировано
31:16	WP2CNT	Счетчик количества срабатываний точки наблюдения до генерации исключительной ситуации

6.10.2 Регистры состояния точки наблюдения (WPxSTAT)

У каждой из трех точек наблюдения есть регистр состояния (WP0STAT, WP1STAT и WP2STAT). Значениями сброса регистров WPxSTAT являются 0x0000 0000. Подробное описание разрядов регистра приведено в таблице 27.

Таблица 27 – Регистр WPxSTAT

Бит	Имя	Назначение
15:0	WPxCNT	Текущее значение счетчика точки наблюдения. Счетчик вычитает 1 каждый раз при срабатывании точки
17:16	EX	Состояние точки наблюдения: 00 –выключена; 01 – активна; 10 – резерв; 11 – переполнился счетчик
31:18	-	Всегда 0

6.10.3 Регистры указателя адреса точки наблюдения (WPxL/WPxH)

Указатели адреса точки наблюдения являются 32-битными указателями, определяющими адрес или диапазон адресов точки наблюдения. После сброса их значение не определено.

6.10.4 Регистр маски монитора производительности (PRFM)

Регистр маски монитора производительности (PRFM) определяет, какие события в процессоре отслеживаются и подсчитываются счетчиком монитора производительности (PRFCNT). Значением сброса регистра PRFM является 0x0000 0000. Подробное описание разрядов регистра приведено в таблице 28.

Таблица 28 – Регистр PRFM

Бит	Имя	Назначение
7:0	-	Резерв
8	Jexe	Подсчет команд, выполненных на линии конвейера J: 1 – включен; 0 – выключен
9	Kexe	Подсчет команд, выполненных на линии конвейера K: 1 – включен; 0 – выключен
10	Xexe	Подсчет команд, выполненных на линиях конвейера X1 и X2: 1 – включен; 0 – выключен
11	Yexe	Подсчет команд, выполненных на линиях конвейера У1 и У2: 1 – включен; 0 – выключен
12	Sexe	Подсчет команд, выполненных на линии конвейера S: 1 – включен; 0 – выключен
15:13	-	Резерв

Бит	Имя	Назначение
16	STALL	Подсчет команд остановов конвейера вызванных ожиданием данных или другими блокирующими конвейер ситуациями: 1 – включен; 0 – выключен
17	BTB_true	Подсчет количества правильных предсказаний буфера переходов: 1 – включен; 0 – выключен
18	ABORT	Подсчет количества программных исключительных ситуаций: 1 – включен; 0 – выключен
19	Uexe	Подсчет количества линий команд, выполненных в режиме пользователя: 1 – включен; 0 – выключен
20	BTB_false	Подсчет количества ошибочных предсказаний буфера переходов: 1 – включен; 0 – выключен
21	BTB_load	Подсчет количества загрузок в буфер переходов: 1 – включен; 0 – выключен
31:22	-	Резерв

Все события, которые отслеживаются монитором производительности, суммируются по «ИЛИ» и при их наступлении происходит увеличение счетчика PRFM на 1. Логичным является разрешение подсчета одного условия из многих.

6.10.5 Регистр счетчика монитора производительности (PRFCNT)

Назначение счетчика производительности – увеличивать свое значение на 1 каждый раз, когда происходит отслеживаемое событие. Отслеживаемое событие определяется регистром маски монитора производительности (PRFM). Регистр очищается после сброса.

6.10.6 Регистры счетчика циклов (CCNTx)

Длина счетчика циклов равна 64 разряда, однако доступен он как два универсальных 32-разрядных регистра – CCNT0 и CCNT1. После сброса значение счетчика равно нулю.

В связи с тем, что счетчик 64 бита, а доступ возможен только к частям регистра, необходимо соблюдение специальной процедуры чтения и записи регистра. При чтении сначала считывается нижняя часть (CCNT0), а затем верхняя часть (CCNT1). При чтении нижней части верхняя часть копируется в буфер и это гарантирует ее корректное значение. При записи сначала пишется нижняя часть (она попадает в буфер), а затем верхняя. При записи верхней полное 64-битное значение попадает в счетчик.

6.10.7 Регистры указателя и буфера трассировки (TRCBx/TRCBPTR)

Каждый раз, когда процессор выполняет переход на непоследовательно исполняемую команду, адрес непоследовательно выбранной команды (адрес перехода) записывается в один из регистров буфера трассировки. Первая запись осуществляется в буфер трассировки 0, вторая – в 1 и далее циклично. Указатель буфера трассировки определяет последний записанный буфер трассировки.

Таблица 29 – Группа регистров буфера трассировки

Имя	Описание	Адрес	Значение по умолчанию
TRCB0	Буфер трассировки 0; только чтение	0x0140	0x0000 0000
TRCB1	Буфер трассировки 1; только чтение	0x0141	0x0000 0000
TRCB2	Буфер трассировки 2; только чтение	0x0142	0x0000 0000
TRCB3	Буфер трассировки 3; только чтение	0x0143	0x0000 0000
TRCB4	Буфер трассировки 4; только чтение	0x0144	0x0000 0000
TRCB5	Буфер трассировки 5; только чтение	0x0145	0x0000 0000
TRCB6	Буфер трассировки 6; только чтение	0x0146	0x0000 0000
TRCB7	Буфер трассировки 7; только чтение	0x0147	0x0000 0000
TRCB8	Буфер трассировки 8; только чтение	0x0148	0x0000 0000
TRCB9	Буфер трассировки 9; только чтение	0x0149	0x0000 0000
TRCB10	Буфер трассировки 10; только чтение	0x014A	0x0000 0000
TRCB11	Буфер трассировки 11; только чтение	0x014B	0x0000 0000
TRCB12	Буфер трассировки 12; только чтение	0x014C	0x0000 0000
TRCB13	Буфер трассировки 13; только чтение	0x014D	0x0000 0000
TRCB14	Буфер трассировки 14; только чтение	0x014E	0x0000 0000
TRCB15	Буфер трассировки 15; только чтение	0x014F	0x0000 0000
TRCB16	Буфер трассировки 16; только чтение	0x0150	0x0000 0000
TRCB17	Буфер трассировки 17; только чтение	0x0151	0x0000 0000
TRCB18	Буфер трассировки 18; только чтение	0x0152	0x0000 0000
TRCB19	Буфер трассировки 19; только чтение	0x0153	0x0000 0000
TRCB20	Буфер трассировки 20; только чтение	0x0154	0x0000 0000
TRCB21	Буфер трассировки 21; только чтение	0x0155	0x0000 0000
TRCB22	Буфер трассировки 22; только чтение	0x0156	0x0000 0000
TRCB23	Буфер трассировки 23; только чтение	0x0157	0x0000 0000
TRCB24	Буфер трассировки 24; только чтение	0x0158	0x0000 0000
TRCB25	Буфер трассировки 25; только чтение	0x0159	0x0000 0000
TRCB26	Буфер трассировки 26; только чтение	0x015A	0x0000 0000
TRCB27	Буфер трассировки 27; только чтение	0x015B	0x0000 0000
TRCB28	Буфер трассировки 28; только чтение	0x015C	0x0000 0000
TRCB29	Буфер трассировки 29; только чтение	0x015D	0x0000 0000
TRCB30	Буфер трассировки 30; только чтение	0x015E	0x0000 0000
TRCB31	Буфер трассировки 31; только чтение	0x015F	0x0000 0000

6.10.8 Регистр данных точки наблюдения 1 (WPDR)

Точка наблюдения 1 имеет дополнительные возможности по мониторингу обмена данными ядра процессора с памятью. Выше описывался механизм срабатывания исключительной ситуации в случае совпадения адресов точки наблюдения с адресами шин J и K. Однако имеется возможность дополнить сравнение адресов еще и сравнением данных. При чтении имеется возможность контролировать данные, которые считывает процессор. Если дополнительно к совпадению адресов происходит совпадение прочитанных данных – точка срабатывает. Также можно контролировать и процесс записи данных. Необходимые для контроля данные записываются в регистр WPDR. Необходимо отметить, что шина чтения или записи имеет разрядность 128 бит. Разрядность регистра данных только 32 бита. Поэтому сравнение выполняется только для 32-разрядного слова шины данных определяемого младшими битами шины адреса независимо от того, какое действие выполняется: чтение или запись 64- или 128-разрядного слова.

6.10.9 Регистр маски точки наблюдения 1 (WPMR)

При использовании шины данных в описании точки наблюдения 1 не все биты шины данных могут понадобиться при анализе читаемых или записываемых данных. Регистр маски позволяет установить, какие биты регистра данных WPDR должны участвовать в сравнении. Если бит установлен в «1» – соответствующий ему бит данных сравнивается с битом шины данных. После сброса регистр маски равен нулю и данные не участвуют в работе точки наблюдения.

7 Архитектура кэш-памяти процессора

Процессор имеет встроенные раздельные кэш команд и кэш данных для кэширования данных внешней памяти. В подразделе 5.1 «Ядро процессора» была подробно описана структура конвейера процессора. Отметим основные стадии конвейера, которые задействуются при работе с внутренней памятью (см. рисунок 11).

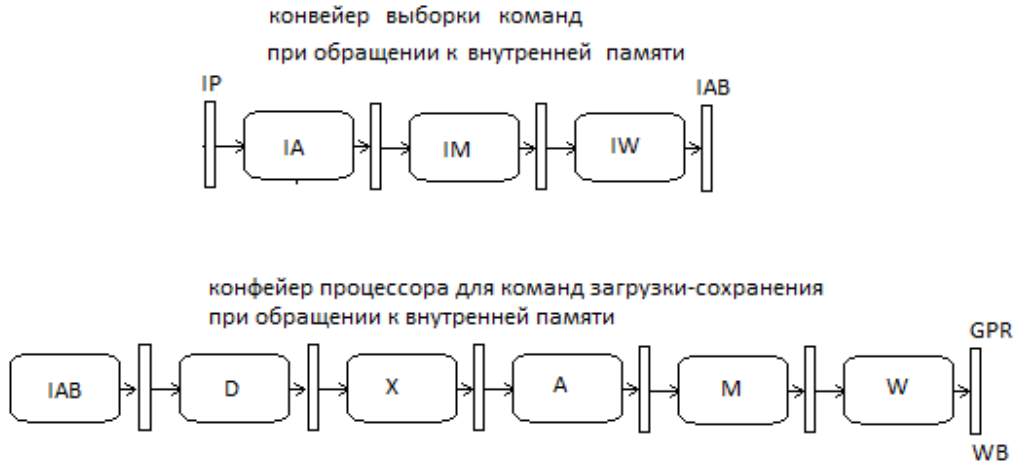


Рисунок 11 – Стадии конвейера процессора

Источником адреса для конвейера выборки команд является указатель на команду (IP). На стадии IA адрес анализируется и при обращении к внутренней памяти направляется в выбранный банк внутренней памяти. На стадии IM происходит чтение накопителя банка памяти и прочитанные данные загружаются в буфер. На стадии IW происходит передача данных из буфера памяти в буфер IAB. Работа вышеописанных стадий команд инициируется только буфером IAB: если в буфере имеется свободное место – формируется запрос на чтение команды.

Буфер команд IAB является источником команд для конвейера обработки. Для команд загрузки (чтения) данных из памяти и для команд сохранения (записи) данных в памяти стадии конвейера представлены на рисунке 11:

– На стадии D происходит декодирование команды и определение операндов источников операции;

– На стадии X происходит вычисление адреса, по которому будет происходить обращение к памяти;

– На стадии A выполняется анализ адреса и определение факта обращения во внутреннюю память;

– При чтении на стадии M выполняется чтение данных из накопителя выбранного банка памяти в буфер;

– На стадии W происходит передача данных из буфера памяти в регистр-приемник процессорного ядра. Если выполняется операция записи, на стадии W происходит запись данных в буфер записи WB.

При обращении к внутренней памяти каждая стадия конвейера выполняется за один такт частоты процессора.

Кэш команд и кэш данных процессора включаются в работу, когда на стадии IA (A) обнаруживается обращение к внешней памяти процессора. В этом случае на стадиях конвейера выборки команд IM и IW, а также на стадиях конвейера выборки данных M и W выполняются другие операции.

Рассмотрим конвейер выборки команд (см. рисунок 12).

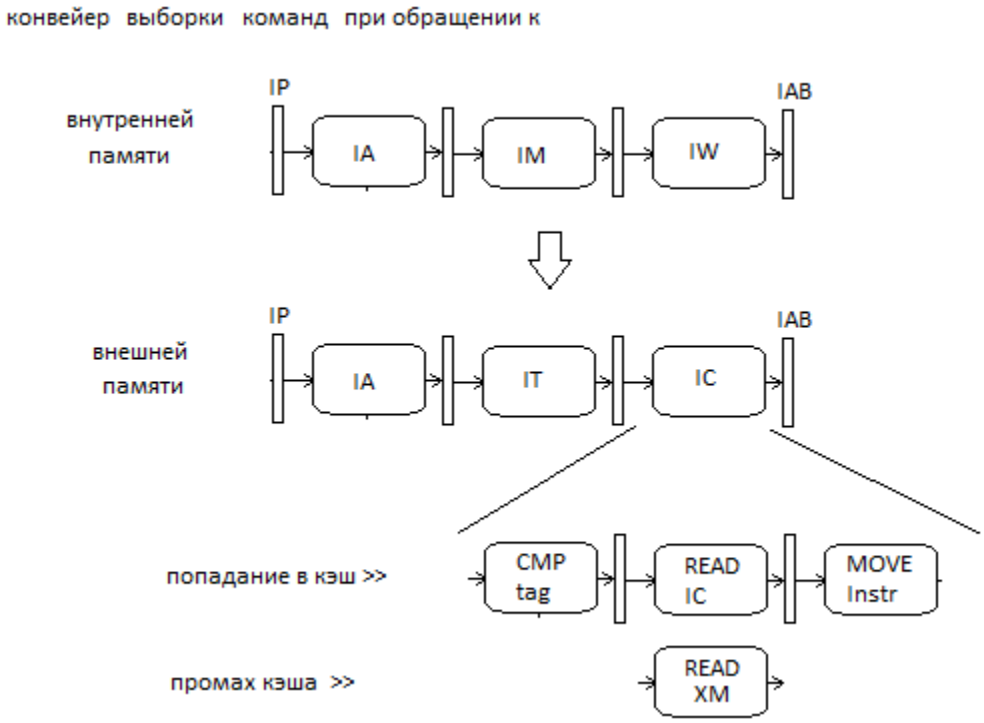


Рисунок 12 – Конвейер выборки команд из внешней памяти

В случае выборки команды из внешней памяти на стадии IT выполняется чтение линии памяти тэгов. Индекс линии определяется адресом чтения. Прочитанное значение линии тэгов передается в буфер. На стадии IC выполняются следующие действия:

1 Сравнение тэгов кэш-памяти с полем тега адреса команды;

2 *При совпадении* тегов – чтение линии команд из накопителя кэш-памяти или *При несовпадении тегов* – обращение к внешней памяти.

Действия, выполняемые на данном этапе зависят от того кэшируемый адрес или некэшируемый.

3 Прочитанная линия команды передается в буфер IAB.

Каждое из перечисленных действий, в случае попадания в кэш, занимает один такт. Таким образом стадия IC будет всегда выполнятся за три такта при попадании в кэш. При промахе кэша количество тактов будет зависеть от соотношения частот процессорного ядра и внешней памяти, от типа памяти и ее характеристик быстродействия. Поскольку быстродействие конвейера определяется быстродействием его самой медленной стадии, то скорость выполнения линейного кода команды при попадании в кэш-команд будет в три раза медленнее, чем выполнение того же кода из внутренней памяти. Однако на самом деле при линейном выполнении кода выборка линий команд будет только в два раза медленнее. Это связано с тем, что устройство

управления успевает отправить два запроса на чтение команд прежде чем поступит сигнал об отсутствии готовности команды.

На рисунке 13 показана временная диаграмма чтения команд с учетом конвейера кэш-памяти команд. Предполагается, что каждый раз происходит попадание в кэш.

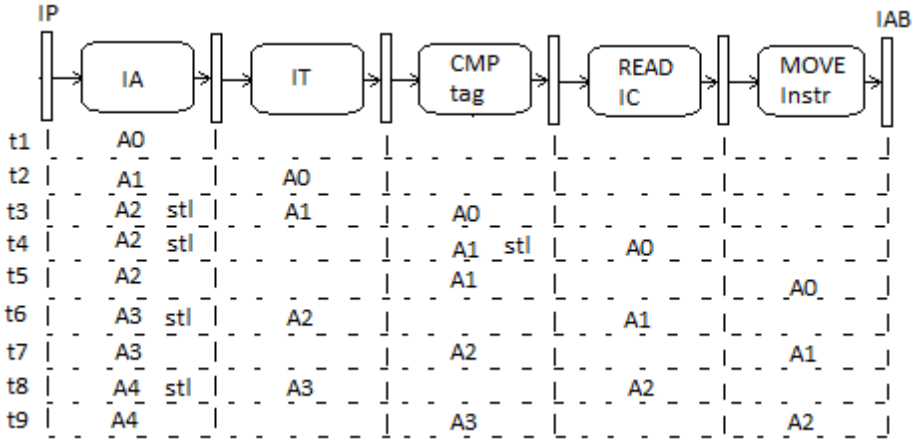


Рисунок 13 – Конвейер команд при попадании в кэш

Из Рисунка 13 видно, что в такте t3 устройство управления получает сигнал отсутствия готовности линии команд, и запрос A2 останавливает свое продвижение. Однако запросы A0 и A1 продолжают движение по конвейеру. В такте t4 видно, что в момент чтения линии команд по запросу A0, запрос A1 останавливается до завершения чтения запроса A0. Начиная с такта t5 возникает ситуация, которая будет повторяться. В данном такте линия команд A0 передается в IAB, и конвейер команд продвигается. В такте t6 запрос A3 останавливается (нет готовности линии команд A1), но запросы A1 и A2 могут продолжать движение. Таким образом, в буфер IAB линии команд будут поступать со скоростью равной половине частоты ядра, т.е. по сравнению с внутренней памятью наблюдается снижение скорости чтения команд в два раза. Дополнительный такт ожидания будет возникать при каждом изменении последовательной выборки команд. Также дополнительные потери производительности будут при выполнении команд переходов. Это связано с тем, что буфер предсказания переходов BTB не выполняет предсказаний переходов для адресов внешней памяти.

При чтении или записи данных, находящихся во внешней памяти, конвейер обработки данных будет иметь вид как на рисунке 14. Действия, выполняемые при чтении данных, практически аналогичны действиям при чтении команд. В случае обращения во внешнюю память на стадии DT выполняется чтение линии памяти тэгов. Индекс линии определяется адресом чтения-записи. Прочитанное значение линии тэгов передается в буфер.

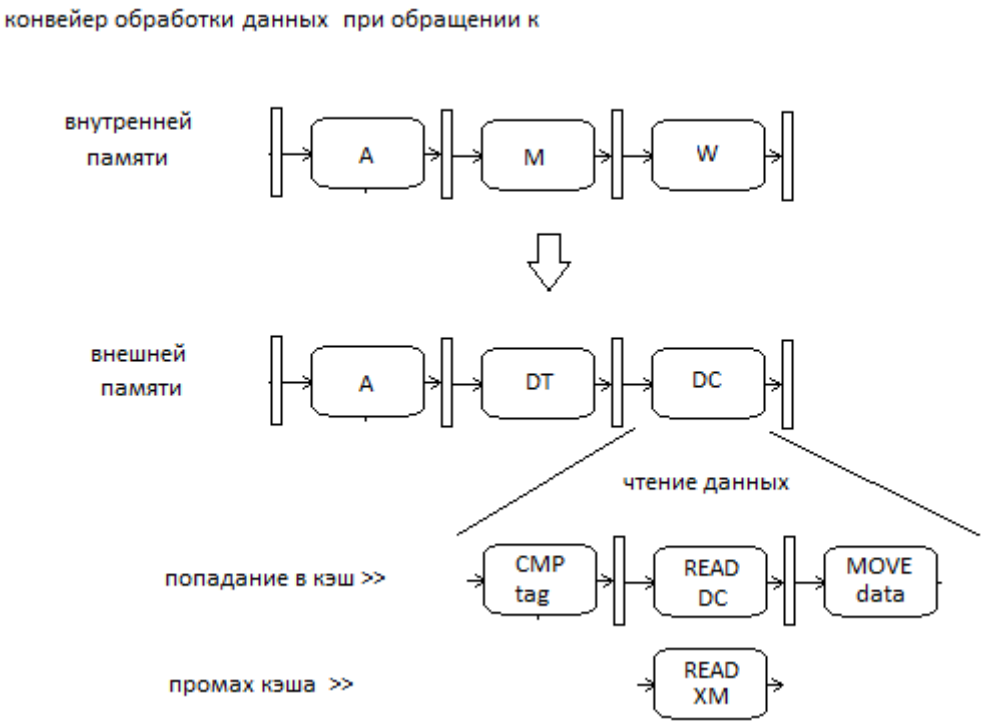


Рисунок 14 – Конвейер обработки данных

На стадии DC выполняются три действия:

1 Сравнение тэгов кэш-памяти данных с полем тега адреса данных:

– Если выполняется *команда записи*, на данном этапе выполнение команды завершается. При попадании в кэш данные записываются в накопитель, а в случае промаха, данные записываются в выходной буфер SOC-интерфейса.

– Если выполняется *команда чтения* данных, необходимо выполнение следующих действий.

2 *При совпадении* тегов происходит чтение линии команд из накопителя кэш-памяти или

При несовпадении тегов происходит обращение к внешней памяти. Действия, выполняемые на данном этапе зависят от того кэшируемый адрес или некэшируемый.

3 Прочитанные данные передаются в регистр-приемник команды загрузки.

Каждое из перечисленных трех действий, в случае попадания в кэш, занимает один такт. Для команды записи данная стадия всегда будет занимать один такт независимо от попадания в кэш. Для команды чтения данных стадия DC будет всегда выполнятся за три такта при попадании в кэш. При промахе кэша количество тактов будет зависеть от соотношения частот процессорного ядра и внешней памяти, от типа памяти и ее характеристик быстродействия.

Таким образом при работе с данными внешней памяти, в случае их размещении в кэш памяти, при чтении всегда будет два дополнительных такта останова конвейера на каждой операции загрузки. Исключением из этого правила будет ситуация, когда в одной линии команд есть два обращения к внешней памяти (от модулей JALU и KALU). Здесь ситуация похожа на кэш команд: два запроса чтения будут обрабатываться друг за другом, и вместо четырех дополнительных тактов останова будет три такта.

Режим выполнения команд из внешней памяти, а также режим обработки данных внешней памяти процессором рассматриваются как неосновные. Поэтому кэш команд и кэш данных не оптимально встроены в конвейер процессора. Для получения максимальной скорости при работе с кэш необходимо увеличение длины конвейера процессора. Однако это привело бы к снижению производительности программ при работе с внутренней памятью. Режим работы с внутренней памятью рассматривается как основной.

7.1 Кэш команд

Кэш команд (далее кэш) имеет размер 8 К 32-разрядных слов и организован в виде 128 линий, каждая из которых имеет четыре входа. Структурная схема кэша команд приведена на рисунке 15. Ассоциативность кэша равна четырем. Каждой линии входа соответствуют четыре квадрослова команд. Каждому квадрослову соответствует свой бит достоверности.

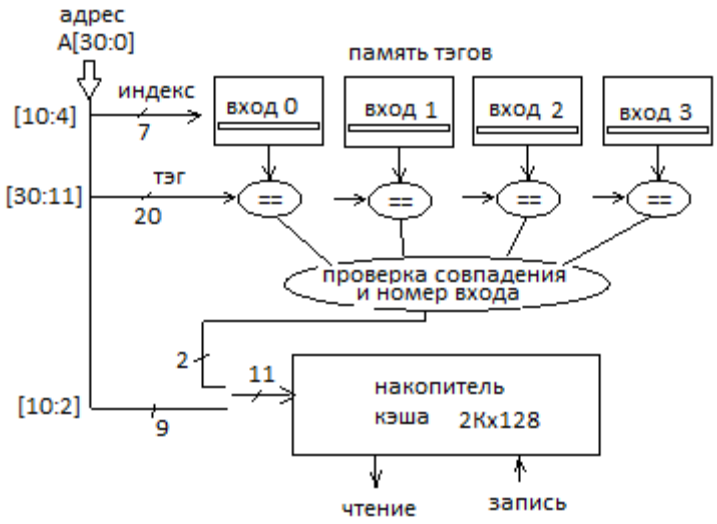


Рисунок 15 – Структура кэша команд

Таким образом 32-разрядный адрес команды при обращении к кэшу имеет следующий вид:

– Биты 1:0 не используются.

– Биты 3:2 указывают на номер квадрослова текущей линии, т.е. выбирают одно из 4-х квадрослов линии.

– Биты 10:4 образуют индекс линии памяти тэгов. Каждая линия имеет 4 входа, это значит, что имеется четыре банка памяти тэгов, и из каждого банка считывается тэг с указанным номером.

– Биты 30:11 образуют тэг, который сравнивается со значением поля тэга каждого банка (входа). По результатам сравнения принимается решение о промахе или попадании в кэш.

– Бит 31 не используется, т.к. всегда равен нулю.

Структура одной записи тэговой памяти представляет собой 20-разрядное поле тэга и четыре бита достоверности для каждого из четырех квадрослов (всего 24 бита).

После чтения тэговой памяти образуются четыре записи для сравнения. Если поле тэга текущего адреса равно одному из полей прочитанных тэгов, и установлен соответствующий запрашиваемому квадрослову бит достоверности, то принимается решение о попадании в кэш, и команда считывается из кэша. Если совпадения не было, принимается решение о промахе, и запрашиваемое квадрослово считывается из внешней памяти и загружается в кэш. Также в тэговую память записывается значение тэга, загруженного квадрослова, и устанавливается соответствующий бит достоверности. Новое значение будет записано в один из четырех входов (банков) тэговой памяти, в зависимости от выбранного алгоритма замещения тэгов в линии. В данном случае реализован самый простой вариант циклического замещения. Это значит, что кроме памяти тэгов имеется 128 счетчиков (каждый размером два бита), которые отслеживают номер банка для записи нового значения. После записи значение соответствующего счетчика увеличивается.

В кэше могут отображаться команды из трех внешних типов (банков) памяти. Каждый банк памяти условно разбивается на 32 страницы. Для динамической памяти размер страницы равен 512 К слов, для статической памяти – 128 К слов. Каждой странице может быть установлен атрибут кэшируемости, т.е. возможность загружать в кэш или нет.

Также для кэша предусмотрены два варианта загрузки в случае промаха:

– загрузка одного квадрослова;

– загрузка двух смежных квадрослов (отличающихся битом 2 адреса).

7.2 Кэш данных

Структура кэша данных (далее кэша) схожа со структурой кэша команд и имеет такой же размер накопителя – 8 К слов, 128 линий по четыре входа каждая. Но в отличие от кэша команд содержимое кэша данных может быть изменено. Для этого требуется введение дополнительных бит в поле тэга для отображения факта модификации. Для каждого квадрослова имеется специальный грязный бит (dirty bit) указывающий на то, что одно из слов или все слова квадрослова были изменены.

Кэш данных обычно поддерживает одну из двух стратегий записи:

– обратную запись (write-back) – данные в случае попадания в кэш записываются только в кэш;

– сквозную запись (write-throw) – данные пишутся и в кэш, и во внешнюю память.

Для сквозной записи «грязные биты» всегда равны нулю. В случае обратной записи возникает проблема, когда линия входа с установленными «грязными битами» должна быть замещена новыми данными. В этом случае старые данные линии должны быть откопированы обратно во внешнюю память (copy-back). Поле «грязных битов» позволяет копировать только модифицированные квадрослова.

Обычно стратегия обратной записи дает более высокую скорость обработки. Но в ряде случаев важно иметь достоверную копию данных и во внешней памяти (например, для видеобуфера или для контроллера DMA). В этом случае предпочтительнее

использовать стратегию сквозной записи. Для каждой страницы банка памяти может быть задан не только бит кэшируемости, но и бит сквозной записи. Таким образом, одни участки памяти могут поддерживать стратегию обратной записи, а другие – сквозной записи.

Кроме типов write-back и write-throw к кэшу данных применимы понятия:

– read-allocate – стратегия подразумевает действия кэша в случае промаха при чтении данных – в данном случае в кэш загружается линия данных;

– write-allocate – стратегия write-allocate подразумевает действия кэша в случае промаха при записи данных – в данном случае в кэш сначала загружается линия данных, а затем выполняется запись в линию.

В процессоре поддерживается только стратегия read-allocate. Стратегия write-allocate не поддерживается, и в случае промаха при записи, данные сразу же отправляются во внешнюю память (никаких загрузок в кэш не выполняется).

Аналогично кэшу команд для кэша данных может быть установлена стратегия подзагрузки сразу двух квадрослов в случае промаха.

Операции с кэш-памятью данных контролируются битами C (кэшируемые данные) и WT (сквозная запись). Биты определяются с помощью специальных регистров, описание которых приведено в подразделе 7.5 «Модуль управления защитой внутренней памяти и управления кэш-памятью». При обращении к кэшу возможно попадание в кэш (HIT) или промах (MISS). Попадание означает, что данные для заданного адреса находятся в кэше. Промах означает отсутствие данных с заданным адресом в кэше. Одна линия кэша одного из четырех входов кэш-памяти имеет атрибуты: 20 бит тэга, четыре бита достоверности (V бит), четыре бита модификации (Db бит). Каждому квадрослову соответствует свой бит достоверности и бит модификации. С кэшем данных кроме операций чтения и записи возможны операции очистки и сброса бит достоверности (см. подраздел 7.3 «Обеспечение когерентности кэшей и внешней памяти»).

Возможные ситуации работы с кэш-памятью приведены в таблице 30, где также показано состояние бит до и после операции.

Таблица 30 – Возможные ситуации работы с кэш-памятью

Операция	Status	V	Db	Тэг	Данные	Действия	V	Db	Тэг	Данные
Сброс процессора		x	x	x	x	F0	0	x	x	x
Чтение (C==1)	miss	0	x	x	x	F1	1	0	Ty	Ty
Чтение (C==1)	miss	1	0	Tx	Dx	F1	1	0	Ty	Dy
Чтение (C==1)	miss	1	1	Tx	Dx	F2 затем F1	1	0	Ty	Dy
Чтение	hit	1	x	Tx	Dx	F3	1	x	Tx	Dx

Операция	Status	V	Db	Тэг	Данные	Действия	V	Db	Тэг	Данные
Запись	miss	x	x	x	x	Нет операций	x	x	x	x
Запись (WT==0)	hit	1	x	Tx	Dx	F4	1	1	Tx	Dy
Запись (WT==1)	hit	1	x	Tx	Dx	F5	1	0	Tx	Dy
Очистка	miss	x	x	x	x	Нет операций	x	x	x	x
Очистка	hit	1	0	x	x	Нет операций	1	0	x	x
Очистка	hit	1	1	Tx	Dx	F2	1	0	Tx	Dx
Сброс линии	miss	x	x	x	x	Нет операций	x	x	x	x
Сброс линии	hit	1	x	x	x	F6	0	x	x	x

Примечание – Обозначения в таблице:

x, Tx, Dx – данные в кэше до и после выполнения операции;

Ty, Dy – данные в кэше после выполнения операции;

F0 – очистка всех бит достоверности;

F1 – загрузка блока (одного или двух квадрослов) данных в выбранную линию;

F2 – обратная запись (copy-back) активной линии в буфер записи. Из всех четырех квадрослов во внешнюю память передаются только те из них, для которых бит модификации установлен в «1»;

F3 – чтение данных активной линии в процессор;

F4 – запись данных в активную линию. Для соответствующего квадрослова устанавливается в «1» его бит модификации;

F5 – запись данных в активную линию. Для соответствующего квадрослова устанавливается в «0» его бит модификации. Записываемые данные также транслируются во внешнюю память;

F6 – очистка бита достоверности выбранной линии

Как упоминалось в описании кэша команд, для каждой линии имеется свой двухбитовый указатель на номер входа. После загрузки линии в текущий вход, значение указателя увеличивается на 1. Для случая чтения и состояния miss в левой части таблицы приводится информация о линии того входа, куда будет загружена новая информация.

7.3 Обеспечение когерентности кэшей и внешней памяти

При использовании кэш-памяти возникает проблема обеспечения когерентности данных в случае, когда к внешней памяти имеет доступ какое-то из дополнительных устройств кроме процессора. Таким устройством является контроллер прямого доступа к

памяти (контроллер DMA). Такая же проблема может возникать в случае модификации программного кода.

В процессоре реализованы команды управления, позволяющие поддержать программную реализацию обеспечения когерентности данных.

В процессоре реализованы следующие команды управления для кэша данных:

– 0000 – очистка линии с адресом A;

– 0001 – очистка линии по номеру линии и номеру входа;

– 0010 – очистка и сброс бит достоверности линии с адресом A;

– 0011 – очистка и сброс бит достоверности линии по номеру линии и номеру входа;

– 0100 – нет операции;

– 0101 – сброс бит достоверности одного входа линии по номеру линии и номеру входа;

– 0110 – сброс бит достоверности всех входов линии по номеру линии;

– 0111 – сброс бит достоверности всего кэша данных.

Таким образом, если необходимо копировать значение из кэша данных по конкретному адресу A, следует очистить биты 3:0 адреса, в поле 3:0 адреса записать номер команды (код смотри выше), а также записать полученный адрес в специальный регистр блока управления кэш-памятью IDC_CR.

Если такой адрес присутствует в кэше данных и имеет установленные «грязные биты», произойдет обратное копирование данных во внешнюю память. Одновременно с копированием можно сбросить биты достоверности.

В процессоре реализованы следующие команды управления для кэша команд:

– 1010 – сброс бит достоверности линии с адресом A;

– 1101 – сброс бит достоверности одного входа линии по номеру линии и номеру входа;

– 1110 – сброс бит достоверности всех входов линии по номеру линии;

– 1111 – сброс бит достоверности всего кэша команд.

Таким образом, если необходимо удалить команды из кэша команд по конкретному адресу A, следует очистить биты 3:0 адреса, в поле 3:0 адреса записать номер команды (код смотри выше), а также записать полученный адрес в специальный регистр IDC_CR. Если такой адрес присутствует в кэше команд, произойдет сброс бит достоверности. Регистр IDC_CR доступен по записи и чтению. При записи информация, записываемая в регистр, содержит код выполняемой операции, а также может содержать передаваемый параметр (адрес, индекс). При чтении регистра IDC_CR значение имеет только бит 0 регистра. Все другие биты при чтении имеют значение ноль. Нулевой разряд регистра выполняет функцию флага занятости регистра IDC_CR. После записи данный флаг устанавливается в «1» и удерживается равным 1 до тех пор, пока команда не будет передана в кэш данных или кэш команд на исполнение. Поэтому, перед записью в регистр IDC_CR может потребоваться предварительное чтение и анализ флага занятости.

Например, следующий код позволяет выполнить обратное копирование всех модифицированных линий кэша данных во внешнюю память:

```
        lc1 = 128*4;; // полное количество линий кэша данных
        j5 = 0x0001;; // команда обратного копирования и начальный индекс
rep_cln:
        IDC_CR = j5;; // обратное копирование, если линия модифицирована
        j5 = j5+0x10;;  // переход к следующей линии
wait_cln_fin:
        j0 = IDC_CR;; // чтение флага готовности
        j0 = j0 and 1;; // проверка флага
        if njeq, jump wait_cln_fin (NP);;  // ожидание, если флаг установлен
        if nlc1e, jump rep_cln;; // очистка следующей линии
```

Скорость выполнения данной процедуры будет зависеть в первую очередь от количества модифицированных линий в кэше данных. Если количество таких линий достаточно велико, поток данных из кэша во внешнюю память будет определяющим в скорости. И, например, если модифицированных линий нет вообще, время выполнения процедуры составит около 128 • 4 • 6 = 3072 такта.

С учетом аппаратных модификаций, возможна следующая процедура очистки:

```
lc1 = 128*4;; // полное количество линий кэша данных
j5 = 0x0001;;  // команда обратного копирования и начальный индекс
rep_cln:      IDC_CR = j5;; // обратное копирование, если линия модифицирована
j5 = j5+0x10;;      // переход к следующей линии
if nlc1e, jump rep_cln;; // очистка следующей линии
```

В этом случае минимальное время будет в два раза меньше. Данный алгоритм требует, чтобы между командами записи в IDC_CR было как минимум две линии команд. Однако, в случае, если количество модифицированных линий в кэше велико, оба алгоритма будут приблизительно равны по времени, т.к. в данном случае определяющим фактором будет скорость обмена с внешней памятью.

7.4 Эффективность кэша команд и кэша данных

При использовании кэша команд команды поступают в буфер со скоростью половины частоты процессора. Одновременно могут считываться четыре команды, что обеспечивает достаточный темп в случае, когда за один такт выполняется меньше четырех команд одной линии. Если в каждом такте выполняются все четыре команды одной линии, и нет задержек на конвейере команд, скорость составит только половину от максимальной. В случае ветвлений кэш команд будет менее эффективен, чем внутренняя память. При ветвлении из-за большей длины конвейера (5 вместо 3) формируется дополнительная задержка на загрузку первой команды перехода. Если переходы очень часты или тело циклов очень короткое, могут возникнуть дополнительные потери. В случае работы с внешней памятью не используется буфер предсказания переходов, что снижает быстродействие.

Для кэша данных запись данных при попадании в кэш или при промахе не вносит дополнительных задержек. При чтении данных из кэша будет дополнительная задержка. В случае чтения из внутренней памяти (стадии конвейера A, M, W) данные готовы на стадии W и могут поступать в регистр. В случае работы с внешней памятью и в случае попадания в кэш, на стадии W обнаруживается попадание в кэш, и формируется запрос на чтение кэша. В следующем такте происходит чтение данных из кэша в буфер (конвейер данных процессора остановлен). В следующем такте данные из буфера записываются в регистр приемника (конвейер остановлен). После записи конвейер может продолжить движение. Таким образом, при чтении данных образуются два дополнительных такта остановки конвейера для случая, когда нет зависимости по данным, т.е. считываемые данные не требуются в следующем такте для обработки. Если такая зависимость есть, образуется три дополнительных такта по сравнению с внутренней памятью.

Ассоциативность кэшей равна четырем. Это означает, что данные могут храниться по четырем адресам с различными тэгами, но одинаковыми индексами. Например, можно максимально эффективно одновременно работать с четырьмя буферами данных во внешней памяти, каждый объемом 2К слов, размещенных по адресам 0x4000_0000, 0x4000_0800, 0x4000_1000, 0x4000_1800. Однако, если потребуется пятый буфер одновременно с четырьмя предыдущими, эффективность использования кэша будет очень низкой, т.к. буферы будут по очереди вытеснять друг друга из кэша.

Кэш эффективен в случае, когда необходимо повторное использование данных. Также он может быть полезен при однократном использовании данных. Например, при чтении слова загружается квадрослово. Это значит, что оставшиеся три слова могут быть считаны из кэша. При установке бита подзагрузки сразу двух квадрослов, эффективность увеличивается.

7.5 Модуль управления защитой внутренней памяти и управления кэш-памятью

В процессор добавлен модуль, который содержит набор регистров для управления кэш-памятью, а также управления защитой внутренней памяти. Данный модуль использует регистры групп 0x1E и 0x1F. Набор регистров приведен в таблице 31. Доступ к регистрам устройства защиты памяти выполняется с помощью обычных команд пересылки IALU.

Внимание! Доступ по записи возможен только в режиме супервизора и при установленном в «1» бите EXT_MODE регистра SQSTAT.

Таблица 31 – Регистры модуля защиты памяти

Номер	Название	Назначение
Группа 0x1E		
0x03C0	MS0_C	Регистр управления кэшированием данных MS0
0x03C1	MS0_WT	Регистр управления сквозной записью MS0
0x03C2	MS0_CI	Регистр управления кэшированием команд MS0
0x03C3 – 0x03C7	-	
0x03C8	MS1_C	Регистр управления кэшированием данных MS1

Номер	Название	Назначение
0x03C9	MS1_WT	Регистр управления сквозной записью MS1
0x03CA	MS1_CI	Регистр управления кэшированием команд MS1
0x03CB – 0x03CF	-	
0x03D0	SDR_C	Регистр управления кэшированием данных SDRAM
0x03D1	SDR_WT	Регистр управления сквозной записью SDRAM
0x03D2	SDR_CI	Регистр управления кэшированием команд SDRAM
0x03D3 – 0x03DF	-	
Группа 0x1F		
0x03E0	PU0	Регистр конфигурации защиты модуля памяти 0
0x03E1	PU1	Регистр конфигурации защиты модуля памяти 1
0x 03E2	PU2	Регистр конфигурации защиты модуля памяти 2
0x03E3	PU3	Регистр конфигурации защиты модуля памяти 3
0x03E4	PU4	Регистр конфигурации защиты модуля памяти 4
0x03E5	PU5	Регистр конфигурации защиты модуля памяти 5
0x03E6	PU6	Регистр конфигурации защиты модуля памяти 6
0x03E7	PU7	Регистр конфигурации защиты модуля памяти 7
0x03E8	PU_SR	Регистр состояния
0x03E9 – 0x03FB	-	
0x03FC	PU_CR	Регистр управления
0x03FD	IDC_CR	Регистр управления операциями с кэш памятью
0x03FE – 0x03FF	-	

7.5.1 Регистр защиты памяти PU

Регистр управляет защитой одного модуля внутренней памяти. Размер модуля и параметры защиты задаются программно. При программировании регистра размер защищаемого модуля памяти задается количеством страниц размером 1 К слов. Назначение разрядов регистра приведено в таблице 32.

Таблица 32 – Регистр защиты PU

Номер бита	Название	Назначение
10-0	STA	Начальный адрес модуля. Соответствует физическому адресу памяти образуемому как {11'b0,START_A[10:0],10'b0}.
11	-	
22-12	ENDA	Конечный адрес модуля. Соответствует физическому адресу памяти образуемому как {11'b0,END_A[10:0],10'b11_1111_1111}
23	-	
25-24	JK_AP	Режим доступа к модулю со стороны шин J и K: 00 – полный доступ; 01 – супервизор чтение и запись, пользователь чтение; 10 – всем только чтение; 11 – супервизор только чтение

Номер бита	Название	Назначение
27-26	I_AP	Режим доступа к модулю со стороны шины I: 00 – полный доступ; 01 – только супервизор; 10 – доступ запрещен; 11 – доступ запрещен
29-28	H_AP	Режим доступа к модулю со стороны шины S (хост, DMA): 00 – чтение и запись; 01 – только чтение; 10 – доступ запрещен; 11 – доступ запрещен
31-30	-	

Поля STA и ENDA используются для сравнения со значениями адресных шин J, K, I, S. В сравнении участвуют только биты 20:10 указанных шин. Проверка выполняется только при доступе к внутренней памяти. Так для K-шины попадание в некоторый модуль означает истинность следующего выражения:

Hit_PUx = (K[20:10]>=STAx) && (K[20:10]<=ENDAx).

Минимальный размер модуля равен странице из 1 К слов. Модуль может рассматриваться как множество из 1 К страниц. При нарушении прав доступа к модулю памяти вырабатывается исключительная ситуация. Исключение составляет контроль доступа со стороны шины S. Нарушение прав доступа будет вызывать блокировку записи и установку флага ошибки. Исключительная ситуация вырабатываться не будет. Исключительная ситуация всегда форсируется после выполнения команды. Поэтому в случае срабатывания защиты по доступу со стороны шины I, процессор выполнит одну линию команд из защищенной области, прежде чем перейдет на обработку исключительной ситуации. Определить событие, вызвавшее срабатывание исключительной ситуации, можно анализируя регистр состояния устройства управления, а также регистр состояния модуля защиты.

7.5.2 Регистр состояния PU_SR

Регистр состояния хранит информацию о событии, которое вызвало срабатывание модуля защиты. После сброса значение регистра равно нулю. Назначение разрядов регистра приведено в таблице 33.

Таблица 33 – Регистр управления PU_SR

Номер бита	Название	Назначение
0	JPF	Срабатывание защиты при доступе со стороны шины J
1	KPF	Срабатывание защиты при доступе со стороны шины K
2	IPF	Срабатывание защиты при доступе со стороны шины I
3	HPF	Срабатывание защиты при доступе со стороны шины S (хост, DMA)
4-31	-	Всегда равны нулю

Если какой-то флаг в регистре состояния установлен, значит при доступе со стороны соответствующей шины произошло срабатывание модуля защиты, что вызвало генерацию исключительной ситуации. При возникновении исключительной ситуации из регистра состояния SQSTAT можно узнать, что ситуация была вызвана срабатыванием защиты по правам доступа. При чтении регистра PU_SR можно определить, была ли данная ситуация вызвана срабатыванием модуля защиты. Если регистр PU_SR равен нулю, значит сработала защита в модулях J или K целочисленного АЛУ. Регистр PU_SR автоматически сбрасывается после выполнения чтения.

7.5.3 Регистр управления PU_CR

Регистр управляет включением различных функций процессора. Назначение разрядов регистра приведено в таблице 34.

Таблица 34 – Регистр управления PU_CR

Номер бита	Название	Назначение
0	DC_ON	1 – включение кэш памяти данных
1	IC_ON	1 – включение кэш памяти команд
2	EN_2dQW	1 – грузить два квадрослова данных при промахе; 0 – загрузка одного квадрослова
3	EN_2iQW	1 – грузить два квадрослова команд при промахе; 0 – загрузка одного квадрослова
4	SOC_speed_up	0 – разрешает выполнение команды на стадии W одновременно с загрузкой данных с SOC-шины; 1 – запрещает выполнение команды на стадии W одновременно с загрузкой данных с SOC-шины. Сначала загружаются данные, затем начинает движение конвейер
5-7	-	Зарезервированы
8	PSD_fun	Управление генерацией исключительной ситуации при загрузке регистров устройства управления и модуля отладки из внешней памяти: 0 – разрешена генерация; 1 – исключительная ситуация только при нарушении прав доступа
9	SP_byte	Управление автоматическим сдвигом указателя стека и фрэйма при работе с байтами и короткими: 0 – функция выключена; 1 – функция включена
10	MIN_MAX	Включение функции поиска номера максимума-минимума: 0 – выключено; 1 – включено

Номер бита	Название	Назначение
11	LC_sup	Выключение анализа выхода из цикла: 0 – анализ работает; 1 – анализ выключен. 0 – разрешение использования процедуры анализа завершения цикла. Позволяет сократить время выхода из цикла; 1 – при реализации циклов с помощью LC0, LC1 будут потери в скорости при выполнении последнего предсказания
12-15		Зарезервированы
16	PU0_EN	Разрешение работы модуля защиты 0: 1 – разрешено; 0 – запрещено
17	PU1_EN	Разрешение работы модуля защиты 1: 1 – разрешено; 0 – запрещено
18	PU2_EN	Разрешение работы модуля защиты 2: 1 – разрешено; 0 – запрещено
19	PU3_EN	Разрешение работы модуля защиты 3: 1 – разрешено; 0 – запрещено
20	PU4_EN	Разрешение работы модуля защиты 4: 1 – разрешено; 0 – запрещено
21	PU5_EN	Разрешение работы модуля защиты 5: 1 – разрешено; 0 – запрещено
22	PU6_EN	Разрешение работы модуля защиты 6: 1 – разрешено; 0 – запрещено
23	PU7_EN	Разрешение работы модуля защиты 7: 1 – разрешено; 0 – запрещено
24-29	-	Зарезервированы
31-30	USOC_EN	Режим доступа к периферийным устройствам SOC-шины (адреса 0x8000000 и выше) в режиме пользователя: 00 – запрещен любой доступ; 01 – запрещена запись, но разрешено чтение; 1x – разрешен любой доступ; По сбросу любой доступ запрещен

Бит SOC_speed_up позволяет ускорить процесс загрузки данных с SOC-шины или внешней памяти. Команда чтения регистра SOC-шины тратит ориентировочно 10 тактов ядра на загрузку данных. В это время процессор простаивает. Бит SOC_speed_up

позволяет управлять операциями в 10-ом такте загрузки, т.е. в момент, когда данные грузятся непосредственно в регистр процессора:

– если бит равен нулю, возможно выполнение операций процессором одновременно с загрузкой, если нет конфликта по записи;

– если бит равен единице, процессор продолжит работу только после записи.

В результате процессор может простаивать девять тактов вместо десяти. При работе с кэшем данных использование данного бита позволяет сократить число тактов останова конвейера процессора при чтении данных из кэша: вместо двух дополнительных тактов ожидания, будет только один такт. Загрузка данных из кэша в регистр-приемник будет происходить одновременно с выполнением команд следующей линии.

В процессоре доступ к устройствам SOC-шины возможен также с помощью обычных команд загрузки и сохранения регистров по определенному адресу. Адресное пространство периферийных устройств от 0x8000_0000 до 0xFFFF_FFFF. Доступ ко всем периферийным устройствам возможен также в режиме пользователя, т.к. устройство защиты памяти защищает только внутреннюю оперативную память. Наибольшую опасность в данном случае представляет некорректное поведение пользовательской программы, способной привести к потере работоспособности всей системы.

В связи с этим, в модуле защиты в регистре управления выделены специальные биты (USOC_EN ==PU_CR[31:30]), которые запрещают доступ по записи и по чтению к адресному пространству периферийных устройств в режиме пользователя, независимо от того, разрешены программные исключения или нет. Коды: 00 – запрет доступа, 01 - чтение разрешено, 1x – полный доступ.

После сброса значение регистра PU_CR равно нулю.

7.5.4 Регистры управления кэшированием данных MS0_C, MS1_C, SDR_C

Регистры используются для задания атрибута кэшируемости (C бит) для отдельных страниц банков внешней памяти. Процессор может адресовать три банка внешней памяти: MS0, MS1, SDRAM (четыре банка). При этом модули MS0 и MS1 имеют деление на страницы по 128 К слов, а модуль SDRAM (банк 0) имеет деление на страницы по 512 К слов. Каждый бит регистра соответствует странице памяти. Так нулевой бит соответствует нулевой странице, первый бит – первой и т.д. Итого можно определить до 32 • 128 = 4 М слов памяти модулей MS0, MS1 и до 16 М слов динамической памяти. Для динамической памяти регистр управляет только банком 0. Другие три области динамической памяти всегда считаются кэшируемыми и используют стратегию обратной записи. Таким образом для памяти MS0, MS1 индекс страницы берется из бит 21:17 адреса слова памяти. Для динамической памяти индекс страницы – из бит 23:19 адреса слова. Если адрес динамической памяти превышает размер 15 М слов, это всегда соответствует кэшируемому адресу.

7.5.5 Регистр управления сквозной записью MS0_WT, MS1_WT, SDR_WT

Аналогично управлению кэшируемостью страниц существует возможность для каждой страницы указать бит стратегии записи (WT бит) при попадании в кэш:

0 – запись только в кэш;

1 – запись в кэш и во внешнюю память.

7.5.6 Регистр управления кэшированием команд MS0_CI, MS1_CI, SDR_CI

Аналогично управлению кэшируемостью страниц для данных существует возможность для каждой страницы указать бит кэшируемости при обращении к командам:

0 – не кэшируется;

1 – кэшируется.

7.6 Кэш память и операции с байтами и короткими словами во внешней памяти

Процессор поддерживает на аппаратном уровне работу с байтами и короткими словами во внешней памяти. Как и в случае с внутренней памятью, адрес байта или короткого слова перед обращением к памяти предварительно преобразуется в адрес слова, и затем выполняется операция. Следовательно, кэш данных всегда будет хранить в тэге адрес слова данных.

Рассмотрим случай, когда в регистре j2 хранится значение 0x4000_4000. В случае байтового доступа с использованием регистра j2 кэш данных и внешняя память будут работать с адресом 0x1000_1000. При обращении к короткому слову рабочий адрес будет равен 0x2000_2000. Для кэша данных и для внешней памяти это будут три разных адреса. Для соответствия между адресами слов, байт и коротких слов необходимо строго придерживаться правил формирования адресов и правил преобразования адресов при операциях с указателями. Если какой-то указатель на слово имеет значение A, то соответствующий данному слову указатель на короткое слово должен иметь значение A*2, а соответствующий указатель на байт значение A*4. Для области MS0, MS1 это означает, что адреса коротких слов будут начинаться с 0x6vvv_vvvv и 0x7vvv_vvvv. Для адресов динамической памяти возникает сложность, т.к. для адреса слова 0x4vvv_vvvv адрес байта выходит за границы 32-разрядного формата. Поэтому динамическая память имеет дополнительное адресное пространство 0x1vvv_vvvv, которое соответствует стандартному адресному пространству. При его использовании (в регистре SYSCON должен быть установлен соответствующий бит) адреса коротких слов будут равны 0x2vvv_vvvv, а адреса байт 0x4vvv_vvvv. При операциях с указателями разных типов должен выполняться соответствующий сдвиг. При обращении к словам во внешней памяти адрес 0x4000_0000 соответствует той же ячейке памяти, что и адрес 0x1000_0000. Однако в кэше данных эти адреса будут разными.

7.7 Алгоритм очистки кэша команд

Для обеспечения когерентности кэша команд возможно использование специальных команд очистки линий кэша команд по значению адреса или по значению индекса. При этом необходимо соблюдать ряд требований к потоку команд на конвейере процессора.

Далее приведен пример кода для удаления блока инструкций из кэша команд. В таких случаях обычно задается стартовый адрес блока и конечный адрес блока. В начале текущий адрес блока равен стартовому. После очистки к текущему адресу прибавляется значение 16 и проверяется выход за пределы блока. Если завершения блока нет, процесс повторяется. Код программы следующий:

#1: IDC_CR = текущий_адрес_очистки; увеличение адреса очистки на 16;

#2: проверка конца блока (в JALU);;

#3: команда перехода на команду #1 если конца нет;; (NP)

Таблица 35 – Команда очистки кэша с использованием адреса

Такт	IA	IM	IW	P	D	X	A	M	W
1					#1				
2					*#2*	#1			
3					*#3*	*#2*	#1		
4	Очистка конвейера команд					*#3*	*#2*	#1	
5	#1							*#2*	#1
6	#x1	#1							*#2*
7		#x1	#1						
8			#x1	#1					
9				#x1	#1				
10					*#2*	#1			
11					*#3*	*#2*	#1		
12	Очистка конвейера команд					*#3*	*#2*	#1	
13	#1							*#2*	#1
14	#x1								*#2*

Программа очистки кэша исполняется из внутренней памяти и не использует кэш команд. Из временной диаграммы таблицы 35 видно, что команда #1 на стадии W запускает дополнительный процесс #x1, который требует четыре последующих такта:

Такт 6 – передача адреса в кэш команд;

Такт 7 – чтение линии памяти тэгов;

Такт 8 – проверка совпадения тэгов с адресом;

Такт 9 – в случае совпадения, запрос к памяти тегов на очистку бита достоверности.

Имеются ограничения на параллельные действия со стороны процессора. В момент, когда специальная команда #x1 читает память тегов (такт 7) на конвейере процессора не должен происходить переход. Иначе переход сбрасывает содержимое конвейера выборки команд, а заодно и команду #x1 на стадии IM.

Если в момент запроса на очистку бита достоверности (такт 9) имеется запрос к памяти тегов со стороны следующей специальной команды, то следующая специальная команда будет приостановлена. Этот факт нужно учитывать при записи значения в регистр IDC_CR – слишком плотный поток команд может вызвать пропуск отдельных команд. По сути специальная команда требует для своего исполнения один в случае промаха или два такта в случае попадания. В связи с этим для команды перехода цикла рекомендуется использовать опцию (NP).

8 Встроенный интерфейс (SOC-интерфейс)

При работе с внутренней памятью ядро процессора использует три внутренние шины I, J, K. Однако для работы с периферийными устройствами процессор использует специальный встроенный интерфейс, который осуществляет передачу запроса от внутренних шин ядра на специальную шину периферийных устройств (SOC – шина). Данная шина соединяет периферийные модули (внешний порт, DMA, порты связи, порт JTAG, контроллер прерываний и др.) с системой памяти через SOC-интерфейс и шину S типа. SOC-шина имеет шину данных шириной 128 бит и адресную шину шириной 32 бита. Она работает с частотой равной половине частоты ядра. Все данные, передаваемые между внутренней памятью или ядром и периферийными модулями, проходят через SOC-интерфейс и SOC-шину.

SOC-интерфейс представляет собой модуль процессора, выполняющий связующую роль между ядром и внутренними периферийными устройствами. Со стороны ядра к интерфейсу подключены четыре шины J-, K-, I- и S-типов, а с другой стороны одна SOC-шина периферийных устройств. SOC-шина имеет пропускную способность 128 бит, и ее тактовый генератор (SOCCLK) работает на частоте, равной половине частоты ядра (CCLK). Периферийными устройствами, которые могут быть ведущими на SOC-шине являются (рисунок 16):

– SOC-интерфейс;
– контроллер DMA.

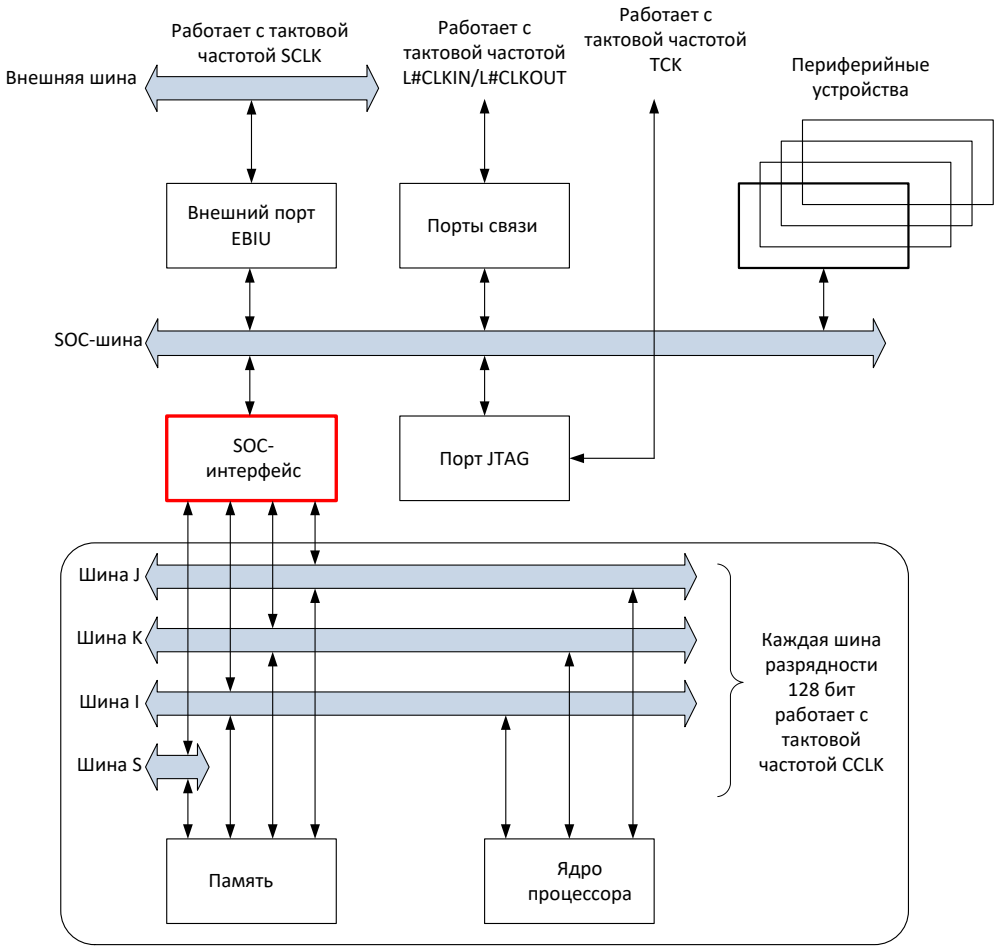


Рисунок 16 – Подключение периферийных модулей

Все внутренние периферийные устройства подключены к SOC-шине и являются ведомыми устройствами.

Структура SOC-интерфейса показана на рисунке 17, а архитектура шин, соединяющих различные модули SOC-шины, приведена на рисунке 18.

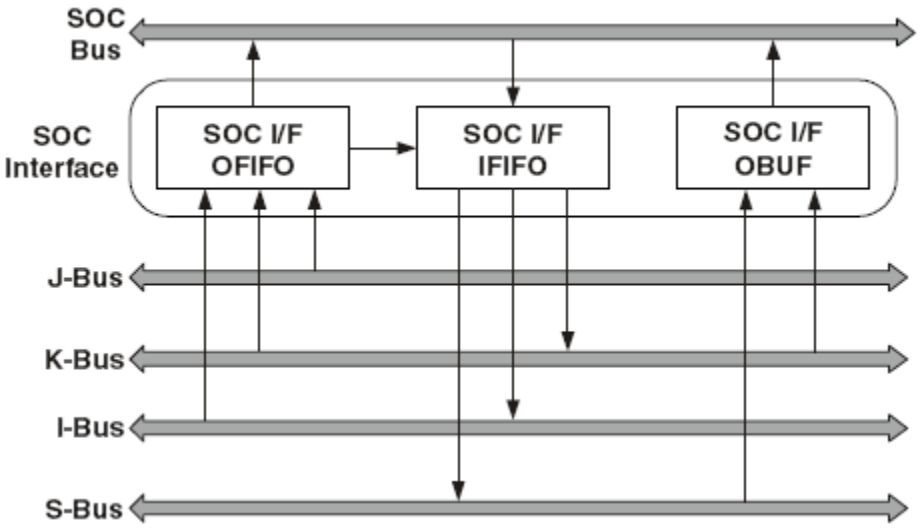


Рисунок 17 – Поток данных через SOC-интерфейс

В связи с тем, что на SOC-шине может быть несколько ведущих устройств, при каждом доступе к шине выполняется арбитраж. При этом доступ к периферийным устройствам возможен со стороны процессора и со стороны контроллера DMA. Со стороны процессора представителем является SOC-интерфейс и при этом он имеет два источника запросов: выходное FIFO интерфейса (OFIFO) и выходной буфер (OBUF). Приоритеты доступа при арбитраже фиксированы. Наивысший приоритет имеет выходной буфер интерфейса OBUF, который может возвращать прочитанное значение в контроллер DMA или в порт связи.

SOC-интерфейс состоит из трех устройств FIFO, через которые проходят все потоки данных. Если процессор обращается во внешнюю память, запрос из SOC-интерфейса поступает сразу во внешний интерфейс. Если шины процессорного ядра посылают в SOC-интерфейс запрос на чтение, прочитанные данные должны вернуться обратно в интерфейс. Для возвращаемых данных используется входное IFIFO. Также в данный буфер помещаются запросы всех внешних ведущих устройств, которые обращаются к внутренней памяти ядра. Если модуль IFIFO получил прочитанные данные, он пересылает их во внутренний регистр ядра. Если же был получен запрос на запись со стороны ведущего устройства, записываемые данные пересылаются по адресу-приемнику (используется шина S). В случае, если запрос на чтение получен от внешнего ведущего устройства, выполняется чтение внутренней памяти (используется шина S) и прочитанные данные помещаются в выходной буфер OBUF. Таким образом, в OBUF попадают только считываемые внешними ведущими устройствами данные ядра.

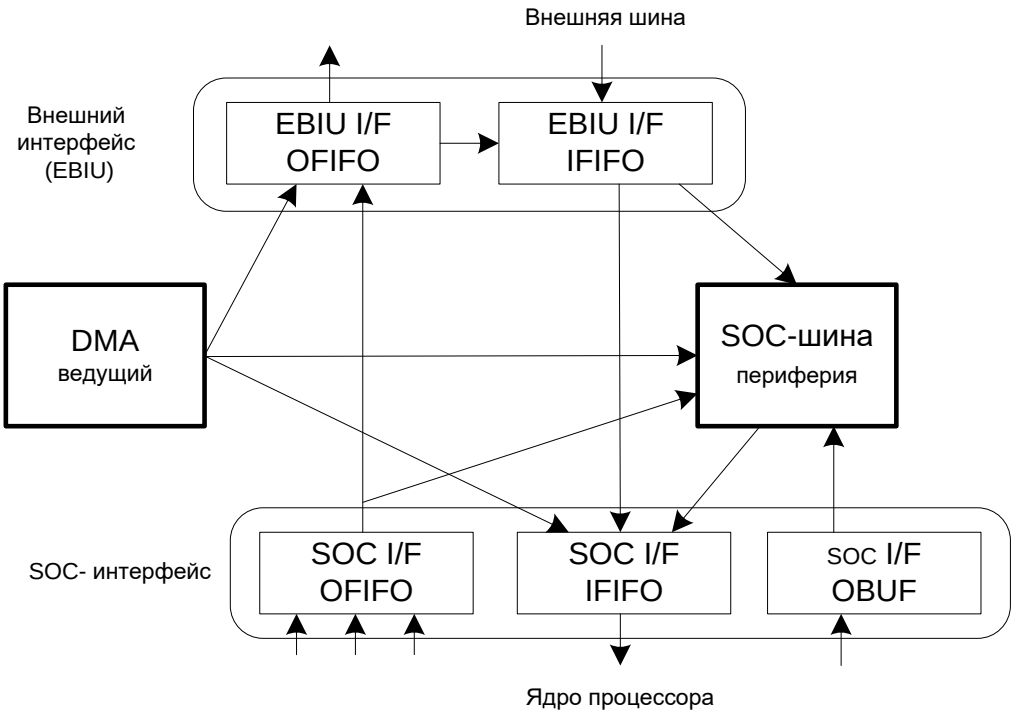


Рисунок 18 – Структура соединений модулей SOC-шины

8.1 Транзакции SOC OFIFO

Все запросы ядра по шинам J, K, I, которые не попадают во внутреннюю память, помещаются в выходное OFIFO. Это могут быть следующие запросы:

– Запись из ядра процессора по шинам J и K в устройства SOC-шины или внешнее адресное пространство;

– Запросы чтения по шинам J и K из устройств SOC-шины или внешнего адресного пространства;

– Запрос чтения команды по шине I из внешней памяти или из регистра инструкций эмулятора (EMUIR).

Все запросы ядра поступают в OFIFO с частотой ядра (CCLK). Считывание запросов из буфера выполняется с частотой SOC-шины (SOCCLK). На выходе буфера осуществляется проверка, к какому ресурсу выполняется запрос (внешняя память или внутренняя периферия), и далее запрос пересылается в пункт назначения. Если выполняется запрос к периферии, он имеет наименьший приоритет в арбитраже доступа. Если выполняется запрос к внешней памяти, также происходит арбитраж доступа к выходному FIFO внешнего интерфейса. За доступ к внешнему ресурсу OFIFO соревнуется с контроллером DMA (рисунок 18). SOC-интерфейс в этом арбитраже всегда имеет наивысший приоритет.

С точки зрения конвейера процессора команда завершается, когда запрос на запись или на чтение помещается в SOC OFIFO. Если это операция записи, она считается выполненной и конвейер ядра продолжает движение, а если операция чтения, конвейер останавливается до момента возвращения данных. При этом неважно имеется зависимость по данным или нет. При чтении команд из внешней памяти после помещения двух запросов в OFIFO конвейер команд останавливается до момента получения

прочитанной команды. При этом конвейер обработки команд может продолжать выбирать команды из буфера выравнивания и отправлять их на исполнение.

Буфер OFIFO имеет размер 8 записей. Это означает, что он может принять до 8-ми запросов без остановки конвейера процессора. При этом имеется следующие особенности:

– Если по шинам J и K одновременно выполняются две записи, процессору потребуется один такт, чтобы поместить их в OFIFO. При этом одна транзакция (J) помещается сразу в OFIFO, а вторая (K) во временный буфер. Из временного буфера запрос пересылается в OFIFO в следующем такте.

– Если процессору не требуется доступ к OFIFO в следующем такте, задержек конвейера не произойдет.

Запросы в OFIFO помещаются с частотой ядра, а считываются с частотой SOC-шины, т.е. в два раза меньшей. Поэтому в случае интенсивного потока записи во внешнюю память возможно заполнение буфера и это вызовет приостановку конвейера процессора.

8.2 Транзакции SOC IFIFO

Входной буфер IFIFO SOC-интерфейса обрабатывает два типа транзакций:

– Запросы чтения ядра процессора. SOC IFIFO принимает прочитанные данные, полученные в результате транзакций чтения, которые были переданы через OFIFO;

– Запросы чтения или записи внутренней памяти ядра от DMA.

В первом случае возвращаемые данные записываются в регистр ядра, используя шину K. Во втором случае IFIFO использует шину S для доступа к внутренней памяти. В случае операции чтения внутреннего ресурса, прочитанные данные помещаются в выходной буфер OBUF.

Имеется три источника запросов к IFIFO (рисунок 18):

– DMA;

– внешний интерфейс;

– SOC-шина.

Наивысший приоритет имеет SOC-шина, т.к. ее запрос означает чтение регистра периферии по запросу ядра, далее идет запрос внешнего интерфейса и наименьший приоритет у контроллера DMA. Запрос от внешнего интерфейса означает возвращение данных в ответ на запрос чтения внешней памяти ядром. Приоритеты доступа расставлены так, чтобы запросы ядра получали наивысший приоритет. В случае чтения это минимизирует простои процессора.

8.3 Транзакции SOC OBUF

Выходной буфер OBUF SOC-интерфейса задействуется только в операциях чтения внутренних ресурсов (памяти) контроллером DMA. Данные из OBUF могут пересылаться в DMA или порты связи.

8.4 Программирование интерфейса SOC

Чтение процессором внешней памяти вызывает передачу запроса от ядра через несколько доменов синхронизации. Все это может приводить к длительным остановам процессора. Для ускорения работы процессора с внешней памятью используется кэш-память данных и кэш-память команд. Процессор может организовывать работу с внешней памятью, как с использованием кэш-памяти, так и без.

Архитектурно SOC-шина представляет собой простой интерфейс к регистрам периферийных устройств, в котором циклы чтения или записи выполняются в течение одного такта. Если какое-либо из устройств не готово выдать (принять) данные в течение одного такта, оно может выставлять сигнал ожидания завершения операции.

9 Таймеры с функцией Захвата/ШИМ

В микросхеме реализовано два блока таймеров общего назначения, каждый из которых может быть использован для широкого спектра применений, включая:

– подсчет циклов частоты TIM_CLK или каких-либо внешних событий;

– формирование прерываний и запросов DMA по заданным событиям;

– захват входных сигналов, в том числе измерение длительности импульсов входных сигналов;

– генерацию различных форм выходных сигналов.

Основу таймеров составляет 32-битный перезагружаемый счетчик. Счет может быть прямой, обратный или двунаправленный. В качестве источника синхросигнала может выступать внутренняя тактовая частота TIM_CLK, внешние сигналы или другие таймеры.

В каждый блок таймера входит четыре канала, которые имеют в своем составе схему захвата и блок ШИМ с функциями формирования «мертвой зоны» и аппаратной блокировки.

Каждый из таймеров позволяет генерировать прерывания и запросы DMA.

9.1 Основные характеристики

Основные характеристики блока таймера:

– 32-битный счетчик: счет прямой, обратный или двунаправленный;

– 32-разрядный предварительный делитель частоты TIM_CLK;

– схема выбора источника тактирования основного счетчика от внешних сигналов или от других таймеров;

– четыре независимых канала, каждый канал может работать в одном из режимов:

– режим захвата: позволяет захватить (сохранить) текущее значение счетчика при изменении некоторого входного сигнала;

– режим ШИМ: позволяет осуществлять непрерывное сравнение заданных значений со значением счетчика для формирования выходных сигналов;

– формирование выходных сигналов в режиме ШИМ:

– сброс в НИЗКИЙ уровень при совпадении;

– установка в ВЫСОКИЙ уровень при совпадении;

– переключение (инвертирование) при совпадении;

– переключение при некотором условии;

– формирование прерываний и запросов DMA по событиям:

– обновление счетчика;

– захват;

– сравнение;

– внешние события по входам ETR и BRK.

9.2 Структурная схема

Структурная схема блока «Таймер» представлена на рисунке 19.

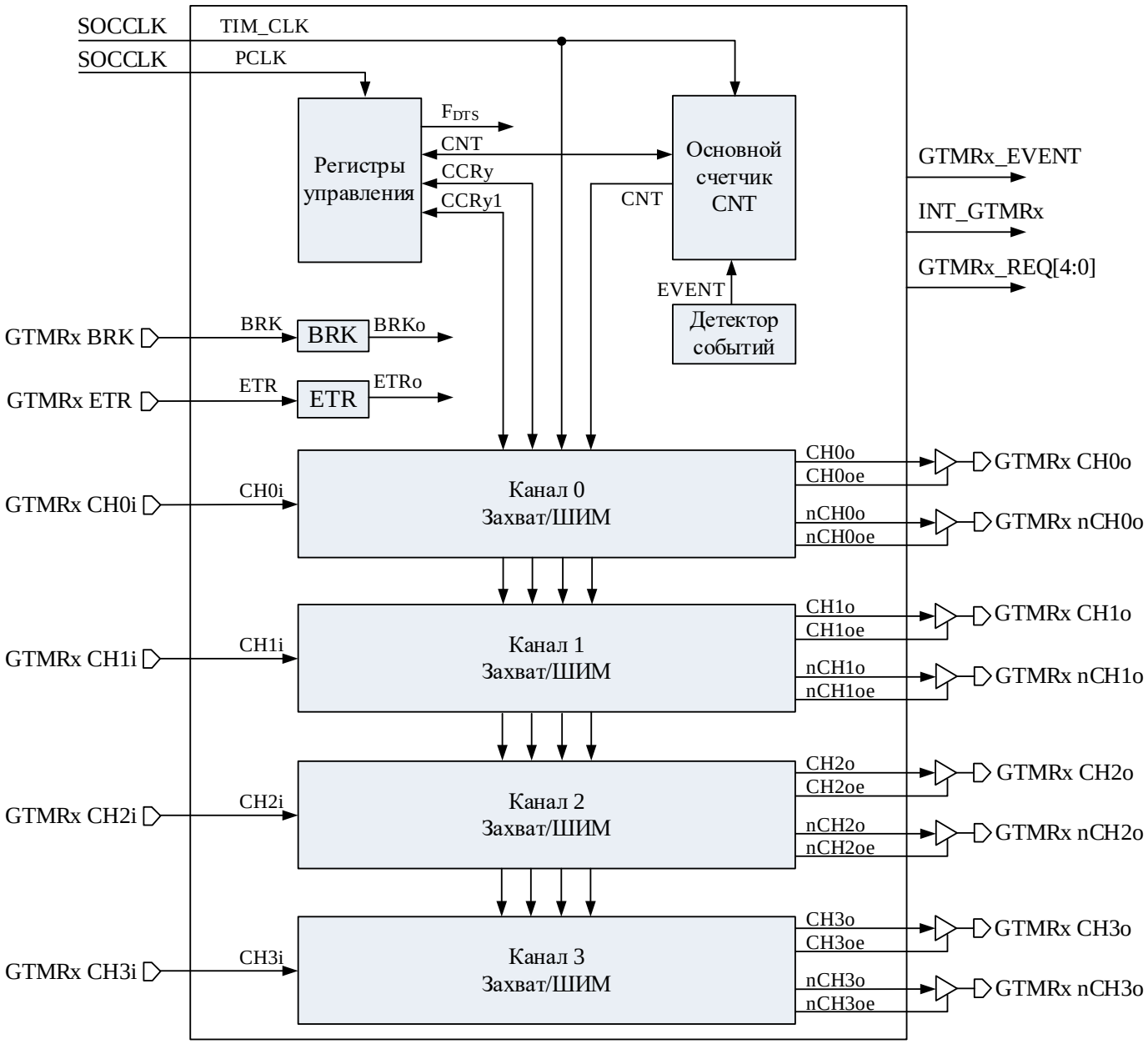


Рисунок 19 – Структурная схема блока «Таймер»

Таймер содержит основной 32-битный счетчик CNT, блок регистров управления и четыре канала схем захвата/ШИМ.

Таймер позволяет работать в режимах:

- таймер;
- расширенный таймер, с объединением нескольких таймеров;
- схема захвата;
- схема ШИМ.

9.2.1 Внешние выводы блоков таймеров

Внешние выводы блоков таймеров приведены в таблице 36.

Таблица 36 – Внешние выводы блоков таймеров

Обозначение вывода (основное)	Обозначение вывода блоков таймеров	Тип вывода	Функциональное назначение
PA[22]	GTMR0 CH0o	I/O	Таймер 0. Выход 0 ШИМ (+)
PA[23]	GTMR0 nCH0o	I/O	Таймер 0. Выход 0 ШИМ (-)
PA[24]	GTMR0 CH1o	I/O	Таймер 0. Выход 1 ШИМ (+)
PA[25]	GTMR0 nCH1o	I/O	Таймер 0. Выход 1 ШИМ (-)
PA[26]	GTMR0 CH2o	I/O	Таймер 0. Выход 2 ШИМ (+)
PA[27]	GTMR0 nCH2o	I/O	Таймер 0. Выход 2 ШИМ (-)
PA[28]	GTMR0 CH3o	I/O	Таймер 0. Выход 3 ШИМ (+)
PA[29]	GTMR0 nCH3o	I/O	Таймер 0. Выход 3 ШИМ (-)
PA[30]	GTMR0 BRK	I/O	Таймер 0. Блокировка выходов
PA[31]	GTMR0 ETR	I/O	Таймер 0. Универсальный регистратор событий
PB[0]	GTMR0 CH0i	I/O	Таймер 0. Вход 0 захвата
PB[1]	GTMR0 CH1i	I/O	Таймер 0. Вход 1 захвата
PB[2]	GTMR0 CH2i	I/O	Таймер 0. Вход 2 захвата
PB[3]	GTMR0 CH3i	I/O	Таймер 0. Вход 3 захвата
PB[8]	GTMR1 CH0o	I/O	Таймер 1. Выход 0 ШИМ (+)
PB[9]	GTMR1 nCH0o	I/O	Таймер 1. Выход 0 ШИМ (-)
PB[10]	GTMR1 CH1o	I/O	Таймер 1. Выход 1 ШИМ (+)
PB[11]	GTMR1 nCH1o	I/O	Таймер 1. Выход 1 ШИМ (-)
PB[12]	GTMR1 CH2o	I/O	Таймер 1. Выход 2 ШИМ (+)
PB[13]	GTMR1 nCH2o	I/O	Таймер 1. Выход 2 ШИМ (-)
PB[14]	GTMR1 CH3o	I/O	Таймер 1. Выход 3 ШИМ (+)
PB[15]	GTMR1 nCH3o	I/O	Таймер 1. Выход 3 ШИМ (-)
PB[16]	GTMR1 BRK	I/O	Таймер 1. Блокировка выходов
PB[17]	GTMR1 ETR	I/O	Таймер 1. Универсальный регистратор событий
PB[18]	GTMR1 CH0i	I/O	Таймер 1. Вход 0 захвата
PB[19]	GTMR1 CH1i	I/O	Таймер 1. Вход 1 захвата
PB[20]	GTMR1 CH2i	I/O	Таймер 1. Вход 2 захвата
PB[21]	GTMR1 CH3i	I/O	Таймер 1. Вход 3 захвата

9.3 Базовый блок таймера

Таймер построен на базе 32-битного счетчика. Базовый блок таймера включает в себя:

- основной счетчик таймера (CNT);
- основание счета (максимальное значение) основного счетчика (ARR);

– делитель частоты TIM_CLK (PSG), используемый для тактирования основного счетчика;

– регистр управления основным счетчиком (CNTRL).

Сигналом для изменения основного счетчика CNT может служить как внутренняя частота TIM_CLK, так и события в других счетчиках, либо внешние входные сигналы (см. подраздел 9.4 «Источники событий для счета»).

9.3.1 Инициализация тактирования таймера

Перед началом работы с таймерами в первую очередь должны быть включены тактовые сигналы. Параметры задаются в блоке «Модуль управления синхронизацией и энергопотреблением».

Для задания тактовой частоты блока необходимо установить бит разрешения тактирования блока (регистр CFG8, бит 19 для таймера 0, бит 20 для таймера 1). Каждый таймер тактируется одной единственной частотой SOC-шины (TIM_CLK = SOCCLK). Затем необходимо разрешить работу таймеров (регистр CFG1, бит 19 для таймера 0, бит 20 для таймера 1).

После подачи тактового сигнала и разрешения работы таймера можно приступать к работе с ним.

9.3.2 Инициализация основного счетчика таймера

Чтобы запустить работу основного счетчика необходимо задать:

– начальное значение основного счетчика таймера в регистре CNT;

– значение основания счета для основного счетчика в регистре ARR;

– режим работы счетчика в регистре CNTRL:

  – выбрать источник события переключения счетчика EVENT_SEL[3:0];

  – режим счета основного счетчика CNT_MODE[1:0]:

    – значения 00 и 01 при тактировании внутренней частотой;

    – значения 10 при тактировании внешними сигналами;

  – направление счета основного счетчика DIR;

– при тактировании внутренней частотой установить значение предварительного делителя в регистре PSG, основной счетчик при этом будет считать на частоте TIM_CLKd = TIM_CLK / (PSG + 1);

– разрешить работу счетчика CNT_EN.

Значения регистров CNT, PSG и ARR можно изменять даже во время работы счетчика. Значения регистров CNT и PSG вступят в силу мгновенно после их записи. Значение регистра основания счета (ARR) может вступить в силу сразу после записи, если в регистре CNTRL бит ARRB_EN = 0.

При установленном бите ARRB_EN = 1 записанное значение ARR применяется при CNT == ARR. Необходимо учитывать, что если установлен прямой счет таймера, то новое значение ARR будет использоваться в следующем периоде счета. Если установлен обратный счет таймера, то новое значение ARR будет использовано через один период счета.

Поле CNT_MODE[1:0] в регистре CNTRL определяет режим работы основного счетчика:

– CNT_MODE[1:0] = 00 или 10 – направление счета определяется битом DIR:
  – DIR = 0 – счет прямой;
  – DIR = 1 – счет обратный;
– CNT_MODE[1:0] = 01 – счет двунаправленный с автоматическим изменением DIR.

9.3.3 Режимы счета

9.3.3.1 *Счет прямой: CNT_MODE[1:0] = 00, DIR = 0*

```
GTIMERx->CNTRL = 0x00000000;   //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMERx->CNT = 0x00000004;     //Начальное значение счетчика
GTIMERx->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMERx->ARR = 0x00000013;     //Основание счета
//Разрешение работы таймера
GTIMERx->CNTRL = 0x00000001;   //Счет прямой по TIM_CLK
```

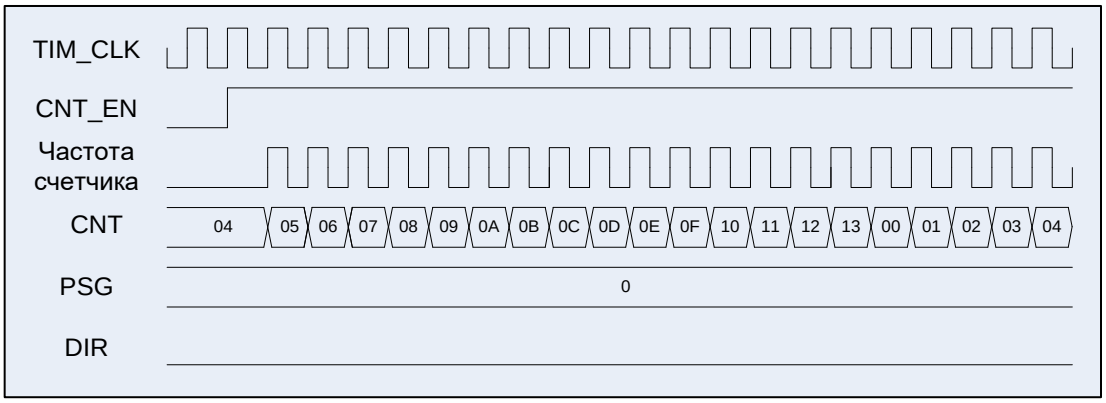


Рисунок 20 – Диаграммы работы таймера, счет прямой от 0 до 0x13, стартовое значение 0x04

9.3.3.2 *Счет обратный: CNT_MODE[1:0] = 00, DIR = 1*

```
GTIMERx->CNTRL = 0x00000000;   //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMERx->CNT = 0x00000004;     //Начальное значение счетчика
GTIMERx->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMERx->ARR = 0x00000013;     //Основание счета
//Разрешение работы таймера.
GTIMERx->CNTRL = 0x00000009;   //Счет обратный по TIM_CLK
```

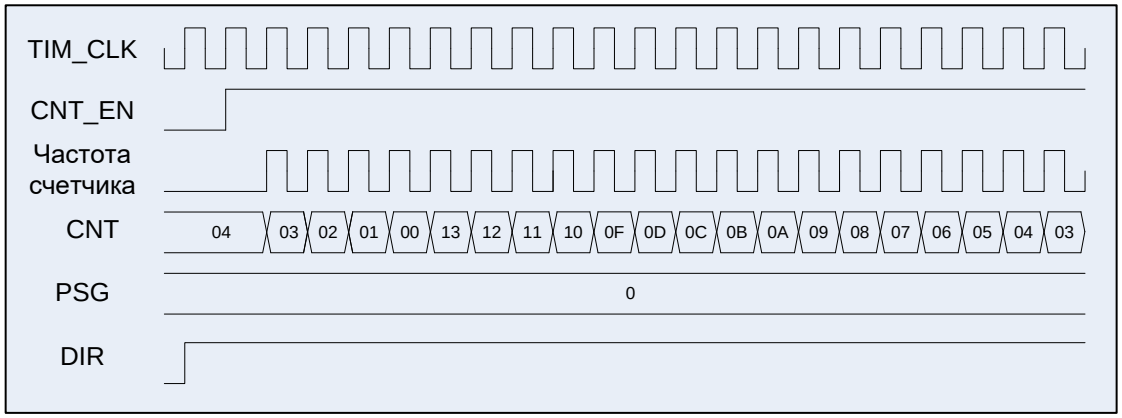


Рисунок 21 – Диаграммы работы таймера, счет обратный от 0x13 до 0, стартовое значение 0x04

9.3.3.3 *Счет двунаправленный: CNT_MODE = 01, DIR = 0*

```
GTIMERx->CNTRL = 0x00000000;   //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMERx->CNT = 0x00000004;     //Начальное значение счетчика
GTIMERx->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMERx->ARR = 0x00000013;     //Основание счета
//Разрешение работы таймера.
GTIMERx->CNTRL = 0x00000041;   //Счет двунаправленный по TIM_CLK
```

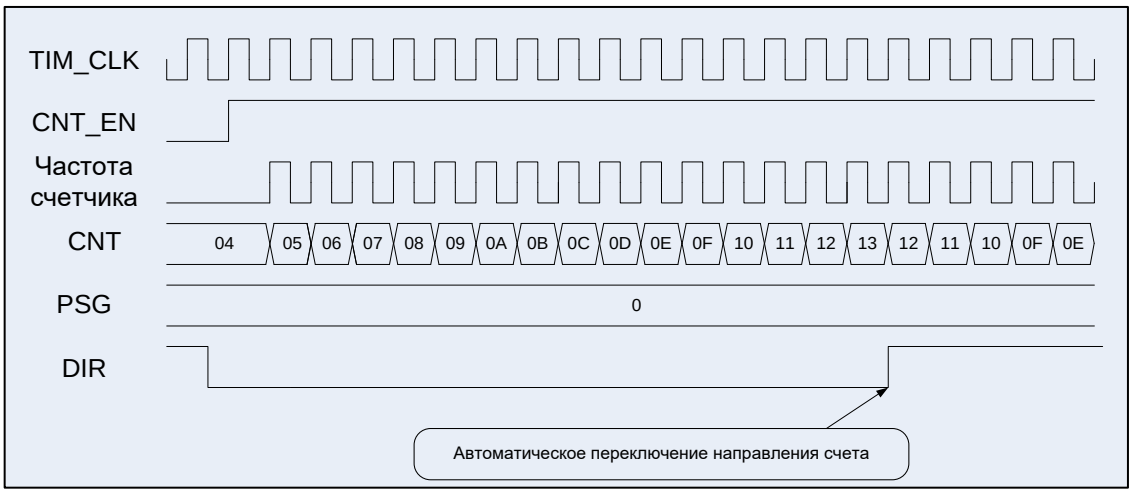


Рисунок 22 – Диаграммы работы таймера, счет двунаправленный, сначала прямой

9.3.3.4 *Счет двунаправленный: CNT_MODE = 01, DIR = 1*

```
GTIMERx->CNTRL  = 0x00000000;  //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMERx->CNT = 0x00000004;     //Начальное значение счетчика
GTIMERx->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMERx->ARR = 0x00000013;     //Основание счета
//Разрешение работы таймера.
GTIMERx->CNTRL = 0x00000049;   //Счет двунаправленный по TIM_CLK
```

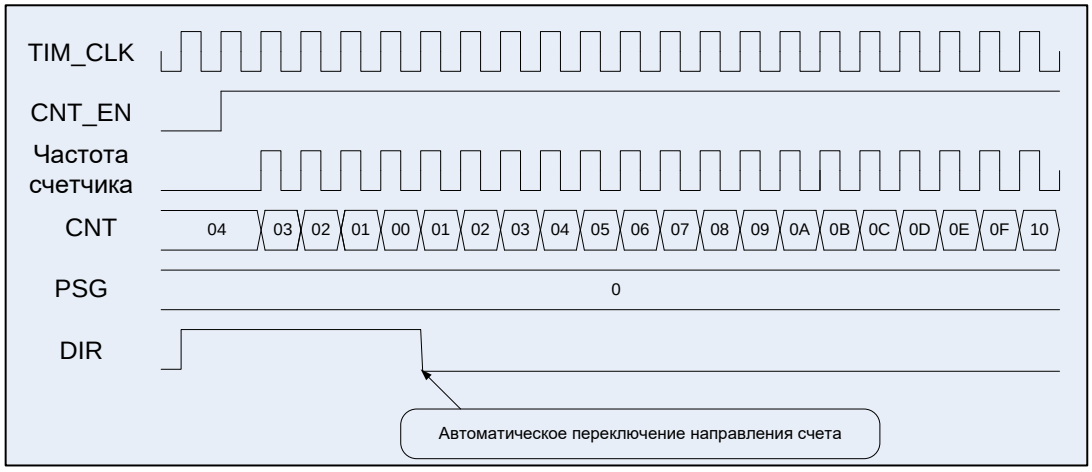


Рисунок 23 – Диаграммы работы таймера, счет двунаправленный, сначала обратный

9.3.4 Тактовая частота F_{DTS}

В блоке таймера предусмотрено формирование дополнительной тактовой частоты F_{DTS}, которая может использоваться для работы генератора «мертвой зоны» и цифровых фильтров на входах ETR и CHyi.

Тактовая частота F_{DTS} формируется из частоты TIM_CLK путём прореживания на заданный коэффициент (1, 2, 3 или 4). Настройка частоты F_{DTS} осуществляется в регистре CNTRL, поле FDTS[1:0].

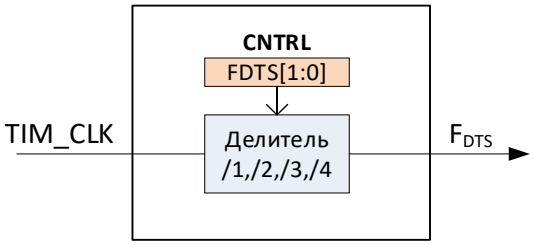


Рисунок 24 – Схема формирования тактовой частоты F_{DTS}

Диаграмма возможных частот F_{DTS} приведена на рисунке X7.

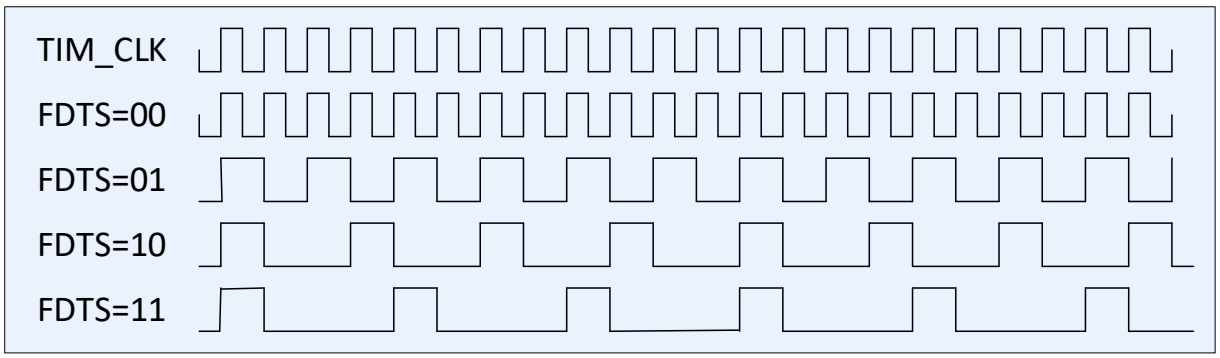


Рисунок 25 – Диаграмма тактовой частоты F_{DTS}
в зависимости от значения FDTS[1:0] в регистре CNTRL

9.4 Источники событий для счета

Тактирование основного счетчика таймера может осуществляться от следующих источников:

– внутренний тактовый сигнал (TIM_CLKd);
– событие в другом таймере (CNT==ARR);
– внешний тактовый сигнал, «Режим 1»: событие фронта на входе канала CHyi;
– внешний тактовый сигнал, «Режим 2»: событие фронта или среза на входе ETR.

Выбор источника тактирования основного счетчика осуществляется в регистре CNTRL, поле EVENT_SEL[3:0]. При выборе любого источника, кроме внутреннего тактового сигнала (EVENT_SEL[3:0] = 0000), необходимо также установить CNT_MODE[1:0] = 10 в регистре CNTRL.

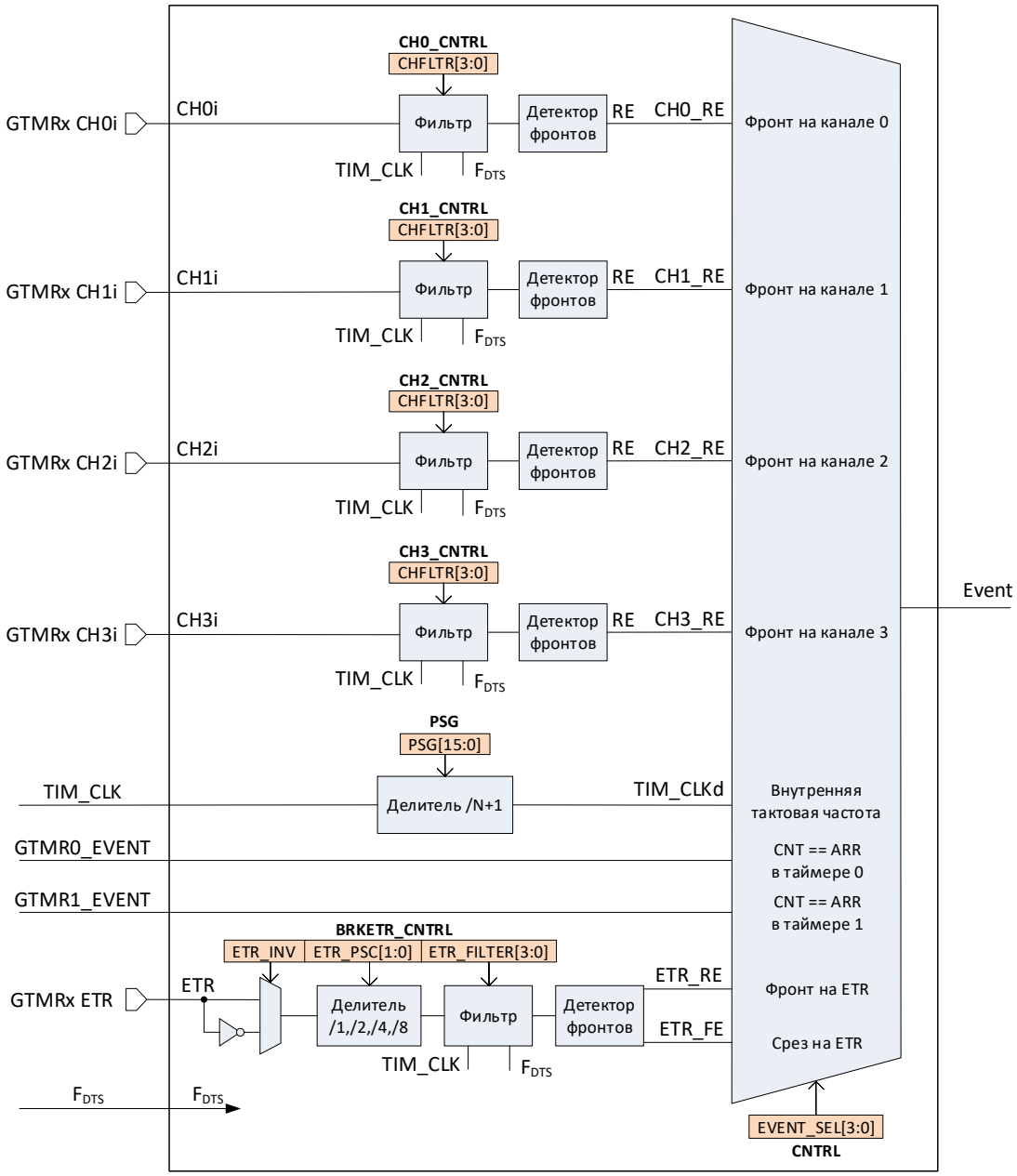


Рисунок 26 – Структурная схема формирования события для счета

9.4.1 Внутренний тактовый сигнал (TIM_CLKd)

Данный режим выбирается, когда EVENT_SEL[3:0] = 0000 и CNT_MODE[1:0] = 0x в регистре CNTRL. Основной счетчик таймера тактируется от внутренней частоты TIM_CLKd, которая формируется путем деления частоты TIM_CLK в соответствии с коэффициентом деления, записанным в регистре PSG.

Если значение предварительного делителя основного счетчика (PSG) неравно нулю, то счетный регистр делителя будет инкрементироваться по каждому импульсу сигнала TIM_CLK до тех пор, пока не достигнет значения, находящегося в регистре делителя. Далее счетный регистр делителя сбрасывается в ноль, содержимое основного счетчика таймера изменяется на 1 и счет начинается заново. Таким образом выходная частота предварительного делителя составляет:

$$\mathrm{TIM\_CLKd} = \frac{\mathrm{TIM\_CLK}}{\mathrm{PSG} + 1} \qquad (1)$$

Значение регистра PSG можно изменять даже во время работы счетчика, новое значение предделителя вступит в силу сразу после записи. На рисунках 27 и 28 приведены диаграммы работы счетчика при обновлении значения PSG.

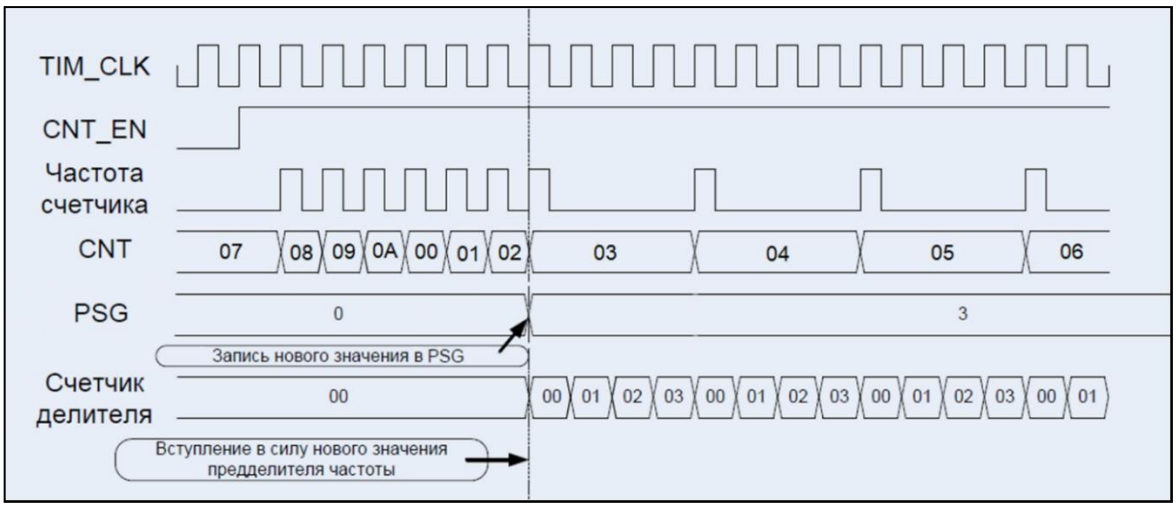


Рисунок 27 – Диаграмма работы счетчика: счет прямой (CNT_MODE[1:0] = 00, EVENT_SEL[3:0] = 0000, DIR = 0)

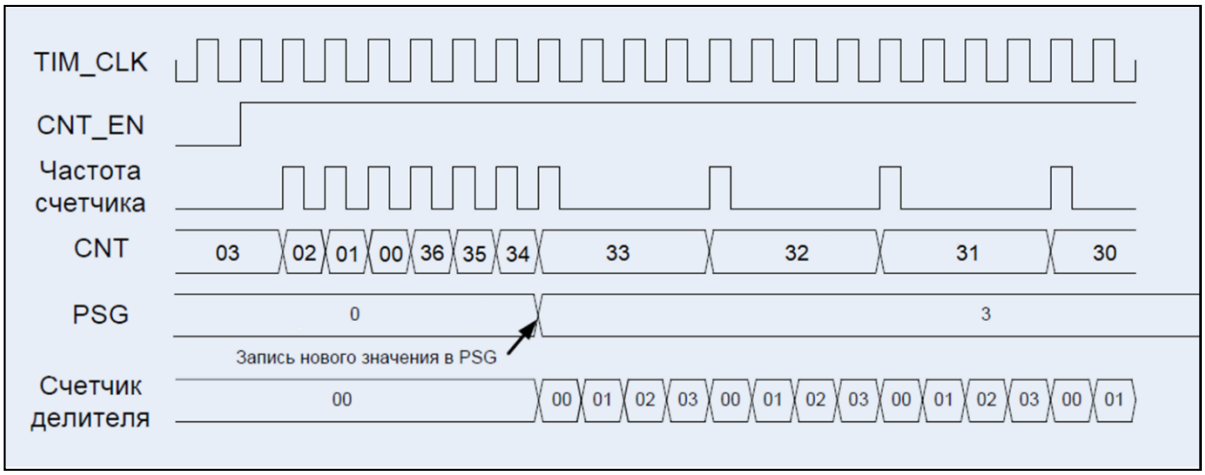


Рисунок 28 – Диаграмма работы счетчика: счет обратный (CNT_MODE[1:0] = 00, EVENT_SEL[3:0] = 0000, DIR = 1)

9.4.2 Событие в другом таймере (CNT==ARR)

Все таймеры полностью независимы друг от друга, но при этом у них предусмотрена возможность синхронизированной работы. Это позволяет создавать более сложные массивы таймеров, которые работают полностью автономно и не требуют написания какого-либо кода программы для выполнения сложных временных функций.

У каждого таймера имеется выход запуска GTMRx_EVENT, который соединен с входами других таймеров. Тактирование от другого таймера выбирается, когда EVENT_SEL[3:0] = 0001 или 0010, а также CNT_MODE[1:0] = 10 в регистре CNTRL. Основной счетчик таймера тактируется от другого таймера по сигналу GTMRx_EVENT, который устанавливается при CNT == ARR. Пересинхронизация сигнала GTMRx_EVENT (CNT == ARR) с одного таймера на другой происходит с задержкой один такт частоты TIM_CLK.

Синхронизация таймеров возможна в различных режимах. На рисунке 29 показан пример каскадного соединения таймеров, диаграммы работы данных таймеров приведены на рисунке 30.

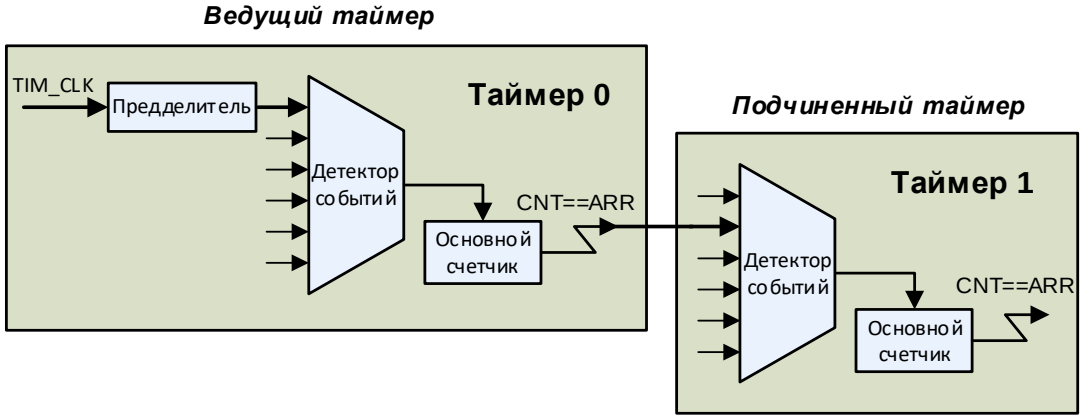


Рисунок 29 – Пример каскадного соединения таймеров

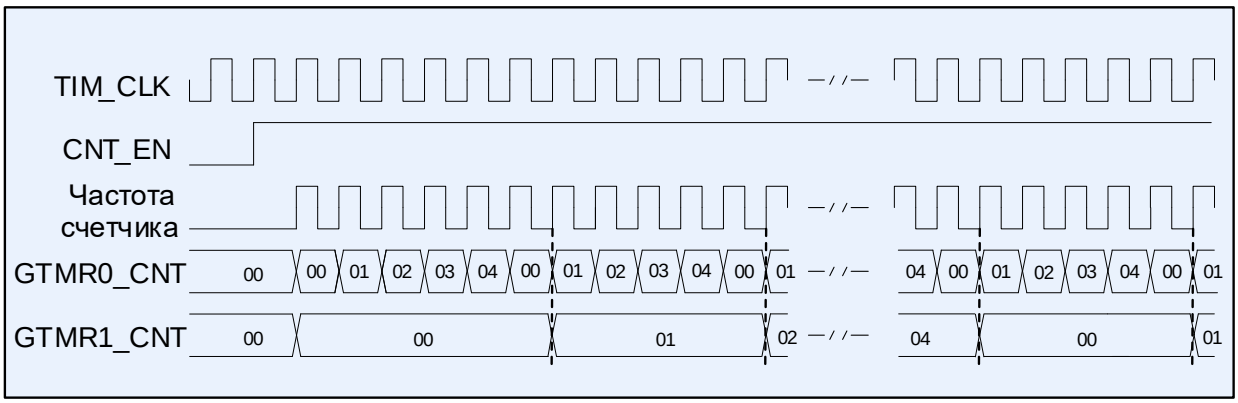


Рисунок 30 – Диаграммы работы двух таймеров в каскаде
GTIMER0: DIR = 0, EVENT_SEL[3:0] = 0000, CNT_MODE[1:0] = 00;
GTIMER1: DIR = 0, EVENT_SEL[3:0] = 0001, CNT_MODE[1:0] = 10.

9.4.3 Внешний тактовый сигнал, «Режим 1»: событие фронта на входе канала CHyi

Данный режим выбирается, когда EVENT_SEL[3:0] = 01xx и CNT_MODE[1:0] = 10. Основной счетчик таймера считает по фронту внешнего сигнала, поступающего на вход канала CHyi. Биты CHSEL[1:0] регистра CHy_CNTRL не оказывают влияния, так как они применяются для работы канала таймера только в режиме захвата.

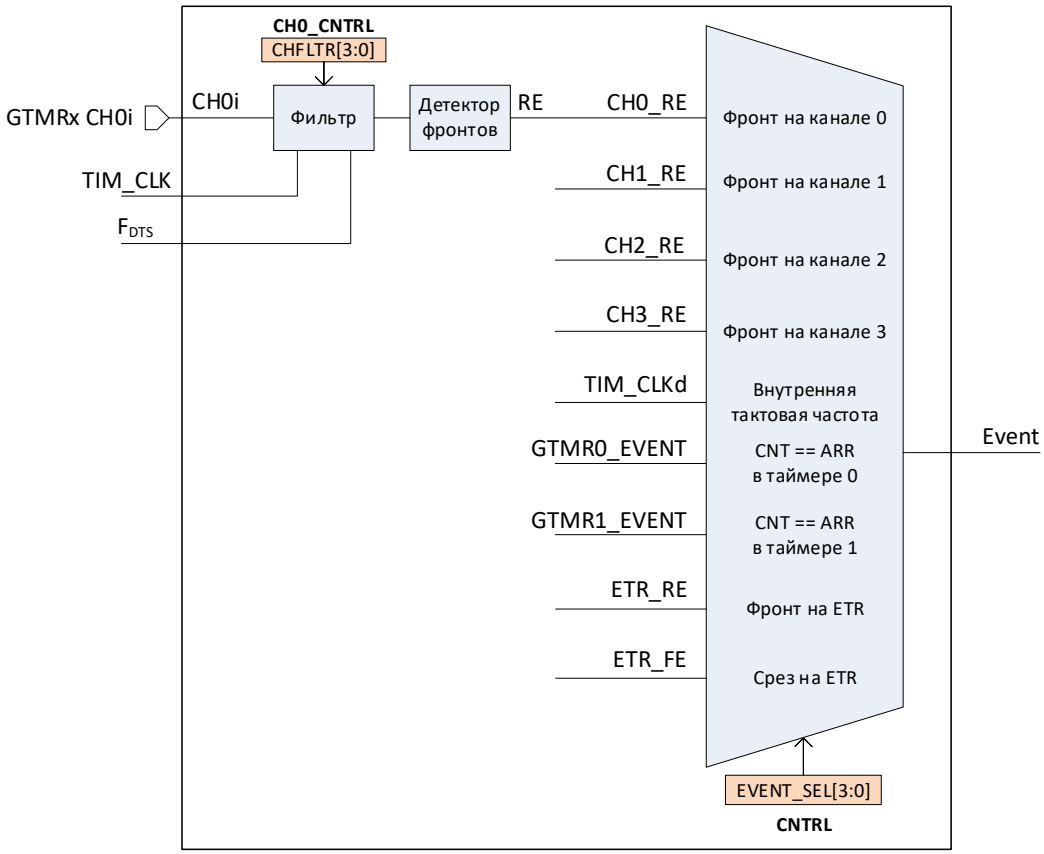


Рисунок 31 – Схема тактирования сигналом со входа канала 0

Со входа CHyi внешний тактовый сигнал поступает в блок цифрового фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 9.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле CHFLTR[3:0] регистра CHy_CNTRL.

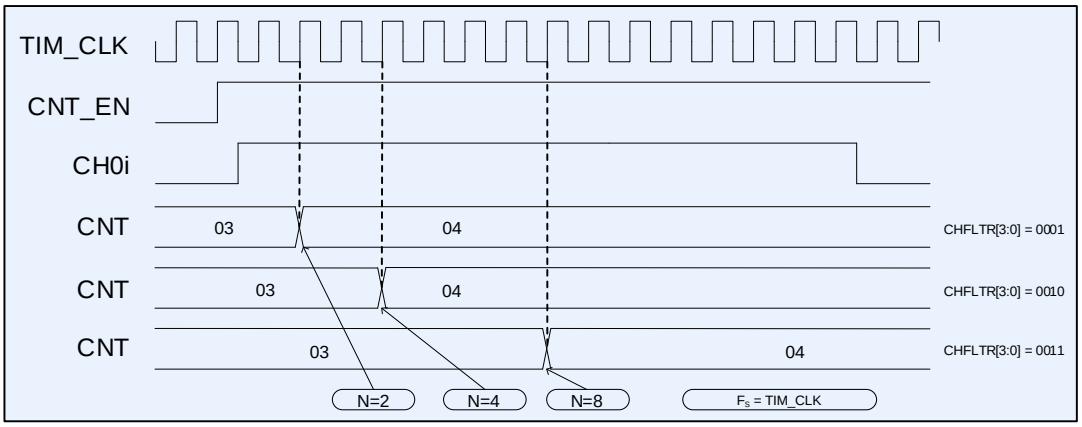


Рисунок 32 – Диаграмма внешнего тактирования с разными вариантами фильтра

9.4.4 Внешний тактовый сигнал, «Режим 2»: событие фронта или среза на входе ETR

Данный режим выбирается, когда EVENT_SEL[3:0] = 100x и CNT_MODE[1:0] = 10 в регистре CNTRL. Основной счетчик таймера может тактироваться по фронту или по срезу внешнего сигнала, поступающего на вход ETR, в зависимости от значения в поле EVENT_SEL[3:0].

Конфигурация тактового сигнала со входа ETR задается в регистре BRKETR_CNTRL. Бит ETR_INV позволяет установить инверсию входного сигнала. Поле ETR_PSC[1:0] задает коэффициент деления асинхронного предделителя внешней частоты (1, 2, 4 или 8). После предделителя тактовый сигнал поступает в блок цифрового фильтра, где он может быть дополнительно отфильтрован с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 9.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле ETR_FILTER[3:0].

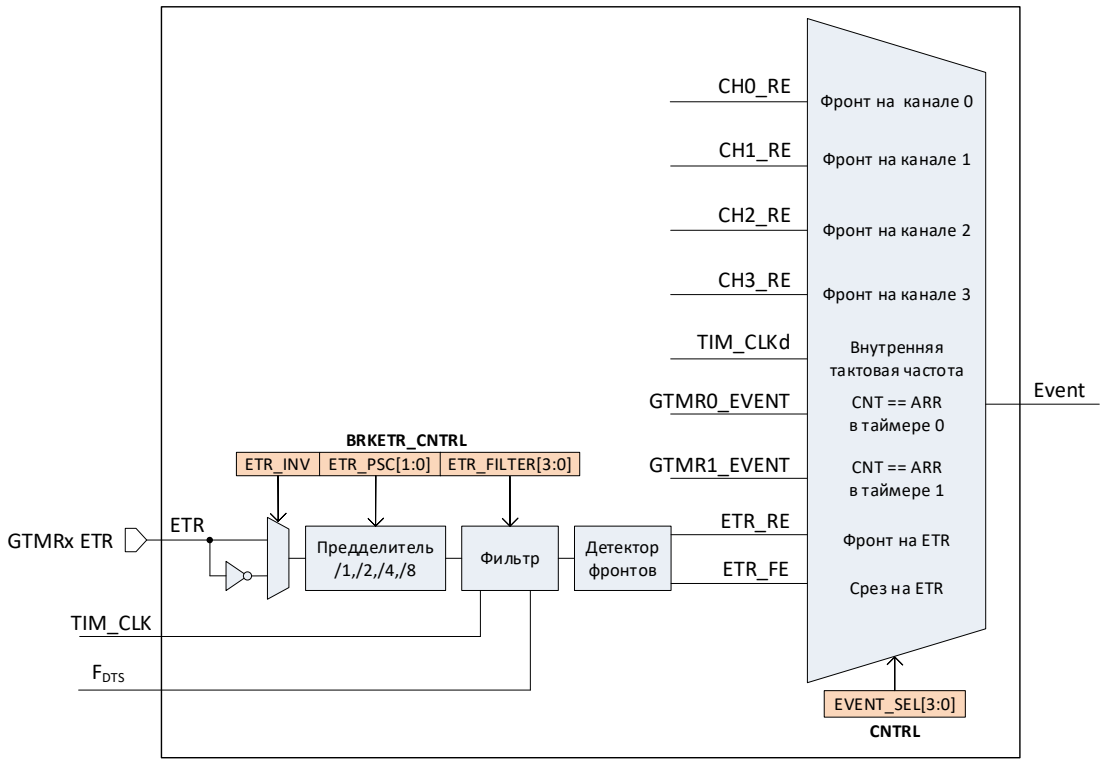


Рисунок 33 – Схема тактирования сигналом со входа ETR

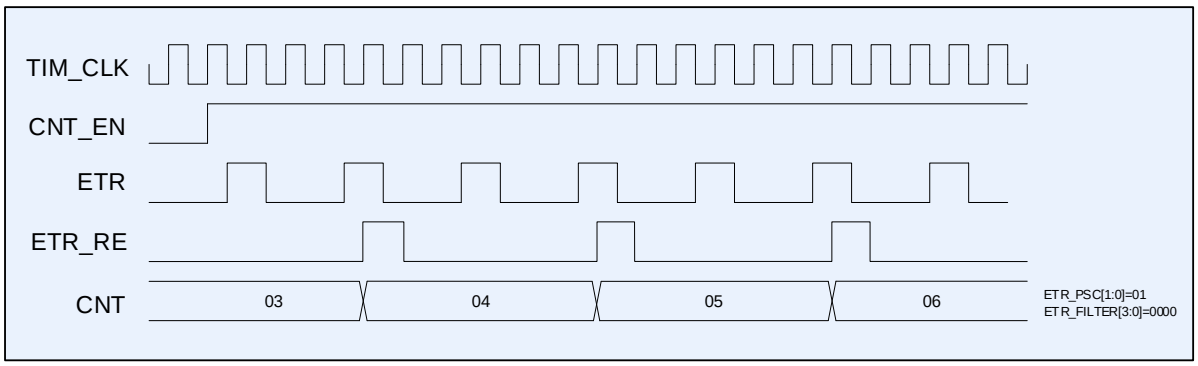


Рисунок 34 – Диаграмма тактирования со входа ETR, EVENT_SEL[3:0] = 1000

9.5 Режим захвата

Каждый канал таймера может быть независимо переведен в режим захвата. В режиме захвата по событию от внешнего входного сигнала происходит фиксация значения основного счетчика CNT в регистры CCRy (CCR) и CCRy1 (CCR1).

Структурная схема блока захвата представлена на рисунке 35.

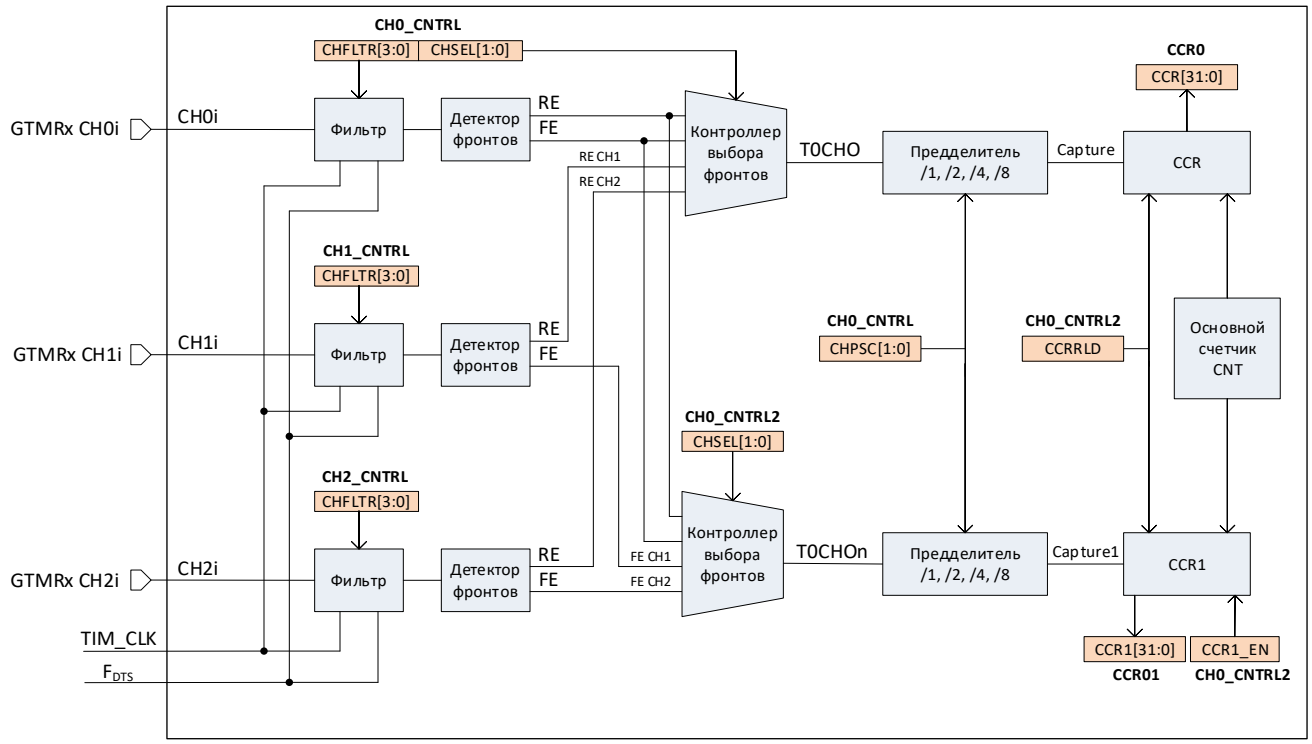


Рисунок 35 – Структурная схема блока захвата на примере канала 0

Для включения режима захвата для определенного канала необходимо записать «1» в бит CAP_NPWM регистра управления каналом CHy_CNTRL. Для использования регистра CCRy1(CCR1) необходимо записать «1» в бит CCR1_EN регистра CHy_CNTRL2.

Внешний сигнал со входа CHyi сначала поступает в блок фильтра. Данный блок позволяет отфильтровать входной сигнал с целью устранения импульсов, длительность которых меньше заданного порога (см. подраздел 9.7 «Блок цифрового фильтра»). Настройки фильтра задаются в поле CHFLTR[3:0] регистра CHy_CNTRL.

Сигнал с блока фильтра поступает в блок «Детектор фронтов». При обнаружении фронта входного сигнала данный блок вырабатывает сигнал RE, а при обнаружении среза входного сигнала – сигнал FE.

В блоке «Контроллер выбора фронтов» производится выбор используемого для захвата события между фронтом сигнала на входе канала, срезом сигнала на входе канала и фронтом или срезом сигналов на входах других каналов. Настройка блока «Контроллер выбора фронтов» для регистра CCRy осуществляется в поле CHSEL[1:0] регистра CHy_CNTRL, а для регистра CCRy1 – в поле CHSEL[1:0] регистра CHy_CNTRL2. Выбранный для захвата сигнал поступает в предварительный делитель, который в зависимости от значения в поле CHPSC[1:0] регистра CHy_CNTRL позволяет фиксировать все события, либо каждое второе, каждое четвертое или каждое восьмое событие.

Предварительный делитель для регистра CCRy формирует сигнал Capture, а предварительный делитель для регистра CCRy1 формирует сигнал Capture1. По сигналам Capture и Capture1 выполняется запись текущего значения основного счетчика CNT в регистры CCRy и CCRy1.

На рисунке 36 показан пример захвата значения основного счетчика CNT в регистр CCRy по фронту на входе канала, а в регистр CCRy1 – по срезу на входе канала.

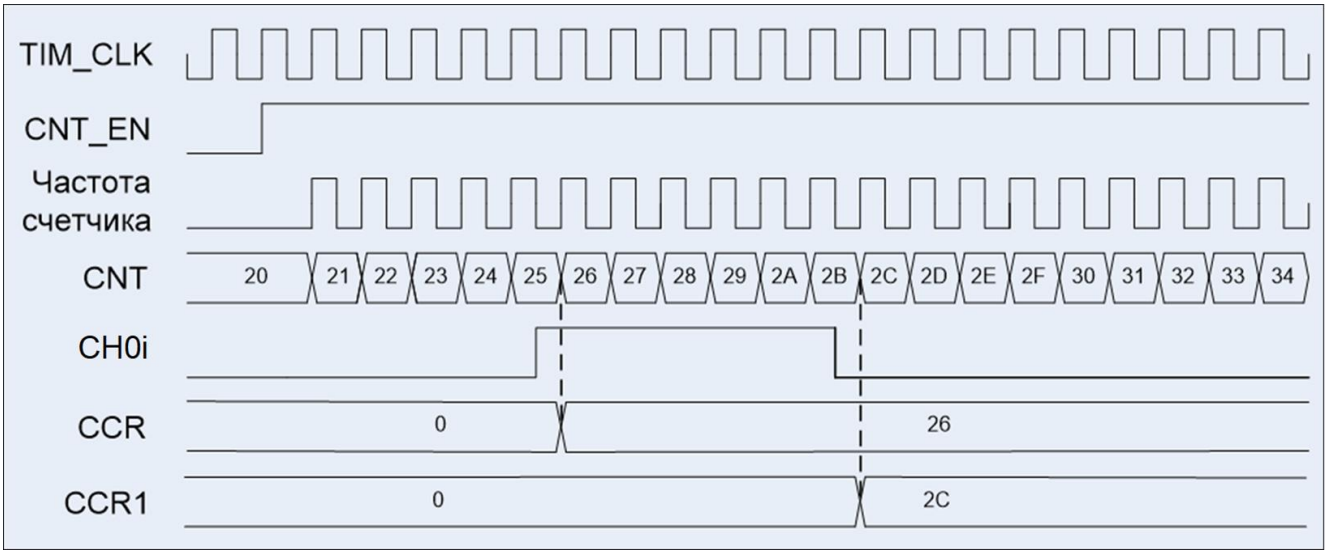


Рисунок 36 – Диаграмма захвата события со входа канала 0

По событию захвата на определенном канале в регистре IE можно разрешить выработку прерываний, а в регистрах DMA_RE и DMA_RE1-DMA_RE4 можно разрешить формирование запросов DMA.

Между формированием события захвата и записью текущего значения основного счетчика CNT в регистры CCRy и CCRy1 может быть установлена задержка с помощью бита EV_DELAY в регистре CHy_CNTRL2. Если бит EV_DELAY равен «0», то сначала формируется событие захвата, а затем через один такт сигнала синхронизации TIM_CLK выполняется запись CNT в регистры CCRy и CCRy1. Если бит EV_DELAY равен «1», то обновление информации в регистрах CCRy и CCRy1 происходит синхронно с событием захвата. При реализации чтения регистров CCRy и CCRy1 по событию захвата рекомендуется устанавливать бит EV_DELAY в «1».

9.6 Режим ШИМ

Каждый канал таймера может быть независимо переведен в режим ШИМ для формирования выходных сигналов с возможностью задания «мертвой зоны». Структурная схема блока формирования ШИМ представлена на рисунке 37.

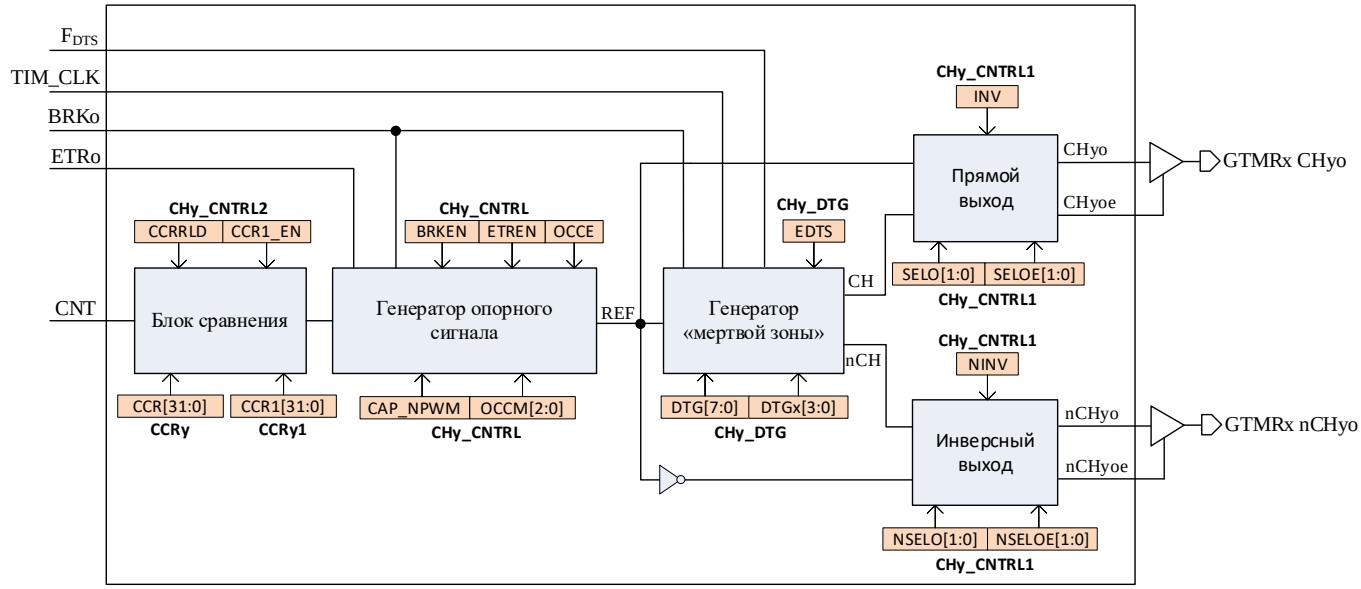


Рисунок 37 – Структурная схема блока формирования ШИМ

Для включения режима ШИМ для определенного канала необходимо в регистре управления каналом CHy_CNTRL записать «0» в бит CAP_NPWM.

9.6.1 Генератор опорного сигнала REF

При работе в режиме ШИМ блок генератора опорного сигнала формирует сигнал REF. Данный сигнал формируется на основании сравнения значения в регистрах CCRy (CCR), CCRy1 (CCR1) и основного счетчика CNT. Формат выработки сигнала REF устанавливается в поле OCCM[2:0] регистра управления каналом таймера CHy_CNTRL.

Если в регистре CHy_CNTRL2 бит CCR1_EN = 0, то для формирования сигнала REF используется только результат сравнения значения в регистре CCRy и основного счетчика CNT.

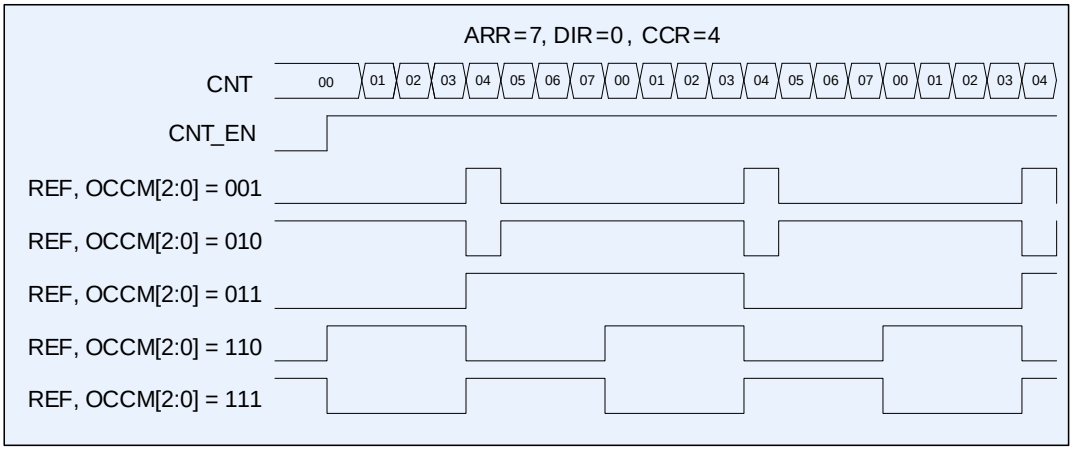


Рисунок 38 – Диаграмма работы в режиме ШИМ, CCR1_EN=0

Если в регистре CHy_CNTRL2 бит CCR1_EN = 1, то для формирования сигнала REF задействуются оба результата сравнения значения в регистрах CCRy, CCRy1 и основного счетчика CNT.

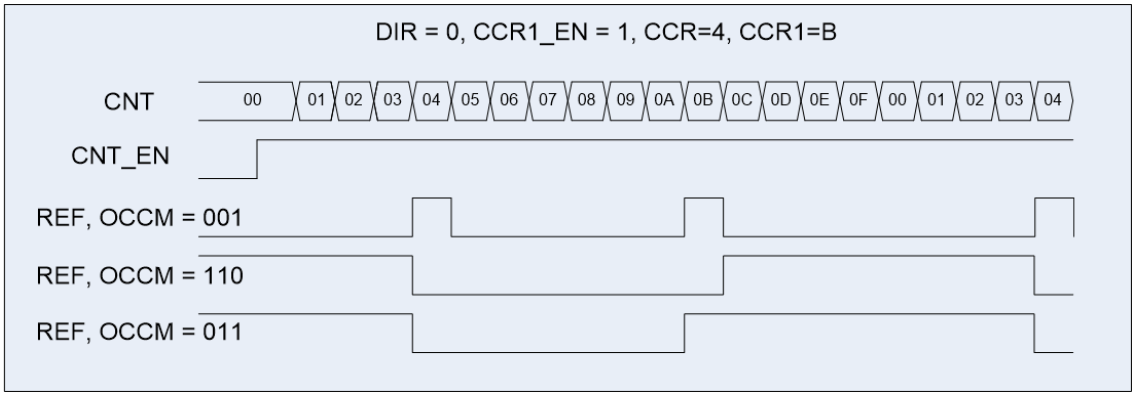


Рисунок 39 – Диаграмма работы в режиме ШИМ, CCR1_EN = 1

Запись новых значений в регистры CCRy и CCRy1 осуществляется немедленно, если в регистре CHy_CNTRL2 бит CCRRLD установлен в «0», иначе регистры CCRy и CCRy1 получат новые значения только при CNT == 0. Процесс обновления значений в регистрах CCRy и CCRy1 обозначается в регистре CHy_CNTRL с помощью флагов WR_CMPL и WR_CMPL1, соответственно. На время выполнения записи флаг WR_CMPL/WR_CMPL1 устанавливается в «1», по окончании записи флаг WR_CMPL/WR_CMPL1 сбрасывается в «0».

Сигнал REF может быть принудительно установлен в «0» с использованием внешнего сигнала сброса, поступающего со входа ETR (высокий активный уровень) или со входа BRK (низкий активный уровень). Активный уровень на входах ETR и BRK может быть изменён с помощью инверсии входного сигнала, регистр BRKETR_CNTRL, биты ETR_INV и BRK_INV, соответственно.

Для разрешения сброса сигнала REF по входу ETR необходимо установить бит ETREN и OCCE в регистре CHy_CNTRL. Активный уровень на входе ETR сбрасывает сигнал REF в «0». После снятия активного уровня на входе ETR сигнал REF остается в «0» до следующего события установки REF в «1», рисунок 40.

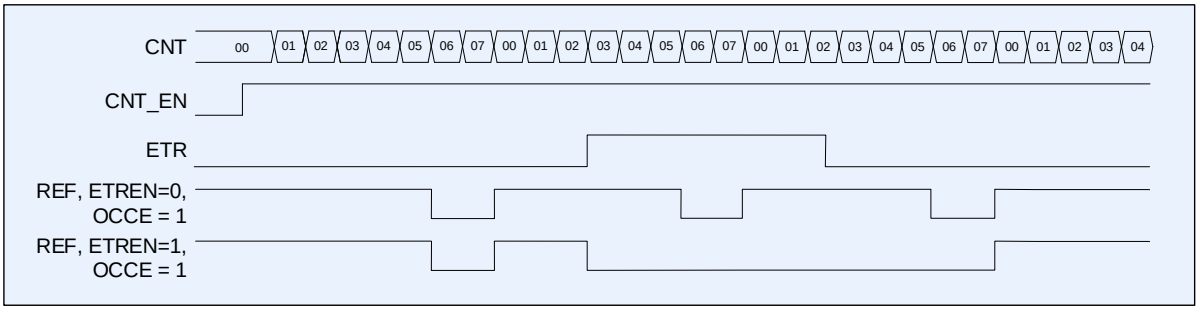


Рисунок 40 – Диаграмма сброса сигнала REF по выводу ETR

Для разрешения сброса сигнала REF по входу BRK необходимо установить бит BRKEN в регистре CHy_CNTRL. Активный уровень на входе BRK сбрасывает сигнал REF в «0» путем маскирования. После снятия активного уровня на входе BRK генерация сигнала REF сразу же восстанавливается, рисунок 41.

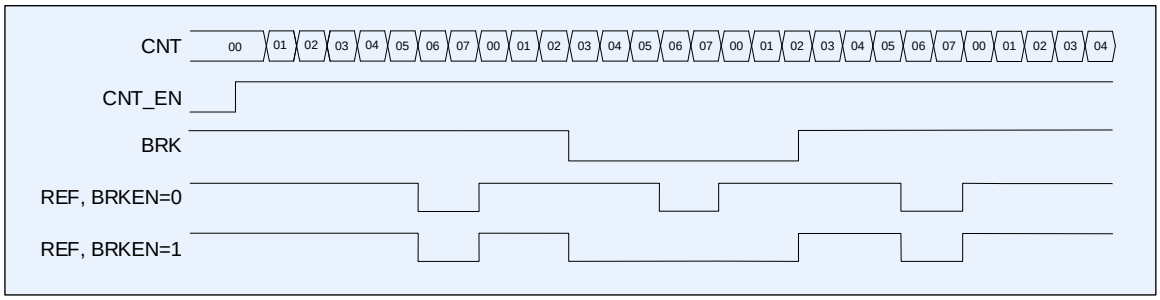


Рисунок 41 – Диаграмма сброса сигнала REF по выводу BRK

9.6.2 Генератор «мертвой зоны»

Блок генератора «мертвой зоны» (dead-time generator, DTG) позволяет на основе сигнала REF формировать комплементарную пару сигналов с «мертвой зоной». Выходные сигналы с блока DTG передаются на выходные блоки следующим образом:

– сигнал на прямом выходе (CHyo, CHyoe) представляет собой инвертированный сигнал REF, в котором фронт задержан на величину DTGdel относительно среза опорного сигнала REF;

– сигнал на инверсном выходе (nCHyo, nCHyoe) представляет собой сигнал REF, в котором фронт задержан на величину DTGdel относительно фронта опорного сигнала REF.

Значение «мертвой зоны» между сигналами на прямом и инверсном выходах рассчитывается в тактах частоты TIM_CLK или F_{DTS} по формуле

$$\mathrm{DTGdel} = \mathrm{DTG} \cdot (\mathrm{DTGx} + 1)\,, \qquad (2)$$

где DTGx – предварительный делитель частоты;

DTG – основной делитель частоты.

Управление блоком DTG осуществляется через регистр CHy_DTG. Выбор источника тактирования для задания «мертвой зоны» задается битом EDTS. Значения делителей DTGx и DTG задаются в полях DTGx[3:0] и DTG[7:0], соответственно. Если задержка DTGdel больше ширины импульса высокого уровня, то соответствующий импульс не генерируется.

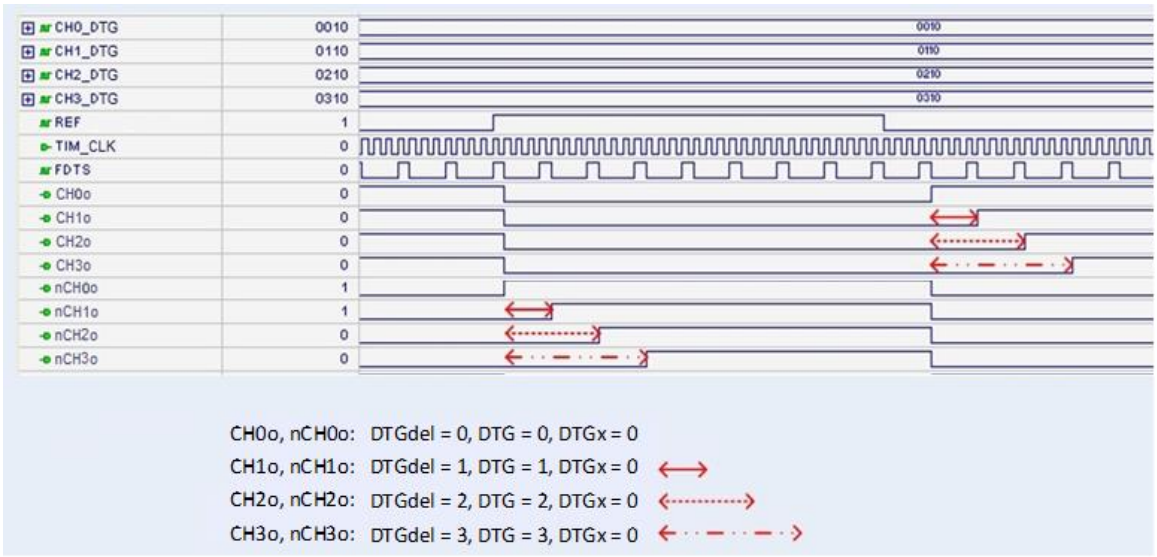


Рисунок 42 – Диаграмма работы блока DTG

Выходные сигналы блока DTG могут быть принудительно установлены в «0» с использованием внешнего сигнала сброса, поступающего со входа BRK (низкий активный уровень). Активный уровень на входе BRK может быть изменён с помощью инверсии входного сигнала, бит BRK_INV в регистре BRKETR_CNTRL. Для разрешения сброса выходных сигналов блока DTG по входу BRK необходимо установить бит BRKEN в регистре CHy_CNTRL. Активный уровень на входе BRK сбрасывает выходные сигналы блока DTG в «0» путем их маскирования.

9.6.3 Выходные блоки

Каждый канал таймера имеет два выходных блока – прямой и инверсный. Каждый выходной блок формирует как сигнал выдачи (CHyo, nCHyo), так и сигнал разрешения выдачи (CHyoe, nCHyoe). В качестве сигналов для прямого (CHyo, CHyoe) и инверсного (nCHyo, nCHyoe) выходов в регистре CHy_CNTRL1 могут быть заданы либо постоянные уровни (0 или 1), либо сигналы, формируемые на основе сигнала REF. К таким сигналам относится сам сигнал REF, а также сигналы, формируемые блоком DTG.

Выбор источника сигнала выдачи для прямого (CHyo) и инверсного (nCHyo) выходов задается в полях SELO[1:0] и NSELO[1:0] регистра CHy_CNTRL1. Дополнительно каждый сигнал выдачи для прямого (CHyo) и инверсного (nCHyo) выходов может быть инвертирован путем установки битов INV и NINV в регистре CHy_CNTRL1. Выбор источника сигнала разрешения выдачи для прямого (CHyoe) и инверсного (nCHyoe) выходов задается в полях SELOE[1:0] и NSELOE[1:0] регистра CHy_CNTRL1. При этом, если сигнал разрешения выдачи равен «0», то соответствующий вывод работает в режиме входа, если сигнал разрешения выдачи равен «1» – то в режиме выхода.

9.7 Блок цифрового фильтра

В тракте входа ETR и входов каналов таймера CHyi предусмотрен блок цифрового фильтра, который позволяет исключить из входного сигнала импульсы высокого и низкого уровня, длительность которых меньше заданного порога.

Конфигурация фильтра для входа ETR выполняется в поле ETR_FILTER[3:0] регистра BRKETR_CNTRL, для входов каналов CHyi – в поле CHFLTR[3:0] регистра CHy_CNTRL. Значение в данных полях позволяет настроить два параметра фильтра:

– частота выборки F_S, на которой входной сигнал захватывается в сдвиговый регистр для накопления. В качестве частоты F_S может использоваться частота TIM_CLK или F_{DTS};

– количество выборок (длина фильтра) N, на протяжении которых входной сигнал должен оставаться стабильным, чтобы не подвергнуться фильтрации.

Если в течение заданного количества выборок N на частоте F_S входной сигнал не изменяется, то значение входного сигнала передаётся на выход фильтра. Иначе внутренний счетчик накопления сбрасывается и захват сигнала начинается заново.

Таким образом, настраивая частоту F_S и количество выборок N, задается минимальная длительность импульсов входного сигнала, которые не будут

отфильтрованы. Диаграмма работы фильтра при использовании частоты TIM_CLK приведена на рисунке 43, частота F_S = TIM_CLK, количество выборок N = 4.

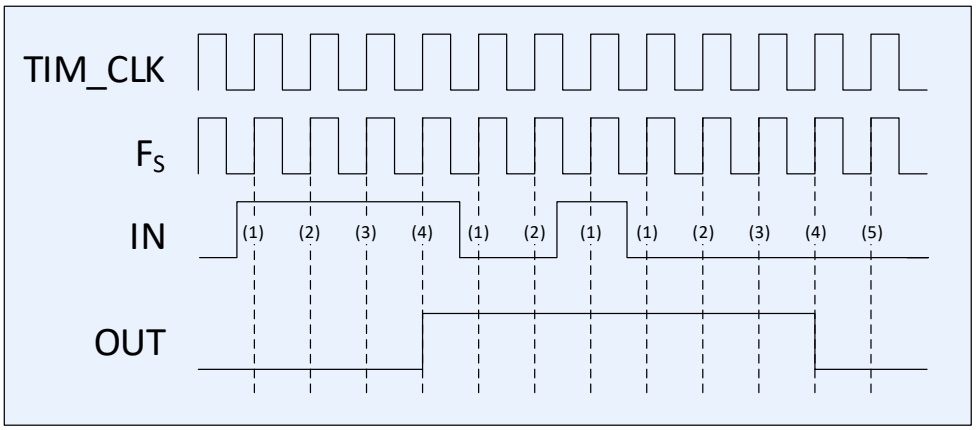


Рисунок 43 – Диаграмма работы фильтра, F_S = TIM_CLK, N = 4

Для задания длительных интервалов накопления входного сигнала имеется возможность использовать частоту F_{DTS}, которая формируется из частоты TIM_CLK путём прореживания на заданный коэффициент (см. пункт 9.3.4 «Тактовая частота F_{DTS}»).

Диаграмма работы фильтра при использовании частоты F_{DTS} приведена на рисунке 44, частота выборки $F_S = F_{DTS}/2$, количество выборок N = 6.

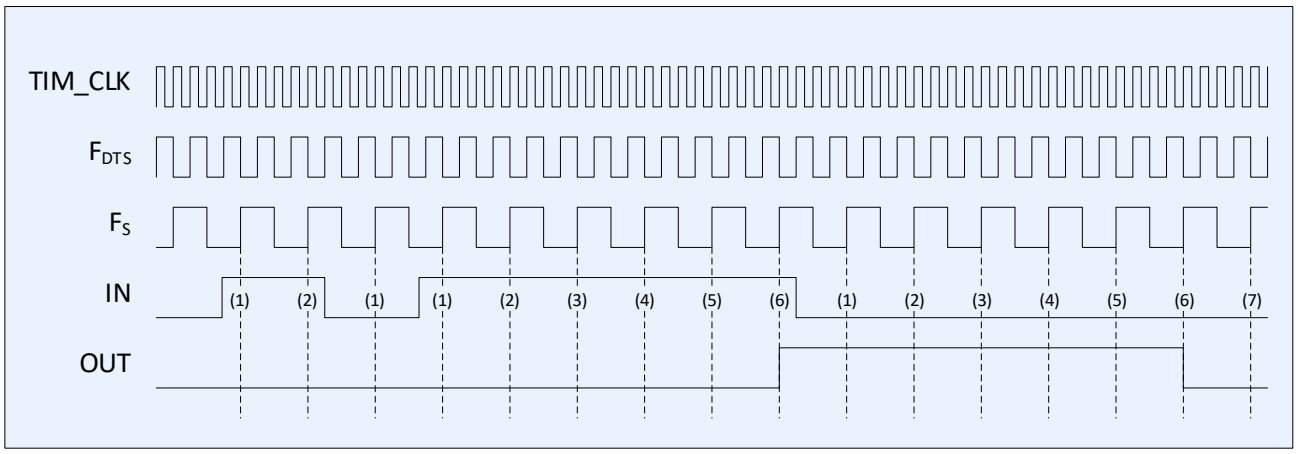


Рисунок 44 – Диаграмма работы фильтра, $F_S = F_{DTS}/2$, N = 6

Возможные варианты настройки блока фильтра приведены в таблице 37.

Таблица 37 – Возможные конфигурации фильтра

CHFLTR[3:0], ETR_FILTER[3:0]	Частота выборки Fs	Количество выборок N	Минимальная длительность импульсов, которые не будут отфильтрованы
0000	F_{DTS}	1	-
0001	TIM_CLK	2	$2 \times T_{TIM\_CLK}$
0010	TIM_CLK	4	$4 \times T_{TIM\_CLK}$
0011	TIM_CLK	8	$8 \times T_{TIM\_CLK}$
0100	$F_{DTS}/2$	6	$12 \times T_{FDTS}$
0101	$F_{DTS}/2$	8	$16 \times T_{FDTS}$
0110	$F_{DTS}/4$	6	$24 \times T_{FDTS}$
0111	$F_{DTS}/4$	8	$32 \times T_{FDTS}$

CHFLTR[3:0], ETR_FILTER[3:0]	Частота выборки Fs	Количество выборок N	Минимальная длительность импульсов, которые не будут отфильтрованы
1000	$F_{DTS}/8$	6	$48 \times T_{FDTS}$
1001	$F_{DTS}/8$	8	$64 \times T_{FDTS}$
1010	$F_{DTS}/16$	5	$80 \times T_{FDTS}$
1011	$F_{DTS}/16$	6	$96 \times T_{FDTS}$
1100	$F_{DTS}/16$	8	$128 \times T_{FDTS}$
1101	$F_{DTS}/32$	5	$160 \times T_{FDTS}$
1110	$F_{DTS}/32$	6	$192 \times T_{FDTS}$
1111	$F_{DTS}/32$	8	$256 \times T_{FDTS}$

9.8 Флаги состояний, прерывания и запросы DMA

В процессе работы блок таймера отслеживает состояние внутренних блоков и формирует 17 событий:

– CNT ZERO EVENT – совпадение значения счетчика CNT с нулем;

– CNT ARR EVENT – совпадение значения счетчика CNT со значением в регистре ARR;

– ETR RE EVENT – фиксация фронта на входе ETR;

– ETR FE EVENT – фиксация среза на входе ETR;

– BRK EVENT – фиксация высокого уровня на входе BRK;

– CCR CAP EVENT[3:0] – запись значения счетчика CNT в регистр CCRy по захвату настроенного фронта (фронт или срез) на входе канала CHyi, события формируются индивидуально для каждого канала;

– CCR REF EVENT[3:0] – фиксация фронта на выходе генератора опорного сигнала REF, события формируются индивидуально для каждого канала;

– CCR CAP1 EVENT[3:0] – запись значения счетчика CNT в регистр CCRy1 по захвату настроенного фронта (фронт или срез) на входе канала CHyi, события формируются индивидуально для каждого канала.

9.8.1 Флаги состояний

При возникновении события устанавливается соответствующий флаг в регистре STATUS. Сброс флагов в регистре STATUS осуществляется записью «0», запись «1» не оказывает влияния. Если запись «0» выполняется одновременно с новым событием, то приоритет у нового события.

9.8.2 Прерывания

Блок таймера на основе флагов в регистре STATUS формирует один общий сигнал запроса прерывания INT_GTMRx. Выбор флагов, формирующих запрос прерывания, осуществляется через регистр разрешения прерываний IE. При формировании запроса прерывания маскированные состояния флагов из регистра STATUS объединяются по схеме ИЛИ.

9.8.3 Запросы DMA

На основе отслеживаемых событий блок таймера формирует сигналы запросов DMA GTMRx_REQ[4:0]. Выбор событий, формирующих запрос DMA, осуществляется через регистры DMA_RE и DMA_RE1-DMA_RE4. Данные регистры имеют одинаковые поля, при этом каждый из регистров отвечает за конфигурацию индивидуального запроса от таймера к DMA:

– регистр DMA_RE конфигурирует формирование запроса 0;

– регистры DMA_RE1-DMA_RE4 конфигурируют формирование запросов 1-4, соответственно.

Запросы 1-4 не привязаны к определенным каналам таймера, поэтому в полях CCR CAP EVENT RE[3:0], CCR REF EVENT RE[3:0] и CCR CAP1 EVENT RE[3:0] регистров DMA_RE1-DMA_RE4 может быть выбрано событие от любого канала таймера. Например, событие настроенного фронта на входе CH2i второго канала таймера (CCR CAP EVENT RE[3:0]=0x4) может активировать запрос 1 при конфигурации регистра DMA_RE1 = 0x80.

9.9 Примеры

В данном разделе приведены примеры инициализации таймера 0 в различных режимах работы. Для другого таймера инициализация выполняется аналогично.

9.9.1 Обычный счетчик

```
CMU->CFG8 &= ~0x00180000;     //Разрешение тактирования таймера 0 и 1
                              //(TIM_CLK = SOCCLK)
CMU->CFG1 |= 0x00080000;      //Разрешение работы таймера 0

GTIMER0->CNTRL = 0x00000000;  //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMER0->CNT = 0x00000000;    //Начальное значение счетчика
GTIMER0->PSG = 0x00000000;    //Предделитель частоты TIM_CLK
GTIMER0->ARR = 0x0000000F;    //Основание счета
GTIMER0->IE  = 0x00000002;    //Разрешение генерировать прерывание при
                              //CNT == ARR
//Разрешение работы таймера
GTIMER0->CNTRL = 0x00000001;  //Счет прямой по TIM_CLK
```

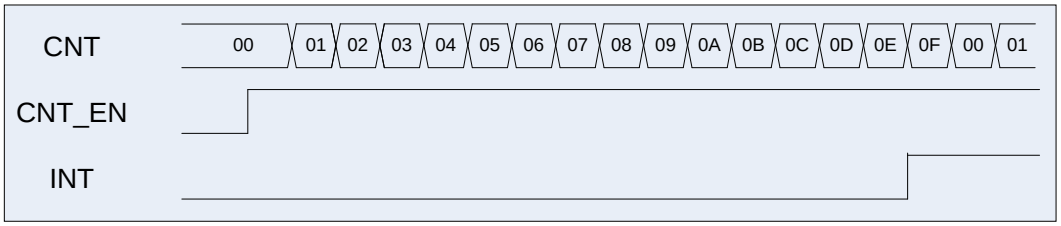


Рисунок 45 – Режим обычного счетчика

9.9.2 Режим захвата

```
CMU->CFG8 &= ~0x00180000;      //Разрешение тактирования таймера 0 и 1
                               //(TIM_CLK = SOCCLK)
CMU->CFG1 |= 0x00080000;       //Разрешение работы таймера 0
GTIMER0->CNTRL = 0x00000000;   //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMER0->CNT = 0x00000000;     //Начальное значение счетчика
GTIMER0->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMER0->ARR = 0x000000FF;     //Основание счета
GTIMER0->IE = 0x000001E0;      //Разрешение генерировать прерывание по событию
                               //настроенного фронта на входах CH0i-CH3i
GTIMER0->CH0_CNTRL = 0x00008000; //Захват по фронту сигнала на входе CH0i,
                                 //фильтрация отключена
GTIMER0->CH1_CNTRL = 0x00008010; //Захват по срезу сигнала на входе CH1i,
                                 //фильтрация отключена
GTIMER0->CH2_CNTRL = 0x00008001; //Захват по фронту сигнала на входе CH2i,
                                 //фильтрация выполняется по 2 выборкам
                                 //на частоте TIM_CLK
GTIMER0->CH3_CNTRL = 0x00008011; //Захват по срезу сигнала на входе CH3i,
                                 //фильтрация выполняется по 2 выборкам
                                 //на частоте TIM_CLK
//Режим работы выходов канала – выходы канала в третьем состоянии
GTIMER0->CH0_CNTRL1= 0x00000000;
GTIMER0->CH1_CNTRL1= 0x00000000;
GTIMER0->CH2_CNTRL1= 0x00000000;
GTIMER0->CH3_CNTRL1= 0x00000000;
//Разрешение работы таймера
GTIMER0->CNTRL = 0x00000001;   //Счет прямой по TIM_CLK
```

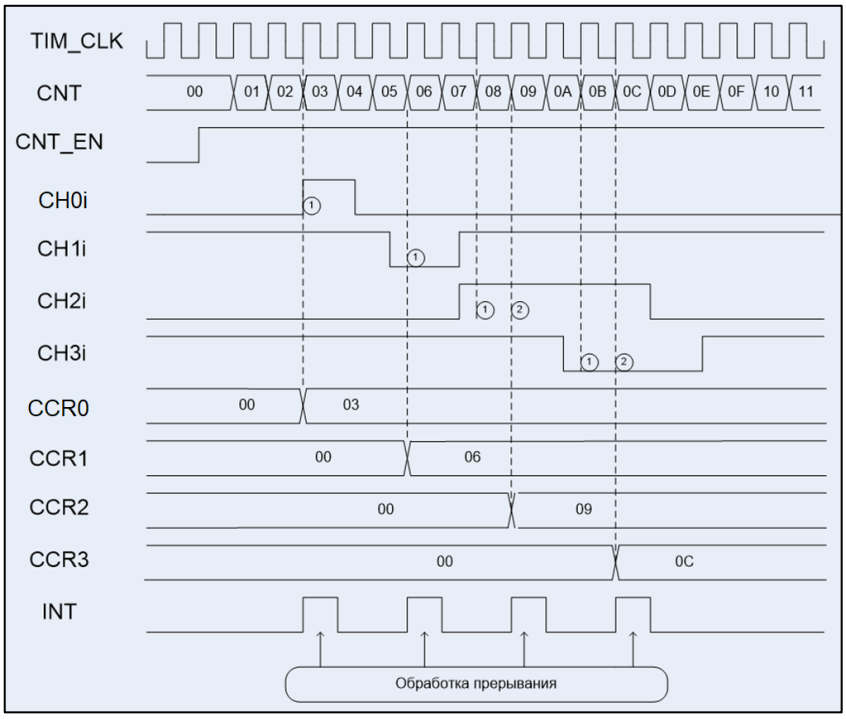


Рисунок 46 – Диаграммы примера работы в режиме захвата

9.9.3 Режим ШИМ

```
CMU->CFG8 &= ~0x00180000;      //Разрешение тактирования таймера 0 и 1
                               //(TIM_CLK = SOCCLK)
CMU->CFG1 |= 0x00080000;       //Разрешение работы таймера 0
GTIMER0->CNTRL = 0x00000000;   //Режим инициализации таймера
//Настраиваем работу основного счетчика
GTIMER0->CNT = 0x00000000;     //Начальное значение счетчика
GTIMER0->PSG = 0x00000000;     //Предделитель частоты TIM_CLK
GTIMER0->ARR = 0x00000010;     //Основание счета
GTIMER0->IE  = 0x00001E00;     //Разрешение генерировать прерывание по событию
                               //фронта на выходе REF для всех каналов
//Режим работы каналов – ШИМ
GTIMER0->CH0_CNTRL = 0x00000200;  //REF = 1, если CNT == CCR
GTIMER0->CH1_CNTRL = 0x00000200;  //REF = 1, если CNT == CCR
GTIMER0->CH2_CNTRL = 0x00000400;  //REF = 0, если CNT == CCR
GTIMER0->CH3_CNTRL = 0x00000600;  //Переключение REF, если CNT == CCR
//Режим работы выхода канала – канал работает на выход,
//на выходы канала выдается сигнал REF
GTIMER0->CH0_CNTRL1= 0x00000909;
GTIMER0->CH1_CNTRL1= 0x00000909;
GTIMER0->CH2_CNTRL1= 0x00000909;
GTIMER0->CH3_CNTRL1= 0x00000909;
//Установка значений CCR, с которыми сравнивается CNT при работе в режиме ШИМ
GTIMER0->CCR0 = 0x00000003;
GTIMER0->CCR1 = 0x00000006;
GTIMER0->CCR2 = 0x00000009;
GTIMER0->CCR3 = 0x0000000F;
//Разрешение работы таймера
GTIMER0->CNTRL = 0x00000001;   //Счет прямой по TIM_CLK
```

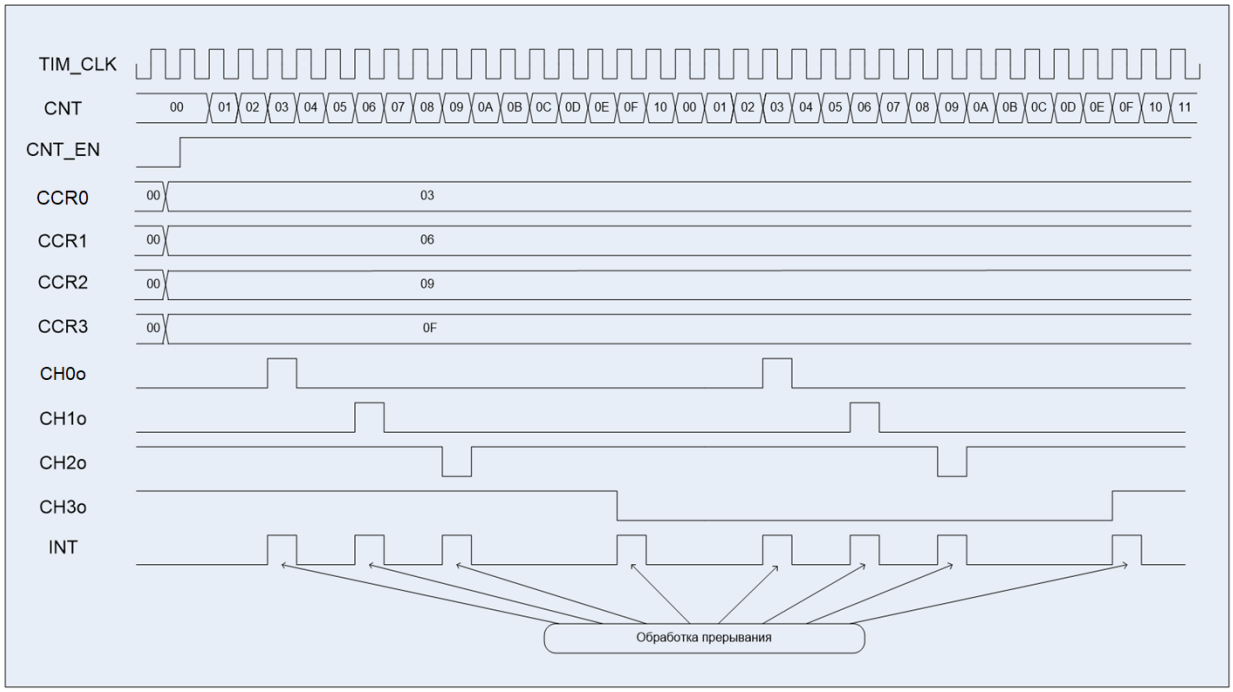


Рисунок 47 – Диаграммы примера работы в режиме ШИМ

9.10 Описание регистров блока таймера

Таблица 38 – Базовые адреса и смещения регистров управления таймера

Адрес	Название	Описание
0x8000_0400	GTIMER0	Контроллер GTIMER0
0x8000_0440	GTIMER1	Контроллер GTIMER1
Смещение		
0x00	CNT[31:0]	Основной счетчик таймера
0x01	PSG[31:0]	Делитель частоты TIM_CLK для тактирования основного счетчика
0x02	ARR[31:0]	Основание счета основного счетчика
0x03	CNTRL[15:0]	Регистр управления основного счетчика
0x14	BRKETR_CNTRL[15:0]	Регистр управления входом BRK и ETR
0x15	STATUS[31:0]	Регистр статуса таймера
0x16	IE[31:0]	Регистр разрешения прерывания таймера
0x17	DMA_RE[31:0]	Регистр разрешения формирования запроса 0 к DMA
0x20	DMA_RE1[31:0]	Регистр разрешения формирования запроса 1 к DMA
0x21	DMA_RE2[31:0]	Регистр разрешения формирования запроса 2 к DMA
0x22	DMA_RE3[31:0]	Регистр разрешения формирования запроса 3 к DMA
0x23	DMA_RE4[31:0]	Регистр разрешения формирования запроса 4 к DMA
Канал 0		
0x04	CCR0[31:0]	Регистр сравнения/захвата для 0 канала таймера
0x08	CH0_CNTRL[15:0]	Регистр управления для 0 канала таймера
0x0C	CH0_CNTRL1[15:0]	Регистр управления 1 для 0 канала таймера
0x10	CH0_DTG[15:0]	Регистр управления DTG для 0 канала таймера
0x18	CH0_CNTRL2[15:0]	Регистр управления 2 для 0 канала таймера
0x1C	CCR01[31:0]	Регистр сравнения/захвата 1 для 0 канала таймера
Канал 1		
0x05	CCR1[31:0]	Регистр сравнения/захвата для 1 канала таймера
0x09	CH1_CNTRL[15:0]	Регистр управления для 1 канала таймера
0x0D	CH1_CNTRL1[15:0]	Регистр управления 1 для 1 канала таймера
0x11	CH1_DTG[15:0]	Регистр управления DTG для 1 канала таймера
0x19	CH1_CNTRL2[15:0]	Регистр управления 2 для 1 канала таймера
0x1D	CCR11[31:0]	Регистр сравнения/захвата 1 для 1 канала таймера
Канал 2		
0x06	CCR2[31:0]	Регистр сравнения/захвата для 2 канала таймера
0x0A	CH2_CNTRL[15:0]	Регистр управления для 2 канала таймера
0x0E	CH2_CNTRL1[15:0]	Регистр управления 1 для 2 канала таймера
0x12	CH2_DTG[15:0]	Регистр управления DTG для 2 канала таймера
0x1A	CH2_CNTRL2[15:0]	Регистр управления 2 для 2 канала таймера
0x1E	CCR21[31:0]	Регистр сравнения/захвата 1 для 2 канала таймера
Канал 3		
0x07	CCR3[31:0]	Регистр сравнения/захвата для 3 канала таймера
0x0B	CH3_CNTRL[15:0]	Регистр управления для 3 канала таймера
0x0F	CH3_CNTRL1[15:0]	Регистр управления 1 для 3 канала таймера

Адрес	Название	Описание
0x8000_0400	GTIMER0	Контроллер GTIMER0
0x8000_0440	GTIMER1	Контроллер GTIMER1
Смещение		
0x13	CH3_DTG[15:0]	Регистр управления DTG для 3 канала таймера
0x1B	CH3_CNTRL2[15:0]	Регистр управления 2 для 3 канала таймера
0x1F	CCR31[31:0]	Регистр сравнения/захвата 1 для 3 канала таймера

9.10.1 CNT

Таблица 39 – Основной счетчик таймера CNT

Номер	31…0
Доступ	R/W
Сброс	0
	CNT[31:0]

Таблица 40 – Описание бит регистра CNT

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	CNT[31:0]	Значение основного счетчика таймера

9.10.2 PSG

Таблица 41 – Делитель частоты TIM_CLK для счета основного счетчика PSG

Номер	31...0
Доступ	R/W
Сброс	0
	PSG[31:0]

Таблица 42 – Описание бит регистра PSG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	PSG[31:0]	Значение предварительного делителя счетчика. Основной счетчик считает на частоте: TIM_CLKd = TIM_CLK/(PSG+1)

9.10.3 ARR

Таблица 43 – Основание счета основного счетчика ARR

Номер	31…0
Доступ	R/W
Сброс	0
	ARR[31:0]

Таблица 44 – Описание бит регистра ARR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	ARR[31:0]	Основание счета для основного счетчика. CNT = [0…ARR]

9.10.4 CNTRL

Таблица 45 – Регистр управления основным счетчиком CNTRL

Номер	31…12	11…8
Доступ	U	R/W
Сброс	0	0
	-	EVENT_SEL[3:0]

Номер	7…6	5…9	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	00	00	0	0	0	0
	CNT_MODE[1:0]	FDTS[1:0]	DIR	WR_CMPL	ARRB_EN	CNT_EN

Таблица 46 – Описание бит регистра CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
31…12	-	Зарезервировано
11...8	EVENT_SEL[3:0]	Биты выбора источника событий: 0000 – внутренняя тактовая частота TIM_CLKd (формируется путем деления частоты TIM_CLK); 0001 – CNT == ARR в таймере 0; 0010 – CNT == ARR в таймере 1; 0011 – зарезервировано; 0100 – событие фронта на канале 0, «Режим 1»; 0101 – событие фронта на канале 1, «Режим 1»; 0110 – событие фронта на канале 2, «Режим 1»; 0111 – событие фронта на канале 3, «Режим 1»; 1000 – событие фронта на ETR, «Режим 2»; 1001 – событие среза на ETR, «Режим 2»; 1010-1111 – зарезервировано
7..6	CNT_MODE[1:0]	Режим счета основного счетчика: 00 – счетчик прямой при DIR = 0; счетчик обратный при DIR = 1; 01 – счетчик двунаправленный с автоматическим изменением DIR при CNT == 0 или CNT == ARR; 10 – счетчик прямой при DIR = 0; счетчик обратный при DIR = 1; 11 – зарезервировано.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
		Режим счета CNT_MODE[1:0] необходимо устанавливать в соответствии со значением в поле EVENT_SEL[3:0]: – EVENT_SEL[3:0] = 0000: CNT_MODE[1:0] = 00 или 01; – EVENT_SEL[3:0] != 0000: CNT_MODE[1:0] = 10
5...4	FDTS[1:0]	Делитель тактовой частоты F_{DTS}: 00 – F_{DTS} = TIM_CLK; 01 – F_{DTS} = TIM_CLK/2; 10 – F_{DTS} = TIM_CLK/3; 11 – F_{DTS} = TIM_CLK/4
3	DIR	Направление счета основного счетчика: 0 – прямой, от 0 до ARR; 1 – обратный, от ARR до 0
2	WR_CMPL	Флаг выполнения записи нового значения в регистры CNT, PSG и ARR: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
1	ARRB_EN	Режим обновления регистра ARR: 0 – ARR будет перезаписан в момент записи в ARR; 1 – ARR будет перезаписан при CNT == ARR
0	CNT_EN	Разрешение работы таймера: 0 – таймер отключен; 1 – таймер включен

9.10.5 CCRy

Таблица 47 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy

Номер	31…0
Доступ	R/W
Сброс	0
	CCR[31:0]

Таблица 48 – Описание бит регистра CCRy

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	CCR[31:0]	Режим захвата: значение CNT, при котором произошел факт захвата события. Режим ШИМ: значение CCR, с которым сравнивается CNT

9.10.6 CCRy1

Таблица 49 – Регистр сравнения/захвата для ‘y’ канала таймера CCRy1

Номер	31…0
Доступ	R/W
Сброс	0
	CCR1[31:0]

Таблица 50 – Описание бит регистра CCRy1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…0	CCR1[31:0]	Режим захвата: значение CNT, при котором произошел факт захвата события. Режим ШИМ: значение CCR1, с которым сравнивается CNT

9.10.7 CHy_CNTRL

Таблица 51 – Регистр управления для ‘y’ канала таймера CHy_CNTRL

Номер	31…17	16	15	14	13
Доступ	U	RO	R/W	RO	R/W
Сброс	0	0	0	0	0
	-	WR_CMPL1	CAP_NPWM	WR_CMPL	ETREN

Номер	12	11…9	8	7…6	5…4	3…0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	000	0	00	00	0000
	BRKEN	OCCM[2:0]	OCCE	CHPSC[1:0]	CHSEL[1:0]	CHFLTR[3:0]

Таблица 52 – Описание бит регистра CHy_CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано.
16	WR_CMPL1	Флаг выполнения записи нового значения в регистр CCRy1: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
15	CAP_NPWM	Режим работы канала: 0 – канал работает в режиме ШИМ; 1 – канал работает в режиме захвата
14	WR_CMPL	Флаг выполнения записи нового значения в регистр CCRy: 0 – новые данные можно записывать; 1 – данные не записаны и идет запись
13	ETREN	Разрешение сброса сигнала REF в «0» при высоком уровне на входе ETR: 0 – запрещен; 1 – разрешен
12	BRKEN	Разрешение сброса сигналов REF и DTG в «0» при низком уровне на входе BRK: 0 – запрещен; 1 – разрешен
11…9	OCCM[2:0]	Формат выработки сигнала REF в режиме ШИМ: Если CCR1_EN=0: 000 – всегда 0; 001 – 1, если CNT==CCR; 010 – 0, если CNT==CCR;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		011 – переключение REF, если CNT==CCR; 100 – всегда 0; 101 – всегда 1; 110 – 1, если DIR=0 (счет прямой), CNT<CCR, иначе 0; 0, если DIR=1 (счет обратный), CNT>CCR, иначе 1; 111 – 0, если DIR=0 (счет прямой), CNT<CCR, иначе 1; 1, если DIR=1 (счет обратный), CNT>CCR, иначе 0. Если CCR1_EN=1: 000 – всегда 0; 001 – 1, если CNT==CCR или CNT==CCR1; 010 – 0, если CNT==CCR или CNT==CCR1; 011 – переключение REF, если CNT==CCR или CNT==CCR1; 100 – всегда 0; 101 – всегда 1; 110 – 0, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 1; 0, если DIR=1 (счет обратный), CCR<CNT<CCR1, иначе 1; 111 – 1, если DIR=0 (счет прямой), CCR≤CNT≤CCR1, иначе 0; 1, если DIR=1 (счет обратный), CCR<CN <CCR1, иначе 0; Необходимо соблюдать условие CCR<CCR1
8	OCCE	Разрешение работы ETR: 0 – запрещен; 1 – разрешен
7…6	CHPSC[1:0]	Предварительный делитель входного канала: 00 – нет деления; 01 – /2; 10 – /4; 11 – /8
5…4	CHSEL[1:0]	Выбор события по входному каналу CHyi для фиксации значения основного счетчика (регистр CNT) в регистр CCRy: 00 – событие фронта на входном канале CHyi; 01 – событие среза на входном канале CHyi; 10 – событие фронта от других каналов: для канала 0 от канала 1; для канала 1 от канала 2; для канала 2 от канала 3; для канала 3 от канала 0; 11 – событие фронта от других каналов: для канала 0 от канала 2; для канала 1 от канала 3; для канала 2 от канала 0; для канала 3 от канала 1
3…0	CHFLTR[3:0]	Конфигурация фильтра на входе канала ‘y’. Выбор частоты выборки Fs и количества выборок N: 0000 – нет фильтрации, $F_S = F_{DTS}$; 0001 – F_S = TIM_CLK, N = 2;

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		0010 – F_S = TIM_CLK, N = 4; 0011 – F_S = TIM_CLK, N = 8; 0100 – F_S = F_{DTS}/2, N = 6; 0101 – F_S = F_{DTS}/2, N = 8; 0110 – F_S = F_{DTS}/4, N = 6; 0111 – F_S = F_{DTS}/4, N = 8; 1000 – F_S = F_{DTS}/8, N = 6; 1001 – F_S = F_{DTS}/8, N = 8; 1010 – F_S = F_{DTS}/16, N = 5; 1011 – F_S = F_{DTS}/16, N = 6; 1100 – F_S = F_{DTS}/16, N = 8; 1101 – F_S = F_{DTS}/32, N = 5; 1110 – F_S = F_{DTS}/32, N = 6; 1111 – F_S = F_{DTS}/32, N = 8

9.10.8 CHy_CNTRL1

Таблица 53 – Регистр управления 1 для ‘y’ канала таймера CHy_CNTRL1

Номер	31…13	12	11…10	9…8	7…5	4	3…2	1…0
Доступ	U	R/W	R/W	R/W	U	R/W	R/W	R/W
Сброс	0	0	00	00	0	0	00	00
	-	NINV	NSELO [1:0]	NSELOE [1:0]	-	INV	SELO [1:0]	SELOE [1:0]

Таблица 54 – Описание бит регистра CHy_CNTRL1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
31...13	-	Зарезервировано
12	NINV	Инверсия инверсного выхода nCHy: 0 – выход не инвертируется; 1 – выход инвертируется
11…10	NSELO[1:0]	Выбор источника сигнала для инверсного выхода nCHy: 00 – на nCHyo выдается 0; 01 – на nCHyo выдается 1; 10 – на nCHyo выдается сигнал nREF; 11 – на nCHyo выдается сигнал с DTG
9…8	NSELOE[1:0]	Режим работы инверсного выхода nCHy: 00 – на nCHyoe выдается 0; 01 – на nCHyoe выдается 1; 10 – на nCHyoe выдается сигнал nREF; 11 – на nCHyoe выдается сигнал с DTG. При nCHyoe = 0 вывод канала в третьем состоянии, при nCHyoe = 1 вывод канала работает в режиме выхода
7…5	-	Зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений.
4	INV	Инверсия прямого выхода CHy: 0 – выход не инвертируется; 1 – выход инвертируется
3…2	SELO[1:0]	Выбор источника сигнала для прямого выхода CHy: 00 – на CHyo выдается 0; 01 – на CHyo выдается 1; 10 – на CHyo выдается сигнал REF; 11 – на CHyo выдается сигнал с DTG
1…0	SELOE[1:0]	Режим работы прямого выхода CHy: 00 – на CHyoe выдается 0; 01 – на CHyoe выдается 1; 10 – на CHyoe выдается сигнал REF; 11 – на CHyoe выдается сигнал с DTG. При CHyoe = 0 вывод канала в третьем состоянии, при CHyoe = 1 вывод канала работает в режиме выхода

9.10.9 CHy_CNTRL2

Таблица 55 – Регистр управления 2 для ‘y’ канала таймера CHy_CNTRL2

Номер	31…5	4	3	2	1…0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	00
	-	EV_DELAY	CCRRLD	CCR1_EN	CHSEL[1:0]

Таблица 56 – Описание бит регистра CHy_CNTRL2

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...5	-	Зарезервировано
4	EV_DELAY	Задержка события захвата до обновления регистров CCRy и CCRy1: 0 – сигнал события захвата устанавливается в момент обнаружения события, при этом обновление регистров CCRy и CCRy1 выполняется через один такт TIM_CLK; 1 – сигнал события захвата устанавливается синхронно с обновлением информации в регистрах CCRy и CCRy1
3	CCRRLD	Режим обновления регистров CCRy и CCRy1: 0 – обновление возможно в любой момент времени; 1 – обновление будет осуществлено только при CNT == 0
2	CCR1_EN	Разрешение работы регистра CCRy1: 0 – CCRy1 не используется; 1 – CCRy1 используется

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1…0	CHSEL[1:0]	Выбор события по входному каналу CHyi для фиксации значения основного счетчика (регистр CNT) в регистр CCRy1: 00 – событие фронта на входном канале CHyi; 01 – событие среза на входном канале CHyi; 10 – событие среза от других каналов: для канала 0 от канала 1; для канала 1 от канала 2; для канала 2 от канала 3; для канала 3 от канала 0; 11 – событие среза от других каналов: для канала 0 от канала 2; для канала 1 от канала 3; для канала 2 от канала 0; для канала 3 от канала 1

9.10.10 CHy_DTG

Таблица 57 – Регистр управления генератором «мертвой зоны» CHy_DTG

Номер	31…16	15…8	7…5	4	3…0
Доступ	U	R/W	U	R/W	R/W
Сброс	0	00000000	000	0	0000
	-	DTG[7:0]	-	EDTS	DTGx[3:0]

Таблица 58 – Описание бит регистра CHy_DTG

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…16	-	Зарезервировано
15…8	DTG[7:0]	Основной делитель частоты DTG. Задержка DTGdel = DTG • (DTGx + 1)
7…5	-	Зарезервировано
4	EDTS	Частота работы DTG: 0 – TIM_CLK; 1 – F_{DTS}
3…0	DTGx[3:0]	Предварительный делитель частоты DTG

9.10.11 BRKETR_CNTRL

Таблица 59 – Регистр BRKETR_CNTRL управления входом BRK и ETR

Номер	31…8	7…4	3…2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0000	00	0	0
	-	ETR_FILTER[3:0]	ETR_PSC[1:0]	ETR_INV	BRK_INV

Таблица 60 – Описание бит регистра BRKETR_CNTRL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...8	-	Зарезервировано
7…4	ETR_FILTER[3:0]	Конфигурация фильтра на входе ETR. Выбор частоты выборки Fs и количества выборок N: 0000 – нет фильтрации, $F_S = F_{DTS}$; 0001 – F_S = TIM_CLK, N = 2; 0010 – F_S = TIM_CLK, N = 4; 0011 – F_S = TIM_CLK, N = 8; 0100 – $F_S = F_{DTS}/2$, N = 6; 0101 – $F_S = F_{DTS}/2$, N = 8; 0110 – $F_S = F_{DTS}/4$, N = 6; 0111 – $F_S = F_{DTS}/4$, N = 8; 1000 – $F_S = F_{DTS}/8$, N = 6; 1001 – $F_S = F_{DTS}/8$, N = 8; 1010 – $F_S = F_{DTS}/16$, N = 5; 1011 – $F_S = F_{DTS}/16$, N = 6; 1100 – $F_S = F_{DTS}/16$, N = 8; 1101 – $F_S = F_{DTS}/32$, N = 5; 1110 – $F_S = F_{DTS}/32$, N = 6; 1111 – $F_S = F_{DTS}/32$, N = 8
3…2	ETR_PSC[1:0]	Асинхронный предделитель частоты со входа ETR: 00 – без деления; 01 – /2; 10 – /4; 11 – /8
1	ETR_INV	Инверсия входа ETR: 0 – без инверсии; 1 – инверсия
0	BRK_INV	Инверсия входа BRK: 0 – без инверсии; 1 – инверсия

9.10.12 STATUS

Таблица 61 – Регистр статуса таймера STATUS

Номер	31...17	16…13	12…9	8…5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT[3:0]	CCR REF EVENT[3:0]	CCR CAP EVENT[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT	ETR FE EVENT	ETR RE EVENT	CNT ARR EVENT	CNT ZERO EVENT

Таблица 62 – Описание бит регистра STATUS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…17	-	Зарезервировано.
16…13	CCR CAP1 EVENT[3:0]	Событие записи значения счетчика CNT в регистр CCRy1 по захвату настроенного фронта (фронт или срез) на входе канала CHyi: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – канал 0; Бит 3 – канал 3
12…9	CCR REF EVENT[3:0]	Событие фронта на выходе генератора опорного сигнала REF: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – канал 0; Бит 3 – канал 3
8…5	CCR CAP EVENT[3:0]	Событие записи значения счетчика CNT в регистр CCRy по захвату настроенного фронта (фронт или срез) на входе канала CHyi: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события. Бит 0 – канал 0; Бит 3 – канал 3
4	BRK EVENT	Событие высокого уровня на входе BRK: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», при условии наличия низкого уровня на входе BRK
3	ETR FE EVENT	Событие среза на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события
2	ETR RE EVENT	Событие фронта на входе ETR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием, приоритет у нового события

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
1	CNT ARR EVENT	Событие совпадения CNT с ARR: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события. Если с момента совпадения до момента программного сброса флага регистры CNT и ARR не изменили состояния, то флаг повторно не взводится
0	CNT ZERO EVENT	Событие совпадения CNT с нулем: 0 – нет события; 1 – есть событие. Сбрасывается записью «0», если запись происходит одновременно с новым событием совпадения, приоритет у нового события. Если с момента совпадения до момента программного сброса флага регистр CNT не изменил состояния, то флаг повторно не взводится

9.10.13 IE

Таблица 63 – Регистр разрешения прерываний таймера IE

Номер	31...17	16…13	12…9	8…5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT IE[3:0]	CCR REF EVENT IE[3:0]	CCR CAP EVENT IE[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT IE	ETR FE EVENT IE	ETR RE EVENT IE	CNT ARR EVENT IE	CNT ZERO EVENT IE

Таблица 64 – Описание бит регистра IE

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…17	-	Зарезервировано.
16...13	CCR CAP1 EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR CAP1 EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – канал 0; Бит 3 – канал 3

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
12…9	CCR REF EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR REF EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – канал 0; Бит 3 – канал 3
8…5	CCR CAP EVENT IE[3:0]	Флаг разрешения прерывания по событию CCR CAP EVENT[3:0] в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено. Бит 0 – канал 0; Бит 3 – канал 3
4	BRK EVENT IE	Флаг разрешения прерывания по событию BRK EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
3	ETR FE EVENT IE	Флаг разрешения прерывания по событию ETR FE EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
2	ETR RE EVENT IE	Флаг разрешения прерывания по событию ETR RE EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
1	CNT ARR EVENT IE	Флаг разрешения прерывания по событию CNT ARR EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено
0	CNT ZERO EVENT IE	Флаг разрешения прерывания по событию CNT ZERO EVENT в регистре STATUS: 0 – прерывание запрещено; 1 – прерывание разрешено

9.10.14 DMA_RE, DMA_RE1-DMA_RE4

Таблица 65 – Регистры DMA_RE, DMA_RE1-DMA_RE4 разрешения запросов DMA

Номер	31...17	16…13	12…9	8…5
Доступ	U	R/W	R/W	R/W
Сброс	0	0	0	0
	-	CCR CAP1 EVENT RE[3:0]	CCR REF EVENT RE[3:0]	CCR CAP EVENT RE[3:0]

Номер	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	BRK EVENT RE	ETR FE EVENT RE	ETR RE EVENT RE	CNT ARR EVENT RE	CNT ZERO EVENT RE

Таблица 66 – Описание бит регистров DMA_RE, DMA_RE1-DMA_RE4

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...17	-	Зарезервировано.
16…13	CCR CAP1 EVENT RE [3:0]	Флаг разрешения запроса DMA по событию CCR CAP1 EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – канал 0; Бит 3 – канал 3
12…9	CCR REF EVENT RE[3:0]	Флаг разрешения запроса DMA по событию CCR REF EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – канал 0; Бит 3 – канал 3
8…5	CCR CAP EVENT RE [3:0]	Флаг разрешения запроса DMA по событию CCR CAP EVENT[3:0]: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен. Бит 0 – канал 0; Бит 3 – канал 3
4	BRK EVENT RE	Флаг разрешения запроса DMA по событию BRK EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
3	ETR FE EVENT RE	Флаг разрешения запроса DMA по событию ETR FE EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
2	ETR RE EVENT RE	Флаг разрешения запроса DMA по событию ETR RE EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
1	CNT ARR EVENT RE	Флаг разрешения запроса DMA по событию CNT ARR EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен
0	CNT ZERO EVENT RE	Флаг разрешения запроса DMA по событию CNT ZERO EVENT: 0 – запрос DMA запрещен; 1 – запрос DMA разрешен

10 Порты общего назначения GPIO

Большинство внешних выводов процессора представляют собой универсальные входы-выходы, которые могут быть перепрограммированы пользователем. Каждый бит портов GPIO программируется индивидуально.

Процессор имеет несколько портов ввода-вывода общего назначения. Часть из них имеет альтернативные функции ввода-вывода и может быть использована различными интерфейсами процессора. Процессор имеет 128 выводов общего назначения, которые могут быть индивидуально сконфигурированы как вход или выход. Также дополнительно имеется 22 вывода общего назначения, которые всегда сконфигурированы как выход.

10.1 Порты PA, PB, PC

Специальный модуль GPIO объединяет три самостоятельных 32-разрядных порта. Регистры этого модуля подключены к шине периферийных устройств и имеют базовый адрес 0x80001000 (порт A), адрес 0x80001040 (порт B), адрес 0x80001080 (порт C). Некоторые регистры имеют четыре адреса для работы: загрузки, установки, сброса и инверсии бит (суффиксы _LD, _SET, _CLR и _INV, соответственно). Краткое описание регистров порта приведено в таблице 67. После сброса значения регистров PiDDR, PiIMR, PiALT и PiPUR равно нулю. Значения всех остальных регистров после сброса не определено.

Для микросхем с ревизии 3

Регистр портов PiDS управляет силой драйвера соответствующих выводов каждого порта. Значение 0 устанавливает силу драйвера 14 мА, значение 1 – 32 мА. По сбросу устанавливается значение 1.

Для микросхем ревизии 2

Примечание – Выводы порта C PC[18], PC[19] и PC[20], работающие только как выходы, будут по сбросу выдавать содержимое регистра данных, значение которого не определено ().

Таблица 67 – Регистры управление портами PA, PB, PC

Номер	Имя	Тип	Назначение
0	PiDR_LD	R/W	Регистр данных для выдачи информации. Загрузка значения
1	PiDR_SET	R/W	Регистр данных для выдачи информации: 1 – установка регистра в 1; 0 – значение регистра не меняется; Чтение регистра возвращает значение регистра PIDR
2	PiDR_CLR	R/W	Регистр данных для выдачи информации: 1 – установка регистра в 0; 0 – значение регистра не меняется; Чтение регистра возвращает значение регистра PIDR
3	PiDR_INV	R/W	Регистр данных для выдачи информации: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется; Чтение регистра возвращает значение регистра PIDR

Номер	Имя	Тип	Назначение
4	PiDDR_LD	R/W	Регистр направления выдачи информации. Загрузка значения: 0 – прием; 1 – выдача.
5	PiDDR_SET	R/W	Регистр направления выдачи информации: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PIDDR
6	PiDDR_CLR	R/W	Регистр направления выдачи информации: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PIDDR
7	PiDDR_INV	R/W	Регистр направления выдачи информации: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PIDDR
8	PiPEIE_LD	R/W	Регистр разрешения прерывания по положительному (из 0 в 1) изменению внешней линии порта. Загрузка значения: 1 – прерывание по изменению из 0 в 1 разрешено; 0 – прерывание по изменению из 0 в 1 запрещено
9	PiPEIE_SET	R/W	Регистр разрешения прерывания по положительному (из 0 в 1) изменению внешней линии порта: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPEIE
10	PiPEIE_CLR	R/W	Регистр разрешения прерывания по положительному (из 0 в 1) изменению внешней линии порта: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPEIE
11	PiPEIE_INV	R/W	Регистр разрешения прерывания по положительному (из 0 в 1) изменению внешней линии порта: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPEIE
12	PiNEIE_LD	R/W	Регистр разрешения прерывания по отрицательному (из 1 в 0) изменению внешней линии порта. Загрузка значения: 1 – прерывание по изменению из 1 в 0 разрешено; 0 – прерывание по изменению из 1 в 0 запрещено
13	PiNEIE_SET	R/W	Регистр разрешения прерывания по отрицательному (из 1 в 0) изменению внешней линии порта: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNEIE

Номер	Имя	Тип	Назначение
14	PiNEIE_CLR	R/W	Регистр разрешения прерывания по отрицательному (из 1 в 0) изменению внешней линии порта: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNEIE
15	PiNEIE_INV	R/W	Регистр разрешения прерывания по отрицательному (из 1 в 0) изменению внешней линии порта: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNEIE
16	PiINVR_LD	R/W	Регистр инверсного значения внешней линии порта, при котором регистрируется прерывание. Загрузка значения 1 – прерывание регистрируется при значении внешней линии 0; 0 – прерывание регистрируется при значении внешней линии 1
17	PiINVR_SET	R/W	Регистр инверсного значения внешней линии порта, при котором регистрируется прерывание: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNVR
18	PiINVR_CLR	R/W	Регистр инверсного значения внешней линии порта, при котором регистрируется прерывание: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNVR
19	PiINVR_INV	R/W	Регистр инверсного значения внешней линии порта, при котором регистрируется прерывание: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiNVR
20	PiIMR_LD	R/W	Регистр маскирования прерывания внешней линии порта. Загрузка значения: 1 – прерывание разрешено; 0 – прерывание запрещено
21	PiIMR_SET	R/W	Регистр маскирования прерывания внешней линии порта: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiIMR
22	PiIMR_CLR	R/W	Регистр маскирования прерывания внешней линии порта: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiIMR
23	PiIMR_INV	R/W	Регистр маскирования прерывания внешней линии порта: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется Чтение регистра возвращает значение регистра PiIMR

Номер	Имя	Тип	Назначение
24	PiALT_LD	R/W	Регистр выбора функции порта. Загрузка значения: 1 – альтернативная функция вывода включена; 0 – вывод общего назначения
25	PiALT _SET	R/W	Выбор функции порта: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiALT
26	PiALT _CLR	R/W	Регистр выбора функции порта: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiALT
27	PiALT _INV	R/W	Регистр выбора функции порта: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiALT
28	PiPUR_LD	R/W	Регистр включения резистора доопределения до «питания» или до «земли» на внешней линии порта. Загрузка значения: 0 – доопределение включено; 1 – доопределение выключено
29	PiPUR_SET	R/W	Регистр включения резистора доопределения до «питания» или до «земли» на внешней линии порта: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPUR
30	PiPUR_CLR	R/W	Регистр включения резистора доопределения до «питания» или до «земли» на внешней линии порта: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPUR
31	PiPUR_INV	R/W	Регистр включения резистора доопределения до «питания» или до «земли» на внешней линии порта: 1 – инверсия содержимого регистра; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PiPUR
32	PiPXD	R	Значение внешней линии порта
33	PiINTREQ	R	Запросы прерывания по входам порта
34	PiDS	R/W	Регистр управления силой выходного драйвера (для микросхем с ревизии 3): 1 – 32 мА (значение по сбросу); 0 – 14 мА
35	PiECLR	R/W	Сброс запросов прерываний по фронту (запись 1 вызывает сброс)
Примечание – Порты A, B и C имеют идентичные регистры. Символом «i» в имени регистра обозначено одно из трех значений: A, B, C			

Каждая линия 32-разрядного порта может быть сконфигурирована как выход или как вход, может сгенерировать прерывание по фронту и/или срезу сигнала или по уровню (выбор уровня задается битом инверсии входа). Все запросы прерываний собираются в один общий запрос, который поступает в контроллер прерывания.

Каждый бит порта может иметь альтернативную функцию. Если бит регистра PiALT имеет нулевое значение, соответствующая линия порта работает под управлением регистров порта. Если бит равен единице, линия порта переходит под управление заданного устройства процессора. Возможные альтернативные функции портов приведены в таблице описания выводов.

Виды прерываний от внешних линий портов разрешаются/запрещаются независимо регистрами PiPEIE и PiNEIE. Если в любом из регистров PiPEIE или PiNEIE установлена «1», будут разрешены соответствующее прерывание по переходу из «0» в «1» (фронт), или из «1» в «0» (срез), или оба. Разрешенное прерывание по фронту или срезу запрещает прерывание по уровню. Итоговое прерывание может быть заблокировано значением «0» бит регистра маски PiIMR.

10.2 Порты PE и PD

Порты PE и PD предназначены для организации интерфейса к внешней памяти. Если нет необходимости подключать внешнюю память к данному интерфейсу, тогда шина адреса и шина данных могут быть использованы как порты общего назначения. При этом порт PE представляет собой 22-разрядный однонаправленный выходной порт, а порт PD – 32-разрядный двунаправленный порт общего назначения. Порт PE имеет только регистр данных. Порт PD имеет регистр данных и регистр направления. Регистры портов подключены к шине обмена периферийных устройств и имеют базовый адрес 0x8000_0090. Регистры портов представлены в таблице 68.

Таблица 68 – Регистры управления портами PE, PD

Номер	Имя	Тип	Описание
0	PD_LOAD	R/W	Регистр данных порта PD. Загрузка всех 32 бит
1	PD_SET	R/W	Регистр данных порта PD: 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PD
2	PD_CLEAR	R/W	Регистр данных порта PD: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PD
3	PD_DIR	R/W	Регистр направления порта PD: 1 – выдача данных; 0 – прием данных
4	PE_LOAD	R/W	Регистр данных порта PE. Загрузка всех 22 бит
5	PE_SET	R/W	Регистр данных порта PE; 1 – установка регистра в 1; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PE

Номер	Имя	Тип	Описание
6	PE_CLEAR	R/W	Регистр данных порта PE: 1 – установка регистра в 0; 0 – значение регистра не меняется. Чтение регистра возвращает значение регистра PE
7	PE_DS	R/W	Регистр управления силой драйвера шины ADDR (для микросхем с ревизии 3): 1 – 32 мА (значение по сбросу); 0 – 14 мА
8	PX_ALT	R/W	Управление альтернативными функциями портов PE и PD: 1 – альтернативная функция включена; 0 – альтернативная функция выключена
9	PD_PIN	R	Чтение внешних контактов порта PD
10	PD_PU	R/W	Управление резисторами доопределения до питания внешних выводов PD[31:0]: 0 – резистор подключен; 1 – резистор отключен
11	PD_DS	R/W	Регистр управления силой драйвера шины DATA (для микросхем с ревизии 3): 1 – 32 мА (значение по сбросу); 0 – 14 мА

10.3 Регистр PX_ALT

Данный регистр задает различные конфигурации для выводов портов PE, PD. Назначение бит регистра приведено в таблице 69.

Таблица 69 – Регистр управления альтернативными функциями

Бит	Имя	Тип	Назначение
0	PDB[0]	R/W	1 – порт PD[7:0] находится под управлением контроллера внешней памяти; 0 – порт PD[7:0] находится под управлением PD_LOAD[7:0], PD_SET[7:0], PD_CLEAR[7:0] и PD_DIR[7:0], т.е. порт общего назначения.
1	PDB[1]	R/W	1 – порт PD[15:8] находится под управлением контроллера внешней памяти; 0 – порт PD[15:8] находится под управлением PD_LOAD[15:8], PD_SET[15:8], PD_CLEAR[15:8] и PD_DIR[15:8], т.е. порт общего назначения.
2	PDB[2]	R/W	1 – порт PD[23:16] находится под управлением контроллера внешней памяти; 0 – порт PD[23:16] находится под управлением PD_LOAD[23:16], PD_SET[23:16], PD_CLEAR[23:16] и PD_DIR[23:16], т.е. порт общего назначения.

Бит	Имя	Тип	Назначение
3	PDB[3]	R/W	1 – порт PD[31:24] находится под управлением контроллера внешней памяти; 0 – порт PD[31:24] находится под управлением PD_LOAD[31:24], PD_SET[31:24], PD_CLEAR[31:24] и PD_DIR[31:24], т.е. порт общего назначения.
4	PEB[0]	R/W	1 – порт PE[7:0] находится под управлением внешнего контроллера внешней памяти; 0 – порт PE[7:0] находится под управлением PE_LOAD[7:0], PE_SET[7:0], PE_CLEAR[7:0], т.е. порт общего назначения.
5	PEB[1]	R/W	1 – порт PE[15:8] находится под управлением внешнего контроллера внешней памяти; 0 – порт PE[15:8] находится под управлением PE_LOAD[15:8], PE_SET[15:8], PE_CLEAR[15:8], т.е. порт общего назначения.
6	PEB[2]	R/W	1 – порт PE[21:16] находится под управлением внешнего контроллера внешней памяти; 0 – порт PE[21:16] находится под управлением PE_LOAD[21:16], PE_SET[2:16], PE_CLEAR[21:16], т.е. порт общего назначения.
7	PDXF0	R/W	Управление подключением периферийных устройств (таблица 70)
8	PDXF1	R/W	Управление подключением периферийных устройств (таблица 70)
31-9	-	-	Не используются

Конфигурация для разрядов с нулевого по 31-й порта PD следующая: если биты PDB[3:0] равны нулю – это порт общего назначения (каждый байт порта управляется своим битом), а если какой-то бит PDB[3:0] равен единице – соответствующие ему разряды порта выполняют альтернативную функцию. При этом для двух младших байт (биты с нулевого по 15-й) альтернативной функцией является только функция внешней шины данных для подключения внешней памяти. *Биты PDB[2] и PDB[3] управляют различными данными, но их значение всегда должно быть одинаковыми (два нуля или две единицы).*

Функции для разрядов с 16-го по 31-й порта PD более сложные и дополнительно зависят от битов 7 и 8 регистра PX_ALT. Возможные конфигурации для старших 16 битов приведены в таблице 70.

Таблица 70 – Функции порта PD[31:16] (биты PDB[3:2]=11b)

Номер	PX_ALT[8:7]	Функция
16-23	x0	Шина данных внешней памяти. Вход-выход. Биты с 16-го по 23-й
	01	NAND флэш-память. Шина данных. Вход-выход. Биты с 0-го по 7-й
	11	МКПД-интерфейс: Входы – 16, 17, 21, 22; Выходы – 18, 19, 20, 23

Номер	PX_ALT[8:7]	Функция
24-30	x0	Шина данных внешней памяти. Вход-выход. Биты с 24-го по 30-й
	01	NAND флэш-память. Шина управления. Выходы
	11	МКПД-интерфейс: Выходы – 24, 25
31	x0	Шина данных внешней памяти. Вход-выход. Бит 31-й
	x1	Вход. Для контроллера NAND флэш-памяти – вход готовности

После сброса регистр альтернативных функций равен нулю и порты PE, PD являются портами общего назначения. При этом состояние портов следующее:

– порт PD находится в отключенном состоянии (вход);

– значение порта PE равно нулю.

При необходимости использования внешней памяти необходимо записать в регистр PX_ALT нужную конфигурацию (задать разрядность шины данных и шины адреса).

При необходимости подключения внешней NAND-памяти или при использовании интерфейса МКПД, также в регистре PX_ALT необходимо задать соответствующую конфигурацию. При использовании данных контроллеров шина данных может иметь максимальную разрядность 16 бит.

Включение режима 16-разрядной внешней шины данных происходит посредством установки бита SYSCON[19] регистра управления. Установка данного бита оказывает влияние только на контроллер внешней шины, т.к. в случае 16-разрядной шины контроллер будет выполнять два последовательных чтения (или две записи) при обращении к внешней памяти. При этом конфигурация шины данных и шины адреса не зависит от бита SYSCON[19]. Можно, например, использовать только восемь младших разрядов шины данных и работать с ними как с 32-разрядной шиной.

10.4 Регистр FLAGREG

В архитектуре вычислительного ядра процессора предусмотрен специальный регистр управления флагами FLAGREG, который осуществляет управление специальными внешними контактами. Если для бит с нулевого по третий порта PC задать альтернативную функцию, выводы PC[3:0] переходят под управление регистра FLAGREG и начинают выполнять функции флагов FLAG3–0.

Каждый контакт может быть сконфигурирован как вход или выход индивидуально. При конфигурации как выход, программы могут использовать выводы флагов для сигналов событий или условий для любого внешнего устройства. При конфигурации как входы, программы могут считывать состояние внешних контактов, используя регистр FLAGREG или использовать значение входа как условие в условных командах.

Для подготовки и использования флагов следует использовать следующую процедуру:

1 С помощью битов FLAGx_EN регистра FLAGREG сконфигурируйте каждый из выводов FLAG3–0 как вход (FLAGx_EN=0) или выход (FLAGx_EN=1).

2 С помощью битов FLAGx_OUT регистра FLAGREG выберите значения выхода для каждого вывода FLAG3–0 (для тех контактов, которые сконфигурированы как выход).

3 Установите в единицу биты с нулевого по третий в регистре альтернативных функций порта PC. С этого момента выводы PC[3:0] переходят под управление регистра флагов.

4 Прочитайте биты FLGx в регистре SQSTAT для определения входного значения для каждого вывода FLAG3–0, или используйте входные значения флагов (FLAGx_IN) в условных командах.

Выводы флагов (FLAG3–0) позволяют процессору посылать сигналы внешним устройствам или получать входные сигналы от них. Отметим, что аналогичные функции по управлению внешними флагами можно реализовать и при помощи регистров порта PC. Отличие будет только в том, что управление с помощью регистра FLAGREG будет происходить быстрее, т.к. регистр FLAGREG относится к группе регистров устройства управления и тактируется клоком ядра CCLK.

Таблица 71 – Альтернативные функции портов PE, PD

Обозначение				Тип	Выполняемая функция
Функция 1	Функция 2	Функция 3	GPIO		
OSCO			PE[22]	O	Синхронизация, выход для подключения резонатора
A[21]			PE[21]	O	Шина адреса внешнего интерфейса
A[20]			PE[20]	O	Шина адреса внешнего интерфейса
A[19]			PE[19]	O	Шина адреса внешнего интерфейса
A[18]			PE[18]	O	Шина адреса внешнего интерфейса
A[17]			PE[17]	O	Шина адреса внешнего интерфейса
A[16]			PE[16]	O	Шина адреса внешнего интерфейса
A[15]			PE[15]	O	Шина адреса внешнего интерфейса
A[14]			PE[14]	O	Шина адреса внешнего интерфейса
A[13]			PE[13]	O	Шина адреса внешнего интерфейса
A[12]			PE[12]	O	Шина адреса внешнего интерфейса
A[11]			PE[11]	O	Шина адреса внешнего интерфейса
A[10]			PE[10]	O	Шина адреса внешнего интерфейса
A[9]			PE[9]	O	Шина адреса внешнего интерфейса
A[8]			PE[8]	O	Шина адреса внешнего интерфейса
A[7]			PE[7]	O	Шина адреса внешнего интерфейса
A[6]			PE[6]	O	Шина адреса внешнего интерфейса
A[5]			PE[5]	O	Шина адреса внешнего интерфейса
A[4]			PE[4]	O	Шина адреса внешнего интерфейса
A[3]			PE[3]	O	Шина адреса внешнего интерфейса

Обозначение				Тип	Выполняемая функция
Функция 1	Функция 2	Функция 3	GPIO		
A[2]			PE[2]	O	Шина адреса внешнего интерфейса
A[1]			PE[1]	O	Шина адреса внешнего интерфейса
A[0]			PE[0]	O	Шина адреса внешнего интерфейса
DATA[31]	NF_RDY	MIL2_RTAP	PD[31]	I/O	Шина данных внешнего интерфейса
DATA[30]	NF_CS[2]	MIL2_RTA4	PD[30]	I/O	Шина данных внешнего интерфейса
DATA[29]	NF_CS[1]	MIL2_RTA3	PD[29]	I/O	Шина данных внешнего интерфейса
DATA[28]	NF_CS[0]	MIL2_RTA2	PD[28]	I/O	Шина данных внешнего интерфейса
DATA[27]	NF_WE	MIL2_RTA1	PD[27]	I/O	Шина данных внешнего интерфейса
DATA[26]	NF_RE	MIL2_RTA0	PD[26]	I/O	Шина данных внешнего интерфейса
DATA[25]	NF_ALE	MIL2_OU2X	PD[25]	I/O	Шина данных внешнего интерфейса
DATA[24]	NF_CLE	MIL2_OU2N	PD[24]	I/O	Шина данных внешнего интерфейса
DATA[23]	NF_D[7]	MIL2_OU2P	PD[23]	I/O	Шина данных внешнего интерфейса
DATA[22]	NF_D[6]	MIL2_IN2N	PD[22]	I/O	Шина данных внешнего интерфейса
DATA[21]	NF_D[5]	MIL2_IN2P	PD[21]	I/O	Шина данных внешнего интерфейса
DATA[20]	NF_D[4]	MIL2_OU1X	PD[20]	I/O	Шина данных внешнего интерфейса
DATA[19]	NF_D[3]	MIL2_OU1N	PD[19]	I/O	Шина данных внешнего интерфейса
DATA[18]	NF_D[2]	MIL2_OU1P	PD[18]	I/O	Шина данных внешнего интерфейса
DATA[17]	NF_D[1]	MIL2_IN1N	PD[17]	I/O	Шина данных внешнего интерфейса
DATA[16]	NF_D[0]	MIL2_IN1P	PD[16]	I/O	Шина данных внешнего интерфейса
DATA[15]			PD[15]	I/O	Шина данных внешнего интерфейса
DATA[14]			PD[14]	I/O	Шина данных внешнего интерфейса
DATA[13]			PD[13]	I/O	Шина данных внешнего интерфейса
DATA[12]			PD[12]	I/O	Шина данных внешнего интерфейса
DATA[11]			PD[11]	I/O	Шина данных внешнего интерфейса
DATA[10]			PD[10]	I/O	Шина данных внешнего интерфейса
DATA[9]			PD[9]	I/O	Шина данных внешнего интерфейса
DATA[8]			PD[8]	I/O	Шина данных внешнего интерфейса
DATA[7]			PD[7]	I/O	Шина данных внешнего интерфейса
DATA[6]			PD[6]	I/O	Шина данных внешнего интерфейса
DATA[5]			PD[5]	I/O	Шина данных внешнего интерфейса
DATA[4]			PD[4]	I/O	Шина данных внешнего интерфейса
DATA[3]			PD[3]	I/O	Шина данных внешнего интерфейса
DATA[2]			PD[2]	I/O	Шина данных внешнего интерфейса
DATA[1]			PD[1]	I/O	Шина данных внешнего интерфейса
DATA[0]			PD[0]	I/O	Шина данных внешнего интерфейса

11 Прерывания

Процессор имеет контроллер прерываний, который принимает и обслуживает запросы от внешних портов и от внутренних периферийных устройств. Возможно задание векторных прерываний и программных прерываний, задаваемых пользователем. Прерывания могут быть вложенными. При обработке прерывания выполняется переход по вектору прерывания, значение которого хранится в таблице векторов прерываний.

Процессор поддерживает обработку прерываний, являющихся индикаторами различных внешних или внутренних событий. Прерывания делятся на программные и аппаратные.

Для каждого запроса прерывания существует:

– регистр в таблице векторов прерываний (группы регистров 0x38 и 0x39);
– бит в регистре защелки прерывания ILAT;
– бит в регистре маски прерывания IMASK;
– бит в регистре приоритета прерывания PMASK.

Номера битов в регистрах одинаковы и соответствуют номеру прерывания. Этот же номер имеет и регистр вектора прерывания в таблице векторов.

Программные прерывания вызываются специальными командами или определенными ситуациями, которые возникают при выполнении команд.

Аппаратные прерывания вызываются различными событиями, которые происходят в периферийных устройствах, или входным сигналом на выводах внешних прерываний.

Прерывание может быть вызвано аппаратными ошибками в процессоре, такими как:

– Ошибка в работе контроллера DMA;
– Ошибка порта связи.

Прерывания могут быть чувствительны к фронту или уровню сигнала:

– *Прерывания чувствительные к фронту* фиксируются при их появлении установкой определенного флага в регистре ILAT. Флаги хранятся до тех пор, пока не будут обслужены или сброшены по команде. Если флаг не сбрасывается во время обработки, новое событие не обнаруживается и запрос игнорируется.

– *Прерывания чувствительные к уровню* должны поддерживать активный уровень запроса до начала обработки, в противном случае они не видны. Если запрос продолжается после обработки, он считается новым запросом прерывания.

Чувствительные к уровню прерывания обычно появляются как некоторый активный результат в определенном доступном регистре и аннулируются чтением этого регистра или записью в него неактивного значения. Чувствительные к фронту прерывания инициируются событием (например, истечением таймера).

Каждое из прерываний процессора имеет регистр вектора прерываний, который является адресом процедуры, обслуживающей данное прерывание. Весь регистровый файл называется Таблица Вектора Прерываний (IVT). Каждый регистр таблицы является 32-битным для того, чтобы подключать процедуры обработки прерываний из внешней или внутренней памяти.

11.1 Группы регистров контроллера прерываний

Данные группы регистров относятся к векторам прерываний, управлению прерываниями и регистрам статуса (таблицы 72 – 74). Все регистры контроллера прерываний доступны только как однословные.

11.1.1 Группы регистров векторов прерываний

Данная группа регистров доступна как из поля команды пересылки, с использованием номера группы и номера регистра в группе, так и с помощью команд загрузки и сохранения по адресу в адресном пространстве периферийных устройств. Базовый адрес таблицы векторов прерываний в адресном пространстве периферийных устройств имеет значение 0x8000_0300. Смещение вектора по отношению к базовому адресу, а также номер регистра с учетом его группы приведены в таблице 72. Номер регистра образуется конкатенацией номера группы (старшие шесть бит) и номера регистра в группе (младшие пять бит).

Таблица 72 – Группа А регистров таблицы вектора прерываний

Имя	Описание	Номер	Смещение
IVKERNEL	Прерывание ядра VDK	0x0700	0
IVGPIO	Прерывание от портов общего назначения PA, PB, PC	0x0701	1
IVTIMER0LP	Низкий приоритет таймера #0	0x0702	2
IVTIMER1LP	Низкий приоритет таймера #1	0x0703	3
IVUART0	Интерфейс UART 0	0x0704	4
IVUART1	Интерфейс UART 1	0x0705	5
IVLINK0	Порт связи #0	0x0706	6
IVLINK1	Порт связи #1	0x0707	7
IVUART2	Интерфейс UART 2	0x0708	8
IVUART3	Интерфейс UART 3	0x0709	9
IVNANDC	NAND флэш контроллер	0x070A	10
IVMIL0	Интерфейс МКПД 0	0x070B	11
IVMIL1	Интерфейс МКПД 1	0x070C	12
IVDIGC	Цифровой коррелятор	0x070D	13
IVDMA0	Регистр DMA #0	0x070E	14
IVDMA1	Регистр DMA #1	0x070F	15
IVDMA2	Регистр DMA #2	0x0710	16
IVDMA3	Регистр DMA #3	0x0711	17
IVARINC_RX	Интерфейс ARINC (приемник)	0x0712	18
IVARINC_TX	Интерфейс ARINC (передатчик)	0x0713	19
IVSPI1	Интерфейс SPI1	0x0714	20
IVSPI2	Интерфейс SPI2	0x0715	21
IVDMA4	Регистр DMA #4	0x0716	22
IVDMA5	Регистр DMA #5	0x0717	23
IVDMA6	Регистр DMA #6	0x0718	24
IVDMA7	Регистр DMA #7	0x0719	25
IVI2C	Интерфейс I2C	0x071A	26

Имя	Описание	Номер	Смещение
IVGTMR0	Таймер 0 с функцией Захвата/ШИМ	0x071B	27
IVGTMR1	Таймер 1 с функцией Захвата/ШИМ	0x071C	28
IVDMA8	Регистр DMA #8	0x071D	29
IVDMA9	Регистр DMA #9	0x071E	30
IVDMA10	Регистр DMA #10	0x071F	31

Таблица 73 – Группа B регистров таблицы вектора прерываний

Имя	Описание	Номер	Смещение
IVDMA11	Регистр DMA #11	0x0720	32
IVADDA0	UP\DOWN модуль 0	0x0721	33
IVADDA1	UP\DOWN модуль 1	0x0722	34
IVADDA2	UP\DOWN модуль 2	0x0723	35
IVADDA3	UP\DOWN модуль 3	0x0724	36
IVDMA12	Регистр DMA #12	0x0725	37
IVHOST	Хост-интерфейс	0x0726	38
IVLCD	Интерфейс LCD	0x0727	39
IVCAN0	Интерфейс CAN0 (для микросхем с ревизии 3)	0x0728	40
IVIRQ0	Вывод регистра IRQ0	0x0729	41
IVIRQ1	Вывод регистра IRQ1	0x072A	42
IVIRQ2	Вывод регистра IRQ2	0x072B	43
IVIRQ3	Вывод регистра IRQ3	0x072C	44
IVSPI	Интерфейс SPI 0	0x072D	45
IVSSI0	Интерфейс SSI 0	0x072E	46
IVSSI1	Интерфейс SSI 1	0x072F	47
VIRPT	Векторный регистр VIRPT	0x0730	48
IVVCAM	Видеокамера	0x0731	49
IVBUSLK	Вектор блокировки шины/Модуль TimeStamp*	0x0732	50
IVCAN1	Интерфейс CAN1 (для микросхем с ревизии 3)	0x0733	51
IVTIMER0HP	Высокий приоритет таймера 0	0x0734	52
IVTIMER1HP	Высокий приоритет таймера 1	0x0735	53
IVALARM	RTC ALARM	0x0736	54
IVTIC	RTC TIC	0x0737	55
IVDOG	RTC WATCHDOG	0x0738	56
IVHW	Аппаратная ошибка	0x0739	57
IVMACTX	Интерфейс Ethernet (передатчик) (для микросхем с ревизии 3)	0x073A	58
IVMACRX	Интерфейс Ethernet (приемник) (для микросхем с ревизии 3)	0x073B	59
IVUSBDMA	Интерфейс USB (DMA) (для микросхем с ревизии 3)	0x073C	60
IVUSB	Интерфейс USB (для микросхем с ревизии 3)	0x073D	61
-	-	-	62-63

* См. бит 10 регистра «Регистр управления прерываниями (INTCTL)» (для микросхем с ревизии 3)

11.1.2 Группа регистров управления контроллером прерываний

В задачу регистров управления прерывания входит прием запроса на прерывание, его обработка, генерация номера прерывания для передачи в процессор. Список регистров приведен в таблице 74. Регистры доступны как по номеру, так и по адресу в пространстве адресов периферийных устройств.

Таблица 74 – Группа регистров управления прерываниями

Имя	Описание	Номер\Адрес	Значение после сброса
ILATL	ILAT, младшие разряды	0x0740\0x8000_0340	0
ILATH	ILAT старшие разряды	0x0741\0x8000_0341	0
ILATSTL	ILAT младшие разряды, установка	0x0742\0x8000_0342	Не определено
ILATSTH	ILAT старшие разряды, установка	0x0743\0x8000_0343	Не определено
ILATCLL	ILAT младшие разряды, сброс	0x0744\0x8000_0344	Не определено
ILATCLH	ILAT старшие разряды, сброс	0x0745\0x8000_0345	Не определено
PMASKL	PMASK младшие разряды	0x0746\0x8000_0346	0
PMASKH	PMASK старшие разряды	0x0747\0x8000_0347	0
IMASKL	IMASK младшие разряды	0x0748\0x8000_0348	0
IMASKH	IMASK старшие разряды	0x0749\0x8000_0349	0
-	-	-	-
INTCTL	Управление прерыванием	0x074E\0x8000_034E	0

11.1.3 Регистр управления прерываниями (INTCTL)

Регистр INTCTL – 32-битный регистр, который управляет чувствительностью внешних входов прерываний IRQ3-0 (прерываний чувствительных к фронту или уровню) и обеспечивает управление стартом и остановкой таймеров. Подробное описание разрядов регистра приведено в таблице 75.

Таблица 75 – Регистр INTCTL

Бит	Имя	Назначение
0	IRQ0_EDGE	Запрос прерывания по входу IRQ0 вызывается: 0 – срезом сигнала (из высокого в низкий); 1 – уровнем (низкий)
1	IRQ1_EDGE	Запрос прерывания по входу IRQ1 вызывается: 0 – срезом сигнала (из высокого в низкий); 1 – уровнем (низкий)
2	IRQ2_EDGE	Запрос прерывания по входу IRQ2 вызывается: 0 – срезом сигнала (из высокого в низкий); 1 – уровнем (низкий)
3	IRQ3_EDGE	Запрос прерывания по входу IRQ3 вызывается: 0 – срезом сигнала (из высокого в низкий); 1 – уровнем (низкий)
4	TMR0RN	Разрешение работы таймера 0: 0 – таймер остановлен; 1 – таймер работает

Бит	Имя	Назначение
5	TMR1RN	Разрешение работы таймера 1: 0 – таймер остановлен; 1 – таймер работает
6	DIS_VREAD	Блокировка чтения памяти векторов при фиксации запроса прерывания (с целью снижения энергопотребления) (для микросхем с ревизии 3): 0 – блокировка чтения не происходит; 1 – блокировака происходит
7	PMODE	Режим задания приоритетов прерываний (для микросхем с ревизии 3): 0 – режим выключен; 1 – режим включен
8	MACTX	Запрос прерывания от передатчика интерфейса Ethernet вызывается (для микросхем с ревизии 3): 0 – фронтом сигнала (из низкого в высокий); 1 – уровнем (из низкого в высокий)
9	MACRX	Запрос прерывания от приемника интерфейса Ethernet вызывается (для микросхем с ревизии 3): 0 – фронтом сигнала (из низкого в высокий); 1 – уровнем (из низкого в высокий)
10	IRQ50_SEL	Выбор источника запроса прерывания (вектор IVBUSLK) (для микросхем с ревизии 3): 0 – Сигнал блокировки шины; 1 – Модуль TimeStamp (1)
31:6	-	Не используются и при чтении всегда равны нулю

Описание регистров таймера и способов работы с таймерами приведено в разделе 9 «Таймеры с функцией Захвата/ШИМ».

11.1.4 Регистры защелки прерываний (ILATL/ILATH)

Все запросы прерываний, которые поступают в контроллер прерываний, фиксируются (запоминаются) в регистре ILAT. Регистр ILAT – это 64-битный регистр доступный только для чтения как два 32-битных регистра ILATH и ILATL:

- ILATL – младшая часть ILAT (биты 31:0);
- ILATH – старшая часть ILAT (биты 63:32).

Каждый бит соответствует одному прерыванию и устанавливается при появлении этого прерывания. Порядок битов запросов прерываний устанавливается в соответствии с приоритетами прерываний: 0 – самый низкий приоритет.

Регистр ILAT имеет псевдонимы для упрощения процедур установки и сброса бит регистра. Для установки используются регистры ILATSTL или ILATSTH, а для сброса – ILATCLL или ILATCLH. Регистры-псевдонимы доступны только по записи. При установке записываемое значение 1 приводит к установке бита, ноль не изменяет

значение. При сбросе запись нуля вызывает сброс, а запись единицы не изменяет значение.

11.1.5 Регистры маскирования прерываний (IMASKL/IMASKH)

Каждый бит запроса прерываний, зафиксированный в регистре ILAT, имеет соответствующий ему бит разрешения обслуживания прерывания (или бит маскирования прерывания). Значение 1 в бите маски разрешает прохождение соответствующего ему запроса прерывания в ILAT для дальнейшей обработки.

Регистр IMASK – 64-битный регистр, доступный как два 32-битных регистра IMASKL и IMASKH. После сброса значение IMASK всегда равно 0.

11.1.6 Регистры приоритетного маскирования прерываний (PMASKL/PMASKH)

Регистр PMASK – 64-битный регистр, доступный как два 32-битных регистра PMASKL и PMASKH. Регистры доступны только для чтения и отображают состояние запросов прерываний, которые поступают на приоритетный шифратор контроллера прерываний.

Каждый бит регистра запроса прерываний соответствует определенному источнику прерываний и имеет соответствующий ему (с тем же номером) бит в регистре маскирования IMASK и бит в регистре PMASK. Назначения бит этих регистров приведены в таблицах 76 и 77.

Таблица 76 – Биты регистров ILATL, IMASKL, PMASKL

Бит	Имя	Описание
0	INT_KERNEL	Прерывание ядра
1	INT_GPIO	Порты PA, PB, PC
2	INT_TIMER0LP	Низкий приоритет таймера #0
3	INT_TIMER1LP	Низкий приоритет таймера #1
4	INT_UART0	UART0
5	INT_UART1	UART1
6	INT_LINK0	Порт связи #0
7	INT_LINK1	Порт связи #1
8	INT_UART2	UART2 (для микросхем с ревизии 3)
9	INT_UART3	UART3 (для микросхем с ревизии 3)
10	INT_NANDC	Контроллер NAND
11	INT_MIL0	МКПД 0
12	INT_MIL1	МКПД 1
13	INT_DIGC	Цифровой коррелятор
14	INT_DMA0	DMA #0
15	INT_DMA1	DMA #1
16	INT_DMA2	DMA #2
17	INT_DMA3	DMA #3
18	INT_ARINC_RX	Приемник ARINC
19	INT_ARINC_TX	Передатчик ARINC
20	INT_SPI1	Контроллер SPI1

Бит	Имя	Описание
21	INT_SPI2	Контроллер SPI2
22	INT_DMA4	DMA #4
23	INT_DMA5	DMA #5
24	INT_DMA6	DMA #6
25	INT_DMA7	DMA #7
26	INT_I2C	Контроллер I2C
27	INT_GTMR0	Таймер 0 с функцией Захвата/ШИМ
28	INT_GTMR1	Таймер 1 с функцией Захвата/ШИМ
29	INT_DMA8	DMA #8
30	INT_DMA9	DMA #9
31	INT_DMA10	DMA #10

Таблица 77 – Биты регистров ILATH, IMASKH, PMASKH

Бит	Имя	Описание
0	INT_DMA11	DMA #11
1	INT_ADDA0	Модуль UP\DOWN 0
2	INT_ADDA1	Модуль UP\DOWN 1
3	INT_ADDA2	Модуль UP\DOWN 2
4	INT_ADDA3	Модуль UP\DOWN 3
5	INT_DMA12	DMA #12
6	INT_HOST	Хост-интерфейс
7	INT_LCD	Контроллер LCD
8	INT_CAN0	Контроллер CAN0 (для микросхем с ревизии 3)
9	INT_IRQ0	Вывод IRQ0
10	INT_IRQ1	Вывод IRQ1
11	INT_IRQ2	Вывод IRQ2
12	INT_IRQ3	Вывод IRQ3
13	INT_SPI0	Контроллер SPI 0
14	INT_SSI0	Контроллер SSI 0
15	INT_SSI1	Контроллер SSI 1
16	INT_VIRPT	Векторный регистр VIRPT
17	INT_VCAM	Контроллер видеокамеры
18	INT_BUSLOCK	Блокировка шины или модуль TimeStamp
19	INT_H264 INT_CAN1	H264 декодер (для микросхем ревизии 2) Контроллер CAN1 (для микросхем с ревизии 3)
20	INT_TIMER0HP	Высокий приоритет таймера 0
21	INT_TIMER1HP	Высокий приоритет таймера 1
22	INT_ALARM	Прерывание ALARM от RTC
23	INT_TIC	TIC прерывание от RTC
24	INT_WDOG	Прерывание от сторожевого таймера
25	INT_HWERR	Аппаратная ошибка
31:26	-	-

11.2 Операции с прерываниями

Операциями с прерываниями являются следующие виды операций:

- установка запросов прерываний;
- обработка прерываний или исключений;
- возврат из прерывания.

Контроллер прерываний включает следующий регистры установки и управления:

- регистр управления прерываниями (INTCTL);
- регистры маски прерываний (IMASK) – IMASKH и IMASKL;
- регистры макси приоритета (PMASK) – PMASKH и PMASKL;
- регистры защелки прерываний (ILAT) – ILATH и ILATL.

Наличие запроса прерывания отражается активным (равен единице) битом в регистре ILAT. Если соответствующий ему бит в регистре маски IMASK установлен, запрос прерывания поступает на дальнейшую обработку. Регистр приоритетов PMASK хранит запросы прерываний, которые процессор принял к обработке. Приоритетом прерывания является номер бита в регистре PMASK (от 0 до 63).

Регистр приоритетов позволяет блокировать низкоприоритетные запросы прерываний до момента окончания обработки более высокоприоритетного. Если прерывание разрешено, и его приоритет выше текущего обрабатываемого прерывания (или нет обработки прерываний в данный момент), контроллер прерываний формирует активный уровень запроса прерывания к процессору, сопровождая его номером прерывания и адрес-вектором процедуры обслуживания данного прерывания. Если бит GIE (бит глобального разрешения прерываний) в регистре SQCTL установлен, процессор прерывает текущий ход программы и переходит к выполнению процедуры обработки прерывания.

Как только прерывание начинает обслуживаться, соответствующий данному прерыванию бит в регистре PMASK устанавливается в «1». Данная установка запрещает прохождение запросов прерываний с приоритетом ниже обрабатываемого. При этом процедура обработки прерывания может разрешить обработку более высокоприоритетных прерываний. Как только обработка прерывания заканчивается, соответствующий ему бит в PMASK сбрасывается. Это происходит при выполнении команд RTI или RDS. При этом команда RTI означает завершение процедуры обработки, а команда RDS означает, что обработка текущего прерывания может быть прервана другим прерыванием, т.е. возможны вложенные прерывания.

Активный запрос прерывания вызывает установку соответствующего бита в регистре ILAT. Существует возможность программно установить бит в этом регистре, т.е. приложение может эмулировать прерывание посредством записи в регистр ILAT. Запись по адресам ILATSTL или ILATSTH обновляет регистр ILAT значением логического «ИЛИ» старого содержимого и нового записываемого. В результате, каждый установленный бит в записанных данных устанавливает соответствующий бит в регистре ILAT. Установка бита прерывания вызывает в процессоре предположение о возникновении соответствующего прерывания. Биты прерывания могут быть также сброшены записью в адреса сброса ILATCLL или ILATCLH. Такая запись применяет

оператор AND к записываемым данным и старым данным регистра ILAT. В этом случае нулевой бит во входных данных сбрасывает соответствующий бит в ILAT, тогда как установленный бит остается неизменным. Таким образом, приложение может сбросить ждущее прерывание до его исполнения.

Операции сброса или установки бит в ILAT должны выполняться при следующих условиях:

– незадействованные (резервные) биты не могут быть установлены. При записи по адресам установки или сброса резервные биты должны быть нулями;

– биты программного исключения и эмуляторного исключения не могут быть установлены через ILATST. Для того чтобы вызвать такие исключения, следует использовать команды TRAP и EMUTRAP;

– прерывания, которые запускаются уровнем, не должны изменяться;

– записи в ILATL и ILATH не должны предприниматься (т.е. только сброс или установка);

– прерывание должно быть сброшено в то время, когда оно запрещено, иначе оно может быть обслужено до того, как будет сброшено;

– адреса установки и сброса работают с одинарными словами.

11.3 Программа обработки аппаратного прерывания

Действия, выполняемые при обработке прерываний можно разделить на две группы:

– действия, выполняемые контроллером прерываний;

– действия, выполняемые процессором.

Рассмотрим последовательность обработки запроса прерывания на примере прерывания от таймера 0. Действия контроллера прерываний будут следующие:

– Когда таймер включен и его счетчик имеет в текущем такте SOCCLK значение «1», вырабатывается активный уровень запроса прерывания. Активный запрос существует только один такт, которого достаточно, чтобы в следующем такте биты 2 и 52 регистра ILAT установились в «1». Бит 2 соответствует запросу прерывания от таймера 0 с низким уровнем приоритета, а бит 52 – с высоким уровнем.

– При ожидании прерывания с высоким уровнем приоритета бит 52 в регистре IMASK должен быть установлен в «1». Логическое «И» между битом запроса прерывания и битом маски равно «1», следовательно, прерывание может быть принято к обслуживанию.

– Далее выполняется анализ регистра PMASK. Если биты регистра с 52-й по 63-й равны нулю, в обработке нет прерываний равного или более высокого приоритета, чем данное. В противном случае придется подождать, пока не будет обслужено более высокоприоритетное прерывание.

– Выполнения описанных выше условий достаточно чтобы контроллер прерываний сформировал запрос прерывания к процессору. Кроме активного уровня запроса на линии, контроллер передает в процессор номер прерывания (52), а также

извлекает из таблицы векторов прерываний значение регистра с номером 52 и передает его в процессор. Это значение является адресом процедуры обработки данного прерывания.

Когда запрос прерывания поступает в процессор (в устройство управления), первым действием процессора является проверка бита глобального разрешения прерываний GIE регистра управления SQCTL. Если бит равен единице – прерывания разрешены и процессор выполняет следующие действия:

– Запоминает номер и адрес-вектор обработки пришедшего прерывания.

– Выполняет сброс всего конвейера чтения команд вместе с буфером IAB.

– Запускает чтение команд по адрес-вектору прерывания.

– Конвейер обработки команд продолжает и заканчивает обработку всех команд, которые находились на конвейере в момент сброса конвейера чтения команд.

– Когда последняя команда конвейера обработки завершает свое выполнение, процессор повторно проверяет бит разрешения прерываний GIE и, если прерывания по-прежнему разрешены, первая команда процедуры обработки прерывания поступает из буфера команд на конвейер обработки.

– При достижении первой командой последней стадии конвейера (W) происходит формирование сигнала о том, что данное прерывание взято на обработку. Этот факт приводит к установке бита 52 регистра PMASK и сброс бита 52 в регистре ILAT. Установленный бит в регистре PMASK запрещает все запросы прерывания уровнем ниже 53 (т.е. новое прерывание от таймера 0 также запрещено). Вместе с этим процессор запоминает в специальном регистре RETI адрес следующей команды, которая должна быть выполнена при выходе из прерывания. Дополнительно процессор устанавливает внутренний флаг EXE_ISR, который запрещает все прерывания. Флаг EXE_ISR указывает на то, что процессор находится в состоянии обработки прерывания. Флаг доступен для чтения как 20-й бит регистра SQSTAT.

– Переход на обработку прерывания вызывает установку режима супервизора (если до этого был режим пользователя).

– Сброс бита 52 в регистре ILAT означает, что бит стал доступен для приема нового запроса прерывания от таймера 0.

– Далее процессор выполняет все команды, которые соответствуют процедуре обслуживания данного прерывания. Для выхода из процедуры обработки используется команда RTI.

– Выполнение команды RTI приводит к сбросу бита 52 в регистре PMASK, что разрешает прохождение всех прерываний, которые ранее блокировались. Также данная команда вызывает сброс флага EXE_ISR, что снимает блокировку прерываний и включает функции бита GIE. Процессор переходит к выполнению команд по адресу из регистра RETI.

Процедура, описанная выше, соответствует типичной ситуации обработки при отсутствии каких-либо аномалий. Однако существуют некоторые особенности:

– Если перед повторной проверкой разрешения прерываний (после полного освобождения конвейера обработки) оказывается, что прерывания запрещены (это может

быть сделано командой сбрасывающей бит GIE), тогда все действия по обработке прерывания отменяются и конвейер перезапускается с адреса, следующего за адресом последней выполненной команды.

– Если запрос прерывания из контроллера пришел в процессор, но бит GIE равен нулю, прерывание будет отложено. В это время в контроллере прерываний может появиться запрос с более высоким приоритетом, чем данное. Контроллер прерываний обработает его и передаст в процессор номер и адрес-вектор нового прерывания. Запрос от таймера 0 в этом случае будет отложен до момента окончания обработки нового прерывания.

– Возможна ситуация, когда запрос пришел из контроллера, но в это же время процессор выполнял сброс бита запроса прерывания в регистре ILAT. Эта ситуация приведет к тому что запрос из контроллера прерываний будет стабильным в течение одного или нескольких тактов. Тем не менее, в такой ситуации запрос может быть обработан. Если снятие запроса произошло после или во время сброса конвейера чтения команд, это означает, что запрос будет обслужен. Для надежного выполнения подобных операций необходимо предварительно запрещать прерывания битом GIE, а затем производить сброс в контроллере прерываний.

Обычно процедура обработки прерывания начинается с сохранения контекста процессора, т.е. сохранения на стеке всех регистров процессора, которые будут использованы программой обработки прерывания. Это необходимо для того, чтобы после обработки прерывания восстановить прежнее состояние процессора и дать возможность прерванной программе корректно возобновить работу. Факт обработки прерывания отражается в установке флага EXE_ISR и это запрещает все прерывание, в том числе и более высокоприоритетные, чем данное. Если время обработки прерывания значительно и это может повредить обработке запроса более приоритетного прерывания, программа может разрешить обслуживание более высокоприоритетных прерываний. Для этого необходимо обязательно сохранить на стеке содержимое регистра RETI, который хранит адрес возврата из прерывания. Для сохранения содержимого RETI используется специальный псевдоним этого регистра – RETIB.

Выполнение команды **j0 = RETIB;;** вызовет копирование регистра RETI в регистр j0 и сброс флага EXE_ISR. Сам регистр j0 должен до этого быть сохранен на стеке. После выполнения указанной команды процессор имеет возможность обслуживать более высокоприоритетные запросы, чем запрос 52. Описываемая ситуация вносит коррективы в последовательность действий при выходе из обработки прерывания. Перед выходом мы имеем ситуацию, когда содержимое регистра возврата RETI неопределенно (новые прерывания могли изменить его). Поэтому необходимо взять наш адрес возврата из регистра j0 (или другого места, где он хранился) и вернуть обратно в RETI. Однако выполнение обычной пересылки **RETI = j0;;** будет некорректным, т.к. после этой команды возможно возникновение прерывания и адрес возврата может быть потерять. Снова следует использовать регистр-псевдоним RETIB. Пересылка **RETIB = j0;;** пересылает адрес возврата в RETI и устанавливает флаг EXE_ISR. Это запрещает прерывания и позволяет безопасно восстановить весь контекст, а затем выполнить финишную команду RTI.

В процедуре, описанной выше, разрешения прерываний с более высоким приоритетом, чем текущее. Разрешаются прерывания с приоритетом от 53 и выше, но прерывание с приоритетом 52 запрещено. Это означает, что, обрабатывая запрос от таймера 0, имеется возможность принять новый запрос прерывания от таймера, т.к. бит 52 в регистре ILAT был сброшен и имеет возможность зафиксировать новое событие, но не обслуживать его (т.к. бит 52 в регистре PMASK установлен). Также нет возможности обслуживать запросы с приоритетом ниже 52.

Разрешить обслуживание запросов с приоритетом ниже 52 можно с помощью команды RDS. Выполнение команды **j0 = RETIB;;** разрешает только прерывания выше 52-го, т.к. бит 52 в регистре PMASK установлен. Выполнение команды RDS вызывает следующие действия процессора:

– сбрасывается в ноль текущий самый высокоприоритетный установленный бит в регистре PMASK;

– сбрасывается флаг EXE_ISR;

– выполняется переход в режим работы, который предшествовал обработке прерывания;

– выполняется перезапуск конвейера в новых условиях.

Перед выполнением команды RDS необходимо сохранить адрес возврата. Если прежним режимом является режим пользователя с соответствующими ограничениями, и возврат в этот режим нежелателен, нужно сначала сохранить регистр SQCTL, и затем установить в нем бит режима супервизора. В этом случае после команды RDS останется режим супервизора.

Вышеописанные действия позволяют реализовать вложенные прерывания. Выход из подобных прерываний выполняется с использованием регистра RETIB.

Описанные действия относятся к обработке любого аппаратного прерывания, а не только 52-го.

Команда возврата из прерывания RTI является командой перехода и может выполняться с опцией предсказания или без. Использование опции предсказания позволяет ускорить выход из прерывания на несколько тактов процессора. Необходимо помнить, что содержимое регистра RETI должно быть корректно в момент выполнения предсказания, т.е. в момент передачи в память адреса линии команд содержащей RTI. Если регистр RETI загружается непосредственно перед командой RTI или в нескольких тактах до RTI, предпочтительнее использовать команду RTI без предсказания перехода, т.к. в данной случае предсказание будет ошибочным, и все выполненные действия будут отменены.

11.4 Особенности сохранения контекста процессора

Во время выполнения программы процессор может находиться в различных режимах. Выделяют два основных режима работы:

– режим супервизора (ядра или системный);

– режим пользователя.

Режим супервизора может иметь несколько подрежимов (или видов):

– выполнение задачи системы;

– обработка прерывания;

– обработка программной исключительной ситуации;

– отладка.

Подрежим можно определить в результате анализа регистра состояния SQSTAT. Для каждого подрежима процессор имеет соответствующие регистры связи:

– CJMP-регистр связи для выполнения вызовов процедур при нормальном ходе программы как в режиме супервизора, так и в режиме пользователя;

– RETI-регистр связи при возникновении прерывания;

– RETS-регистр связи при возникновении программной исключительной ситуации;

– DBGE-регистр связи при вызове эмулятора.

Регистры связи RETI, RETS, DBGE используются только при входе в исключительную ситуацию и при выходе из нее. При нормальном ходе программы для связи используется регистр CJMP. В процессоре используется один общий для всех режимов регистр-указатель стека. Также в качестве указателя стека может быть использован любой от нулевого до 30-го регистр IALU, но по умолчанию компилятор использует в качестве указателя стека регистр J27/K27. На аппаратном уровне процессор поддерживает раздельные регистры J27/K27 для режима супервизора (KSP или SP) и режима пользователя (USP).

Режим работы с двумя указателями стеков возможен только при установленном бите SQCTL[14]. Если бит SQCTL[14] равен нулю, процессор работает с общим для всех режимов указателем стека. В режиме ядра регистр j27/k27 соответствует KSP, а также имеется возможность чтения и записи указателя USP через резервный регистр номер 27 группы регистров базы и длины циклической адресации модулей IALU. В режиме пользователя j27/k27 доступен только как USP. Выбор указателя стека осуществляется следующим образом:

– если процессор находится в режиме обработки прерывания – используется KSP;

– если процессор находится в режиме обработки исключительной ситуации – используется KSP;

– если процессор находится в режиме эмулятора – используется KSP;

– если процессор исполняет прикладную задачу и бит SQCTL[9] равен нулю – используется USP;

– если процессор исполняет прикладную задачу и бит SQCTL[9] равен единице – выбор указателя определяется битом SQCTL[7]: если бит равен нулю – KSP, иначе - USP.

Таким образом, в режиме супервизора возможна работа как с указателем ядра, так и с указателем пользователя.

Основные трудности в выборе указателя возникают в момент смены режима, т.к. из-за наличия конвейера фазы чтения и записи регистра разнесены во времени, и различным моментам времени могут соответствовать различные режимы.

Процесс смены режима посредством модификации бита SQCTL[9] наиболее безопасный, т.к. каждый раз после записи регистра SQCTL происходит перезапуск конвейера. Это гарантирует, что команды, следующие за сменой режима, извлекаются и выполняются в требуемом режиме.

Смена режима посредством программной исключительной ситуации (установка SWI_mode) безопасна во время входа в обработчик, т.к. возникновение исключительной ситуации приводит к сбросу конвейера, его очистке и последующему переходу на программу обработки. Безопасный выход из обработчика должен быть реализован соответствующим образом. Выход из обработчика (сброс SWI_mode) осуществляется посредством выполнения команды RTI. Данная команда должна исполняться с опцией RTI (NP). Это гарантирует сброс конвейера после смены режима и последующее корректное использование указателя стека (а также работу устройства защиты).

Смена режима посредством аппаратного прерывания (установка ISR_mode) безопасна во время входа в обработчик. Это гарантируется аппаратурой процессора. Признак прерывания движется по конвейеру одновременно с командами обработчика и осуществляет выбор корректного указателя стека, если первые команды обработчика используют указатель. Безопасный выход из прерывания (сброс ISR_mode) должен быть реализован соответствующим образом. Выход осуществляется посредством выполнения команды RTI. Данная команда должна исполняться с опцией RTI (NP). Это гарантирует сброс конвейера после смены режима и последующее корректное использование указателя стека (а также работу устройства защиты).

Сброс режима обработки аппаратного прерывания или программного прерывания может быть выполнен не только командой RTI, но и командой RDS. Опасность использования команды RDS при работе с разными указателями стека, а также с устройством защиты, связано с тем, что выполнение команды RDS приводит к возврату в режим, предшествовавший прерыванию. При работе в режиме раздельных указателей стека после выполнения команды RDS будет выполнен перезапуск конвейера. До выполнения команды RDS будет доступен указатель стека супервизора, после – это может быть либо указатель стека супервизора или пользователя.

Поскольку возможно использование сразу двух указателей стека, сохранение контекста может быть ускорено, если стеки размещаются в различных банках памяти. Это позволяет за такт сохранить восемь регистров процессора. Если использование двух стеков неудобно, можно использовать факт разбиения каждого банка внутренней памяти на две половины, одна из которых – четные квадрослова, вторая – нечетные. В этом случае регистр К27 может хранить копию указателя J27, и при сохранении контекста возможна работа сразу с парой квадрослов. Это позволяет сохранять и восстанавливать сразу восемь регистров.

11.5 Обработка программных прерываний (исключений)

Исключения – программные прерывания, которые вызваны исполняемым программным кодом. При этом могут быть специальные команды (TRAP, EMUTRAP) или ситуации, которые возникают при выполнении команд. К таким ситуациям относятся аномалии при вычислении с плавающей запятой, доступ к длинным словам по невыровненному адресу и др.

Обработка программного прерывания отличается от аппаратного следующим:

– Бит GIE не имеет значения при обработке исключений. Для программных прерываний используется специальный бит глобального разрешения SWIE.

– Часть программных прерываний имеет индивидуальные биты разрешения. Это касается исключений, которые возникают в вычислительных модулях X и У. Другие исключения не имеют специальных бит для разрешения, кроме глобального.

– Программное прерывание происходит сразу за выполнением команды, инициировавшей это событие.

– Для хранения адреса возврата используется специальный регистр RETS, а не регистр RETI.

– В качестве адреса-вектора процедуры обработки исключений используется содержимое регистра IVSW.

– Возникновение исключения приводит к установке специального бита EXE_SWI. Бит виден в 21-м разряде регистра SQSTAT. Бит может быть установлен только при возникновении исключительной ситуации. Бит может быть сброшен только при выходе из обработки исключения или командой RDS. Других механизмов установки или сброса данного бита нет.

– Установка бита EXE_SWI приводит к переходу в режим супервизора (если ранее был режим пользователя), а также к запрещению всех программных и аппаратных прерываний.

– Если при обработке исключительной ситуации возникло новое исключение, оно не будет обработано.

Последовательность действий процессора, выполняемых при возникновении исключительной ситуации, следующая:

– В поле SQSTAT[11:8] процессор сохраняет код исключительной ситуации. Если ситуация была вызвана командой TRAP, в поле SQSTAT[7:3] дополнительно сохраняется параметр данной команды. Все это позволяет идентифицировать источник исключения.

– Сбрасывается весь конвейер процессора и начинается чтение команд по адресу из регистра IVSW.

– В регистре RETS сохраняется адрес возврата, а именно адрес первой команды линии следуемой за линией команд вызвавшей исключение.

– Устанавливается внутренний флаг EXE_SWI, переводя процессор в режим супервизора и запрещая все аппаратные и программные прерывания.

– Процессором выполняются все команды процедуры обработки исключительной ситуации.

– Перед выходом из прерывания программа копирует содержимое регистра RETS в регистр RETI с помощью команды RETI = RETS;;. Регистр RETIB, в данном случае, не должен использоваться.

– Выход из обработчика осуществляется с помощью команды RTI.

При возврате из программного прерывания используется регистр RETI, поэтому последовательность выхода может быть такой:

[j31+temporary address] = RETI;;
RETI = RETS;;
RTI (NP); RETI = [j31+temporary address];;

Таким образом видно, что программное прерывание при своем возникновении создает очень строгие ограничения в поведении процессора, также как и аппаратное. Однако возникновение аппаратного прерывания не запрещает обработку программного прерывания, если оно возникнет при выполнении команд аппаратного обработчика. При обработке программного прерывания можно разрешить аппаратные прерывания, выполнив команду RDS. Выполнение команды RDS приведет к сбросу бита EXE_SWI. Аппаратные прерывания станут управляться битом GIE, а программные – SWIE. Произойдет возврат к предыдущему режиму работы.

При использовании команды RDS необходимо заранее учесть влияние последующих событий на корректное исполнение программного обработчика.

Программное прерывание может быть вложенным в аппаратное, если биты EXE_ISR и EXE_SWI равны единице. В подобной ситуации команды RTI или RDS оказывают влияние только на бит EXE_SWI, и их исполнение означает переход на уровень аппаратного обработчика.

11.6 Обработка прерываний эмулятора

Прерывания эмулятора – события, которые приводят к переходу процессора в режим эмуляции, когда все его действия управляются посредством команд через интерфейс JTAG. Подобное прерывание может быть вызвано:

– командой EMUTRAP;

– срабатыванием точки наблюдения;

– аппаратным запросом из JTAG интерфейса.

Обработка данного типа прерываний имеет следующие особенности:

– Специальный бит глобального разрешения DBGEN. Биты GIE и SWIE не влияют.

– Прерывания от точек наблюдения и из JTAG имеют дополнительные биты разрешения.

– Прерывание происходит сразу за командой, которая покинула последнюю стадию конвейера W в момент возникновения запроса.

– Для хранения адреса возврата используется специальный регистр DBGE.

– В качестве адреса-вектора процедуры обработки используется фиксированный адрес 0x001F_03A4.

– Возникновение исключения приводит к установке специального бита EMUL. Бит виден в 22-м разряде регистра SQSTAT. Бит может быть установлен только при возникновении данной исключительной ситуации. Бит может быть сброшен только при выходе из обработки исключения командой RTI. Других механизмов установки или сброса данного бита не предусмотрено.

– Установка бита EMUL приводит к переходу в режим эмулятора. Также это приводит к запрещению всех программных и аппаратных прерываний.

Последовательность действий процессора, выполняемых при возникновении прерывания эмулятора, следующая:

– В поле SQSTAT[15:12] процессор сохраняет код исключительной ситуации, что позволяет идентифицировать источник исключения.

– Сбрасывается весь конвейер процессора и начинается чтение команд по адресу 0x001F_03A4. При этом изменения адреса не происходит. Данный адрес соответствует регистру команд интерфейса JTAG, т.е. процессор все время исполняет те команды, которые ему выставляет JTAG интерфейс.

– В регистре DBGE сохраняется адрес возврата.

– Устанавливается внутренний флаг EMUL, переводя процессор в режим эмулятора и запрещая все аппаратные и программные прерывания.

– Процессором выполняются все команды, которые предоставляет ему JTAG интерфейс.

– Перед выходом из прерывания программа копирует содержимое регистра DBGE в регистр RETI с помощью команды RETI = DBGE;;. Регистр RETIB, в данном случае, не должен использоваться.

– Выход из обработчика осуществляется с помощью команды RTI.

При возврате из программного прерывания используется регистр RETI, поэтому последовательность выхода может быть такой:

[j31+temporary address] = RETI;;
RETI = DBGE;;
RTI (NP); RETI = [j31+temporary address];;

Прерывание эмулятора накладывает серьезные ограничения на перечень команд, которые должны поставляться интерфейсом JTAG. Например, запрещены любые команды перехода и т.д. При переходе в режим эмулятора блокируются некоторые аппаратные ресурсы. Например, таймеры останавливаются и не меняют свое значение до момента выхода из режима эмуляции.

11.7 Источники прерываний процессора

Таблица 78 – Таблица векторов прерываний

Приоритет	Прерывание	Описание	Срабатывание
63 – 62	-	Зарезервирован	-
61	USB	Интерфейс USB (для микросхем с ревизии 3)	По фронту или по уровню
60	USB_DMA	Контроллер DMA интерфейса USB (для микросхем с ревизии 3)	По фронту или по уровню
59	MAC_RX	Интерфейс Ethernet (приемник) (для микросхем с ревизии 3)	По фронту или по уровню
58	MAC_TX	Интерфейс Ethernet (передатчик) (для микросхем с ревизии 3)	По фронту или по уровню
57	HWERR	Аппаратная ошибка	По уровню
56	WDOG	RTC. Сторожевой таймер	По фронту
55	TIC	RTC. Интервальный таймер	По фронту
54	ALARM	RTC. Будильник	По фронту
53	TIMER1H	Высокий приоритет таймера 1	По фронту
52	TIMER0H	Высокий приоритет таймера 0	По фронту
51	CAN1	Интерфейс CAN1 (для микросхем с ревизии 3)	По уровню
50	BUSLOCK	Блокировка шины	По фронту
49	VCAM	Интерфейс видеокамеры	По уровню
48	VIRPT	Векторное прерывание	По фронту
47	SSI1	Интерфейс I2S_1	По уровню
46	SSI0	Интерфейс I2S_0	По уровню
45	SPI0	Интерфейс SPI 0	По уровню
44	IRQ3	Вывод IRQ3	По фронту или по уровню
43	IRQ2	Вывод IRQ2	По фронту или по уровню
42	IRQ1	Вывод IRQ1	По фронту или по уровню
41	IRQ0	Вывод IRQ0	По фронту или по уровню
40	CAN0	Интерфейс CAN0 (для микросхем с ревизии 3)	По уровню
39	LCD	Интерфейс ЖКИ	По уровню
38	HOST	Хост-интерфейс	По фронту
37	DMA12	DMA канал 12	По фронту
36	ADDA3	UP\DOWN 3	По фронту
35	ADDA2	UP\DOWN 2	По фронту
34	ADDA1	UP\DOWN 1	По фронту
33	ADDA0	UP\DOWN 0	По фронту
32	DMA11	DMA канал 11	По фронту
31	DMA10	DMA канал 10	По фронту
30	DMA9	DMA канал 9	По фронту
29	DMA8	DMA канал 8	По фронту
28	GTMR1	Таймер 1 с функцией Захвата/ШИМ	По уровню
27	GTMR0	Таймер 0 с функцией Захвата/ШИМ	По уровню

Приоритет	Прерывание	Описание	Срабатывание
26	I2C	Интерфейс I2C	По уровню
25	DMA7	DMA канал 7	По фронту
24	DMA6	DMA канал 6	По фронту
23	DMA5	DMA канал 5	По фронту
22	DMA4	DMA канал 4	По фронту
21	INT_SPI2	Интерфейс SPI2	По уровню
20	INT_SPI1	Интерфейс SPI1	По уровню
19	ARINC_TX	Интерфейс ARINC (передатчик)	По уровню
18	ARINC_RX	Интерфейс ARINC (приемник)	По уровню
17	DMA3	DMA канал 3	По фронту
16	DMA2	DMA канал 2	По фронту
15	DMA1	DMA канал 1	По фронту
14	DMA0	DMA канал 0	По фронту
13	DIGC	Цифровой коррелятор	По фронту
12	MIL1	Интерфейс МКПД 1	По фронту
11	MIL0	Интерфейс МКПД 0	По фронту
10	NANDC	Интерфейс NAND флэш	По уровню
9	UART3	Интерфейс UART 3 (для микросхем с ревизии 3)	По уровню
8	UART2	Интерфейс UART 2 (для микросхем с ревизии 3)	По уровню
7	LINK1	Запрос порта связи 1	По уровню
6	LINK0	Запрос порта связи 0	По уровню
5	UART1	Интерфейс UART1	По уровню
4	UART0	Интерфейс UART 0	По уровню
3	TIMER1L	Низкий приоритет таймера 1	По фронту
2	TIMER0L	Низкий приоритет таймера 0	По фронту
1	GPIO	Порты PA,PB,PC	По уровню
0 (самый низкий)	KERNEL	Ядро	По фронту

11.7.1 Прерывания истечения таймера

Возможны четыре прерывания от таймеров: два для каждого таймера, одно – с высоким приоритетом и одно – с низким. Два приоритетов для одного таймера дают возможность пользователю выбрать уровень приоритета.

Регистры вектора прерываний таймера: IVTIMER0HP, IVTIMER1HP, IVTIMER0LP и IVTIMER1LP. В случае прерывания от таймера, устанавливаются оба бита приоритетов (высокий и низкий) в ILAT (например, для таймера 0 – это биты 2 и 52).

Сброс или установка бита запроса прерывания от таймера в регистре ILAT с помощью адресов ILATSTx или ILATCLx оказывает влияние сразу на оба бита таймера, т.е. нельзя разграничить запрос низкого от запроса высокого приоритета.

11.7.2 Прерывания на обслуживание порта связи

Существуют два канала портов связи, которые обычно работают с выделенными им каналами DMA. Регистры вектора обслуживания порта связи: IVLINK0, IVLINK1. Когда данные поступают в буфер приемника порта связи, и соответствующий ему канал DMA не инициализирован, порт связи формирует запрос прерывания на обслуживание.

11.7.3 Прерывания завершения работы каналов DMA

Каждый из 13 каналов DMA может генерировать прерывание. Регистры вектора: IVDMA0, IVDMA1, IVDMA2, IVDMA3, IVDMA4, IVDMA5, IVDMA6, IVDMA7, IVDMA8, IVDMA9, IVDMA10, IVDMA11 и IVDMA12. Если бит разрешения прерывания в регистре управления передачей DMA установлен, канал DMA генерирует прерывание по завершении передачи блока.

11.7.4 Прерывания от внешних источников

Порты общего назначения PA, PB, PC могут формировать запрос на прерывание от каждого из 96-ти внешних контактов. Каждый запрос может быть запрограммирован индивидуально. Однако все запросы от портов общего назначения имеют один вектор прерывания IVGPIO. Существует четыре внешних прерывания общего назначения IRQ3-0, которые могут быть обработаны особым образом. Эти внешние входы соответствуют контактам 3-0 порта PC. Для них существуют отдельные вектора прерывания (IRQ3–0): IVIRQ0, IVIRQ1, IVIRQ2 и IVIRQ3. Когда активируется один из входов, запускается прерывание (если они разрешены). Тип прерывания по фронту или уровню программируется в регистре INTCTL.

11.7.5 Векторное прерывание

Векторное прерывание – прерывание общего назначения для использования другим мастером. Другое ведущее устройство или процессор могут записывать адрес в этот регистр. Запись вызывает установку запроса прерывания, а записанное значение в регистре – адрес обработки прерывания. Регистр вектора прерывания – VIRPT.

11.7.6 Прерывание от захвата шины

Прерывание захвата шины происходит, когда установлен бит захвата шины в регистре BUSLOCK, и процессор становится хозяином шины. Регистр вектора захвата шины – IVBUSLK. Данное прерывание сохранено для совместимости с микросхемой К1967ВН028. В микросхеме К1967ВН044 внешняя шина всегда принадлежит процессору, поэтому выполнение процедуры захвата шины не требуется. Однако установка бита BUSLOCK приведет к генерации прерывания.

11.7.7 Прерывание от аппаратной ошибки

Прерывание аппаратной ошибки указывает на одну из возможных ошибок:

– Ошибка в DMA. При инициализации некоторого канала обнаружена ошибочная ситуация.

– Широковещательное чтение из внешней памяти (аналогично К1967ВН028). Бит 16 в SYSTAT активен, если произошла данная ошибка.

– Доступ к отключенной SDRAM. Бит 18 в SYSTAT активен, если произошло данное событие.

– Обращение к несуществующему адресному пространству. Бит 19 в SYSTAT активен, если произошло данное событие.

– Ошибка порта связи. В одном из регистров LSTATx в полях RER или TER активна ошибка.

Регистр вектора прерываний аппаратных ошибок – IVHW.

11.7.8 Прерывания RTC

Модуль часов реального времени RTC может формировать три запроса прерывания. Прерывание ALARM происходит при совпадении счетчика времени с заданным значением, что позволяет реализовать функции будильника. Прерывание TIC является периодическим и формируется через заданный интервал времени. Данное прерывание может быть полезно при работе ОС. Прерывание сторожевого таймера формируется в момент, когда счетчик сторожевого таймера принимает значение «1». Это служит напоминанием системе о необходимости выполнить повторную инициализацию сторожевого таймера. Если повторная инициализация не выполняется в течение определенного времени, сторожевой таймер инициирует сброс всей системы.

11.7.9 Прерывания от периферийных устройств

В процессоре имеется ряд периферийных устройств, которые могут формировать запросы прерываний к процессору. Подробное описание порядка формирования и обслуживания прерывания приведено в главах, посвященных каждому из периферийных устройств.

11.7.10 Программные исключения

Программные исключения – прерывания, появляющиеся во время выполнения программного кода. Если происходит исключение и бит SWIE равен нулю – исключение теряется. Вектор прерываний программных исключений – IVSW.

Тип исключительной ситуации можно определить по номеру в регистре состояния процессора устройства управления. Соответствие номера типу исключительной ситуации приведено в таблице 79. Используя номер исключительной ситуации в регистре состояния, можно выполнить соответствующую обработку возникшей исключительной ситуации. При этом может понадобиться дополнительная информация, доступная из

других источников или полученная в результате анализа кода командной строки, вызвавшей исключение.

Таблица 79 – Программные исключения

Номер	Имя	Описание события
0	trap	Вызывается выполнением команды TRAP. Дополнительный номер исключения можно получить из регистра состояния
1	Wp_abort	Срабатывание точки наблюдения в модуле отладки
2	I0	Исключительная ситуация в модулях ХУ
3	UNDEF	Неопределенная команда или сочетание команд
4	I1	Невыровненный адрес при доступе к длинным словам
5	I3+prot	Нарушение прав доступа к памяти
6	PRFMC	Переполнение счетчика событий
7	I2	Чтение из области 0x0C00_0000 – 0x0FFF_FFFF
8		
9	H264	Ошибочная ситуация в модуле H264
10-15		

12 Прямой доступ к памяти

Встроенный контроллер DMA осуществляет передачу данных без участия процессора. Данные и код могут быть загружены в процессор через передачу DMA, которая может быть осуществлена между:

– внутренней памятью процессора и внешней памятью, внешней периферией, внутренней памятью;

– внешней памятью и внешними периферийными устройствами;

– внешней памятью и портами связи или между двумя портами связи.

Внешние выводы контроллера приведены в таблице 80.

Таблица 80 – Внешние выводы контроллера DMA

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PB[4]	nDMAR[0]	I/O	Запрос канала 0 контроллера DMA
PB[5]	nDMAR[1]	I/O	Запрос канала 1 контроллера DMA
PB[6]	nDMAR[2]	I/O	Запрос канала 2 контроллера DMA
PB[7]	nDMAR[3]	I/O	Запрос канала 3 контроллера DMA
PB[16]	nDMAR[1]	I/O	Запрос канала 1 контроллера DMA
PB[22]	nDMAR[4]	I/O	Запрос канала 4 контроллера DMA
PB[23]	nDMAR[5]	I/O	Запрос канала 5 контроллера DMA
PB[24]	nDMAR[6]	I/O	Запрос канала 6 контроллера DMA
PB[25]	nDMAR[7]	I/O	Запрос канала 7 контроллера DMA
PB[26]	nDMAR[2]	I/O	Запрос канала 2 контроллера DMA
PB[27]	nDMAR[8]	I/O	Запрос канала 8 контроллера DMA
PB[28]	nDMAR[9]	I/O	Запрос канала 9 контроллера DMA
PB[29]	nDMAR[10]	I/O	Запрос канала 10 контроллера DMA
PB[30]	nDMAR[11]	I/O	Запрос канала 11 контроллера DMA
PB[31]	nDMAR[3]	I/O	Запрос канала 3 контроллера DMA

Контроллер DMA имеет 13 каналов. Четыре канала позволяют выполнять универсальные пересылки типа «память-память». Восемь каналов выполняют пересылки типа «память-периферийное устройство». При этом четыре канала работают с устройствами-приемниками, а другие четыре – с устройствами-передатчиками. Один канал (12-й) является специализированным и может работать только с модулем цифрового коррелятора.

Прямой доступ к памяти (direct memory access, DMA) – механизм передачи данных без исполнения команд в ядре процессора. Встроенный контроллер DMA избавляет процессор от необходимости передачи данных между внутренней памятью и внешними устройствами\памятью или между внутренними периферийными устройствами и внешней или внутренней памятью. Он позволяет ядру процессора указать каналу команду передачи данных и вернуться к нормальной работе, тогда как контроллер DMA выполнит передачу данных в фоновом режиме.

Модуль DMA может быть ведущим или ведомым на шине SOC. При доступе к регистрам DMA со стороны процессорного ядра, модуль DMA выступает в роли ведомого устройства. При работе канала DMA по пересылке данных модуль DMA выступает в роли ведущего устройства и соревнуется в доступе к различным ресурсам с другими ведущими устройствами. Ресурсы, к которым посылает свои запросы контроллер DMA, приведены на рисунке 18. Это интерфейсы внутренней или внешней памяти, а также устройства SOC-шины.

В данном подразделе используются следующие термины:

– *Блок управления передачей (TCB)* – блок из четырех слов, который определяет набор параметров для операций DMA.

– *Регистр TCB* – 128-разрядный регистр, который содержит TCB.

– *Загрузка цепочки TCB* – процесс, при котором контроллер DMA загружает новый TCB из памяти и автоматически инициализирует регистр TCB.

– *Транзакция канала DMA* – последовательность действий канала по пересылке одного операнда.

– *Операнд канала* – порция данных (одно, два или четыре слова), размер которой задается регистром управления и которой оперирует канал в одной транзакции.

– *Режим квитирования* – режим работы канала, в котором инициатором одной транзакции выступает запрос от устройства.

– *Выровненный адрес* – означает, что для двойного слова (64 бита) младший бит адреса всегда равен нулю, а для квадрослова (128 бит) два младших разряда адреса должны быть равны нулю.

12.1 Архитектура контроллера DMA

Контроллер DMA включает в себя 13 каналов, предназначенных для выполнения различных типов передачи:

– каналы с нулевого по третий являются универсальными и позволяют задавать произвольный источник и приемник, которые имеют адрес в карте памяти процессора.

– каналы с четвертого по седьмой жестко закреплены за внутренними периферийными устройствами, которые выполняют функции передатчиков данных (выдача данных на внешние контакты). Т.е. источник данных этих каналов DMA программируемый (внутренняя или внешняя память). Для определения приемника существует два режима работы. При выборе первого режима работы приемник данных и инициатор запросов DMA один и тот же и однозначно определяется значением в регистре DMACFGL. При выборе второго режима работы инициатор запросов определяется регистром DMACFGL, приемник данных – специальным регистром DCA (подробнее см. описание ниже). Каналы с четвертого по седьмой могут выполнять только пересылки память-устройство.

– каналы с восьмого по 11 жестко закреплены за внутренними периферийными устройствами, которые выполняют функции приемников данных (прием информации с внешних контактов). Т.е. приемник данных программируемый (внутренняя или внешняя память). Для определения источника данных существует два режима работы. При выборе первого режима работы источник данных и инициатор запросов DMA один и тот же и

однозначно определяется значением в регистре DMACFGH. При выборе второго режима работы инициатор запросов определяется регистром DMACFGH, источник данных – специальным регистром DCA (подробнее см. описание ниже). Каналы с восьмого по 11 могут выполнять только пересылки устройство-память.

– канал 12 может быть использован совместно с цифровым коррелятором и позволяет организовать передачу данных от 16 каналов коррелятора во внутреннюю память процессора.

Передачи DMA характеризуются направлением потока данных от источника к приемнику. Если в роли источников или приемников выступает память, они описываются регистром ТСВ. Универсальные каналы с нулевого по третий имеют два ТСВ-регистра для описания источника и приемника, каналы с четвертого по 12-й имеют – по одному регистру ТСВ и регистр DCA. Рисунок 48 показывает блок-схему контроллера DMA.

DMA поддерживает так называемые цепочки операций, когда одна запрограммированная в канале пересылка после завершения работы вызывает автоматическое чтение из внутренней памяти новой команды пересылки без участия процессора. Эта техника в первую очередь используется для одного канала DMA, но в некоторых случаях может использоваться и для разных каналов.

Процессор также имеет четыре контакта внешнего запроса DMA (nDMAR3–0), которые позволяют внешним устройствам ввода\вывода запрашивать сервис DMA. В ответ на внешний запрос по входу nDMARi, соответствующий i-й канал контроллера DMA выполняет передачу в соответствии с его инициализацией. Для инициализации канала программа должна записать в блок управления передачей (ТСВ) командное слово.

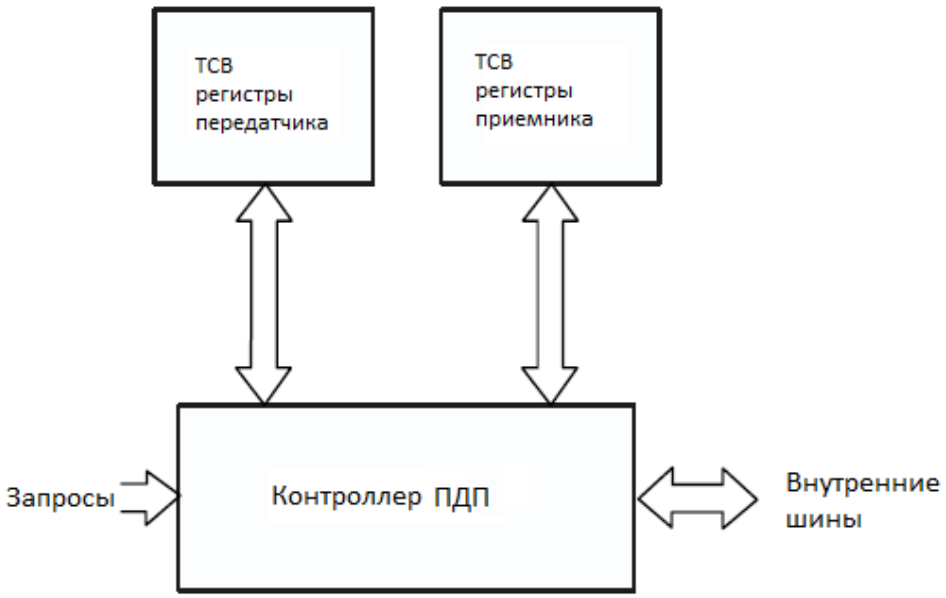


Рисунок 48 – Блок-схема DMA

Блок управления передачей ТСВ является регистром счетверенного слова (128 бит), который содержит информацию, необходимую для работы канала DMA. Вид регистра ТСВ представлен на рисунке 49. В одном канале может присутствовать приемник и передатчик, только приемник или только передатчик. Названия ТСВ передатчика и приемника имеют внутриканальный смысл: ТСВ передатчика передает

данные ТСВ приемнику. Сам же ТСВ передатчик берет данные из источника информации, а ТСВ приемника передает принятые данные в пункт назначения.

В случае ТСВ передатчика, четыре слова содержат данные адреса источника информации (DI), количество слов для передачи (DX/DY), приращение адреса (DX/DY), контрольные разряды (DP).

В случае ТСВ приемника эти четыре слова содержат адрес назначения (DI), количество слов, которое нужно получить (DX/DY), приращение адреса (DX/DY), контрольные разряды (DP).

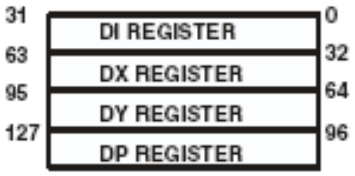


Рисунок 49 – Регистры блока управления передачей DMA

Одна передача (пересылка или транзакция) при работе DMA означает копирование порции данных (операнда) из источника данных в приемник данных. Размер порции данных описывается в регистре управления DP и может быть равен одному, двум или четырем 32-разрядным словам. В регистре DX задается количество слов для передачи. Количество транзакций будет равно количеству слов, деленному на размер порции данных. После каждой транзакции канал осуществляет изменение адреса на величину приращения, задаваемую в регистре DX(DY). Канал DMA может поддерживать двухмерные пересылки данных, когда адрес изменяется по горизонтальной составляющей X и по вертикальной составляющей У. Поэтому в ТСВ присутствуют описатели DX и DY. Обычный режим передачи – одномерный, используется только регистр DX. Регистр DY используется только в двухмерном режиме.

Особенность дополнительного DCA регистра адреса устройства по сравнению, например, с каналами 0-3 состоит в том, что это не полноценный ТСВ регистр, а 64-разрядный регистр, хранящий адрес устройства и некоторую управляющую информацию. Обращение к паре регистров возможно только как к длинному слову. Младший (четный) регистр содержит адрес устройства. Старший регистр содержит управляющую информацию.

Аналогично ТСВ регистру, введем обозначения – DCA:DI и DCA:DP. DCA:DI определяет адрес периферийного устройства. Невозможны никакие модификации этого адреса в процессе выполнения обмена. Он всегда постоянный. Невозможны изменения значения этого регистра при выполнении цепочки операций. Также введены ограничения на разрядность регистра DCA:DI: для каналов 4-7 разрядность регистра 16 бит (старшие 16 бит 32-разрядного адреса предполагаются равными 0x8000), для каналов 8-11 разрядность регистра 32 бита. Регистр DCA:DP использует только три бита – с 31 по 29-й. Любое ненулевое значение в этих разрядах (рекомендуется использовать значение 001b) вызывает включение нового регистра в использование вместо старой функции. После сброса новый регистр управления равен нулю и его функции выключены.

Контроллер поддерживает словную адресацию памяти, и минимальной адресуемой единицей данных для него является 32-разрядное слово.

Соответствие между номером канала и его характеристиками приведено в таблице 81.

Таблица 81 – Особенности программирования каналов контроллера

Номер канала	Источник данных	Приемник данных	Инициирование транзакции
0-3	Программируется в ТСВ передатчика	Программируется в ТСВ приемника	Программируется. Может использовать запрос устройства
4-7	Программируется в ТСВ	Режим 1. Периферийное устройство – инициатор транзакции. Номер устройства задается в регистре конфигурации DMACFGx. Режим 2. Адрес устройства задается в специальном регистре DCA	Всегда только по запросу периферийного устройства. Номер устройства задается в регистре конфигурации DMACFGx
8-11	Режим 1. Периферийное устройство – инициатор транзакции. Номер устройства задается в регистре конфигурации DMACFGx. Режим 2. Адрес устройства задается в специальном регистре DCA	Программируется в ТСВ	Всегда только по запросу периферийного устройства. Номер устройства задается в регистре конфигурации DMACFGx
12	Запись данных цифровым коррелятором в регистр канала	Программируется в ТСВ приемника	Всегда при наличии данных во внутреннем буфере

Каналы контроллер DMA не однотипны. Каналы 0-3 позволяют выполнять пересылки: память-память, память-устройство, устройство-память. Каналы 4-7 позволяют выполнять только пересылки память-устройство, т.е. работают только с устройствами-передатчиками. Каналы 8-11 позволяют выполнять только пересылки устройство-память, т.е. работают только с устройствами-приемниками. При необходимости работы с внутренними устройствами с помощью каналов 0-3 возникают дополнительные ограничения. Каналы 0 и 2 могут быть запрограммированы только для работы с устройствами-передатчиками, а каналы 1 и 3 только с устройствами-приемниками.

Контроллер DMA может быть ведомым на SOC-шине устройств, и его внутренние ресурсы могут быть доступны процессору посредством чтения и записи регистров DMA. Доступ к регистрам контроллера DMA может быть выполнен как с помощью команд пересылки процессора (используя номер регистра), так и с помощью команд загрузки и сохранения (используя адрес в карте памяти).

12.1.1 Регистры управления и статуса

Базовый адрес регистров контроллера в карте памяти периферийных устройств равен **0x8000_0000**. В таблицах описания регистров указано смещение относительно базового адреса.

Общее описание группы регистров управления и статуса приведено в таблице 82.

Таблица 82 – Группа регистров управления контроллер DMA

Имя	Описание	Адрес-смещение	Значение по умолчанию
DCNT	Управление контроллер DMA	0x 0060	0x0000 0000
-	-	0x 0061–0x 0063	-
DCNTST	Управление контроллер DMA, биты установки	0x 0064	Не определено
-	-	0x 0065–0x 0067	-
DCNTCL	Управление контроллер DMA, биты сброса	0x 0068	Не определено
-	-	0x 0069–0x 006B	-
DSTAT	Статус контроллер DMA	0x 006C	0x0000 0000
-	-	0x 006D–0x 006F	-
DSTATCL	Статус контроллер DMA, биты сброса	0x 0070	Не определено
-	-	0x 0071–0x 0077	-
DMACFGL	Задание номера устройства Для каналов с 0 по 7	0x0078	0
DMACFGH	Задание номера устройства Для каналов с 8 по 15	0x0079	0
-	-	0x 007A–0x 007F	-

Особенно отметим функции регистров DMACFGx (DMACFGL, DMACFGH). В процессоре имеется несколько периферийных устройств, способных поддерживать работу совместно с контроллером DMA. Каждому устройству в контроллере DMA ставится в соответствие номер. Для того чтобы некоторому каналу DMA поставить в соответствие выбранное устройство, в определенном поле регистра DMACFGx осуществляется программирование номера устройства. Для каждого канала в регистре конфигурации отведено четыре бита, что позволяет задать 16 источников запросов на обмен (биты 0-3 регистра DMACFGL соответствуют каналу 0, биты 4-7 каналу 1, биты 8-11 каналу 2 и т. д). Регистр DMACFGH обслуживает каналы с 8 по 11. После сброса значение регистра равно нулю и соответствует начальному варианту работы контроллер DMA. Значение в поле регистра соответствует номеру источника запроса.

Имеются две группы источников запросов: передатчики (таблица 83) и приемники (таблица 84).

Таблица 83 – Устройства-передатчики

Номер	Обслуживаемый запрос	Размер данных для обмена
0	Источник запроса определяется номером канала nDMA: 0 – nDMAR0; 1 – nDMAR1; 2 – nDMAR2; 3 – nDMAR3; 4 – LINK 0; 5 – LINK 1; 6 – 7 – 8 – LINK 0; 9 – LINK 1; 10 – 11 –	-
1	Готовность передатчика UART_0 принять данные	32 бита
2	Готовность передатчика UART_1 принять данные	32 бита
3	Готовность передатчика SPI0 принять данные	32 бита
4	Готовность интерфейсас ЖКИ принять данные	128 бит
5	Готовность передатчика SSI0 контроллера принять данные	32 бита
6	Готовность передатчика SSI1 контроллера принять данные	32 бита
7	Готовность буфера NANDF контроллера принять данные для записи	32 или 128 (определяется контроллером NANDF)
8	Готовность модуля UP\DOWN1 принять данные	128 бит
9	Готовность модуля UP\DOWN0 принять данные	128 бит
10	Готовность модуля UP\DOWN2 принять данные	128 бит
11	Готовность модуля UP\DOWN3 принять данные	128 бит
12	Готовность передатчика SPI1 принять данные	
13	Готовность передатчика SPI2 принять данные	
14	Интерфейс Ethernet (передатчик) (для микросхем с ревизии 3)	128 бит
15	Готовность передатчика интерфейса UART2 принять данные (для микросхем с ревизии 3)	32 бита
16	Внешний запрос nDMAR[4] (каналы 4-7) или nDMAR[8] (каналы 8-11)	
17	внешний запрос nDMAR[5] (каналы 4-7) или nDMAR[9] (каналы 8-11)	
18	Внешний запрос nDMAR[6] (каналы 4-7) или nDMAR[10] (каналы 8-11)	
19	Внешний запрос nDMAR[7] (каналы 4-7) или nDMAR[11] (каналы 8-11)	
20	Запрос 0 от таймера 0 с функцией Захвата/ШИМ	

Номер	Обслуживаемый запрос	Размер данных для обмена
21	Запрос 1 от таймера 0 с функцией Захвата/ШИМ	
22	Запрос 2 от таймера 0 с функцией Захвата/ШИМ	
23	Запрос 3 от таймера 0 с функцией Захвата/ШИМ	
24	Запрос 4 от таймера 0 с функцией Захвата/ШИМ	
25	Запрос 0 от таймера 1 с функцией Захвата/ШИМ	
26	Запрос 1 от таймера 1 с функцией Захвата/ШИМ	
27	Запрос 2 от таймера 1 с функцией Захвата/ШИМ	
28	Запрос 3 от таймера 1 с функцией Захвата/ШИМ	
29	Запрос 4 от таймера 1 с функцией Захвата/ШИМ	
30	Запрос от таймера 0 контроллера прерываний	
31	Запрос от таймера 1 контроллера прерываний	

Использование номера запроса от 16 и выше требует определения адреса в дополнительных регистрах каналов.

Далее рассмотрена ситуация при работе с несколькими каналами 0-3, некоторые из которых работают по запросам таймера:

1 Все каналы имеют одинаковый приоритет (не важно – высокий или стандартный) – тогда, каналы, работающие по запросам таймеров, начнут передавать данные только после того, как закончат работу все остальные каналы. При этом, в случаи с каналами 0-3 запросы копятся, и поэтому по завершению работы каналов без запросов, каналы с запросами отправят столько данных сколько запросов накопилось за время их простоя. А затем, каналы с запросами таймеров начнут работать непосредственно по самим запросам таймера. Стоит отметить, что после включения каналов с запросами таймеров, даже во время простоя, статус у этих каналов все равно равен 1.

2 Каналы с запросами имеют высокий приоритет, все остальные –стандартный. В этом случае, каналы с запросами от таймеров работают параллельно с остальными каналами.

Таблица 84 – Устройства-приемники

Номер	Обслуживаемый запрос	Размер данных для обмена
0	Источник запроса определяется номером канала nDMA: 0 – nDMAR0; 1 – nDMAR1; 2 – nDMAR2; 3 – nDMAR3; 4 – LINK 0; 5 – LINK 1; 6 – 7 – 8 – LINK 0; 9 – LINK 1; 10 –	

Номер	Обслуживаемый запрос	Размер данных для обмена
	11 –	
1	Готовность приемника UART_0 выдать данные	32 бита
2	Готовность приемника UART_1 выдать данные	32 бита
3	Готовность приемника SPI0 выдать данные	32 бита
4	Готовность контроллера видеокамеры выдать данные	128 бит
5	Готовность приемника SSI0 контроллера выдать данные	32 бита
6	готовность приемника SSI1 контроллера выдать данные	32 бита
7	Готовность буфера NANDF контроллера выдать данные	32 или 128 (определяется контроллером NANDF)
8	Готовность модуля UP\DOWN1 выдать данные	128 бит
9	Готовность модуля UP\DOWN0 выдать данные	128 бит
10	Готовность модуля UP\DOWN2 выдать данные	128 бит
11	Готовность модуля UP\DOWN3 выдать данные	128 бит
12	Готовность приемника SPI1 выдать данные	
13	Готовность приемника SPI2 выдать данные	
14	Интерфейс Ethernet (передатчик) (для микросхем с ревизии 3)	128 бит
15	Готовность передатчика интерфейсас UART2 принять данные (для микросхем с ревизии 3)	32 бита
16	Внешний запрос nDMAR[4] (каналы 4-7) или nDMAR[8] (каналы 8-11)	
17	Внешний запрос nDMAR[5] (каналы 4-7) или nDMAR[9] (каналы 8-11)	
18	Внешний запрос nDMAR[6] (каналы 4-7) или nDMAR[10] (каналы 8-11)	
19	Внешний запрос nDMAR[7] (каналы 4-7) или nDMAR[11] (каналы 8-11)	
20	Запрос 0 от таймера 0 с функцией Захвата/ШИМ	
21	Запрос 1 от таймера 0 с функцией Захвата/ШИМ	
22	Запрос 2 от таймера 0 с функцией Захвата/ШИМ	
23	Запрос 3 от таймера 0 с функцией Захвата/ШИМ	
24	Запрос 4 от таймера 0 с функцией Захвата/ШИМ	
25	Запрос 0 от таймера 1 с функцией Захвата/ШИМ	
26	Запрос 1 от таймера 1 с функцией Захвата/ШИМ	
27	Запрос 2 от таймера 1 с функцией Захвата/ШИМ	
28	Запрос 3 от таймера 1 с функцией Захвата/ШИМ	
29	Запрос 4 от таймера 1 с функцией Захвата/ШИМ	
30	Запрос от таймера 0 контроллера прерываний	
31	Запрос от таймера 1 контроллера прерываний	

Каждое из устройств имеет для передачи и (или) для приема встроенный буфер FIFO. При работе с устройствами с помощью каналов 0-3 необходимо задать источник запроса с помощью регистра конфигурации, а также обязательно прописать адрес источника или приемника в соответствующем регистре канала. При работе с каналами 4-11 программируется только номер источника запроса.

12.1.2 Группа регистров TCB каналов 0-3 контроллер DMA

Доступ к регистрам данной группы осуществляется только как к счетверенным словам, т.е. операции выполняются сразу с четырьмя регистрами.

Таблица 85 – Группа регистров TCB каналов 0-3

Имя	Описание	Адрес-смещение	Значение по умолчанию
DCS0:DI	канал 0, адрес источника	0x0000	0
DCS0:DX	канал 0, модификатор X источника	0x0001	0x01000004
DCS0:DY	канал 0, модификатор У источника	0x0002	0
DCS0:DP	канал 0, управление источником	0x0003	0
DCD0:DI	канал 0, адрес приемника	0x0004	0
DCD0:DX	канал 0, модификатор X приемника	0x0005	0x01000001
DCD0:DY	канал 0, модификатор У приемника	0x0006	0
DCD0:DP	канал 0, управление приемником	0x0007	0
DCS1:DI	канал 1, адрес источника	0x0008	0
DCS1:DX	канал 1, модификатор X источника	0x0009	0x01000004
DCS1:DY	канал 1, модификатор У источника	0x000A	0
DCS1:DP	канал 1, управление источником	0x000B	0
DCD1:DI	канал 1, адрес приемника	0x000C	0
DCD1:DX	канал 1, модификатор X приемника	0x000D	0x01000001
DCD1:DY	канал 1, модификатор У приемника	0x000E	0
DCD1:DP	канал 1, управление приемником	0x000F	0
DCS2:DI	канал 2, адрес источника	0x0010	0
DCS2:DX	канал 2, модификатор X источника	0x0011	0x01000004
DCS2:DY	канал 2, модификатор У источника	0x0012	0
DCS2:DP	канал 2, управление источником	0x0013	0
DCD2:DI	канал 2, адрес приемника	0x0014	0
DCD2:DX	канал 2, модификатор X приемника	0x0015	0x01000001
DCD2:DY	канал 2, модификатор У приемника	0x0016	0
DCD2:DP	канал 2, управление приемником	0x0017	0
DCS3:DI	канал 3, адрес источника	0x0018	0
DCS3:DX	канал 3, модификатор X источника	0x0019	0x01000004
DCS3:DY	канал 3, модификатор У источника	0x001A	0
DCS3:DP	канал 3, управление источником	0x001B	0
DCD3:DI	канал 3, адрес приемника	0x001C	0
DCD3:DX	канал 3, модификатор X приемника	0x001D	0x01000001
DCD3:DY	канал 3, модификатор У приемника	0x001E	0
DCD3:DP	канал 3, управление приемником	0x001F	0

Особенности при работе с несколькими каналами 0-3, некоторые из которых работают по запросам таймера:

1 В случае, когда все каналы имеют одинаковый приоритет (высокий или стандартный), каналы, работающие по запросам таймеров, начнут передавать данные только после того, как закончат работу все остальные каналы. При этом, в случаи с

каналами 0-3 запросы копятся, и по завершении работы каналов без запросов, каналы с запросами отправят столько данных сколько запросов накопилось за время их простоя. Затем, каналы с запросами таймеров, начнут работать непосредственно по самим запросам таймера. После включения каналов с запросами таймеров, даже во время простоя, статус у этих каналов все равно равен 1.

2 В случае, когда каналы с запросами имеют высокий приоритет, все остальные – стандартный, каналы с запросами от таймеров, работают параллельно с остальными каналами.

12.1.3 Группа регистров TCB каналов 4-7 контроллер DMA

Доступ к регистрам DCx осуществляется только как к счетверенным словам, т.е. операции выполняются сразу с четырьмя регистрами. Доступ к регистрам DCAx осуществляется только как к двойным словам, т.е. операции выполняются сразу с двумя регистрами.

Таблица 86 – Группа регистров TCB каналов 4-7

Имя	Описание	Адрес-смещение	Значение по умолчанию
DC4:DI	канал 4, адрес источника	0x 0020	Не определено
DC4:DX	канал 4, модификатор X источника	0x 0021	Не определено
DC4:DY	канал 4, модификатор У источника	0x 0022	Не определено
DC4:DP	канал 4, управление источником	0x 0023	0
DC5:DI	канал 5, адрес источника	0x 0024	Не определено
DC5:DX	канал 5, модификатор X источника	0x 0025	Не определено
DC5:DY	канал 5, модификатор У источника	0x 0026	Не определено
DC5:DP	канал 5, управление источником	0x 0027	0
DC6:DI	канал 6, адрес источника	0x 0028	Не определено
DC6:DX	канал 6, модификатор X источника	0x 0029	Не определено
DC6:DY	канал 6, модификатор У источника	0x 002A	Не определено
DC6:DP	канал 6, управление источником	0x 002B	0
DC7:DI	канал 7, адрес источника	0x 002C	Не определено
DC7:DX	канал 7, модификатор X источника	0x 002D	Не определено
DC7:DY	канал 7, модификатор У источника	0x 002E	Не определено
DC7:DP	канал 7, управление источником	0x 002F	0
DCA4:DI	Канал 4, адрес устройства-приемника	0x0030	Не определено
DCA4:DP	Канал 4, включение	0x0031	0
DCA5:DI	Канал 5, адрес устройства-приемника	0x0032	Не определено
DCA5:DP	Канал 5, включение	0x0033	0
DCA6:DI	Канал 6, адрес устройства-приемника	0x0034	Не определено
DCA6:DP	Канал 6, включение	0x0035	0
DCA7:DI	Канал 7, адрес устройства-приемника	0x0036	Не определено
DCA7:DP	Канал 7, включение	0x0037	0
		0x0038 – 0x003F	

Устройство всегда находится на SOC-шине внутренних периферийных устройств.

Значение 001b в разрядах DCA:DP[31:29] вызывает включение регистра DCA (регистр задания адреса периферийного устройства по которому можно выполнить запись данных). После сброса регистр DCA управления равен нулю и его функции выключены. Направление передачи данных при работе каналов 4-7 показано на рисунке 50.

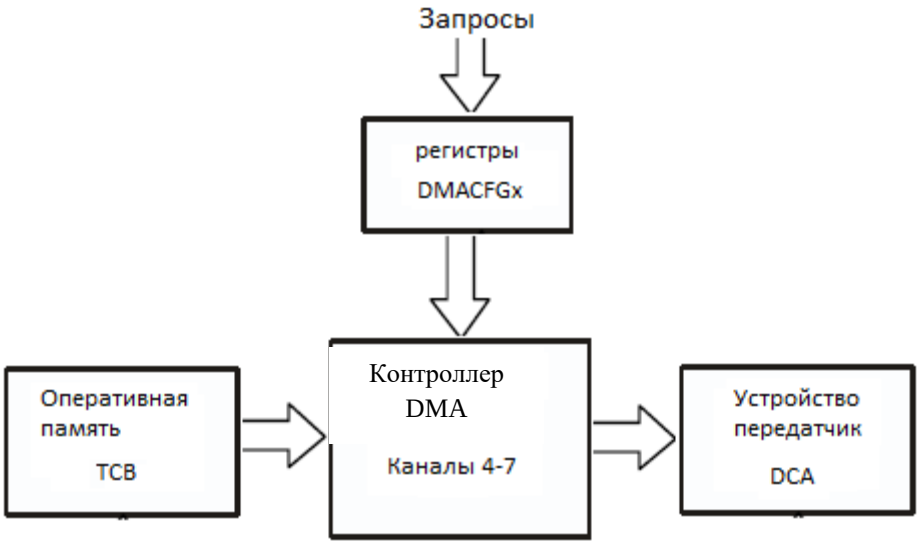


Рисунок 50 – Передача данных в каналах 4-7

Используя запросы от таймеров или иных источников, можно запрограммировать в каналах 4-7 произвольные пересылки из любого типа памяти в любое **внутреннее** устройство SOC-шины.

Если необходимо обращение к устройству-приемнику на внешней шине должны использоваться только каналы 0-3.

12.1.4 Группа регистров TCB каналов 8-11

Доступ к регистрам DCx осуществляется только как к счетверенным словам, т.е. операции выполняются сразу с четырьмя регистрами. Доступ к регистрам DCAx осуществляется только как к двойным словам, т.е. операции выполняются сразу с двумя регистрами.

Таблица 87 – Группа регистров TCB каналов 8-11

Имя	Описание	Адрес-смещение	Значение по умолчанию
DC8:DI	канал 8, адрес приемника	0x0040	0
DC8:DX	канал 8, модификатор X приемника	0x0041	0
DC8:DY	канал 8, модификатор У приемника	0x0042	0
DC8:DP	канал 8, управление приемником	0x0043	0
DC9:DI	канал 9, адрес приемника	0x0044	0
DC9:DX	канал 9, модификатор X приемника	0x0045	0
DC9:DY	канал 9, модификатор У приемника	0x0046	0
DC9:DP	канал 9, управление приемником	0x0047	0

Имя	Описание	Адрес-смещение	Значение по умолчанию
DC10:DI	канал 10, адрес приемника	0x0048	0
DC10:DX	канал 10, модификатор X приемника	0x0049	0
DC10:DY	канал 10, модификатор У приемника	0x004A	0
DC10:DP	канал 10, управление приемником	0x004B	0
DC11:DI	канал 11, адрес приемника	0x004C	0
DC11:DX	канал 11, модификатор X приемника	0x004D	0
DC11:DY	канал 11, модификатор У приемника	0x004E	0
DC11:DP	канал 11, управление приемником	0x004F	0
DCA8:DI	Канал 8, адрес устройства-источника	0x0050	0
DCA8:DP	Канал 8, включение	0x0051	0
DCA9:DI	Канал 9, адрес устройства- источника	0x0052	0
DCA9:DP	Канал 9, включение	0x0053	0
DCA10:DI	Канал 10, адрес устройства- источника	0x0054	0
DCA10:DP	Канал 10, включение	0x0055	0
DCA11:DI	Канал 11, адрес устройства- источника	0x0056	0
DCA11:DP	Канал 11, включение	0x0057	0

После сброса регистр DCA:DP каналов 8-11 равен нулю, и новые функции выключены. Если записать в регистр DCA:DP ненулевую информацию, то канал 8-11 будет использовать в качестве адреса источника значение из регистра DCA:DI.

Значение, записываемое в регистр управления DCA:DP имеет, кроме функции включения, дополнительный смысл. Если поле управление равно четырем, обращение происходит к устройству на SOC-шине (если адрес соответствует SOC-шине) или на внешнюю шину процессора. Если значение поля равно единице, то обращение выполняется к памяти ядра процессора. Таким образом, каналы 8-11 могут выполнять пересылку «устройство_на_внешней_шине – память». Значение регистра DCA:DI в процессе работы канала не изменяется. Направление передачи данных при работе каналов 8-11 показано на рисунке 51.

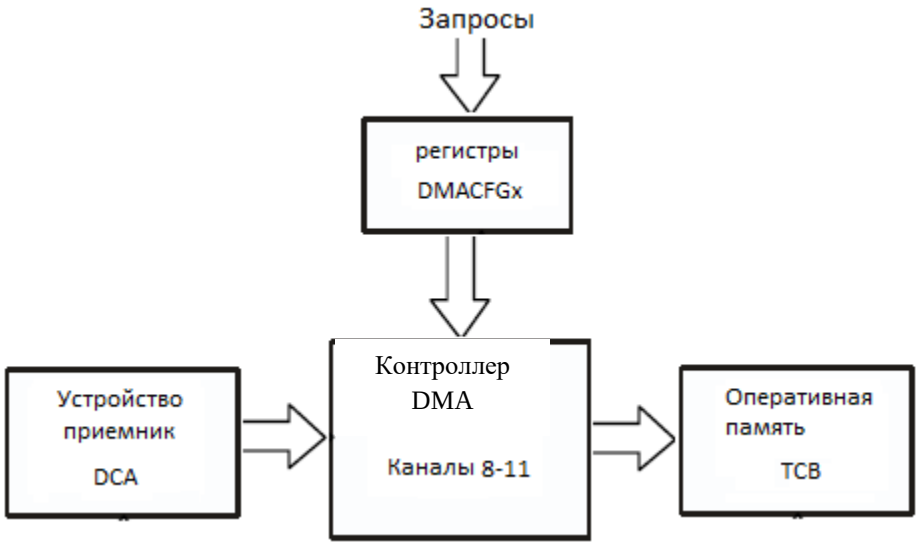


Рисунок 51 – Передача данных в каналах 8-11

Если каналы 4-11 завершают свою работу в нормальном режиме, т.е. по завершении происходит очистка управляющего поля регистра DSP, DDP, то и управляющее поле дополнительного регистра DCA:DP также сбрасывается.

В противном случае необходима дополнительная очистка регистра. Запись в регистр DCA никак каналом не отслеживается и не влияет на последующее включение-выключение канала.

Последовательность использования функций каналов 4-11 должна быть следующей:

1 При выключенном канале необходимо записать в регистр конфигурации DMACFGx номер источника запроса. Используются 5 бит номера. Старшие биты расширения номера хранятся в разрядах 27-16 регистра DMACFGH. По одному для каждого из каналов 11-0.

2 В регистре DCA выбранного канала необходимо записать адрес устройства и значение управляющего кода.

3 Включить канал в работу стандартным образом, т.е. записать активное значение TCB канала.

Назначение разрядов регистра DMACFGx каналам контроллера описано в таблице 88. Поле H относится к регистру DMACFGH, а поле L – к регистру DMACFGL.

Таблица 88 – Назначение разрядов регистра DMACFGx

Канал	Биты DMACFGx
0	H[16].L[3:0]
1	H[17].L[7:4]
2	H[18].L[11:8]
3	H[19].L[15:12]
4	H[20].L[19:16]
5	H[21].L[23:20]
6	H[22].L[27:24]
7	H[23].L[31:28]
8	H[24].H[3:0]
9	H[25].H[7:4]
10	H[26].H[11:8]
11	H[27].H[15:12]

12.1.5 Группа регистров TCB канала 12

Таблица 89 – Группа регистров TCB канала 12

Имя	Описание	Адрес-смещение	Значение по умолчанию
DC12:DI	Канал 12, адрес приемника	0x0058	0x0000 0000
DC12:DX	Канал 12, модификатор X приемника	0x0059	0x0100 0001
DC12:DY	Канал 12, модификатор У приемника	0x005A	0x0000 0000
DC12:DP	Канал 12, управление приемником	0x005B	0x5380 0000

Канал 12 также называется каналом AutoDMA в связи с тем, что при выполнении записи данных в него ведущим устройством, он автоматически выполняет пересылку принятых данных в запрограммированный в него приемник. Для выполнения операций записи в канал 12 используются определенные регистры.

12.1.6 Группа регистров AutoDMA

Регистры данной группы относятся к каналу 12 контроллера DMA и используются для реализации режима «подчиненный» (slave) DMA. Данные регистры могут быть доступны через адресное пространство периферийных устройств.

Таблица 90 – Группа регистров AutoDMA

Имя	Описание	Адрес-смещение	Значение по умолчанию
AUTODMA0:0	Регистр AutoDMA 0, разряды данных 31–0	0x 03E0	Не определено
AUTODMA0:1	Регистр AutoDMA 0, разряды данных 63–32	0x 03E1	Не определено
AUTODMA0:2	Регистр AutoDMA 0, разряды данных 95–64	0x 03E2	Не определено
AUTODMA0:3	Регистр AutoDMA 0, разряды данных 127–96	0x 03E3	Не определено
-	-	0x 03E4– 0x 03FF	-

12.2 Настройка передач DMA

Операции DMA могут программироваться ядром процессора или внешним хост-устройством. Операция программируется записью в регистры TCB. Регистр TCB – регистр счетверенных слов, описывающий передачу блока DMA. Канал DMA настраивается записью счетверенного слова в каждый из регистров TCB. Каждый регистр должен быть загружен начальным адресом для блока, приращением адреса, количеством слов и управляющей информацией.

Если TCB запрограммирован на генерацию прерывания, запрос прерывания формируется, когда блок данных полностью передан.

Регистры TCB могут быть доступны только как счетверенные слова.

12.2.1 Регистры блока управления передачей (TCB)

Каждый TCB-регистр имеет длину 128 бит и разделен на четыре 32-битных регистра:

- регистр индекса (DI);
- регистр X количества и приращения (DX);
- регистр Y количества и приращения (DY);
- регистр управления и указателя цепочки (DP).

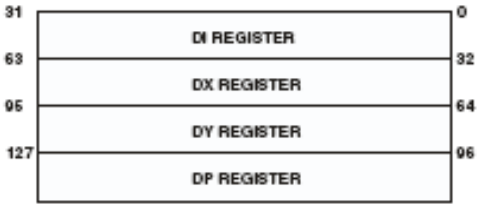


Рисунок 52 – Регистры блока управления передачей

Эти четыре регистра образуют счетверенное слово TCB, которое может быть загружено как выровненное счетверенное слово из внутренней памяти посредство цепочки или с помощью ядра.

Для инициирования нового обмена, после завершения текущего, программа должна записать новые параметры в регистры TCB.

Таблица 91 – Каналы DMA и связанные регистры TCB

Канал DMA	Регистры TCB	Описание
0	DCS0 DCD0	TCB регистр источника канала 0 TCB регистр назначения канала 0
1	DCS1 DCD1	TCB регистр источника канала 1 TCB регистр назначения канала 1
2	DCS2 DCD2	TCB регистр источника канала 2 TCB регистр назначения канала 2
3	DCS3 DCD3	TCB регистр источника канала 3 TCB регистр назначения канала 3
4	DC4	TCB регистр источника канала 4
5	DC5	TCB регистр источника канала 5
6	DC6	TCB регистр источника канала 6
7	DC7	TCB регистр источника канала 7
8	DC8	TCB регистр назначения канала 8
9	DC9	TCB регистр назначения канала 9
10	DC10	TCB регистр назначения канала 10
11	DC11	TCB регистр назначения канала 11
12	DC12	TCB регистр назначения канала 12

Регистр DI

32-разрядный регистр DI содержит начальный адрес блока данных и используется для прямого доступа к памяти. Он может указывать на адрес внешней памяти или внутренней памяти.

Регистр DX

32-разрядный регистр DX имеет два поля:

– DXM – младшие биты (с 0-го по 15-й или с 0-го по 21-й) хранят значение модификатора адреса (DX_MODIFY), которое используется для изменения адреса после каждой транзакции.

– DXC – старшие биты (с 16-го по 31-й или с 22-го по 31-й) хранят значение количества слов (DX_COUNT) в блоке данных который необходимо передать.

Например, если необходимо передать 16 слов данных, непрерывно размещенных в памяти, порциями по четыре слова в каждой передаче, параметр DXC будет равен 16, а значение DXM будет равно четырем. Длина операнда (порции данных) в регистре DP устанавливается равной счетверенному слову.

Существуют ограничения, которые следует принимать во внимание при программировании TCB:

– указатель DI должен содержать адрес, выровненный на границе операнда, определяемого в регистре DP;

– значение DXM должно быть кратно размеру операнда;

– значение DXC должно быть кратным длине операнда.

Выбор длины полей модификатора и количества определяется типом обмена, задаваемым полем TY в регистре DP.

Регистр DY

32-разрядный регистр DY используется вместе с регистром DX, когда двумерный режим DMA разрешен. Регистр DY содержит два поля:

– DYM – младшие биты (с 0-го по 15-й или с 0-го по 21-й) хранят значение модификатора адреса (DY_MODIFY), которое используется для изменения адреса после завершения передачи блока по координате X. Модификация Y – разница между адресом последнего элемента данных в строке и адресом первого элемента в следующей строке.

– DYC – старшие биты (с 16-го по 31-й или с 22-го по 31-й) хранят значение количества попыток (DY_COUNT) передач по координате X.

Если двумерный (2D) режим DMA запрещен, регистр DY загружается значением 0 или любым другим допустимым значением.

Выбор длины полей модификатора и количества определяется типом обмена, задаваемым полем TY в регистре DP.

Регистр DP

Данный регистр задает режим работы передатчика\приемника канала. Подробное описание разрядов регистра приведено в таблице 92.

Таблица 92 – Описание разрядов регистра DP

Бит	Имя	Назначение
18:0	CHPT	Указатель цепочки. Эти разряды включают в себя разряды 20–2 адреса внутренней памяти, где находится значение следующего регистра TCB. Разряды 1–0 адреса не указываются, поскольку адрес должен быть выровненным на границе квадрослова, т.е биты равны нулю. Например, если содержимое следующего TCB сохраняется в памяти 0x87128-0x8712B тогда поле CHPT будет содержать b0010_0001_1100_0100_1010 (0x21C4A)

Бит	Имя	Назначение
21:19	CHTG	Канал-приемник следующей цепочки. Это поле имеет значение *только для каналов 4-11* и указывает в какой канал DMA будет загружено новое значение TCB: 000 – канал 8; 001 – канал 9; 010 – канал 10; 011 – канал 11; 100 – канал 4; 101 – канал 5; 110 – канал 6; 111 – канал 7
22	CHEN	Разрешение загрузки следующей цепочки: 1 – разрешено; 0 – запрещено. При разрешении загрузки после окончания передачи всего блока данных канал выполнит чтение новой TCB из внутренней памяти и загрузит ее в канал назначения
23	DRQ	Разрешение анализа запроса от периферийного устройства: 1 – разрешено; 0 – запрещено. Если разрешено, канал выполняет одну транзакцию только по запросу. Если запрещено, канал выполняет передачу всего блока без ожидания запроса. Бит имеет значение для каналов 0-3 и позволяет им анализировать запросы от устройств. Для каналов 4-11 бит не имеет значения, т.к. они всегда работают только по запросу
24	INT	Генерация запроса прерывания после окончания работы канала: 1 – разрешено; 0 – запрещено
26:25	LEN	Длина передаваемых данных (операнда) в одном цикле обмена: 00 – резерв; 01 – слово 32 бита; 10 – длинное слово 64 бита; 11 – квадрослово 128 бит. Длина операнда накладывает ограничения на адрес источника или приемника – они всегда должны быть выравнены. Также накладываются ограничение на модификаторы и количество передаваемых слов в X
27	2D	Включение режима двухмерной пересылки: 1 – двумерная пересылка; 0 – одномерная пересылка
28	PR	Приоритет циклов обмена: 1 – высокий; 0 – обычный. Уровень приоритета используется во время арбитража каналов между собой. Также уровень приоритета оказывает влияние на контакт DPA внешней шины

Бит	Имя	Назначение
31:29	TY	Тип обмена (выбор источника или приемника): 000 – канал выключен; 001 – линк порт (только в случае передачи линк-линк); 010 – внутренняя память (16\16); 011 – внутренняя память (22\10); 100 – внешняя память (16\16); 101 – зарезервировано; 110 – загрузочное EPROM. nBMS контакт; 111 – внешняя память (22\10)

Ограничения при программировании регистра DP

Неправильное задание конфигурации TCB может вызвать генерацию ошибки каналом. На использование *регистра управления TCB* накладываются *следующие ограничения*:

– Установка длины операнда (LEN). В каналах 0-3 поле LEN должно быть одинаковым в обоих TCB.

– Если поле TY источника или получателя TCB имеет тип 6 (загрузочная EPROM), поле LEN должно быть 0x1 (стандартное слово).

Задание полей количества (DX и DY) имеет следующие ограничения:

– Для каналов 0-3 общее количество слов передатчика должно равняться количеству слов приемника.

– Для обычного режима обмена количество слов равно DXC.

– Для двухмерного режима количество слов равно DXC•DYC. При этом бит 2D может быть установлен в одном из TCB и сброшен в другом (для каналов с нулевого по третий).

– Нет ограничений для регистра количества в каналах DMA 4-11. Во всех случаях, если канал не простаивает, DXC не может быть нулевым. Если 2D режим установлен, DYC не может быть «0» или «1»

Из поля TY регистра управления видно, что можно задать семь типов обмена. Однако в зависимости от номера канала существует *ограничения в выборе типа*. Разрешенные комбинации (отмечены +) типов в каналах 0-3 приведены в таблице 93.

Таблица 93 – Допустимые комбинации передач каналов с 0 по 3

Тип передатчика	Тип приемника						
	1	2	3	4	5	6	7
1							
2				+		+	+
3				+		+	+
4		+	+	+		+	+
5							+
6		+	+				
7		+	+	+	+	+	+

Все другие комбинации при их установке в каналах 0-3 вызовут генерацию ошибки канала.

Для каналов 4-7 также существует ограничение в выборе типа обмена: в качестве источника данных могут быть выбраны только внутренняя (тип 2, 3) или внешняя (тип 4, 7) память.

Ограничения в выборе типа для каналов 8-11: в качестве приемника данных могут быть выбраны только внутренняя (2, 3), внешняя (4, 7) память или порт связи (тип 1).

Для канала 12 единственным типом может быть только внутренняя память (тип 2, 3).

Ограничения запросов DMA

В каналах 0-3 бит DRQ должен быть одинаковым в обоих регистрах TCB.

Ограничения выравнивания

Если поле LEN равно двум (двойное слово), следующие поля должны быть четными (бит 0 очищен):

– DI;
– DXM;
– DYM (при необходимости);
– DXC.

Если поле LEN равно трем (счетверенное слово) значения этих же полей должны быть кратны чеытрем (биты 1–0 очищены).

Для типа 6 (загрузочная EPROM) данные регистры должны быть все кратны четырем (биты 1–0 очищены).

Ограничения диапазона адресов

Следующая связь должна существовать в любой момент работы TCB между полем TY в регистре DP и адресом:

– TY = 1 (канал связи) ⇒ адрес – один из регистров передачи порта связи.

– TY = 2 или 3 (внутренняя память) ⇒ адрес – в диапазоне внутренней памяти и не в регистрах.

– TY = 4 или 7(внешняя память) ⇒ адрес – в диапазоне внешней памяти (биты 31-22 отличны от нуля).

12.2.2 Регистр статуса DMA (DSTAT/DSTATC)

Регистр статуса DSTAT предназначен только для чтения и отражает текущее состояние каналов контроллера. Регистр имеет разрядность 64 бита и два адреса доступа. Второму адресу соответствует имя DSTATCL регистра. При чтении регистра DSTATCL все коды ошибок в регистре статуса очищаются, и состояние канала изменяется на состояние выключено. Регистры статуса DMA (DSTAT, DSTATCL) могут быть доступны только как двойное или счетверенное слово. В регистре состояния каждому каналу контроллера отведено три бита. С помощью этих бит кодируется состояние канала. Возможные значение состояния канала:

– 000 – выключен;
– 001 – активен (работает);

– 010 – завершил работу;
– 011 – резерв;
– 100 – ошибка. Инициализация активного канала;
– 101 – ошибка. Запрещенная конфигурация канала;
– 110 – резерв;
– 111 – ошибка. Некорректный адрес.

Назначение разрядов регистра состояния приведено в таблице 94.

Таблица 94 – Биты регистра состояний

Бит	Имя	Описание
2:0	CH0	Состояние канала 0
5:3	CH1	Состояние канала 1
8:6	CH2	Состояние канала 2
11:9	CH3	Состояние канала 3
14:12	CH4	Состояние канала 4
17:15	CH5	Состояние канала 5
20:18	CH6	Состояние канала 6
23:21	CH7	Состояние канала 7
31:24	-	-
34:32	CH8	Состояние канала 8
37:35	CH9	Состояние канала 9
40:38	CH10	Состояние канала 10
43:41	CH11	Состояние канала 11
49:44	-	-
52:50	CH12	Состояние канала 12
63:53	-	-

Активный статус DMA устанавливается, когда канал DMA включен (когда поле TY регистра DPx установлено).

Статус завершения DMA устанавливается, когда последняя транзакция DMA завершена.

Статус ошибки DMA устанавливается, если обнаружено запрещенное или ошибочное состояние.

В случае, если обнаруживается ошибочное состояние, DMA может генерировать аппаратное прерывание, если оно включено, и биты статуса могут быть очищены только чтением из регистра DSTATC. Регистры DSTAT и DSTATC должны читаться как сдвоенное или счетверенное слово. Чтение этих регистров как стандартное слово недопустимо.

12.2.3 Регистр управления DMA

Биты регистра управления выполняют единственную функцию – приостановка работы соответствующего им канала DMA. Подробное описание разрядов регистра управления приведено в таблице 95.

Таблица 95 – Биты регистра управления

Бит	Имя	Описание
0	DMA0_P	Пауза (если 1) для канала 0 DMA
1	DMA1_P	Пауза (если 1) для канала 1 DMA
2	DMA2_P	Пауза (если 1) для канала 2 DMA
3	DMA3_P	Пауза (если 1) для канала 3 DMA
4	DMA4_P	Пауза (если 1) для канала 4 DMA
5	DMA5_P	Пауза (если 1) для канала 5 DMA
6	DMA6_P	Пауза (если 1) для канала 6 DMA
7	DMA7_P	Пауза (если 1) для канала 7 DMA
9:8		-
10	DMA8_P	Пауза (если 1) для канала 8 DMA
11	DMA9_P	Пауза (если 1) для канала 9 DMA
12	DMA10_P	Пауза (если 1) для канала 10 DMA
13	DMA11_P	Пауза (если 1) для канала 11 DMA
15:14		
16	DMA12_P	Пауза (если 1) для канала 12 DMA
29:17		-
30	nDMAR_ALT	Если бит 30 регистра установить в 1, то запросы nDMAR[3:1] используют альтернативные входы согласно таблице 1
31	PMOD	Управление режимом паузы

Регистр управления DMA имеет три адреса:

– DCNT – регулярный адрес для чтения и записи.

– DCNTST – адрес для установки бит. Доступен только по записи. Запись «1» устанавливает соответствующий бит, а запись «0» не изменяет его значения.

– DCNTCL – адрес для очистки бит. Доступен только по записи. Запись «0» сбрасывает значение бита, а запись «1» не изменяет его значение.

Назначение бита паузы состоит в том, что с его установкой в «1» соответствующий канал DMA приостанавливает свою работу и возобновляет ее только после того, как бит станет равен нулю.

Назначение бита PMOD состоит в том, что, если установить бит паузы при установленном бите PMOD регистра управления, блокировки SOC-шины не будет. Вместо этого в регистре состояния выбранного канала будет отражаться код 3 до момента завершения всех начатых транзакций. Это позволяет отследить момент завершения операций каналом и затем безопасно перепрограммировать его. Отсутствие блокировки SOC-шины положительно влияет на оперативность действий процессора в случае прерывания.

Регистры управления DMA (DCNT, DCNTST, DCNTCL) могут быть доступны как нормальное, удлиненное и счетверенное слово. Начальное значение DCNT после сброса равно нулю.

Управление битом паузы каналами 4-12 довольно простое, т.к. установка бита паузы для них означает остановку генерации транзакций каналом. При этом уже запущенные каналами транзакции спокойно завершаются.

Управление битом паузы для каналов 0-3 более сложное. Это связано с тем, что транзакции этих каналов состоят из двух самостоятельных частей: транзакции чтения данных из источника в буфер DMA и затем транзакции записи данных из буфера по адресу приемника. Канал может запускать до восьми транзакций чтения, прежде чем хотя бы одна транзакция записи выполнится. Бит паузы влияет только за запуск транзакций чтения. Но установка бита паузы не означает, что канал остановил работу. Это будет сделано, только когда все запущенные транзакции чтения загрузят в буфер данные, и затем выполнятся все соответствующие им транзакции записи. Изменение настроек канала после установки бита паузы приведет к потере прочитанных данных, т.к. операции записи могут быть сброшены. Чтобы этого не произошло, после установки бита паузы для каналов 0-3 оценивается наличие у канала DMA незавершенных транзакций записи, и если таковые имеются – канал генерирует сигнал блокировки доступа к SOC-шине со стороны ядра. Таким образом, если процессор сразу после записи бита паузы пытается произвести операцию на SOC-шине, он может быть остановлен на некоторое время. Данное время может быть значительным, если канал работает с внешней памятью. Это обстоятельство нужно учитывать при работе с каналами 0-3. По возможности не стоит использовать бит паузы для каналов 0-3, если источником данных для каналов является внешняя память и при этом необходима высокая скорость реакции на прерывание.

Установка бита PMOD регистра управления в «1» отразится на поведении контроллера DMA при установке бита паузы. Если установить бит паузы при установленном бите PMOD регистра управления, то блокировки SOC-шины не произойдет. Вместо этого в регистре состояния выбранного канала будет отражаться код 3 до момента завершения всех начатых транзакций. Это позволяет отследить момент завершения операций каналом и затем безопасно перепрограммировать его. Отсутствие блокировки SOC-шины положительно влияет на оперативность действий процессора в случае прерывания.

Пример

Рассмотрим пример использования каналов 4-11 для организации пересылок. Допустим, необходимо переслать массив из восьми чисел из одной области внутренней оперативной памяти в другую область внутренней оперативной памяти, используя при этом в качестве инициатора пересылки таймер 0 процессора 1986. Задача решается с использованием четвертого и восьмого каналов, а также с использованием интерфейса SPI1.

Обозначения:

```
#define base_DST_ch74   0x80000030
#define base_SRC_ch118  0x80000050
#define base_DMACFG     0x80000078
```

Наш массив для пересылки

```
.var spm[8] = {0x11111111, 0x11111122,0x11111333,0x11114444,
                0x11155555, 0x11666666,0x17777777,0x88888888};
```

Предположим, что таймер 0 включен, работает и с определенным периодом формирует запросы к каналам DMA. Поскольку каналы DMA не включены и не настроены на прием запросов от таймера, то никаких событий не происходит. Включим SPI1 интерфейс для передачи 32-разрядных данных в тестовом режиме с включенной обратной связью.

```
        j8 = base_SPI1;;

xr0 = 0x1001f;;
[j8+0] = xr0;; // cr1
xr0 = 0x0d;;   //
[j8+1] = xr0;; // cr2
xr0 = 0xff;;
[j8+3] = xr0;; // st
```

Определим источники запросов для активации четвертого и восьмого каналов. Четвертый канал будем использовать для пересылки данных из памяти в передатчик SPI1 по запросу таймера 0 (запрос 0). Восьмой канал будем использовать для пересылки данных от приемника SPI1 во внутреннюю память. Итого код активации четвертого канала равен 20 (см. таблицу 83), а код активации восьмого – 12 (см. таблицу 84). Формируем 64-разрядный код и пишем его в регистр конфигурации:

```
            j8 = base_DMACFG;;
xr0 = 0x00040000;;   // GTMR0 req
                xr1 = 0x0010000c;;
l[j8+0] = xr1:0;;
```

Определяем устройство в которое четвертый канал будет записывать данные для передачи. Адрес устройства есть буфер данных SPI1. Активируем регистр DCA записав в поле активации значение 001b.

```
                j8 = base_DST_ch74;;
xr0 = 0x80000362;;   // dest  SPI1
                xr1 = 0x20000000;;// код активации
l[j8+0] = xr1:0;;
```

Определяем устройство, из которого восьмой канал будет читать данные. Отметим, что в данном случае можно было бы обойтись без новых функций, т.к. источник запроса и обслуживающее устройство – одно и то же устройство. Но будем использовать регистр DCA. Запишем адрес-источник данных – это буфер данных приемника SPI1. Код активации равен 100b, что означает обращение к устройству на SOC-шине.

```
j8 = base_SRC_ch118;;
xr0 = 0x80000362;; // src  SPI1
                xr1 = 0x80000000;;// код активации
l[j8+0] = xr1:0;;
```

Включаем каналы «стандартным» образом. Включаем четвертый канал для передачи. Адрес источника – это массив SPM. Передача идет 32-разрядными словами. Количество слов равно восьми.

```
XR0 = spm;; // address
XR1 = 0x00080001;;
XR2 = 0x00000000;;
XR3 = 0x42080000;;

DC4   = XR3:0;;
```

Включаем восьмой канал «стандартным» образом. Адрес приемника равен 0x40000. Количество слов равно восьми. Обмен идет 32-разрядными словами.

```
XR0 = 0x00040000;; // internal address
XR1 = 0x00800001;;
XR2 = 0x00000000;;
XR3 = 0x53080000;;

DC8   = XR3:0;;
```

Ожидаем некоторый промежуток времени, пока пересылка закончится. Это можно сделать, анализируя регистр состояния DMA или по прерыванию от восьмого канала. Проверяем, что данные успешно переданы.

```
  lc1 = 8;;

  j0 = spm;;
  j1 = 0x40000;;

rpp_dd:
  j2 = [j0+=1];;
  j3 = [j1+=1];;
  comp(j2,j3);;
  if njeq, jump error (NP);;
  if nlc1e, jump rpp_dd;;
```

12.3 Операции контроллера DMA

12.3.1 DMA управление каналами 4-11

Внутренние периферийные устройства процессора способны использовать 12 каналов DMA для обработки приема и передачи данных. Для передачи из внутренней или внешней памяти в устройства используются каналы 0, 2, 4-7. При использовании каналов 4-7 выбор периферийного устройства для требуемого канала выполняется с помощью соответствующих каналу бит регистра конфигурации DMACFGx. Регистры источника TCB программируются адресом внутренней/внешней памяти DI, инкрементом адреса DXM, количеством слов для передачи DXC и управляющими разрядами DP. По окончании программирования TCB автоматически начинается передача и продолжается

до завершения передачи блока. При этом транзакции всегда инициируются устройством в момент, когда оно готово принять новую порцию данных. Данный процесс повторяется, пока блок данных не передается полностью.

Для передачи из устройств во внутреннюю или внешнюю память, используются каналы 1, 3, 8-11. При использовании каналов 8-11 выбор периферийного устройства для требуемого канала выполняется с помощью соответствующих каналу бит регистра конфигурации DMACFGx. Регистры приемника TCB программируются адресом внутренней/внешней памяти DI, инкрементом адреса DXM, количеством слов для передачи DXC и управляющими разрядами DP. По окончании программирования TCB автоматически начинается передача. При этом транзакции канала инициируются периферийным устройством в момент, когда у него есть принятые данные. После генерации запроса, принятые данные считываются соответствующим каналом DMA и отсылаются по адресу получателя. Данный процесс повторяется, пока блок данных не передается полностью.

Количество слов для передачи или приема определяется как количество DXC или количество DXC умноженное на количество DYC (если режим 2D установлен).

12.3.2 DMA управление каналами 0-3

Особенностью каналов 0-3 является возможность выполнения передачи память-память. В связи с этим каналы 0-3 имеют два регистра TCB. Для передачи из внутренней во внешнюю память в регистры источника TCB записывается адрес внутренней памяти DI, приращение адреса DXM, количество слов для передачи DXC и управляющие разряды DP. Регистры TCB приемника заполняются адресом внешней памяти DI, приращением адреса DXM, количеством слов для передачи DXC и управляющими разрядами DP. При передаче в обратном направлении источник и приемник меняются местами.

После окончания настройки передача начинается автоматически и продолжается до полной передачи блока. Передачи выполняются порциями (по одному операнду на транзакцию). При этом канал может инициировать транзакции непрерывно.

Каналы 0-3 также могут выполнять пересылки устройство-память или память-устройство. В этом случае в TCB канала программируются адреса памяти и устройства. Основное отличие в такой передаче – это инициирование обмена по запросу устройства. Для этого в регистре конфигурации DMACFGx для заданного канала осуществляется программирование номера устройства. Выбранное устройство будет выполнять инициирование обменов. Для каналов 0-3 возможно задание внешнего источника запроса с помощью выводов nDMAR3–0, т.е. обмен может выполняться с устройством на внешней шине.

Устройство ввода-вывода, в отличие от памяти, сообщает каналу DMA о готовности к передаче данных. Устройство-источник готово, если у него есть данные для записи. Приемник готов, если у него есть место в буфере записи. Для передачи данных между внутренней и внешней памятью могут использоваться данные размером 32, 64 и 128 бит. При доступе к внешней памяти контроллер DMA использует те же интерфейсные сигналы, что и ядро процессора.

12.3.3 DMA управление каналом 12

Канал 12 использует режим работы, в котором контроллер DMA выступает в роли ведомого устройства, а цифровой смеситель (ЦС) – в роли ведущего. Для ЦС канал 12 виден как буфер приема информации с индикацией готовности. Когда ЦС имеет данные для передачи, он анализирует флаг готовности буфера приема канала 12 и записывает в буфер свои данные. Как только в буфере канала появляются данные, канал выполняет их пересылку во внутреннюю память в соответствии с запрограммированным в нем TCB. Для передачи данных между ЦС и внутренней памятью могут использоваться внутренние данные размером только 128 бит.

Канал 12 включен, если поле TY в регистре DP не равно нулю. Канал может поддерживать цепочки операций.

Если происходит запись в канал 12, когда он выключен и не занят загрузкой нового TCB цепочки, записанные данные теряются, происходит прерывание аппаратной ошибки и ошибка отражается в регистре SYSTAT.

Если запись в канал происходит, когда он выключен, но занят загрузкой TCB новой цепочки, данные помещаются в буфер и передаются по назначению, когда загрузка TCB для канала завершается. В этом случае нет генерации ошибки. В TCB канала программируется только базовый адрес буфера приемника. Смещение относительно базы передает сам ЦС вместе с соответствующей ему порцией данных. Канал 12 формирует итоговый адрес, суммируя значения базы и смещения.

Канал 12 работает с ЦС, только если установлен бит DRQ в регистре управления TCB. Если данный бит сброшен, канал 12 может выполнять обмен в случае, когда в его регистр AutoDMA производит запись ведущее устройство.

12.4 Передача DMA

12.4.1 Адресация памяти

Контроллер DMA поддерживает словную адресацию памяти. Это означает, что минимальной адресуемой единицей является 32-разрядное слово, и адрес в 32-разрядном индексном регистре есть адрес слова.

Благодаря 32-разрядному регистру адреса DMA может получать доступ к полному адресному пространству процессора. Каждый канал включает регистр индекса DI и поле модификации DXM в своем регистре TCB, которые используются для адресации буфера данных в памяти. 32-битный регистр TCB DI содержит начальный адрес блока и должен инициализироваться стартовым адресом буфера данных. Поле модификации DXM содержит значение со знаком (т.е. положительное или отрицательное) размером 16 или 22 бита и определяет величину изменения адреса после выполнения каждой транзакции. Измененное значение адреса используется в следующей транзакции. Само значение адреса в регистре DI не указывает на адресацию внутренней или внешней памяти. Чтобы указать, какому типу памяти соответствует адрес, используется дополнительное поле TY в регистре управления DP блока TCB. Если в результате модификации, адрес выйдет за границы существующей памяти или сменит

область внутренней памяти на внешнюю (или наоборот), контроллер DMA все равно будет обращаться к области памяти в соответствии с полем TY.

Передаваемые данные могут быть стандартными, сдвоенными или счетверенными словами. Это указывается в поле LEN регистра управления DP.

Каждый канал DMA имеет поле количества DXC в регистре DX, которое должно быть инициализировано количеством слов для передачи, независимо от длины данных (32, 64 или 128 бит). Счетчик количества уменьшается после каждой передачи DMA в данном канале. Значения уменьшения равны единице, двум или четырем и определяются полем LEN. Когда количество достигает значения ноль, DMA завершается и может быть сгенерировано прерывания для информирования процессора.

Если поле количества DXC в TCB DX инициализируется нулем, DMA передача на этом канале возможна и будет соответствовать выполнению передачи максимально возможного количества данных. Это происходит из-за того, что первая передача начинается до того, как проверяется значение количества.

Значение количества анализируется после вычитания из него единицы. Надежным способом отключения канала DMA является сброс поля TY в соответствующем управляющем регистре.

12.4.2 Приоритеты каналов DMA

Система приоритетов используется при выборе канала DMA для обслуживания, т.к. в одном такте может поступить активный запрос более чем от одного канала DMA. В зависимости от бита PR в регистре DP блока TCB канал может относиться к группе каналов с высоким приоритетом или к группе с низким приоритетом. Внутри каждой группы каналов используется фиксированная система приоритетов между каналами. Описание каналов DMA в порядке уменьшения приоритетов приведено в таблице 96.

Приоритет устанавливается исходя из следующих соображений:

– Входящие данные имеют более высокий приоритет, чем исходящие. Приемники устройств имеют более высокий приоритет, чем передатчики.

– Каналы устройств имеют более высокий приоритет, чем каналы общего назначения.

– Каналы общего назначения дополнительно имеют циклически изменяемый приоритет в последовательности 3-2-1-0.

Контроллер DMA анализирует запросы от всех каналов внутри двух групп и в течение одного такта определяет канал с самым высоким приоритетом. Выбранный канал внутри высокоприоритетной группы имеет преимущество над каналом обычной группы. Выбранному каналу в текущем такте предоставляется право выполнить транзакцию. В следующем такте арбитраж повторяется.

Загрузка цепочки TCB получает приоритет канала. Например, передача данных канала с более высоким приоритетом производится перед загрузкой цепочки TCB с более низким приоритетом.

Таблица 96 – Приоритеты каналов DMA

Канал	Описание	Приоритет
Канал 12	Интерфейс ЦС	Наиболее высокий
Канал 11	Устройства-приемники	
Канал 10		
Канал 9		
Канал 8		
Канал 7	Устройства-передатчики	
Канал 6		
Канал 5		
Канал 4		
Канал 3	Каналы общего назначения	
Канал 2		
Канал 1		
Канал 0		Наиболее низкий

12.4.3 Циклически присваиваемый приоритет

Для каналов 0-3 сделана дополнительная модификация – циклический приоритет внутри их группы. Это означает, что после каждого цикла арбитража приоритет каналов 0-3 меняется циклически. Приоритеты каналов после сброса следующие:

- канал 3 (самый высокий);
- канал 2;
- канал 1;
- канал 0 (самый низкий).

Каждый раз, если одному из каналов 0-3 предоставляется возможность выполнить транзакцию, приоритет каналов меняется. Только что работавший канал получает высочайший приоритет, остальные приоритеты расставляются как описано выше, но по циклу. Например, если канал 1 в текущем такте выполняет транзакцию, в следующем такте приоритет каналов следующий:

- канал 1 (самый высокий);
- канал 0;
- канал 3;
- канал 2 (самый низкий).

Если следующую транзакцию выполняет канал 0, приоритеты следующие:

- канал 0 (самый высокий);
- канал 3;
- канал 2;
- канал 1 (самый низкий).

Приоритеты каналов 0-3 остаются неизменными при выборе каналов с более высоким приоритетом (4-12). По окончании транзакции канала с более высоким приоритетом, порядок приоритетов в каналах 0-3 сохраняется прежним.

В примерах выше подразумевалось, что все четыре канала не устанавливают бит приоритета в ТСВ или все устанавливают, т.е. принадлежат одной группе. Если один или более канал установлен на высокий приоритет, циклический алгоритм применяется для каждой группы в отдельности. Например, если после сброса у канала 2 установлен бит приоритета, приоритеты следующие:

– канал 2 (установлен бит);
– канал 3 (самый высокий);
– канал 1;
– канал 0 (самый низкий).

Если канал 1 запрашивает доступ и получает его, схема приоритетов следующая:

– канал 2 (установлен бит);
– канал 1 (самый высокий);
– канал 0;
– канал 3 (самый низкий).

Если в течение работы канала 1, канал 2 генерирует запрос, он получает доступ и работа канала 1 прерывается. После окончания передачи каналом 2 схема приоритетов для каналов с более низкими приоритетами не меняется, и канал 1 выигрывает арбитраж и продолжает передачу. Если более одного канала имеют установленный бит приоритета, арбитраж между этими каналами проходит циклически.

12.4.4 Приоритет контроллер DMA на внутренних шинах процессора

Канал DMA, выигравший арбитраж приоритетов среди всех каналов внутри контроллера, получает возможность выполнить транзакцию и генерирует запрос к одному из трех ресурсов системы (см. рисунок 18):

– SOC IFIFO – входной буфер SOC-интерфейса для доступа к внутренней памяти;

– EBIU OFIFO – выходной буфер внешнего интерфейса для доступа к внешней памяти;

– SOC-шина периферийных устройств.

При арбитраже запросов к SOC IFIFO контроллер DMA имеет самый низкий приоритет. Первыми получают доступ данные, считываемые процессором с SOC-шины, или запрос от EBIU IFIFO внешнего интерфейса.

При арбитраже запросов к EBIU OFIFO контроллер DMA также имеет самый низкий приоритет. Первым получает доступ ядро процессора. При этом не имеет значения высокоприоритетный запрос DMA или нет.

При арбитраже запросов к SOC-шине периферийных устройств DMA имеет более высокий приоритет, чем ядро процессора, но ниже чем возвращаемые из ядра или внешней памяти данные. Данные, возвращаемые из ядра или внешней памяти, являются результатом пересылки DMA типа память-устройство. Канал DMA формирует запрос к SOC-шине только при выполнении пересылки устройство-память.

12.4.5 Цепочка DMA

Традиционный алгоритм работы DMA заключается в программировании процессором канала DMA, выполнение каналом передачи данных и генерации прерывания после завершения работы. В ответ на запрос прерывания процессор может запрограммировать канал на новую передачу.

Использование цепочек в DMA позволяет контроллеру автоматически инициализировать себя следующей передачей блока данных после завершения передачи текущего блока. При этом участие процессора не требуется, и он экономит время, которое потратил бы на обработку прерывания и программирование нового обмена. Используя механизм цепочки, могут быть организованы множественные операции DMA с различными атрибутами и устройствами ввода-вывода.

В первую очередь для поддержки цепочки используются биты регистра TCB DP:

– Бит CHEN разрешения цепочки;

– Поле CHPT указателя на цепочку. Это адрес в памяти, который содержит следующий TCB для загрузки (где содержится адрес, инкремент, количество слов для передачи и управляющие разряды для следующего передаваемого блока);

– Поле CHTG приемника нового TCB. Прочитанное из памяти значение TCB загружается в канал DMA указанный в данном поле.

Таким образом, в каждом TCB канала имеется механизм для загрузки в регистр TCB нового значения командного слова после окончания работы. Из-за ограничения размера указателя хранение значений цепочек возможно только во внутренней памяти процессора.

Существует несколько ограничений на цепочки DMA между каналами:

– Для каналов 0-3 возможна поддержка только одноканальной цепочки, т.е. поле CHTG не имеет значения, и новый TCB цепочки может быть загружен только в вызвавший ее канал. При этом в обоих TCB канала бит CHEN должен быть установлен, т.к. для работы этих каналов нужна загрузка двух TCB;

– Для каналов DMA 4-11 возможна межканальная цепочка, когда один канал может запускать работу в другом канале. Например, канал 8, приняв блок данных и загрузив его в память, может инициировать канал 4 на передачу принятых данных.

Включение и выключение цепочек

Передачи DMA запускаются записью счетверенного слова в регистр TCB (каналы 0-3 требуют загрузки обоих регистров TCB). Цепочка образуется при установленном бите разрешения CHEN, и, если указатель цепочки CHPT является разрешенным адресом.

Поле CHPT в TCB DP может быть загружено:

– при инициализации канала;

– во время работы канала (вставка цепочки).

В первом варианте пользователь заранее знает, что необходимо выполнить несколько разнообразных передач, и программирует цепочку операций, загружая первый TCB в канал, а все последующие TCB сохраняя во внутренней памяти.

Во втором случае пользователь запускает передачу одного блока, но в процессе выполнения программы (возможно, другим процессом) возникает необходимость в передаче нового блока данных. В этом случае программа определяет, что требуемый канал активен и устанавливает для данного канала бит паузы. Это вызывает приостановку работы канала и позволяет программе:

– Если цепочка операций в активном TCB не включена – организовать вставку цепочки путем установки бита CHEN в «1» и загрузки поля указателя CHPT необходимым значением. Значение TCB новой передачи сохраняется во внутренней памяти;

– Если цепочка операций включена – добавить в конец (или в любое место) списка цепочек свою запись.

После выполнения описанных действий процессор может сбросить бит паузы и канал продолжит работу. После сброса бита паузы приостановленный активный канал выполняет повторный контроль своего состояния и, в случае корректной работы, продолжает работу.

Генерация прерывания в цепочке операций

Пользователь может сам выбрать режим прерывания при работе цепочки. Если необходима генерация прерывания при передаче конкретного блока, бит INT в регистре TCB блока нужно установить в «1». Прерывание произойдет после передачи данного блока. Если необходимо сгенерировать прерывание только после окончания работы всей цепочки, бит INT должен быть установлен только в последнем TCB цепочки.

TCB и загрузка цепочек

Во время загрузки TCB цепочки, регистры TCB канала DMA загружаются значениями, полученными из внутренней памяти. TCB хранится в четырех последовательных словах выровненного счетверенного слова. Чтение нового TCB цепочки инициируется после передачи всего блока данных. Для каналов 0-3 необходимо чтение двух TCB. При необходимости чтения цепочки канал формирует запрос на чтение TCB со своим приоритетом, т.е. чтение цепочки каналом имеет такой же приоритет, как и чтение данных.

12.4.6 Двухмерный DMA

Данный раздел описывает изменения в работе, которые происходят при включении режима двухмерного DMA в процессоре. Режим двухмерного DMA включается установкой бита 2D в TCB DP регистре. Этот режим использует внешние и внутренние адреса. Преимущество двухмерной адресации состоит в перераспределении данных из одномерной строки (вектора) в представление в виде двухмерного массива (матрицы) с координатами X и Y. Измерения X и Y определяются в регистрах TCB передатчика и/или приемника. При этом измерение X определяет количество слов в строке массива, а измерение Y – количество строк.

– Измерения массива передатчика и приемника могут отличаться. Главное требование – итоговое число слов в источнике и получателе должно быть равно;

– Двухмерный массив памяти может быть перемещен в одномерный массив и наоборот, если итоговое число слов в источнике и получателе равно;

– Двухмерный блок в памяти может быть передан в порты связи, а блок, полученный из порта связи или из AutoDMA, может быть размещен в памяти как двухмерный массив.

12.4.7 Организация канала двухмерного DMA

В режиме двухмерного DMA адресация к массиву может быть выполнена на любом канале DMA, приемнике или передатчике. Регистр индекса (DI) загружается адресом первого элемента массива и обновляется после каждой передачи операнда на значение модификатора DXM, пока поле количества DXC не достигнет нуля. Величина модификатора DXM в регистре DX содержит смещение, добавляемое к текущему значению адреса для перехода на следующий элемент в измерении X (следующая колонка).

Для режима 2D поле количества DXC в регистре DX имеет дополнительную буферизацию (внутренний буфер CIX). При загрузке TCB, поле количества DXC и буфер CIX загружаются одинаковым значением. В процессе передачи по измерению X количество в DXC уменьшается до нуля и буфер CIX используется для перезагрузки количества в DXC. Можно сказать, что измерение X работает в режиме 2D так же, как в обычном режиме, только после передачи всей строки X происходит перезагрузка значения количества.

Регистр DY в TCB используется для задания количества строк массива и для перехода от одной строки к другой. В процессе передачи текущей строки к адресу элемента прибавляется значение DXM и осуществляется переход к следующему элементу строки (элементы не обязательно последовательны). При передаче последнего элемента строки (поле DXC достигает значения ноль) к адресу последнего элемента прибавляется модификатор DYM (модификатор DXM не добавляется). Это позволяет перейти к первому элементу следующей строки. Значение модификации DYM в регистре DY является целочисленным числом со знаком, что позволяет уменьшать или увеличивать адрес.

Значение количества строк в DYC уменьшается на единицу и, если это была не последняя строка, количество DXC перезагружается из CIX.

Когда поле количества DYC достигает нуля, передача блока DMA окончена, образуя матрицу размером X на Y.

Пример задания двухмерной передачи

Матрица с количеством колонок COL и количеством строк ROW. Каждый элемент матрицы имеет размер TXS (он определяется полем LEN в DP и может быть равен единице, двум или четырем). Элементы матрицы размещаются в памяти последовательно, причем сначала элементы первого столбца, затем второго и т.д. Например, необходимо выполнить передачу массива построчно. Следующая формула может быть применена для расчета параметров DX и DY.

DX = TXS*COL << 16; // количество слов данных в строке
DX |= TXS*ROW; // шаг от текущего элемента строки к следующему

DY = ROW << 16; // количество строк
DY |= (-TXS*(((COL-1)*ROW)-1))&0xFFFF; // переход к следующей строке

Операция «И» с 0xFFFF использовалась для того чтобы ограничить длину отрицательного модификатора размеров в 16 бит перед операцией логического ИЛИ с количеством строк.

Двухмерные DMA операции

Рассмотрим подробно действия, которые происходят внутри канала при двухмерной передаче. Все действия происходят в течение одного такта, но имеют причинно-следственную связь:

– Берется текущий адрес в регистре DI и запускается транзакция передачи операнда длиной LEN (чтение или запись);

– При подтверждении отправки транзакции происходит вычитание из DXC числа слов в операнде. Результат вычитания анализируется;

– Если результат вычитания не равен нулю, к содержимому регистра DI прибавляется значение модификации DXM. Результат вычитания помещается в поле количества DXC, а новое значение адреса в DI. На этом действия канала в текущей транзакции закончены;

– Если результат вычитания равен нулю, происходит вычитание единицы из поля количества DYC. Результат вычитания анализируется;

– Если результат вычитания количества DYC равен нулю – канал завершает работу;

– Если результат вычитания не равен нулю, в регистр количества DXC загружается начальное значение из буфера CIX, к значению регистра DI прибавляется значение модификатора DYM, результат вычитания загружается в поле количества DYC. Процесс повторяется.

Особенность двухмерного DMA (и любого DMA) в том, что первая передача начинается до анализа количества. Это означает, что DMA не может быть выключен установкой в ноль количества в регистре DXC или DYC. Для отключения двухмерного DMA необходимо очистить бит 2D в регистре управления DP. Для отключения канала DMA, необходимо очистить поле TY в регистре DP.

12.4.8 Прерывания DMA

Прерывание окончания DMA генерируется, если бит INT в регистре канала DP установлен.

DMA генерирует прерывание, когда количество передач в активном канале DMA доходит до нуля и передача заканчивается. Когда назначением DMA является внутренняя память, окончание передачи означает, что последний элемент данных попал в буфер SOC IFIFO.

Когда назначением DMA является внешняя память, окончание передачи означает, что последний элемент данных попал в буфер EBIU OFIFO.

Если назначением передачи является устройство-передатчик, окончание передачи означает, что последний элемент данных был загружен в буфер передатчика.

Видно, что генерация прерывания для случая, когда пункт назначения данных в памяти, не означает, что последние данные уже находятся в памяти. Это произойдет с задержкой. Для случая внутренней памяти задержка может достигать:

– 32 тактов процессора (если запись идет в блок памяти, куда одновременно обращаются с чтением все три внутренних процессорных шины);

– 16 тактов (если запись идет в блок памяти без конфликтов).

Минимальная задержка три такта. Процессор не в состоянии за это время обработать запрос прерывания и попытаться прочитать последнее слово переданных данных.

Задержка для внешней памяти зависит от частоты и настроек системной шины.

Каждый канал DMA имеет собственный бит запроса прерывания в контроллере прерываний. Прерывания DMA фиксируются в регистре ILAT и разрешаются в регистреIMASK. Приоритет запросов прерываний от каналов DMA фиксированный и определен в контроллере прерываний.

Опрос регистра DSTAT может использоваться как альтернатива прерываниям для определения окончания одиночной последовательности DMA. Если включены цепочки, опрос DSTAT нельзя использовать, т.к. следующая последовательность DMA может быть на подходе в момент возврата статуса по результатам опроса.

12.4.9 Запуск и окончание последовательностей DMA

Данный подраздел рассматривает запуск, приостановку, возобновление и окончание последовательностей DMA.

12.4.9.1 *Запуск последовательности DMA*

Последовательность DMA для каналов 0-3 запускается в одном из случаев:

– включен канал DMA, бит запроса DRQ равен нулю;

– включен канал x DMA, установлен бит запроса DRQ и получен запрос от выбранного устройства.

Последовательность DMA для каналов 4-7 запускается в случае, когда:

– включен канал DMA и имеется запрос от устройства-передатчика на прием данных в буфер передатчика. Бит запроса DRQ не имеет значения.

Последовательность DMA для каналов 8-11 запускается в случае, когда:

– включен канал DMA и имеется запрос от устройства-приемника о наличии данных в буфере приемника. Бит запроса DRQ не имеет значения.

Последовательность DMA для канала 12 запускается в случае, когда:

– Включен канал DMA и имеются данные в буфере канала. Буфер канала может принять до четырех операндов. Бит запроса DRQ не имеет значения.

Для запуска новой последовательности DMA после завершения текущей, программа должна записать новые параметры в регистры TCB. Запись активного TCB в активный TCB регистр вызывает ошибку.

12.4.9.2 *Приостановка последовательности DMA*

Последовательность операций DMA в некотором канале приостанавливается, когда соответствующий ему бит паузы в регистре DCNT установлен. Установка бита паузы может произойти в любой момент работы канала: канал может работать, загружать цепочку или завершить работу. Поэтому после установки бита паузы пользователь должен перед продолжением работы проанализировать состояние канала.

Если момент установки паузы пришелся на загрузку цепочки, бит паузы не остановит загрузку цепочки, и до момента загрузки цепочки поле TY регистра TCB будет читаться как 0. Хотя биты состояния в этот момент будут показывать, что канал активен.

Если момент установки бита паузы пришелся на завершение работы каналом, поле TY регистра TCB будет равно нулю, и регистр состояния укажет, что канал не активен.

Установка бита паузы в момент нормальной работы канала (пересылка данных) вызывает останов операций передачи. При этом каналы с четвертого по 12 немедленно прекращают операции пересылки. Каналы с нулевого по третий немедленно прекращают операции чтения данных из источника, но продолжают выполнять незавершенные операции записи.

Внимание! При наличии незавершенных операций записи после установки бита паузы SOC-шина становится недоступной для ядра процессора.

Разблокировка доступа произойдет только после завершения операций записи.

Установка бита паузы позволяет записать новое активное значение TCB (поле TY не равно нулю) в активный регистр TCB. Такая запись приводит к генерации аппаратной ошибки. Однако, в случае паузы такая запись разрешена. Если случай записи без бита паузы рассматривается как сбой в программе, запись с установленным битом паузы является осознанной последовательностью действий по смене режима работы канала. Это позволяет отложить задачу, выполняемую каналом, и запрограммировать канала на более приоритетную задачу. После ее завершения состояние канала может быть восстановлено.

Имеется особенность в перепрограммировании каналов с нулевого по третий. Если новое активное TCB имеет поле TY такое же, как активный регистр TCB канала, данная запись рассматривается только как попытка модификации регистра DP (например, попытка вставить цепочку). В этом случае при записи обновляется только значение регистра DP. Все другие значения неизменны. Поэтому для каналов с нулевого по третий, для задания нового обмена необходимо:

- сохранить текущие параметры регистра TCB;
- записать в регистр TCB значение с полем TY равным нулю;
- записать активное значение в TCB регистр.

12.4.9.3 *Возобновление последовательности DMA*

Последовательность операций DMA в некотором канале возобновляется, когда соответствующий ему бит паузы в регистре DCNT сбрасывается в ноль. Для каналов 4-12 это означает немедленное возобновление работы в соответствии с командой TCB. Для каналов 0-3 сброс бита паузы может привести к процедуре проверки правильности конфигурации TCB, если во время паузы были записи в TCB регистры.

12.4.9.4 *Окончание последовательности DMA*

Последовательность DMA заканчивается в одном из случаев:

– Регистры количества равны нулю и цепочка заканчивается.

– Канал отключен сбросом поля TY в одном из TCB и DP регистрах. При этом такая запись неактивного TCB может происходить в любой момент. Для каналов 0-3 могут быть запущены до восьми транзакций чтения во внутреннюю или внешнюю память, и после отключения канала все эти данные поступят во внутренний буфер контроллера DMA и будут утеряны, если канал выключен. Если к этому времени канал будет включен для новой задачи, оставшиеся от предыдущей задачи данные вызовут ошибку в работе канала, т.к. обнаружится запись данных, которые не читались.

12.5 Каналы DMA общего назначения

Четыре канала DMA с нулевого по третий являются каналами общего назначения, т.к. обладают двумя регистрами TCB и позволяют программировать произвольный источник и приемник данных. Регистры TCB каналов определяют следующее:

– канал включен, если оба поля TY в TCB DP не равны нулю;

– приоритет канала высокий, если бит PR установлен в одним из регистров TCB DP;

– прерывание DMA включено, если бит INT установлен в одном из регистров TCB DP;

– режим запроса DMA включен, если бит DRQ установлен в одном из регистров TCB DP;

– поле LEN должно быть одинаковым в обоих регистрах;

– бит CHEN должен быть одинаковым в обоих регистрах;

– передача внутренней и внешней памяти устанавливается в поле TY регистров TCB DP, что позволяет эффективно передавать данные между внутренней памятью процессора и внешней памятью или устройством;

– поле CHTG не имеет смысла для этих каналов. В случае цепочки каждый регистр TCB загружается новое значение в свой канал в соответствии с его полем CHPT.

Передача DMA между внутренней памятью процессора и внешней памятью требует от одного канала DMA генерации адресов для обоих типов памяти. Внешний адрес генерируется в соответствии с данными в регистре TCB внешней памяти. Адрес внутренней памяти генерируется в соответствии с данными в регистре TCB внутренней памяти. Если бит 2D в TCB DP установлен в одном из регистров TCB, адрес этого TCB также обновляется в соответствии со значением модификатора в регистре DY. Передачи внутри внешней памяти и внутренней памяти поддерживаются. Также поддерживается передача между внешней памятью и внешними устройствами.

Поле TY в регистре DP указывает тип передачи. Ранее приводилось описание допустимых комбинаций типов в одном канале (таблица 84). Каждый канал с нулевого по третий имеет регистр TCB передатчика и приемника, где передатчик считывает данные из памяти и передает их приемнику, а приемник отсылает данные в память.

Большинство параметров TCB не имеют отношения к типу памяти, а являются описателями блока памяти (который может размещаться в любом месте) или описывают поведение канала во время работы и при завершении обмена.

Описатели блока памяти:

– регистры DX и DY задают размеры блока и расположение элементов блока в памяти;

– бит 2D определяет, рассматривается блок данных как линейный или как двухмерный;

– биты LEN определяют длину операнда блока.

Описатели поведения канала во время работы и при завершении обмена:

– бит PR указывает, что канал имеет высокий приоритет;

– бит DRQ определяет активацию транзакции по запросу или без него;

– бит INT разрешает формирование запроса прерывания к процессору по окончании обмена;

– бит CHEN разрешает цепочку операций;

– поле CHPT хранит адрес цепочки;

– поле CHTG определяет приемник цепочки.

Рассмотренные выше параметры не зависят от типа обмена. Поэтому далее при описании различных вариантов обмена будут рассматриваться параметры, которые зависят от выбранного типа памяти.

12.5.1 Передача из внешней памяти во внутреннюю память

Для передачи данных из внешней памяти во внутреннюю необходимо запрограммировать TCB передатчика для чтения внешней памяти, а TCB приемника – для записи во внутреннюю память. TCB передатчика программируется следующим образом:

– Регистр DI

Инициализируется начальным адресом блока данных во внешней памяти.

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память. Выбор между типом 4 или 7 влияет только на длину полей DXC, DXM, DYC, DYM.

TCB приемника данных описывается аналогичным образом со следующими особенностями, связанными с выбором внутренней памяти:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память. Выбор между типом 2 и 3 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней памяти.

Все другие параметры TCB для передатчика и приемника описывают блоки памяти и поведение канала. В описанных выше параметрах имеются ограничения, основные из которых:

– поле LEN должно быть одинаковым в обоих TCB;

– бит CHEN должен быть одинаковым в обоих TCB.

Выбор между типами обменов 2 и 3 или 4 и 7 зависит от размера величины модификации адреса. Если при переходе от одного элемента к другому необходимо изменить адрес на величину более чем 32768 (в положительную сторону или отрицательную), недостаточно 16 бит поля DXM для хранения такого значения. Поэтому следует использовать тип обмена 3 или 7 место 2 или 4, чтобы поле модификации было 22 бита. В этом случае уменьшается размер поля количества.

12.5.2 Передача из внутренней памяти во внешнюю память

В случае передачи из внутренней памяти во внешнюю память нет никаких особенностей. По сравнению с режимом, описанным в пункте 12.5.1 «Передача из внешней памяти во внутреннюю память», TCB передатчика и приемника просто меняются местами.

TCB передатчика:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память. Выбор между типом 2 и 3 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней памяти.

TCB приемника:

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память. Выбор между типом 4 и 7 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

– Регистр DI

Инициализируется начальным адресом блока данных во внешней памяти.

12.5.3 Передача из внешней памяти во внешнюю память

Подобный тип передачи выполняется в случае, когда в TCB приемника и передатчика поля типа TY равны четырем или семи. Адреса в регистрах DI должны указывать на блоки данных во внешней памяти.

12.5.4 Передача из внутренней памяти во внутреннюю память

Данный тип передачи программируется следующим образом.

TCB передатчика:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память. Выбор между типом 2 и 3 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней памяти.

TCB приемника:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память. Выбор между типом 2 и 3 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

– Регистр DI

Инициализируется начальным адресом блока данных внутренней памяти.

Необходимо отметить, что пересылки во внутренней памяти с помощью DMA выполняются как минимум вдвое медленнее, чем такие же пересылки, выполненные ядром процессора. Однако полезным может быть тот факт, что пересылка выполняется в фоновом режиме и процессор может выполнять в это время другую работу.

12.5.5 Передача из устройства ввода во внутреннюю или внешнюю память

Данный тип передачи программируется следующим образом.

TCB передатчика:

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память. Выбор между типом 4 и 7 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

Бит DRQ должен быть установлен, что указывает на инициирование обмена только по запросу от устройства.

– Регистр DI

Инициализируется адресом буфера периферийного устройства. При этом если бит 31 адреса равен единице, это адрес внутреннего периферийного устройства на SOC-шине. Если бит равен нулю, это внешнее устройство.

TCB приемника:

– Регистр DP

Поле TY должно быть равно двум (трем) или четырем (семи), что указывает на внутреннюю или внешнюю память. Выбор влияет только на длину полей DXC, DXM, DYC, DYM приемника.

Бит DRQ должен быть установлен, что указывает на инициирование обмена только по запросу от устройства.

– Регистр DI

Инициализируется начальным адресом блока данных внутренней или внешней памяти.

Регистр конфигурации DMACFGx задает номер периферийного устройства, которое будет формировать сигналы запроса на обмен.

12.5.6 Передача из внутренней или внешней памяти в устройство вывода

Данный тип передачи программируется следующим образом.

TCB передатчика:

– Регистр DP

Поле TY должно быть равно двум (трем) или четырем (семи), что указывает на внутреннюю или внешнюю память. Выбор типа влияет только на длину полей DXC, DXM, DYC, DYM приемника.

Бит DRQ должен быть установлен, что указывает на инициирование обмена только по запросу от устройства.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней или внешней памяти.

TCB приемника:

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память. Выбор между типом 4 и 7 влияет только на длину полей DXC, DXM, DYC, DYM приемника.

Бит DRQ должен быть установлен, что указывает на инициирование обмена только по запросу от устройства.

– Регистр DI

Инициализируется адресом буфера периферийного устройства. При этом если бит 31 адреса равен единице, это адрес внутреннего периферийного устройства на SOC-шине. Если бит равен нулю, это внешнее устройство.

Регистр конфигурации DMACFGx задает номер периферийного устройства, которое будет формировать сигналы запроса на обмен.

Канал DMA не контролирует корректность соответствия номера устройства в регистре DMACFGx адресу буфера устройства в регистре TCB. Периферийное устройство выступает только в роли инициатора транзакции. Однако пересылка данных определяется только TCB источника и приемника. Поэтому по инициативе периферийного устройства с помощью каналов 0-3 можно выполнять любой вариант пересылки, который можно запрограммировать в канале. Отличие каналов 0 и 2 от каналов 1 и 3 только в том, что они имеют различные источники запросов обмена. При этом нужно помнить, что, если какое-то устройство сформировало запрос на обмен, для формирования следующего запроса устройство должно быть кем-то обслужено, т.е. из источника должны быть прочитаны данные, а в приемник записаны.

12.6 Каналы DMA с четвертого по 11 для работы с устройствами

Четыре канала DMA определяют передачи от устройств к внутренней памяти или внешней памяти. Другие четыре канала определяют передачи из внутренней памяти или внешней памяти в устройства.

Особенностью каналов DMA является то, что все они имеют только один регистр TCB. Для каналов с четвертого по седьмой это TCB передатчика, который читает данные из внешней или внутренней памяти и отправляет данные в устройство-передатчик. Для каналов с восьмого по 11 это TCB приемника, который берет данные из устройства-приемника и пересылает их во внутреннюю или внешнюю память. Наличие только одного TCB регистра в каналах 4-11 не позволяет определить вторую сторону пересылки (источник или приемник). Для разрешения данной проблемы и используется дополнительный регистр DMACFGx.

Как в случае каналов общего назначения, большинство параметров TCB не имеют отношения к устройству и описывают только блок данных или поведение канала во время работы или при завершении передачи. Поле TY регистра TCB указывает на тип передачи.

12.6.1 Передача из устройства во внутреннюю \ внешнюю память

Для переноса данных от устройства во внутреннюю или внешнюю память должны использоваться только каналы с восьмого по 11. Приемник TCB этих каналов должен быть запрограммирован соответствующим образом.

Для передачи во внутреннюю память:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней памяти куда будут пересылаться данные из порта связи.

Для передачи во внешнюю память:

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память.

– Регистр DI

Инициализируется начальным адресом блока данных во внешней памяти, куда будут пересылаться данные из порта связи.

Для выбранного канала в регистре DMACFGH должен быть задан номер устройства, с которым работает канал. Биты регистра DMACFGH имеют следующее соответствие: биты 0-3 – каналу 8, биты 4-7 – каналу 9, биты 8-11 – каналу 10, биты 12-15 – каналу 11. Передача всегда инициируется устройством. Как только буфер устройства-приемника загружается принятыми данными, посылается запрос к каналу DMA, и он выполняет пересылку данных в блок данных описываемый регистром TCB.

Каналы DMA 4-11 имеют одну особенность при организации цепочки операций: если в универсальном канале новое значение TCB может быть загружено только в регистр

того же канала, для каналов 4-11 можно указать номер канала-приемника, следующего TCB. Правда номер канала-приемника лежит в диапазоне от четырех до 11. Поэтому после окончания работы один канал может инициировать работу другого канала.

12.6.2 Внутренняя/внешняя память – устройство-передатчик

Для пересылки данных из внутренней или внешней памяти в устройство должны использоваться только каналы с четвертого по седьмой. Передатчик TCB этих каналов должен быть запрограммирован соответствующим образом.

Для передачи из внутренней памяти:

– Регистр DP

Поле TY должно быть равно двум или трем, что указывает на внутреннюю память.

– Регистр DI

Инициализируется начальным адресом блока данных во внутренней памяти, откуда будут читаться данные для порта связи.

Для передачи из внешней памяти:

– Регистр DP

Поле TY должно быть равно четырем или семи, что указывает на внешнюю память.

– Регистр DI

Инициализируется начальным адресом блока данных во внешней памяти, откуда будут читаться данные для порта связи

Для выбранного канала в регистре DMACFGL должен быть задан номер устройства, с которым работает канал. Биты регистра DMACFGL имеют следующее соответствие: биты 16-19 – каналу 4, биты 20-23 – каналу 5, биты 24-27 – каналу 6, биты 28-31 – каналу 7.

Передача всегда инициируется устройством-передатчиком. Как только буфер передатчика имеет возможность принять данные - посылается запрос к каналу DMA, и он выполняет пересылку данных из блока данных, описываемого регистром TCB, в буфер передатчика.

12.6.3 Пересылка между портами связи

Контроллер DMA поддерживает возможность пересылки данных между портами связи без использования промежуточных буферов в памяти. Понятно, что передача может идти от приемника порта связи в передатчик порта связи.

В процессоре имеется два порта связи. Поскольку инициатором передачи в данном случае выступает приемник порта связи, такие передачи поддерживаются только каналами DMA с 8 по 9. Отличием такой пересылки от пересылки в память является необходимость дополнительного контроля готовности передатчика канала связи принять данные от приемника. Особенности программирования TCB приемника каналов с 8 по 9 состоят в следующем:

– Регистр DP

Поле TY должно быть равно единице, что указывает на порт связи.

– Регистр DI

Инициализируется адресом передатчика порта связи, который будет принимать данные. Адрес 0x1F04A0 для порта связи 0, 0x1F04A8 для порта связи 1. На самом деле в данном адресе важны только биты 3 и 4. С их помощью кодируется номер передатчика порта связи. Все другие биты безразличны.

– Модификатор DXM должен быть сброшен.

Передача всегда инициируется готовностью приемника канала связи и готовностью передатчика канала связи. Как только буфер приемника загружается принятыми данными, посылается запрос к каналу DMA. Канал DMA анализирует состояние буфера передатчика, в который он запрограммирован записать данные. Если буфер передатчика канала связи пуст – происходит пересылка.

12.7 Канал 12

Канал 12 поддерживает возможность пересылки данных от ведущего устройства во внутреннюю память процессора. В качестве ведущего устройства может выступать только модуль цифрового смесителя (ЦС). Блок данных в TCB приемника описывается стандартным образом. Параметры работы канала и завершения работы канала определяются стандартным образом. Единственным ограничением в TCB является то, что тип обмена TY должен быть равен двум или трем, т.е. приемник данных всегда внутренняя память.

Канал 12 имеет буфер данных и запрашивает выполнение транзакции, как только в буфере появляются данные. Размер буфера данных составляет четыре 128-разрядных слова. Требованием в работе канала является соответствие размера данных, записываемых в буфер, размеру операнда, который определяется полем LEN в регистре TCB.

Канал принимает данные в буфер только, когда он активен. Если происходит запись в буфер при неактивном канале – генерируется флаг ошибки.

12.8 Пример

Рассмотрим пример использования каналов 4-11 для организации пересылок с учетом новых возможностей. Допустим, необходимо переслать массив из 8-ми чисел из одной области внутренней оперативной памяти в другую область внутренней оперативной памяти, используя при этом в качестве инициатора пересылки таймер 0 процессора 1986. Решим задачу с использованием каналов 4 и 8, а также с использованием интерфейса SPI1.

Введем обозначения:

```
#define base_DST_ch74   0x80000030
#define base_SRC_ch118  0x80000050
#define base_DMACFG     0x80000078
Наш массив для пересылки
.var spm[8] = {0x11111111, 0x11111122,0x11111333,0x11114444,
                    0x11155555, 0x11666666,0x17777777,0x88888888};
```

Предполагаем, что таймер 0 у нас уже включен, работает и с определенным периодом формирует запросы к каналам DMA. Поскольку каналы DMA не включены и не настроены на прием запросов от таймера, то никаких событий не происходит.

Включим SPI1 интерфейс для передачи 32-разрядных данных в тестовом режиме с включенной обратной связью.

```
        j8 = base_SPI1;;

xr0 = 0x1001f;;
[j8+0] = xr0;; // cr1
xr0 = 0x0d;;   //
[j8+1] = xr0;; // cr2
xr0 = 0xff;;
[j8+3] = xr0;; // st
```

Определим источники запросов для активации каналов 4 и 8. Канал 4 будем использовать для пересылки данных из памяти в передатчик SPI1 по запросу таймера 0 (запрос 0). Канал 8 будем использовать для пересылки данных от приемника SPI1 во внутреннюю память. Итого код активации канала 4 равен 20 (таблица 83), а код активации канала 8 равен 12 (таблица 84). Формируем 64-разрядный код и пишем его в регистр конфигурации

```
          j8 = base_DMACFG;;
xr0 = 0x00040000;;   // GTMR0 req
              xr1 = 0x0010000c;;
l[j8+0] = xr1:0;;
```

Определяем устройство, в которое канал 4 будет записывать данные для передачи. Адрес устройства есть буфер данных SPI1. Активируем регистр DCA, записав в поле активации значение 001b.

```
              j8 = base_DST_ch74;;
xr0 = 0x80000362;;   // dest  SPI1
              xr1 = 0x20000000;;// код активации
l[j8+0] = xr1:0;;
```

Определяем устройство, из которого канал 8 будет читать данные. Отметим, что в данном случае можно было бы обойтись без новых функций, т.к. источник запроса и обслуживающее устройство это одно и тоже устройство. Но будем использовать регистр DCA. Запишем адрес-источник данных – это буфер данных приемника SPI1. Код активации равен 100b что означает обращение к устройству на SOC-шине.

```
j8 = base_SRC_ch118;;
xr0 = 0x80000362;; // src  SPI1
        xr1 = 0x80000000;;// код активации
l[j8+0] = xr1:0;;
```

Включаем каналы «стандартным» образом. Включаем канал 4 для передачи. Адрес источника – это массив SPM. Передача идет 32-разрядными словами. Количество слов равно 8.

```
XR0 = spm;; // address
XR1 = 0x00080001;;
XR2 = 0x00000000;;
XR3 = 0x42080000;;

DC4    = XR3:0;;
```

Включаем канал 8 «стандартным» образом. Адрес приемника равен 0x40000. Количество слов равно 8. Обмен идет 32-разрядными словами.

```
XR0 = 0x00040000;; // internal address
XR1 = 0x00800001;;
XR2 = 0x00000000;;
XR3 = 0x53080000;;

DC8    = XR3:0;;
```

Ожидаем некоторый промежуток времени, пока пересылка закончится. Это можно сделать, анализируя регистр состояния DMA либо по прерыванию от канала 8.Проверяем, что данные успешно переданы.

```
lc1 = 8;;

j0 = spm;;
j1 = 0x40000;;

rpp_dd:
j2 = [j0+=1];;
j3 = [j1+=1];;
comp(j2,j3);;
if njeq, jump error (NP);;
if nlc1e, jump rpp_dd;;
```

13 Интерфейс внешней памяти

Большинство выводов процессора являются универсальными, т.е. они индивидуально могут быть запрограммированы как вход или выход. Вместе с тем процессор может быть сконфигурирован так, что набор внешних контактов будет образовывать внешний порт, который обеспечивает доступ к внешней памяти, устройствам ввода-вывода, отображаемыми на пространство памяти.

Внешний порт процессора обеспечивает связь с внешней памятью и периферией. Внешняя память и периферия включены в унифицированное адресное пространство процессора. Внутренние шины процессора мультиплексируются на внешний порт, создавая шину внешней системы с одной 22-битной шиной адресов и одной 8*\16\32-битной шиной данных. Внешняя память и устройства могут иметь разрядность 8*, 16 или 32 бита. Процессор автоматически упаковывает внешние данные в слова длиной 32, 64 или 128 бит. Встроенное декодирование адресных линий (для генерации сигналов выбора блоков памяти) обеспечивает адресацию устройств внешней памяти. Специальные линии контроля также формируются для упрощения адресации страничного режима динамической памяти.

Примечание – 8-битная шина данных, а также внешняя память и устройства с разрядностью 8 бит доступны для микросхем с ревизии 3.

Процессор использует синхронный конвейер при обменах на внешней шине. Это позволяет осуществлять связь с синхронной DRAM. Опция допускает асинхронную связь для медленных устройств.

Внешние данные могут быть доступны по каналам DMA или через ядро. При доступе посредством ядра задержка чтения может быть значительной, что может вызывать длительные остановы процессора.

Программируемое количество тактов ожидания позволяет работать с периферией, имеющей различные требованиями на время доступа.

Внешние выводы внешней памяти приведены в Таблице 97

Таблица 97 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода памяти	Тип вывода	Функциональное назначение
PC[7]	SCLK	O	Внешний интерфейс. Синхросигнал
PC[8]	SD_CKE	I/O	Интерфейс SDRAM. Разрешение синхронизации
PC[9]	MSSD[0]	I/O	Интерфейс SDRAM. Активация модуля 0
PC[10]	MSSD[1]	I/O	Интерфейс SDRAM. Активация модуля 1
PC[11]	MSSD[2]	I/O	Интерфейс SDRAM. Активация модуля 2
PC[12]	MSSD[3]	I/O	Интерфейс SDRAM. Активация модуля 3
PC[13]	SD_nCAS	I/O	Интерфейс SDRAM. Выбор колонки
PC[14]	SD_nRAS	I/O	Интерфейс SDRAM. Выбор строки
PC[15]	SD_nWE	I/O	Интерфейс SDRAM. Признак записи
PC[16]	SD_DQM	I/O	Интерфейс SDRAM. Маска

Обозначение вывода (основное)	Обозначение вывода памяти	Тип вывода	Функциональное назначение
PC[17]	SD_A10	I/O	Интерфейс SDRAM. Бит 10 адреса
PC[18]	nMS[1]	O	Интерфейс статической памяти. Выбор типа 1
PC[19]	nMS[0]	O	Интерфейс статической памяти. Выбор типа 0
PC[20]	nBMS	O	Интерфейс статической памяти. Выбор внешней памяти начальной загрузки
PC[21]	nRD	I/O	Интерфейс статической памяти. Строб чтения
PC[22]	nWR	I/O	Интерфейс статической памяти. Строб записи
PC[23]	ACK	I/O	Интерфейс статической памяти. Сигнал готовности
PE[0]	A[0]	O	Внешний интерфейс. Шина адреса
PE[1]	A[1]	O	Внешний интерфейс. Шина адреса
PE[2]	A[2]	O	Внешний интерфейс. Шина адреса
PE[3]	A[3]	O	Внешний интерфейс. Шина адреса
PE[4]	A[4]	O	Внешний интерфейс. Шина адреса
PE[5]	A[5]	O	Внешний интерфейс. Шина адреса
PE[6]	A[6]	O	Внешний интерфейс. Шина адреса
PE[7]	A[7]	O	Внешний интерфейс. Шина адреса
PE[8]	A[8]	O	Внешний интерфейс. Шина адреса
PE[9]	A[9]	O	Внешний интерфейс. Шина адреса
PE[10]	A[10]	O	Внешний интерфейс. Шина адреса
PE[11]	A[11]	O	Внешний интерфейс. Шина адреса
PE[12]	A[12]	O	Внешний интерфейс. Шина адреса
PE[13]	A[13]	O	Внешний интерфейс. Шина адреса
PE[14]	A[14]	O	Внешний интерфейс. Шина адреса
PE[15]	A[15]	O	Внешний интерфейс. Шина адреса
PE[16]	A[16]	O	Внешний интерфейс. Шина адреса
PE[17]	A[17]	O	Внешний интерфейс. Шина адреса
PE[18]	A[18]	O	Внешний интерфейс. Шина адреса
PE[19]	A[19]	O	Внешний интерфейс. Шина адреса
PE[20]	A[20]	O	Внешний интерфейс. Шина адреса
PE[21]	A[21]	O	Внешний интерфейс. Шина адреса
OSCO	A[22]	O	Внешний интерфейс. Шина адреса
PD[0]	D[0]	I/O	Внешний интерфейс. Шина данных
PD[1]	D[1]	I/O	Внешний интерфейс. Шина данных
PD[2]	D[2]	I/O	Внешний интерфейс. Шина данных
PD[3]	D[3]	I/O	Внешний интерфейс. Шина данных
PD[4]	D[4]	I/O	Внешний интерфейс. Шина данных
PD[5]	D[5]	I/O	Внешний интерфейс. Шина данных
PD[6]	D[6]	I/O	Внешний интерфейс. Шина данных
PD[7]	D[7]	I/O	Внешний интерфейс. Шина данных
PD[8]	D[8]	I/O	Внешний интерфейс. Шина данных
PD[9]	D[9]	I/O	Внешний интерфейс. Шина данных

Обозначение вывода (основное)	Обозначение вывода памяти	Тип вывода	Функциональное назначение
PD[10]	D[10]	I/O	Внешний интерфейс. Шина данных
PD[11]	D[11]	I/O	Внешний интерфейс. Шина данных
PD[12]	D[12]	I/O	Внешний интерфейс. Шина данных
PD[13]	D[13]	I/O	Внешний интерфейс. Шина данных
PD[14]	D[14]	I/O	Внешний интерфейс. Шина данных
PD[15]	D[15]	I/O	Внешний интерфейс. Шина данных
PD[16]	D[16]	I/O	Внешний интерфейс. Шина данных
PD[17]	D[17]	I/O	Внешний интерфейс. Шина данных
PD[18]	D[18]	I/O	Внешний интерфейс. Шина данных
PD[19]	D[19]	I/O	Внешний интерфейс. Шина данных
PD[20]	D[20]	I/O	Внешний интерфейс. Шина данных
PD[21]	D[21]	I/O	Внешний интерфейс. Шина данных
PD[22]	D[22]	I/O	Внешний интерфейс. Шина данных
PD[23]	D[23]	I/O	Внешний интерфейс. Шина данных
PD[24]	D[24]	I/O	Внешний интерфейс. Шина данных
PD[25]	D[25]	I/O	Внешний интерфейс. Шина данных
PD[26]	D[26]	I/O	Внешний интерфейс. Шина данных
PD[27]	D[27]	I/O	Внешний интерфейс. Шина данных
PD[28]	D[28]	I/O	Внешний интерфейс. Шина данных
PD[29]	D[29]	I/O	Внешний интерфейс. Шина данных
PD[30]	D[30]	I/O	Внешний интерфейс. Шина данных
PD[31]	D[31]	I/O	Внешний интерфейс. Шина данных

13.1 Внешняя память

Интерфейс внешней памяти позволяет организовать подключение различных типов памяти к процессору. Основными типами памяти являются асинхронная или синхронная статическая память (SRAM), а также синхронная динамическая память (SDRAM).

Контроллер SDRAM расположен на кристалле процессора. Он управляет всеми контрольными сигналами SDRAM (RAS, CAS, SDWE, SDCKE, SDQM) и поддерживает инициализацию и регенерацию микросхем SDRAM. Максимальная производительность SDRAM при передаче данных равна одной передаче за один тактовый цикл. Производительность может удерживаться близко к максимальной, если последовательные доступы выполняются к той же странице

Процессор поддерживает также протокол медленного устройства. Этот протокол следует использовать для устройств не критичных в плане производительности.

Внешняя шина включает в себя шину данных, с пропускной способностью 32, 16 или 8 разрядов, 23-разрядную шину адреса и управляющие сигналы. Процессор всегда ведущий на внешней шине.

В разделе 9 «Таймеры с функцией Захвата/ШИМ» отмечалось, что интерфейс внешней памяти реализуется на базе портов общего назначения PE и PD. Определяющим

конфигурационным регистром является регистр PX_ALT (таблица 69). Для реализации полнофункционального внешнего интерфейса (32-разрядная шина данных и 23-разрядная шина адреса) в регистр PX_ALT необходимо записать значение 0_0111_1111b. Это соответствует тому, шина данных и шина адреса полностью контролируются модулем интерфейса внешней памяти. Линии управления внешнего интерфейса используют порт PC и также нуждаются в их конфигурировании. Регистр PC_ALT позволяет задать необходимую конфигурацию в зависимости от типа используемой памяти.

После того, как модуль внешнего интерфейса получает контроль над внешними выводами микросхемы, он может быть дополнительно сконфигурирован под различные настройки путем записи в конфигурационные регистры SYSCON и SDRCON. Данные регистры задают параметры непосредственно для контроллера внешнего интерфейса. Регистры контроллера внешнего интерфейса доступны как регистры периферийных устройств на SOC-шине. Базовый адрес регистров 0x8000_0080. Список доступных регистров приведен в таблице 98. Номер регистра образуется конкатенацией номера группы (старшие шесть бит) и номера регистра в группе (младшие пять бит).

Таблица 98 – Группа регистров интерфейса шины

Имя	Описание	Номер\смещение	Значение после сброса
SYSCON	Регистр конфигурации внешней шины	0x0480 \ 0	0x0008_0067
BUSLOCK	Регистр блокировки внешней шины	0x0483 \ 3	0
SDRCON	Регистр конфигурации SDRAM	0x0484 \ 4	0
SDRSRF	Запрос режима selfrefresh	0x0485 \ 5	0
SYSTAT	Регистр статуса системы; только чтение	0x0486 \ 6	0
SYSTATCL	Регистр статуса системы; адрес для сброса ошибок. Только чтение	0x0487 \ 7	0
SD_REF	16-ти разрядный регистр периода регенерации SDRAM	Адрес 0x8000009C	0x0000_00FF

13.1.1 Особенности внешней шины

– Ширина шины данных: 8, 16 или 32 разряда;

– Конвейерные передачи с программируемым числом этапов конвейера;

– Программируемые состояния простоя (IDLE);

– Протокол поддерживает циклы ожидания, добавляемые ведомым устройством с использованием вывода ACK;

– Интерфейс EPROM и FLASH: 8-разрядная шина данных с фиксированным числом циклов ожидания, поддержкой чтения и записи;

– интерфейс SDRAM;

– Поддержка медленных устройств ввода/вывода;

– Поддержка передач через DMA для внешних устройств ввода/вывода в режиме квитирования.

13.1.2 Внешние контакты ввода/вывода интерфейса шины

Ниже представлено описание основных выводов внешнего интерфейса, используемых для реализации обмена со всеми типами памяти. Буква «**n**» в начале названия вывода указывает на низкий активный уровень сигнала.

ADDR22-0

Шина адреса. Выдает на эти линии адрес устройства, к которому выполняется доступ.

DATA31–0

Шина данных с тремя состояниями. Процессор выдает данные на эти выводы или принимает данные или команды с помощью этих выводов. Имеется возможность задания размера шины (8, 16 или 32 бита) для работы с различными устройствами системы.

nRD

Чтение памяти. nRD активируется всякий раз, когда процессор осуществляет чтение из подчиненного устройства системы. Активный уровень низкий. Не используется при доступе к SDRAM.

nWR

Запись слова. nWR активируется в случае, когда происходит запись во внешнее устройство. nWR изменяется одновременно с выводами ADDR. Активный уровень низкий. Не используется при доступе к SDRAM.

ACK

Подтверждение готовности. Процессор анализирует вход ACK во время доступа к устройствам. Внешние подчиненные устройства могут деактивировать (установить в низкий уровень) ACK чтобы добавить состояние ожидания во время доступа к ним. ACK используется подчиненными устройствами на фазе данных. Высокий уровень на линии ACK означает готовность устройства, низкий – требование подождать. ACK не используется при доступе к SDRAM.

nBMS

Выбор загрузочной памяти. nBMS активизируется, когда процессор выполняет доступ к загрузочной памяти (EPROM или флэш). nBMS изменяется одновременно с выводами ADDR. Активный уровень низкий.

nMS1-0

Выбор памяти. nMS0 или nMS1 активируются в то время, когда процессор желает получить доступ к банкам памяти 0 или 1 соответственно. Выходы nMS1–0 являются декодированными сигналами адреса памяти и изменяются параллельно с выводами ADDR. Когда линии ADDR31–27 равны 0b00110, nMS0 активирован. Когда линии ADDR31–27 равны 0b00111, nMS1 активирован. Активный уровень низкий.

MSSD3–0

Выборка памяти SDRAM. MSSD0, MSSD1, MSSD2 или MSSD3 активны (один из четырех), всякий раз, когда процессор выполняет доступ к пространству памяти SDRAM. MSSD3–0 являются выводами декодированного адреса памяти и активны в то время, когда процессор выполняет командный цикл SDRAM. Некоторые команды активируют сразу все четыре линии.

nRAS

Выборка адреса строки. Во время чтения или записи SDRAM низкий уровень сигнала RAS указывает на то, что на шине адреса находится достоверный адрес строки. В остальных случаях доступа к SDRAM он определяет тип операции для выполнения в соответствии со спецификацией SDRAM.

nCAS

Выборка адреса столбца. Во время чтения или записи SDRAM низкий уровень сигнала CAS указывает на то, что на шине адреса находится достоверный адрес столбца. В остальных случаях доступа к SDRAM он определяет тип операции для выполнения в соответствии со спецификацией SDRAM.

DQM

Маска слова данных SDRAM. При высоком уровне сигнала переводит буферы данных DQ SDRAM в третье состояние. При чтении всегда равен нулю. При операциях записи DQM является активным (равным единице), для запрещения записи слова.

SDA10

Вывод разряда 10 адреса SDRAM. SDRAM использует отдельный вывод адресной линии номер 10 из-за того, что она используется в специальных операциях и может задействоваться в то время, когда процессор выполняет на внешней шине транзакцию не связанную с SDRAM.

SDCKE

Разрешение тактового сигнала SDRAM. Вход синхронизации микросхемы динамической памяти может быть постоянно активным, но микросхема будет управляться данным синхросигналом, только если на линии SDCKE высокий уровень. Низкий уровень сигнала SDCKE используется для перевода памяти в режим саморегенерации.

nSDWE

Разрешение записи SDRAM. При низком уровне сигнала во время активного CAS, SDWE указывает на выполнение записи SDRAM. При высоком уровне сигнала во время активного CAS, SDWE указывает на выполнение чтения SDRAM. В остальных операциях SDRAM, SDWE определяет тип операции для выполнения в соответствии со спецификацией SDRAM.

13.1.3 Структура интерфейса внешней памяти

Интерфейс внешней памяти может иметь 8-, 16- или 32-разрядную организацию шины данных, при этом оперировать можно 8-, 16-, 32-, 64- или 128-битными данными. Адрес этих данных должен быть выровнен по их размеру.

Минимальным размером передачи на внешней шине является транзакция стандартного слова (32 бита). В зависимости от ширины внешней шины транзакция стандартного слова будет занимать различное число циклов:

– при 32-разрядной шине: один цикл;
– при 16-разрядной шине: два цикла.

При 16-разрядной шине данных транзакция стандартного слова разбивается на два цикла следующим образом: сначала передается младшая часть слова, а затем старшая. Основным отличием при 16-разрядной шине данных при выдаче адреса является то, что

32-разрядный внутренний адрес сдвигается влево на один разряд перед выдачей на внешние контакты. Самый младший бит адреса в первом цикле обмена всегда равен нулю, а во втором цикле равен единице.

В случае, когда размер обращения к внешней памяти больше стандартного слова (32 бита), передача данных выполняется за несколько транзакций стандартного слова.

Обращения к байтам или коротким словам интерфейс внешней шины также преобразует в транзакции стандартного слова. При этом перед обращением к памяти адрес байта или короткого слова предварительно преобразуется в адрес слова, и затем выполняется операция. Подробнее про операции с байтами и короткими словами приведено в ТСКЯ.431281.002РП, Команды IALU (загрузка/сохранение/пересылка).

При чтении байта или короткого слова из внешней памяти всегда считывается стандартное слово, а затем осуществляется выбор необходимого значения.

При записи байта или короткого слова во внешнюю память обращение выполняется в два этапа:

– выполняется считывание из внешней памяти стандартного слова по адресу обращения;

– выполняется модификация 8- или 16-битных данных в считанном слове, после чего осуществляется запись модифицированного слова в память.

Число обращений для выполнения операций чтения и записи данных различной разрядности при различной организации шины данных (16/32 разряда) представлено в таблице 99.

Таблица 99 – Число обращений на внешней шине

Тип операции	Разрядность данных, бит	Разрядность шины, бит	Число обращений на шине
Чтение	128	32	4 чтения
Чтение	64	32	2 чтения
Чтение	32, 16, 8	32	1 чтение
Чтение	128	16	8 чтений
Чтение	64	16	4 чтения
Чтение	32, 16, 8	16	2 чтения
Запись	128	32	4 записи
Запись	64	32	2 записи
Запись	32	32	1 запись
Запись	16, 8	32	1 чтение + 1 запись
Запись	128	16	8 записей
Запись	64	16	4 записи
Запись	32	16	2 записи
Запись	16, 8	16	2 чтения + 2 записи

Внешний порт (EBIU) всегда является мастером на внешней шине, а также мастером и подчиненным на внутренних шинах процессора. Рисунок 18 показывает структуру внутренних шин, соединяющих внешний порт с внутренними модулями процессора.

В структуре внешнего порта два главных элемента передачи данных:

– выходное FIFO (OFIFO) передачи запросов на внешнюю шину;

– входное FIFO (IFIFO) входных запросов с внешней шины или возвращаемых данных с внешней шины.

Буфер IFIFO используется в случае передачи данных во внутренний приемник в ответ на чтение ядром процессора или контроллером DMA устройств на внешней шине. В этой ситуации внешний интерфейс работает как мастер на внешней шине и возвращает прочитанные данные. Далее данные отправляются на SOC-шину (контроллер DMA или устройства) или в ядро процессора (в IFIFO SOC-интерфейса).

Выходной буфер OFIFO внешнего порта используется для всех транзакций, направленных во внешнее адресное пространство. Это могут быть запросы процессорного ядра или контроллера DMA.

Потоки данных, которые имеют место при различных вариантах передач, представлены на рисунке 53. Возможны операции чтения и операции записи. Операции, выполняемые ядром или контроллером DMA.

Если операция является операцией записи, она завершается:

– для ядра процессора в момент записи данных в SOC OFIFO;

– для контроллера DMA в момент записи в EBIU OFIFO.

Если выполняется операция чтения, данные проходят следующие этапы:

– При чтении ядром процессора, запрос помещается в SOC OFIFO, далее он перемещается в EBIU OFIFO, где читается и анализируется внешним интерфейсом. Выполняется цикл чтения внешнего устройства и прочитанные данные размещает в EBIU IFIFO, откуда они пересылаются в SOC IFIFO. SOC интерфейс принятые данные отправляет в регистр назначения.

– При чтении контроллером DMA, запрос помещается в EBIU OFIFO, где затем читается и анализируется внешним интерфейсом. Выполняется цикл чтения внешнего устройства и прочитанные данные размещает в EBIU IFIFO, откуда они пересылаются в буфер контроллера или буфера передатчиков устройств.

При доступе к ресурсам со стороны многих устройств используется приоритет доступа. EBIU IFIFO участвует в доступе к SOC-шине и к SOC-IFIFO. При доступе к SOC-шине приоритет следующий:

– SOC OBUF (наивысший);

– EBIU IFIFO;

– процессорное ядро (низший).

При доступе к SOC IFIFO приоритет следующий:

– чтение ядром регистра SOC-шины;

– EBIU IFIFO;

– контроллер DMA.

Приоритет устройств фиксированный и не зависит от приоритета транзакции DMA или приоритета транзакции внешнего мастера.

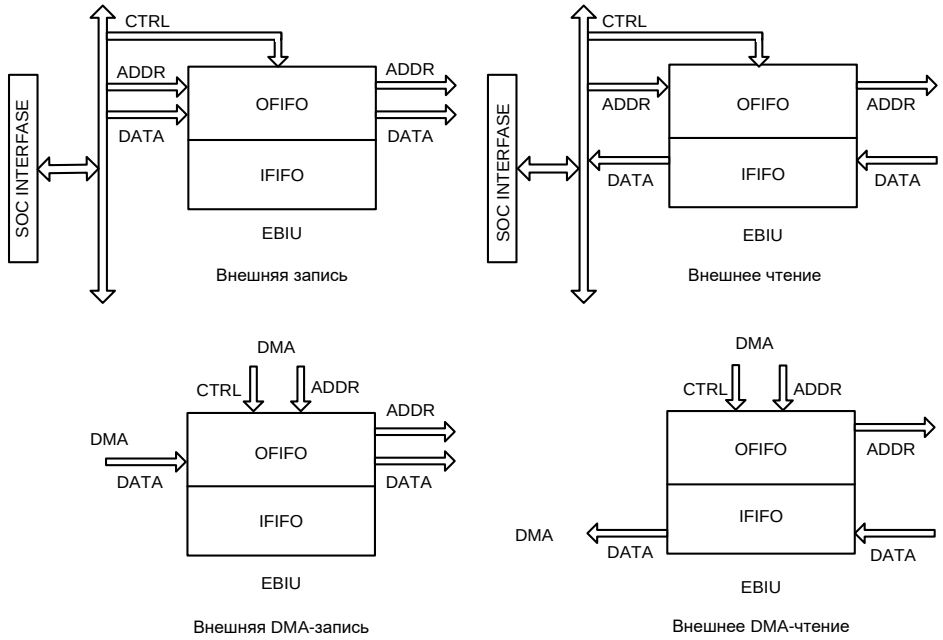


Рисунок 53 – Передача данных через интерфейс внешней шины (EBIU)

13.1.4 Программирование регистра SYSCON

Регистр SYSCON является регистром конфигурации контроллера внешнего интерфейса и должен быть запрограммирован до того, как начнется передача данных на шине (если значения по умолчанию должны изменяться). В режиме пользователя доступ к SYSCON невозможен. Описание разрядов регистра приведено в таблице 100.

Таблица 100 – Регистр SYSCON

Бит	Имя	Назначение
0	BNK0IDLE	Вставка пустого цикла между операциями доступа к банку памяти MS0: 1 – вставить цикл; 0 – нет
2:1	BNK0WAIT	Количество внутренних циклов ожидания при обращении к банку MS0: 00 – ноль циклов; 01 – один цикл; 10 – два цикла; 11 – три цикла
4:3	BNK0PIPE	Глубина конвейера банка памяти MS0: 00 – один цикл; 01 – два цикла; 10 – три цикла; 11 – четыре цикла
5	BNK0SLOW	Тип протокола обмена для банка MS0: 1 – медленный (асинхронный); 0 – синхронный (конвейерный)
6	BNK1IDLE	Вставка пустого цикла между операциями доступа к банку памяти MS1: 1 – вставить цикл; 0 – нет

Бит	Имя	Назначение
8:7	BNK1WAIT	Количество внутренних циклов ожидания при обращении к банку MS1: 00 – ноль циклов; 01 – один цикл; 10 – два цикла; 11 – три цикла
10:9	BNK1PIPE	Глубина конвейера банка памяти MS1: 00 – один цикл; 01 – два цикла; 10 – три цикла; 11 – четыре цикла
11	BNK1SLOW	Тип протокола обмена для банка MS1: 1 – медленный (асинхронный); 0 – синхронный (конвейерный)
17:12		Зарезервировано
18	-	Всегда 0
19	MEMWIDTH	Ширина шины данных при обращении к внешней памяти 1 – 8 (для микросхем с ревизии 3) или 16 бит; 0 – 32 бита
21:20		Зарезервировано
22	MS0_PIPE	Управление конвейерным режимом по записи для nMS[0] (для микросхем с ревизии 3): 0 – задержка данных 1 такт; 1 – задержка данных 2 такта
23	MS1_PIPE	Управление конвейерным режимом по записи для nMS[1] (для микросхем с ревизии 3): 0 – задержка данных 1 такт; 1 – задержка данных 2 такта
24		Режим работы входного FIFO внешнего интерфейса 0 – нормальный режим; 1 – ускорение передачи данных. Бит можно только установить. Сброс только аппаратно
25		Режим работы выходного буфера внешнего интерфейса 0 – нормальный режим; 1 – ускорение передачи данных. Бит можно только установить. Сброс только аппаратно
26		Всегда 0
27	XSDM	Модификация адресного пространства: 0 – стандартное адресное пространство; 1 – модифицированное адресное пространство, в котором адреса с 0x10000000 по 0x2FFFFFFF соответствуют адресному пространству динамической памяти. Используется для поддержки байтовой адресации внешней памяти
31:28		Всегда 0

Ширина шины данных

Физически шина данных внешнего интерфейса имеет 32 линии. Однако это не значит, что подключаемая память или внешнее устройство обязательно должны иметь такую же разрядность шины данных. С помощью регистра PX_ALT можно задать ширину шины данных равной 32, 16 или 8 бит. Контроллер внешнего интерфейса будет управлять тем количеством выводов шины данных, значение которых определено в регистре PX_ALT. Вместе с этим сам контроллер поддерживает алгоритм работы с 32-, 16- или 8-разрядной внешней шиной данных. Это означает, что если в регистре SYSCON[19] задана разрядность шины данных 32 бита, то, несмотря на установки в регистре PX_ALT, интерфейс будет вести себя так, как будто бы шина имеет разрядность 32 бита. Если в регистре SYSCON[19] установлена разрядность шины данных 16 бит, все словные операции интерфейс будет превращать в две последовательные операции по 16 бит.

Для микросхем с ревизии 3

Если в регистре SYSCON[19] задана разрядность 16 бит, а в регистре PX_ALT – 8 бит, то все словные операции будут представлять собой четыре последовательных операции на внешней восьмибитной шине.

Для 8-, 16- и 32-битной шины есть отличие в операциях записи при байтовой адресации памяти. Для 16- и 32- битной шины операция записи представляет собой две операции – чтение и последующую запись (чтение-модификация-запись). Для восьмибитной же шины предварительного чтения для записи байта не требуется.

Протокол медленного устройства для шины

Для настройки протокола медленного устройства устанавливается в разряд 1 протокола медленного устройства BNK#SLOW (разряд 5 для банка 0, разряд 11 для банка 1). В случае выбора протокола медленного устройства некоторые поля (поле глубины конвейера PIPE, разряд IDLE) являются безразличными и не используются. Поле ожидания определяет число внутренних циклов ожидания при медленных доступах. Если число внутренних ожиданий равно нулю, механизм внешних ожиданий (анализ входа АСК) не может использоваться для данных транзакций.

Конвейерный протокол для шины

Для выбора конвейерного протокола бит BNK#SLOW (HOSTSLOW) соответствующего банка памяти устанавливается в «0», а поле PIPE устанавливается на заданную глубину конвейера (0b00 для одного цикла, 0b01 для двух циклов, 0b10 для трех циклов и 0b11 для четырех циклов). Так определяется глубина конвейера только для транзакций чтения. Для микросхем ревизии 2 глубина конвейера для транзакций записи всегда равна одному циклу, для микросхем с ревизии 3 – определяется битами MS0_PIPE, MS1_PIPE. Разряд IDLE устанавливается, если на шине есть несколько подчиненных устройств в одном банке памяти, с целью предотвращения конфликта на шине данных между различными подчиненными устройствами при последовательном чтении. Если банк памяти содержит только одно подчиненное устройство, разряд IDLE сбрасывается. Поле внутреннего ожидания безразлично при конвейерных транзакциях.

Начальные значения для работы шины

После сброса регистр SYSCON имеет начальное значение равное 0x80067. Программа начальной загрузки меняет содержимое регистра SYSCON в зависимости от выбранного способа загрузки (см. раздел 32 «Начальный старт процессора» и текст загрузочной программы). Пользователь может не менять данное значение, если оно соответствует конфигурации его внешней шине. Начальное значение определяет:

– банк 0: медленный протокол, три состояния ожидания, включен холостой режим;

– банк 1: конвейерный протокол, конвейер глубиной один цикл, нет состояний ожидания, включен холостой режим;

В зависимости от выбранного способа загрузки BOOT[2:0] регистр SYSCON принимает значения, указанные в таблице 91.

Таблица 101 – Значение регистра SYSCON в зависимости от режима загрузки

BOOT[2:0]	Значение регистра SYSCON
000	0x809e7
001	0x9e7
010 – 110	0x80067

13.1.5 Интерфейс конвейерного протокола

Процессор использует конвейерный протокол для соединения с быстрой синхронной памятью, подключаемой к nMS1-0. Конвейерный протокол обеспечивает передачу данных со скоростью частоты внешней шины. При конвейерном доступе адрес и управляющие разряды выдаются одновременно в одном такте, а данные лишь спустя несколько циклов: от одного до четырех в зависимости от направления передачи и установленных параметров. Процессор может выдавать адрес следующей транзакции еще до прихода данных предыдущей транзакции.

Управляющие сигналы, используемые в конвейерных транзакциях:

– **nRD** – если активен указывает на транзакцию чтения;

– **nWR**– транзакция записи, если один из этих сигналов активный;

– **ACK** – управляется адресуемым подчиненным устройством в течение цикла данных. Если имеет высокий уровень, подчиненное устройство готово завершить цикл данных, в противном случае мастер шины должен ждать готовности подчиненного устройства.

Базовая конвейерная транзакция

Временная диаграмма базовой конвейерной транзакции представлена на рисунке 54. Во время адресного цикла, адрес передается вместе с сигналами RD или WR.

Цикл данных может начинаться с задержкой от одного до четырех тактов, в соответствии с параметрами, указанным в регистре конфигурации для подчиненного устройства, и направлением транзакции. Задержка между циклом адреса и циклом данных есть глубина конвейера. В примере глубина конвейера равна двум (рисунок 54).

В цикле данных данные передаются в соответствии с направлением транзакции (чтение или запись). Если подчиненное устройство готово к завершению транзакции, оно активирует сигнал ACK (равен единице) и принимает или выставляет данные. Если подчиненное устройство не готово, оно деактивирует сигнал ACK (равен нулю) и задерживает данные.

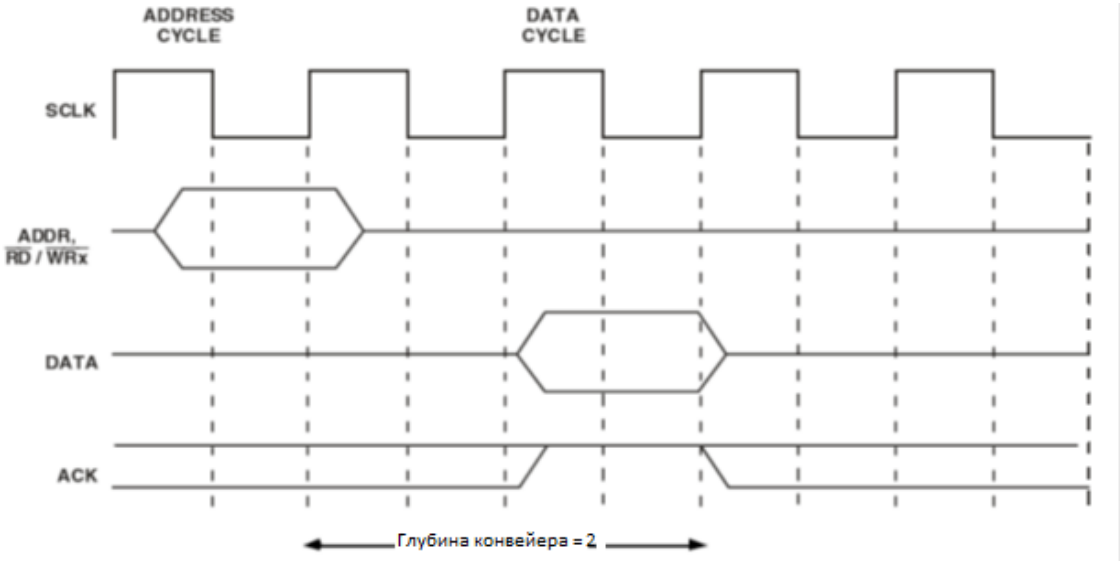


Рисунок 54 – Основные конвейерные передачи

Глубина конвейера программируется для каждого из банков памяти, а также зависит от типа транзакции (чтение или запись). Для микросхем ревизии 2 глубина конвейера для транзакций записи всегда равна одному циклу. Для микросхем с ревизии 3 – определяется битами MS0_PIPE, MS1_PIPE. При чтении из внешних банков памяти глубина конвейера может быть до четырех циклов. Глубина конвейера может быть выбрана индивидуально для каждого их банков в регистре SYSCON после сброса.

Циклы ожидания

В транзакциях чтения, если подчиненное устройство вовремя готово к циклу данных выполняемой транзакции, подчиненное устройство активирует сигнал ACK в цикле данных. Если подчиненное устройство не готово, оно деактивирует сигнал ACK в цикле данных и держит его неактивным (равным нулю) пока не будет готово продолжить транзакцию. ACK также может быть деактивирован после транзакции записи, если подчиненное устройство не может вовремя отправить данные в место назначения.

Сигнал ACK может быть выставлен в любом цикле данных в многоцикловых транзакциях. Нет никаких ограничений на продолжительность удерживания ACK в нуле.

Рисунок 55 демонстрирует использование сигнала ACK. В этом примере, данные DA1 не готовы вовремя и выставляются на один такт позже. Сигнал ACK деактивируется во время цикла данных DA1 и активируется в следующем цикле, чтобы указать на достоверные данные. Из-за этого цикла ожидания:

– мастер берет с шины данные на один цикл позже;

– адресный цикл, следующий за сигналом ACK (AB2) растягивается на один цикл;

– все подчиненные устройства с приостановленными транзакциями во время цикла, следующего за сигналом ACK, замедляются на число циклов, которые были деактивированы сигналом ACK. Например, между адресным циклом AB0 и циклом данных DB0 проходит пять циклов, хотя глубина конвейера равна только четырем.

При этом адресные и управляющие сигналы на линиях в цикле, где ACK был в нуле, должны защелкиваться подчиненными устройствами.

Транзакции записи используют циклы ожидания ACK не так, как транзакции чтения. В случае записи подчиненное устройство активирует сигнал ACK, пока у него имеется свободное место в буфере приема.

Ниже представлен пример, в котором адресный цикл AA3 заставляет подчиненное устройство деактивировать сигнал ACK (рисунок 56). Это может быть вызвано тем, что подчиненное устройство приняло 128-разрядное слово в буфер, и он занят для следующей транзакции. Остановка удлиняет конвейер транзакции на два цикла. Циклы, которые были остановлены ACK, начинаются спустя один такт после деактивации сигнала ACK – например, AB2 и DB1 дополнены двумя циклами.

При этом все подчиненные устройства принимают адрес AB1, а устройство, которому адресованы данные DB0, также принимает эти данные во внутренний буфер.

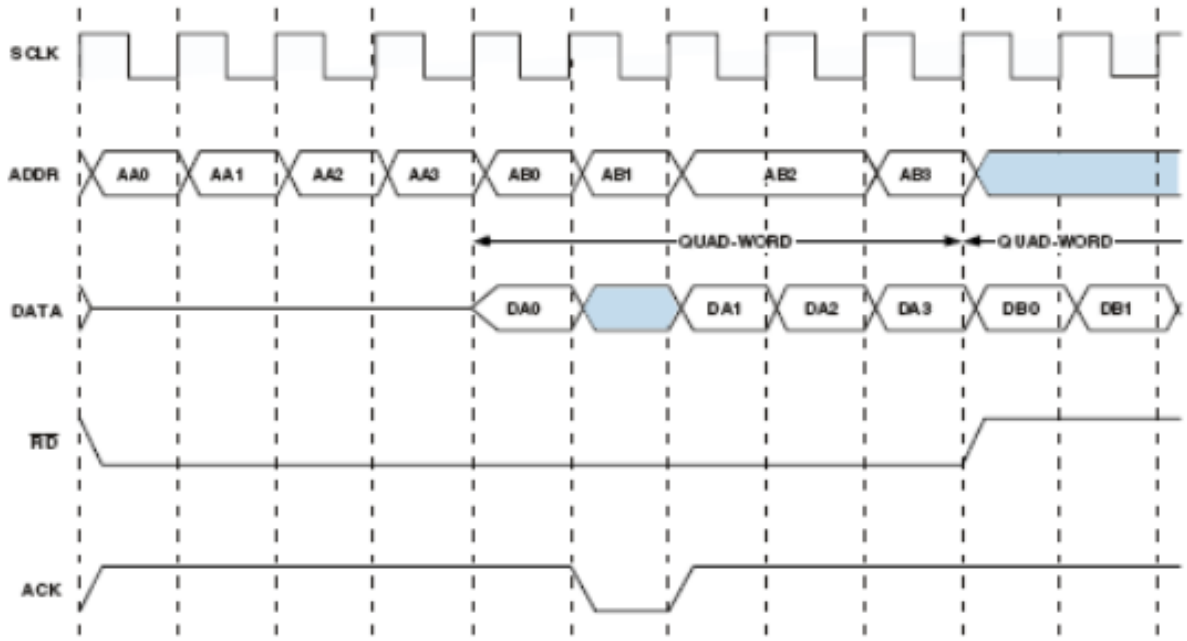


Рисунок 55 – Доступ к пакетному чтению с последующим пакетным чтением

Транзакции AB0 и AB1 адресуют другое подчиненное устройство (рисунок 56). Если первое подчиненное устройство переводит сигнал ACK на низкий уровень и следующие транзакции соответствуют другому подчиненному устройству, новое подчиненное устройство транзакции не может управлять сигналом ACK до тех пор, пока первое подчиненное устройство не завершит транзакцию.

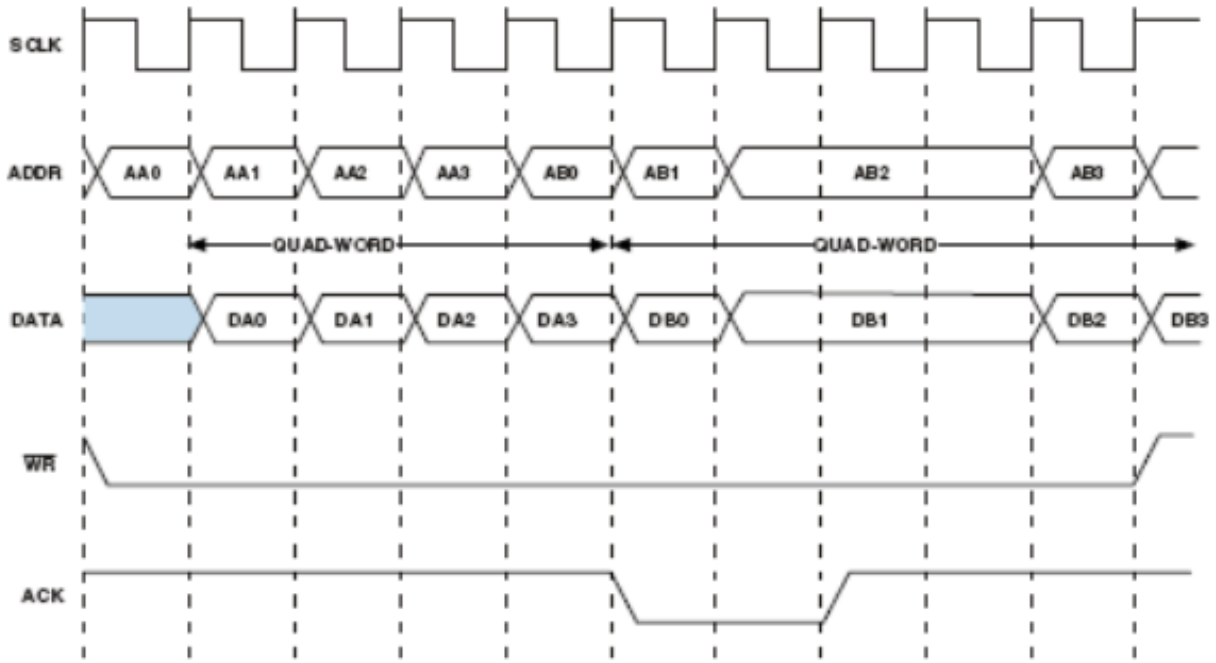


Рисунок 56 – Доступ к пакетной записи с последующей пакетной записью

13.1.6 Интерфейс протокола медленного устройства

Процессор поддерживает протокол медленных устройств, который может быть использован для простых устройств. Этот протокол может быть сконфигурирован для адресных пространств, принадлежащих банку 0 или банку 1. Протокол устанавливается посредством программирования соответствующих разрядов в регистре SYSCON:

– бит SLOW равен единице;

– биты PIPE глубины конвейера равны нулю;

– циклы ожидания WAIT программируются в соответствии с требованиями системы;

– цикл IDLE (холостой цикл) равен единице.

Целью таких установок является прямое соединение с простыми, медленными, некритическими модулями памяти или периферийными устройствами. Протокол может работать с синхронными и асинхронными устройствами. Базовый протокол с конфигурацией «без циклов ожидания» представлен на рисунках 57, 58.

В первом цикле выдается адрес и активируется линия выбора соответствующего банка памяти.

Во втором цикле активируются сигналы RD или WR (в зависимости от типа транзакции). Если транзакция записи, в этом цикле процессор выдает данные. В случае транзакции чтения в этом цикле подчиненное устройство начинает выдавать данные. В случае чтения, в конце цикла процессор защелкивает данные с шины во внутреннем регистре.

В третьем цикле процессор продолжает удерживать адрес и данные (если выполняется запись), но деактивирует сигналы RD или WR. В случае чтения устройство перестает выдавать данные на шину.

Для конфигурации «без циклов ожидания», циклы ожидания не могут быть добавлены деактивирующим сигналом ACK.

Одно из ключевых требований при организации протокола медленного устройства – гарантированное время удержания адреса и данных. Из временной диаграммы видно, что процессор обеспечивает один такт предустановки адреса и один такт удержания адреса и данных. Дополнительные такты предустановки данных могут обеспечиваться внутренними программируемыми циклами ожидания или внешним сигналом АСК.

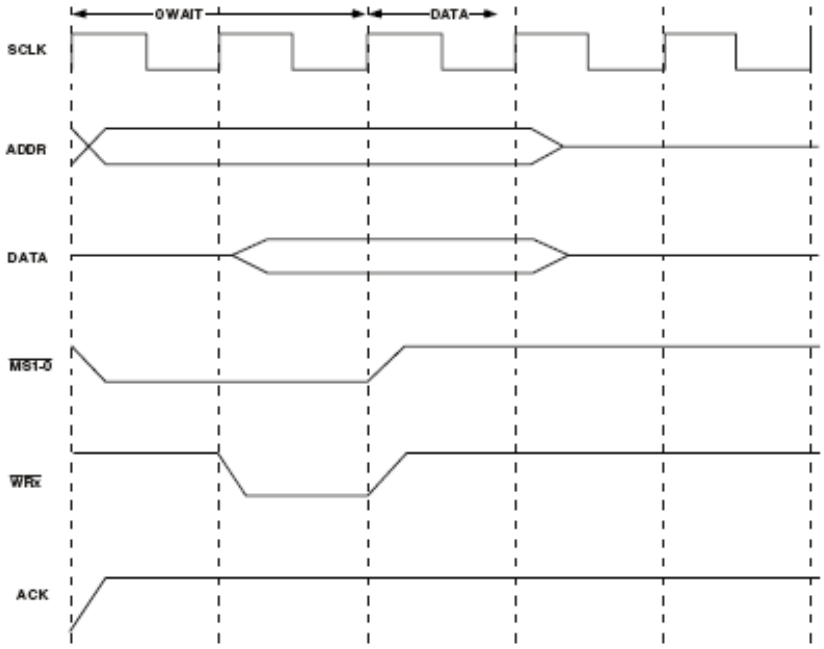


Рисунок 57 – Медленный протокол. Запись с нулевым временем ожидания

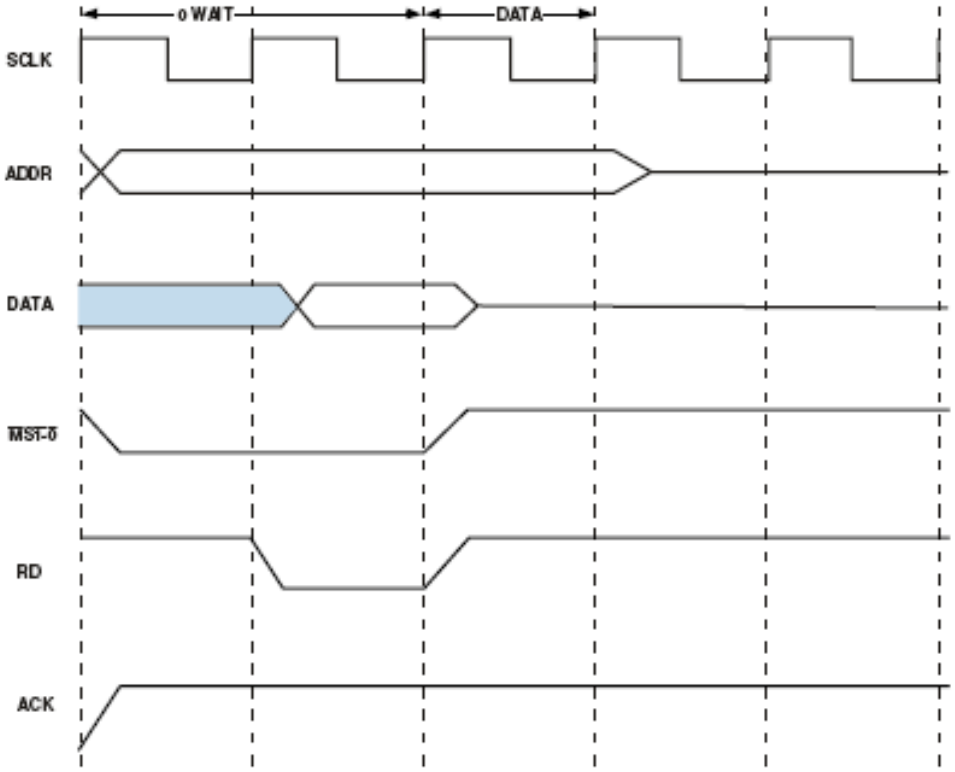


Рисунок 58 – Медленный протокол. Чтение с нулевым временем ожидания

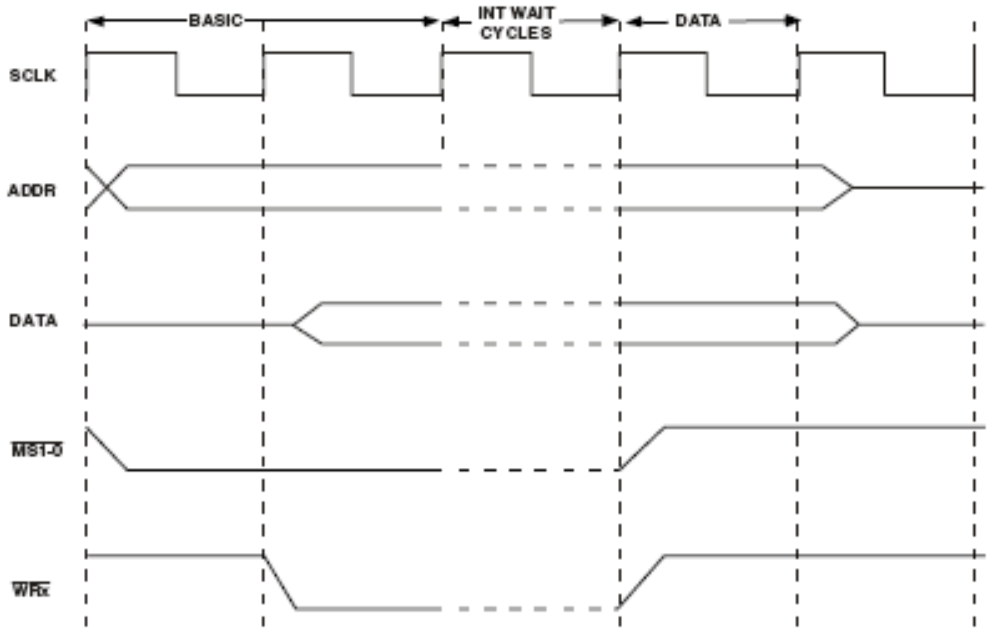


Рисунок 59 – Медленный протокол. Запись с циклами ожидания

В протоколе медленного устройства при необходимости могут быть вставлены циклы ожидания. Регистр SYSCON включает в себя поле задания некоторого числа внутренних циклов ожидания. Значения могут быть от нуля до трех.

Рисунки 59 и 60 показывают медленные транзакции с одним или более циклами ожидания. Ситуация похожа на транзакцию без циклов ожидания, но второй цикл (цикл в котором активированы RD или WR) повторяется в соответствии с числом циклов внутреннего ожидания.

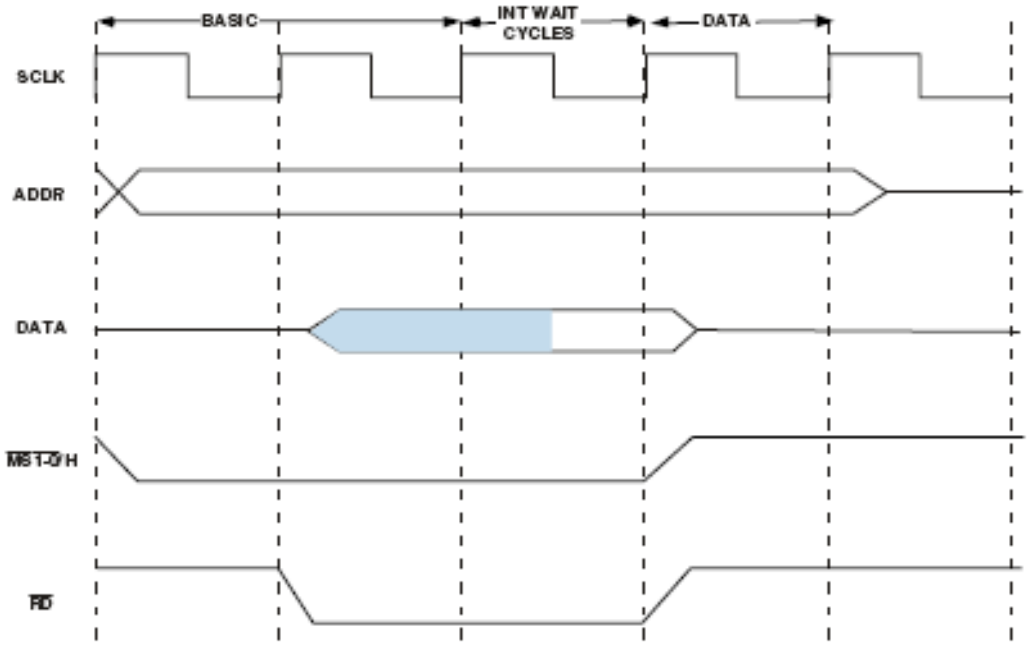


Рисунок 60 – Медленный протокол. Чтение с циклами ожидания

Дополнительные циклы внешнего ожидания могут быть вставлены путем деактивации сигнала ACK. Это может быть сделано только тогда, когда в регистре SYSCON поле WAIT не равно нулю. Для того, чтобы добавить циклы внешнего ожидания, сигнал ACK должен быть деактивирован в течение одного или более циклов перед последним циклом ожидания (см. рисунок 61).

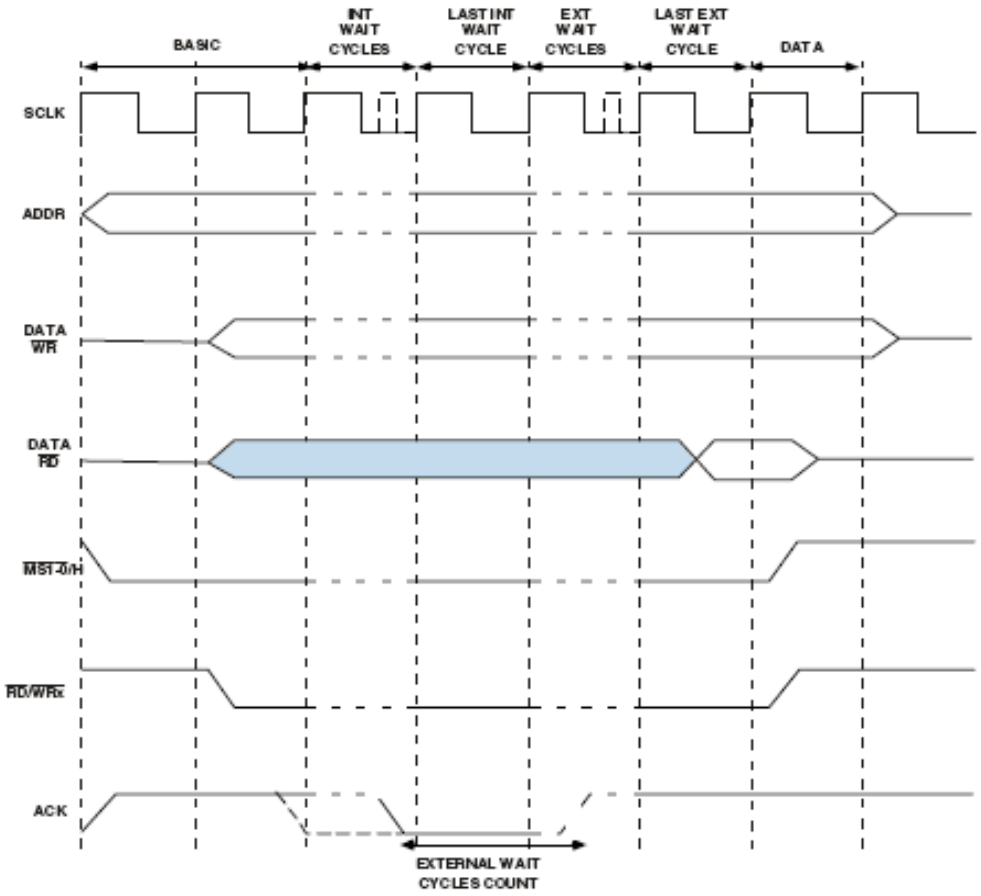


Рисунок 61 – Медленный протокол. Запись/чтение с внешними циклами ожидания

13.1.7 Интерфейс EPROM

Процессор имеет возможность работать с внешней восьмиразрядной перепрограммируемой памятью типа EPROM. Во время сброса процессор может быть сконфигурирован для загрузки из внешнего EPROM. В этом случае программа загружается из EPROM во внутреннюю память автоматически как часть последовательности сброса. Для этой цели используется канал 0 DMA.

Поскольку разрядность шины данных памяти EPROM составляет 8 бит, внешний интерфейс процессора упаковывает принимаемые байты в 32-разрядные данные.

Процессор может работать с EPROM не только во время сброса. Имеется возможность доступа и в процессе выполнения программы. Однако в работе с EPROM имеется одна особенность – память EPROM не является частью адресного пространства процессора. Процессор поддерживает словную адресацию внешней памяти, а память EPROM имеет байтовую организацию. В связи с этим для работы с EPROM можно использовать только канал общего назначения DMA. Для работы с EPROM тип обмена в TCB канала устанавливается равным шести. При чтении EPROM, обмен может происходить только между EPROM и внутренней памятью. Внешний интерфейс определяет, что текущий запрос есть запрос DMA и его тип равен шести. Это позволяет интерфейсу выбрать для обмена протокол медленного устройства с фиксированным числом циклов внутреннего ожидания равным 16. При выполнении внешней транзакции процессор активизирует линию BMS для выбора EPROM. Для передачи данных используются биты с нулевого по седьмой шины данных.

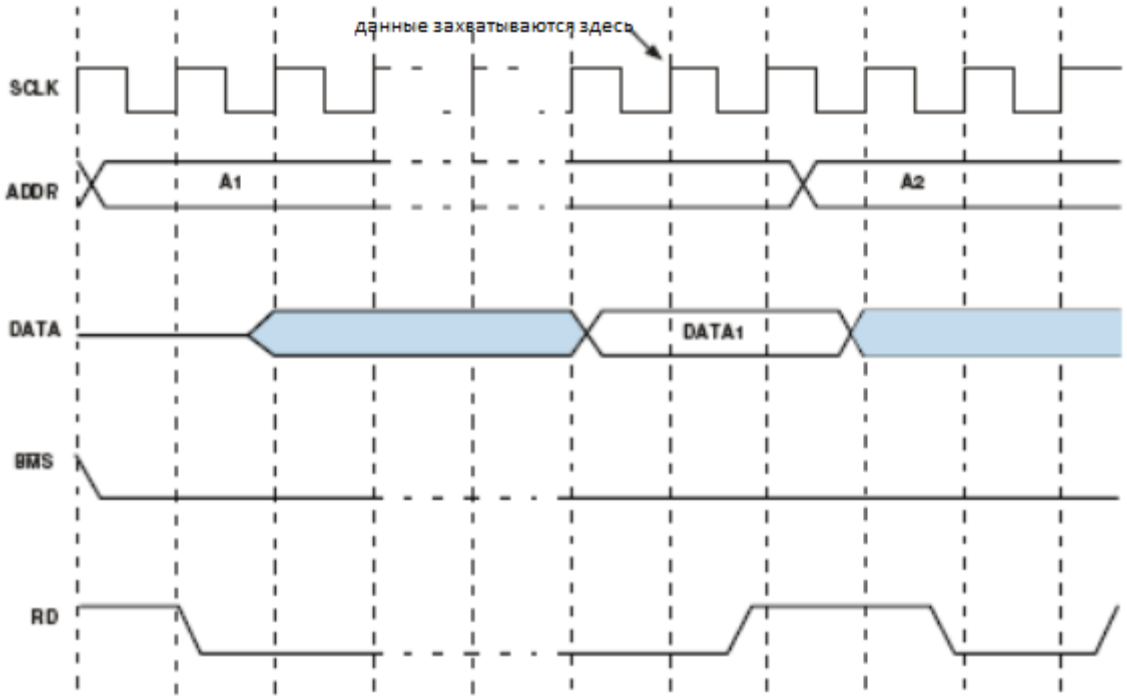


Рисунок 62 – Доступ к загрузке EPROM – 16 циклов ожидания

Возможно, как чтение внешней EPROM, так и операция записи для программирования памяти. Во время чтения интерфейс упаковывает байты в слова и передает их во внутреннюю память. Операция записи идентична чтению, но с одним отличием – при записи нет распаковки слова в байты. Процессор может выдавать на внешнюю шину только 32-разрядные слова, а EPROM может принимать только младший байт. Поэтому при записи программист сам должен подготовить информацию в удобном для записи виде. При записи EPROM источником данных может быть как внутренняя, так и внешняя память.

Во внутреннем или внешнем ОЗУ процессора данные должны быть организованы так, как показано на рисунке 63. Слева исходные данные для программирования. Справа показаны распакованные в слова байты для передачи во внешнюю EPROM.

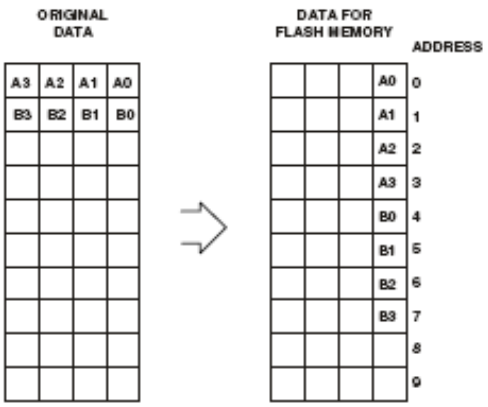


Рисунок 63 – Подготовка данных для записи в восьмиразрядную EPROM

При выполнении обмена с EPROM, внешний интерфейс всегда устанавливается размер шины данных 32 разряда, несмотря на установки в SYSCON. При этом адрес EPROM не должен быть в используемом адресном пространстве SDRAM или банков MS1 и MS0. Лучше всего старшие биты адреса установить в ноль. При чтении внешний

интерфейс автоматически устанавливает размер данных равным квадрослову. Это позволяет произвести четыре цикла чтения, взять из них по одному байту и сформировать одно слово. Канал DMA при этом должен выполнять обмен словами, но увеличивать адрес на четыре. Таким образом, первое считывание будет по адресам 0, 1, 2, 3, а второе по адресам 4, 5, 6, 7. Такой способ используется при чтении.

При записи используется размер данных, который задает канал DMA. По сути при записи внешний интерфейс работает с EPROM так как будто эта память имеет разрядность 32 бита и передает ей слова данных. Основной хитростью здесь является предварительная подготовка данных для записи в памяти.

13.1.8 Интерфейс SDRAM

Процессор имеет выделенное адресное пространство для подключения синхронной динамической памяти SDRAM. Четыре банка в этом адресном пространстве выбираются линиями выборки MSSD3–0. SDRAM доступно в адресном диапазоне с 0x4000 0000 до 0x7FFF FFFF.

Процессор поддерживает стандартную (3,3 В) и пониженной мощности (2,5 В) SDRAM. Память работает с использованием частоты внешней шины SCLK (системная частота). Все входы защелкиваются и все выходы корректны при положительном (из «0» в «1») перепаде частоты. Синхронный интерфейс позволяет выполнять передачу данных каждый такт. SDRAM поддерживает несколько типов пакетных доступов в зависимости от инициализации регистра режима SDRCON. Каждый тип доступа предваряется запуском соответствующей команды в SDRAM. Специальные режимы SDRAM должны инициализироваться в регистре SDRCON.

SDRAM имеет внутреннюю организацию в виде двух или четырех банков. Выводы выбора банка SDRAM определяют, к какому банку происходит обращение. SDRAM имеет программируемый параметр задержки чтения, который должен инициализироваться приложением в соответствии с типом устройства и рабочей тактовой частотой.

Для соответствия требуемым временным характеристикам SDRAM, процессор может выполнять конвейерную передачу адреса и управляющих команд SDRAM. Для этого используется бит глубины конвейера в регистре SDRCON.

Микросхемы SDRAM поставляются различными производителями. Каждый поставщик имеет свои временные требования касательно параметров задержки команд ACT - PRE (tRAS) и PRE - ACT (tRP). Для поддержки всех основных поставщиков и различных уровней скорости, регистр SDRCON программируем для того, чтобы разработчик мог удовлетворить временные требования микросхемы SDRAM.

Такими параметрами являются битовые поля: CAS задержка, PRE - RAS задержка, RAS - PRE задержка.

Наилучшая скорость обмена с SDRAM достигается при использовании блочного обмена, когда процессор выполняет несколько последовательных чтений или записей в один банк памяти. Интерфейс процессора имеет возможность отслеживания и поддержки пакетного доступа.

Особенностью работы динамической памяти является необходимость регенерации всех ячеек памяти в течение заданного интервала времени. Это гарантирует сохранность информации. Процессор имеет возможность программировать частоту регенерации для соответствия временным требованиям микросхемы.

При описании интерфейса SDRAM используются следующие определения:

– Команда активации банка ACT

Активирует выбранный банк и фиксирует в нем новую строку. Должна использоваться перед командой чтения или записи.

– Длина пакета (Burst Length)

Определяет количество слов, которые поступают на вход или выход SDRAM после детектирования команды чтения или записи. Микросхема памяти всегда программируется для длины пакета равного полной странице.

– Тип пакетирования (Burst Type)

Определяет порядок, в котором SDRAM отправляет или принимает пакетные данные после детектирования команды чтения или записи. Процессор поддерживает только последовательный доступ.

– Задержка CAS (CAS Latency)

Задержка, в тактах, между тем, когда SDRAM определяет команду чтения и когда данные поступают на выходные выводы. Величина запаздывания определяется уровнем скорости устройства и тактовой частотой шины. Приложение должно программировать этот параметр в регистре SDRCON.

– Маскирование ввода-вывода DQM данных

Вывод DQM используется для маскирования контроллером операций записи.

– Регистр SDRCON

Регистр, который содержит программируемые конфигурационные параметры для возможности работы контроллера SDRAM с микросхемой SDRAM.

– Регистр режима (Mode register)

Регистр конфигурации SDRAM, который содержит определяемые пользователем параметры, согласованные с регистром SDRCON. Этот регистр находится внутри микросхемы памяти.

– Размер страницы (Page Size)

Процессор поддерживает микросхемы SDRAM с размером страниц (строк) 1024, 512 и 256 столбцов. Размер страницы может быть запрограммирован в регистре SDRCON.

– Команда подзарядки (Precharge)

Подзаряжает активный банк.

– Период регенерации (Refresh Rate)

Программируемая величина в регистре SD_REF. Период регенерации позволяет приложениям координировать скорость SCLK с требуемой частотой регенерации SDRAM.

– Саморегенерация (Self-Refresh)

Состояние микросхемы памяти, в котором она использует внутренний таймер и периодически инициирует автоматические регенерации памяти без внешних команд. Этот режим работы обеспечивает SDRAM режим низкого потребления.

– tRAS

Требуемая задержка между запуском команды активации строки банка и запуском команды подзарядки. Величина зависит от производителя и задается в регистре SDRCON.

– tRC

Требуемая задержка между последовательными командами активации одного банка. Данный параметр зависит от производителя и определяется как tRC = tRP + tRAS. Процессор фиксирует значение этого параметра, так что это непрограммируемая опция.

– tRCD. RAS - CAS задержка

Требуемая задержка между командой ACT и началом первой операции записи или чтения. Данный параметр зависит от производителя и определяется как tRCD = CL. Процессор использует фиксированное значение этого параметра, так что это непрограммируемая опция.

– tRP

Требуемая задержка между запуском команды подзарядки и командой активации. Данный параметр зависит от производителя и определяется в SDRCON.

Рисунок 64 показывает интерфейс контроллера SDRAM между внутренним ядром процессора и внешним устройством SDRAM.

Процессор обычно генерирует адрес внешней памяти, который затем активизирует соответствующую линию выбора SDRAM памяти (MSSD3–0), а также указывает на тип операции обмена: чтение или запись. Эта информация анализируется контроллером SDRAM. Внутренняя 32-битная шина адреса мультиплексируется контроллером SDRAM для создания соответствующего сигнала выбора памяти, адреса строки, адреса колонки и банка в SDRAM.

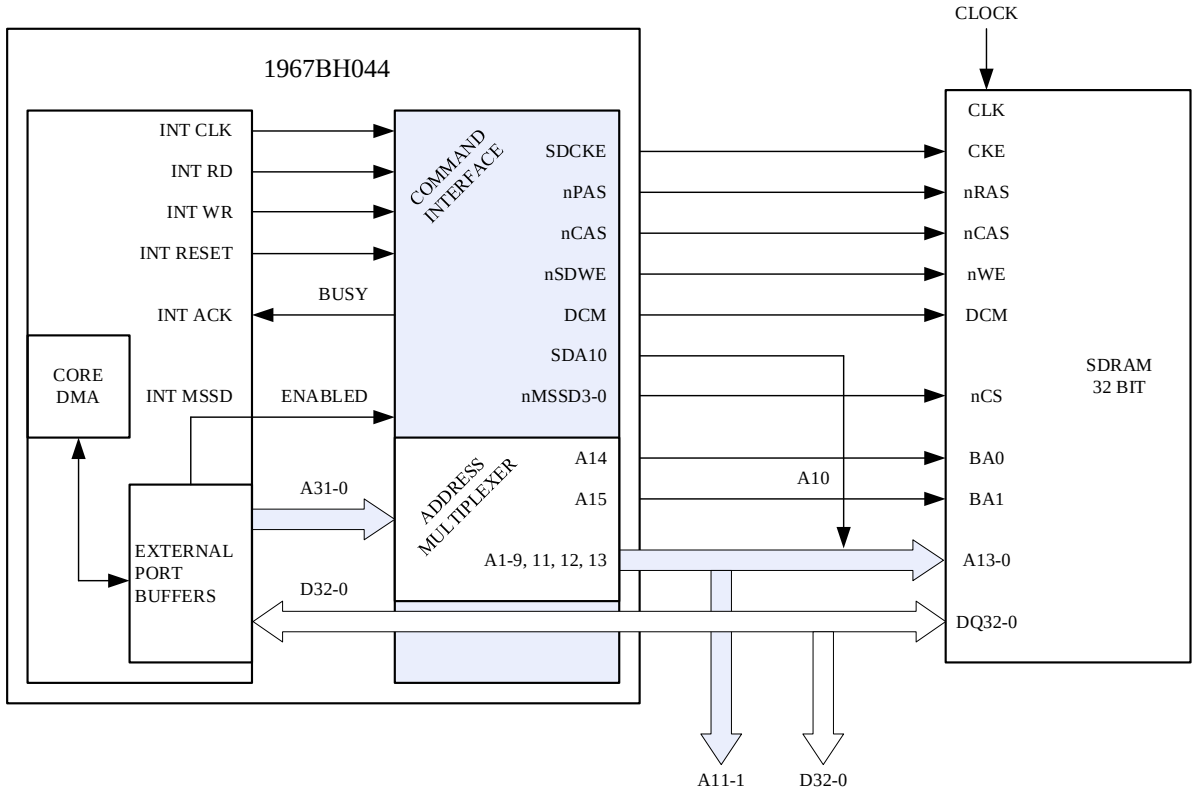


Рисунок 64 – Интерфейс контроллера SDRAM
(для конфигурации с 32-разрядной шиной)

Контакты ввода/вывода интерфейса SDRAM

Интерфейс SDRAM порта внешней шины использует для обмена с SDRAM памятью общую шину данных и часть шины адреса. Для управления микросхемой памяти используются дополнительные линии:

MSSD3–0;
RAS;
CAS;
DQM;
SDA10;
SDCKE;
SDWE.

Назначение линий было описано ранее. Выводы данных и адреса SDRAM могут соединяться напрямую с выводами данных и адреса процессора. Ширина шины данных может быть выбрана 16 или 32 разряда при программировании регистра SYSCON.

Подключения линий адреса:

– разряды 9-0 адреса SDRAM соединяются с выводами ADDR9–0 процессора;
– разряд 10 адреса SDRAM соединяется с выводом SDA10 процессора;
– разряды 15-11 адреса SDRAM соединяются с выводами ADDR15–11 (столько разрядов сколько требуется) процессора.

Выводы выбора банка

Внутренняя организация микросхемы динамической памяти включает в себя два или четыре банка. Для выбора банка микросхема имеет специальные выводы BS1-0. Эти выводы подключаются к адресным линиям процессора. Соединение выводов адреса процессора в качестве линий выбора банка для устройства SDRAM меняется в зависимости от следующих факторов:

– рабочее напряжение, используемое для SDRAM (стандартно 3,3 В или 2,5 В в SDRAM с пониженной мощностью);
– число банков.

Для двух банков BS равен ADDR11 или любой адресной линии в диапазоне ADDR15–11. Для четырех банков BS1–0 равен ADDR12–11 или BS1–0 может быть равен другой паре адресов в диапазоне ADDR15–11.

Для SDRAM с пониженной мощностью использование линий адресов в качестве сигналов выбора банка ограничено ADDR15-14. Для двух банков может быть использована любая из двух адресных линий ADDR14 или ADDR15. Для четырех банков должны использоваться ADDR15–14.

Физическое соединение внутреннего адреса процессора и адреса микросхемы SDRAM

Выше отмечалось, что SDRAM может подключаться к шине данных шириной 16 бит или 32 бита. Выбор подключения зависит от объема памяти, которую пользователь хочет установить в системе, и от желаемой скорости обмена.

В таблицах 102, 103 и 104 приведено выравнивание данных внешнего порта и соединения SDRAM для 32-разрядной системной шины.

В таблицах приведено соответствие между внешними контактами адреса процессора и контактами адреса микросхемы SDRAM, а также указано какие разряды внутреннего 32-битного адреса выдаются на эти контакты в разных фазах доступа (т.е. в фазе активации банка и выбора активного ряда (строки)), а также в фазе чтения или записи при выборе колонки (номера слова в строке).

При 16-разряной шине для передачи данных в операциях чтения (RD) и записи (WR) всегда используются два такта: сначала младшая часть слова, а затем старшая. Основным отличием при 16-разрядной шине данных при выдаче адреса является то, что 32-разрядный внутренний адрес сдвигается влево на один разряд перед выдачей на внешние контакты. Самый младший бит адреса в первом такте обмена всегда равен нулю, а во втором такте равен единице.

Таблица 102 – Размер страницы 256 столбцов, 32-разрядная шина

Физический вывод процессора	Внутренний адрес строки в цикле активации банка (ACT)	Внутренний адрес столбца в цикле доступа (Read, Write)	Физический вывод SDRAM
A0	8	0	A0
A1	9	1	A1
A2	10	2	A2
A3	11	3	A3
A4	12	4	A4
A5	13	5	A5
A6	14	6	A6
A7	15	7	A7
A8	16	8	A8
A9	17	9	A9
A10	несущественный	несущественный	NC
SDA10	18	0	A10/AP
A11	19	19	A11 или банк
A12	20	20	A12 или банк
A13	21	21	A13 или банк
A14	22	22	A14 или банк
A15	23	23	A15 или банк

Таблица 103 – Размер страницы 512 столбцов, 32-разрядная шина

Физический вывод процессора	Внутренний адрес строки в цикле активации банка (ACT)	Внутренний адрес столбца в цикле доступа (Read, Write)	Физический вывод SDRAM
A0	9	0	A0
A1	10	1	A1
A2	11	2	A2

Физический вывод процессора	Внутренний адрес строки в цикле активации банка (ACT)	Внутренний адрес столбца в цикле доступа (Read, Write)	Физический вывод SDRAM
A3	12	3	A3
A4	13	4	A4
A5	14	5	A5
A6	15	6	A6
A7	16	7	A7
A8	17	8	A8
A9	18	9	A9
A10	несущественный	несущественный	NC
SDA10	19	нулевой	A10/AP
A11	20	20	A11 или банк
A12	21	21	A12 или банк
A13	22	22	A13 или банк
A14	23	23	A14 или банк
A15	24	24	A15 или банк

Таблица 104 – Размер страницы 1 K столбцов, 32-разрядная шина

Физический вывод процессора	Внутренний адрес строки в цикле активации банка (ACT)	Внутренний адрес столбца в цикле доступа (Read, Write)	Физический вывод SDRAM
A0	10	0	A0
A1	11	1	A1
A2	12	2	A2
A3	13	3	A3
A4	14	4	A4
A5	15	5	A5
A6	16	6	A6
A7	17	7	A7
A8	18	8	A8
A9	19	9	A9
A10	несущественный	несущественный	NC
SDA10	20	нулевой	A10/AP
A11	21	21	A11 или банк
A12	22	22	A12 или банк
A13	23	23	A13 или банк
A14	24	24	A14 или банк
A15	25	25	A15 или банк

Программирование параметров SDRAM

Микросхемы SDRAM могут быть получены от разных поставщиков и в зависимости от поставщика, микросхемы могут иметь различные требования к

последовательности включения и временным параметрам. С целью возможности работы с большим количеством поставщиков, в процессоре предусмотрен регистр SDRCON в котором можно запрограммировать некоторые параметры SDRAM. Описание разрядов регистра приведено в таблице 105.

Таблица 105 – Регистр SDRCON

Бит	Имя	Назначение
0	SDREN	Включение контроллера SDRAM: 1 – включен; 0 – выключен
2:1	CAS	Задержка CAS: 00 – один цикл; 01 – два цикла; 10 – три цикла; 11 – резерв
3	PIPE	Дополнительный конвейер: 1 – используется; 0 – нет
5:4	PAGE	Размер страницы (количество столбцов в строке микросхемы SDRAM): 00 – 256 столбцов; 01 – 512 столбцов; 10 – 1024 столбца; 11 – резерв
8:6	-	Зарезервировано
10:9	PRC2RAS	Задержка от подзаряда до строба RAS (в тактах SCLK): 00 – 2; 01 – 3; 10 – 4; 11 – 5
13:11	RAS2PRC	Задержка от строба RAS до команды подзаряда (в тактах SCLK): 000 – 2; 001 – 3; 010 – 4; 011 – 5; 100 – 6; 101 – 7; 110 – 8; 111 – 9
14	INIT	Выбор последовательности инициализации: 1 – команда MRS после регенерации памяти; 0 – команда MRS перед регенерацией памяти
15	EMREN	Использование дополнительного регистра режима: 1 – разрешено; 0 – не используется
31:16	-	Всегда 0

Регистр SDRCON хранит информацию о конфигурации интерфейса SDRAM. Ниже подробно описывается назначение каждого из параметров регистра.

Включение контроллера SDRAM

Разряд SDREN должен быть установлен, если в системе есть SDRAM. В противном случае, он должен быть сброшен. Любой доступ к SDRAM, пока этот разряд сброшен, вызывает аппаратное прерывание ошибки доступа. Установка данного бита включает в работу контроллер SDRAM внешнего интерфейса. При включении контроллер сразу же начинает процедуру инициализации внешней динамической памяти

Выбор значения задержки CAS Latency

Значение задержки CAS определяет задержку в тактовых циклах системы (SCLK), между временем, когда SDRAM обнаруживает команду чтения и временем, когда он выдает данные на его внешние выводы. Этот параметр позволяет процессору знать, в каком такте после передачи в SDRAM команды чтения он может принять прочитанные данные.

Задержка CAS не используется в циклах записи.

Опция буферизации SDRAM. Глубина конвейера

Ранее были описаны ситуации, при которых пользователю может понадобиться использовать буферизацию адресных и управляющих сигналов. Связано это с максимальной нагрузочной способностью выводов процессора и требуемой скоростью обмена. В случае использование буферизации в конвейере чтения данных появляется дополнительный цикл. Чтобы данный цикл был учтен контроллером SDRAM и используется бит PIPE. Шина данных не буферизируется. Поэтому при установленном разряде PIPE (1), контроллер SDRAM задерживает данные в цикле записи в течение одного такта.

В случае чтения контроллер SDRAM осуществляет захват данных на один цикл позже.

Ниже приведен пример одного процессора, в котором интерфейс SDRAM соединяется с множеством банков SDRAM (рисунок 65). Микросхема SDRAM имеет шину данных шириной четыре бита. Для обеспечения 32-разрядной шины данных используется восемь микросхем. Чтобы уменьшить нагрузку на внешние выводы процессора используется буферный регистр, который состоит из двух регистров A и B, каждый из которых запоминает одинаковую информацию, но передает ее на свои четыре микросхемы памяти

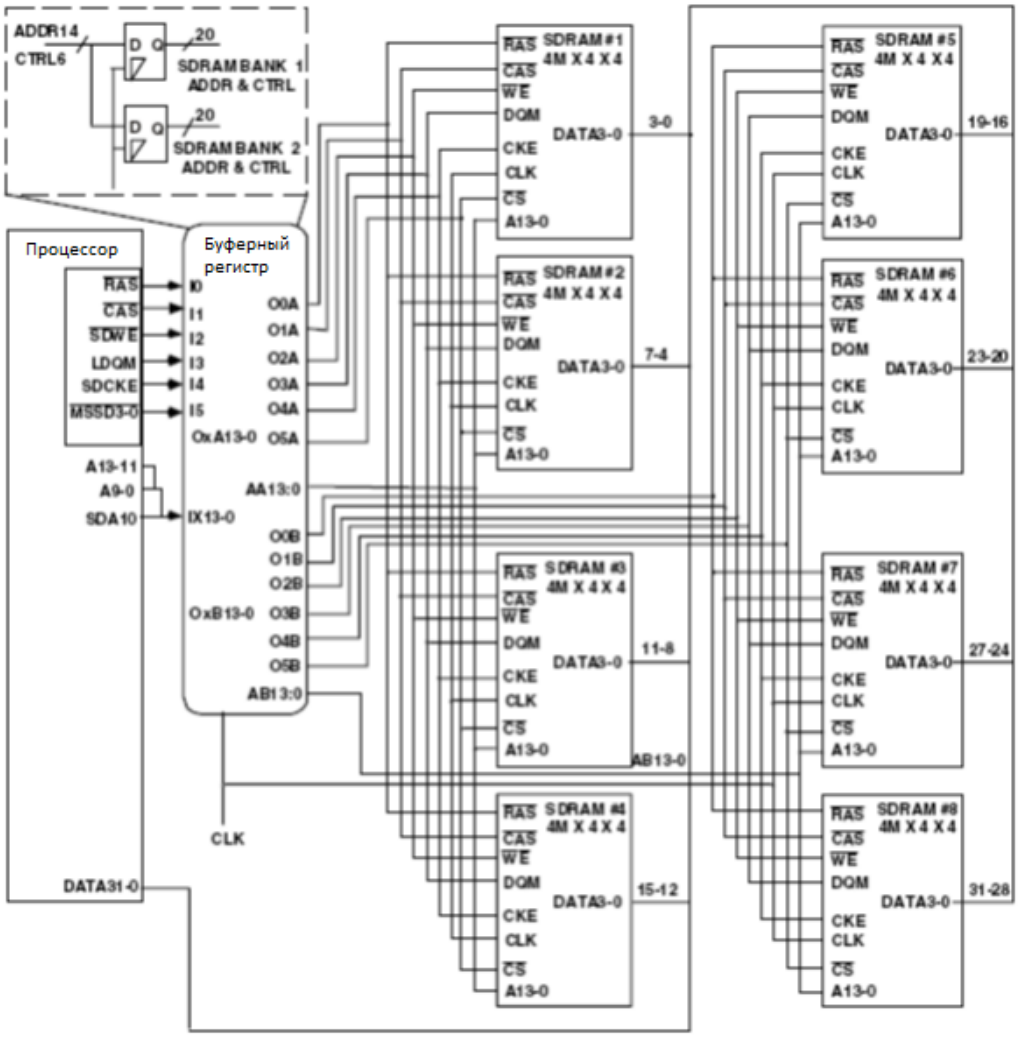


Рисунок 65 – Однопроцессорная система со стандартными устройствами SDRAM (32-разрядная шина)

Выбор размера страницы SDRAM

Размер страницы определяется конкретным типом микросхемы SDRAM. Разряды PAGE определяют размер страницы одной микросхемы SDRAM, а именно количество столбцов в строке памяти. Общее количество 32-разрядных слов в одной странице, сформированное всеми подключенными микросхемами памяти SDRAM, составляет SDRCON.PAGE (столбцов) • SYSCON.MEMWIDTH (бит) / 32. Знание размера страницы необходимо контроллеру SDRAM для отслеживания ситуаций обращения к одной и той же странице. Это позволяет существенно увеличить скорость обмена. Также размер страницы влияет на распределение разрядов внутреннего адреса между адресами строки, столбца и выбора банка памяти.

Период регенерации памяти

Память SDRAM требует, чтобы в течение определенного интервала времени все ее ячейки были обновлены (регенерированы). Это гарантирует сохранность информации. В связи с этим контроллер SDRAM должен периодически посылать в микросхемы памяти команды регенерации. Период подачи команд программируется. Период внутри контроллера задается **в тактах клока внешней шины SCLK** в регистре SD_REF. По сбросу отдельный счетчик обнуляется. Каждый раз при достижении нулевого значения

счетчик выставляет запрос контроллеру SDRAM на команду регенерации памяти, загружает значение, записанное в регистре SD_REF и декрементирует свое значение каждый фронт синхросигнала внешней шины.

Задержка от подзаряда до строба RAS

Протокол обмена с динамической памятью требует, чтобы при доступе к новому ряду банка или перед выполнением регенерации памяти, выполнялся подзаряд (закрытие) текущей активной строки. Для подзаряда используется команда **PRE**. После выполнения подзаряда невозможно сразу перейти к активизации следующего ряда, т.к. спецификация микросхемы требует, чтобы между подзарядом и активизацией нового ряда была обеспечена некоторая минимальная задержка. Значение этой задержки, выраженное числом системных тактовых циклов (SCLK), и программируется в поле PRC2RAS.

Выбор задержки между RAS и предварительным зарядом

Аналогично предыдущему параметру, существует требование минимального интервала времени между подачей команды RAS (активизация ряда банка) и подачей команды PRE подзаряда банка. Данный интервал задается числом системных тактовых циклов SCLK. Значение поля RAS2PRC может изменяться от нуля до шести, это соответствует числу циклов SCLK от двух до восьми. Если, например, поле RAS2PRC равно 011 это означает, что минимальный интервал равен пяти циклам. Это не означает, что после команды RAS минимум через пять тактов должна быть команда PRE. Это означает, что если процессор выполнил команду RAS, а затем по каким-то причинам (например, регенерация) ему необходимо выполнить подзаряд, процессор должен проверить прошло ли с момента команды RAS как минимум пять тактов. Тогда он может подавать команду подзаряда.

Выбор последовательности инициализации SDRAM

Для того чтобы микросхема SDRAM корректно работала, она должна быть предварительно проинициализирована. Процессор обеспечивает две наиболее часто используемые последовательности инициализации.

Бит последовательности инициализации INIT в регистре SDRCON выбирает режим включения SDRAM. Когда бит установлен в «1», контроллер SDRAM последовательно генерирует: команду PRE, восемь циклов регенерации и команду MRS (установка регистра режима). Когда бит INIT сброшен в «0», контроллер SDRAM работает в следующем порядке: команда PRE, команда MRS и восемь циклов регенерации.

Разрешение регистра расширенного режима (EMR)

Бит EMREN в регистре SDRCON разрешает программирование регистра расширенного режима. Когда бит установлен в «1», контролер SDRAM формирует цикл EMRS, предшествующий команде MRS. Когда бит сброшен в «0», последовательность инициализации происходит так, как описано в главе выше.

Бит разрешающий EMR должен быть установлен только во время обмена данными с устройствами SDRAM пониженной мощности (2,5 В). Именно в таких микросхемах используется дополнительный регистр режима.

Включение после сброса

После сброса содержимое регистра SDRCON равно нулю и контроллер SDRAM выключен. Как только приложение пользователя выполняет запись в регистр SDRCON данных с установленным битом включения контроллера SDRAM, контроллер тут же инициирует выбранную последовательность включения динамической памяти. Последовательность инициализации определяется разрядом INIT и разрядом регистра расширенного режима в регистре SDRCON. Программный сброс не вызывает сброса контроллера SDRAM и не инициирует повторно последовательность включения.

13.2 Команды контроллера SDRAM

Этот раздел описывает каждую команду, которую контроллер SDRAM использует для управления интерфейсом SDRAM. Ниже представлен обзор различных команд, используемых встроенным контроллером:

– MRS

(установка регистра режима). Инициализирует рабочие параметры SDRAM во время последовательности включения.

– PRE

(подзаряд). Осуществляет подзаряд банка.

– ACT

(активация банка). Активирует страницу в запрашиваемом банке.

– Read

Чтение из SDRAM.

– Write

Запись в SDRAM.

– REF

(регенерация). Переводит SDRAM в режим регенерации.

– SREF

(саморегенерация). Помещает SDRAM в режим саморегенерации, в котором память управляет своими операциями регенерации изнутри.

– NOP

(нет операций). Передается после чтения или записи с целью разрешения пакетных операций или с целью установления режима ожидания для различных доступов SDRAM. Во время этой команды MSSD3–0 деактивирован.

13.2.1 Команда установки регистра режима (MRS)

Установка регистра режима является частью последовательности инициализации SDRAM. Каждая микросхема SDRAM имеет внутри регистр режима (количество регистров режима может достигать четырех) и команда MRS позволяет передать данные,

используя разряды адреса ADDR13–0, в микросхему для записи в регистр. Таким образом, MRS инициализирует рабочие параметры SDRAM.

Регистр режима (регистр номер 0) имеет разрядность 13 бит и команда MRS инициализирует следующие параметры:

– Биты 2:0 – Burst length. Значение 0b111, что соответствует полной странице.

– Бит 3 – Burst type. Значение 0, что соответствует последовательному изменению адреса.

– Биты 6:4 – CAS Latency. Значение параметра определяется в соответствии с программированием регистра SDRCON[2:1].

Все оставшиеся биты регистра режима программируются нулевым значением.

При выполнении команды MRS, SDRAM должна находиться в состоянии подзаряда во всех своих банках.

Передача команды осуществляется соответствующей комбинацией значений на линиях управления. Эти значения приведены в таблице 106. Происходит запись сразу во все микросхемы памяти.

Таблица 106 – Состояние выводов во время выполнения команды MRS

Разряд	Состояние
MSSD3–0	0 (все)
CAS	0
RAS	0
SDWE	0
SDCKE	1
A14 == A30	0
A15 == A31	0 – MRS 1 – EMRS

Если в микросхеме памяти имеется дополнительный регистр режима и для инициализации требуется его запись, в регистре SDRCON должны быть сделаны соответствующие установки (бит EMREN должен быть равен tlbybwt). Во время последовательности инициализации контроллер SDRAM запишет нулевое значение в дополнительный регистр режима. Выполняется это с помощью той же команды MRS, но бит 15 адреса (и бит 31) будет установлен в «1», что соответствует дополнительному регистру режима. С помощью адресных линий 14 и 15 можно задать номер дополнительного регистра режима как 01 или 10.

13.2.2 Команда подзаряда (PRE)

Особенность работы микросхемы памяти SDRAM состоит в том, что в некоторых ситуациях необходимо выполнять подзаряд (предварительный заряд) внутренних шин банков памяти. Команда PRE (таблица 107) исполняется в двух ситуациях:

– Прерывание цикла чтения или записи. Точный момент, когда данная команда генерируется контроллером SDRAM, зависит от последовательности выполняемых транзакций и адресов, по которым выполняется доступ.

– Предварительная зарядка активного банка. При выполнении команды на линии A10 всегда высокий уровень, что соответствует подзаряду всех банков микросхемы. Однако подзаряжается только активный банк. Алгоритм работы контроллера SDRAM такой, что он поддерживает активным только один банк.

Передача команды PRE возможна в следующих случаях:

– Во время последовательной инициализации SDRAM.

– Перед командой ACT (доступ к новой странице). Доступ к новой странице требует, чтобы активная страница была закрыта. Это делается путем выполнения команды подзаряда.

– Перед циклом регенерации. Выполнение команды регенерации требует, чтобы все банки памяти были подзаряжены.

– Перед тем, как процессор освобождает внешнюю шину. Это гарантирует однозначные условия начала работы нового мастера шины.

Передача команды PRE в микросхему памяти требует, чтобы в течение определенного времени к микросхеме не посылались команды активации. Данное время программируется в регистре SDRCON в соответствии с характеристиками микросхемы памяти.

Таблица 107 – Состояние выводов во время выполнения команды PRE

Вывод	Состояние
MSSD3–0	0 (все)
SDWE	0
RAS	0
CAS	1
SDCKE	1
SDA10	1

13.2.3 Команда выбора активного банка (ACT)

Команда ACT (таблица 108) посылается контроллером перед любым чтением или записью страницы, которая в данный момент не является активной. Перед командой ACT идет команда PRE, если в каком-то банке имеется другая активная страница. Команда ACT открывает доступ к странице SDRAM в некотором банке, и данная страница остается открытой до тех пор, пока не будет закрыта следующей командой предварительной зарядки PRE.

Таблица 108 – Состояние вывода во время выполнения команды ACT

Вывод	Состояние
MSSD3–0	0 (один из четырех)
CAS	1
RAS	0
SDWE	1
SDCKE	1

13.2.4 Команда чтения (Read)

Команда Read (таблица 109) выполняет чтение данных из активной страницы памяти SDRAM. Если чтение страницы выполняется в первый раз после активации командой ACT, между командами ACT и Read должна быть задержка, которая определяется параметром tRCD. Если это уже не первое чтение, отслеживание данной задержки не выполняется. Команда Read задает начальный адрес слова в странице, с которого начинается чтение. Микросхема памяти будет использовать переданный адрес как стартовый и каждый такт увеличивать его, если необходимо выполнить последовательное чтение блока памяти. После подачи команды Read данные появляются на внешних выводах микросхемы через время равное задержке CAS Latency.

Одна транзакция чтения занимает различное количество циклов шины в зависимости от размера операнда и ширины внешней шины:

– чтение байта или короткого слова на 32-разрядной шине – один цикл;
– чтение байта или короткого слова на 16-разрядной шине – два цикла;
– чтение одинарного слова на 32-разрядной шине – один цикл;
– чтение одинарного слова на 16-разрядной шине – два цикла;
– чтение двойного слова на 32-разрядной шине – два цикла;
– чтение двойного слова на 16-разрядной шине – четыре цикла;
– чтение счетверенного слова на 32-разрядной шине – четыре цикла;
– чтение счетверенного слова на 16-разрядной шине – восемь циклов.

Если после команды Read на SDRAM не поступает какая-то другая команда, SDRAM продолжает последовательное чтение данных, наращивая внутренний адрес. Этот режим чтения называется «страничный режим» или «пакет». Такой режим удобен, когда транзакция чтения длится более одного цикла.

Когда транзакция полностью завершена, SDRAM может продолжить работу по нескольким путям:

– если есть следующая транзакция чтения SDRAM на той же странице, новая транзакция начинается с подачи команды Read;

– если нет новых транзакций SDRAM, контроллер посылает команду BSTOP. Эта команда прерывает последовательное чтение страницы;

– если после чтения приходит транзакция записи SDRAM на той же странице, команда BSTOP также останавливает чтение и запись начинается после приема данных;

– если есть доступ SDRAM к другой странице, поступает команда BSTOP и после нее посылается команда закрытия активной страницы PRE.

Таблица 109 – Состояние вывода во время выполнения команды Чтение

Вывод	Состояние
MSSD3–0	0 (один из четырех)
CAS	0
RAS	1
SDWE	1
SDCKE	1

Временная диаграмма использования команды Read для случая ширины шины 32 бита приведена на рисунке 66. Задержка CAS Latency равна двум.

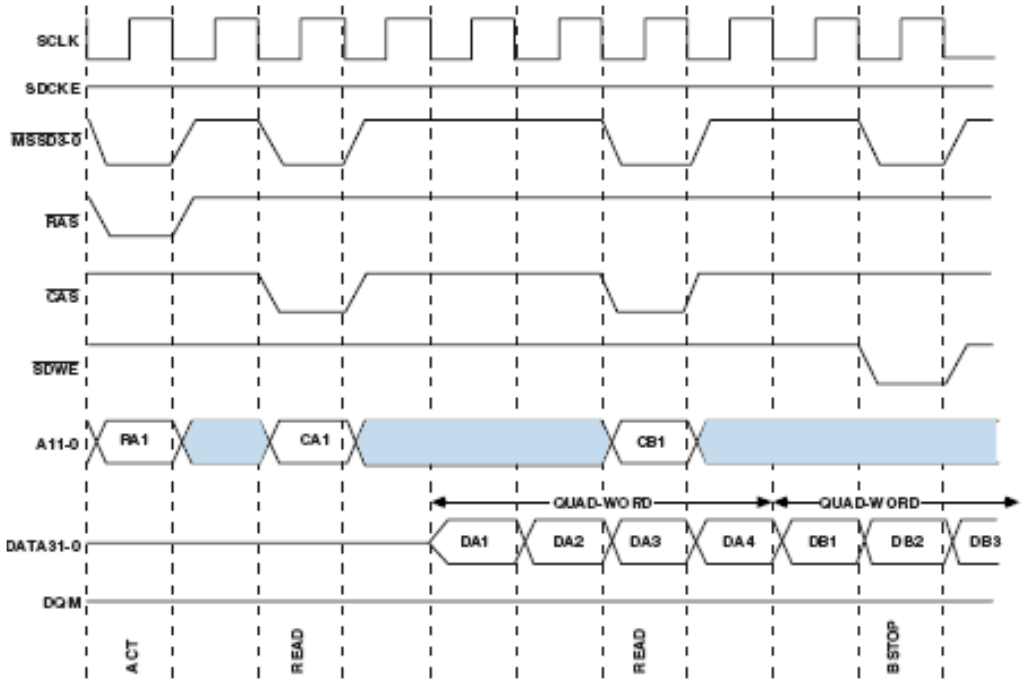


Рисунок 66 – Шина 32 разряда
(пакетное чтение с последующим пакетным чтением на той же странице)

13.2.5 Команда записи (write)

Команда Write (таблица 110) выполняет запись данных в активную страницу. Если это первая запись после активизации страницы, перед подачей команды записи должен быть соблюден интервал времени tRCD. Команда записи устанавливает начальный адрес слова страницы, с которого начнется запись. Как и в случае чтения, если после записи нет других команд, на следующем такте будет выполняться запись по последовательно увеличенному адресу. Если запись нужно остановить, контроллер SDRAM должен передать команду BSTOP.

Одна транзакция записи занимает разное количество циклов в зависимости от размера операнда транзакции и ширины внешней шины:

– запись байта или короткого слова на 32-разрядной шине – два цикла (одно чтение и одна запись);

– запись байта или короткого слова на 16-разрядной шине – четыре цикла (два чтения и две записи);

– запись одного слова на 32-разрядной шине – один цикл;

– запись одного слова на 16-разрядной шине – два цикла;

– запись двойного слова на 16-разрядной шине – четыре цикла;

– запись двойного слова на 32-разрядной шине – два цикла;

– запись счетверенного слова на 16-разрядной шине – восемь циклов;

– запись счетверенного слова на 32-разрядной шине – четыре цикла.

Одновременно с подачей начального адреса, процессор выставляет на шину данных и записываемые данные. Последовательная запись блока данных называется

«страничный режим» или «пакет». При этом в последующих циклах нет необходимости подавать адрес, достаточно только передавать следующие данные.

Когда транзакция записи завершена, контроллер SDRAM может продолжить работу по нескольким путям:

– если нет новых транзакций SDRAM, поступает команда BSTOP, которая указывает на завершение записи блока данных;

– если следующая транзакция есть тоже запись в эту же страницу, контроллер может подавать новую команду записи с новым начальным адресом и данными;

– если следующая транзакция – это чтение SDRAM на той же странице, команда BSTOP останавливает транзакцию записи и транзакция чтения начинается с нового цикла;

– если есть доступ SDRAM к другой странице, поступает команда BSTOP и далее подаются команды закрыть текущую страницу (PRE) и активизировать следующую. При этом соблюдаются все необходимые задержки, запрограммированные в регистре конфигурации.

Таблица 110 – Состояние вывода во время выполнения команды Запись

Вывод	Состояние
MSSD3–0	0 (один из четырех)
CAS	0
RAS	1
SDWE	0
SDCKE	1

Временная диаграмма использования команды Write для случая ширины шины 32 бита приведена на рисунке 67. Задержка CAS Latency равна двум.

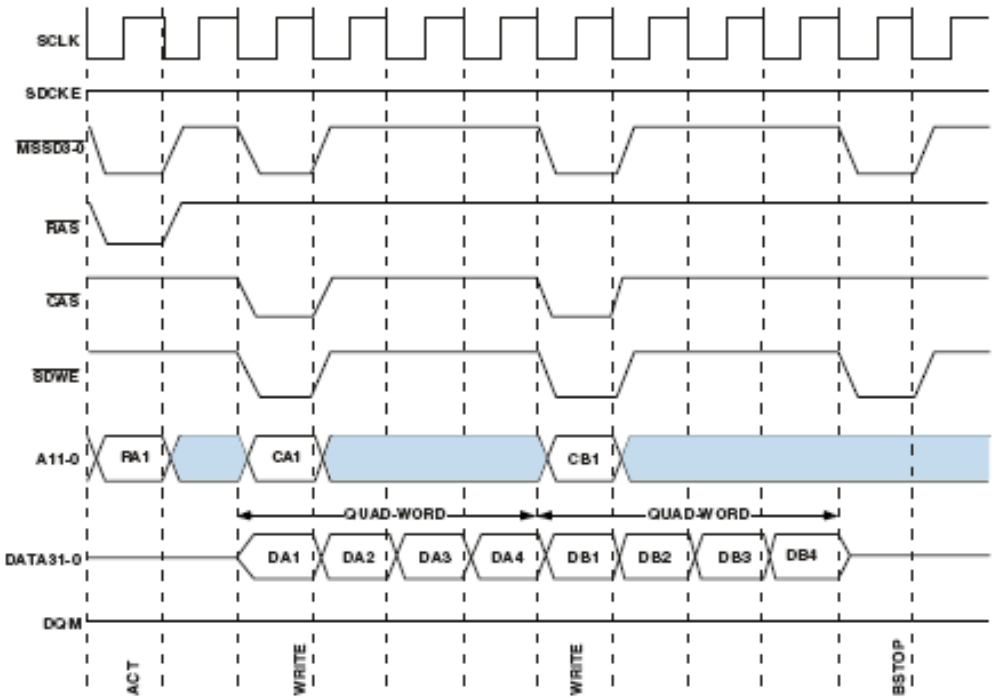


Рисунок 67 – Шина 32 разряда
(пакетная запись с последующей пакетной записью на той же странице)

13.2.6 Команда регенерации (REF)

Память SDRAM требует периодической регенерации своих ячеек (строк) для сохранности информации. Команда REF (таблица 111) является запросом к SDRAM для выполнения транзакции регенерации. Эта команда генерируется автоматически процессором и вызывает регенерацию данных по адресу, который находится внутри SDRAM. Перед выполнением команды REF контроллер SDRAM выполняет проверку наличия активных страниц и, если они имеются – выполняет команду предварительной зарядки (PRE) для активного банка.

Следующая команда активации (ACT) выдается контроллером только после минимальной задержки, равной t_{RC}, где:

$$t_{RC} = t_{RP} + t_{RAS} + 1\,. \qquad (3)$$

Частота регенерации (период подачи команд регенерации) устанавливается в регистре SD_REF.

Таблица 111 – Состояние выводов во время выполнения команды REF

Вывод	Состояние
MSSD3–0	0(все)
CAS	0
RAS	0
SDWE	1
SDCKE	1

13.2.7 Команда саморегенерации (SREF)

Описанная выше команда регенерации REF требует от процессора непрерывно отслеживать период повторения циклов регенерации и своевременно посылать команды REF. Эта команда может выполняться на фоне выполнения транзакций процессором к другим типам памяти (не SDRAM). Также существует режим, когда память отвечает за процедуру регенерации и самостоятельно поддерживает данные достоверными. Этот режим называется режимом саморегенерации и процессор переводит SDRAM в этот режим путем подачи команды SREF (таблица 112). В данном режиме память недоступна для использования.

При выполнении команды саморегенерации должны быть выполнены все те же временные требования, что и при выполнении команды регенерации.

Перед выполнением команды SREF контроллер подзаряжает все банки микросхем. После команды SREF память выполняет операции регенерации самостоятельно без внешнего контроля. После выхода из режима саморегенерации контроллер SDRAM ждет количество циклов tRC перед выполнением команды активизации банка (ACT).

Память переводится в режим саморегенерации установкой нулевого бита регистра SDRSRF в единичное значение. Данный режим может быть использован как

составная часть режима пониженного энергопотребления системы. Для выхода из режима саморегенерации необходимо очистить нулевой бит регистра SDRSRF.

Таблица 112 – Состояние вывода во время выполнения команды SREF

Вывод	Состояние
MSSD3–0	0 (все)
CAS	0
RAS	0
SDWE	1
SDCKE	0

Режим саморегенерации поддерживается тем, что на линии SDCKE постоянно низкий уровень, что запрещает использование входа синхронизации.

13.3 Вспомогательные регистры интерфейса внешней шины

В микросхемах К1967ВН044 функции данных регистров ограничены.

13.3.1 Регистр управления блокировкой шины (BUSLOCK)

Регистр BUSLOCK определяет состояние запроса блокировки шины. В этом 32-разрядном регистре определен только один нулевой бит. Все другие биты (с 1 по 31) не используются. Запись значения «1» в нулевой бит вызывает запрос на блокировку шины со стороны процессора. В микросхемах К1967ВН044 внешний интерфейс всегда является мастером шины, и нет необходимости в регистре блокировки.

13.3.2 Регистр статуса системы (SYSTAT)

Регистр SYSTAT предназначен только для чтения и указывает на состояние некоторых параметров системы.

Таблица 113 – Регистр SYSTAT

Бит	Имя	Назначение
9:0	-	Всегда 0
10		Всегда 1
11	-	Всегда 0
12	-	Бит 0 регистра CFG1
13	MRSCOMP	Признак окончания процедуры инициализации динамической памяти: 1 – завершена. Память готова к использованию; 0 – нет
14	BUSLKACT	Признак захвата шины: 1 – текущий процессор стал мастером на шине и удерживает ее; 0 – нет захвата шины
15	-	Всегда 0

Бит	Имя	Назначение
16	BROADREADERR	Ошибка доступа к адресному пространству 0x0C…: 1 – была попытка чтения из указанного пространства; 0 – нет ошибки
17	AUTODMAERR	Признак ошибки в работе каналов AutoDMA: 1 – была попытка записи в выключенный канал; 0 – нет ошибки
18	SDRAMERR	Признак ошибки доступа к SDRAM: 1 – была попытка доступа неинициализированной внешней динамической памяти; 0 – нет ошибки
19	MPREADERR	Признак ошибки чтения процессором запрещенного адресного пространства: 1 – ошибка; 0 – нет ошибки
31:20	-	Всегда 0

Программы могут считывать регистр SYSTAT, используя имена SYSTAT или SYSTATCL (сброс SYSTAT):

- SYSTAT – чтение без изменений в содержимом регистра;
- SYSTATCL – разряды 19–16 будут сброшены после чтения.

14 Последовательный хост-интерфейс

Процессор содержит синхронный последовательный интерфейс, который может быть использован для загрузки информации в процессор внешним ведущим устройством, а также для анализа состояния процессора посредством чтения его ресурсов.

Интерфейс подключен к периферийной шине и имеет базовый адрес 0x8000_005C.

Внешние выводы хост-интерфейса приведены в таблице 114.

Таблица 114 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода для Хост-интерфейса	Тип вывода	Функциональное назначение
PC[4]	HCLK	I/O	Хост интерфейс. Синхросигнал
PC[5]	HDI	I/O	Хост интерфейс. Вход данных
PC[6]	HDO	I/O	Хост интерфейс. Выход данных

14.1 Регистры интерфейса

Краткое описание регистров интерфейса приведено в таблице 115.

Таблица 115 – Регистры хост-интерфейса

Номер\ смещение	Обозначение регистра	R/W	Описание	Состояние по сбросу
0x00	-	R	Неопределенное значение	0
0x01	-	R	Неопределенное значение	0
0x02	MCR	R	Управление запросом прерываний к процессору	0
0x03	-	R	Неопределенное значение	0

Единственный доступный для процессора регистр хост-интерфейса MCR можно прочитать только посредством чтения квадрослова по адресу 0x8000_005C. Регистр может использоваться как один из возможных способов передачи информации от хоста к процессору.

14.2 Аппаратная реализация интерфейса

Для организации обмена синхронный интерфейс использует три линии: линию синхронизации HCLK и линии данных HDI, HDO. Эти линии функционально совмещены с линиями задания конфигурации BOOT[2:0].

Во время сброса системы линии BOOT выполняют функцию задания варианта начального старта системы, а после завершения сброса они могут быть использованы для управления интерфейсом. Для реализации интерфейса необходимо использовать шесть линий:

- питание;
- общий;
- сброс nRESET;

– BOOT.0/HCLK;
– BOOT.1/HDI;
– BOOT.2/HDO.

Начальная последовательность действий ведущего при этом будет следующей:

– установить на входе nRESET низкий уровень (сброс системы) и задать на входах BOOT[2:0] необходимую начальную конфигурацию;

– после некоторой задержки, установить на nRESET высокий уровень, сохраняя на входах BOOT[2:0] старое значение (обеспечивая время удержания);

– передать контроль над выводами BOOT[2:0] ведущему устройству.

14.2.1 Протокол обмена по последовательному интерфейсу

Информация принимается каналом по фронту (переход из «0» в «1») синхросигнала HCLK. Выдача информации осуществляется также по фронту.

Обмен всегда осуществляется 32-разрядными словами. Одна посылка состоит, как минимум из 34 бит: старт бит («0»), 32 бита данных (старший бит первый) и стоп-бит (значение «1»). Примеры обмена данными приведены на рисунках 68 и 69.

После сброса процессора хост-интерфейс некоторое время недоступен. Ведущее устройство после сброса процессора должно посылать сигнал синхронизации и значение «1» на входе данных HDI. Как только интерфейс включается (на него перестает действовать внутренний сброс процессора), на выходной линии данных появится активное значение «0» и через несколько тактов синхронизации – значение «1». После этого интерфейс готов к работе.

Посредством интерфейса возможно выполнение операций чтения и записи, а также внутренних операций.

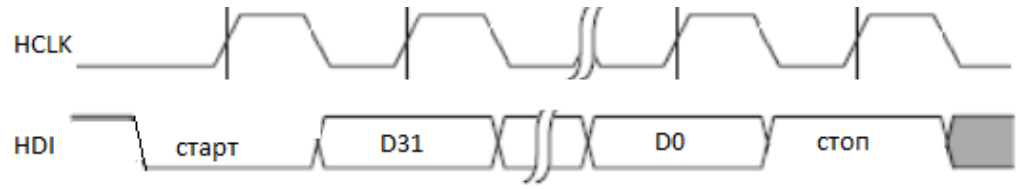


Рисунок 68 – Прием информации контроллером интерфейса

Рисунок 69 – Передача информации контроллером интерфейса

14.2.2 Стандартный формат обмена

Имеется два формата команд: стандартный и короткий. В стандартном формате начальный адрес обмена передается отдельным словом, а в коротком он упакован в управляющее слово. Для выполнения операции записи в стандартном формате выполняется следующая процедура обмена:

1 передается управляющее слово;

2 передается начальный адрес;

3 передаются данные.

Структура управляющего слова приведена в таблице 116.

Таблица 116 – Структура управляющего слова стандартного формата

Номер	Название	Описание
31:28	TYPE	Задают тип операции: 00x0 – чтение; 01x0 – запись; 10xx – внутренняя операция; 11xx – резерв
27:20	DDP	Поле управления обменом
19:17	-	
16	MDF	1 – разрешение модификации адреса после каждого обмена
15:0	CNT	Количество 32-разрядных слов данных при обмене

Для выполнения операции чтения стандартного формата выполняется следующая процедура обмена:

1 передается управляющее слово;

2 передается начальный адрес;

3 принимаются данные.

При выполнении внутренней операции имеется возможность загрузить разряды 29:0 управляющего слова в специальный регистр управления интерфейсом MCR. В настоящее время используется только бит 0 регистра управления. При записи в него значения «1» происходит генерация запроса прерывания к процессору, т.е. имеется возможность послать запрос на прерывания, не выполняя процедуры обмена данными.

14.2.3 Короткий формат обмена

Для выполнения операции записи в коротком формате выполняется следующая процедура обмена:

1 передается управляющее слово;

2 передается слово данных.

Структура управляющего слова приведена в таблице 117.

Таблица 117 – Структура управляющего слова короткого формата

Номер	Название	Описание
31:28	TYPE	Задают тип операции: 00x1 – чтение; 01x1 – запись; 10xx – внутренняя операция; 11xx – резерв
27:20	DDP	Поле управления обменом
19	A[31]	Бит 31 адреса. Используется для доступа к регистрам периферии

Номер	Название	Описание
18:16	A[20:18]	Биты адреса 20:18 при обращении к внутренней памяти. Все другие биты равны нулю
15:0	A[15:0]	Младшие биты адреса

При использовании короткого формата возможна запись или чтение только одного слова (в соответствии с размером) данных. Адрес не изменяется. Имеется возможность доступа ко всем ресурсам процессора.

Для выполнения операции чтения в коротком формате выполняется следующая процедура обмена:

1 передается управляющее слово;

2 принимаются данные.

При использовании короткого формата нет прямого указания на количество передаваемых слов данных. Количество передаваемых 32-разрядных слов определяется значением поля LEN (см. таблицу 118). Для двойного слова это два слова, а для квадрослова это четыре слова.

14.2.4 Поле управления обменом DDP

Биты TYPE управляющей посылки задают только общую информацию о типе обмена. Если выполняется чтение или запись данных, то дополнительная информация о параметрах обмена кодируется в поле DDP.

Таблица 118 – Структура поля управления обменом DDP

Номер	Название	Описание
7:5	TY	Выбор источника либо приемника: 000 – нет операции (выключено); 001 – запрещено; 010 – внутренняя память; 011 – регистры процессорного ядра; 100 – внешняя память и регистры периферии; 101 – запрещено; 110 – внешний EPROM; 111 – запрещено
4	PR	Приоритет обмена: 1 – высокий; 0 – обычный
3	-	Резерв. Всегда должен быть равен 0
2:1	LEN	Длина передаваемых данных в одном цикле обмена: 00 – резерв; 01 – слово 32 бита; 10 – длинное слово 64 бита; 11 – квадрослово 128 бит
0	INT	Генерация запроса прерывания после окончания работы канала: 1 – разрешено; 0 – запрещено

Если формат обмена стандартный, то принимается следующее слово, которое рассматривается как начальный адрес источника либо приемника. Если формат обмена короткий, то начальный адрес формируется как {A[31], 10'b0, A[20:18], 2'b00, A[15:0]}. Отметим, что биты TY поля DDP имеют более высокий приоритет при декодировании адреса. Если, например, адрес имеет значение 0xA5A5, а биты TY равны 100, то будет выполнено обращение к внешней памяти по адресу 0xA5A5.

Бит MDF стандартной посылки используется для указания модификации адреса после выполнения одного цикла обмена. Если этот бит равен нулю, то данные все время будут читаться или записываться по одному и тому же адресу. Если бит MDF равен единице, то после чтения или записи одной порции данных (размер порции определяется битами LEN), будет выполнена модификация адреса (увеличение адреса на значение один, два или четыре в соответствии со значением поля LEN).

Количество 32-разрядных слов обмена определяется значением поля CNT стандартной посылки, либо значением поле LEN короткой посылки. После каждой передачи порции данных, значение внутреннего счетчика слов уменьшается от начального (значение CNT) на величину 1, 2 или 4 в соответствии со значением поля LEN.

Бит PR задает приоритет интерфейса при арбитраже в сравнении с каналами контроллера DMA и процессором. Желательно использовать значение PR равное единице, т.к. в случае низкого приоритета хост интерфейс может быть приостановлен работой более высокоприоритетных каналов контроллера DMA. Особенно это важно во время записи блока данных во внешнюю память.

Бит запроса прерывания INT позволяет сгенерировать запрос прерывания к процессору после окончания обмена данными. Хост-интерфейс имеет соответствующий бит запроса прерываний в контроллере прерываний (бит 6 регистра ILATH, IMASKH), а также регистр вектора прерываний IVHOST.

Обращаем внимание на разрешенные комбинации в разрядах TY. Использование запрещенных комбинаций приведет к непредсказуемому поведению интерфейса.

Также, как и при программировании каналов контроллера DMA, необходимо соблюдать правила выравнивания адресов при работе с длинными словами и квадрословами. Значение количества данных CNT должно быть кратно двум при работе с длинными словами и кратно четырем при работе с квадрословами.

14.2.5 Внутренняя операция интерфейса

При выполнении внутренней операции содержимое принятого управляющего слова (биты с нулевого по 29-й) записываются в регистр MCR. Нулевой бит регистра MCR может формировать запрос прерывания к процессору. Данное прерывание анализируется в контроллере прерываний как запрос номер 6 регистра ILATH. Процессор может прочитать значение регистра MCR. Чтение выполняется посредством чтения квадрослова по адресу 0x8000_005C и последующем использовании второго слова прочитанного квадрослова. Отметим, что запрос прерывания от хост-интерфейса к процессору можно также сформировать и при выполнении операций чтения либо записи. Особенно это удобно делать при выполнении операции записи. В этом случае процессору

может быть передана информация и после этого сформирован запрос прерывания. Такой способ позволяет передать процессору больше информации, чем через регистр MCR.

14.2.6 Вход синхронизации HCLK

Вход синхронизации используется для задания тактового импульса обмена по последовательному интерфейсу. Прием и выдача информации осуществляется по фронту HCLK. Отметим, что вход HCLK во время сброса выполняет функцию задания варианта начального старта (BOOT[0]). Также этот вывод является входом-выходом порта общего назначения PC[4]. Во время сброса требуемый уровень на входе HCLK должен задаваться резистором (либо внешним ведущим устройством). После сброса вход HCLK может управляться ведущим устройством (при его наличии), либо иметь постоянное значение. Дело в том, что любое переключение на выводе PC[4] рассматривается интерфейсом как подача частоты HCLK. Поэтому данный вывод не может произвольно изменяться. Если пользователь процессора не предусматривает подключения внешнего ведущего устройства к хост-интерфейсу, т.е. хост-интерфейс не планируется использовать, то желательно заблокировать вход HCLK посредством установки бита H_OFF (бит 13) в регистре CFG1 (см. раздел 30.5 «Схема подключения отладчика JEM-LYNX»). После этого вывод HCLK(PC[4]) может использоваться без ограничений для реализации других функций.

14.2.7 Вход данных HDI

Вход используется для передачи процессору управляющих команд и данных. Во время сброса вход выполняет функцию задания варианта начального старта(BOOT[1]). Также этот вывод является входом-выходом порта общего назначения PC (бит 5). Во время сброса требуемый уровень на входе HDI должен задаваться резистором (либо внешним ведущим устройством). После сброса вход HDI может управляться ведущим устройством (при его наличии), либо выполнять функции порта общего назначения PC[5]. При обмене с интерфейсом ведущее устройство всегда посылает 32-разрядные данные с дополнительными старт и стоп битами. Если нет необходимости передавать информацию по входу HDI, на входе должен удерживаться высокий уровень. При наличии частоты синхронизации HCLK высокий уровень гарантирует отсутствие входных данных. Интерфейс обнаруживает наличие входных данных при принятии старт бита. После этого внутренний счетчик отсчитывает 32 такта данных. Стоп-бит нужен как дополнительный такт синхронизации для выполнения внутренних операций. Значение стоп-бита не контролируется. Однако для приема следующего слова, во входном потоке должна появиться 1 и только после нее будет детектирован старт бит.

14.2.8 Выход данных HDO

Выход данных HDO используется для передачи данных от процессора к ведущему устройству. В момент сброса процессора он выполняет функцию задания варианта начального старта (BOOT[2]), поэтому во время сброса данный вывод находится в

отключенном состоянии (т.е. настроен на прием информации с внешнего контакта). Если на входе HCLK имеются переключения, то они рассматриваются как наличие частоты синхронизации от ведущего устройства. В этом случае данные с входа HDI записываются в приемный сдвиговый регистр. Если после завершения внутреннего сигнала сброса будет принято не менее 32 единичных значений с входа HDI, то интерфейс включится, и выход HDO станет активным выходом. Для задания варианта начального старта на выводе HDO может использоваться резистор, подключенный к шине общий. Таким образом, на выводе HDO будет удерживаться уровень нуля до тех пор, пока интерфейс не включится и не начнет выдавать активную единицу. При чтении внутренней ячейки памяти процессора, внешнее хост-устройство заранее не знает сколько тактов понадобится для считывания значения. Поэтому ведущее устройство после передачи команды чтения должно непрерывно подавать сигналы синхронизации HCLK и удерживать на входе HDI высокий уровень. Как только данные поступят в буфер интерфейса, на выводе HDO появится значение «0» (старт бит), и за ним будут переданы 32 бита данных. После них будет выдан стоп бит. Выдача данных на вывод HDO осуществляется автоматически, как только в буфере появляются данные.

14.2.9 Алгоритм работы внутренней машины состояний

Внутренний автомат интерфейса работает с тактовой частотой SOC-шины (SOCCLK) и постоянно анализирует поступающую информацию. Обработка информации осуществляется в соответствии со следующим алгоритмом:

Состояние S0 Ожидание управляющего слова. Если слово принято – запись слова во внутренний регистр HCR и переход в состояние S1.

Состояние S1 Анализ регистра HCR. Если стандартный формат чтения – переход в состояние S4. Короткий формат чтения – переход в состояние S7. Если стандартный формат записи – переход в состояние S8. Короткий формат записи – переход в состояние S10. Если внутренняя операция – переход в состояние S2. В случае резервной операции – зависание машины состояний (!).

Состояние S2 Выполнение внутренней операции. Перезапись регистра HCR в регистр MCR. Переход в исходное состояние S0.

Состояние S4 Ожидание адреса для выполнения чтения. После приема адреса – запись управляющей информации (начальный адрес, счетчик слов, управляющее слово) в канала обмена. Переход в состояние S6 - ожидания выполнения операции чтения. Если в момент записи в канал обмена последний находится в активном состоянии, устанавливается флаг ошибки и канал выключается.

Состояние S6 Ожидание завершения работы канала. Канал завершит работу только после чтения всего блока данных. При этом моментом окончания работы канала является факт загрузки последнего слова в буфер и начало его передачи в ведущее устройство. После передачи последнего слова данных - переход в исходное состояние S0.

Состояние S7 Запись управляющей информации (начальный адрес, счетчик слов, управляющее слово) в канал обмена. Переход в состояние S6 - ожидания выполнения операции чтения. Если в момент записи в канал обмена последний находится в активном состоянии, устанавливается флаг ошибки и канал выключается.

Состояние S8 Ожидание адреса для выполнения операции записи. После приема адреса – запись управляющей информации (начальный адрес, счетчик слов, управляющее слово) в канал обмена. Переход в состояние S9 - ожидания приема данных для записи. Если в момент записи в канал обмена последний находится в активном состоянии, устанавливается флаг ошибки и канал выключается.

Состояние S9 Ожидание данных для выполнения операции записи. После приема данных – запись принятого слова в буфер канала. Ожидание нового слова данных либо завершения работы канала. Канал инициирует запись данных в процессор только если в буфере данных достаточно данных для запрограммированной записи. После записи последнего слова данных - переход в исходное состояние S0.

Состояние S10 Запись управляющей информации (начальный адрес, счетчик слов, управляющее слово) в канал обмена. Переход в состояние S9 - ожидания данных для выполнения операции записи. Если в момент записи в канал обмена последний находится в активном состоянии, устанавливается флаг ошибки и канал выключается.

Машина состояний интерфейса требует корректного завершения всех операций. У ведущего процессора нет возможности анализировать состояние интерфейса, и в случае различных аномалий возвращать его функционирование в начальное состояние. Только сброс всего процессора является выходом из возможной ошибочной ситуации.

14.2.10 Доступ к регистрам ядра

Выше было отмечено, что в случае, когда биты TY поля DDP равны 011’b, хост устройство может осуществлять доступ к внутренним регистрам вычислительного ядра процессора. Отметим, что это единственный способ получить возможность доступа к регистрам для внешнего устройства. Для доступа к регистрам используются только разряды адреса с нулевого по 10-й. Биты с пятого по 10-й образуют номер группы регистров, а биты с нулевого по четвертый номер регистра в группе. Номера групп при доступе со стороны хост-устройства совпадают с номерами групп, которые используются системой команд процессора для выполнения пересылок. Используемые номера групп приведены в таблице 119.

Таблица 119 – Группы регистров вычислительного ядра

Номер группы	Название группы
0	Регистры общего назначения вычислительного модуля X: XR[0-31]
2	Регистры общего назначения вычислительного модуля Y: YR[0-31]
10	Регистры модуля отладки
12	Регистры общего назначения J модуля IALU : J[0-31]
13	Регистры общего назначения K модуля IALU : K[0-31]
14	Дополнительные регистры J модуля IALU : JB[0-3], JL[0-3]
15	Дополнительные регистры K модуля IALU : KB[0-3], KL[0-3]
26	Регистры устройства управления
27	Регистры модуля отладки
30	Регистры модуля защиты и управления кэш-памятью
31	Регистры модуля защиты и управления кэш-памятью

14.2.11 Ограничения интерфейса

При режиме старта BOOT[2:0]==000 процессор переходит в режим ожидания прерывания. Хост-устройство может выполнить загрузку данных в память процессора и запустить процессор на выполнение программы. Последовательный хост-интерфейс позволяет ведущему устройству осуществлять полный контроль процессора. Передача данных по последовательному интерфейсу выполняется с частотой HCLK, а все внутренние операции процессора в контроллере интерфейса – с частотой SOC-шины. Чтобы не было проблем с переполнением приемного буфера интерфейса, желательно иметь частоту HCLK не выше частоты SOC-шины. Особое внимание необходимо уделить начальным операциям, выполняемым после сброса, т.к. в этом случае процессор работает на входной частоте синхронизации, и она может быть недостаточно высокой. В этом случае можно анализировать параметры частот процессора и менять частоту обмена HCLK. Особое внимание нужно уделять операциям записи блока данных в процессор, т.к. в этом случае внутренние операции должны успевать выполниться до приема новой порции данных.

Операции чтения и записи интерфейса независимы. Интерфейс всегда начинает выдачу данных на выход HDO, если в его буфер поступают данные. Таким образом, выдачу данных можно вести одновременно с приемом новой информации. Единственное ограничение - это совместное использование буфера интерфейса, как для операций чтения, так и для операций записи. Поэтому операция записи может начинаться только после завершения чтения последнего слова данных.

Для корректной работы интерфейса рекомендуется не начинать новой операции до момента полного завершения предыдущей.

Операции чтения области регистров внутренних периферийных устройств запрещены. При выполнении таких операций интерфейс будет заблокирован. См. описание ошибки 0009 в документе «Errata Notice».

15 Порты связи

Процессор имеет два LINK-порта связи для обеспечения 8\4\1-битного приема и 8\4\1-битной передачи в мультипроцессорных системах. Оба LINK-порта могут быть сконфигурированы как один 16-ти битный порт для одновременного приема и передачи данных (полный дуплекс).

Порты связи имеют следующие характеристики:

– тактовая скорость связи равна 1/2 от выходной частоты собственной PLL;

– данные порта связи упакованы в 128-битные слова для DMA передачи во внутреннюю и внешнюю память;

– каждый порт связи имеет собственные буферные регистры;

– передачи каждого порта контролируются подтверждающим протоколом;

– порты связи поддерживают полный дуплекс и передачу в\из внешних портов или других каналов связи.

Порты связи предназначены для использования при двухточечной связи между процессорами в системе, но могут использоваться в качестве интерфейса для коммуникации с любыми другими устройствами, разработанными для работы по тому же протоколу.

Порты связи процессора используют схему LVDS (дифференциальные сигналы низкого напряжения). Данные выдаются и принимаются как на фронте, так и на срезе тактового сигнала. Каждый из портов имеет канал приема и передачи, и способен работать в полностью дуплексном режиме.

Управление передачей ядром процессора выполняется посредством механизмов прерывания и опроса. Управление передачей контроллером DMA выполняется через назначенные каналы DMA приема/передачи. При этом все каналы DMA портов поддерживают цепочки операций, а также могут быть использованы при загрузке процессора во время старта.

Внешние выводы портов связи приведены в таблице 120.

Таблица 120 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода для портов связи	Тип вывода	Функциональное назначение
PC[24]	L0ACKO	O	LINK-порт 0. Выход разрешения передачи
PC[25]	L0ACKI	I	LINK-порт 0. Вход разрешения передачи
PC[26]	L0BCMPO	O	LINK-порт 0. Выход окончания блока
PC[27]	L0BCMPI	I	LINK-порт 0. Вход разрешения передачи
PC[28]	L1ACKO	O	LINK-порт 1. Выход разрешения передачи
PC[29]	L1ACKI	I	LINK-порт 1. Вход разрешения передачи
PC[30]	L1BCMPO	O	LINK-порт 1. Выход окончания блока
PC[31]	L1BCMPI	I	LINK-порт 1. Вход разрешения передачи
L0DATOP[0]	-	O	LINK-порт 0. Выход данных

Обозначение вывода (основное)	Обозначение вывода для портов связи	Тип вывода	Функциональное назначение
L0DATON[0]	-	O	LINK-порт 0. Выход данных
L0DATOP[1]	-	O	LINK-порт 0. Выход данных
L0DATON[1]	-	O	LINK-порт 0. Выход данных
L0DATOP[2]	-	O	LINK-порт 0. Выход данных
L0DATON[2]	-	O	LINK-порт 0. Выход данных
L0DATOP[3]	-	O	LINK-порт 0. Выход данных
L0DATON[3]	-	O	LINK-порт 0. Выход данных
L0DATOP[4]	-	O	LINK-порт 0. Выход данных
L0DATON[4]	-	O	LINK-порт 0. Выход данных
L0DATOP[5]	-	O	LINK-порт 0. Выход данных
L0DATON[5]	-	O	LINK-порт 0. Выход данных
L0DATOP[6]	-	O	LINK-порт 0. Выход данных
L0DATON[6]	-	O	LINK-порт 0. Выход данных
L0DATOP[7]	-	O	LINK-порт 0. Выход данных
L0DATON[7]	-	O	LINK-порт 0. Выход данных
L0CLKOP	-	O	LINK-порт 0. Выход синхронизации
L0CLKON	-	O	LINK-порт 0. Выход синхронизации
L0DATIP[0]	-	I	LINK-порт 0. Вход данных
L0DATIN[0]	-	I	LINK-порт 0. Вход данных
L0DATIP[1]	-	I	LINK-порт 0. Вход данных
L0DATIN[1]	-	I	LINK-порт 0. Вход данных
L0DATIP[2]	-	I	LINK-порт 0. Вход данных
L0DATIN[2]	-	I	LINK-порт 0. Вход данных
L0DATIP[3]	-	I	LINK-порт 0. Вход данных
L0DATIN[3]	-	I	LINK-порт 0. Вход данных
L0DATIP[4]	-	I	LINK-порт 0. Вход данных
L0DATIN[4]	-	I	LINK-порт 0. Вход данных
L0DATIP[5]	-	I	LINK-порт 0. Вход данных
L0DATIN[5]	-	I	LINK-порт 0. Вход данных
L0DATIP[6]	-	I	LINK-порт 0. Вход данных
L0DATIN[6]	-	I	LINK-порт 0. Вход данных
L0DATIP[7]	-	I	LINK-порт 0. Вход данных
L0DATIN[7]	-	I	LINK-порт 0. Вход данных
L0CLKIP	-	I	LINK-порта 0. Вход синхросигнала(+)
L0CLKIN	-	I	LINK-порта 0. Вход синхросигнала(-)
L1DATOP[0]	-	O	LINK-порт1. Выход данных
L1DATON[0]	-	O	LINK-порт1. Выход данных
L1DATOP[1]	-	O	LINK-порт1. Выход данных
L1DATON[1]	-	O	LINK-порт1. Выход данных
L1DATOP[2]	-	O	LINK-порт1. Выход данных
L1DATON[2]	-	O	LINK-порт1. Выход данных
L1DATOP[3]	-	O	LINK-порт1. Выход данных

Обозначение вывода (основное)	Обозначение вывода для портов связи	Тип вывода	Функциональное назначение
L1DATON[3]	-	O	LINK-порт1. Выход данных
L1DATOP[4]	-	O	LINK-порт1. Выход данных
L1DATON[4]	-	O	LINK-порт1. Выход данных
L1DATOP[5]	-	O	LINK-порт1. Выход данных
L1DATON[5]	-	O	LINK-порт1. Выход данных
L1DATOP[6]	-	O	LINK-порт1. Выход данных
L1DATON[6]	-	O	LINK-порт1. Выход данных
L1DATOP[7]	-	O	LINK-порт1. Выход данных
L1DATON[7]	-	O	LINK-порт1. Выход данных
L1CLKOP	-	O	LINK-порт 1. Выход синхросигнала(+)
L1CLKON	-	O	LINK-порт 1. Выход синхросигнала(-)
L1DATIP[0]	-	I	LINK-порт 1. Вход данных
L1DATIN[0]	-	I	LINK-порт 1. Вход данных
L1DATIP[1]	-	I	LINK-порт 1. Вход данных
L1DATIN[1]	-	I	LINK-порт 1. Вход данных
L1DATIP[2]	-	I	LINK-порт 1. Вход данных
L1DATIN[2]	-	I	LINK-порт 1. Вход данных
L1DATIP[3]	-	I	LINK-порт 1. Вход данных
L1DATIN[3]	-	I	LINK-порт 1. Вход данных
L1DATIP[4]	-	I	LINK-порт 1. Вход данных
L1DATIN[4]	-	I	LINK-порт 1. Вход данных
L1DATIP[5]	-	I	LINK-порт 1. Вход данных
L1DATIN[5]	-	I	LINK-порт 1. Вход данных
L1DATIP[6]	-	I	LINK-порт 1. Вход данных
L1DATIN[6]	-	I	LINK-порт 1. Вход данных
L1DATIP[7]	-	I	LINK-порт 1. Вход данных
L1DATIN[7]	-	I	LINK-порт 1. Вход данных
L1CLKIP	-	I	LINK-порт 1. Вход синхросигнала(+)
L1CLKIN	-	I	LINK-порт 1. Вход синхросигнала(-)

15.1 Архитектура портов связи

Порты связи подключены к SOC-шине как периферийное устройство (см. рисунок 70).

Порт связи состоит из двух частей – передатчик и приемник. Каналы передачи и приема имеют буфер, как показано на рисунке 71. Буферные регистры, подключенные к SOC-шине, доступны программисту как регистры LBUFTX и LBUFRX. Они являются 128-разрядными, отображаемыми в памяти универсальными регистрами. Регистры дополнительных буферов и сдвиговые регистры являются программно недоступными.

Дополнительная буферизация позволяет конвейеризировать операции приема-передачи порта связи и операции контроллера DMA. Также имеет значение, что

сдвиговые регистры приема-передачи имеют собственную частоту работы, которая отличается от частоты SOC-шины.

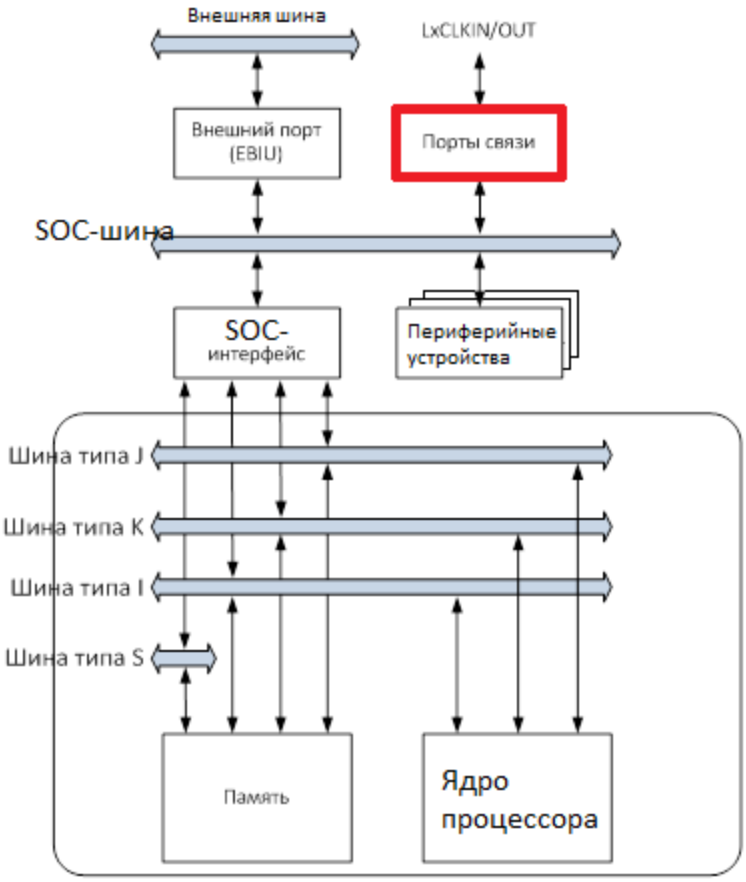


Рисунок 70 – Подключение портов связи на кристалле

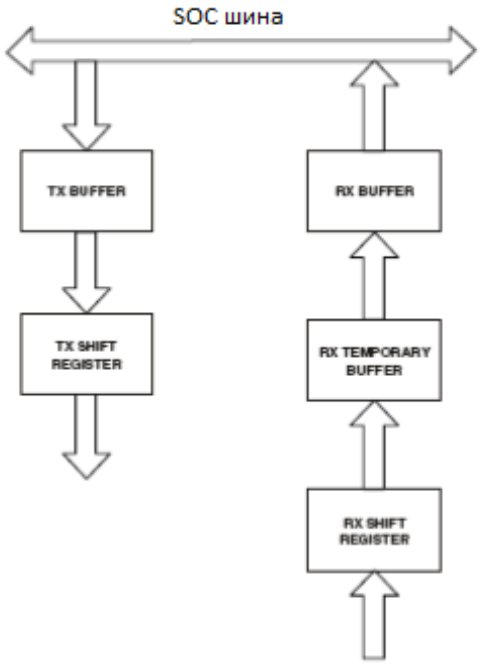


Рисунок 71 – Архитектура порта связи

15.2 Внешние выводы портов связи

Для выполнения обмена данными с внешними устройствами порты связи используют внешние выводы микросхемы, которые описаны в таблице 121. В таблице приведен перечень выводов одного порта связи. Знак ‘x’ в имени сигнала указывает на порт связи – 0 или 1. В набор внешних выводов порта связи входят как дифференциальные выводы, так и обычные выводы. Для каждого дифференциального вывода существует пара линий: P и N. На линии P передается истинное значение сигнала, а на линии N противоположное. Раздельные шины данных для приема и для передачи имеют разрядность 8 бит каждая. При этом имеется режим передачи с использованием только одного разряда шины данных из восьми.

Таблица 121 – Описание выводов – порты связи

Сигнал	Описание
LxDATO7–0P	Шина данных 7–0 передача LVDS P
LxDATO7–0N	Шина данных 7–0 передача LVDS N
LxCLKOUTP	Тактовый генератор передачи LVDS P
LxCLKOUTN	Тактовый генератор передачи LVDS N
LxACKI	Входной сигнал подтверждения передатчика. Используя этот сигнал, приемник указывает передатчику, что можно продолжать передачу
nLxBCMPO	Завершение блока. Когда передача выполняется с использованием DMA, данный сигнал указывает приемнику, что передаваемый блок закончился. Во время сброса вывод L1BCMPO (PC[30]) используется как вход для конфигурирования процессора. Перед началом инициализации порта, порт связи выдает сигнал высокого уровня на nLxBCMPO (деактивирован), указывая приемнику, что порт подсоединен
LxDATI7–0P	Данные 7–0 прием LVDS P
LxDATI7–0N	Данные 7–0 прием LVDS N
LxCLKINP	Тактовый генератор приема LVDS P
LxCLKINN	Тактовый генератор приема LVDS N
LxACKO	Выходной сигнал подтверждения приемника. Используя этот сигнал, приемник указывает передатчику, что можно продолжать передачу.
nLxBCMPI	Завершение блока. Когда передача выполняется с использованием DMA, данный сигнал указывает приемнику, что передаваемый блок закончился. После сброса сигнал высокого уровня на nLxBCMPI указывает на то, что к приемнику порта подключен внешний передатчик

Каждый порт связи имеет два независимых канала, один для приема и второй для передачи, которые могу работать одновременно. Канал передачи передает данные на другое устройство, а канал приема получает данные с другого устройства. Каждый канал осуществляет обмен данными, используя до восьми бит данных и используя сигналы LxCLKOUTP/N, LxACKI, LxCLKINP/N и LxACKO для управления передачей данных.

Сигналы nLxBCMPI и nLxBCMPO используются для информирования о том, что текущая передача блока данных завершена. Порты связи должны соединяться так, как показано на рисунках 72 и 73.

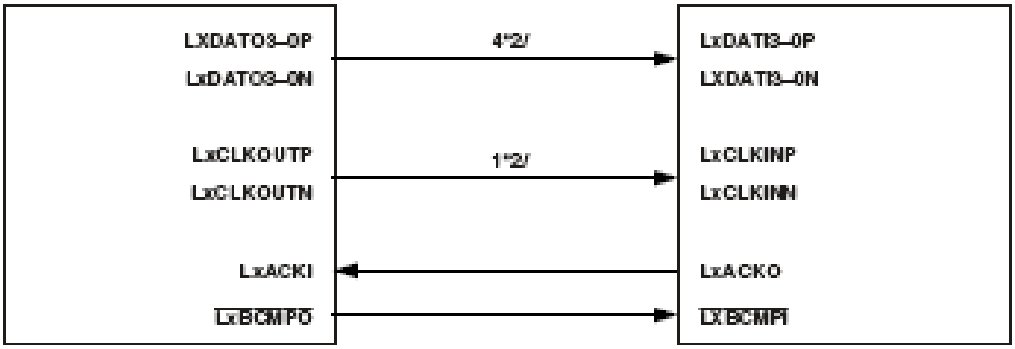


Рисунок 72 – Конфигурация порта связи (четырехразрядный режим)

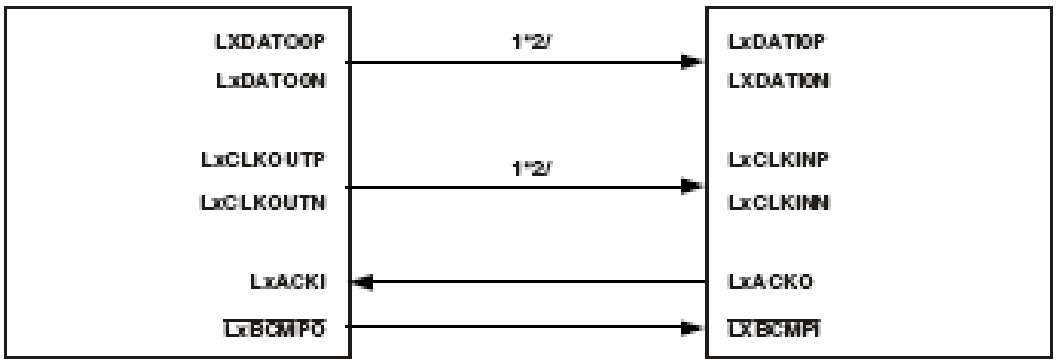


Рисунок 73 – Конфигурация порта связи (одноразрядный режим)

Порты связи процессора являются тактируемыми высокоскоростными LVDS портами данных. LVDS является стандартом для дифференциальной передачи сигналов между удаленными элементами. LVDS обеспечивает более высокую частоту, более высокий уровень устойчивости к шумам, более низкое энергопотребление и меньшее число электромагнитных помех.

Передача сигналов LVDS требует дифференциального завершения. Рисунок 74 показывает процесс передачи через порт от одного процессора к другому. Внешние резисторы согласования (RT) 100 Ом должны быть установлены на плате как можно ближе к выводам микросхемы. Линии на плате должны быть по возможности одинаковыми для того, чтобы поддерживать одинаковое время задержки для всех выводов данных и тактовых сигналов.

Линии несимметричных сигналов (LxACKI, LxACKO, nLxBCMPI и nLxBCMPO) не так критичны, но их задержки должны были близки к задержкам соответствующих им дифференциальных сигналов.

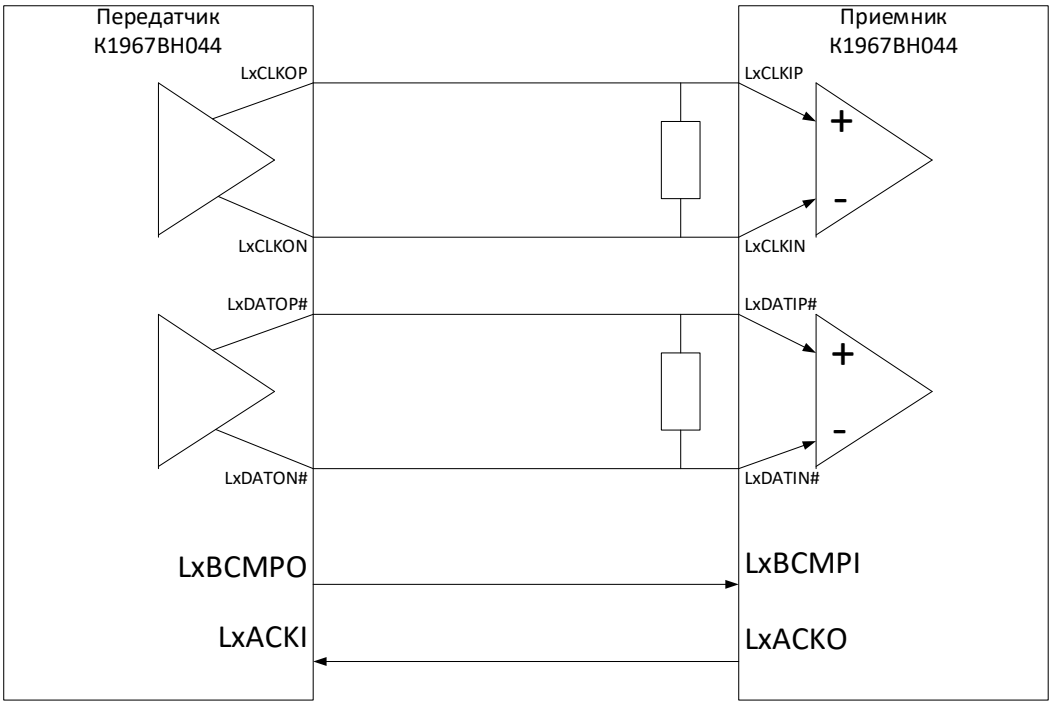


Рисунок 74 – Конфигурация передачи между процессорами

15.3 Группа регистров портов связи

Регистры буфера портов связи доступны как счетверенные слова. Две группы регистров данного типа описаны в таблицах 122 и 123. Регистры управления и статуса доступны как одинарные слова. Базовый адрес группы регистров буфера приема/передачи портов связи 0x8000_00A0 (таблица 122), базовый адрес группы регистров статуса и контроля портов связи 0x8000_00E0 (таблица 123).

Таблица 122 – Группа регистров буфера приема/передачи порта связи

Имя	Смещение	Тип	Значение после сброса	Описание
LBUFTX0:0	0x00	R/W	-	Данные передатчика порта связи 0
LBUFTX0:1	0x01	R/W	-	Данные передатчика порта связи 0
LBUFTX0:2	0x02	R/W	-	Данные передатчика порта связи 0
LBUFTX0:3	0x03	R/W	-	Данные передатчика порта связи 0
LBUFRX0:0	0x04	R	-	Данные приемника порта связи 0
LBUFRX0:1	0x05	R	-	Данные приемника порта связи 0
LBUFRX0:2	0x06	R		Данные приемника порта связи 0
LBUFRX0:3	0x07	R	-	Данные приемника порта связи 0
LBUFTX1:0	0x08	R/W	-	Данные передатчика порта связи 1
LBUFTX1:1	0x09	R/W	-	Данные передатчика порта связи 1
LBUFTX1:2	0x0A	R/W	-	Данные передатчика порта связи 1
LBUFTX1:3	0x0B	R/W	-	Данные передатчика порта связи 1
LBUFRX1:0	0x0C	R	-	Данные приемника порта связи 1
LBUFRX1:1	0x0D	R	-	Данные приемника порта связи 1
LBUFRX1:2	0x0E	R	-	Данные приемника порта связи 1
LBUFRX1:3	0x0F	R	-	Данные приемника порта связи 1

Таблица 123 – Группа регистров статуса и контроля порта связи

Имя	Смещение	Тип	Значение после сброса	Описание
LRCTL0	0x00	R/W	0x0	Регистр управления приемником порта связи 0
LRCTL1	0x01	R/W	0x0	Регистр управления приемником порта связи 1
-	0x02 - 0x03		-	Зарезервировано
LTCTL0	0x04	R/W	0x0	Регистр управления передатчиком порта связи 0
LTCTL1	0x05	R/W	0x0	Регистр управления передатчиком порта связи 1
LIM_VAL0	0x08	R/W	0x0	Регистр количества принимаемых данных при работе от внешнего события (для порта связи 0) (для микросхем с ревизии 3)
LIM_VAL1	0x09	R/W	0x0	Регистр количества принимаемых данных при работе от внешнего события (для порта связи 1) (для микросхем с ревизии 3)
-	0x0A - 0x0B		-	Зарезервировано
LRDLY0	0x0C	R/W	0x0	Регистр управления задержками LVDS-приемников порта связи 0
LRDLY1	0x0D	R/W	0x0	Регистр управления задержками LVDS-приемников порта связи 1
-	0x0E - 0x0F		-	Зарезервировано
LRSTAT0	0x10	R	0x0	Регистр состояния приемника порта связи 0
LRSTAT1	0x11	R	0x0	Регистр состояния приемника порта связи 1
-	0x12 - 0x13		-	Зарезервировано
LTSTAT0	0x14	R	0x2	Регистр состояния передатчика порта связи 0
LTSTAT1	0x15	R	0x2	Регистр состояния передатчика порта связи 1
-	0x16 - 0x17		-	Зарезервировано
LRSTATC0	0x18	R	0x0	Регистр сброса состояния приемника порта связи 0
LRSTATC1	0x19	R	0x0	Регистр сброса состояния приемника порта связи 1
-	0x1A - 0x1B		-	Зарезервировано
LTSTATC0	0x1C	R	0x2	Регистр сброса состояния передатчика порта связи 0
LTSTATC1	0x1D	R	0x2	Регистр сброса состояния передатчика порта связи 1
-	0x1E - 0x1F		-	Зарезервировано

15.3.1 Регистр сброса состояния приемника/передатчика порта связи

Чтение регистра возвращает значение регистра состояния передатчика/приемника с автоматическим обнулением некоторых бит регистра статуса приемника/передатчика.

15.4 Прием и передача данных

Обмен данными осуществляется посредством записи в буфер передачи и чтением из буфера приема. Длина передаваемых данных всегда равна 128 битам. Все данные, записанные в буфер передачи, копируются в сдвиговый регистр, как только он становится пустым, и после этого передаются. Приемник разрешает продолжение приема данных, только когда его сдвиговый регистр пуст или, когда в его буферных регистрах есть достаточно места для приема данных из сдвигового регистра, когда прием этого счетверенного слова будет завершен. После того, как целое счетверенное слово получено, приемник перемещает данные из сдвигового регистра в буфер приема, когда он свободен.

Если приемник не готов принять следующее слово данных, он использует линию подтверждения LxACKO для приостановки работы передатчика.

В таблицах 122, 123 приведены имена регистров, которые пользователь может использовать при программировании портов связи.

15.5 Связь с DMA

Каждый порт связи может соединяться с двумя каналами DMA. Один канал используется для передачи данных, в то время как второй используется для приема данных. Оба канала DMA связаны через интерфейс с внутренней памятью, внешней памятью или другими буферами портов связи. Передатчики портов связи с нулевого по первый соединены соответственно с четвертого по пятый каналами DMA. Приемники портов связи с нулевого по первый соединены соответственно с восьмого по девятый каналами DMA. Каналы DMA с четвертого по пятый называются каналами передачи, т.к. передают данные передатчикам порта связи, а каналы с восьмого по девятый называются каналами приема, т.к. принимают данные от приемников порта связи.

Передатчик порта связи формирует сервисный запрос к каналу передачи DMA, когда буферный регистр LBUFTX пуст и канал DMA включен. Приемник порта связи формирует запрос к каналу приема DMA, когда он записывает счетверенное слово данных в буферный регистр LBUFRX и канал DMA включен.

Канал-приемник DMA (с восьмого по девятый) может также использоваться для транзитных передач посредством записи данных из приемника своего порта связи в буферный регистр передатчика любого другого порта связи. Канал DMA при этом отслеживает, что буферный регистр передатчика свободен.

15.6 Завершение блочной передачи

Функция завершения передачи блока позволяет передатчику информировать приемник о том, что блочная передача завершена.

Поскольку канал передачи DMA точно знает, сколько слов необходимо передать и может отследить момент передачи последнего слова, то вместе с записью последнего слова в буфер передатчика, он формирует сигнал завершения передачи блока и передает

его в передатчик вместе с данными. Порт связи активизирует nLxBCMPO, когда он передает последнее счетверенное слово и в регистре управления LTCTL установлен разряд BCMPE.

Когда вход nLxBCMPI активен, то принимающий порт связи передает эту информацию каналу DMA вместе с запросом. Это позволяет каналу приемнику DMA отследить завершение приема и закончить свою работу.

15.7 Прерывания портов связи

Порты связи имеют специализированные прерывания для управления потоком данных, когда порт связи осуществляет передачу с использованием ядра процессора (в отличие от передачи с использованием каналов DMA). Прерывание от приемника порта связи является активным только тогда, когда канал DMA для порта связи не включается. Порт приема активизирует прерывание, когда счетверенное слово, получаемое портом связи, ожидает в регистре LBUFRX. Прерывание приема порта связи является чувствительным по уровню и поэтому, если канал DMA, взаимодействующий с портом связи, становится активным после разрешения прерывания порта связи, прерывание приема порта связи будет деактивировано и будет сформирован запрос к DMA.

15.8 Инициализация после сброса и загрузка через порт связи

Старт процессора может быть выполнен посредством загрузки начального кода программы через порты связи. В этом режиме порт работает как подчиненное устройство, настроенное на прием данных. После сброса все каналы DMA порта приема могут настраиваться для передачи блока из 256 слов во внутреннюю память по адресам с 0 до 255 и настраиваются для формирования прерывания по окончании передачи блока. Соответствующие вектора прерываний для каналов DMA устанавливаются в нулевой адрес.

При сбросе или запрещении портов связи через регистры управления, содержимое буферов порта связи сбрасывается.

15.9 Протокол передачи данных порта связи

Каждый порт связи имеет два независимых канала (канал передачи и канал приема), которые могут работать одновременно. Канал передачи передает данные на внешнее устройство, канал приема получает данные с внешнего устройства. Каждый канал осуществляет передачу данных через одно-, четырех- или восьмиразрядную шину данных с использованием трех типов контрольных сигналов (таблица 122). Сигналы LxCLKOUTP/N, LxCLKINP/N и LxACKI, LxACKO используются для управления передачей данных. Сигналы nLxBCMPI, nLxBCMPO используются для оповещения о завершении текущей передачи блока.

Два порта связи могут быть сконфигурированы как один порт с 16-разрядной шиной данных по каждому направлению (TX и RX). Управление объединенным портом связи выполняется через регистры порта связи 0. Для приема данных используются выводы L1DATI7-0P/N, L0DATI7-0P/N и L0CLKINP/N, а для передачи L1DATO7-0P/N,

L0DATO7-0P/N и L0CLKOUTP/N (Рисунок 75). Управление передачей данных осуществляется с помощью выводов L0ACKI, L0ACKO, nL0BCMPI и nL0BCMPO.

Существует несколько общих правил, которые применяются в протоколе порта связи (рисунки 75 – 83). Знание этих правил позволяет понять примеры, приведенные на диаграммах. Общие правила таковы:

– первые данные передаются по фронту (из «0» в «1») тактового сигнала порта связи (условно LxCLKOUTP);

– последние данные передаются по срезу (из «1» в «0») тактового сигнала порта связи (условно LxCLKOUTP);

– LxCLKOUTP переводится в низкий уровень, когда порт связи находится в состоянии простоя.

Минимальный объем передачи данных через порт равен счетверенному слову. Счетверенное слово может быть передано:

– в течение 4 тактовых циклов, когда используется 16 разрядов шины данных;
– в течение 8 тактовых циклов, когда используется 8 разрядов шины данных;
– в течение 16 тактовых циклов, когда используются 4 разряда шины данных;
– в течение 64 циклов, когда используется только 1 разряд данных.

На рисунках 75 – 78 показаны случаи обмена для различных шин данных.

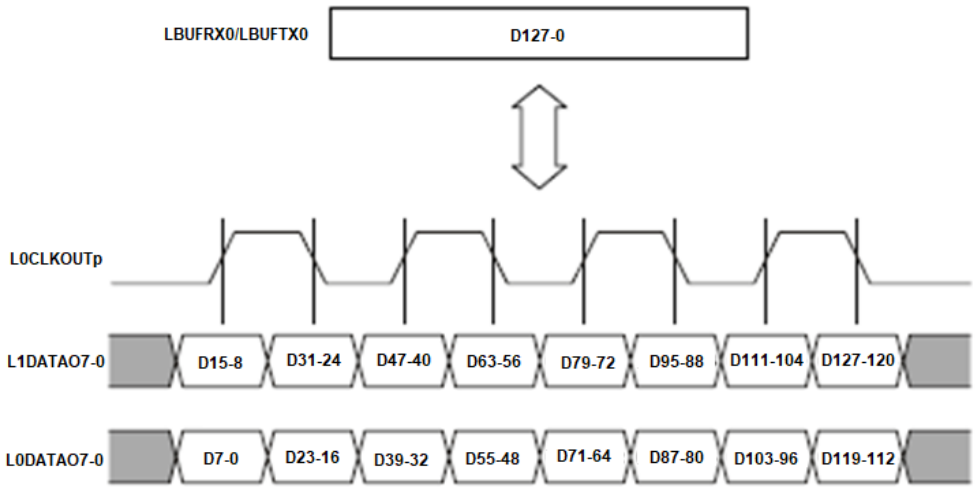


Рисунок 75 – Шина данных 16-разрядов

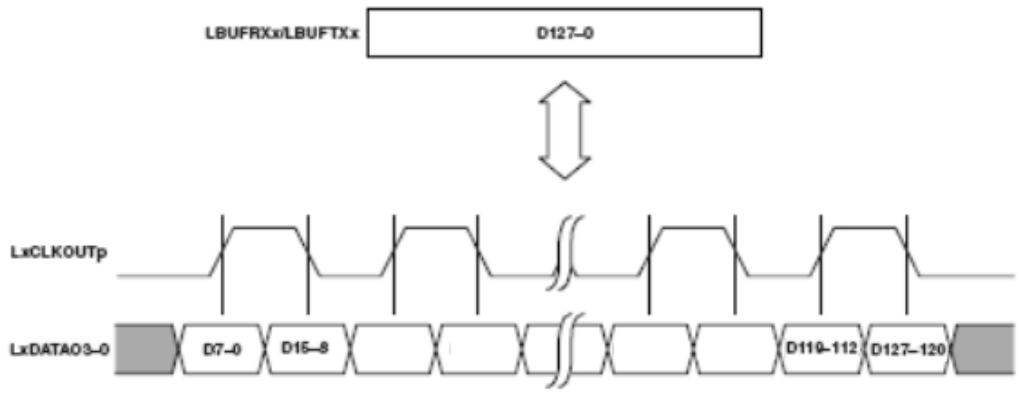


Рисунок 76 – Шина данных восемь разрядов

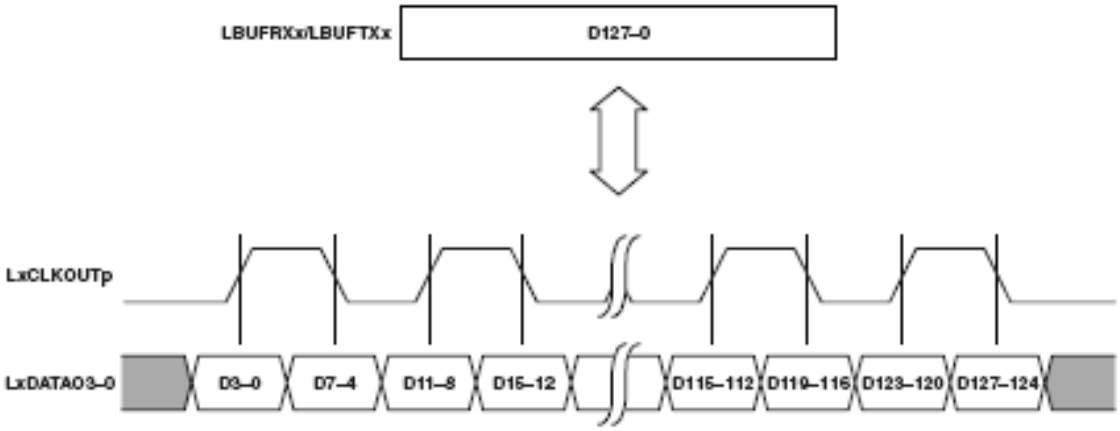


Рисунок 77 – Шина данных четыре разряда

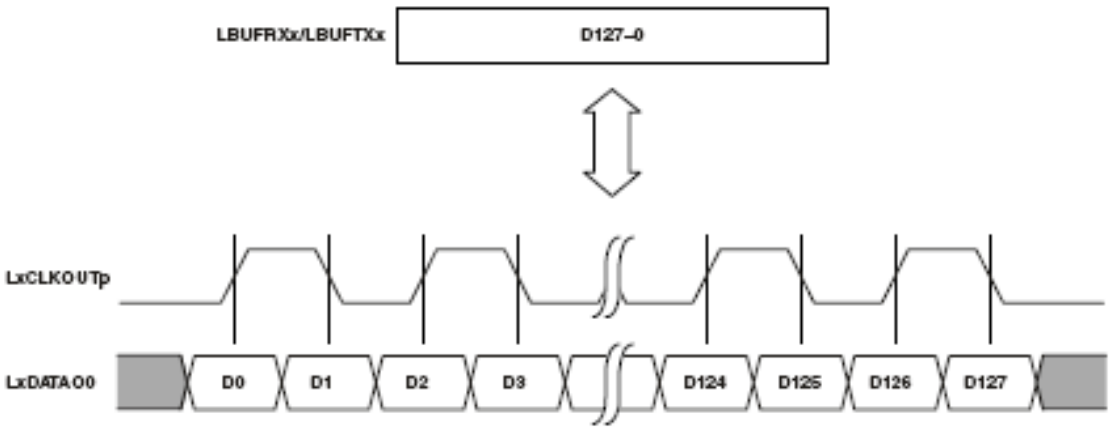


Рисунок 78 – Шина данных один разряд

Передача данных начинается, когда на LxACKI высокий уровень сигнала, что обозначает, что буфер приема свободен. Как показано на рисунках, первые биты данных достоверны до первого фронта LxCLKOUTP и следующие за ними биты данных достоверны до среза тактового сигнала.

Если на момент окончания передачи текущих данных буфер передатчика не пуст, и при этом на LxACKI установлен высокий уровень сигнала (буфер приемника свободен), то сразу же начинается передача следующих данных (рисунок 79). Таким образом при наличии данных в буфере передатчика и наличии свободного места в буфере приемника реализуется передача непрерывного потока данных.

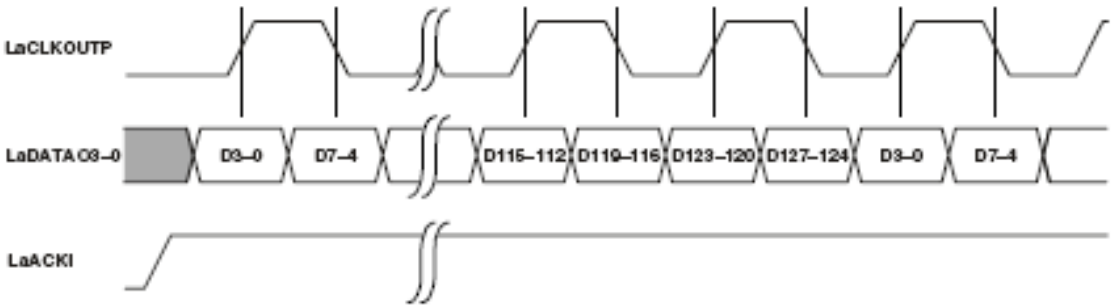


Рисунок 79 – Порт связи «a» передает слова порту связи «b» одно за другим

Рисунок 80 иллюстрирует состояние порта связи «b», когда он не готов принимать больше данных. В этом случае после передачи текущего слова данных передатчик будет ждать, пока сигнал подтверждения не примет высокий уровень.

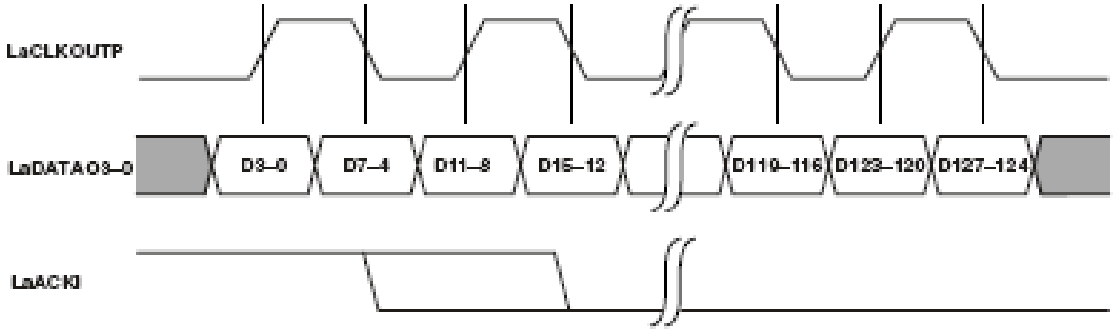


Рисунок 80 – Порт связи «a» передает порту связи «b» (приемник заполнен)

После установки сигнала подтверждения в высокий уровень задержка между фронтом сигнала LxACKI и первым фронтом тактового сигнала LxCLKOUT (начало передачи данных) составляет не более четырех периодов тактовой частоты LINK_PLL.

При работе с внешними приемными устройствами, такими как DAC, которые не формируют сигнал подтверждения, предусмотрен режим работы передатчика, при котором сигнал LxACKI игнорируется, и передача данных всегда разрешена. Данный режим активен, когда в регистре управления LTCTL установлен разряд TX_DAC_MODE.

На последовательность передачи влияет установка разрядов верификации данных. Это разряд RVERE в регистре LRCTL и разряд TVERE в регистре LTCTL. Если установлен разряд TVERE, байт контрольной суммы отправляется после последнего байта в счетверенном слове. За байтом контрольной суммы всегда следует «пустой» байт. Отправка байта контрольной суммы (биты Vx) и «пустого» байта (Xx) выполняется в течение двух циклов для шины четырехразрядных данных и восьми циклов для шины одноразрядных данных. Рисунок 81 иллюстрирует последовательность передачи для случая одноразрядной шины данных.

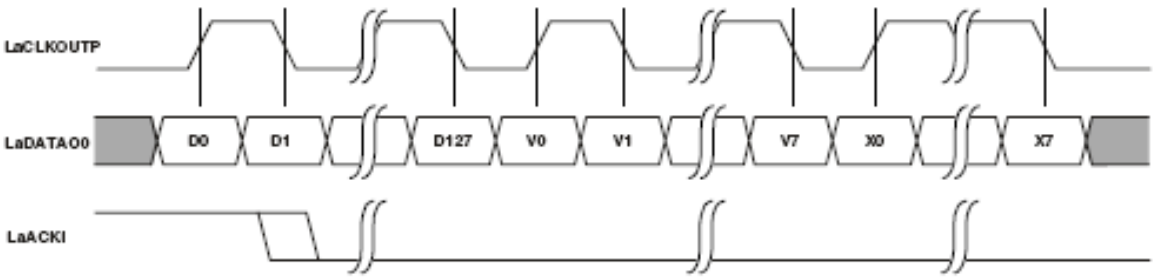


Рисунок 81 – Порт связи «a» передает порту связи «b» с верификацией

Сообщения между различными устройствами в системе обычно имеют различную длину. В некоторых случаях блоки данных также имеют различную длину. При этом знание длины блока не всегда доступно для принимающего устройства и не может быть передано передающим устройством. В этом случае на помощь приходят управляющие сигналы nLxBCMPO и nLxBCMPI.

Передающий порт связи указывает порту связи приема, что блок завершен с использованием выходного сигнала nLxBCMPO передатчика, который связан с входным сигналом приемника nLxBCMPI.

Когда приемник распознает этот сигнал, он передает информацию каналу DMA о том, что блок данных завершен. В результате этого канал DMA игнорирует то, что его счетчик слов еще не достиг значения нуля, и работает так, как при завершении блока.

Сигнал nLxBCMPO указывает на завершение блока переходом на низкий уровень после первого фронта LxCLKOUT в последнем счетверенном слове блока. На nLxBCMPO устанавливается высокий уровень сигнала (неактивный) перед последним срезом сигнала LxCLKOUT того же счетверенного слова. Сигнал является неактивным, если разряд TBCMPE сбрасывается в регистр LTCTL или, когда передачи порта связи выполняются под управлением ядра процессора.

Рисунок 82 показывает, как порт связи «a» передает сигнал порту связи «b» о том, что текущий буфер завершен.

Рисунок 82 – Порт связи «a» передает порту связи «b» с завершением блока

Порт связи «a» передает сигналы порту связи «b» о том, что текущий буфер заполнен (рисунок 83). Новое счетверенное слово следует за завершенным блоком.

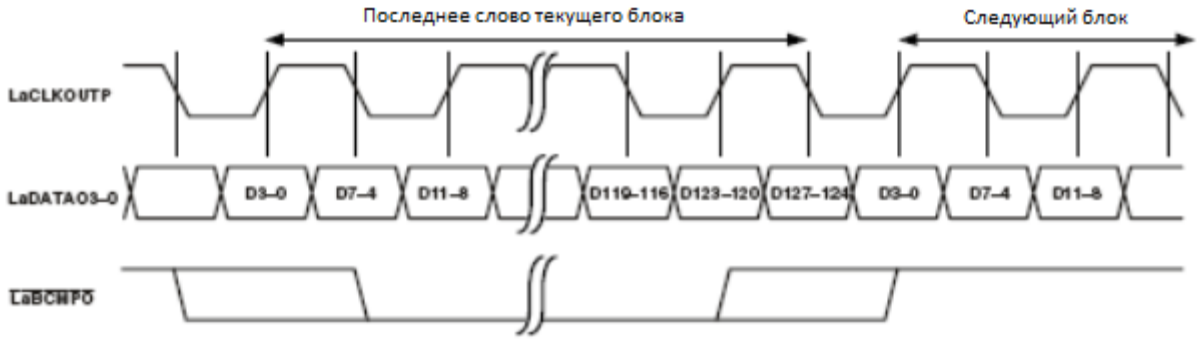


Рисунок 83 – Порт связи «a» передает порту связи «b» с завершением блока
(новый блок из порта связи «a» следует немедленно после завершения первого блока)

15.10 Задержки передачи через порт связи

Порты связи должны быть в состоянии преодолевать задержки между источником и пунктом назначения. Задержки для различных типов сигналов (LxCLKOUT и LxDAT7-O) должны соответствовать допускам, указанным в техническом описании. Существует два типа ограничений для задержек в системе:

– задержка от LxCLKOUT в передатчике к LxCLKIN в приемнике плюс задержка от LxACKO в приемнике к LxACKI в передатчике должны быть меньше чем половина периода передачи счетверенного слова.

– максимальная разница между задержкой трассировки nLxBCMPO и задержкой трассировки тактового сигнала (и данных) составляет одну треть периода передачи счетверенного слова.

Первое ограничение связано со следующим фактором. Если передатчик отправляет данные, а приемник передает сигнал о том, что он не готов принять больше данных через LxACKO, передатчик должен сначала обработать LxACKI до того, как он отправит следующие данные. Если LxACKI не проанализирован вовремя, данные будут утеряны приемником.

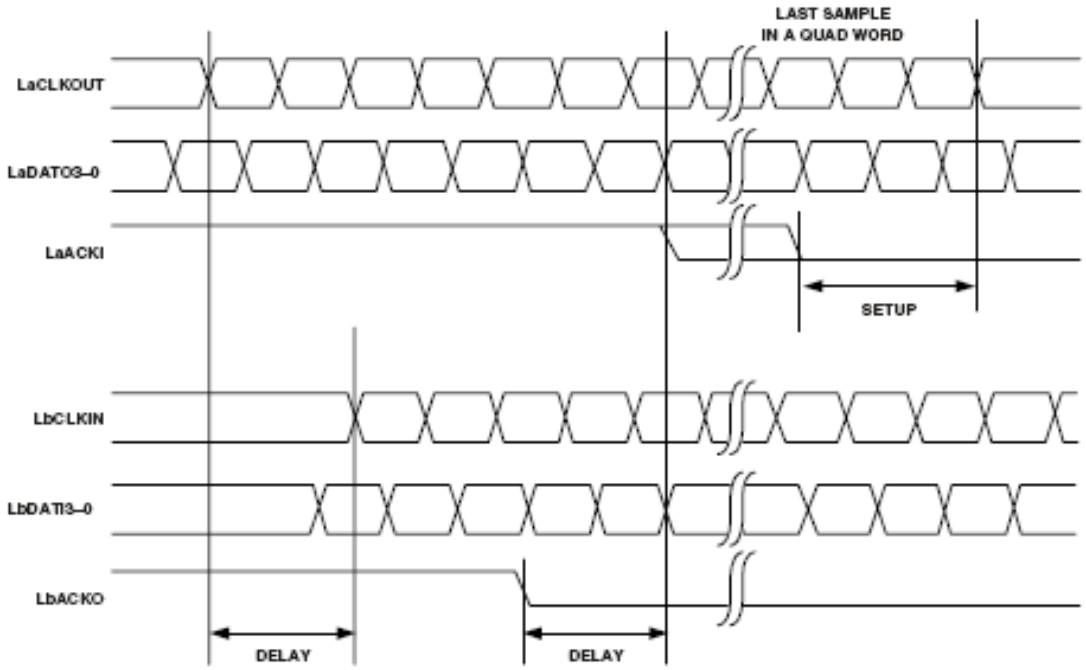


Рисунок 84 – Порт связи «a» передает порту связи «b» с задержкой

15.11 Временные характеристики входных и выходных сигналов порта связи

В зависимости от источника внешнего синхросигнала, сопровождающего входные данные, есть различия во временах предустановки и удержания данных, что иллюстрируется на рисунках 85 и 86. Значения времен предустановки и удержания в зависимости от напряжения питания U_{CCIO} приведены в таблицах 124 и 125. Временная диаграмма выходных данных порта связи приведена на

рисунке 87, значения времен предустановки и удержания выходных данных в зависимости от напряжения питания U_{CCIO} приведены в таблице 126.

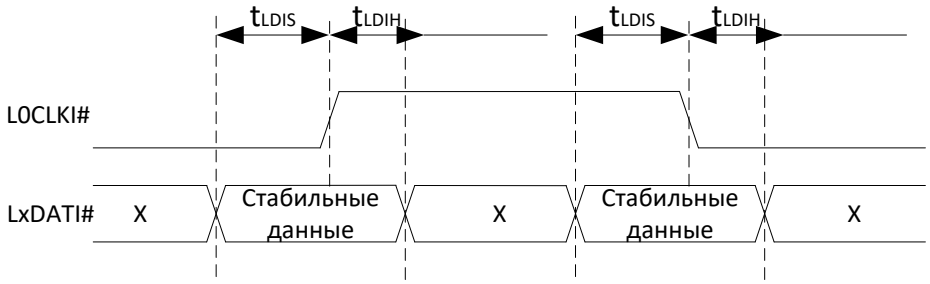


Рисунок 85 – Временная диаграмма входных данных порта связи при тактировании от L0CLKI#

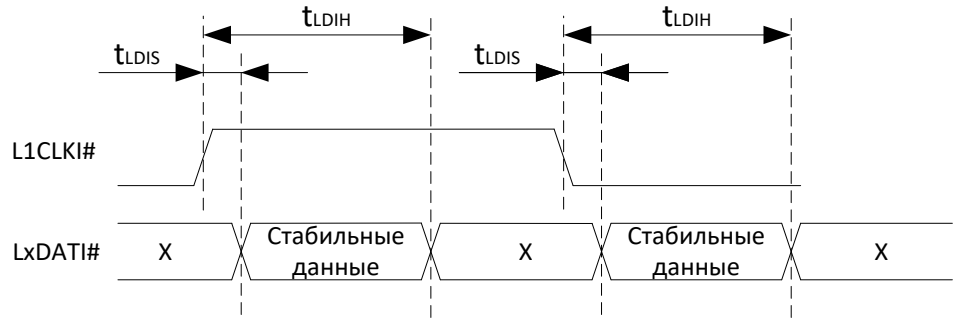


Рисунок 86 – Временная диаграмма входных данных порта связи при тактировании от L1CLKI#

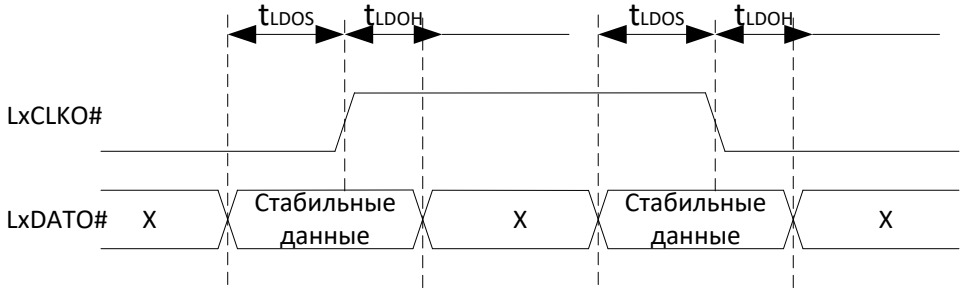


Рисунок 87 – Временная диаграмма выходных данных порта связи

Таблица 124 – Временные характеристики входных данных порта связи при тактировании от L0CLKI#

Наименование параметра, единица измерения	Буквенное обозначение параметра	Напряжение питания U_{CCIO}, В	Норма параметра	
			не менее	не более
Время предустановки входных данных LxDATI#, нс	t_{LDIS}	3,0	0,450	–
		3,3	0,400	–
		3,6	0,450	–
Время удержания входных данных LxDATI#, нс	t_{LDIH}	3,0	0,400	–
		3,3	0,250	–
		3,6	0,200	–

Таблица 125 – Временные характеристики входных данных порта связи при тактировании от L1CLKI#

Наименование параметра, единица измерения	Буквенное обозначение параметра	Напряжение питания U_{CCIO}, В	Норма параметра	
			не менее	не более
Время предустановки входных данных LxDATI#, нс	t_{LDIS}	3,0	–	0,250
		3,3	–	0,250
		3,6	–	0,250
Время удержания входных данных LxDATI#, нс	t_{LDIH}	3,0	1,4	–
		3,3	1,3	–
		3,6	1,15	–

Таблица 126 – Временные характеристики выходных данных порта связи

Наименование параметра, единица измерения	Буквенное обозначение параметра	Напряжение питания U_{CCIO}, В	Норма параметра	
			не менее	не более
Время предустановки выходных данных LxDATO#, нс	t_{LDOS}	3,0	2,3	–
		3,3	2,4	–
		3,6	2,5	–
Время удержания выходных данных LxDATO#, нс	t_{LDOH}	3,0	1,9	–
		3,3	2,0	–
		3,6	2,0	–

15.12 Механизмы определения ошибок порта связи

Порты связи поддерживают функции обнаружения ошибок в процессе работы. Когда порт связи обнаруживает состояние ошибки, он продолжает работу в соответствии со следующим алгоритмом:

– Если обнаружена ошибка истечения времени, устанавливаются соответственно разряды TTER (ошибка истечения времени передачи) в регистр LTSTAT или RTER (ошибка истечения времени приема) в регистр LRSTAT. Если ошибки истечения времени разрешены в регистрах LRCTL или LTCTL, активируется аппаратное прерывание.

– Если обнаружено переполнение данных в приемнике, разряд ROVER устанавливается в LRSTAT. Если разрешена ошибка переполнения данных приема в LRCTL, активируется аппаратное прерывание.

– Если верификация разрешена в LRCTL и обнаружена неверная контрольная сумма, разряд RCSER устанавливается соответственно в LRSTAT и активируется аппаратное прерывание.

– Если активное значение записывается в LRCTL или LTCTL и значение в регистре управления является активным, перезапись регистра предотвращается, разряд RWER устанавливается в LRSTAT или разряд TWER устанавливается в LTSTAT соответственно, активируется аппаратное прерывание.

Поля в LRSTAT и LTSTAT, указывающие на ошибку, могут быть очищены с помощью чтения из соответствующих очищенных регистров LRSTATC и LTSTATC.

15.12.1 Время ожидания передачи через порт связи

Если в передатчике LxACKI деактивирован в течение периода 2048 тактовых циклов и в передатчике есть данные, готовые к передаче, происходит ошибка времени ожидания (time-out) и разряд TTER устанавливается в LTSTAT. На линии LxACKI имеется слабый резистор, понижающий логический уровень. В случае если нет подключения приемника, вход будет в неактивном состоянии. Данная ошибка позволяет обнаружить отсутствие приемника вообще или сбой в его работе. Также эта ошибка позволяет предотвратить возможное зависание системы в случае ожидания готовности передачи.

15.12.2 Время ожидания приемника

Если принимающий буфер приемника полон и нет доступа для чтения в течение 2048 тактовых циклов, происходит ошибка окончания времени ожидания и разряд RTER устанавливается в LRSTAT. Данная ошибка информирует процессор о возможной некорректной работе программы обслуживания канала связи.

15.12.3 Ошибка верификации порта связи

Передатчик имеет возможность формировать байт контрольной суммы и отправлять его вместе с передаваемым счетверенным словом. Передача контрольного байта разрешается установкой разряда TVERE в регистре LTCTL. Контрольный байт рассчитывается как сумма всех байтов данных, которые были переданы. 128-разрядное слово состоит из 16 байтов и контрольный байт отправляется в конце передачи каждых 16 байтов. За байтом контрольной суммы всегда следует «пустой» байт.

Для разрешения верификации данных приемником необходимо установить разряд RVERE в регистре LRCTL. В этом случае приемник принимает первые 16 байт данных, производит вычисление собственной контрольной суммы и сравнивает ее с принятым значением. Если два байта различаются, разряд RCSER устанавливается в регистр LSTAT и формируется аппаратное прерывание.

Алгоритм контрольной суммы следующий:

– Checksum = младший байт от суммы (B0 + B1 + … + B15).

15.12.4 Ошибка записи приема/передачи через порт связи

Запись активного значения в регистр управления приемника или передатчика является допустимой только при выключенном приемнике или передатчике. Запись активного значения в активный управляющий регистр вызывает ошибку записи. Таким образом для того, чтобы поменять некоторые параметры обмена необходимо предварительно выключить соответствующий приемник или передатчик и затем включить его снова с требуемым значением.

15.13 Регистр управления приемником порта связи (LRCTLx)

Регистры LRCTLx определяют параметры приема для порта связи. Подробное описание разрядов регистра приведено в таблице 127.

Таблица 127 – Регистр LRCTL

Бит	Имя	Назначение
0	REN	Бит включения приемника: 1 – включен; 0 – выключен
1	RVERE	Разрешение контроля при приеме: 1 – разрешено; 0 – запрещено
2	RTOE	Разрешение прерывания в случае обнаружения ошибки time out
3	RBCMPE	Разрешение анализа входа nLxBCMPI: 1 – разрешено; 0 – запрещено
5:4	RDSIZE	Размер шины данных приема: 11 – 16 бит (только для порта связи 0. Для порта связи 1 значение зарезервировано). В этом режиме программируется только порт связи 0 и порядок бит в принимаемом слове такой: {L1DATI*, L1DATI*}; 10 – 8 бит; 01 – 4 бита; 00 – 1 бит
6	ROVRE	Разрешение прерывания при переполнении буфера приемника: 1 – разрешено; 0 – запрещено
7	RX_EXT_EN	Для микросхем ревизии 2: Зарезервировано. Для микросхем с ревизии 3: Управление приемом по сигналу события P# (либо с внешнего вывода L0BCMPI, либо с бита CFG12[0], подробнее см. описание регистра XCR): 0 – функция старта по сигналу события выключена; 1 – функция старта по сигналу события включена
8	-	Зарезервировано
9	GPS_CLK_EN	Разрешение генерации клока GPS
10	-	Зарезервировано
11	DATA_SRC	Выбор приемника данных: 0 – буфер порта связи; 1 – модуль UP-DOWN
13:12	ADCW	Режим для ADC 8 бит шины данных: 00 – 16 бит (LxDATI7–0P/N); 01 – 14 бит (LxDATI7–1P/N); 10 – 12 бит (LxDATI7–2P/N); 11 – 10 бит (LxDATI7–3P/N)
14	COMPL	Режим представления данных (только для шины данных 8 бит): 0 – входные данные представлены в дополнительном коде и не преобразуются для дальнейшей обработки; 1 – входные данные представлены в смещенном коде и преобразуются в дополнительный для дальнейшей обработки (каждые 16 бит входных данных рассматриваются как независимые и модифицируются инверсией знакового бита)

Бит	Имя	Назначение
15	ADC_DDR	Режим ADC DDR: 0 – первая половина бит передается по фронту, вторая по срезу; 1 (для микросхем ревизии 2) – четные/нечетные биты передаются по фронту/срезу; 1 (для микросхем с ревизии 3) – четные/нечетные биты передаются по фронту/срезу в зависимости от бита EVEN_ODD
16	RX_CLK_SRC	Источник клока: 0 – L0CLKI; 1 – L1CLKI
17	EVEN_ODD	Только при ADC_DDR = 1 (для микросхем с ревизии 3): 0 – четные биты по фронту, нечетные по срезу; 1 – нечетные биты по фронту, четные по срезу
31:16	-	Зарезервировано

Регистр не может изменяться в процессе выполнения операции обмена. Запись активного значения в контрольный регистр допускается, только когда регистр имеет неактивное значение (REN сброшен). С целью изменения настроек во время работы порта связи, должно быть записано неактивное значение, чтобы прекратить работу порта, после чего следует записать новое активное значение. Игнорирование данного правила может вызвать аппаратное прерывание ошибки и новое значение не будет записано.

Выбор разрядности шины данных для обмена осуществляется в соответствии со следующим правилом:

– если бит 2 регистра CFG_APB модуля CMU установлен в «1», разрядность шины данных приемников портов связи равна четыре бита;

– если бит 2 регистра CFG_APB модуля CMU сброшен в «0», разрядность шины данных приемников портов связи определяется согласно значению бит RDSIZE.

15.14 Регистр управления передатчиком порта связи (LTCTLx)

Регистр устанавливает параметры передачи для порта связи. Значением сброса для регистров LTCTLx является «0». Подробное описание разрядов регистра приведено в таблице 128).

Таблица 128 – Регистр LTCTL

Бит	Имя	Назначение
0	TEN	Бит включения передатчика: 1 – включен; 0 – выключен
1	TVERE	Разрешение формирования контрольной суммы при передаче: 1 – разрешено; 0 – запрещено
2	TTOE	Разрешение прерывания в случае обнаружения ситуации time out
3	TBCMPE	Разрешение формирования выхода LxBCMPO: 1 – разрешено**;** 0 – запрещено

Бит	Имя	Назначение
5:4	TDSIZE	Размер шины передачи: 11 – 16 бит (только для порта связи 0. Для порта связи 1 значение зарезервировано). В этом режиме программируется только порт связи 0 и порядок бит в передаваемом слове такой: {L1DATO*, L1DATO*}; 10 – 8 бит; 01 – 4 бита; 00 – 1 бит
8:6	-	Зарезервировано
10:9	TXC	Источник синхросигнала передатчика: 00, 10 – внутренний генератор (частота формируется синтезатором тактовой частоты LINK_PLL); 01 – синхросигнал приемника; 11 – инвертированный синхросигнал приемника
11	TX_DATA_DST	Выбор источника данных для передачи: 0 – передатчик порта связи; 1 – модуль UP-DOWN
12	TX_COMPL	Режим представления данных: 0 – выходные данные представлены без изменения; 1 – каждые 16 бит выходных данных рассматриваются как независимые и модифицируются инверсией знакового бита
13	TX_DAC_MODE	Режим внешнего DAC: 0 – нормальная работа. Сигнал LxACKI анализируется; 1 – работа с непрерывным потоком данных. Сигнал LxACKI игнорируется
31:14	-	Зарезервировано

Регистр не может изменяться в процессе выполнения операции обмена. Запись активного значения в контрольный регистр допускается, только когда регистр имеет неактивное значение (TEN сброшен). С целью изменения настроек во время работы порта связи, должно быть записано неактивное значение, чтобы прекратить работу порта, после чего следует записать новое активное значение. Игнорирование данного правила может вызвать аппаратное прерывание ошибки и новое значение не будет записано.

15.15 Регистры задания размеров пакетов данных при приеме по событию LIM_VAL (для микросхем с ревизии 3)

Регистры устанавливают значения размеров пакетов (в квадрословах), которые должны быть приняты в режиме синхронизации по внутреннему или внешнему событию. Подробное описание разрядов регистров приведено в таблице 129.

Таблица 129 – Регистр LIM_VAL0, LIM_VAL1

Бит	Имя	Назначение
31-0	LIM_VAL#	Количество квадрослов в пакете

Регистр 32-разрядный и используется приемником в режиме работы с синхронизацией приема по внешнему или внутреннему событию. После сброса значение регистра равно 0x00000000. Пользователь должен задать нужное ему значение до момента включения приемника. Регистр не имеет никакого смысла, если режим синхронизации по событию не используется. Значение LIM_VAL#=0 при включенном режиме синхронизации по событию включает режим бесконечного приема данных. Количество принимаемых квадрослов всегда на 1 больше числа, записываемого в регистр LIM_VAL#. То есть, минимальное значение LIM_VAL# = 1, что означает два квадрослова в пакете.

15.16 Регистр управления задержками приемника порта связи (LRDLYx)

Регистр устанавливает индивидуальные задержки на каждый бит приемника порта связи. Меняя задержки для каждого бита, можно снижать влияние внешней линии связи для обеспечения устойчивого приема. В зависимости от варианта микросхемы, напряжения питания и температуры (PVT), величина единичной задержки может находиться в пределах от 224 до 346 пс. Полная задержка определяется как произведение единичной задержки на десятичное значение из DxDLY.

Таблица 130 – Регистр LRDLY

Бит	Имя	Назначение
2:0	D0_DLY	Задержка бита 0 данных
5:3	D1_DLY	Задержка бита 1 данных
8:6	D2_DLY	Задержка бита 2 данных
11:9	D3_DLY	Задержка бита 3 данных
14:12	D4_DLY	Задержка бита 4 данных
17:15	D5_DLY	Задержка бита 5 данных
20:18	D6_DLY	Задержка бита 6 данных
23:21	D7_DLY	Задержка бита 7 данных
26:24	CLK_DLY	Задержка синхросигнала

15.17 Регистр состояния приемника порта связи (LRSTATx)

Регистры LRSTATx указывают статус приемника порта связи. Значением сброса для регистров LRSTATx является 0x0000 0000. Подробное описание разрядов регистра приведено в таблице 131.

Таблица 131 – Регистр LRSTAT

Бит	Имя	Назначение
0	RSTAT	Состояние буфера приемника: 0 – буфер пустой; 1 – буфер не пустой
1	RSTAT_ADD	Состояние промежуточного буфера приемника (вспомогательный бит, не является битом готовности данных для считывания программой):

Бит	Имя	Назначение
		0 – вспомогательный буфер пустой; 1 – вспомогательный буфер не пустой
2	RTER	1 – приемник обнаружил ситуацию time out; 0 – нет ошибки. Бит может быть сброшен чтением регистра LRSTATCx
3	RWER	1 – приемник обнаружил ошибку записи; 0 – нет ошибки. Бит может быть сброшен чтением регистра LRSTATCx
4	RCSER	1 – обнаружена ошибка контрольной суммы; 0 – нет ошибки. Бит может быть сброшен чтением регистра LRSTATCx
5	ROVER	1 – ошибка переполнения приемника; 0 – нет ошибки. Бит может быть сброшен чтением регистра LRSTATCx
6	RINIT	Состояние инициализации приемника: 1 – проинициализирован; 0 – не инициализировался
31:7	-	Всегда ноль

15.18 Регистр состояния передатчика порта связи (LTSTATx)

Регистры LTSTATx указывают статус передатчика порта связи. Значением сброса регистров LTSTATx является = 0x0000 0002. Подробное описание разрядов регистра приведено в таблице 132.

Таблица 132 – Регистр LTSTAT

Бит	Имя	Назначение
0	TVACANT	1 – буфер может принять данные; 0 – буфер не может принять данные
1	TEMPTY	1 – передатчик пуст; 0 – передатчик занят. Бит может быть использован при выключении передатчика.
2	TTER	1 – ошибка передачи time out; 0 – нет ошибки. Бит может быть сброшен чтением регистра LTSTATCx
3	TWER	1 – ошибка записи; 0 – нет ошибки. Активное значение было записано в регистр управления во время, когда передатчик был включен. Бит может быть сброшен чтением регистра LTSTATCx
31:4	-	Всегда ноль

15.19 Режим работы с синхронизацией по внешнему или внутреннему событию (для микросхем с ревизии 3)

При установленном бите RX_EXT_EN регистра LRCTL# управления приемником, поведение приемника меняется. В этом случае прием данных начинается с момента прихода внешнего или внутреннего события. Для задания внешнего события используется вход L0BCMPI, для внутреннего – бит 0 регистра CFG12. Выбранный источник события является общим для приемников обоих портов связи. Наличие переключений на входе сигнала синхронизации приемника является необходимым, но не достаточным условием для начала приема данных. При этом предполагается что входной сигнал синхронизации постоянно активен. Установка сигнала RX_EXT_EN должна быть выполнена перед включением приемника. На внешнем входе L0BCMPI должен быть низкий уровень (соответственно, CFG12[0] должен быть равен 0). После включения приемника происходит постоянный анализ входа источника события на предмет наличия на нем импульса высокого уровня. Импульс захватывается по положительному фронту сигнала синхронизации приемника. Временная диаграмма начала приема показана на рисунке 88 на примере внешнего события на L0BCMPI.

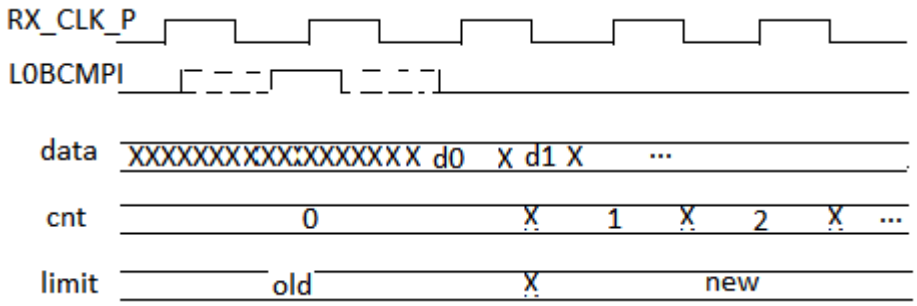


Рисунок 88. Временная диаграмма старта приёма по внешнему событию (импульс на L0BCMPI).

Как видно из рисунка, захват данных начинается по положительному фронту синхросигнала сразу за захватом импульса сигнала события. При этом для импульса сигнала события важен переход из 0 в 1. Счетчик cnt служит для подсчета числа тактов принимаемого квадрослова. Если все время держать сигнал события в 1, то прием данных начнется сразу после включения приемника и будет периодически повторяться. Каждый раз после приема запрограммированного в LIM_VAL# числа квадрослов, приемник инициализируется повторно. Каждый приемник имеет свой внутренний буферный регистр подсчета слов (limit на рисунке 88). Этот регистр меняет свое значение на новое в следующем такте после внешнего события.

После приема запрограммированного числа слов данных, приемник останавливает работу и ждет следующего импульса (высокого уровня на L0BCMPI). Временная диаграмма завершения приема приведена на рисунке 89.

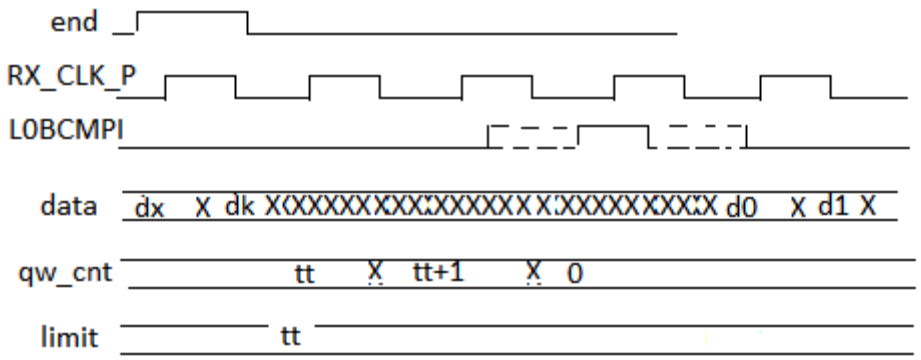


Рисунок 89. Временная диаграмма завершения приёма, инициированного по сигналу события (вход L0BCMPI).

Внутренний счетчик формирует сигнал окончания приема последней порции данных квадрослова (end). В следующем такте данный сигнал разрешает сравнение счетчика принятых квадрослов с регистром LIM_VAL# (limit на рисунке 89). В случае совпадения прием данных прекращается. Внутренний признак включения приема сбрасывается. В следующем (после сброса) такте он готов к анализу внешнего импульса начала приема.

После приема первого (за фронтом сигналом события) квадрослова данных приемник формирует признак первого квадрослова группы и передает этот признак вместе с данными в модуль UP\DOWN. Этот признак используется для инициализации модуля DOWN, т.е. модуль должен начать обработку новой группы данных с начальных условий, очистив конвейер от предыдущих данных.

Пользователь несет ответственность за корректность формирования сигнала события. Этот сигнал может захватываться не одновременно не только приемниками разных процессоров из-за разбежки синхросигнала по плате, но даже приемниками одного процессора из-за разных путей синхросигнала внутри процессора. Нужно обеспечивать достаточное время предустановки и удержания импульса относительно фронта синхросигнала. В процессоре предусмотрена возможность генерации импульса синхронизации, а также подбор его фазы относительно входного синхросигнала. Подробное описание этого механизма см. в подразделе 31.6 «Регистр синхронизации приемников портов связи CFG12».

15.20 Последовательность включения портов связи

Приемники и передатчики LVDS имеют время включения, равное 500 нс. Таким образом, после установки сигналов разрешения на приемник и передатчик запрещается осуществлять прием и передачу данных в течение 500 нс. Сигналы разрешения формируются только для линий данных и управляются битами TEN (регистр LTCTLx) и REN (регистр LRCTLx), соответственно. Сигналы разрешения для LVDS, отвечающих за линии синхросигнала, всегда включены в «1». При программировании портов связи необходимо убедиться, что последовательность их включения, а также отправка данных в передатчик LINK (DMA или ядром) учитывают эти времена.

Сигналы готовности передатчика LxACKI и приемника LxACKO не учитывают эти особенности. В системах, где для передачи данных по портам связи используются

другие микросхемы, для индикации готовности портов связи процессора могут быть использованы линии GPIO.

В силу особенностей примененных схемотехнических решений при работе с портами связи необходимо также учитывать, что после снятия внутреннего сброса по включению питания (сигнал por_b_33) разрешается работа аналоговой части передатчика LVDS для линий синхросигнала, при этом в этот момент на данных линиях возможно появление ложного импульса. В связи с этим необходимо обеспечить, чтобы в момент разрешения работы аналоговой части передатчика LVDS для линий синхросигнала внешний приемник LVDS был неактивен.

16 Последовательный асинхронный интерфейс UART

Микросхема ревизии 2 имеет два интерфейса UART, с ревизии 3 – четыре интерфейса. Интерфейс имеет гибкую систему программирования, позволяющую задать различные скорости обмена, длину посылки, контроль четности, а также различные ситуации прерываний. Интерфейс имеет встроенные FIFO глубиной 8 байт для приема и передачи. Возможности интерфейса позволяют ему работать с контроллером DMA.

Внешние выводы интерфейса UART приведены в таблице 133.

Таблица 133 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода для интерфейса UART	Тип вывода	Функциональное назначение
PA[0]	U0_TXD	O	Интерфейс UART0. Выход передатчика
PA[1]	U0_RXD	I	Интерфейс UART0. Вход приемника
PA[2]	U1_TXD	O	Интерфейс UART1. Выход передатчика
PA[3]	U1_RXD	I	Интерфейс UART1. Вход приемника
PB[27]	U2_TXD	O	Интерфейс UART2. Выход передатчика. Для коммутации этого вывода необходимо разрешить альтернативную функцию вывода PB[27] и работу блока UART2 в регистре CFG8 (для микросхем с ревизии 3)
PB[28]	U2_RXD	I	Интерфейс UART2. Вход приемника (для микросхем с ревизии 3)
PB[29]	U3_TXD	O	Интерфейс UART3. Выход передатчика Для коммутации этого вывода необходимо разрешить альтернативную функцию вывода PB[29] и работу блока UART3 в регистре CFG8 (для микросхем с ревизии 3)
PB[30]	U3_RXD	I	Интерфейс UART3. Вход приемника (для микросхем с ревизии 3)

Интерфейсы имеют различные базовые адреса в карте памяти процессора:

- UART_0 – **0x80000100**;
- UART_1 – **0x80000120**;
- UART_2 – **0x80000500** (для микросхем с ревизии 3);
- UART_3 – **0x80000520** (для микросхем с ревизии 3).

Каждый из интерфейсов имеет набор регистров, приведенный в таблице 134.

Таблица 134 – Регистры интерфейса

Номер регистра	Обозначение регистра	Описание	Состояние по сбросу
0	UARTDR	Регистр данных	-
1	URXSTAT	Регистр состояния приемника	0
2	UCR_load	Регистр управления загрузка	0
3	UCR_set	Регистр управления установка	0
4	UCR_clear	Регистр управления сброс	0
5	UBitRate	Регистр управления скоростью обмена	0
6	UFLAG	Регистр флагов	-
7	UINTM	Регистр запросов прерываний (маскированный)	0
8	UINT	Регистр запросов прерываний (немаскированный)	0

Условная структурная схема интерфейса приведена на рисунке 90. Регистры интерфейса позволяют пользователю задать требуемый режим работы: формат посылки, скорость обмена, источник прерываний и др. Рассмотрим более подробно назначение регистров интерфейса.

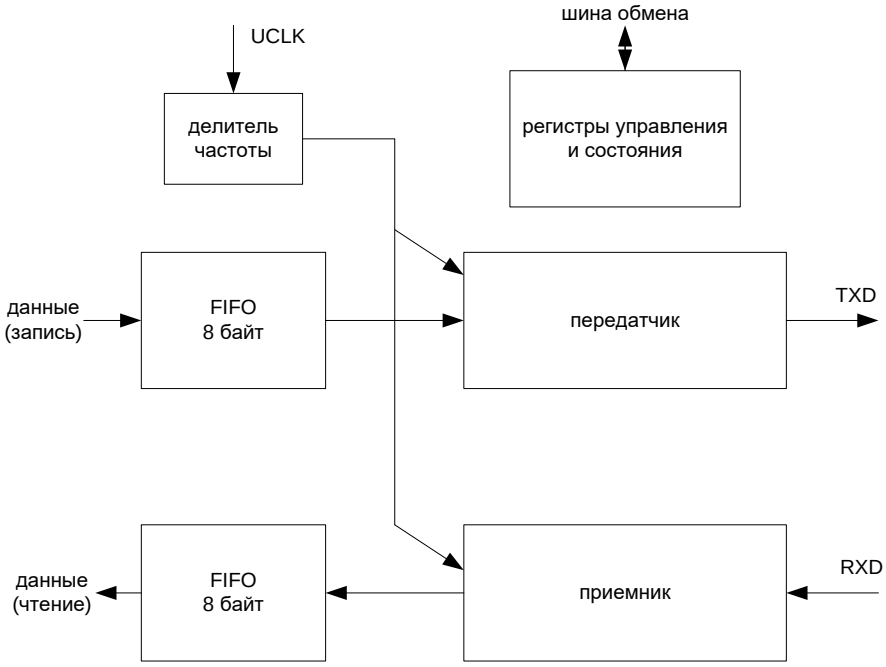


Рисунок 90 – Структурная схема UART

16.1 Регистр данных UARTDR

Регистр *UARTDR* представляет собой буфер данных передатчика при записи и буфер данных приемника при чтении. В качестве такого буфера используется FIFO емкостью восемь байт. Это позволяет выполнить единовременную загрузку 8 байт данных в буфер передатчика или обслуживать приемник только в случае заполнения буфера приемника не менее, чем наполовину. Это сокращает количество прерываний процессора на обслуживание интерфейса. Буфер приемника содержит не только байты

данных, но и соответствующие им признаки ошибок приема-передачи. Во время чтения байта данных, происходит копирование соответствующих ему признаков в регистр состояния RXSTAT. Несмотря на то, что интерфейс подключен к 32-разрядной шине, при обмене данными используются только восемь младших бит шины.

16.2 Регистр состояния RXSTAT

Таблица 135 – Регистр RXSTAT

Бит	Обозначение	Тип	Описание
0	FRAME	R	Ошибка в формате посылки: 0 – нет ошибки; 1 – ошибка
1	PARITY	R	Ошибка четности: 0 – нет ошибки; 1 – ошибка
2	OVERRUN	R	Переполнение буфера приемника: 0 – нет переполнения; 1 – переполнение. Бит очищается после чтения регистра
3	ERROR	R	Устанавливается в «1», если одна из выше описанных ошибок обнаруживается
7:4		-	Не используется

Бит FRAME устанавливается в «1», если во время приема не был обнаружен Стоп-бит, т.е. посылка не завершена корректно. Бит PARITY устанавливается в «1», если в принятом байте обнаружена ошибка четности\нечетности. Бит OVERRUN устанавливается в «1», если после приема очередного байта обнаруживается, что буфер приемника уже полон и нет места для записи принятого байта. В этом случае происходит потеря принятого байта и установка признака переполнения. Если после чтения текущего байта данных и его регистра состояния обнаруживается, что бит переполнения, установлен, то еще, по крайней мере, 7 байт данных в буфере являются достоверными.

16.3 Регистры управления скоростью обмена UBitRate

Интерфейс содержит в себе программируемый делитель частоты обмена разрядностью 16 бит.

Таблица 136 – Регистр UBitRate

Бит	Обозначение	Тип	Описание
15:0	BR	R/W	Биты делителя
21:16	BRF	R/W	Биты делителя (дробная часть) (для микросхем с ревизии 3)
31:22	-	-	Зарезервировано

Входной частотой интерфейса является внешний вход синхронизации OSCI. Внутри интерфейса она может дополнительно делиться на величину, заданную в регистре управления скоростью обмена, увеличенную на 1. Так, если значение равно 0x03,

делитель будет делить входную частоту на четыре. Эта частота будет использована приемником для сканирования принимающей линии. При этом предполагается, что длительность одного бита принимаемой посылки определяется битом UHBRE регистра UCR и может составлять четыре или 16 тактов базовой частоты. Таким образом, реальная скорость приема – передачи по интерфейсу равна

$$\frac{\text{OSCI}}{\text{UBitRate} + 1} \text{ / (16 или 4)} . \quad (4)$$

Приемник UART корректно работает, только если отклонение между заданной скоростью обмена и реальной скоростью внешнего передатчика не превышает допуск приемника UART, см. таблицу 137.

Таблица 137 – Допуск приемника UART при U_{CC} = 1,2 В, U_{CCIO} = U_{CCA} =3,3 В, T = 25 ºC

Количество стоп-бит	Отклонение скорости приемника относительно скорости внешнего передатчика, %			
	UHBRE = 0		UHBRE = 1	
	не менее	не более	не менее	не более
1	-4	6,3	0	6
2			-2,7	6

16.3.1 Для микросхем с ревизии 3

Коэффициент деления для формирования скорости передачи данных состоит из 22 бит, при этом 16 бит выделено для представления его целой части, а 6 бит – дробной части. Возможность задания нецелых коэффициентов деления позволяет осуществлять обмен данными со стандартными информационными скоростями, при этом используя в качестве UARTCLK тактовый сигнал с произвольной частотой более 3,6864 МГц.

Целая часть коэффициента деления записывается в 16-битный регистр BR. Шестиразрядная дробная часть записывается в регистр BRF. Значение коэффициента деления связано с содержимым указанных регистров следующим образом

$$\text{Коэффициент деления} = \frac{\text{OSCI}}{16 \cdot \text{Скорость передачи данных}} = \text{BR} + \text{BRF} , \quad (5)$$

где BR – целая часть коэффициента деления;
BRF – дробная часть коэффициента деления.

Шестибитное значение, записываемое в регистр BRF, вычисляется путем выделения дробной части требуемого коэффициента деления, умножения ее на 64 и округления до ближайшего целого числа:

$$\text{M} = \text{integer}(\text{BRF} \cdot 64 + 0{,}5) , \quad (6)$$

где integer – операция отсечения дробной части числа.

16.4 Регистр управления интерфейсом UCR

Таблица 138 – Регистр UCR

Бит	Обозначение	Тип	Описание
0	UARTEN	R/W	Разрешение работы интерфейса UART: 0 – выключен; 1 – включен
1	UTXDIS	R/W	Управление передатчиком: 0 – включен; 1 – выключен
2	URXDIS	R/W	Управление приемником: 0 – включен; 1 – выключен
3	DMAONERR	R/W	Управление запросом к контроллеру DMA во время обнаружения ошибки приема: 0 – обнаружение ошибки не влияет формирование запроса к контроллеру DMA на обслуживание; 1 – формирование запроса к контроллеру DMA на обслуживание запрещено при обнаружении ошибки
4	UTXFDIS	R/W	Управление буфером передатчика: 0 – FIFO включено; 1 – FIFO выключено
5	URXFDIS	R/W	Управление буфером приемника: 0 – FIFO включено; 1 – FIFO выключено
7:6	UHBRE	R/W	Управление длительностью одного бита посылки: 00 – 1/16 clock; 01 – 1/4 clock; 1x – 1/4 clock, выключение внутреннего делителя
8	BREAK	R/W	Если равен 1, линия TXD передатчика устанавливается в высокий уровень
9	PRTEN	R/W	Управление проверкой и формированием бита четности: 0 – запрещено; 1 – разрешено
10	EVENPRT	R/W	Выбор проверки на четность\нечетность: 0 – дополнение посылки до четного; 1 – дополнение посылки до нечетного
11	XSTOP	R/W	Количество Стоп-битов (передача): 0 – один стоп-бит; 1 – два стоп-бита. приемник не проверяет количество принятых стоп-бит.
12	UFIFOEN	R/W	Управление буфером FIFO 0 – выключено (глубина буфера равна 1 слову); 1 – включение FIFO приемника и передатчика

Бит	Обозначение	Тип	Описание
14:13	WRDLEN	R/W	Выбор длины передаваемых данных 00 – 8 бит; 01 – 7 бит; 10 – 6 бит; 11 – 5 бит
15	INV	R/W	1 – инверсия TXD линии
16	TXINT	R/W	Управление разрешением прерывания от флага TXINT: 0 – запрещено; 1 – разрешено
17	RXINT	R/W	Управление разрешением прерывания от флага RXINT: 0 – запрещено; 1 – разрешено
18	RXERRINT	R/W	Управление разрешением прерывания от флага RXERRINT: 0 – запрещено; 1 – разрешено
19	MSINT	R/W	Управление разрешением прерывания от флага MSINT: 0 – запрещено; 1 – разрешено
20	UDINT	R/W	Управление разрешением прерывания от флага UDINT: 0 – запрещено; 1 – разрешено
21	UTXEINT	R/W	Управление разрешением прерывания от флага UTXEINT: 0 – запрещено; 1 – разрешено
22	URXTINT	R/W	Управление разрешением прерывания от флага URXTINT: 0 – запрещено; 1 – разрешено
30:23	RSV	-	Не используется
31	LBM	R/W	Тестовый режим работы интерфейса при котором выход передатчика коммутируется на вход приемника (когда ==1)

Бит UARTEN разрешает работу интерфейса. Он используется одновременно с битами UTXDIS и URXDIS, которые индивидуально управляют передатчиком и приемником соответственно.

Бит DMAONERR позволяет управлять интерфейсом во время его работы с контроллером DMA. В случае если при приеме данных возникнет ошибка, то при установленном бите DMAONERR произойдет блокировка запроса к контроллеру DMA, что исключит прием ошибочных данных.

Бит UTXFDIS используется только для управления выключением FIFO передатчика. При установленном бите FIFO выключен.

Бит URXFDIS используется только для управления выключением FIFO приемника. При установленном бите FIFO выключен.

Биты UHBRE определяют количество тактов частоты синхронизации (полученной после деления входной частоты блока UART на значение UBRCR)

необходимых для приема (передачи) одного бита данных. Значение 00 задает режим, в котором длительность одного бита передаваемой (принимаемой) информации равна 16 периодам частоты внутреннего делителя. Этот режим обеспечивает надежный прием в условиях помех. Значения 01, 10, 11 задают режим, в котором длительность одного бита передаваемой (принимаемой) информации равна четырем периодам частоты внутреннего делителя. В случае значений 10 или 11, значение коэффициента внутреннего делителя частоты равно единице вне зависимости от значения в регистре ***UBitRate***.

Таким образом, реальная скорость обмена интерфейса UART определяется как входная частота OSCI, деленная на запрограммированный коэффициент в регистре ***UBitRate***, затем деленная на 16 или четыре в передатчике (приемнике) интерфейса. Максимальная скорость обмена составляет OSCI /4.

Бит BREAK. Установка данного бита приводит к установке высокого уровня на линии TXD передатчика.

Бит PRTEN разрешает (если равен единице) автоматическую генерацию дополнительного бита паритета при передаче байта данных. При приеме, установка этого бита разрешает проверку паритета принятого байта данных и генерацию флага об ошибке.

Бит EVENPRT позволяет задать тип проверки: четное или нечетное количество единичных бит в посылке. Если этот бит равен нулю, дополнительный бит четности дополняет количество единичных бит в посылке до четного количества. К примеру, если передается значение 0x15, бит четности равен единице. Если передается значение 0x55, то бит четности равен 0. Соответственно, если значение бита EVENPRT равно единице, осуществляется дополнение до нечетного количества единичных бит.

Бит XSTOP позволяет задать количество Стоп-битов в одиночной посылке. Значение «0» соответствует одному стоп-биту, а значение «1» – двум стоп-битам. Этот бит используется только передатчиком. Приемник не проверяет количество принятых стоп-битов больше одного. Увеличение количества стоп-битов до двух иногда полезно в случаях, когда существует расхождение в скорости обмена передающей и принимающей сторон.

Бит UFIFOEN позволяет управлять буферами приемника и передатчика. Если этот бит, равен 0, флаги состояния буферов формируются так, как будто размер буфера равен одному байту. При установленном бите, размер буфера равен 8.

Биты WDRLEN позволяют запрограммировать количество реальных информационных бит в посылке. Таким образом, посылка может быть длиной от семи бит (старт, пять бит данных, стоп) до 12 бит (старт, восемь бит данных, бит четности и два стоповых).

16.5 Регистр состояния интерфейса UFLAG

Таблица 139 – Регистр UFLAG

Бит	Обозначение	Тип	Описание
2:0			
3	UBUSY	R	Передатчик занят. Бит равен единице во время передачи данных, включая стоп-биты

Бит	Обозначение	Тип	Описание
4	URXFE	R	Состояние буфера приемника: 0 – буфер приемника хранит данные; 1 – буфер приемника пуст
5	UTXFF	R	Состояние буфера передатчика: 0 – буфер передатчика может принять новые данные; 1 – буфер передатчика заполнен полностью
7:6			

Бит UBUSY показывает состояние передатчика в текущий момент времени. Единичное значение бита указывает на процесс передачи информации. Флаг может быть использован для определения момента, когда передатчик полностью закончил работу.

Бит URXFE является признаком состояния FIFO приемника. Значение флага не зависит от того включено FIFO или нет. Единичное значение флага указывает на то, что буфер приемника пуст.

Бит UTXFF является признаком состояния FIFO передатчика. Значение флага зависит от того включено FIFO или нет. Если FIFO выключено, единичное значение флага указывает на то, что в буфере передатчика имеется байт данных для передачи. Если FIFO включено, единичное значение флага указывает на то, что буфер передатчика заполнен полностью.

16.6 Регистры управления прерываниями UINT, UINTM

Интерфейс имеет регистр состояния запросов прерывания UINT. Каждый бит регистра состояния имеет соответствующий бит разрешения прерывания. Каждый из интерфейсов системы имеет только один вход запроса прерывания в контроллере прерываний. Поэтому на программиста возлагается задача определения, какая именно ситуация вызвала запрос на прерывание.

Таблица 140 – Регистр UINT

Бит	Обозначение	Тип	Описание
0	TXINT	R	Запрос прерывания от передатчика
1	RXINT	R	Запрос прерывания от приемника
2	RXERRINT	R	Ошибка приема во время работы с контроллером DMA
3	MSINT	R/W	Запрос прерывания от модема
4	UDINT	R	Запрос прерывания от приемника в состоянии «выключен»
5	UTXEINT	R/W	Запрос прерывания от буфера передатчика при отсутствии в нем данных
6	URXTINT	R/W	Запрос прерывания от приемника при обнаружении ситуации «time-out»
7	RSV	-	Не используется

Бит TXINT – запрос прерывания от передатчика. Он отражает состояние FIFO передатчика. Если FIFO выключено, запрос на прерывание активен (равен 1) если буфер пуст. Если же FIFO включено, запрос на прерывание активен, если буфер заполнен на половину или менее.

Бит RXINT – запрос прерывания от приемника. Запрос может быть активен в следующих случаях: FIFO приемника выключено и в нем имеются принятые данные, FIFO приемника включено и заполнено принятыми данными на половину или больше, FIFO приемника содержит данные и не поступило новых данных за время более чем период, равный передаче 32 бит данных.

Бит RXERRINT отражает информацию о возможных ошибках при приеме данных. Наиболее эффективно данный бит может быть использован при работе интерфейса с контроллером DMA. В этом случае, если контроллер DMA прочитает данные, при приеме которых были обнаружены ошибки, то интерфейс автоматически выработает прерывание, что позволит процессору правильно обработать возникшую ситуацию. Данный бит автоматически блокирует генерацию запроса к контроллеру DMA на обслуживание (при разрешении в регистре управления).

Бит MSINT отражает запрос на прерывание от линий состояния модема. Изменение состояния любого из входов DCD, DSR, CTS вызывает установку бита MSINT.

Бит UDINT позволяет определить ситуацию, когда приемник интерфейса выключен, но на линии приемника RXD обнаружен старт бит. Это приводит к установке запроса на прерывание.

Бит UTXEINT активен в случае, когда интерфейс включен и буфер передатчика пуст.

Бит URXTINT отражает наличие ситуации в приемнике, когда нет приема за период равный передаче 32 бит данных. Этот флаг может быть использован для определения ситуации конца пакета данных.

Биты регистра состояния, помеченные как R/W, могут быть сброшены программно. Это осуществляется записью «1» в соответствующий бит регистра состояния. Запись нуля не изменяет значения бита.

Необходимо отметить следующую деталь – при чтении регистра UINTM происходит чтение значения регистра состояния, маскированное значением разрешения. Для чтения оригинального значения регистра состояния используется регистр UINT.

16.7 Программирование интерфейса UART

Перед началом работы с интерфейсом UART необходимо разрешить альтернативную функцию для соответствующих битов портов общего назначения. Необходимо заранее установить требуемую скорость обмена и значения соответствующих коэффициентов деления в конкретном интерфейсе. После программирования скорости обмена необходимо задать другие параметры обмена (длину посылки, контроль ошибок, ситуации прерываний) и включить интерфейс. Интерфейсы UART_0, UART_1 могут работать с контроллером DMA.

17 Последовательный синхронный интерфейс SPI

Интерфейс включает сигнал синхронизации, разрешения обмена, вход и выход данных. Он имеет гибкую систему программирования, позволяющую задать различные скорости обмена, длину посылки, а также различные ситуации прерываний. Интерфейс имеет встроенные FIFO глубиной 8 байт для приема и передачи. Возможности интерфейса позволяют ему работать с контроллером DMA.

Внешние выводы интерфейса SPI приведены в таблице 141

Таблица 141 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода для интерфейса SPI	Тип вывода	Функциональное назначение
PA[4]	SPI0_CLK	I/O	Интерфейс SPI0. Синхросигнал
PA[5]	SPI0_DO	I/O	Интерфейс SPI0. Выход данных
PA[6]	SPI0_DI	I/O	Интерфейс SPI0. Вход данных
PA[7]	SPI0_CS[0]	I/O	Интерфейс SPI0. Выбор SPI устройства 0
PA[8]	SPI0_CS[1]	O	Интерфейс SPI0. Выбор SPI устройства 1
PA[9]	SPI0_CS[2]	O	Интерфейс SPI0. Выбор SPI устройства 2
PA[10]	SPI0_CS[3]	O	Интерфейс SPI0. Выбор SPI устройства 3
PA[11]	SPI0_CS[4]	O	Интерфейс SPI0. Выбор SPI устройства 4
PA[12]	SPI0_CS[5]	O	Интерфейс SPI0. Выбор SPI устройства 5
PA[13]	SPI1_CLK	I/O	Интерфейс SPI1. Синхросигнал
PA[14]	SPI1_DO	I/O	Интерфейс SPI1. Выход данных
PA[15]	SPI1_DI	I/O	Интерфейс SPI1. Вход данных
PA[16]	SPI1_CS	I/O	Интерфейс SPI1. Выбор SPI устройства
PB[0]	SPI2_CLK	I/O	Интерфейс SPI2. Синхросигнал
PB[1]	SPI2_DO	I/O	Интерфейс SPI2. Выход данных
PB[2]	SPI2_DI	I/O	Интерфейс SPI2. Вход данных
PB[3]	SPI2_CS	I/O	Интерфейс SPI2. Выбор SPI устройства

Для обмена посредством интерфейса SPI используются четыре линии: синхронизация, вход данных, выход данных, разрешение обмена. В ряде случаев используется еще дополнительная линия выбора мастера-подчиненного, однако, в данной реализации такой выбор осуществляется посредством программирования регистров управления. Направление линий интерфейса, в зависимости от выбранного режима работы, приведено в таблице 142.

Таблица 142 – Режимы работы интерфейса

	SPI (Ведущий)	SPI (Ведомый)
SPI_CLK	Выход	Вход
SPI_DO	Выход	Вход
SPI_DI	Вход	Выход
SPI_CS	Выход	Вход

Интерфейс доступен для программирования как набор регистров, позволяющих задать нужную конфигурацию и осуществить прием-передачу данных. Регистры интерфейса подключены к шине обмена периферийных устройств. В микросхеме реализовано три независимых блока SPI, характеристики и базовые адреса которых приведены в таблице 143

Таблица 143 – Описание

Блок	Базовый адрес регистра	Описание
SPI0	0x80000140	До шести одновременно адресуемых устройств в режиме «Ведущий», одно ведущее устройство в режиме «Ведомый»
SPI1	0x80000360	Одно устройство в режимах «Ведущий», «Ведомый»
SPI2	0x80000380	Одно устройство в режимах «Ведущий», «Ведомый»

Краткая информация о регистрах приведена в таблице 144.

Таблица 144 – Регистры интерфейса

Номер	Имя	Назначение	По сбросу
0	SPCR0	Регистр управления 0	0
1	SPCR1	Регистр управления 1	0
2	SPDR	Регистр данных	-
3	SPSR	Регистр состояния	0
4	RX_CNT	Счетчик приемника	0

Алгоритм работы интерфейса предполагает одновременный прием и передачу данных. Интерфейс имеет встроенные буфера (FIFO) для приема и передачи данных. Размер каждого из буферов составляет восемь 32-разрядных слов. Для того чтобы осуществить обмен посредством интерфейса, необходимо:

– запрограммировать параметры обмена в регистрах управления;
– записать данные для передачи в буфер передатчика;
– прочитать принятые данные из буфера приемника.

Если необходимо только передать данные в какое-то устройство, интерфейс все равно будет принимать данные с входной линии данных. Поэтому после окончания передачи необходимо сбросить флаги состояния приемника.

Если необходимо только принять информацию, в этом случае нужно записать в буфер передатчика фиктивные данные для передачи. В этом случае количество принятых данных будет соответствовать количеству переданных.

Интерфейс SPI0 имеет возможность подключать до 6-ти внешних устройств. Каждое из устройств имеет общие линии синхронизации и данных, но индивидуальную линию выбора устройства. Номер устройства задается в регистре управления и должен быть неизменным на весь период обмена. Имеется возможность индивидуального задания неактивного уровня для каждого устройства.

Интерфейсы SPI1, SPI2 имеют возможность работы только с одним устройством, во всем остальном абсолютно идентичны интерфейсу SPI0.

Если возникает необходимость работы интерфейса в режиме «подчиненный», ведущее устройство должно осуществлять выборку интерфейса посредством входа CS1. Также если выбран вариант начальной загрузки с использованием SPI-флэш памяти, данная память должна быть подключена к CS0.

Практически количество подключаемых внешних SPI-устройств может быть и больше чем шесть. В этом случае в качестве сигнала выбора устройства можно использовать любой из свободных выходов общего назначения, а в интерфейсе при осуществлении обмена задавать номер устройства 7.

Интерфейс имеет два 32-разрядные регистра управления, позволяющие задать параметры работы интерфейса. Ниже, в таблицах, приведено краткое описание разрядов регистров управления.

17.1 Регистр SPCR0

Таблица 145 – Регистр управления SPCR0

Бит	Имя	Тип	Назначение
4:0	DSS	R/W	Выбор размера данных: 0-2 – Не используется; 3-31 – длина данных (DSS+1)
5	-		
6	SPO	R/W	Полярность SPI_CLK: 0 – SPI_CLK высокий импульс; 1 – SPI_CLK низкий импульс
7	SPH	R/W	Фаза SPI_CLK: 0 – SPI_CLK активен на срезе; 1 – SPI_CLK активен на фронте
8	TWI_on	R/W	0 – стандартный SPI интерфейс; 1 – трехпроводной интерфейс с двунаправленной линией
9	TWI_rw	R/W	Чтение(0) или запись(1) в режиме TWI
10	LMSF	R/W	Выбор: 0 – старший бит первый; 1 – младший бит первый
11	-		
14:12	CSN		Выбор номера SPI-устройства
15	-		
27:16	SCR	R/W	Коэффициент деления частоты (не может быть равным 0). Скорость обмена = SOC_CLK / (SCR + 1)
31:27	-		

Биты DSS задают количество передаваемых бит в одной посылке. Максимальное количество бит в одной посылке равно 32. Если для работы интерфейса выбрана меньшая разрядность передаваемых данных (от четырех до 32), для передачи используются только младшие значащие биты. Так, если размер одиночной посылки равен восьми битам, старшие биты (с восьмого по 32) слова не будут передаваться.

Биты SPO и SPH управляют полярностью и фазой синхросигнала SPI_CLK. Возможные варианты обмена при передаче четырех бит данных показаны на рисунке 91.

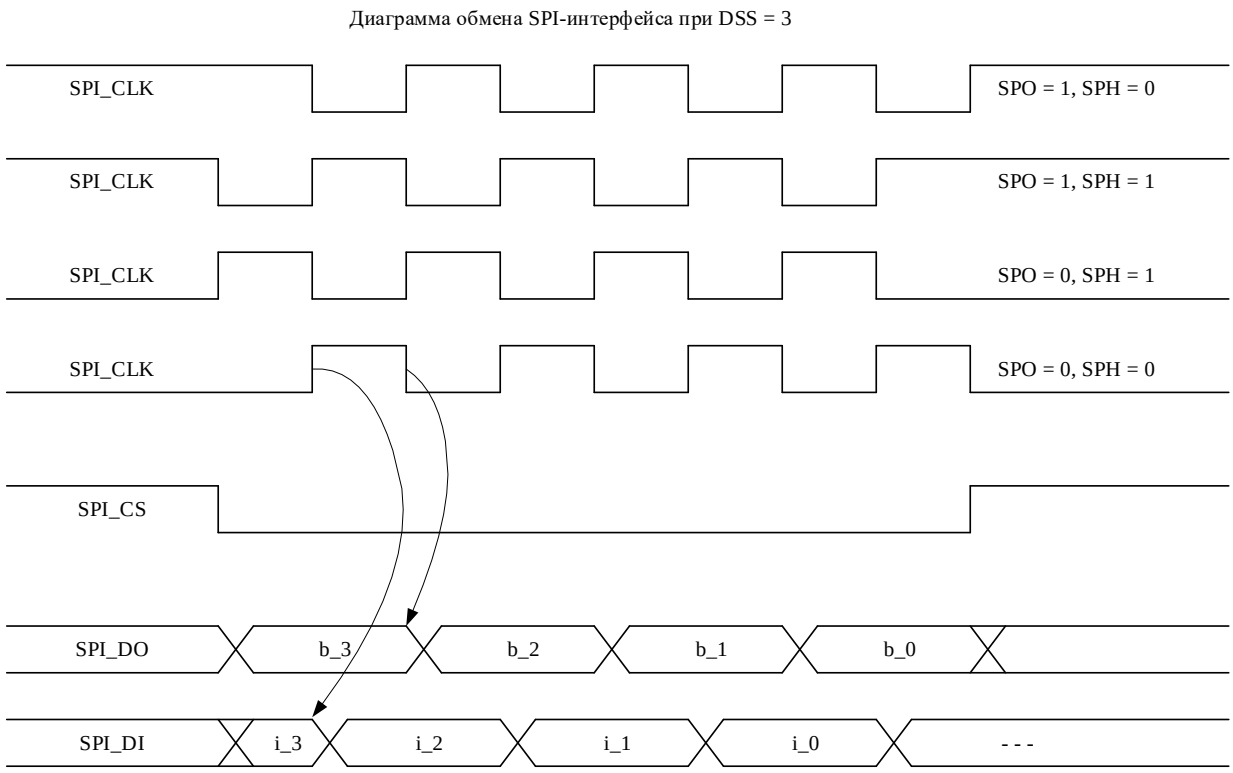


Рисунок 91 – Диаграмма обмена в режиме master

Биты SCR задают коэффициент деления для внутреннего делителя частоты. Входная частота синхронизации модуля SPI поступает с периферийной шины, затем она может дополнительно делиться на запрограммированный коэффициент деления согласно формуле

$$\text{SPI\_CLK} = \frac{\text{SOC\_CLK}}{\text{SCR} + 1}. \quad (7)$$

Примечание – Значение SCR не может быть равным 0.

17.2 Регистр SPCR1

Таблица 146 – Регистр управления SPCR1

Бит	Имя	Тип	Назначение
0	RIM	R/W	Управление прерыванием от приемника интерфейса: 0 – запрещено; 1 – разрешено
1	TIM	R/W	Управление прерыванием от передатчика интерфейс: 0 – запрещено; 1 – разрешено
2	LBM	R/W	Разрешение тестового режима работы: 0 – обычный режим работы; 1 – тестовый режим работы

Бит	Имя	Тип	Назначение
3	SPE	R/W	Разрешение работы интерфейса 0 – выключен; 1 – включен
4	MS	R/W	Режим работы: 0 – мастер; 1 – подчиненный
5	TUM	R/W	Разрешение прерывания по исчерпанию буфера передатчика при работе в режиме подчиненного: 0 – запрещено; 1 – разрешено
6	ROM	R/W	Разрешение прерывания при переполнении буфера приемника: 0 – запрещено; 1 – разрешено
7	-	R/W	
8	R_RQM	R/W	Запрос на обслуживание со стороны FIFO приемника: 0 – FIFO загружено на половину или более; 1 – в FIFO имеются данные
9	T_RQM	R/W	Запрос на обслуживание со стороны FIFO передатчика: 0 – FIFO загружено на половину или менее; 1 – FIFO не заполнено полностью
10	TXO	R/W	Только передача: 0 – одновременный прием-передача; 1 – только передача
11	RXO	R/W	Только прием: 0 – одновременный прием-передача; 1 – только прием
12	CIM	R/W	Разрешение прерывания от счетчика приемника: 0 – запрещено; 1 – разрешено
13			
14	HOLDCS	R/W	Удержание линии CS активной после завершения обмена: 0 – после обмена линия становится неактивной (работает совместно с полем DLY_T); 1 – после очередного обмена линия остается активной.
15	ROTL		Выходной уровень на выходе данных в случае работы в режиме «только прием»
21:16	CSAL	R/W	Выбор активного уровня на выходе CSi: 0 – активный низкий; 1 – активный высокий
23:22	-		Не используются
27:24	DLY_T	R/W	Длительность неактивного уровня CS (при HOLDCS=0): 0 – неактивного уровня нет; Остальные значения – DLY_T тактов синхросигнала SPI
31:28	-		Не используются

Бит SPE разрешает работу интерфейса. Если его значение равно нулю – интерфейс выключен. Для включения интерфейса необходимо записать в данные бит значение «1».

Бит MS при включенном интерфейсе определяет режим работы интерфейса – мастер или подчиненный. В случае работы в режиме мастер (MS = 0) интерфейс самостоятельно вырабатывает сигнал синхронизации и сигнал разрешения обмена. В случае работы в режиме «подчиненный» (MS = 1), интерфейс работает под управлением внешнего синхросигнала и от внешнего сигнала разрешения работы. При работе в режиме подчиненный имеются некоторые ограничения на максимальную частоту обмена. Она должна быть как минимум в три раза ниже частоты обмена по внутренней шине периферийных устройств.

Бит RIM управляет разрешением прерывания от приемника интерфейса. Момент генерации прерывания зависит от бита R_RQM. Если бит R_RQM равен единице, то прерывание будет сформировано, если в FIFO приемника есть данные. Если бит R_RQM равен нулю, то прерывание будет сформировано, если в FIFO приемника есть четыре или более слова данных, либо если в количество данных меньше четырех, но прием данных закончен. Факт окончания приема определяется по факту отсутствия данных в буфере передатчика и отсутствии в данный момент обмена.

Бит TIM управляет разрешением прерывания от передатчика интерфейса. Момент генерации прерывания зависит от бита T_RQM. Если бит T_RQM равен единице, то прерывание будет сформировано, если FIFO передатчика заполнено не полностью. Если бит T_RQM равен 0, то прерывание будет сформировано, если в FIFO передатчика есть 4 или более свободные ячейки для записи данных.

Бит LBM позволяет перевести интерфейс в тестовый режим работы (LBM = 1), в котором выход данных передатчика подается на вход данных приемника. Это позволяет произвести проверку оборудования без подключения внешнего устройства. Для нормального режима работы необходимо устанавливать LBM = 0.

Бит TXO позволяет, при необходимости, упростить передачу информации. Как уже отмечалось выше, при обычной работе интерфейс при передаче данных осуществляет одновременный прием информации с линии входа данных. Порой эти данные не нужны, т.к. требуется выполнить только передачу. Но буфер приемника все равно заполняется. Чтобы исключить это неудобство был добавлен бит TXO, который при значении «1» выполняет блокировку приема данных и не изменяет состояние приемника.

Бит RXO позволяет, при необходимости, упростить прием информации. Как уже отмечалось выше, при обычной работе интерфейс для приема данных должен осуществить фиктивную передачу информации. Чтобы исключить это неудобство был добавлен бит RXO, который при значении «1» не требует наличия фиктивных данных в буфере передатчика при приеме данных. Для обеспечения функции «только прием» был добавлен дополнительный регистр RX_CNT.

Бит CIM разрешает генерацию запроса прерывания при значении счетчика RX_CNT равным нулю.

Бит CSAL позволяет задать активный уровень на выходе CS. Одни устройства могут активизироваться высоким уровнем, а другие – низким. Данный бит дает возможность задать активный уровень.

Бит HOLDCS позволяет осуществить удержание вывода CS в активном состоянии после завершения обмена. Бит должен быть установлен до начала обмена. После такого, как с началом обмена линия CS активизируется, она будет оставаться активной до тех пор, пока бит HOLDCS не будет сброшен. Сбросить бит можно во время осуществления последнего обмена или после его завершения. Использование данной функции позволяет осуществить передачи или прием посылки произвольной длительности.

17.3 Регистр счета принимаемых данных RX_CNT

Регистр используется только при установленным бите RXO == 1, что соответствует режиму «только прием». В этом случае в регистр RX_CNT необходимо записать количество принимаемых слов данных (не более 4095). Если значение регистра не равно нулю, это автоматически формируется запрос на прием данных. После приема очередного слова осуществляется вычитание единицы из счетчика. Возможна генерация прерывания от счетчика в случае достижения им нулевого значения. Это происходит при установленном бите CIM.

17.4 Регистр данных SPDR

Регистр данных имеет разрядность 32 бит. При записи, данные попадают в верхний свободный регистр FIFO передатчика, а при чтении, считываются из нижнего регистра FIFO приемника. Размер FIFO приемника и передатчика составляет по восемь 32-разрядных слов каждый. Каждый из FIFO имеет указатели считывания и записи. Интерфейс имеет несколько флагов, которые отражают состояние FIFO передатчика и приемника. Интерфейс подключен к 32-разрядной шине периферийных устройств.

17.5 Регистр состояния SPSR

Таблица 147 – Регистр состояния SPSR

Бит	Имя	Тип	Назначение
0	TFE	R	Флаг заполнения FIFO передатчика («пустой»): 0 – в FIFO передатчика данные имеются; 1 – в FIFO передатчика данные отсутствуют
1	TNF	R	Флаг заполнения FIFO передатчика («полный»): 0 - FIFO передатчика заполнено полностью; 1 – FIFO передатчика не заполнено
2	RNE	R	Флаг пустого FIFO приемника («не пустой»): 0 – нет данных в FIFO приемника; 1 – в FIFO приемника имеются данные
3	BSY	R	Занятость интерфейса SPI: 0 – простаивает или выключен; 1 – занят передачей-приемом

Бит	Имя	Тип	Назначение
4	TFS	R	Флаг состояния FIFO передатчика: 0 – буфер передатчика более чем на половину заполнен (5 или более слов записано); 1 – буфер передатчика заполнен на половину или менее (4 или меньше слов записано).
5	RFS	R	Флаг состояния FIFO приемника: 0 – буфер приемника заполнен меньше чем на половину (3 или меньше слов записано); 1 – буфер приемника заполнен на половину или более (4 или больше слов записано)
6	ROR	R/W	Флаг переполнения FIFO приемника: 0 – нет ошибки; 1 – во время приема произошло переполнение буфера
7	TUR	R/W	Флаг состояния FIFO передатчика (только в режиме «подчиненный») («чтение пустого буфера»): 0 – нет ошибки; 1 – передатчик пытался передать данные из пустого буфера
8	RFF	R	Флаг состояния FIFO приемника («полный»): 0 – буфер приемника заполнен не полностью; 1 – буфер приемника заполнен полностью
9	TI_REQ	R	Запрос передатчика к процессору или DMA: 0 – неактивный уровень; 1 – активный уровень
10	RI_REQ	R	Запрос приемника к процессору или DMA: 0 – неактивный уровень; 1 – активный уровень
31:11	RSV	-	Не используются

Биты регистра SPSR отражают состояние интерфейса во время приема-передачи данных.

Бит TFE отражает наличие данных в буфере передатчика. Значение бита 1 соответствует ситуации, когда буфер пуст. Бит может быть использован для отслеживания ситуации, когда интерфейс закончил передачу.

Бит TNF может быть использован при заполнении буфера передатчика данными. Значение бита TNF=1 означает, что буфер может принять новые данные. Значение бита равное нулю означает что, буфер заполнен полностью.

Бит RNE отражает состояние буфера приемника. Значение бита RNE=1, означает наличие данных в буфере приемника, которые могут быть прочитаны.

Бит BSY отражает занятость интерфейса приемом\передачей данных. Бит может быть использован для определения момента полного завершения работы интерфейса. Так отсутствие данных в FIFO передатчика и равенство бита BSY нулю, означает завершение работы интерфейса.

Бит TFS отражает состояние буфера передатчика. Равенство бита единице означает, что FIFO передатчика заполнено на половину или менее (т.е. в FIFO записано от одного до четырех слов).

Бит RFS отражает состояние буфера приемника. Равенство бита единице означает, что FIFO приемника заполнено на половину или более (т.е. в FIFO записано от четырех до восьми слов).

Бит ROR отслеживает ситуацию переполнения FIFO приемника, т.е. случая, когда в буфер приемника была произведена попытка записи новых данных в то время, когда он был заполнен. Бит может вызвать запрос прерывания процессору, если соответствующий ему бит ROM регистра управления установлен.

Бит TUR имеет смысл при работе интерфейса в режиме подчиненного и отслеживает ситуацию, когда при запросе обмена со стороны мастера, буфер передатчика пуст. При работе в режиме подчиненного обмен инициирует внешний мастер, а данные должны быть загружены в буфер предварительно. Бит может вызвать запрос прерывания процессору, если соответствующий ему бит TUM регистра управления установлен.

Биты ROR и TUR устанавливаются аппаратно, но могут быть сброшены программно. Для очистки бита необходимо произвести запись в соответствующий бит значения «1».

Бит RFF отслеживает ситуацию, когда буфер приемника заполнен полностью.

17.6 Особенности работы в режиме “slave”

Интерфейс имеет возможность работы в режиме подчиненного («slave») устройства. Данный режим задается при помощи установки в «1» бита MS регистра SPCR1. В данном режиме интерфейс использует вход SPI_CLK (PA[4]) как вход синхронизации, вход SPI_CS[1] (PA[8]) как вход выборки интерфейса, вход SPI_DO (PA[5]) как вход данных для приема информации и выход SPI_DI (PA[6]) для передачи данных ведущему устройству. В отличие от режима мастера, все внешние контакты интерфейса меняют направление на противоположное. Работа в режиме подчиненного устройства имеет ряд ограничений. Вход синхронизации SPI_CLK используется, так как он есть на внешнем выводе, т.е. нет никаких преобразований полярности и фазы сигнала синхронизации. Выдача данных на SPI_DI осуществляется только по срезу (из «1» в «0») SPI_CLK. Прием данных с SPI_DO выполняется только по фронту (из «0» в «1») SPI_CLK. Пример временной диаграммы представлен на рисунке 92.

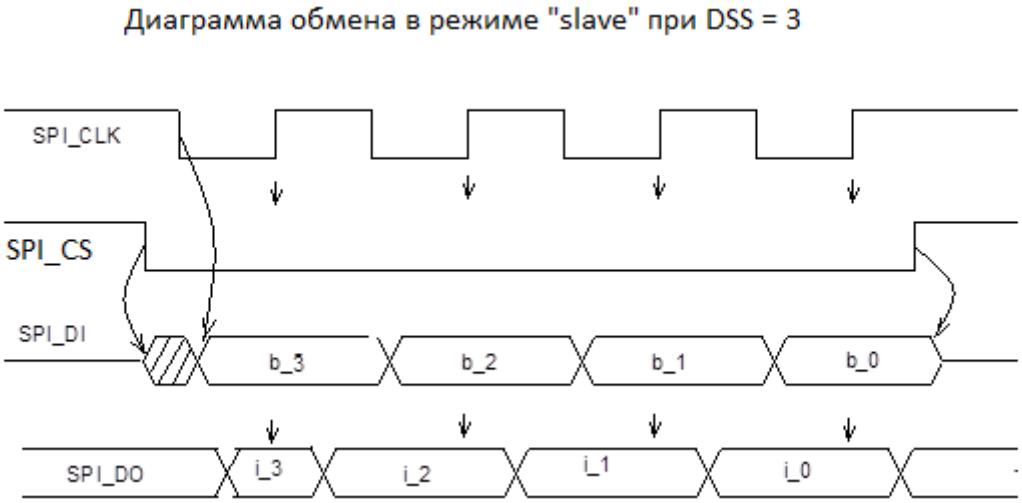


Рисунок 92 – Диаграмма обмена в режиме slave

Как следует из рисунка, выход SPI_DI становится активным, как только на входе SPI_CS появляется активный уровень. Активный уровень на входе SPI_CS может программироваться. Если бит CSAL[1] равен нулю, активным уровнем на входе SPI_CS будет низкий уровень. Если CSAL[1] равен единице – активный уровень высокий. Однако установка активного уровня на SPI_CS не означает, что на выходе SPI_DI сразу же появляются истинные данные. Только после первого среза на SPI_CLK на выходе SPI_DI появится достоверная информация (бит b_3). Ведущее устройство должно принять данные с линии SPI_DI по фронту SPI_CLK. В этот же момент и подчиненное устройство примет данные с линии SPI_DO.

В момент приема бита b_0 подчиненное устройство зафиксирует окончание приема посылки и передаст принятые данные в FIFO приемника. При передаче данных в FIFO будет выполнена передача из домена синхронизации SPI_CLK в домен синхронизации SOC_CLK. Для корректной работы интерфейса необходимо, чтобы частота SOC-шины была как минимум в три раза выше частоты SPI_CLK.

Поскольку момент начала передачи по интерфейсу в режиме подчиненного является случайным, данные для передачи должны быть записаны до момента начала передачи. В противном случае будет сформирован флаг ошибки. Моментом захвата данных из FIFO передатчика является первый фронт сигнала SPI_CLK. К этому моменту данные для передачи должны быть записаны в FIFO. Сброс указателя на взятые из FIFO данные выполняется немного позже. Возможна ситуация, когда данные захвачены некорректно, но флаг ошибки не установится. Это может случиться, если одновременно с записью данных в FIFO приходит первый фронт SPI_CLK.

Прием-передача данных в подчиненном устройстве выполняется в сдвиговом регистре тактируемым SPI_CLK. Таким образом, для обеспечения высокой скорости обмена желательно использовать большую длину посылки. В этом случае у процессора будет больше времени для обслуживания одной порции принятых данных.

18 Контроллер видеокамеры

Данный контроллер позволяет организовать прием информации от внешней видеокамеры или иного устройства, поддерживающего заданный интерфейс.

Интерфейс видеокамеры представляет собой контроллер, подключенный к шине периферийных устройств и способный принимать данные от внешней цифровой видеокамеры (или другого устройства) согласно заданному протоколу обмена. Принятые данные помещаются в буфер и могут быть переданы в запоминающие устройства процессора с помощью контроллера DMA или с помощью процессора.

Внешние выводы контроллера видеокамеры приведены в таблице 148.

Таблица 148 – Внешние выводы контроллера

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PA[25]	VC_CLK	I	Интерфейс видеокамеры. Синхросигнал
PA[26]	VC_VSYNC	I	Интерфейс видеокамеры. Вход вертикальной развертки
PA[27]	VC_HSYNC	I	Интерфейс видеокамеры. Вход горизонтальной развертки видеоинтерфейса
PA[28]	VC_D[0]	I	Интерфейс видеокамеры. Бит 0 данных
PA[29]	VC_D[1]	I	Интерфейс видеокамеры. Бит 1 данных
PA[30]	VC_D[2]	I	Интерфейс видеокамеры. Бит 2 данных
PA[31]	VC_D[3]	I	Интерфейс видеокамеры. Бит 3 данных
PB[0]	VC_D[4]	I	Интерфейс видеокамеры. Бит 4 данных
PB[1]	VC_D[5]	I	Интерфейс видеокамеры. Бит 5 данных
PB[2]	VC_D[6]	I	Интерфейс видеокамеры. Бит 6 данных
PB[3]	VC_D[7]	I	Интерфейс видеокамеры. Бит 7 данных

Временные характеристики интерфейса приведены на рисунке 93.

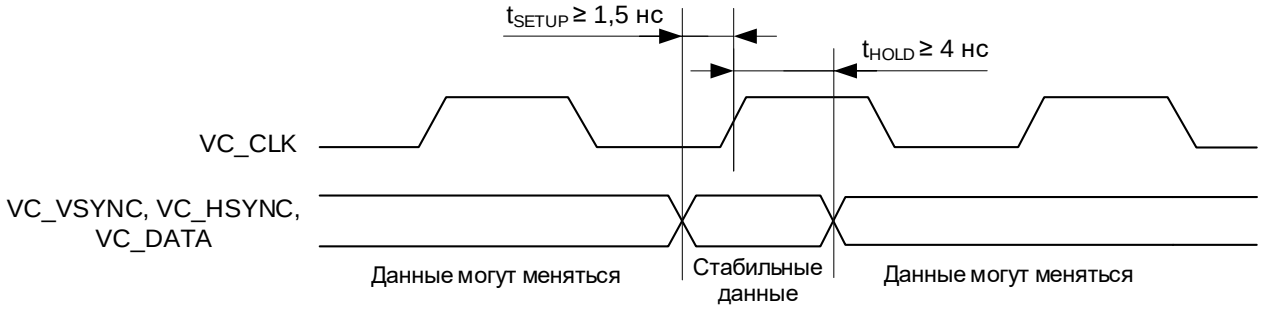


Рисунок 93 – Временная диаграмма интерфейса

18.1 Регистры интерфейса

Регистры интерфейса подключены к шине обмена периферийных устройств и имеют базовый адрес **0x800001A0**. Краткое описание регистров приведено в таблице 149.

Таблица 149 – Регистры интерфейса

Номер	Название	Назначение
0	VC_DR	128-разрядный регистр данных. Доступен только по чтению и является выходом FIFO
4	VC_SR	Регистр состояния
5	VC_CR	Регистр управления

18.1.1 Регистр управления CR

Таблица 150 – Регистр управления

Бит	Имя	Тип	Назначение
0	VCON	R/W	Включение интерфейса. При записи 1 интерфейс начинает прием данных во внутреннее FIFO.
1	VCIE	R/W	Разрешение прерывания. При записи 1 интерфейс вырабатывает прерывание если во внутреннем буфере имеются данные
2	SMODE	R/W	Режим приема информации: 0 – видеокамера; 1 – ведущее устройство
3	VPOL	R/W	Полярность сигнала VSYNC: 0 – активные данные передаются при уровне VSYNC = 0; 1 – активные данные передаются при уровне VSYNC = 1*
* При работе в режиме «ведущее устройство» устанавливать VSYNC = 1			

18.1.2 Регистр состояния SR

Таблица 151 – Регистр состояния

Биты	имя	Описание
0	EMPTY	1 – нет данных в буфере; 0 – имеются данные
1	FULL	0 – буфер заполнен не полностью; 1 – буфер заполнен полностью
2	OVERF	1 – было переполнение буфера во время приема данных; 0 – не было переполнения
3	UNDERF	1 – при чтении данных произошло чтение пустого буфера

18.1.3 Регистр данных DR

Посредством этого регистра происходит чтение данных после приема. Регистр имеет разрядность 128 бит и поэтому обращение к нему должно выполняться как к квадрослову. Размер буфера приема данных равен четырем 128-разрядным словам.

Интерфейс имеет внешние входы:

- DCLK – вход синхронизации;
- VSYNC – строб начала картины;
- HSYNC – строб начала линии;
- HD[7:0] – шина данных.

18.2 Режим приема «видеокамера»

Интерфейс является синхронным и осуществляет прием данных по фронту сигнала синхронизации DCLK. Для начала приема информации, контроллер должен быть включен путем установки бита VCON регистра управления. После включения интерфейс начинает отслеживать стартовую ситуацию для начала приема данных. В случае режима видеокамеры для начала приема данных необходимо наличие на входе VSYNC высокого уровня. Этот факт запоминается во внутреннем флаге. В момент, когда оба входа VSYNC и HSYNC станут равным нулю (не обязательно одновременно как на рисунке), будет установлен внутренний флаг camera_ON, означающий возможность приема данных. Таким образом, к моменту прихода нового кадра (VREF активный), интерфейс готов к приему данных. Временная диаграмма для случая VPOL = 1 приведена на рисунке 94.

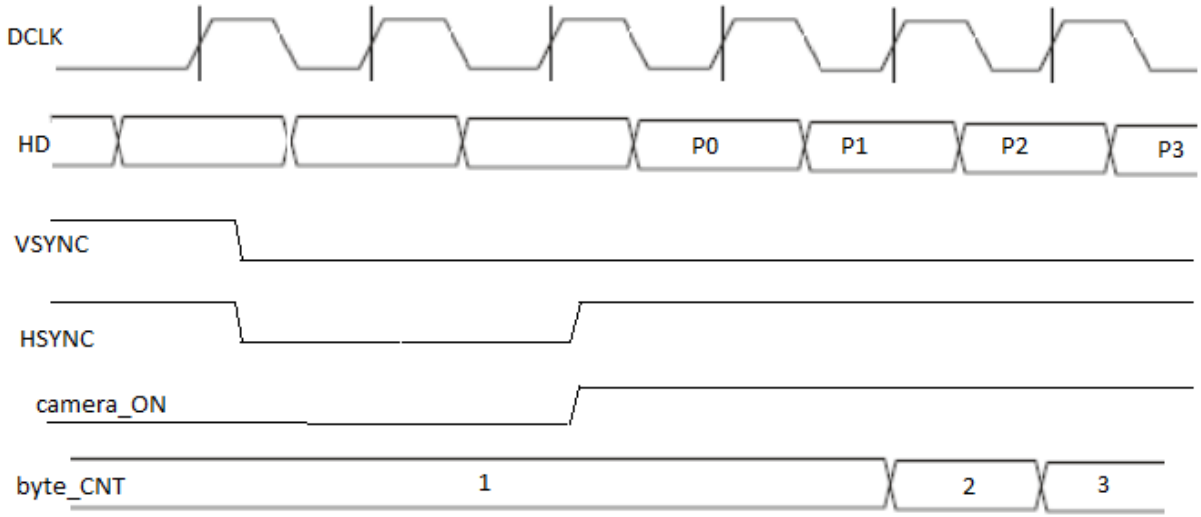


Рисунок 94 – Начало приема данных в режиме «видеокамера»

Прием данных начнется в момент, когда внутренний флаг camera_ON будет установлен и на входе HSYNC будет высокий уровень, что означает наличие данных горизонтальной строки на входе контроллера. На временной диаграмме видно расположение позиции первого принимаемого байта (P0). Каждый раз, когда HSYNC равен единице и включен прием, происходит наращивание счетчика байт byte_CNT. Блок данных из 16 байт формирует квадрослово данных. Когда счетчик данных достигает значения 16 – квадрослово данных принято, формируется флаг готовности и принятая информация записывается в выходное FIFO. Счетчик опять начинает счет сначала. Размер выходного буфера равен четырем квадрословам. При перезаписи данных в выходное FIFO выполняется передача информации из домена синхронизации DCLK в домен синхронизации SOC-шины. Как только в выходном FIFO появляются данные, формируется запрос к контроллеру DMA на обслуживание. К DMA может прочитать данные из буфера интерфейса и переслать во внутреннюю или внешнюю память (в соответствии с настройками канала DMA). В канале контроллера DMA происходит отслеживание объема переданной информации и при необходимости может быть сформирован запрос на прерывание для анализа принятой информации. Если контроллер DMA не успевает принимать поток данных от интерфейса, возможно переполнение буфера. В этом случае будет установлен соответствующий флаг в регистре состояния.

Интерфейс устроен так, что после приема кадра необходимо выключить интерфейс и затем для принятия нового кадра опять включить его.

18.3 Режим приема «ведущее устройство»

К внешним контактам интерфейса видеокамеры может быть подключено некоторое ведущее устройство, имеющее стандартный интерфейс микропроцессорной шины (адрес, данные, сигналы выборки и записи), с целью передачи информации по определенному протоколу. Интерфейс видеокамеры поддерживает простой вариант обмена, удобный для такого подключения. Данный вариант включается при установке бита SMOD в 1. В этом случае возможно следующее подключение сигналов:

– DCLK как сигнал выборки интерфейса nVCS;
– VSYNC как сигнал адресной шины A[1];
– HSYNC как сигнал адресной шины A[0];
– HD[7:0] как младший байт шины данных D[7:0].

Сигнал выборки nVCS должен быть сформирован как логическое «или» сигналов выборки и записи, т.е. nVCS = nCS or nWE (активный уровень низкий). Дополнительно к сигналу выборки может подключаться дешифратор адреса для дополнительного выбора устройства. При таком подключении обычный цикл записи по указанному адресу вызывает загрузку байта данных в интерфейс контроллера. Пример временной диаграммы приведен на рисунке 95.

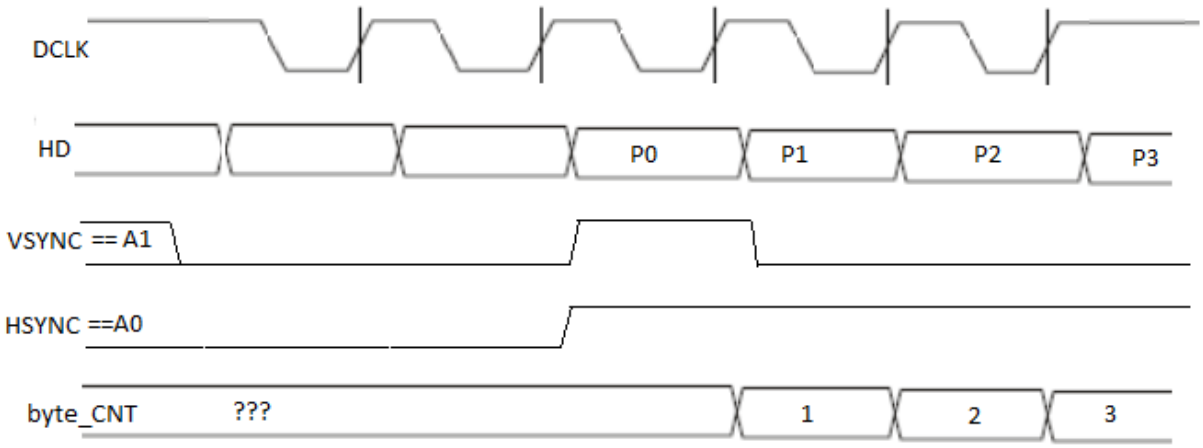


Рисунок 95 – Начало приема данных в режиме «ведущее устройство»

Для ведущего устройства передача данных в интерфейс выглядит как обычный цикл записи по адресам 0, 1, 2, 3. При этом:

– в интерфейс передаются только 8 бит данных, прием данных осуществляется квадрословами;

– запись по адресу 0 выглядит как подача одного такта синхросигнала DCLK без выполнения каких-либо действий внутри интерфейса. Для включения контроллера необходимо осуществить не менее двух записей по адресу 0.

– запись по адресу 3 инициирует прием первого байта данных и установку счетчика принятых байт в «1». Это позволяет ведущему устройству определить начальное значение счетчика;

– запись по адресу 1 приводит к записи следующего байта и автоматическому увеличению счетчика принятых байт на единицу. После приема 16-ти байт квадрослова счетчик автоматически устанавливается равным нулю, а принятое квадрослово пересылается в выходной FIFO буфер;

– запись по адресу 2 выполняет обнуление счетчика байт. Счетчик считает 0, 1, ...15, 0, 1... Установить в счетчике фиксированное значение можно только записью по адресу 2 или 3;

– скорость подачи данных ограничивается возможностями процессора по обслуживанию принимаемых данных.

После приема 16 байт выполняется передача квадрослова данных в выходное FIFO, и формируется запрос к контроллеру DMA или прерывание к процессору. Если передаваемый блок данных не кратен 16 байт, его необходимо дополнить недостающими байтами. При формировании сигнала записи (DCLK) необходимо обеспечить требуемое время предустановки входных сигналов и время удержания относительно фронта строба записи.

Таким образом, в режиме «ведущее устройство», интерфейс видеокамеры может рассматриваться как ведомый порт для приема информации от ведущего устройства. В зависимости от скорости передачи информации для обслуживания интерфейса может использоваться контроллер DMA или процессор.

19 Интерфейс NAND флэш-памяти

К процессору может быть подключена внешняя NAND флэш-память.

Внешние выводы интерфейса NAND флэш-памяти приведены в таблице 152.

Таблица 152 – Внешние выводы

Обозначение вывода (основное)	Обозначение вывода для интерфейса NAND	Тип вывода	Функциональное назначение
PA[2]	NF_RE	O	Интерфейс NAND Flash. Строб чтения
PA[3]	NF_WE	O	Интерфейс NAND Flash. Строб записи
PA[4]	NF_ALE	O	Интерфейс NAND Flash. Подсветка адреса
PA[5]	NF_D[0]	I/O	Интерфейс NAND Flash. Бит данных 0
PA[6]	NF_D[1]	I/O	Интерфейс NAND Flash. Бит данных 1
PA[8]	NF_CS[1]	O	Интерфейс NAND Flash. Выбор модуля 1
PA[9]	NF_D[2]	I/O	Интерфейс NAND Flash. Бит данных 2
PA[10]	NF_D[3]	I/O	Интерфейс NAND Flash. Бит данных 3
PA[11]	NF_D[4]	I/O	Интерфейс NAND Flash. Бит данных 4
PA[12]	NF_D[5]	I/O	Интерфейс NAND Flash. Бит данных 5
PA[13]	NF_CLE	I/O	Интерфейс NAND Flash. Подсветка команды
PA[14]	NF_D[6]	I/O	Интерфейс NAND Flash. Бит данных 6
PA[15]	NF_D[7]	I/O	Интерфейс NAND Flash. Бит данных 7
PA[17]	NF_CS[0]	O	Интерфейс NAND Flash. Выборка модуля 0
PA[18]	NF_RDY	I	Интерфейс NAND Flash. Вход готовности
PD[16]	NF_D[0]	I/O	Интерфейс NAND Flash. Бит данных 0
PD[17]	NF_D[1]	I/O	Интерфейс NAND Flash. Бит данных 1
PD[18]	NF_D[2]	I/O	Интерфейс NAND Flash. Бит данных 2
PD[19]	NF_D[3]	I/O	Интерфейс NAND Flash. Бит данных 3
PD[20]	NF_D[4]	I/O	Интерфейс NAND Flash. Бит данных 4
PD[21]	NF_D[5]	I/O	Интерфейс NAND Flash. Бит данных 5
PD[22]	NF_D[6]	I/O	Интерфейс NAND Flash. Бит данных 6
PD[23]	NF_D[7]	I/O	Интерфейс NAND Flash. Бит данных 7
PD[24]	NF_CLE	O	Интерфейс NAND Flash. Строб записи
PD[25]	NF_ALE	O	Интерфейс NAND Flash. Строб чтения
PD[26]	NF_RE	O	Интерфейс NAND Flash. Подсветка адреса
PD[27]	NF_WE	O	Интерфейс NAND Flash. Подсветка команды
PD[28]	NF_CS[0]	O	Интерфейс NAND Flash. Выборка модуля 0
PD[29]	NF_CS[1]	O	Интерфейс NAND Flash. Выборка модуля 1
PD[30]	NF_CS[2]	O	Интерфейс NAND Flash. Выборка модуля 2
PD[31]	NF_RDY	I	Интерфейс NAND Flash. Вход готовности

Специальный контроллер позволяет осуществить преобразование запроса процессора на запись или чтение данных в последовательность команд обращения к флэш-памяти. Имеется поддержка коррекции ошибок.

При использовании данного контроллера разрядность шины данных контроллера внешней памяти уменьшается до 16 бит. Процессор не имеет прямого доступа во флэш-

память, и все операции осуществляются посредством программирования операций контроллера.

Возможно подключение до трех внешних банков памяти. Выбор конкретного банка определяется разрядами 31:30 регистра адреса AR:

- 00 – NF_CS 0;
- 01 – NF_CS 1;
- 10 – NF_CS 2.

Биты AR[29:0] позволяют прямо адресовать 4 Гбайт каждого банка памяти. Если подключаемая память имеет больший объем, можно использовать биты расширения ADDRH[2:0] регистра NAND_CFG. Это позволяет адресовать до 32 Гбайт каждого банка флэш-памяти. Чтение и запись в NAND флэш-память имеет свои особенности. При описании протокола обмена с флэш-памятью, будет подразумеваться работа только(!) с флэш-памятью фирмы Samsung, а также памятью с аналогичным интерфейсом от других компаний.

Контроллер флэш-памяти имеет набор регистров, позволяющих задать параметры внешней подключенной памяти и режимы работы. Регистры контроллера доступны со стороны шины периферийных устройств и имеют базовый адрес 0x80000240.

С помощью регистров конфигурации пользователь может задать направление обмена (чтение или запись), начальный адрес обмена, количество передаваемых слов, а также размер слова (int, short, char). После задания необходимых параметров контроллер самостоятельно читает данные из внешней памяти во внутренний буфер (в случае чтения) или записывает данные из внутреннего буфера во внешнюю память (в случае записи). Для передачи данных между внутренним буфером контроллера и памятью системы, может использоваться процессор или контроллер прямого доступа. Условно-графическое обозначение модуля контроллера флэш-памяти показано на рисунке 96.

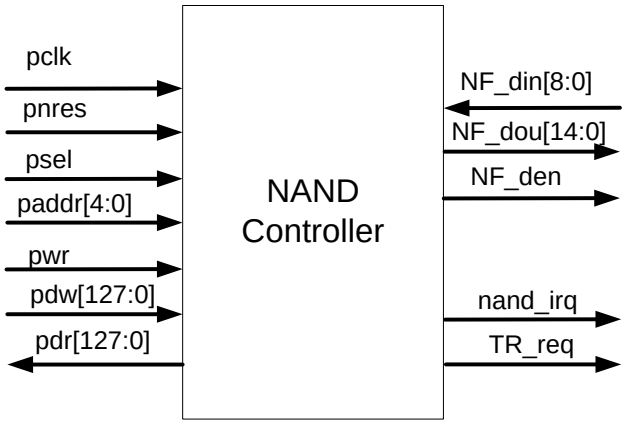


Рисунок 96 – Условное графическое обозначение контроллера

На рисунке показаны интерфейс подключения контроллера к шине периферийных устройств, посредством которого пользователь может осуществлять программирование операций обмена, а также интерфейс для взаимодействия с внешней флэш-памятью. Подробнее описание сигналов модуля приведено в таблице 153.

Таблица 153 – Сигналы контроллера флэш-памяти

Сигнал	Направление	Активный уровень	Описание
Интерфейс APB			
pclk	Вход	-	Тактовый сигнал
pnres	Вход	Низкий	Сигнал сброса
psel	Вход	Высокий	Сигнал выбора
paddr[4:0]	Вход	-	Адрес
pwr	Вход	-	Сигнал чтения или записи (низкий – чтение, высокий – запись)
pdw[127:0]	Вход	-	Данные на запись
pdr[127:0]	Выход	-	Прочитанные данные
Интерфейс NAND			
NF_din[7:0]	Вход	-	Шина входных данных
NF_din[8]	Вход	-	NF_RDY – сигнал готовности микросхемы FLASH (1 - готово)
NF_dou[7:0]	Выход	-	Шина выходных данных (команды, адреса)
NF_dou[8]	Выход	Высокий	NF_CLE – строб команды
NF_dou[9]	Выход	Высокий	NF_ALE – строб адреса
NF_dou[10]	Выход	Низкий	NF_RE – строб записи
NF_dou[11]	Выход	Низкий	NF_WE – строб чтения
NF_dou[12]	Выход	Низкий	NF_CE[0] – выборка микросхемы 0
NF_dou[13]	Выход	Низкий	NF_CE[1] – выборка микросхемы 1
NF_dou[14]	Выход	Низкий	NF_CE[2] – выборка микросхемы 2

NF_den	Выход	-	Сигнал, определяющий направление для двунаправленного порта DIO микросхемы флэш-памяти:
			1 – данные поступают во флэш;
			0 – данные поступают из флэш).
			Пример:
			assign IO_bus = NF_den ? NF_dou[7:0] : 8'bz;
			assign NF_di[7:0] = IO_bus;

Сигналы запросов			
nand_irq	Выход	Высокий	Запрос на прерывание
TR_req	Выход	Высокий	Запроса на обмен с системным DMA

Контроллер может использоваться для совместной работы с контроллером DMA. Для этих целей предусмотрен выход TR_req. Изменение выхода TR_req из состояния 0 в состояние 1 сигнализирует контроллер DMA о необходимости провести обмен данными.

Соответствие сигналов управления внешней флэш-памятью контактным площадкам микросхемы приведено в таблице 154. Предусмотрено два варианта соединения выводов контроллера с внешними выводами:

Вариант 1 – при начальной загрузке значением выводов BOOT[2:0] = 100 и по умолчанию при работе (подробнее см. GPIO.Функции порта PD);

Вариант 2 – при начальной загрузке значением выводов BOOT[2:0] = 110 и битами CFG1[18:17] (подробнее см. подраздел 31.1 «Регистр конфигурации периферийных модулей CFG1»).

Таблица 154 – Внешние контакты контроллера

Имя	Порт		Функция
	Вариант 1	Вариант 2	
NF_RDY	PD[31]	PA[18]	Вход готовности флэш-памяти
NF_CS[2]	PD[30]	–	Выход. Выбор банка 2
NF_CS[1]	PD[29]	PA[8]	Выход. Выбор банка 1
NF_CS[0]	PD[28]	PA[17]	Выход. Выбор банка 0
NF_CLE	PD[24]	PA[13]	Выход. Строб команды
NF_ALE	PD[25]	PA[4]	Выход. Строб адреса
NF_RE	PD[26]	PA[2]	Выход. Строб чтения
NF_WE	PD[27]	PA[3]	Выход. Строб записи
NF_D[7]	PD[23]	PA[15]	Шина данных. Бит 7
NF_D[6]	PD[22]	PA[14]	Шина данных. Бит 6
NF_D[5]	PD[21]	PA[12]	Шина данных. Бит 5
NF_D[4]	PD[20]	PA[11]	Шина данных. Бит 4
NF_D[3]	PD[19]	PA[10]	Шина данных. Бит 3
NF_D[2]	PD[18]	PA[9]	Шина данных. Бит 2
NF_D[1]	PD[17]	PA[6]	Шина данных. Бит 1
NF_D[0]	PD[16]	PA[5]	Шина данных. Бит 0

19.1 Регистры управления контроллером

Поведение контроллера, параметры интерфейса обмена определяются посредством регистров конфигурации контроллера.

В таблице 155 приведен список регистров контроллера. Регистры контроллера доступны со стороны шины периферийных устройств и имеют базовый адрес 0x80000240.

Таблица 155 – Регистры контроллера флэш-памяти

Смещение	Имя	Описание	Сброс
0	IO_CFG	Регистр конфигурации временных параметров	0x00a3ffff
1	WCT_CFG	Регистр конфигурации времени ожидания	0x000000ff
2	NAND_CFG	Регистр конфигурации протокола обмена	0x0fff035b
3	WR_CFG	Регистр команд записи	0x00100080
4	RD_CFG	Регистр команд чтения	0x00003000
5-7	-	Не используются	
8	CR	Регистр управления	
9	SR	Регистр состояния	
10	AR	Регистр начального адреса обмена	
11	CNTR	Счетчик количества передаваемых слов	
12	DR	Буфер данных.	
13	ERR12	Регистр номеров ошибок 1 и 2	
14	ERR34	Регистр номеров ошибок 3 и 4	
15	SP_BUF	Регистр номера ошибочной подстраницы	

19.1.1 IO_CFG – регистр конфигурации временных параметров

Данный регистр позволяет определить основные временные параметры протокола обмена. Основные сигналы интерфейса обмена и основные временные параметры изображены на рисунке 97). Длительность стробов выражается количеством тактов частоты синхронизации, на которой работает контроллер. В качестве базовой частоты используется частота периферийной шины.

Таблица 156 – Регистр IO_CFG

Бит	Имя	Описание	Сброс
2:0	CSCA	Число периодов(+1) CLK с момента активизации NF_CS [2:0] до момента активизации сигналов nRE или nWE	111
7:3	CA	Число периодов (+1) активности сигнала nRE во время чтения памяти	11111
10:8	BWD	Число периодов (+1) активности nWE во время записи.	111
14:11	BRT	Число периодов (+1) от момента снятия одного NF_CSx до момента активизации NF_CSx другого банка, с целью избежать конфликт на шине данных	1111
17:15	BHT	Число периодов (+1) CLK в течение которых NF_CSx находится в неактивном состоянии между последовательными выборками. А также число периодов (+1) после снятия nRE или nWE до следующего активного сигнала.	111
18	MDT	Тип интерфейса. Используется совместно с VGA битом	0
20:19	-	Не используется. Значение безразлично	00
21	VGA	Тип интерфейса. Имеет значение при установленном MDT	1
22	-	Не используется. Значение безразлично	
26:23	CSKPL	Управление линиями NF_CSx между доступами к флэш-памяти: 0 – линии NF_CSx становятся неактивными между выборками; 1 – линии NF_CSx активны (=0) между выборками	0001
31:27	-	Не используется. Значение безразлично	

Отметим функции некоторых бит, значение которых может быть недостаточно понятно из краткого описания в таблице.

Биты MDT и VGA позволяют управлять функцией линий интерфейса: ALE, CLE, nRE (таблица 157).

Таблица 157 – Дополнительные функции интерфейса

MDT	VGA	ALE	CLE	nRE
0	-	ALE	CLE	nRE
1	0	BA[3]	BA[2]	nRE
1	1	nWRITE	BA[2]	STB

Отметим, что строб STB = ~(nRE & nWE). Таким образом, можно запрограммировать обмен с устройствами, которые имеют более простой интерфейс обмена. Данная функция работает только в случае, когда фазы передачи команды и адреса выключены.

Биты CSKPL необходимо использовать в следующей ситуации. При чтении или записи выборка кристалла (NF_CS) активна только на период одного обмена. Поскольку в операциях с памятью всегда используется последовательный режим обмена, то в некоторых микросхемах флэш-памяти снятие выборки после очередного последовательного чтения вызовет останов наращивания адреса внутри флэш-памяти и возврат ее внутренней машины состояния в исходное состояние. Чтобы этого не произошло необходимо установить бит CSKPL = 1 для соответствующего NF_CS. В этом случае выборка микросхемы памяти будет активна постоянно, начиная с первого обращения.

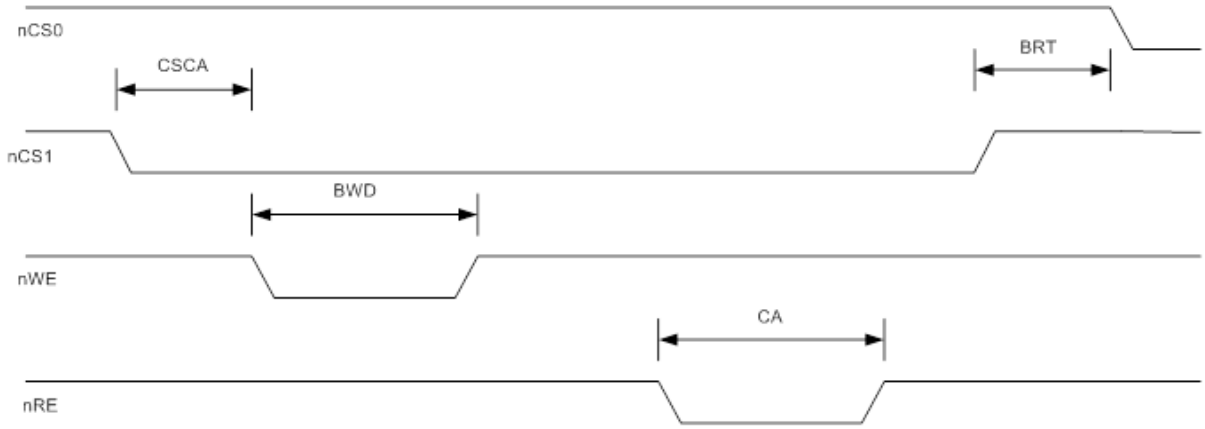


Рисунок 97 – Основные временные параметры обмена

19.1.2 WCT_CFG – регистр конфигурации времени ожидания

Регистр позволяет управлять анализом сигнала готовности внешней памяти и программировать максимальное время ожидания.

Таблица 158 – Регистр WCT_CFG

Бит	Имя	Описание	Сброс
3:0	ENWT	Разрешение анализа входа RnB во время обмена с одним из банков NF_CS[3:0]: 0 – RnB не анализируется; 1 – RnB используется для анализа во время цикла обмена	1111
7:4	CCSE	Управление активностью сигнала выборки банка Эти биты позволяют снять выборку банка во время RnB = 0. Биты регистра 7:4 соответствуют NF_CS[3:0]: 0 – установить неактивный уровень если RnB =0; 1 – разрешить активный уровень на NF_CSx во время RnB=0	1111
30:21	WTOC	Длительность периода «time-out» Эти биты определяют длительность периода, используемого для контроля ситуации «time-out». Превышение длительности RnB=0 над длительностью периода WTOC•1024 приводит к ситуации «time-out» и установке бита TOEX	0

Бит	Имя	Описание	Сброс
31	TOE	Разрешение контроля ситуации «time-out» 0 – запрещено; 1 – разрешено	0

Биты ENWT позволяют разрешать для каждого NF_CS анализ сигнала RnB. При подключении устройств к контроллеру флэш-памяти может оказаться, что некоторые NF_CS используются для выборки устройств, которые не управляют сигналом RnB. В этом случае необходимо очистить бит ENWT соответствующего NF_CS.

Функции бит CCSE могут быть использованы в следующем случае. При чтении новой страницы или при записи флэш-памяти вырабатывается сигнал RnB=0 на продолжительное время. Если бит CCSE равен нулю для соответствующего NF_CS, то в случае RnB = 0 сигнал выборки станет неактивным на период времени, когда RnB равен нулю. Для некоторых типов микросхем это позволяет снизить мощность потребления.

Биты WTOC позволяют запрограммировать интервал времени time-out. При этом будет отслеживаться период равный WTOC•1024 тактов.

Бит TOE разрешает анализ длительности обмена.

19.1.3 NAND_CFG – регистр конфигурации протокола обмена

Позволяет задать основные параметры флэш-памяти и режимы работы.

Таблица 159 – Регистр NAND_CFG

Бит	Имя	Описание	Сброс
2:0	VOLCOL	Размер страницы: 000 – 128 байта; 001 – 512 байта; 010 – 1024 байта; 011 – 2048 байта; 100 – 4096 байта; 101 – 8192 байта	011
3	ROWBT	Число байт адреса для выборки строки: 0 – два байта; 1 – три байта	1
5:4	COMCLW	Число команд для записи: 00 – 1; 01 – 2; 10 – 3 (расширенный режим); 11 – не используется	01
7:6	COMCLR	Число команд для чтения: 00 – 1; 01 – 2; 10 – не используется; 11 – не используется	01

Бит	Имя	Описание	Сброс
9:8	ADRCL	Число адресных циклов: 00 – один цикл; 01 – только выбор столбца; 10 – только выбор строки; 11 – строка и столбец	11
10	-	Не используется	0
11	-	Не используется	0
12	ADRSC	Управление пропуском адресного цикла: 0 – передавать адрес во время обмена; 1 – пропустить	0
13	DATSC	Управление пропуском цикла данных: 0 – передавать данные во время обмена; 1 – пропустить	0
14		Не используется.	
15	COMSC	Управление пропуском цикла передачи команд: 0 – передавать команды во время обмена; 1 – пропустить	0
20:16	RESBF	Время задержки до анализа активности RnB. Биты задают длительность (*2) периода от момента передачи последнего байта адреса до момента анализа сигнала RnB.	0x1F
27:21	RESAF	Время задержки после завершения активности RnB. Биты задают длительность (*2) периода от момента, когда сигнал RnB стал неактивным до момента чтения или записи данных.	0x7F
31:28	ADDRH	Биты адреса расширения	0000

Биты VOLCOL определяют размер страницы флэш-памяти.

Бит ROWBT определяет количество байт адреса, которые необходимо передать во флэш-память для выборки страницы. Для задания номера начального байта внутри страницы используется два байта, что позволяет использовать страницы размером до 64 Кбайт.

Биты COMCLW определяют количество команд (байт команд), используемых при записи данных во флэш-память.

Биты COMCLR выполняют аналогичную функцию при чтении.

Биты ADRCL позволяют управлять передачей адреса.

Если бит ADRSC установлен, контроллер пропускает фазу передачи адреса.

Если бит DATSC установлен, контроллер пропускает фазу управления данными.

Бит COMSC управляет пропуском фазы передачи команд.

При чтении новой страницы флэш-память вырабатывает низкий уровень на выходе RnB. Его длительность равна времени считывания новой страницы из накопителя в буфер (флэш-памяти). Флэш-память изменяет значение на выходе RnB после передачи последнего байта адреса с некоторой задержкой.

Биты RESBF позволяют задержать анализ вывода RnB после передачи адреса перед началом чтения. Это позволяет избежать срабатывания неверного чтения.

Биты RESAF позволяют запрограммировать время готовности памяти после того как на линии RnB установится высокий уровень.

Биты ADDRH позволяют подключить к одному NF_CS микросхемы памяти объемом больше 4Г байт.

В одном из режимов старта системы предусмотрен старт с начальной загрузкой из флэш-памяти. После сброса контроллер настроен на работу с микросхемой, у которой страница памяти равна 2К байт. Для выбора строки будет использовано три байта адреса, а для выбора столбца два байта адреса. При передаче адреса будут переданы все пять байт адреса и две команды чтения.

19.1.4 WR_CFG и RD_CFG – регистры конфигурации

Регистры используются для задания значения команд, используемых при чтении или записи флэш-памяти.

Таблица 160 – Регистр WR_CFG

Бит	Имя	Описание	Сброс
7:0	COMW1	первая команда записи	0x80
15:8	COMW2	вторая команда записи	0x00
23:16	COMW3	третья команда записи	0x10
31:24	-		

Таблица 161 – Регистр RD_CFG

Бит	Имя	Описание	Сброс
7:0	COMR1	первая команда чтения	0x00
15:8	COMR2	вторая команда чтения	0x30
23:16	COMR3	третья команда чтения	0x00
31:24	-		

19.1.5 CR – регистр управления

Назначение разрядов регистра приведено в таблице 162. В данном регистре программируются дополнительные параметры обмена.

Таблица 162 – Регистр CR

Бит	Название	Функция
0	EN	Разрешение работы (1)
1	RW	Чтение (0) или запись (1)
2	-	
3	SQE	Разрешение последовательного доступа (когда 0)
4	RIM	Разрешение прерывания при чтении (1 – разрешение)
5	TIM	Разрешение прерывания при записи (1 – разрешение)
6	CIM	Разрешение прерывания при достижении счетчика значения 0 (1 – разрешение)

Бит	Название	Функция
7	-	
9:8	SZ	Размер передаваемых данных: 00 – четыре байта; 10 – два байта; 11 – байт
10	-	
11	DBSZ	Размер шины данных FIFO контроллера: 0 – 32 бита; 1 – 128 бит
12	ERI_EN	Разрешение прерывания в случае обнаружения ошибочной ситуации в работе контроллера, т.е. когда обнаруживается ситуация тайм-аут
13	RnBI_EN	Разрешение прерывания в случае обнаружения фронта на входе RnB, т.е. когда флаг RnB_Fрегистра состояния равен 1
14	EIM	1 – разрешение прерывания при обнаружении ошибки ECC
15	ECC_ON	1 – включение режима ECC
16	EN_CORR	Бит управления обнаружением ошибок (для микросхем с ревизии 3): 1 – модуль ECC только детектирует ошибки; 0 – модуль ECC корректирует ошибки
17	WT_ESRV	Ожидание обслуживания (для микросхем с ревизии 3): 0 – нет ожидания; 1 – приостановка работы до момента окончания обслуживания модуля
31:18	-	Не используются

После сброса значение регистра управления равно 0x0001, т.е. контроллер включен, настроен на чтение слов с размерностью шины данных в 32 бита.

Далее будут описаны типовые процедуры чтения и записи данных. Они состоят из передачи управляющих команд, адреса и блока данных. Вся эта процедура выполняется, когда бит SQE равен нулю. Если установить бит SQE в единицу, указанный протокол будет выполняться для каждого слова данных, т.е. после передачи команды, адреса и чтения четырех байт данных (биты SZ равны 00), перед чтением следующих четырех байт будет повторно выполнена передача команд и адреса. Данный режим очень медленный и не рекомендуется для применения.

Биты разрешения прерывания RIM, TIM, CIM могут быть использованы для организации работы контроллера вместе с процессором. Например, если выполняется чтение данных, разрешение прерывания RIM будет сигнализировать процессору о том, что во внутреннем буфере имеются данные для чтения. Если выполняется запись данных, разрешение прерывания TIM будет сигнализировать процессору о том, что во внутреннем буфере имеется место для записи данных. Прерывание CIM сигнализирует процессору о завершении обмена, т.е. во внутренний буфер или из внутреннего буфера были передано количество слов данных указанное в счетчике CNTR.

Бит RW управляет направлением передачи данных. Значение «0» определяет чтение данных из внешней памяти во внутренний буфер, а значение «1» задает передачу данных из внутреннего буфера во внешнюю память. Отметим, что чтение данных из внешней памяти инициируется только, если есть свободное место во внутреннем буфере,

а запись во внешнюю память инициируется только когда есть данные во внутреннем буфере.

Бит EN разрешает работу контроллера. Если в данный бит записать ноль, произойдет очистка внутренних указателей внутреннего буфера и перевод машины состояний в исходное положение.

Биты SZ задают размер слова передаваемых данных. Размер один или два байта должен использоваться только для спецопераций.

Биты DBSZ задают размер шины данных FIFO. При смене размера шины необходимо выключить контроллер NAND флэш-памяти.

19.1.6 DR – регистр данных

Контроллер имеет FIFO данных объемом восемь 32-разрядных слов. Оно выполняет функцию буфера между системой и внешней флэш-памятью. При чтении данных из флэш-памяти, данные записываются в FIFO. При записи – данные считываются из FIFO и передаются в флэш-память. Можно установить размер шины данных FIFO. Если бит DBSZ равен нулю, размер FIFO составит 8×32 бита, а если DBSZ равен единице, размер FIFO – 2×128. В последнем случае процессор или DMA должны производить обмен квадрословами. Признаки готовности данных также формируются с учетом размерности шины данных.

Рассмотрим поведение буфера данных в зависимости от размерности шины данных FIFO.

Шина данных 32 бита

В этом случае буфер имеет размер восемь слов по 32 бита. Имеются указатели на запись и чтение каждый разрядностью три бита. Запись в буфер увеличивает указатель записи на единицу. Чтение из буфера уменьшает указатель чтения на единицу.

Признак готовности буфера принимать данные – TFS активен (равен единице) в случаях, когда буфер пуст, или, когда буфер не заполнен полностью, т.е. TFS==1 означает что в буфер можно записать одно слово данных.

Признак RFS (для случая, когда буфер настроен на чтение) активен (равен единице) в случаях, когда имеются данные для чтения, т.е. когда буфер не пуст. Это означает, что из буфера можно прочитать одно слово данных.

Шина данных 128 бит

В этом случае буфер имеет разный вид со стороны процессора и со стороны контроллера флэш-памяти. Для процессора буфер имеет размер два слова по 128 бит, а для контроллера по-прежнему восемь слов по 32 бита. Указатели на запись и чтение изменяются в зависимости от направления передачи:

– Запись

Запись в буфер увеличивает указатель записи на четыре, т.к. в буфер помещаются сразу четыре слова данных. Чтение из буфера уменьшает указатель чтения на единицу. Признак готовности буфера принимать данные TFS активен (равен единице) в случаях, когда буфер пуст или, когда буфер не заполнен полностью, т.е. TFS==1 означает, что в буфер можно записать одно квадрослово данных.

– Чтение

Запись в буфер со стороны флэш-памяти увеличивает указатель записи на единицу, т.к. в буфер помещается одно слово данных. Чтение из буфера уменьшает указатель чтения на четыре, т.к. считывается сразу четыре слова. Признак RFS активен (равен единице) в случаях, когда имеется хотя бы одно квадрослово данных для чтения, т.е. когда из памяти прочитаны как минимум четыре слова. Это означает, что из буфера можно прочитать одно квадрослово данных.

Бит разрядности шины данных имеет отношение только к буферу данных. Доступ к регистрам управления осуществляется только словами по 32 бита. При смене разрядности шины данных буфера **обязательно** нужно выполнить выключение контроллера (CR[0]=0). Это приведет к обнулению указателей чтения и записи и таким образом гарантирует их дальнейшую корректную работу.

Для операций с флэш-памятью, требующих чтения менее, чем четырех слов данных необходимо использовать только 32-разрядный режим шины данных буфера.

19.1.7 AR – регистр адреса

Перед началом обмена необходимо задать адрес, начиная с которого будет производиться чтение или запись. В последующем этот адрес будет автоматически увеличиваться на единицу. При записи значения в регистр адреса необходимо помнить, что данный адрес есть адрес слова. Внутри контроллера адрес расширяется AF[33:0]={AR[31:0],2’b00} двумя нулевыми битами для формирования указателя на байт. Биты AF[33:32] используются для выбора микросхемы памяти, а биты AF[31:0] прямо адресуют память. Это позволяет адресовать 4 Гбайт памяти. Внутри контроллера биты AF[31:0] дополняются битами расширения адреса AX[35:0]=={ADDRH[3:0],AF[31:0]}. Биты расширения имеют значение, если объем подключенной микросхемы памяти более 4 Гбайт.

Внутри контроллера исполнительный адрес AX делится на адрес столбца и адрес строки. Значимым параметром является адрес столбца (column). Количество бит, выделяемое для адреса столбца, зависит от размера страницы флэш-памяти. Например, для случая, когда страница равна 2К байт в качестве адреса столбца будут использованы биты AX[10:0]. Биты AX[35:11] будут адресовать строку.

19.1.8 CNTR – счетчик количества слов

Задает количество слов, которое будет прочитано из флэш-памяти или записано во флэш-память. Счетчик имеет разрядность 24 бита. При этом длина слова может быть один, два или четыре байта.

19.1.9 SR – регистр состояния

Отражает значение флагов запросов прерываний. Биты регистра описаны в таблице 163.

Таблица 163 – Регистр SR

Бит	Название	Функция
0	EMPTY	FIFO данных пусто (1)
1	FULL	FIFO данных заполнено полностью (1)
2	TFS	Запрос от FIFO данных на запись новых данных процессором или контроллером прямого доступа. FIFO наполовину пусто (1)
3	RFS	Запрос от FIFO данных на чтение новых данных процессором или контроллером прямого доступа. FIFO наполовину заполнено (1)
4	IRQ	Общий запрос прерывания от контроллера. Более детально причина запроса прерывания может быть установлена согласно состояния бит 20:16 регистра: 0 – нет запроса; 1 – запрос на прерывание
7:5	-	-
11:8	TOEX	Флаги состояния Time-Out. Биты отражают факт обнаружения ситуации «time-out» в процессе обмена с соответствующим NF_CSx: 0 – нет ошибки; 1 – обнаружена ситуация «time-out» Биты очищаются программно посредством записи в них 1
12	-	
13	RnB_F	Флаг изменения входа RnB из низкого уровня «0» в высокий «1»: 0 – не было изменения; 1 – был переход линии RnB из «0» в «1». Бит очищается программно посредством записи в него единицы
15:14	-	
16	RFS_I	Запрос на прерывание, когда RFS==1 и бит RIM==1. Запрос может быть очищен посредством чтения буфера данных
17	TFS_I	Запрос на прерывание, когда TFS==1 и бит TIM==1. Запрос может быть очищен посредством записи в буфер данных
18	CNTZ_I	Запрос на прерывание, когда значение счетчика становится равным нулю и бит CIM==1. Запрос может быть очищен посредством записи нового значения счетчика
19	TO_I	Запрос на прерывание, когда TOEX==1 и бит ERI_EN ==1. Запрос может быть очищен посредством записи 1 в бит TOEX
20	RnBF_I	Запрос на прерывание, когда RnB_F==1 и бит RnBI_EN ==1. Запрос может быть очищен посредством записи 1 в бит RnB_F
30:21	-	-
31	RnB	Вход готовности NAND флэш-памяти. Отражает состояние линии RnB

Контроллер интерфейса содержит внутренний счетчик, который позволяет отследить ситуацию, когда обмен с памятью превышает допустимый временной интервал (ситуация time-out). Это служит защитой от блокировки работы процессора, выйти из которой можно только сбросив всю систему.

Если ситуация time-out произошла при обращении к флэш-памяти, подключенной к некоторому NF_CS выводу, соответствующий бит TOEX в регистре состояния будет установлен и цикл обмена завершится некорректно, т.е. считанные данные будут неверными или запись будет осуществлена неверно. Подобная ситуация может произойти если длительность низкого уровня на входе RnB превышает период time-out. Процесс аварийного завершения обмена может вызвать прерывание работы процессора, если соответствующее прерывание разрешено.

Сбросить биты TOEX можно только записав в них значение «1». Запись «0» в соответствующие биты не вызывает их изменения.

Бит RnB отражает состояние соответствующего выхода готовности микросхемы флэш-памяти. Бит может быть использован для определения готовности флэш-памяти.

19.2 Чтение NAND флэш-памяти

Чтение флэш-памяти всегда осуществляется в последовательном режиме и представляет собой передачу во внешнюю память команд, байт адреса и затем чтение непосредственно самих данных. Различные варианты чтения отличаются друг от друга количеством передаваемых байт команд и адреса, а также количеством принимаемых байт данных.

При чтении данных вначале происходит передача одного (или двух) байта команды, затем передаются байты адреса, после этого читаются данные. Первую команду, с которой начинается чтение, необходимо записать в поле COMR1 регистра RD_CFG.

Если размер страницы равен 512 байт, команды для доступа к первой половине страницы и ко второй половине различны (00h – для первой, 01h – для второй). Контроллер отслеживает эту ситуацию сам. Пользователь должен записать в поле COMR1 значение 00h и определить размер страницы как 512 байт. Если при чтении контроллер обнаруживает, что обращение осуществляется ко второй половине страницы, он автоматически устанавливает команду 01h вместо 00h.

После передачи команды контроллер посылает во флэш-память байты адреса. Количество адресных байт определяется в регистре конфигурации и зависит от емкости микросхемы памяти. После этого контроллер читает заданное количество байт данных последовательно и помещает их во внутренний буфер.

Передаваемое значение адреса формируется из регистра адреса AR (биты 29:0) и поля расширения адреса ADDRH (биты 2:0). Поле расширения определяет старшие биты передаваемого адреса.

Контроллер ориентирован на чтение количества байт кратных четырем. Однако для выполнения специальных операций (сброс микросхемы, чтение регистра состояния микросхемы) можно выполнять более короткие операции чтения. Для чтения одного байта необходимо в счетчик слов CNTR записать значение «1» и установить размер передаваемых данных в регистре управления CR равным байту. Для чтения двух байт в счетчик слов записываем «1», а размер данных устанавливаем равным полуслову.

19.3 Запись в NAND флэш-память

Операция записи во многом аналогична операции чтения только направление передачи данных противоположное, т.е. из внутреннего буфера во внешнюю микросхему памяти. Команды, используемые при записи во флэш-память, должны быть записаны в поля регистра конфигурации WR_CFG. В типичном случае протокол записи данных в память состоит из передачи первого командного байта, передачи байт адреса, записи данных и передачи второго байта команды, инициирующего процесс записи внутри флэш-памяти.

Необходимо отметить, что во время программирования текущей страницы во внутреннем накопителе, флэш-память становится недоступной. Это занимает довольно продолжительное время. При выполнении операций контроллер анализирует готовность памяти и в случае обращения процессора за новыми данными в момент программирования предыдущих данных, будет осуществлена остановка работы контроллера до момента готовности флэш-памяти.

Для записи данных необходимо определить две команды: для инициализации процесса записи и для его окончания. Эти две команды записываются в поля COMW1 и COMW3 регистра конфигурации. Вначале посылается первая команда, затем байты адреса, байты данных. Во время пересылки данных во внешнюю память контроллер анализирует состояние счетчика CNTR (после пересылки каждого слова значение счетчика уменьшается на единицу) и в случае равенства счетчика значению «1», контроллер отсылает последнее слово данных и за ним следует вторая команда записи. После приема второй команды записи микросхема памяти начинает процесс перезаписи принятых данных во внутренний накопитель. Флэш-память тратит значительное время для перезаписи внутреннего буфера в накопитель. На все это время память становится недоступной. Имеется возможность анализировать сигнал готовности посредством чтения регистра состояния.

19.4 Типичные процедуры работы с NAND флэш-памятью фирмы Samsung

19.4.1 Сброс машины состояния флэш-памяти

– **IO_CFG =** K, устанавливаем временные параметры соответствующие подключенной микросхеме. Если тип микросхемы неизвестен, используем максимальные значения временных параметров (т.е. худший случай)

– **NAND_CFG =** 0x3000, устанавливаем пропуск фаз передачи адреса и чтения данных, число команд чтения равно единице.

– **RD_CFG =** 0xFF, определяем первую команду чтения (COMR1).

– **CR =** 0x0301. Устанавливаем чтение байта.

– **CNTR =** 1, инициируем процесс чтения.

– происходит фиктивное чтение байта из флэш-памяти. Поскольку фазы адреса и данных будут пропущены, переданы будут только команды **FFh** во флэш-память. Тем не менее, нужно помнить, что байт данных (неизвестное значение) все равно будет

загружен во внутренний буфер, и этот байт нужно вычитать, чтобы привести указатели в исходное положение.

19.4.2 Чтение ID

Согласно спецификации, для чтения ID необходимо передать команду **90h**, затем один байт адреса (значение 0) и затем прочитать два байта ID. Это можно сделать следующим образом:

– здесь и далее предполагаем, что **IO_CFG** был определен ранее

– **RD_CFG =** 0x90, определяем первую команду чтения (COMR1).

– **NAND_CFG =** 0, устанавливаем передачу только одного байта адреса, число команд чтения равно единице.

– **CR =** 0x0001. Устанавливаем чтение слова.

– **AR =** 0, адрес равен нулю

– **CNTR =** 1, инициируем процесс чтения.

– выполняется чтение слова по адресу 0. Прочитанные четыре байта и есть ID.

19.4.3 Чтение регистра состояния

Согласно спецификации, для чтения состояния необходимо передать команду **70h**, а затем прочитать байт состояния, т.е.:

– здесь и далее предполагаем, что **IO_CFG** был определен ранее

– **RD_CFG =** 0x70, определяем первую команду чтения (COMR1).

– **NAND_CFG =** 0x1000, устанавливаем пропуск адреса, число команд чтения равно единице.

– **CR =** 0x0301. Устанавливаем чтение байта.

– **CNTR =** 1, инициируем процесс чтения.

– Прочитанный байт и есть регистр состояния. При этом необходимо помнить, что перед чтением данных из внутреннего буфера необходимо убедиться в их наличии. Для этого необходимо сканировать регистр состояния SR и ожидать установки бита RFS в единицу. Это будут сигнализировать о том, что операция завершилась и данные готовы. При чтении регистра состояния внешней памяти необходимо помнить, что если память занята (например, производит стирание блока), то она может притормозить контроллер посредством сигнала RnB. В этом случае перед чтением регистра состояния лучше выключить анализ RnB для выбранного банка памяти.

19.4.4 Очистка (erase) блока флэш-памяти

Согласно спецификации, для стирания блока флэш-памяти необходимо передать команду **60h**, передать адрес блока, пропустить запись данных и в заключение передать команду **d0h**. Таким образом, последовательность действий может быть следующей:

– **RD_CFG =** 0xd060, определяем первую(COMR1) и вторую (COMR2) команды чтения.

– **NAND_CFG =** 0x2248, устанавливаем пропуск данных, число команд чтения равно двум, передачу только адреса для выбора блока.

– **AR =** address, записываем в регистр адреса значение соответствующее номеру стираемого блока. Здесь необходимо рассчитать какие именно биты адреса регистра AR используются для указания номера блока. Если размер страницы указан как 2 Кбайт, то биты AR[8:0] будут использоваться для адресации байта внутри страницы (плюс дополнительно два нулевых бита). Биты AR[29:9] будут указывать на номер страницы. Стираемый блок может состоять из нескольких страниц. Если такое число страниц равно, например, 64, то передаваемые биты AR[14:9] безразличны. Учитываются только биты AR[29:15].

– **CR =** 0x0301. Устанавливаем чтение байта.

– **CNTR =** 1, инициируем процесс чтения.

– выполняется фиктивное чтение байта. Это приводит к передаче во флэш-память требуемой последовательности байт. После этого можно анализировать флаг готовности флэш-памяти путем чтения регистра состояния памяти или путем анализа флага RnB регистра состояния SR. Также не забываем о прочитанном байте в буфере.

–

19.4.5 Запись во флэш-память в последовательном режиме

Согласно спецификации, для записи данных необходимо передать команду **80h**, передать адрес, записать данные последовательно в текущую страницу, после этого необходимо передать команду **10h**, которая инициирует процесс программирования. Таким образом:

– **WR_CFG =** 0x100080, определяем первую команду записи (COMW1=80h), вторая команда записи не используется, третья команда записи (COMW3=10h).

– **NAND_CFG =** 0x035B, устанавливаем передачу полного адреса, число команд записи равно 2, размер страницы 2 Кбайт.

– **AR =** address, записываем в регистр адреса значение начального адреса слова(!) начиная с которого мы хотим произвести запись блока данных.

– **CR =** 0x0043. Устанавливаем операцию записи слова размером 4 байта и прерывание по достижению счетчика значения ноль.

– **CNTR =** К, инициируем процесс записи К*4 байт данных.

– выполняем запись последовательного блока данных во внутренний буфер, анализируя при этом его доступность для записи. Как только мы записали последнее слово в буфер, переходим к ожиданию прерывания о завершении операции записи.

–

19.4.6 Чтение из флэш-памяти в последовательном режиме

Согласно спецификации, для чтения данных необходимо передать команду **00h**(**01h**, **50h**), передать адрес, прочитать данные последовательно. Таким образом:

– **RD_CFG =** 0x3000, определяем команды чтения (COMR1=00h, COMR2=30h).

– **NAND_CFG =** 0x035B, устанавливаем передачу полного адреса, число команд записи равно двум, размер страницы 2 Кбайт.

– **AR =** address, записываем в регистр адреса значение начального адреса слова(!) начиная с которого мы хотим произвести чтение блока данных.

– **CR =** 0x0001. Устанавливаем операцию чтения слова размером равным четырем.

– **CNTR =** К, инициируем процесс чтения К*4 байт данных.

– переходим к анализу флага состояния RFS буфера данных. Как только он устанавливается в «1», производим чтение данных. После прочтения всех данных завершаем операцию.

19.5 Рекомендации по организации работы с прерываниями

Для корректной работы NAND контроллера с контроллером DMA нужно будет строго соблюдать последовательность действий. При этом данная последовательность будет разной при чтении и при записи. Исходное состояние – все выключено.

При чтении

– программируем канал DMA. После включения он ждет импульса от NAND контроллера.

– программируем NAND контроллер и включаем обмен.

– при чтении данных из памяти в FIFO генерируется запрос к контроллеру DMA (или прерывание процессору) о том, что можно выполнить дальнейшую пересылку.

– Контроллер DMA пересылает данные.

– после чтения последнего слова NAND-контроллер может сгенерировать прерывание для процессора. Однако это не очень разумно, т.к. данные еще не переместились в память. Поэтому при чтении лучше использовать прерывание от канала DMA.

– запрос прерывания от контроллера DMA говорит о том, что все операции завершены.

При записи

– программируем канал DMA. После включения он ждет импульса от NAND-контроллера.

– программируем NAND-контроллер и включаем обмен, сразу идет запрос к контроллеру DMA.

– контроллер DMA пересылает данные в FIFO и они передаются в память.

– после пересылки последнего слова контроллер DMA может генерировать прерывание. Но при записи лучше использовать прерывание от NAND. Оно будет говорить о том, что данные уже переданы в память.

20 Интерфейс ЖКИ

Возможно подключение ЖКИ (LCD-панелей, тип TFT) к процессору. Параметры контроллера позволяют поддерживать панели различных разрешений.

Структура контроллера LCD-панели представлена на рисунке 98.

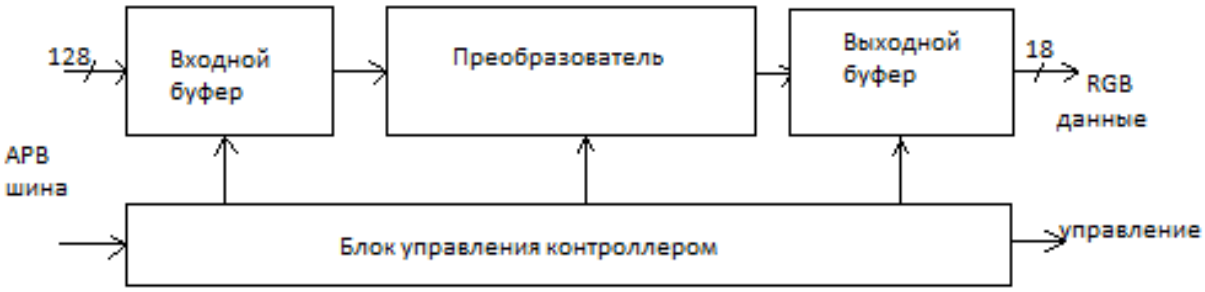


Рисунок 98 – Структура контроллера

Назначение внешних выводов контроллера приведено в таблице 164.

Таблица 164 – Назначение внешних выводов контроллера

Обозначение вывода	Назначение вывода контроллера	Тип	Функциональное назначение
PB[4]	LC_B[0]	O	Интерфейс ЖКИ. Бит синего цвета 0
PB[5]	LC_B[1]	O	Интерфейс ЖКИ. Бит синего цвета 1
PB[6]	LC_B[2]	O	Интерфейс ЖКИ. Бит синего цвета 2
PB[7]	LC_B[3]	O	Интерфейс ЖКИ. Бит синего цвета 3
PB[8]	LC_B[4]	O	Интерфейс ЖКИ. Бит синего цвета 4
PB[9]	LC_B[5]	O	Интерфейс ЖКИ. Бит синего цвета 5
PB[10]	LC_G[0]	O	Интерфейс ЖКИ. Бит зеленого цвета 0
PB[11]	LC_G[1]	O	Интерфейс ЖКИ. Бит зеленого цвета 1
PB[12]	LC_G[2]	O	Интерфейс ЖКИ. Бит зеленого цвета 2
PB[13]	LC_G[3]	O	Интерфейс ЖКИ. Бит зеленого цвета 3
PB[14]	LC_G[4]	O	Интерфейс ЖКИ. Бит зеленого цвета 4
PB[15]	LC_G[5]	O	Интерфейс ЖКИ. Бит зеленого цвета 5
PB[16]	LC_R[0]	O	Интерфейс ЖКИ. Бит красного цвета 0
PB[17]	LC_R[1]	O	Интерфейс ЖКИ. Бит красного цвета 1
PB[18]	LC_R[2]	O	Интерфейс ЖКИ. Бит красного цвета 2
PB[19]	LC_R[3]	O	Интерфейс ЖКИ. Бит красного цвета 3
PB[20]	LC_R[4]	O	Интерфейс ЖКИ. Бит красного цвета 4
PB[21]	LC_R[5]	O	Интерфейс ЖКИ. Бит красного цвета 5
PB[22]	LC_T[0]	O	Интерфейс ЖКИ. Дополнительный выход 0
PB[23]	LC_T[1]	O	Интерфейс ЖКИ. Дополнительный выход 1
PB[24]	LC_T[2]	O	Интерфейс ЖКИ. Дополнительный выход 2
PB[25]	LC_T[3]	O	Интерфейс ЖКИ. Дополнительный выход 3
PB[26]	LC_PWM	O	Интерфейс ЖКИ. Выход ШИМ для управления яркостью ЖКИ
PB[27]	LC_DRDY	O	Интерфейс ЖКИ. Сигнал готовности данных для ЖКИ

Обозначение вывода	Назначение вывода контроллера	Тип	Функциональное назначение
PB[28]	LC_VSYNC	O	Интерфейс ЖКИ. Сигнал вертикальной синхронизации ЖКИ
PB[29]	LC_HSYNC	O	Интерфейс ЖКИ. Сигнал горизонтальной синхронизации ЖКИ
PB[30]	LC_CLK	O	Интерфейс ЖКИ. Синхросигнал

Контроллер осуществляет прием входной информации, ее преобразование и выдачу на внешнюю LCD-панель в соответствии с заданным режимом работы. Задание режима работы контроллера осуществляется посредством программирования его регистров. Регистры контроллера LCD-панели подключены к пользовательской периферийной шине и имеют базовый адрес 0x8000_0160. Перечень регистров приведен в таблице 165.

Таблица 165 – Регистры контроллера

Номер	Название	Функция
0	CTRL	Регистр управления
1	STATUS	Регистр состояния
2	HTIM	Fpline (старт, стоп)
3	VTIM	Fpframe (старт, стоп)
4	HVLEN	Размер экрана
5	HDxTIM	Горизонтальная выдача данных (старт, стоп). Дополнительное окно
6	VDxTIM	Вертикальная выдача данных (старт, стоп). Дополнительное окно
7	Vsize	Размер видеобуфера
8	FON	Данные для заполнения фоновой области
9	PXDV	Делитель пиксель клока
10	HDTIM	Горизонтальная выдача данных (старт, стоп). Основное окно
11	VDTIM	Вертикальная выдача данных (старт, стоп). Основное окно
12	PANEL_CFG	Регистр конфигурации панели
13	PWM_CR	Регистр управления ШИМ
14	SLP_PERIOD	Регистр задания интервала сна
15	-	
16	TIM_GP0	Управление сигналом GPIO_0
17	TIM_GP1	Управление сигналом GPIO_1
18	TIM_GP2	Управление сигналом GPIO_2
19	TIM_GP3	Управление сигналом GPIO_3
20	EXT_MEM_ADDR	Начальный адрес внешней памяти для выделенного канала DMA

Управление внешней LCD-панелью осуществляется посредством набора внешних выходов. Внешние выводы контроллера состоят из 26-разрядного порта общего

назначение и выхода синхронизации PIX_CLK (или FPSHIFT). Назначение выводов порта общего назначение приведено в таблице 166.

Таблица 166 – Внешние выводы управления панелью

Разряды порта	Название	Функция
17:0	fpdat	Шина данных пикселей
18	Gpio_0	Дополнительное управление панелью
19	Gpio_1	Дополнительное управление панелью
20	Gpio_2	Дополнительное управление панелью
21	Gpio_3	Дополнительное управление панелью
22	pwm	Выход ШИМ
23	drdy	Готовность данных
24	fpframe	Импульс начала фрейма
25	fpline	Импульс начала линии
26	fpshift	Пиксель-клок

Входной буфер имеет входную шину данных разрядностью 128 бит для загрузки данных. Емкость буфера составляет пять 128-разрядных слов. Входной буфер формирует запрос к контроллеру DMA на прием данных, если он не заполнен. Из входного буфера данные поступают на преобразователь порциями по 32 бита. Модуль преобразователя осуществляет формирование информации о RGB-пикселе на основании полученной информации и заданных блоком управления параметрах преобразования. На выходе преобразователя всегда данные RGB пикселя разрядностью 18 бит. Каждая цветовая компонента имеет разрядность 6 бит. Эти данные поступают в выходной буфер. Емкость выходного буфера составляет 256 пикселей. При формировании временной диаграммы управления панелью, данные из выходного буфера в нужный момент времени выдаются на внешнюю панель. В задачу преобразователя входит постоянная подкачка данных в выходной буфер по мере его освобождения. Использование выходного буфера позволяет осуществить равномерную выдачу информации на панель, даже в случаях, когда контроллер DMA может быть блокирован на некоторое время более высокоприоритетными обменами.

Регистры контроллера позволяют настроить контроллер на работу с различными типами LCD-панелей. При этом возможно задание, как дополнительных внешних управляющий линий, так и настройка контроллера для работы с выбранным разрешением экрана. Регистры управления позволяют задать количество вертикальных линий панели, а также количество пикселей в линии. Дополнительно задается область выдачи данных, т.к. количество линий и пикселей в линии может превышать количество пикселей, задействованных в отображении информации. В зависимости от требований панели, контроллер может выполнять как непрерывную выдачу данных (автоматический переход на начало повторной выдачи картинки после завершения текущей выдачи), так и выдачу данных по запросу. В последнем случае контроллер после завершения выдачи картинки переходит в спящий режим и начинает новую выдачу по истечению запрограммированного интервала времени или по требованию процессора. Данный

режим может использоваться, только если панель (внешнее устройство-приемник) допускает останов пиксель клока на некоторое время.

Контроллер LCD-панели можно также рассматривать как параллельный синхронный 16-разрядный порт для выдачи информации с настраиваемыми параметрами выдачи данных и генерации сигналов управления.

Типовая временная диаграмма работы контроллера приведена на рисунке 99. Задача контроллера сводится к выдаче на внешние контакты одного фрейма (картины) изображения. Начало картины всегда начинается с выдачи активного уровня FPFRAME (на рисунке активный низкий уровень). Изображение состоит из некоторого множества линий. Начало каждой линии сопровождается активным уровнем FPLINE (на рисунке активный низкий уровень). Информация о пикселях текущей линии выдаются только при активном уровне на линии готовности данных DRDY (на рисунке активный высокий уровень). Значение пикселя на шине данных FPDAT выдается относительно среза (из высокого в низкий) синхросигнала FPSHIFT. Фронтом синхросигнала данные должны приниматься в устройстве приемнике информации.

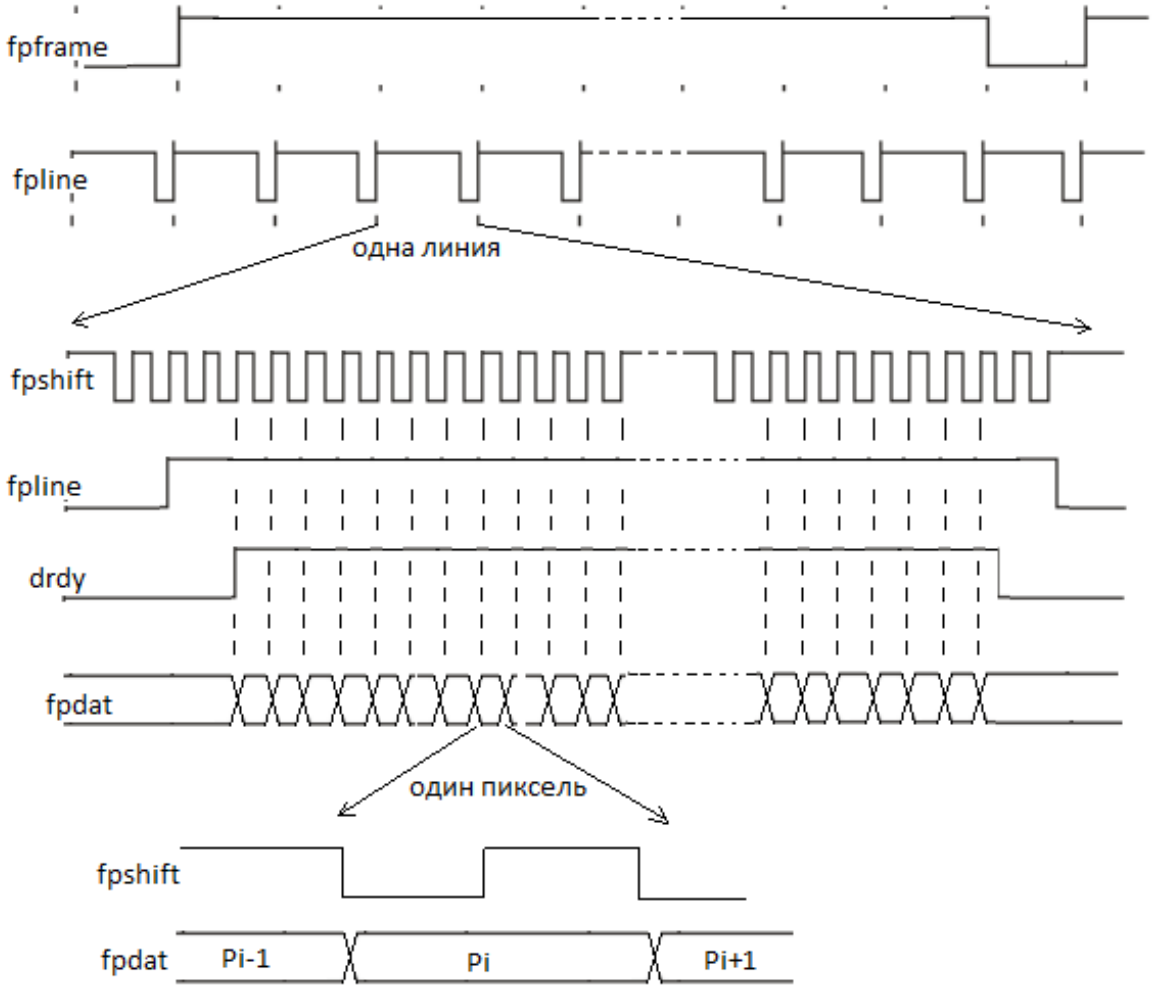


Рисунок 99 – Временная диаграмма работы контроллера

Регистры управления контроллером предназначены для задания различных временных характеристик обмена. После выдачи одного фрейма контроллер повторяет свои действия, но, возможно, уже с другими пиксельными данными.

20.1 Регистр управления CTRL

Назначение разрядов регистра управления приведено в таблице 167.

Таблица 167 – Разряды регистра управления CTRL

Бит	Имя	Функция
0	VEN	Разрешение работы контроллера: 0 – выключен; 1 – включен
1	VIE	Разрешение прерывания после окончания фрейма: 0 – запрещено; 1 – разрешено
2	HIE	Разрешение прерывания после окончания одной линии: 0 – запрещено; 1 – разрешено
3	VBIE	Разрешение прерывания после загрузки всего видеобанка данных в выходной буфер
4	SLPIE	Разрешение прерывания после завершения сна
5	-	
6	XRQEN	Разрешение прерывания после завершения чтения DMA из внешней памяти
8:7	VBL	Количеств бит в пикселе. Зависит от выбранного режима (биты CD). См. таблицу 168
10:9	CD	Количество бит для одного пикселя входной информации: 00 – 8; 01 – 16; 10 – 24; 11 – 32
11	-	
12	HLDM	Останов повторной загрузки видеоданных после завершения выгрузки видеобуфера (1 – разрешено)
13	HLDV	Останов повторной генерации управляющих сигналов панели после завершения выгрузки видеобуфера: 0 – останов запрещен; 1 – останов разрешен
14	-	
15	BL	Активный уровень линии готовности данных DRDY (0 – высокий, 1 – низкий)
16	-	
17	VBGR	1 – BGR режим, 0 – RGB режим
18	-	
19	-	
20	SLP_MODE	1 – спящий режим после выгрузки видеобуфера
21	SLP_PCLK	Стоп делитель пиксель клока в спящем режиме (1)
22	SLP_PXEN	Стоп разрешения работы пиксель клока в спящем режиме
23	SLP_HOLD	Останов повторной генерации управляющих сигналов панели после завершения выгрузки видеобуфера (1 – разрешено)

Бит	Имя	Функция
24	SLP_CLRF	1 – автоматический сброс флагов после задержки спящего режима; 0 – ожидание программного обслуживания интерфейса
25	DMA_2QW	Разрешение режима загрузки двух квадрослов за одну транзакцию DMA: 1 – режим разрешен; 0 – режим запрещен
27:26	–	Зарезервировано
28	W2W_EN	Разрешение использования регистров дополнительной активной области: 0 – запрещено; 1 – разрешено
29	PXP_EN	Разрешение останова пикселькклока в случае если в выходном FIFO нет данных: 0 – запрещено. Будет генерироваться ошибка; 1 – разрешено. Клок продолжит тактирование, когда будут данные

Бит VEN включает контроллер. Этот бит должен быть установлен в единицу, только когда все параметры контроллера определены.

Биты VIE и HIE позволяют задать формирование запроса прерывания после окончания выдачи всей картинки (VIE) или после окончания очередной одной горизонтальной линии (HIE). Данные прерывание имеют скорее отладочную функцию, т.к. их сложно использовать при нормальном функционировании контроллера.

Бит VBIE позволяет сформировать запрос прерывания после завершения загрузки последнего 32-разрядного слова фрейма из входного буфера в преобразователь. Генерация данного прерывания означает завершение чтения всех данных видеофрейма.

Бит SLPIE позволяет сформировать запрос прерывания к процессору после истечения периода сна контроллера.

Биты CD задают количество бит во входном слове, которые используются для кодирования информации о пикселе. Значения 0, 1, 2 и 3 соответствуют 8, 16, 24 и 32 битам.

В случае использования 32 бит, старшие 8 бит отбрасываются. Использование 32 бит равносильно режиму 24 бит, однако информация о каждом пикселе выровнена на границе слова. Из-за этого впустую теряется один байт. Восьмибитный режим может использоваться для режима черно-белого изображения, т.к. в этом случае каждая цветовая RGB-компонента равна значению байта. В случае использования 16-битный режима возможны несколько вариантов кодирования. Эти варианты кодируются с использованием битов VBL. Возможные варианты кодирования приведены в таблице 168. Значение VBL равное нулю (или единице) соответствует стандартному формату RGB565.

Бит DMA_2QW вводит режим загрузки двух квадрослов за одну транзакцию. Также для разрешения этого режима необходимо установить в «1» бит 31 старшего дополнительного регистра приемника каналов DMA 4-7. При использовании данного режима поле «адрес устройства» дополнительного регистра канала DMA должно быть 0x80000180.

Таблица 168 – Режимы формирования пиксельных данных

CD	VBL	byte	R	G	B
11	-	-	Din[23:16]	Din[15:8]	Din[7:0]
10	-	-	Din*[23:16]	Din*[15:8]	Din*[7:0]
00	x0	0	mod3(Din[7:5])	mod3(Din[4:2])	mod2(Din[1:0])
00	x0	1	mod3(Din[15:13]	mod3Din[12:10]	mod2Din[9:8]
00	x0	2	mod3(Din[23:21]	mod3Din[20:18]	mod2Din[17:16]
00	x0	3	mod3(Din[31:29]	mod3Din[28:26]	mod2Din[25:24]
00	x1	0	mod2(Din[7:6]	mod3(Din[5:3]	mod3(Din[2:0]
00	x1	1	mod2(Din[15:14]	mod3(Din[13:11]	mod3(Din[10:8]
00	x1	2	mod2(Din[23:22]	mod3(Din[21:19]	mod3(Din[18:16]
00	x1	3	mod2(Din[31:30]	mod3(Din[29:27]	mod3(Din[26:24]
01	0x	-	{ Din[15:11],000}	{ Din[10:5],00}	{ Din[4:0],000}
01	10	-	{ Din[13:10],0000}	{ Din[8:5],0000}	{ Din[3:0],0000}
01	11	-	{ Din[14:11],0000}	{ Din[9:6],0000}	{ Din[4:1],0000}

mod2(a) = (a[1:0]==2’b11)? 8’hFF: {a[1:0], 000000},
mod3(a) = (a[2:0]==3’b111)? 8’hFF: {a[2:0], 00000}

Из всех приведенных режимов, отметим режим RGB565, как единственный случай, при котором на внешние контакты информация выдается без потерь входной информации. Данный режим позволяет рассматривать LCD-порт как синхронный 16-разрядный порт.

Если используемая LCD-панель допускает кратковременную остановку сигнала синхронизации, полезным будет использование бита PXP_en. Если этот бит установлен, то в случае, когда необходимо выдать новое значение данных и выходной буфер пуст, контроллер остановит сигнал синхронизации панели до момента готовности данных в выходном буфере. Это гарантирует отсутствие сбоев при выдаче информации.

20.2 Регистр состояния STATUS

Регистр STATUS отражает текущее состояние запросов прерываний к процессору.

Таблица 169 – Регистр состояния STATUS

Бит	Имя	Функция
0	-	
1	LUINT	Флаг прерывания при ошибке чтения выходного буфера. Если флаг установлен, то при необходимости выдачи данных, буфер был пуст. Сбрасывается только при записи в данный бит значения 1
2	STA_SLEEP	Таймер сна начал отсчет. Устанавливается в момент перехода контроллера в режим сна
3	FIN_SLEEP	Таймер сна закончил отсчет. Устанавливается в момент завершения режима сна
4	VINT	Флаг завершения выдачи фрэйма

Бит	Имя	Функция
5	HINT	Флаг завершения выдачи линии
6	VBSINT	Флаг завершения чтения данных видеобуфера. Устанавливается при передаче последнего слова данных из входного буфера в преобразователь. Сбрасывается программно или аппаратно
7	-	Зарезервировано
19-8		
20	SLEEP_EXE	Режим сна (если 1). Только чтение
31-21		

Флаги регистра устанавливаются аппаратно. Сброс флагов может выполняться программно путем записи в необходимый бит значения 1. Также флаги могут сбрасываться аппаратно в момент выхода контроллера из режима сна. Флаг LUINT может быть сброшен только программно.

20.3 Регистр управления сигналом fpline (HTIM)

Этот регистр осуществляет управление импульсом начала линии и задает смещение импульса относительно начала строки, а также ширину импульса и его активный уровень. Единицей измерения является длительность PX_CLK сигнала.

Таблица 170 – Регистр состояния управления сигналом fpline (HTIM)

Бит	Имя	Функция
9:0	HPS	FPLINE начальная позиция (hps+1)
15	HPL	FPLINE активный уровень (0 – высокий, 1 – низкий)
25:16	HPW	FPLINE конечная позиция (hpw+1)

20.4 Регистр управления сигналом fpframe (VTIM)

Этот регистр осуществляет управление импульсом начала фрейма и задает смещение импульса относительно начала фрейма, а также ширину импульса и его активный уровень. Единицей измерения является одна строка.

Таблица 171 – Регистр управления сигналом fpframe (VTIM)

Бит	Имя	Функция
9:0	VPS	FPFRAME начальная позиция
15	VPL	FPFRAME активный уровень: 0 – высокий; 1 – низкий
25:16	VPW	FPFRAME конечная позиция (vpw+1)

20.5 Регистр управления размером экрана (HVLEN)

Этот регистр задает количество пикселей в одной линии и количество линий на экране. Не все пиксели линии могут быть отображаемыми на экране. Также не все линии

могут быть отображаемыми на экране. Регистр HVLEN задает общее количество пикселей и линий, которое включает всю информацию (отображаемую и нет).

Таблица 172 – Регистр управления размером экрана (HVLEN)

Бит	Имя	Функция
6:0	HT	Размер по горизонтали ((ht + 1) • 8)
25:16	VT	Размер по вертикали (vt + 1)

20.6 Регистр размера видео буфера (VSIZE)

Регистр задает количество 32-разрядных слов, которые образуют видеобуфер. Контроллер использует это число, чтобы отследить момент окончания загрузки данных и сформировать запрос на прерывание. Момент формирования запроса прерывания соответствует чтению последнего слова видеобуфера из входного FIFO в преобразователь.

Таблица 173 – Регистр размера видео буфера (VSIZE)

Бит	Имя	Функция
19:0	VSIZE	Размер видеобуфера в 32-бит словах + 1
31:20	-	

После чтения последнего слова в преобразователь, происходит

– сброс бита 5 регистра управления (не используется).

– установка бита 6 регистра состояния (флаг завершения чтения видеобуфера данных).

Контроллер всегда используется совместно с выбранным для него каналом DMA. В канале DMA также задается число слов для передачи и запрос прерывания от канала DMA может быть более информативным, чем запрос прерывания от контроллера LCD на обслуживание видеобуфера.

Если бит 12 (HLDM) регистра управления установлен, после установки бита 6 регистра состояния, входное FIFO будет постоянно посылать в преобразователь признак того, что оно пусто. При этом входной буфер может формировать запрос к контроллеру DMA и если канал работает, то во входной буфер может быть загружено до пяти квадрослов данных. Данные из входного буфера не будут поступать в преобразователь до тех пор, пока бит 6 регистра состояния не будет сброшен. Бит может быть сброшен программно или аппаратно при выходе из режима сна.

20.7 Делитель для сигнала синхронизации панели (PXDV)

Контроллер LCD использует для своей работы сигнал синхронизации периферийной шины SOC_CLK. Данный синхросигнал используется для формирования сигнала синхронизации пикселей PX_CLK.

$$\text{PX\_CLK} = \frac{\text{SOC\_CLK}}{\text{P\_div} + 1}. \quad (8)$$

Таблица 174 – Делитель для сигнала синхронизации панели (PXDV)

Бит	Имя	Функция
7:0	P_DIV	Значение делителя
8	EN_DIV	Включение делителя: 0 – выключен; 1 – включен

20.8 Управление горизонтальной активной областью панели (HDTIM)

Этот регистр осуществляет управление видимой областью линии и задает смещение данной области относительно начала строки, а также ширину области. Единицей измерения является длительность PX_CLK сигнала. Как отмечалось ранее, регистр HVLEN задает общее количество пикселей в строке. Счетчик пикселей строки будет изменяться от 0 до Np, где Np это количество пикселей в строке. Регистр HDTIM задает начальный номер счетчика, после которого начинается выдача пиксельной информации, а также номер счетчика после которого прекращается выдача информации. Таким образом, будет задана активная область горизонтальной строки.

Таблица 175 – Управление горизонтальной активной областью панели (HDTIM)

Бит	Имя	Функция
9:0	HDPS	Начальная позиция активной области (hdps+1)
25:16	HDPE	Конечная позиция активной области (hdpe+1)

20.9 Управление вертикальной активной областью панели (VDTIM)

Этот регистр осуществляет управление видимой областью экрана и задает смещение данной области относительно начала экрана, а также ширину области. Единицей измерения является одна линия. Данный регистр задает номер линии, с которой начинается отображение информации на экране, а также номер линии, с которой начинается неотображаемая область. На рисунке 100 показаны функции, выполняемые регистрами при отображении информации.

Таблица 176 – Управление вертикальной активной областью панели (VDTIM)

Бит	Имя	Функция
9:0	VDPS	Номер начальной отображаемой линии
25:16	VDPE	Номер конечной отображаемой линии

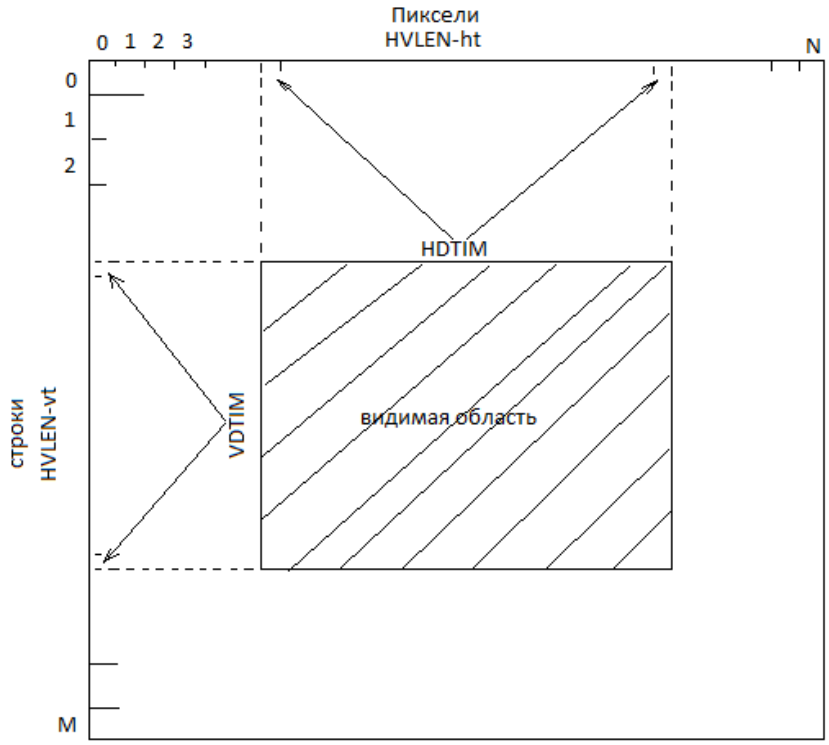


Рисунок 100 – Параметры отображаемой информации

20.10 Управление дополнительной горизонтальной активной областью панели (HDxTIM)

Регистр HDxTIM осуществляет управление дополнительной видимой областью линии и задает смещение данной области относительно начала строки, а также ширину области. Единицей измерения является длительность PX_CLK сигнала. Регистр HDxTIM используется совместно с регистром HDTIM и позволяет задать пересечение двух активных областей. Информация будет выводиться только в ту область, которая принадлежит обоим множествам. В область, которая принадлежит множеству HDTIM, но не принадлежит множеств HDxTIM, будет выводиться значение фонового регистра FON.

Таблица 177 – Управление дополнительной горизонтальной активной областью панели (HDxTIM)

Бит	Имя	Функция
9:0	HXDPS	Начальная позиция активной области (HXDPS+1)
25:16	HXDPE	Конечная позиция активной области (HXDPE+1)

20.11 Управление дополнительной вертикальной активной областью панели (VDxTIM)

Регистр VDxTIM осуществляет управление видимой областью экрана и задает смещение данной области относительно начала экрана, а также ширину области. Единицей измерения является одна линия. Регистр VDxTIM используется совместно с регистром VDTIM и позволяет задать пересечение двух активных областей. Информация будет выводиться только в ту область, которая принадлежит обоим множествам. В область, которая принадлежит множеству строк VDTIM, но не принадлежит множеству VDxTIM, будет выводиться значение фонового регистра FON.

Таблица 178 – Управление дополнительной вертикальной активной областью панели (VDxTIM)

Бит	Имя	Функция
9:0	VXDPS	Номер начальной отображаемой линии
25:16	VXDPE	Номер конечной отображаемой линии

Регистры HDxTIM и VDxTIM могут быть использованы в случае, когда размер отображаемой информации значительно меньше разрешения панели. В этом случае данная функция позволяет значительно сократить трафик шины, т.к. передаваться будет только реально отображаемая информация.

20.12 Регистр FON

При использовании регистров задания дополнительного окна, данные выводятся только в область пересечения двух множеств: основного и дополнительного окон. В область основного окна, которая не входит в область дополнительного окна выводятся данные из регистра FON.

Таблица 179 – Регистр FON

Бит	Имя	Функция
7:0	B_Fon	Данные для B выхода (используются биты 7:2)
15:8	G_Fon	Данные для G выхода (используются биты 15:10)
23:16	R_Fon	Данные для R выхода (используются биты 23:18)

20.13 Регистр конфигурации панели (PANEL_CFG)

Таблица 180 – Регистр конфигурации панели (PANEL_CFG)

Бит	Имя	Функция
5:0	-	
6	FPSHI	Инверсия пиксель клока
8:7	-	
9	DVI_MODE	При записи всегда должен быть 0
10	DVI_CLK	При записи всегда должен быть 0
11	-	
31:12	-	Всегда ноль

20.14 Регистр управления выходом ШИМ (PWM_CR)

Регистр может быть использован для управления яркостью панели.

Таблица 181 – Регистр управления выходом ШИМ (PWM_CR)

Бит	Имя	Функция
0	CLKEN	Разрешение работы (1)
2:1	-	Должен быть 0
3	FRSH	Когда равен 1, на выходе PWM постоянный высокий уровень вне зависимости от работы внутренних счетчиков

Бит	Имя	Функция
7:4	PWM_Dv	Делитель: 0 – 1; 1 – 2; 2 – 4; 3 – 8; …… 0xF – 32768
15:8	DUTY	Длительность активного уровня: 0 – всегда 0; 1 – высокий для 1 периода; 2 – высокий для 2 периодов; …… 0xFF – высокий для 255 периодов
23:16	RELOAD	8-бит верхнее значение. Когда счетчик достигнет это значение, он загрузит ноль и начнет отсчет заново.

В качестве базовой частоты PWM модуля используется SOC_CLK синхросигнал. Предварительно SOC_CLK может быть поделен на число, задаваемое битами PWM_DV. После чего мы будем иметь рабочую частоту PWM счетчика. Счетчик осуществляет увеличение на единицу каждый такт своей рабочей частоты. Если значение счетчика меньше значения «duty», на внешнем выходе присутствует высокий уровень. В противном случае – низкий. При достижении счетчиком значения RELOAD, он сбросит свое значение в ноль и начнет счет заново.

20.15 Регистр управления сигналом GPIO_0, 1, 2, 3

Регистр позволяет задать требуемую временную диаграмму на выводе GPIO_0.

Таблица 182 – Регистр управления сигналом GPIO_0, 1, 2, 3

Бит	Имя	Функция
9:0	GP0_ST	GPIO_0 начальная позиция (+1)
15	HPL	GPIO_0 активный уровень: 0 – высокий; 1 – низкий
25:16	GP0_SP	GPIO_0 конечная позиция (+1)

Если значение начала импульса совпадает с его окончанием, сигнал изменяет свое значение на противоположное. Единицей измерений является пикселькклок.

Регистры GPIO_1, 2, 3 аналогичны по функциям регистру GPIO_0. Регистры позволяют сформировать, при необходимости, дополнительные управляющие сигналы.

20.16 Организация «спящего режима»

В обычном режиме работы, после включения контроллера, выполняется процесс генерации импульсов управления панелью, а также выдача информации. После выдачи одного фрейма данных (одной картины) контроллер обычно приступает к повторной

выдачи новой или старой (если видеобуфер не обновлялся) информации. При этом ряд панелей требуют непрерывной подачи сигнала синхронизации и регенерации данных. Однако существуют панели, которые после выдачи одного фрейма данных, допускают останов синхронизации и некоторый перерыв в подаче информации. В этом случае контроллер после выдачи данных может перейти в спящий режим на некоторое время. Интервал, в течение которого контроллер не выполняет передачи данных, задается с помощью регистра SLP_PERIOD. По истечении заданного периода контроллер возобновит чтение информации из видеобуфера и выдачу данных на внешний интерфейс. Переход в режим сна выполняется при завершении формирования всех управляющих сигналов одного фрейма и при установленном бите Slp_MODE.

В режиме сна контроллер:

– Загружает в счетчик интервала сна значение регистра SLP_PERIOD и начинает обратный отсчет.

– Если бит SLP_PCLK установлен в «1», выполняет останов делителя пикселькклока и таким образом останавливает синхронизацию всех внутренних счетчиков контроллера.

– Если бит SLP_PXEN установлен в «1», запрещает работу всех внутренних счетчиков, даже если делитель пикселькклока работает (т.е. бит SLP_PCLK равен нулю). В этом случае активным может быть только выход fpshift (если бит SLP_PCLK равен нулю). Все другие выходы управления остановлены.

– Если бит SLP_HOLD установлен в «1», запрещает изменение счетчика пикселей и устанавливает его в нулевое значение. Аналогичен действию бита HLDV.

– Если бит SLP_CLRF установлен в «1», при завершении интервала сна, контроллер выполнит очистку всех флагов, которые были установлены после выгрузки предыдущего фрейма.

Когда счетчик интервала сна уменьшает свое значение до нуля, происходит выход из режима сна и контроллер возобновляет генерацию управляющих сигналов и выдачу информации. При реализации режима сна и установленном бите SLP_CLRF, обязательно должен использоваться бит 12 (HLDM). Бит HLDM должен быть установлен в «1». Если бит SLP_CLRF установлен в «1», при выходе из режима сна также инициализируется счетчик количества слов буфера видеоданных. Однако если бит HLDM не будет установлен, то это может вызвать установку некорректного значения, т.к. к моменту окончания сна некоторое количество слов данных может быть прочитано (если канала DMA включен), обработано и размещено в выходном буфере.

Возможность использования спящего режима при работе контроллер, существенно снижает трафик к памяти процессора и сокращает потребление системы.

21 Интерфейс к аудиокодеку AC97/I2S

Специальный интерфейс позволяет осуществить вывод аудио данных на внешний аудиокодек типа AC97. Также данный интерфейс может работать по протоколу I2S шины и ее различных вариантов. Интерфейс имеет встроенные FIFO и может работать с контроллером DMA.

В микросхеме реализовано два контроллера этих интерфейсов. Назначение внешних выводов интерфейсов приведено в таблице 183.

Таблица 183 – Назначение внешних выводов интерфейсов IIS#/AC97_#

Обозначение вывода	Назначение вывода контроллера	Тип	Функциональное назначение
PA[13]	SSI0_TCLK/AC97_0_CLK	I/O	SSI0. Синхросигнал передатчика AC97_0. Синхросигнал
PA[14]	SSI0_TFS/AC97_0_RESET	O	SSI0. Строб передатчика AC97_0. Сброс
PA[15]	SSI0_TXD/AC97_0_SDO	O	SSI0. Выход данных AC97_0. Выход данных
PA[16]	SSI0_RCLK	I	SSI0. Синхросигнал приемника
PA[17]	SSI0_RFS/AC97_0_SYNC	I/O	SSI0. Импульс синхронизации AC97_0.Синхросигнал
PA[18]	SSI0_RXD/AC97_0_SDI	I	SSI0. Вход данных AC97_0. Вход данных
PA[19]	SSI1_TCLK/AC97_1_CLK	I/O	SSI1. Синхросигнал передатчика AC97_1. Синхросигнал
PA[20]	SSI1_TFS/AC97_1_RESET	O	SSI1. Строб передатчика AC97_1. Сброс
PA[21]	SSI1_TXD/AC97_1_SDO	O	SSI1. Выход данных AC97_1. Выход данных
PA[22]	SSI1_RCLK	I	SSI1. Синхросигнал приемника
PA[23]	SSI1_RFS/AC97_1_SYNC	I/O	SSI1. Импульс синхронизации AC97_1.Синхросигнал
PA[24]	SSI1_RXD/AC97_1_SDI	I	SSI1. Вход данных AC97_1. Вход данных

21.1 Регистры интерфейса AC97/I2S

Регистры интерфейса AC97/I2S подключены к пользовательской периферийной шине и имеют базовый адрес 0x8000_0200 (интерфейс 0) и 0x8000_0220 (интерфейс 1).

Таблица 184 – Регистры интерфейса

Номер	Название	Режим	Значение после сброса	Описание
0	SICR0	R/W	0	Регистр управления 0
1	SINT	R	0	Регистр запросов прерываний
2	SICR2	R/W	0	Регистр управления 2

Номер	Название	Режим	Значение после сброса	Описание
3	SISR	R	0	Регистр состояния после маскирования
4	SIRSR	R	0	Регистр состояния до маскирования
5	SIIER	R/W	0	Регистр разрешения прерывания
6	SIIDR	R/W	0	Регистр запрещения прерывания
7	SIICR	R/W	0	Регистр сброса флагов запросов прерывания
8	SIADR	R	0	Регистр-буфер аудио данных
9	SIMDR	R	0	Регистр-буфер модемных данных
10	ACCAR	R/W	0	AC канал, регистр адреса команды
11	ACCDR	R/W	0	AC канал, регистр данных команды
12	ACSAR	R	0	AC канал, регистр адреса состояния
13	ACSDR	R	0	AC канал, регистр данных состояния
14	ACGDR	R/W	0	AC канал, регистр данных GPIO
15	ACGSR	R	0	AC канал, регистр состояния GPIO
16	I2S_T_CR	R/W	0	Регистр управления передатчиком I2S
17	I2S_R_CR	R/W	0	Регистр управления приемником I2S
…	-			
28	SICR3	R/W	0	Регистр управления 3

21.1.1 Регистр управления SICR0

Регистр используется только для управления режимом работы AC97. Назначение разрядов регистра приведено в таблице 185.

Таблица 185 – Биты регистра управления SICR0

Бит	Имя	Функция
0	ENB	Включение режима AC97. Включен (1) или выключен (0)
1	-	Для включения режима AC97 должен быть равен 0. Если равен 1 – запрещает включение режима AC97.
2	MODEN	Модемный кодек присутствует (1)
3	AMCCH	AMC кодек присутствует (1)
4	RST	Запись 1 вызывает сброс внешнего кодека
5	LPBK	Тестовый режим AC97 (когда равен 1)
6	BCKD	Источник синхронизации: 0 – используется внешний источник синхронизации; 1 – используется внутренний источник (биты делителя REQLP)
7	MONO_DA	0 – стерео выход; 1 – моно выход
8	BIT8_DA	0 – 16 бит выход; 1 – 8 бит выход
9	MONO_AD	0 – стерео вход; 1 – моно вход
10	BIT8_AD	0 – 16 бит вход; 1 – 8 бит вход

Бит	Имя	Функция
11	TO_EN	Разрешение детектирования отсутствия клока аудиокодека. Если установлен в 1, осуществляется проверка отсутствия синхронизации на входе BITCLK (либо частота очень низкая).
12	DIV_en	Разрешение делителя клока AC97. Когда равен 1 разрешает включение внутреннего делителя клока. Значение делителя задается полем REQLP.
31:13	-	Не используются

Возможно подключение только аудио кодека, раздельных аудио и модемного кодеков, совмещенного AMC кодека.

В тестовом режиме выход данных соединен с входом данных.

21.1.2 Регистр запросов прерываний SINT

Биты регистра запросов прерываний есть логическое произведение между соответствующими битами регистра состояний SIRSR и регистра разрешения прерываний SIIER. Назначение разрядов регистра приведено в таблице 186.

Таблица 186 – Биты регистра запросов прерываний SINT

Бит	Имя	Функция
0	DTD	Прерывание по завершению передачи данных AC97
1	RDD	Прерывание по завершению приема данных AC97
2	GTD	Прерывание по завершению передачи данных GPIO
7:3	-	
8	ATFS	Прерывание буфера передатчика аудио данных
9	ARFS	Прерывание буфера приемника аудио данных
10	ATUR	Прерывание по обнаружению ошибки в буфере передатчика аудио данных (отсутствие данных в момент передачи)
11	AROR	Прерывание по переполнению буфера приемника аудио данных
13:12	-	
14	MTFS	Прерывание буфера передатчика модемных данных
15	MRFS	Прерывание буфера приемника модемных данных
16	MTUR	Прерывание по обнаружению ошибки в буфере передатчика модемных данных (отсутствие данных в момент передачи)
17	MROR	Прерывание по переполнению буфера приемника модемных данных
18	RSTO	Прерывание по таймауту внешнего кодека
21:19	-	
22	RESU	Прерывание по возобновлению работы
23	GINT	Прерывание по изменению состояния GPIO
31:24	-	

21.1.3 Регистр управления SICR2

Регистр используется только для управления режимом работы AC97. Поля REQLP, EREC и ERPL используются в режиме I2S. Назначение разрядов регистра приведено в таблице 187.

Таблица 187 – Биты регистра управления SICR2

Бит	Имя	Функция
0	EREC	Разрешение записи (1)
1	ERPL	Разрешение воспроизведения (1)
2	EINC	Разрешение приема модемных данных (1)
3	EOUT	Разрешение передачи модемных данных (1)
4	EGPIO	Разрешение приема GPIO данных (1)
5	WKUP	Запись 1 вызывает инициализацию процедуры пробуждения кодека.
6	DRSTO	Если 1 – выключает AC-link status read time-out function
22:7	REQLP	Коэффициент деления для формирования синхросигнала в режиме I2S и AC97
31:23	-	Не используются

21.1.4 Регистр управления SICR3

Регистр используется только для управления режимом работы AC97. Назначение разрядов регистра приведено в таблице 188.

Таблица 188 – Биты регистра управления SICR3

Бит	Имя	Функция
0	COMSEL	Разрешение вторичного кодека (1)
2:1	C2ID	ID вторичного кодека
5:3	-	Биты общего назначения
31:6	-	Не используются

21.1.5 Регистр состояния SIRSR и SISR

Регистры отражают состояние внутренних ресурсов интерфейса. Регистр SIRSR отражает состояние бит до маскирования с SIIER. Регистр SISR отражает состояние после маскирования с SIIER, кроме бит 27:24, 21:19, 13:12, 7:6 и 4, которые не маскируются и всегда отражают состояние соответствующих бит SIRSR. Бит 5 (ATNE) регистра SISR всегда равен 0.

Регистры используется только для управления режимом работы AC97. Режим I2S использует буфер аудио данных для своей работы. В связи с этим биты, отражающие состояние буфера аудио данных, могут быть использованы в режиме I2S. Назначение разрядов регистра приведено в таблице 189.

Таблица 189 – Биты регистра состояния SISR

Бит	Имя	Функция
0	DTD	Завершение передачи данных. Устанавливается в «1» после завершения передачи в кодек адреса и данных. Сбрасывается в «0» посредством записи значения 1 в нулевой бит регистра SIICR
1	RDD	Завершение чтения данных. Устанавливается в «1» после завершения чтения из кодека адреса и данных. Сбрасывается в «0» посредством записи значения 1 в первый бит регистра SIICR
2	GTD	Завершение передачи GPIO данных. Устанавливается в «1» после завершения передачи в кодек GPIO данных. Сбрасывается в «0» посредством записи значения 1 во второй бит регистра SIICR
3	-	
4	BSY	Занят: 0 – интерфейс в нерабочем состоянии или выключен; 1 – интерфейс передает или принимает фрейм в текущий момент
5	ATNE	Буфер передатчика аудио данных не пуст: 0 – FIFO передатчика пусто; 1 – FIFO передатчика содержит данные
6	ATNF	Буфер передатчика аудио данных заполнен не полностью: 0 – FIFO передатчика аудио данных заполнено полностью; 1 – FIFO передатчика аудио данных заполнено не полностью
7	ARNE	Буфер приемника аудио данных не пуст: 0 – в FIFO приемника нет данных; 1 – в FIFO приемника есть данные
8	ATFS	Запрос буфера передатчика аудио данных: 0 – FIFO передатчика аудио данных заполнено больше чем наполовину; 1 – FIFO передатчика аудио данных заполнено меньше чем наполовину
9	ARFS	Запрос буфера приемника аудио данных: 0 – FIFO приемника аудио данных заполнено меньше чем наполовину; 1 – FIFO приемника аудио данных заполнено больше чем наполовину
10	ATUR	Отсутствие аудио данных для передачи. Устанавливается в «1» при попытке чтения аудио данных из FIFO передатчика в момент, когда оно пусто. Сбрасывается в «0» посредством записи значения 1 в 10 бит регистра SIICR
11	AROR	Переполнение буфера приемника аудио данных. Устанавливается в «1» при попытке записи аудио данных в FIFO приемника в момент, когда оно заполнено полностью. Сбрасывается в «0» посредством записи значения 1 в 11 бит регистра SIICR
12	MTNF	Буфер передатчика модема не заполнен: 0 – FIFO передатчика модемных данных заполнено полностью; 1 – FIFO передатчика модемных данных заполнено не полностью

Бит	Имя	Функция
13	MRNE	Буфер приемника модема не пуст: 0 – в FIFO приемника нет данных; 1 – в FIFO приемника есть данные
14	MTFS	Запрос от буфера передатчика модема: 0 – FIFO передатчика модемных данных заполнено больше чем наполовину; 1 – FIFO передатчика модемных данных заполнено меньше чем наполовину
15	MRFS	Запрос от буфера приемника модема: 0 – FIFO приемника модемных данных заполнено меньше чем наполовину; 1 – FIFO приемника модемных данных заполнено больше чем наполовину
16	MTUR	Отсутствие данных в буфере передатчика модема. Устанавливается в «1» при попытке чтения модемных данных из FIFO передатчика в момент, когда оно пусто. Сбрасывается в «0» посредством записи значения 1 в 16 бит регистра SIICR
17	MROR	Переполнение буфера приемника модема. Устанавливается в «1» при попытке записи модемных данных в FIFO приемника в момент, когда оно заполнено полностью. Сбрасывается в «0» посредством записи значения 1 в 17 бит регистра SIICR
18	RSTO	Флаг состояния «тайм-аут» Устанавливается в «1» при чтении из кодека состояния time-out. Сбрасывается в «0» посредством записи значения 1 в 18 бит регистра SIICR
19	CLPM	Флаг отсутствия частоты синхронизации. Равен единице если частота кодека меньше частоты APB шины больше, чем 25 раз. Детектирует отсутствие частоты синхронизации кодека.
20	CRDYPR	Первичный кодек готов.
21	CRDYSC	Вторичный кодек готов.
22	RESU	Возобновление. Устанавливается в «1» при обнаружении sdata_in & EN_AC & CLPM. Сбрасывается в «0» посредством записи значения 1 в 22 бит регистра SIICR
23	GINT	Флаг изменения состояния GPIO. Устанавливается в «1» при приеме GPIO[0] = 1 Сбрасывается в «0» посредством записи значения 1 в 23 бит регистра SIICR
24	RS3V	Слот 3 принятых данных достоверен Отражает значение бита достоверности 3-го слота.
25	RS4V	Слот 4 принятых данных достоверен
26	RS5V	Слот 5 принятых данных достоверен
27	RS12V	Слот 12 принятых данных достоверен
31:28	-	Не используются

21.1.6 Регистр разрешения прерывания SIIER

Биты регистра разрешают запрос прерывания от соответствующих им разрядов регистра состояния. Запись 1 в соответствующий бит вызывает разрешение прерывания. Запись 0 не вызывает изменения разряда. Регистр используется только для управления режимом работы AC97. Режим I2S использует буфер аудио данных для своей работы. В связи с этим биты, отражающие состояние буфера аудио данных, могут быть использованы и в режиме I2S. Назначение разрядов регистра приведено в таблице 190.

Таблица 190 – Биты регистра разрешения прерывания SIIER

Бит	Имя	Разрешение прерывание если
0	DTD	Завершению передачи данных AC97
1	RDD	Завершению приема данных AC97
2	GTD	Завершению передачи данных GPIO
7:3	-	
8	ATFS	Запросу буфера передатчика аудио данных
9	ARFS	Запросу буфера приемника аудио данных
10	ATUR	Ошибке в буфере передатчика аудио данных (отсутствие данных в момент передачи)
11	AROR	Переполнению буфера приемника аудио данных
13:12	-	
14	MTFS	Запросу буфера передатчика модемных данных
15	MRFS	Запросу буфера приемника модемных данных
16	MTUR	Ошибке в буфере передатчика модемных данных (отсутствие данных в момент передачи)
17	MROR	Переполнению буфера приемника модемных данных
18	RSTO	Флагу таймаута внешнего кодека
21:19	-	
22	RESU	Установке флага возобновления работы
23	GINT	Изменению состояния GPIO
31:24	-	

21.1.7 Регистр запрещения прерывания SIIDR

Биты регистра запрещают запрос прерывания от соответствующих им разрядов регистра состояния. Запись «1» в соответствующий бит вызывает запрещение прерывания. Запись «0» не вызывает изменения разряда. Назначение разрядов регистра аналогично разрядам регистра SIIER.

21.1.8 Регистр адреса команды ACCAR

Регистр используется только для управления режимом работы AC97. При программировании регистров внешнего аудиокодека необходимо задать адрес регистра и записываемые данные. Адрес регистра кодека записывается в регистр ACCAR интерфейса.

Назначение разрядов регистра приведено в таблице 191.

Таблица 191 – Биты регистра адреса команды ACCAR

Бит	Имя	Функция
11:0	-	Не используются
18:12	IX	Индекс управляющего регистра кодека. Бит 12 всегда должен быть равен единице
19	RW	Бит чтения (1) или записи (0)
31:20	-	Не используются

21.1.9 Регистр данных команды ACCDR

Регистр используется только для управления режимом работы AC97. При программировании регистров внешнего аудиокодека необходимо задать адрес регистра и записываемые данные. Данные регистра кодека записывается в регистр ACCDR интерфейса.

Назначение разрядов регистра приведено в таблице 192.

Таблица 192 – Биты регистра данных команды ACCDR

Бит	Имя	Функция
3:0	-	Не используются
19:4	CDR	16 бит записываемые данные
31:20	-	Не используются

21.1.10 Регистр состояния адреса команды ACSAR

Регистр используется только для управления режимом работы AC97. Назначение разрядов регистра приведено в таблице 193.

Таблица 193 – Биты регистра состояния адреса команды ACSAR

Бит	Имя	Функция
11:0	-	Не используются
18:12	SAR	Эхо индекса управляющего регистра кодека, данные которого были прочитаны
31:19	-	Не используются

21.1.11 Регистр состояния данных команды ACSDR

Регистр используется только для управления режимом работы AC97. Назначение разрядов регистра приведено в таблице 194.

Таблица 194 – Биты регистра состояния данных команды ACSDR

Бит	Имя	Функция
3:0	-	Не используются
19:4	SDR	16 бит данных, прочитанных из кодека
31:20	-	Не используются

21.1.12 Регистр данных GPIO ACGDR

Регистр используется только для управления режимом работы AC97. Имеется возможность управлять некоторыми линиями общего назначения кодека. Данные для записи в биты GPIO кодека должны быть предварительно записаны в регистр ACGDR.

Назначение разрядов регистра приведено в таблице 195.

Таблица 195 – Биты регистра данных GPIO ACGDR

Бит	Имя	Функция
3:0	-	Не используются
19:4	GDAT	16 бит данных, которые будут записаны в GPIO кодека
31:20	-	Не используются

21.1.13 Регистр состояния данных GPIO ACGSR

Регистр используется только для управления режимом работы AC97. Назначение разрядов регистра приведено в таблице 196.

Таблица 196 – Биты регистра данных GPIO ACGSR

Бит	Имя	Функция
3:0	-	Не используются
19:4	SDR	16 бит GPIO данных, прочитанных из кодека
31:20	-	Не используются

21.1.14 Регистр аудио данных SIADR

Этот регистр используется для записи аудио отсчетов, передаваемых во внешний аудио-кодек при воспроизведении звука. Этому регистру соответствует FIFO глубиной восемь 32-разрядных слов. Формат записываемых данных зависит от режима МОНО или СТЕРЕО, а также от восьми или 16 бит разрешения:

– для 16-бит стерео данных младшие 16 бит регистра данных соответствуют правому каналу, а старшие – 16 бит левому;

– для 16-разрядного моно режима младшая половина хранит текущий отсчет, а старшая – следующий за ним;

– для восьмиразрядного стерео: биты 7:0 – правый канал N, биты 15:8 – правый канал N+1, биты 23:16 – левый канал N, биты 31:24 – левый канал N+1;

– для восьмиразрядного моно: биты 7:0 – отсчет N, биты 15:8 – отсчет N+1, биты 23:16 – отсчет N+2, биты 31:24 – отсчет N+3.

При чтении регистра данных, данные считываются из FIFO приемника. FIFO приемника имеет глубину восемь 32-битных слов.

Буфер аудио данных используется как в режиме AC97, так и в режиме I2S. Для корректного формирования данных, передаваемых по интерфейсу I2S, нужно задавать соответствующий формат передаваемых данных.

21.1.15 Регистр модемных данных SIMDR

Этот регистр используется для записи данных, передаваемых во внешний модем. Этому регистру соответствует FIFO глубиной восемь 16-разрядных слов. При чтении данные берутся из FIFO приемника модема, имеющего глубину восемь 16-бит слов. Регистр используется только для работы в режиме AC97.

21.2 Режимы работы

21.2.1 Режим работы AC97

Интерфейс имеет возможность поддержки стандартов AC97(только аудио), MC97(только модем), AMC97(аудио и модем совмещены). AC97 и AMC97 могут работать только как первичные (primary), MC97 может быть сконфигурирован как первичный и как вторичный (secondary). Протокол AC-канала определяет полнодуплексный последовательный интерфейс между внешним кодеком и контроллером. Внешний кодек (или контроллер, если нет первичного кодека в системе)

генерирует 12,288 МГц частоту, поступающую в контроллер. В контроллере входная частота делится на 256, получая, таким образом, 48 кГц звуковую частоту. Она используется контроллером и отсылается в кодек как сигнал SYNC.

Передача данных между кодеком и интерфейсом осуществляется в виде фреймов. Один фрейм содержит 13 слотов (рисунок 101). Каждый слот (исключая нулевой) содержит 20 бит последовательных данных. Сигнал SYNC определяет начало фрейма. Он активен (высокий уровень) только в течение нулевого слота (начальный слот). Нулевой слот имеет длительность 16 бит. Хотя каждый слот имеет длительность 20 бит, только 16 бит используется.

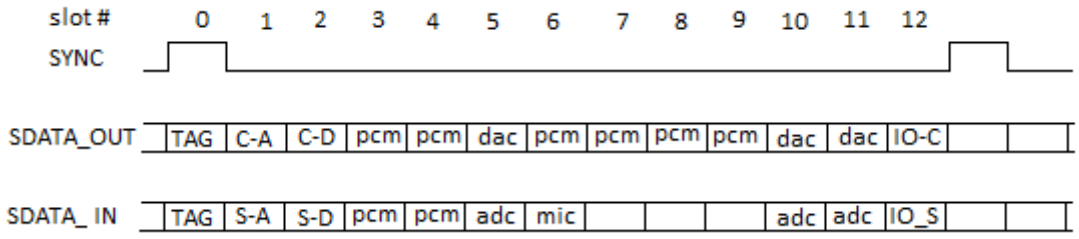


Рисунок 101 – Структура фрейма

Таким образом, один фрейм содержит 256 бит информации (20•12+16). Сигнал SYNC формируется контроллером и имеет частоту следования равную 12,288/256 = 48 кГц. Длительность высокого уровня сигнала SYNC равна 16-ти периодам сигнала синхронизации. Структура тэга фрейма и последовательность бит слотов показана на рисунке 102.

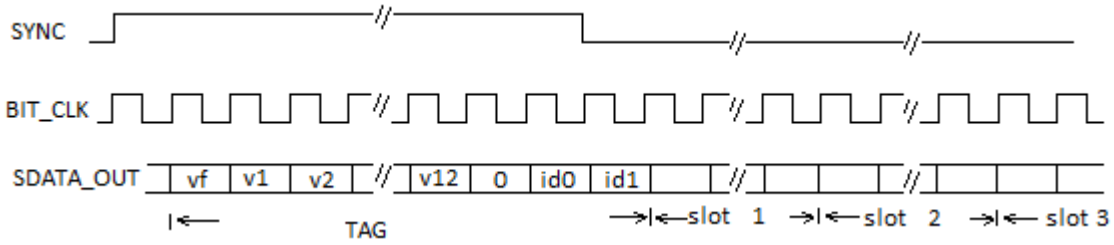


Рисунок 102 – Структура тега

Фрейм имеет бит достоверности – vf. Также имеют биты достоверности и все 12 последующих слотов. Контроллер имеет доступ к регистрам внешнего кодека посредством отсылки адреса регистра в слоте 1 и данных в слоте 2. Аудио данные передаются в 3, 4 и 5 слотах. Имеется возможность управлять GPIO выводами внешнего кодека, а также анализировать их состояние. Для этого используется последний 12 слот. Формат обмена полнодуплексный, т.е. одновременно с передачей по линии SDATA_OUT контроллер принимает данные по линии SDATA_IN. Для ознакомления с подробным описанием каждого слота необходимо изучение спецификации AC97.

Контроллер имеет возможность принимать сообщение о пробуждении (wake-up) внешнего кодека и генерировать соответствующий запрос прерывания.

Назначение выводов микросхемы для AC97 приведено в таблице 197.

Таблица 197 – Назначение внешних выводов для блоков AC97 (интерфейс 0, интерфейс 1)

Обозначение вывода	Назначение вывода для AC97	Тип	Функциональное назначение
PA[13]	AC97_0_CLK	Вход/Выход	Синхросигнал
PA[19]	AC97_1_CLK		
PA[17]	AC97_0_SYNC	Вход/Выход	Синхронизация фрейма
PA[23]	AC97_1_SYNC		
PA[18]	AC97_0_SDI	Вход/Выход	Входные данные
PA[24]	AC97_1_SDI		
PA[15]	AC97_0_SDO	Вход/Выход	Выходные данные
PA[21]	AC97_1_SDO		
PA[14]	AC97_0_RESET	Вход/Выход	Сброс
PA[20]	AC97_1_RESET		

21.2.2 Режим работы I2S

Интерфейс имеет возможность поддержки стандарта I2S.

Назначение выводов микросхемы для I2S приведено в таблице 197.

Таблица 198 – Назначение внешних выводов для блоков I2S (интерфейс 0, интерфейс 1)

Обозначение вывода	Назначение вывода для I2S	Тип	Функциональное назначение
PA[13]	SSI0_TCLK	I/O	Интерфейс SSI0. Синхросигнал передатчика
PA[14]	SSI0_TFS	I/O	Интерфейс SSI0. Начало кадра/выбор канала левый-правый
PA[15]	SSI0_TXD	I/O	Интерфейс SSI0. Данные передатчика
PA[16]	SSI0_RCLK	I/O	Интерфейс SSI0. Синхросигнал приемника
PA[17]	SSI0_RFS	I/O	Интерфейс SSI0. Начало кадра/выбор канала левый-правый
PA[18]	SSI0_RXD	I/O	Интерфейс SSI0. Данные приемника
PA[19]	SSI1_TCLK	I/O	Интерфейс SSI1. Синхросигнал передатчика
PA[20]	SSI1_TFS	I/O	Интерфейс SSI1. Начало кадра/выбор канала левый-правый
PA[21]	SSI1_TXD	I/O	Интерфейс SSI1. Данные передатчика
PA[22]	SSI1_RCLK	I/O	Интерфейс SSI1. Синхросигнал приемника
PA[23]	SSI1_RFS	I/O	Интерфейс SSI1. Начало кадра/выбор канала левый-правый
PA[24]	SSI1_RXD	I/O	Интерфейс SSI1. Данные приемника

Предусмотрены два управляющих регистра:

– Регистр управления передатчиком I2S_T_CR;

– Регистр управления приемником I2S_R_CR.

21.2.2.1 *Регистр управления передатчиком I2S_T_CR*

Таблица 199 – Биты регистра управления передатчиком I2S_T_CR

Бит	Имя	Функция
0	TEN	Разрешение работы (1)
1	MODE	0 – I2S режим; 1 – DSP режим
2	SONY	Выбор стандарта I2S: SONY (1) или Philips (0) Имеет смысл только в режиме I2S
3	MS	Master (1) or slave (0). Бит определяет того, кто генерирует синхросигнал и строб начала фрейма (левый-правый)
9:4	DSS	Длина передаваемых данных – от 1 до 64 бит. Значение в битах DSS всегда на 1 меньше количества бит передаваемых данных.
10	PNIS	Определяет момент приема выбора канала: 0 – прием по фронту (из «0» в «1»); 1 – прием по срезу
11	PNOS	Определяет момент выдачи данных и выбора канала: 0 – выдача по фронту (из «0» в «1»); 1 – выдача по срезу
12	SWHW	1 – указывает на необходимость перестановки 16-разрядных полуслов в одном слове
13	PACKH	Имеет смысл при длине передаваемых данных 16 бит и меньше. 1 – указывает на то, что в одном 32-разрядном слове упакованы два значения. Иначе в одном слове хранится одно значение
14	LRSP	1 и установлен режим DSP – указывает на то, что общее отсылаемое слово состоит из двух симметричных частей (левой и правой) которые размещаются в разных 32-бит словах. 0 – указывает на то, что передаваемая последовательность не делится на части и представляет собой непрерывный поток бит
15		

Регистр передатчика имеет длину 32 бита. Выдача бит происходит всегда «старший бит первый». При передаче данных, длина которых меньше 32 бит, данные необходимо выровнять влево. Если данные упакованы в полуслова, то при передаче данных длиной менее 16 бит их необходимо выровнять влево до границы полуслова.

В режиме I2S передатчик не может передавать данные длиной более 32 бит. При включенном передатчике передача данных инициируется импульсом начала фрейма (DSP режим) или изменением выбора канала (I2S режим). В случае если новые данные отсутствуют в буфере передатчика, происходит повторная передача последних переданных данных.

В режиме DSP можно передавать данные длиной более 32 бит. При этом эти данные могут состоять из равных половин (левой и правой), если LRSP == 1, или представлять собой непрерывный поток бит расположенных в последовательных словах.

Пример передачи четырехразрядных данных в режиме мастера представлен на рисунке 103.

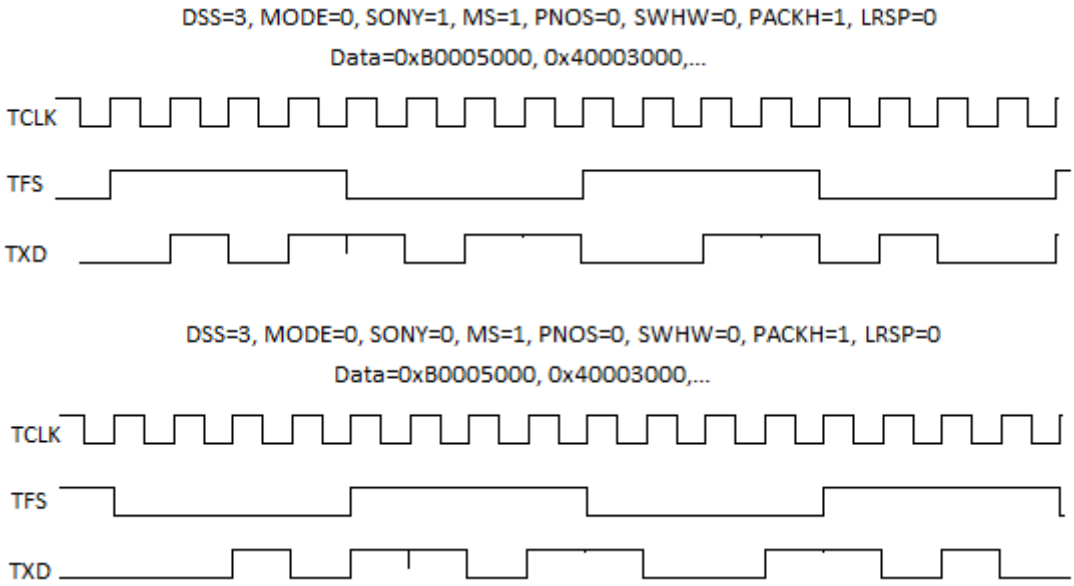


Рисунок 103 – Передача четырехразрядных данных

Передача происходит в режиме I2S (бит MODE=0). Линия TFS указывает на левый или правый канал передачи. Вся информация выдается относительно фронта синхросигнала. Видим, что данные упакованы по 16 бит в одно 32-разрядное слово и выравнены по направлению к старшему биту. Основное отличие между режимами SONY и Philips состоит в том, что в режиме Philips данные выдаются с задержкой на один такт относительно импульса начала выбора канала. Если установить бит PNOS в 1, выдача сигналов выбора канала и данных будет происходить относительно среза синхросигнала. Пример приведен на рисунке 104.

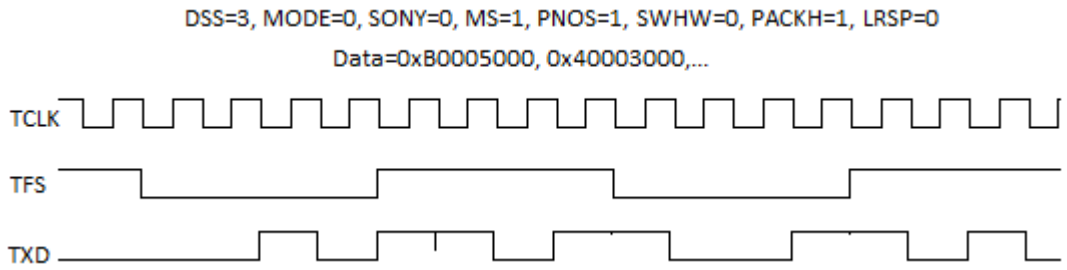


Рисунок 104 – Выдача информации по срезу TCLK

Это позволяет более гибко адаптироваться при подключении к приемнику информации и обеспечить надежное время предустановки и удержания данных.

Установка бита SWHW в «1» вызовет перестановку полуслов в слове данных. Пример приведен на рисунке 105. Данный прием позволяет поменять местами левый и правый каналы.

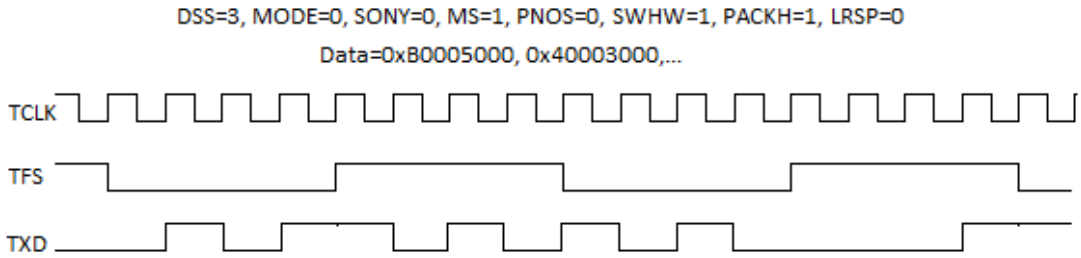


Рисунок 105 – Перестановка полуслов в слове

Интерфейс может работать в модифицированном режиме или режиме DSP. Основное отличие данного режима состоит в формировании импульса TFS. В режиме DSP он не переключается между правым и левым каналом, а импульсом информирует о начале нового фрейма. Пример приведен на рисунке 106.

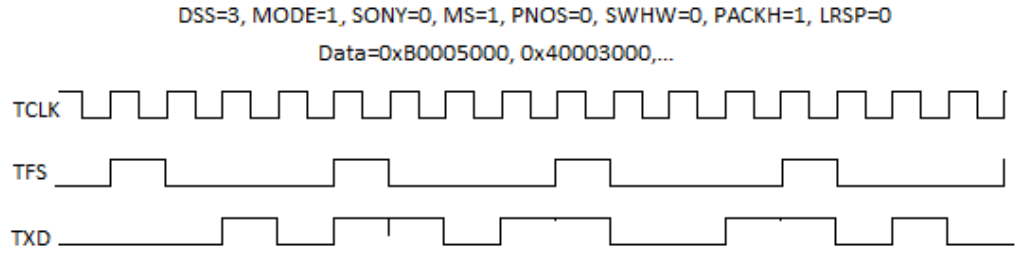


Рисунок 106 – Режим передачи DSP

Импульс на выходе TFS указывает, что в следующем такте начинается выдачи информации нового фрейма. Каждые четыре бита данных сопровождаются импульсом начала фрейма (рисунок 106). В данном случае уже нет деления на левый и правый каналы. Однако если необходимо передать информацию, состоящую из двух частей, то возможны два способа. Первый состоит в упаковке двух половин информации в одно 32-разрядное слово и передаче как единого 32-разрядного фрейма. Второй способ может использоваться, если две порции информации находятся в последовательных словах, следующих друг за другом. В этом случае необходимо установить бит LRSP в «1». Для нашего примера передачи данных временная диаграмма представлена на рисунке 107.

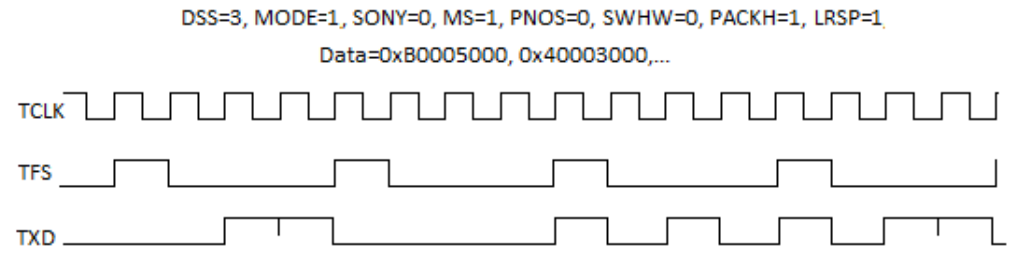


Рисунок 107 – Режим передачи смежными словами

В случае, когда информация находится в смежных словах (в нашем случае в смежных полусловах, т.к. длина данных меньше 16 бит и установлен признак PACK) количество передаваемых бит делится пополам и сначала передаются биты одной половины, а затем второй. В нашем случае передается четыре бита, поэтому сначала будут переданы два старшие бита младшего полуслова (биты 01b от значения 5), а затем два старшие бита старшего полуслова (биты 10b от значения B). Данный режим обычно используется, если длина объединенной посылки превышает 32 бита и в одном слове невозможно упаковать две передаваемые части.

Бит MS в регистре управления определяет источник формирования сигнала синхронизации и сигнала выбора канала (строб начала фрейма). Если бит установлен в «1», передатчик формирует данные сигналы. В противном случае внешнее устройство подает сигналы синхронизации и выбора канала. Последний случай обычно используется, если внешнее устройство требует формирования заданной частоты, которая не может быть получена в процессоре путем деления частоты SOC-шины на константу. Отметим, что в случае работы в режиме подчиненного (MS=0) интерфейс принимает сигнал выбора канала. В этом случае имеет значение бит PNIS, т.к. он задает момент приема достоверного значения сигнала выбора канала.

21.2.2.2 *Регистр управления приемником I2S_R_CR*

Таблица 200 – Биты регистра управления приемником I2S_R_CR

Бит	Имя	Функция
0	REN	Разрешение работы (1)
1	MODE	0 – I2S режим; 1 – DSP режим
2	SONY	Выбор стандарта SONY (1) или Philips (0) Имеет смысл только в режиме I2S
3	MS	Master (1) or slave(0). Бит определяет того кто генерирует синхросигнал и строб начала фрейма (выбор канала левый-правый)
9:4	DSS	Длина передаваемых данных – от 1 до 64 бит
10	PNIS	Определяет момент приема данных и выбора канала: 0 – выдача по фронту (из «0» в «1»); 1 – выдача по срезу (из «1» в «0»)
11	PNOS	Определяет момент выдачи выбора канала: 0 – выдача по фронту (из «0» в «1»); 1 – выдача по срезу (из «1» в «0»)
12	SWHW	1 – указывает на необходимость перестановки 16-разрядных полуслов в одном слове
13	R_INIT	В случае приема сообщения длина которого меньше 16 бит, либо больше 16 бит, но меньше 32, возможна предварительная инициализация сдвигового регистра. Регистр инициализируется значением, которое зависит от бита SEXT. Функция инициализации работает если бит установлен
14	LRSP	1 и установлен DSP режим – общее принимаемое слово состоит из двух симметричных частей (левой и правой), которые необходимо принять раздельно в разные 32-разрядные слова. 0 – принимаемая последовательность не делится на части и представляет собой непрерывный поток бит
15	SEXT	Если бит R_INIT равен единице, бит SEXT определяет значение, которым инициализируется сдвиговый регистр приемника: 0 – регистр инициализируется нулями; 1 – биты регистра инициализируются значением первого принимаемого бита (расширение знака)

В режиме I2S используются FIFO передатчика и приемника аудио данных. Поддержка модемного кодека невозможна.

В случае приема данных в режиме I2S, если длина данных меньше или равна 16 бит, данные будут упаковываться в 32-разрядные слова. При этом возможна перестановка полуслов в слове. В случае длины принимаемых данных больше 16 бит, данные помещаются в 32-разрядные слова. При этом достоверны только DSS+1 младших бит 32 разрядного слова. В случае приема данных, длина которых больше 32 бит, возможны два варианта приема:

1 если LRSP == 0, первые 32 бита будут размещены в первом слове, а оставшиеся биты в следующем слове;

2 если LRSP == 1, принимаемые данные делятся поровну на левую и правую половины, каждая из которых размещается в индивидуальном слове.

Прием данных и выбора слова может происходить как по срезу синхросигнала, так и по фронту. Выбор фронта/среза программируется.

Как приемник, так и передатчик в режиме мастер могут выдавать наружу сигнал синхронизации обмена и строб начала обмена. Для задания частоты обмена используется внутренний делитель. Базовой частотой делителя является клок периферийной шины. Значение коэффициента деления находится в регистре SICR2[22:7]. Значение коэффициента деления на единицу больше записываемого значения.

Биты разрешения работы только включают передатчик или приемник. Для того чтобы полностью включить прием или/и передачу необходимо установить соответствующие биты в регистре SICR2[1:0]. Если приемник или передатчик выключен, то все его внутренние регистры сбрасываются в ноль. Очень важным для корректного начала работы может быть необходимость установки FIFO передатчика и приемника в начальное состояние. Это можно сделать битом SICR0[4], который производит сброс всех FIFO интерфейса (указателей и флагов) в исходное состояние. Для сброса бит нужно установить, а затем программно очистить. После этого FIFO передатчика можно заполнить значением 0, если передаваемые данные еще не готовы.

22 Контроллер USB канального уровня (для микросхем с ревизии 3)

Ядро контроллера USB канального уровня соответствует стандарту USB 2.0 для high-/full-speed функций. Устройства, которые являются исключительно периферийными, могут инициировать USB-трафик через протокол запроса сеанса (SRP), в то время как устройства, играющие двойную роль, поддерживают протокол SRP и Host Negotiation Protocol (HNP) и могут выполнять роль хоста или периферийного устройства по мере необходимости. Контроллер также поддерживает разделенные передачи данных, что позволяет поддерживать использование high-/full-speed устройств с хабом USB 2.0.

В контроллере USB можно настроить до четырех конечных точек передачи и/или до четырех конечных точек приема в дополнение к точке 0. (Использование этих конечных точек для передачи данных зависит от того, используется ли контроллер как периферийное устройство или как хост).

С каждой конечной точкой требуется связать FIFO. Контроллер имеет интерфейс RAM для подключения к блоку синхронного однопортового ОЗУ, который используется для всех FIFO конечных точек.

Для конечной точки 0 требуется FIFO с размером 64 байт для буферизации одного пакета. Интерфейс ОЗУ настраивается с учетом других FIFO точек, размер которых может составлять от 8 до 8192 байт и может буферизировать 1 или 2 пакета. Некоторые FIFO могут быть связаны со всеми конечными точками: в качестве альтернативы точка TX и точка RX с одним и тем же номером могут быть сконфигурированы для использования одного и того же FIFO, например, для уменьшения необходимого размера блока RAM, если они никогда не могут быть активны одновременно.

Назначение выводов интерфейса приведено в таблице 201

Таблица 201 – Назначение внешних выводов интерфейса USB

Обозначение вывода	Назначение вывода контроллера	Тип	Функциональное назначение
PB[10]	USB_CLK	I	Синхросигнал
PB[11]	USB_DIR	O	Направление передачи
PB[12]	USB_NXT	O	Готовность приема
PB[13]	USB_STP	I	Остановка передачи
PB[14]	USB_D[0]	I/O	Выход данных, бит 0
PB[15]	USB_D[1]	I/O	Выход данных, бит 1
PB[16]	USB_D[2]	I/O	Выход данных, бит 2
PB[17]	USB_D[3]	I/O	Выход данных, бит 3
PB[18]	USB_D[4]	I/O	Выход данных, бит 4
PB[19]	USB_D[5]	I/O	Выход данных, бит 5
PB[20]	USB_D[6]	I/O	Выход данных, бит 6
PB[21]	USB_D[7]	I/O	Выход данных, бит 7

22.1 Поддержка нескольких устройств

22.1.1 Распределение устройств между конечными точками

Отдельные функции подключенных устройств назначаются конечным точкам в ядре контроллера USB через группу из трех регистров, которые связаны с каждой конечной точкой Rx или Tx, реализованной в ядре (включая конечную точку 0).

Это регистры Tx/RxFuncAddr, Tx/RxHubAddr и Tx/RxHubPort. (Расположение этих регистров зависит от того, какая из конечных точек контроллера настраивается).

Информацией, которая должна быть записана в регистр Tx/RxFuncAddr, является адрес целевой функции, к которой необходимо получить доступ через выбранную конечную точку.

Регистры Tx/RxHubAddr и Tx/RxHubPort предусмотрены для случая, когда full-speed или low-speed устройство подключено к контроллеру через high-speed USB 2.0 хаб, который выполняет требуемую трансляцию транзакций между high-speed передачей и low-/full-speed передачей. В этом случае регистры Tx/RxHubAddr и Tx/RxHubPort должны хранить адрес хаба, который выполняет трансляцию транзакции, и порт хаба, через который конечная точка должна получить доступ к устройству.

Регистр Tx/RxHubAddr также используется для записи того, предлагает ли хаб несколько транзакционных трансляторов или только один. Это существенно влияет на общую достижимую пропускную способность.

В дополнение к записи адреса целевой функции через эти три регистра, необходимо записать номер конечной точки, скорость работы целевого устройства и тип трансляции, которая будет выполнена.

Для конечной точки Tx эту информацию необходимо установить в регистре TxType, когда регистр Index установлен на нужную конечную точку. Для конечной точки Rx эту информацию необходимо установить в регистре RxType, когда регистр Index установлен на нужную конечную точку. В обоих случаях номер конечной точки записывается в битах D3-D0, тип транзакции выбирается через биты D5-D4, а рабочая скорость выбирается через биты D7-D6.

В случае конечной точки 0 необходимо установить только скорость (эта конечная точка имеет средства для обработки только транзакций управления и, следовательно, всегда связана с конечной точкой 0 этого устройства). Эта установка скорости выполняется через биты D7-D6 регистра Type 0, когда в регистре Index установлен 0.

22.1.2 Описание работы

После того как распределение функций по конечным точкам было сделано и скорость работы целевого устройства установлена, большинство операций в многоточечной настройке ничем не отличаются от операций для эквивалентных действий, в которых ядро подключено только к одному устройство.

Однако необходимы дополнительные действия в следующих ситуациях:

– используется опция динамического распределения функций по конечным точкам (например, чтобы поддерживать более широкий диапазон устройств)

– управляющие пакеты, обычно ассоциируемые с конечной точкой 0, обрабатываются через другую конечную точку.

Если используется динамическое распределение, пользователю необходимо отслеживать текущее состояние Data Toggle конечной точкой и каждого из устройств, связанного с этой конечной точкой. Это необходимо, чтобы пользователь смог выбрать правильное состояние Data Toggle при переключении между устройствами.

Состояние Data Toggle может быть изменено при помощи изменения соответствующего регистра TxCSR/RxCSR и установкой туда битов Data Toggle Write Enable и Data Toggle, которые доступны, пока ядро находится в режиме хоста.

Если управляющие пакеты обрабатываются через конечную точку, отличную от конечной точки 0, пользователь должен позаботиться о том, чтобы каждый Setup токен был отправлен. Это включает в себя настройку бита SetupPkt из регистра TxCSR и бита TkPktRdy, когда ядро работает в режиме хоста. Если бит SetupPkt не установлен, будет отправлен токен OUT.

Использование другой конечной точки для этой функции возможно, как описано выше, но с учетом замечаний:

1 Управляющая функция должна быть назначена только на пару конечных точек Rx/Tx (то есть с тем же номером конечной точки).

2 Выбранные конечные точки должны быть связаны с FIFO, который сможет вместить размер пакета транзакции EP0, на выбранной рабочей скорости (т. е. минимум 8 байт для транзакций low-/full-speed, или 64 байта для транзакций high-speed).

22.2 Схема программирования

В этом и следующих разделах рассмотрены действия, которые необходимо выполнить устройству, управляющему ядром контроллера USB, и некоторые аспекты работы ядра, которые влияют на это.

22.2.1 Обработка USB прерываний

Когда USB прерывает ЦП, ему необходимо прочитать регистр статуса прерывания, чтобы определить, какие конечные точки вызвали прерывание и перейти к соответствующей процедуре обработки. Если несколько конечных точек вызвали прерывание, сначала необходимо обработать конечную точку 0, а затем остальные конечные точки. Прерывание приостановки должно обрабатываться последним.

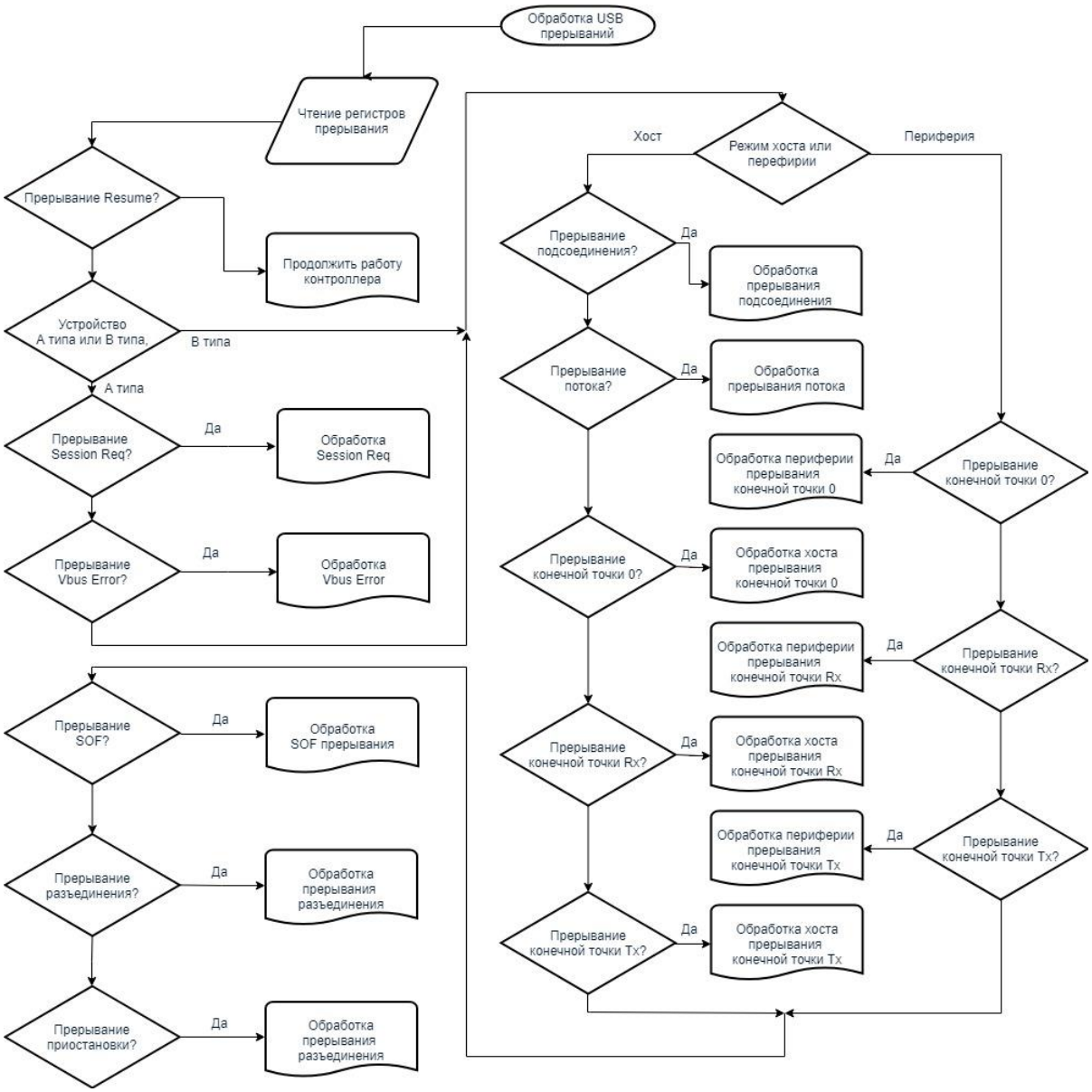


Рисунок 108 – Блок-схема процедуры обслуживания прерываний USB

22.2.2 Приостановка/возобновление

Как контроллер USB входит и выходит из режима Suspend (приостановки) зависит от того, работает ли он в настоящее время как хост или как периферия.

Когда контроллер работает как хост

1 Вход в режим Suspend. При работе в качестве хоста контроллер может перейти в режим Suspend, установив бит SuspendMode в регистр Power. После установки этого бита, контроллер завершит текущую транзакцию, а затем остановит планировщик транзакций и счетчик кадров. Дальнейшие транзакции не будут происходить, и SOF-пакеты не будут создаваться.

Если бит Enable SuspendM (Power.D0) установлен, UTMI + PHY перейдет в режим с низким энергопотреблением, если контроллер перейдет в режим Suspend и остановит XCLK.

2 *Передача сигналов Resume.* Когда программа требует, чтобы контроллер покинул режим Suspend, ему необходимо очистить бит Suspend в регистре Power,

установить бит возобновления и оставить его на 20 мс. Пока этот бит установлен, контроллер будет генерировать сигналы Resume на шине. Через 20 мс ЦП должен сбросить бит Resume, после чего будет запущен счетчик кадров и планировщик транзакций.

3 *Ответ на удаленную активацию.* Если контроллер в режиме Suspend получил сигнал Resume, UTMI + PHY будет выведен из режима низкого энергопотребления и перезапустит XCLK. Контроллер выйдет из режима Suspend и автоматически установит бит Resume в регистре питания (D2), чтобы взять на себя генерацию сигнала Resume. Если прерывание Resume разрешено, то оно будет сгенерировано.

Когда контроллер работает как периферийное устройство

1 Вход в режим Suspend. При работе в качестве периферийного устройства контроллер анализирует активность на устройстве USB и, если в течение 3 мс не было активности, переходит в режим Suspend. Если прерывание Suspend разрешено, то в этот момент оно будет отправлено. Уровень выхода SUSPENDM также будет низким (если включен).

На этом этапе контроллер остается активным (и, следовательно, способен обнаружить, когда на накопителе USB появляется сигнал Resume), или программа может отключить контроллер, остановив тактировщик. Однако во втором случае контроллер не сможет обнаружить сигнал Resume на USB, а значит и перезапустить тактировщик.

2 *Когда на шине возникает сигнал Resume,* сначала необходимо перезапустить тактировщик. Затем контроллер автоматически выйдет из режима Suspend. Если прерывание Resume доступно, то оно будет создано.

3 *Инициирование удаленной активации.* Если программа хочет инициировать удаленную активацию, в то время как контроллер находится в режиме Suspend, он должен в регистре Power установить бит возобновления (D2).

Примечание – Если источник тактирования контроллера был остановлен, его необходимо перезапустить до того, как запись бита произойдет.

Программа должна оставить этот бит установленным в течение приблизительно 10 мс (минимум 2 мс, максимум 15 мс) перед тем, как сбросить. К этому времени хаб должен будет получить сигнал Resume на USB.

Примечание – когда программа инициирует удаленную активацию, прерывание Resume не генерируется.

22.2.3 Перезагрузка USB

Когда контроллер USB работает как периферийное устройство, и на устройстве USB обнаружено состояние сброса, устройство выполнит следующие действия:

- сбросит FAddr;
- сбросит Index;
- очистит все FIFO конечных точек;
- сбросит все регистры Control/Status;
- включит все прерывания конечных точек;

– сгенерирует прерывание Reset.

Если бит HS Enab в регистре Power (D5) был установлен, контроллер также попытается согласоваться для работы в режиме high-speed. Получится ли согласоваться для работы в high-speed режиме отобразится в бите HS Mode (Power.D4).

Когда ПО ***в режиме периферийного устройства,*** ПО, управляющее контроллером, при получении прерывание Reset должно закрыть все открытые каналы и ждать начала перечисления шины.

В хост режиме если бит Reset установлен в регистре Power, когда контроллер находится в режиме хоста, контроллер сгенерирует сигнал Reset на шине. Если был установлен бит HS Enab в регистре питания (D5), он также попытается согласоваться для работы в high-speed режиме.

ЦП должен поддерживать бит сброса установленным не менее 20 мс, чтобы обеспечить правильный сброс целевого устройства.

После того как ЦП сбросит бит, контроллер запустит счетчик кадров и планировщик транзакций. Будет ли выбран high-speed режим, будет указано в бите HS Mode (Power.D4).

22.3 Контроль транзакций (при помощи конечной токи 0)

22.3.1 Контроль транзакций в качестве периферийного устройства

22.3.1.1 *Запрос нулевых данных*

Запросы нулевых данных содержат всю свою информацию в восьмибайтной команде и не требуют передачи дополнительных данных. Примеры стандартных запросов «Нулевые данные»: SET_FEATURE, CLEAR_FEATURE, SET_ADDRESS, SET_CONFIGURATION, SET_INTERFACE.

Последовательность событий начнется, как и при всех запросах, когда программное обеспечение получит прерывание конечной точки 0. Также будет установлен бит RxPktRdy (CSR0.D0). Затем команда из восьми байт должна прочитаться из FIFO конечной точки 0, декодироваться и далее, соответствующие ей действия должны быть приняты.

Затем биты ServicedRxPktRdy (D6) (указывающий, что команда была прочитана из FIFO) и DataEnd (D3) (указывающий, что для этого запроса не ожидается никаких дополнительных данных) установятся в регистр CSR0.

Когда хост переходит на этап запроса, будет отправлено второе прерывание конечной точки 0, указывающее, что запрос завершился. Никаких дополнительных действий от программного обеспечения не требуется: второе прерывание – это просто подтверждение успешного завершения запроса.

Если команда не опознана или по какой-либо другой причине не может быть выполнена, тогда, после декодировки, установятся биты ServicedRxPktRdy (D6) и SendStall (D5) в регистр CSR0. Когда хост переходит на этап запроса, контроллер отправит STALL, чтобы сообщить хосту, что запрос не был выполнен. Будет создано второе прерывание конечной точки 0, и установится бит SentStall (CSR0.D2).

Если хост продолжает отправляет данные после того, как бит DataEnd был установлен, контроллер отправит STALL. Прерывание конечной точки 0 будет сгенерировано и бит SentStall (CSR0.D2) будет установлен.

22.3.1.2 *Запрос на запись*

Запросы на запись включают дополнительный пакет (или пакеты) данных, отправляемых с хоста после восьмибайтной команды. Примером стандартного запроса на запись является: SET_DESCRIPTOR.

Последовательность событий начнется, как и при всех запросах, когда программное обеспечение получит прерывание конечной точки 0. Также будет установлен бит RxPktRdy (CSR0.D0). Затем восьмибайтная команда должна прочитаться из FIFO конечной точки 0 и декодироваться.

Как и при нулевом запросе данных, бит ServicedRxPktRdy (D6) (указывающий, что команда была прочитана из FIFO) должен быть установлен в регистре CSR0, но бит DataEnd (D3) не должен быть установлен (указывая, что ожидается больше данных).

Когда получено второе прерывание конечной точки 0, регистр CSR0 должен быть прочитан для проверки состояния конечной точки. Бит RxPktRdy (CSR0.D0) должен быть установлен, чтобы указать, что пакет данных получен. Затем регистр COUNT0 должен быть прочитан для определения размера этого пакета данных. Затем пакет данных можно читать из FIFO конечной точки 0.

Если длина данных, связанных с запросом (определяемая полем wLength в команде), больше, чем максимальный размер пакета для конечной точки 0, будут отправлены еще пакеты данных. В этом случае бит ServicedRxPktRdy должен быть установлен в регистр CSR0, но бит DataEnd должен быть сброшен.

Когда все ожидаемые пакеты данных получены, биты ServicedRxPktRdy и DataEnd (указывающий, что больше данных не ожидается) должны быть установлены в регистре CSR0.

Когда хост переходит на этап запроса, будет создано другое прерывание конечной точки 0, указывающее, что запрос завершен. Никаких дальнейших действий не требуется от программного обеспечения, прерывание является просто подтверждением успешного завершения запроса.

Если команда не распознана или по какой-либо другой причине не может быть выполнена, то после декодирования биты ServicedRxPktRdy (D6) и SendStall (D5) должны будут установиться в регистр CSR0. Если хост продолжит отправлять данные, то контроллер отправит STALL, чтобы сообщить хосту, что запрос не был выполнен. Будет отправлено прерывание конечной точки 0, и установится бит SentStall (CSR0.D2).

22.3.1.3 *Запрос на чтение*

Запросы на чтение имеют пакет (или пакеты) данных, отправленный из функции на хост после восьмибайтной команды. Примеры стандартных запросов на чтение: GET_CONFIGURATION, GET_INTERFACE, GET_DESCRIPTOR, GET_STATUS, SYNCH_FRAME.

Последовательность событий начнется, как и во всех запросах, когда программное обеспечение получит прерывание конечной точки 0, а также будет

установлен бит RxPktRdy (CSR0.D0). Затем восьмибайтная команда должна прочитаться из FIFO конечной точки 0 и декодироваться. Затем бит ServicedRxPktRdy (указывающий, что команда прочитала FIFO) должен быть установлен в регистре CSR0.

Затем данные, которые должны быть отправлены хосту, должны быть записаны в FIFO конечной точки 0. Если отправляемые данные больше максимально допустимого размера пакета для конечной точки 0, в FIFO должна быть записана только часть пакета максимально допустимого размера. Затем бит TxPktRdy (D1) (указывающий, что в FIFO будет отправлен пакет) должен быть установлен в регистр CSR0. Когда пакет будет отправлен на хост, сгенерируется еще одно прерывание конечной точки 0, и следующий пакет данных может быть загружен в FIFO.

Когда последний пакет данных будет записан в FIFO, биты TxPktRdy и DataEnd (D3) (указывающий, что после этого пакета больше нет данных) должны быть установлены в регистре CSR0.

Когда хост переходит на этап запроса, будет создано еще одно прерывание конечной точки 0, чтобы указать, что запрос завершился. Никаких дальнейших действий от программного обеспечения не требуется: прерывание – это просто подтверждение того, что запрос успешно завершен.

Если команда не распознана или по какой-либо другой причине не может быть выполнена, после декодировки должны быть установлены биты ServicedRxPktRdy (D6) и SendStall (D5) в регистре CSR0. Если хост запросит данные, контроллер отправит STALL, чтобы сообщить хосту, что запрос не был обработан. Будет сгенерировано прерывание конечной точки 0, и будет установлен бит SentStall (CSR0.D2).

Если хост запрашивает данные после установки DataEnd (D3), контроллер отправит STALL. Будет сгенерировано прерывание конечной точки 0, и будет установлен бит SentStall (CSR0.D2).

22.3.1.4 *Состояния конечной точки 0*

Когда контроллер USB работает как периферийное устройство, контроль конечной точки 0 может происходить в трех состояниях: IDLE, TX и RX.

Режим IDLE устанавливается по умолчанию при включении или сбросе.

RxPktRdy (CSR0.D0) устанавливается, когда конечная точка 0 находится в состоянии IDLE, указывая на новый запрос устройства. После того, как запрос устройства выгружается из FIFO, контроллер декодирует дескриптор, чтобы проверить наступление фазы передачи данных и, если она наступила, направление передачи данных передачи (чтобы установить направление FIFO).

В зависимости от направления передачи данных конечная точка 0 переходит в состояние TX или RX. Если фаза данных отсутствует, конечная точка 0 остается в состоянии IDLE, чтобы принять следующий запрос устройства.

Действия, которые процессор должен выполнять на разных этапах возможных передач (например, загрузка FIFO, настройка TxPktRdy), указаны на диаграмме.

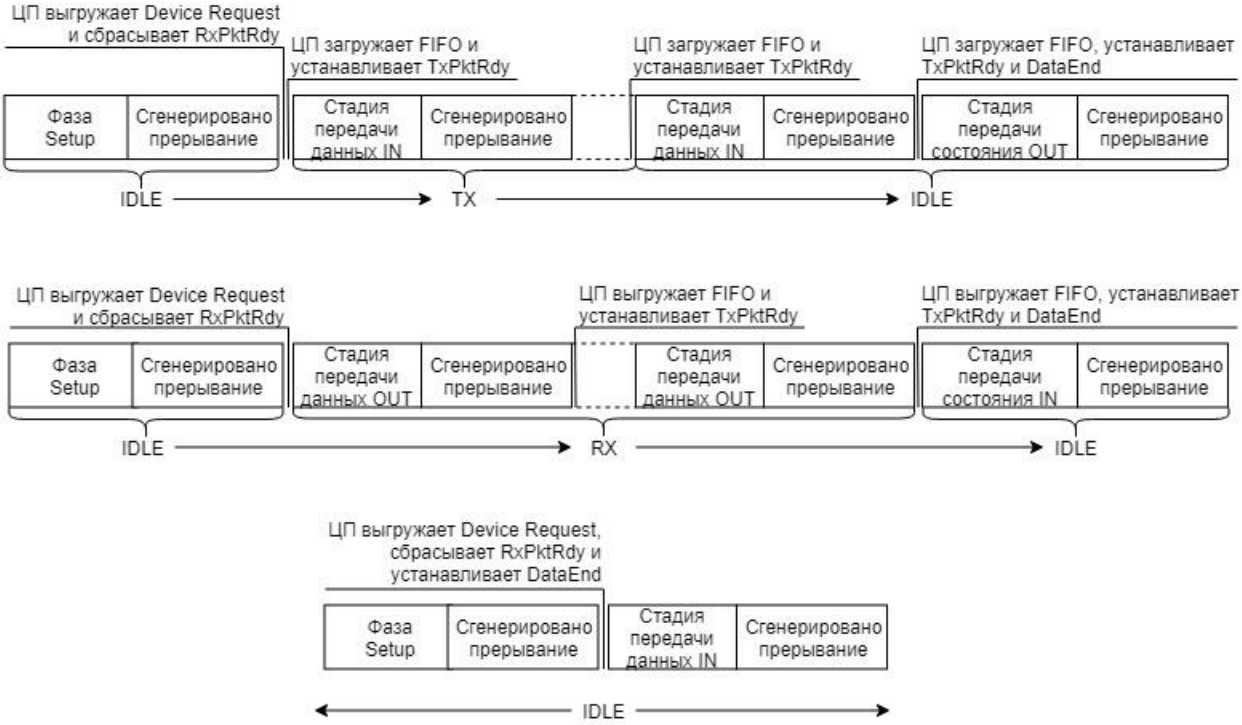


Рисунок 109 – Диаграмма передачи данных

22.3.1.5 *Процедура обслуживания конечной точки 0 как периферийного устройства*

Прерывание конечной точки 0 генерируется в следующих случаях:

– ядро устанавливает бит RxPktRdy (CSR0.D0) после получения действительного токена и записи данных в FIFO;

– ядро очищает бит TxPktRdy (CSR0.D1) после того, как пакет данных в FIFO был успешно передан хосту;

– ядро устанавливает бит SentStall (CSR0.D2) после завершения транзакции управления из-за нарушения протокола;

– ядро устанавливает бит SetupEnd (CSR0.D4), потому что передача управления завершилась до того, как установлен DataEnd (CSR0.D3).

Каждый раз, когда инициализируется сервис контрольной точки 0, прошивка должна проверить, была ли текущая передача управления завершена из-за состояния STALL или из-за преждевременного завершения передачи управления. Если передача управления заканчивается из-за условия STALL, бит SentStall должен быть установлен. Если передача управления заканчивается из-за преждевременного окончания передачи управления, бит SetupEnd должен быть установлен. В любом случае прошивка должна прервать обработку текущей передачи управления и установить состояние IDLE.

После того, как прошивка определила, что прерывание не было создано неразрешенным состоянием шины, следующее действие будет зависеть от состояния конечной точки.

Если конечная точка 0 находится в состоянии «IDLE», единственной допустимой причиной, из-за которой может быть отправлено прерывание, является результат получения данных ядра с шины USB. Сервис должен проверить это, проверив

бит RxPktRdy (CSR0.D0). Если бит установлен, то ядро получило пакет данных SETUP. Он должен быть выгружен из FIFO и декодирован для определения дальнейшего действия. В зависимости от этого действия, конечная точка 0 примет одно из трех следующих состояний:

– если команда представляет собой однопакетную транзакцию (SET_ADDRESS, SET_INTERFACE и т.д.) без фазы данных, конечная точка останется в состоянии IDLE;

– если команда имеет фазу передачи данных OUT (SET_DESCRIPTOR и т.д.), конечная точка перейдет в состояние Rx;

– если команда имеет фазу передачи данных IN (GET_DESCRIPTOR и т.д.), конечная точка перейдет в состояние TX.

Если конечная точка находится в состоянии Tx, прерывание укажет на то, что ядро получило токен IN, и данные из FIFO были отправлены. Прошивка должна ответить на это либо путем размещения большего количества данных в FIFO, если хост все еще ожидает данные, либо путем установки бита DataEnd, чтобы указать, что фаза передачи данных завершена. Как только фаза передачи данных транзакции будет завершена, конечная точка 0 будет возвращена в состояние IDLE для ожидания следующей транзакции управления.

Если конечная точка находится в состоянии Rx, прерывание укажет на то, что пакет данных получен. Прошивка должна ответить выгрузкой полученных данные из FIFO. Затем прошивка должна будет определить, получил ли он все ожидаемые данные. Если это так, прошивка должна установить бит DataEnd и вернуть конечную точку 0 в состояние IDLE. Если ожидается получение большего количества данных, прошивка должна установить бит ServicedRxPktRdy (CSR0.D6), чтобы указать, что она прочитал данные из FIFO и оставить конечную точку в состоянии RX.

Режим IDLE

Режим IDLE – это режим, который должен выбрать контроллер конечной точкой 0 при включении или перезагрузке и является режимом, в который контроллер конечной точки 0 должен возвращаться при завершении режимов RX и TX.

Это также режим, в котором обрабатывается фаза передачи SETUP (как показано на рисунке 110.

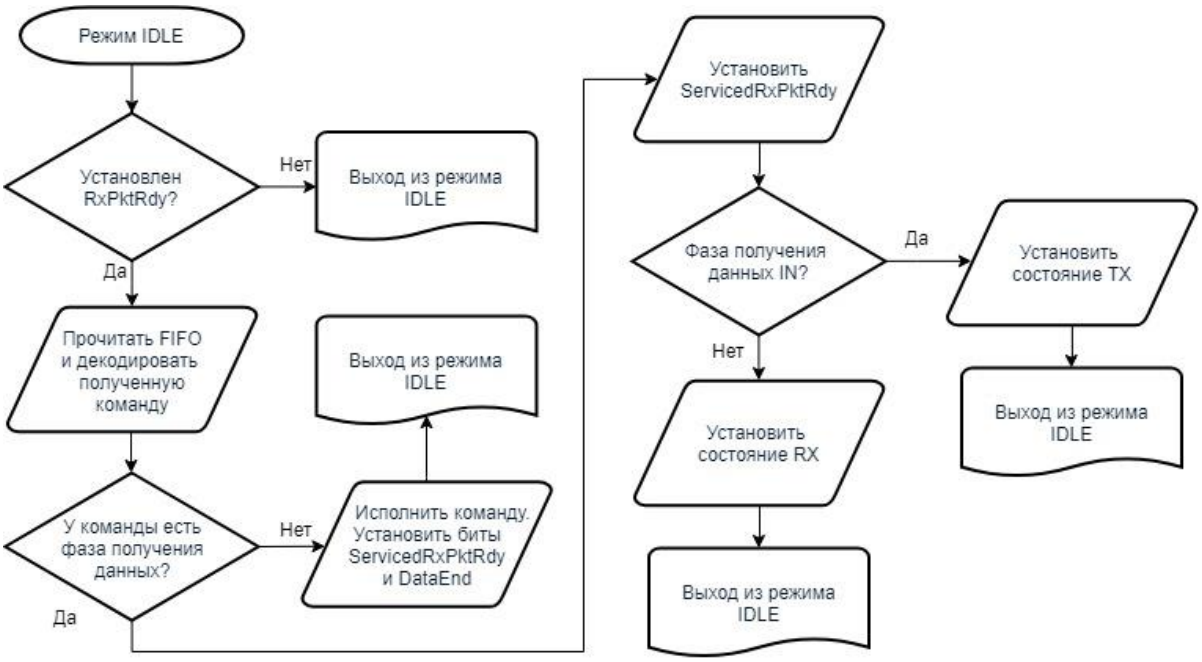


Рисунок 110 – Алгоритм режима IDLE

Режим TX

Когда конечная точка находится в состоянии TX, все входящие токены должны обрабатываются как часть фазы передачи данных до тех пор, пока требуемый объем данных не будет отправлен на хост. Если принимается либо токен SETUP, либо токен OUT, и конечная точка находится в состоянии TX, то это вызовет условие SetupEnd, поскольку ядро ожидает только токены IN.

Три события могут привести к переключению режима TX до того, как ожидаемый объем данных будет отправлен:

– хост отправляет недействительный токен, вызывающий условие SetupEnd (установит CSR0.D4);

– прошивка отправляет пакет, содержащий данных меньше максимального размера (MaxP);

– прошивка отправляет пустой пакет данных.

Пока транзакция не будет завершена, прошивка должна загружать FIFO при получении прерывания, которое указывает на то, что пакет был отправлен из FIFO (Прерывание генерируется, когда TxPktRdy очищается).

Когда прошивка принудительно завершает передачу (путем отправки короткого или пустого пакета данных), должен быть установлен бит DataEnd (CSR0.D3), чтобы указать ядру, что фаза передачи данных завершена и, что затем ядро должно получить подтверждающий пакет.

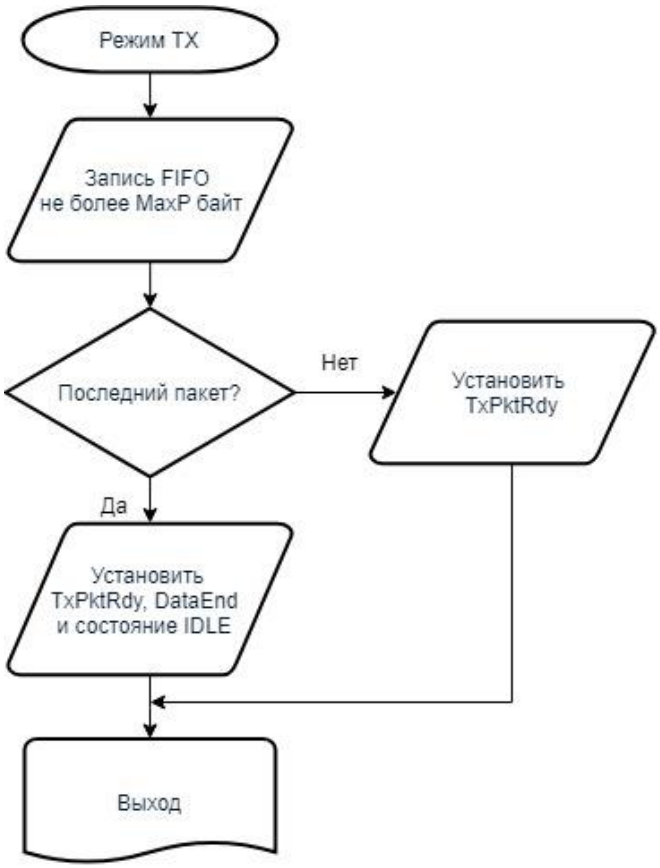


Рисунок 111 – Алгоритм режима TX

Режим RX

В режиме RX все поступающие данные должны рассматриваться как часть фазы передачи данных до тех пор, пока ожидаемый объем данных не будет получен. Если принимается либо SETUP, либо IN токен, и конечная точка находится в состоянии RX, то это приводит к возникновению условия SetupEnd, поскольку ядро ожидает только токены OUT.

Три события могут привести к прекращению режима RX до того, как ожидаемый объем данных будет получен:

– хост отправляет недействительный токен, вызывающий условие SetupEnd (установит CSR0.D4);

– хост отправляет пакет, содержащий данных меньше максимального размера пакета для конечной точки 0;

– хост отправляет пустой пакет данных.

Пока транзакция не будет завершена, прошивка должна выгружать FIFO при получении прерывания, которое указывает на то, что пришли новые данные (установлен RxPktRdy (CSR0.D0)) и сбрасывать RxPktRdy, устанавливая бит ServicedRxPktRdy (CSR0.D6).

Когда прошивка обнаруживает завершение передачи (путем получения либо ожидаемого количества данных, либо пустого пакета данных), он должен установить бит DataEnd (CSR0.D3), чтобы указать ядру, что фаза передачи данных завершена, и что далее ядро должно получить пакет подтверждения.

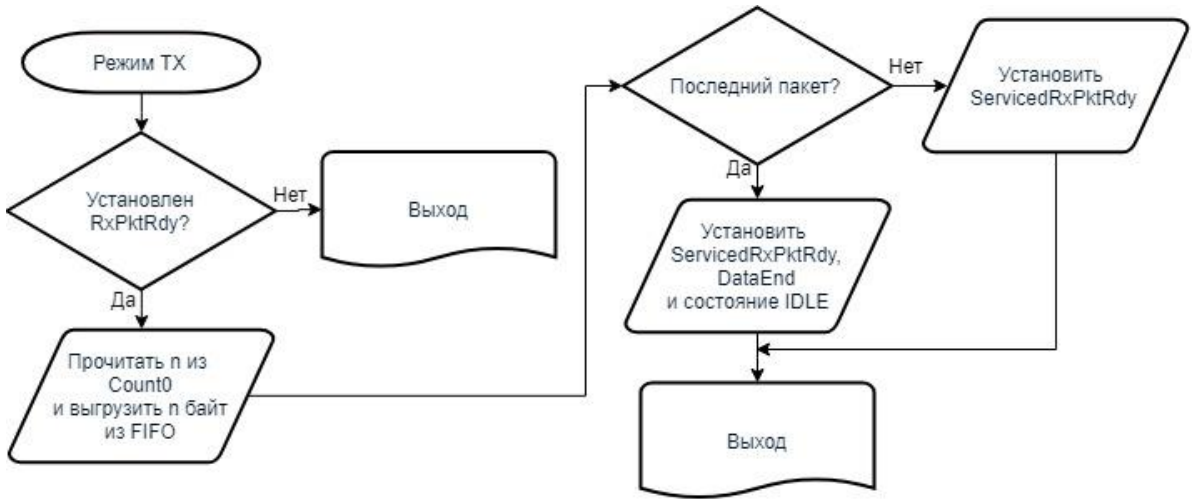


Рисунок 112 – Алгоритм режима RX

22.3.1.6 *Работа с ошибками как периферийного устройства*

Передача управления может быть прервана из-за ошибки протокола на USB, если хост преждевременно прекращает передачу, или если программное обеспечение функций контроллера хочет прервать передачу (например, потому что оно не может обработать команду).

Контроллер USB автоматически обнаружит ошибки протокола и отправит пакет STALL на хост при следующих условиях:

– хост отправляет больше данных во время фазы передачи данных OUT запроса на запись, чем указано в команде. Это условие обнаруживается, если хост отправляет токен OUT после того, как бит DataEnd (CSR0.D3) был установлен;

– хост запрашивает больше данных во время фазы передачи данных IN запроса на чтение, чем указано в команде. Это условие обнаруживается, когда хост отправляет токен IN после того, как бит DataEnd (CSR0) был установлен;

– хост отправляет больше, чем MaxP байт данных в пакете данных OUT;

– хост отправляет пакет DATA1 с ненулевой длиной во время фазы STATUS запроса на чтение.

Когда контроллер USB отправит пакет STALL, он установит бит SentStall (CSR0.D2) и сгенерирует прерывание. Когда программное обеспечение получит прерывание конечной точки 0 с установленным битом SentStall, оно должно прервать текущую передачу, сбросить бит SentStall и вернуться в состояние IDLE.

Если хост преждевременно завершит передачу, введя фазу STATUS до того, как все данные запроса будут переданы, или отправив новый пакет SETUP до завершения текущей передачи, то установится бит SetupEnd (CSR0.D4) и сгенерируется прерывание конечной точки 0. Когда программное обеспечение получит прерывание конечной точки 0 с установленным битом SetupEnd, оно должно прервать текущую передачу, установить бит ServicedSetupEnd (CSR0.D7) и вернуться в состояние IDLE. Если бит RxPktRdy (CSR0.D0) установлен, это указывает на то, что хост отправил еще один пакет SETUP, и программное обеспечение должно обработать эту команду.

Если программное обеспечение хочет прервать текущую передачу, поскольку оно не может обработать команду или имеет некоторую другую внутреннюю ошибку, то оно

должно установить бит SendStall (CSR0.D5). Затем контроллер USB отправит пакет STALL на хост, установит бит SentStall (CSR0.D2) и сгенерирует прерывание конечной точки.

22.3.1.7 *Дополнительные действия*

При работе в качестве периферии ядро контроллера USB автоматически реагирует на определенные условия на шине USB или действиях хоста. Подробности приведены ниже:

STALL на управляющих передачах

Ядро USB контроллера автоматически выдает STALL на управляющие передачи при следующих условиях:

– хост отправляет больше данных во время фазы передачи данных OUT управляющей передачи, чем указано в запросе устройства во время фазы SETUP. Это условие обнаруживается контроллером, когда хост отправляет токен OUT (вместо токена IN) после того, как ЦП выгрузил последний пакет OUT и установил DataEnd;

– хост запрашивает больше данных во время фазы передачи данных IN управляющей передачи, чем указано в запросе устройства во время фазы SETUP. Это условие обнаруживается контроллером, когда хост отправляет токен IN (вместо токена OUT) после того, как ЦП сбросил TxPktRdy и установил DataEnd в ответ на ACK, выданный хостом тому пакету, который должен был быть последним;

– хост отправляет больше, чем MaxP данных с токеном данных OUT;

– хост отправляет неверный идентификатор PID для фазы состояния OUT для управляющей передачи;

– хост отправляет пакет данных нулевой длины для фазы состояния OUT.

Пакет данных OUT с нулевой длинной в управляющих передачах

Пакет данных OUT с нулевой длинной используется для указания конца передачи управления. При нормальной работе такие пакеты должны приниматься только после того, как будет передан весь запроса устройства (т.е. после того, как ЦП установит DataEnd). Если, однако, хост отправляет пакет данных OUT с нулевой длиной до передачи всего запроса устройства, то это говорит о преждевременном завершении передачи. В этом случае контроллер автоматически очистит все токены IN, загруженные ЦП, готовые для фазы передачи данных из FIFO, и установит SetupEnd (CSR0.D4).

22.3.2 Управляющие передачи в режиме хоста

Управляющие передачи в режиме хоста проводятся через конечную точку 0, а программное обеспечение необходимо для обработки всех стандартных запросов к устройству, которые могут быть отправлены или получены через конечную точку 0 (как описано в спецификации универсальной последовательной шины, редакция 2.0, глава 9).

Что касается USB периферии, необходимо обработать три категории стандартных запросов к устройствам: запросы нулевой информации (в которых вся информация включена в команду); запросы на запись (в котором за командой последуют

дополнительные данные); запросы на чтение (в которых устройство должно отправить данные обратно на хост).

– Запросы нулевых данных содержат команду SETUP, за которой следует фаза состояния IN

– Запросы на запись включают команду SETUP, за которой следует фаза передачи данных OUT, которая, в свою очередь, сопровождается фазой состояния IN.

– Запросы на чтение включают команду SETUP, а затем фазу передачи данных IN, которая, в свою очередь, сопровождается фазой состояния OUT.

Таймаут может быть установлен так, чтобы ограничить время, в течение которого USB контроллер будет повторять транзакцию, которая не может быть получена целью (NAK). Этот предел может составлять от 2 до 215 кадров/микрокадров и устанавливается через регистр NAKLimit0.

Примечание – Перед началом любых транзакций в качестве хоста, необходимо установить регистр FAddr для адресации периферийного устройства. Если устройство подключается первым, в FAddr должен быть установлен ноль. После того, как появляется команда SET_ADDRESS, в FAddr должен быть установлен новый адрес цели.

22.3.2.1 *Фаза установки в качестве хоста*

Для фазы настройки управляющей транзакции ЦП, управляющий хост устройством, должен:

1 Загрузить восемь байт требуемой команды запроса устройства в FIFO конечной точки 0.

2 Затем, он должен установить SetupPkt и TxPtRdy (бит CSR0.D3 и CSR0.D1, соответственно). Примечание – Эти биты необходимо установить вместе.

3 Затем USB контроллер отправит токен SETUP, за которым последует восьмибайтная команда для конечной точки 0 адресного устройства, и при необходимости повторит попытку.

4 В конце попытки отправки данных, контроллер USB должен сгенерировать прерывание конечной точки 0 (т.е. установить IntrTx.D0). Затем ЦП должен прочитать CSR0, чтобы узнать, был ли установлен бит RxStall (D2), бит Error (D4) или NAK Timeout (D7).

Если RxStall установлен, то цель не приняла команду (например, потому что она не поддерживается целевым устройством) и поэтому выдала ответ STALL.

Если установлена ошибка, это означает, что USB контроллер попытался отправить пакет SETUP и следующий пакет данных три раза, не получив ответа.

Если установлен NAK Timeout, то USB контроллер получил ответ NAK, для каждой попытки отправить пакет SETUP за время большее, чем время, установленное в регистре NAKLimit0. Затем USB контроллер может либо продолжить попытки передачи этой транзакции (пока время передачи снова не истечет), очистив бит времени ожидания NAK, либо прервать транзакцию, очистив FIFO, прежде чем сбросить бит времени ожидания NAK.

Если ни один из параметров RxStall, Error или NAK Timeout не установлен, то фаза настройки прошла, и ЦП должен перейти к следующей фазе передачи данных IN, OUT или фазе состояния IN, указанной для конкретного стандартного запроса к устройству.

22.3.2.2 *Фаза передачи данных IN в качестве хоста*

Для фазы передачи данных IN управляющей транзакции ЦП, управляющий хост-устройством, должен:

1 Установить ReqPkt (CSR0.D5).

2 Подождать, пока USB контроллер отправит токен IN и получит требуемые данные.

3 Когда контроллер USB сгенерирует прерывание конечной точки 0 (то есть установит IntrTx.D0), ЦП должен прочитать CSR0, чтобы установить, какой из бит RxStall (D2), Error (D4), NAK Timeout(D7) или RxPktRdy (D0) был установлен:

– Если RxStall установлен, то цель выдала ответ STALL.

– Если установлен Error, то USB контроллер попытался отправить требуемый токен три раза и не получил ответ.

– Если установлен NAK Timeout, то USB контроллер получил NAK, для каждой попытки отправить пакет SETUP за время превышающее установленное в регистре NAKLimit0. Затем USB контроллер может либо продолжить попытки передать эту транзакцию, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив ReqPkt перед сбросом бита NAK Timeout.

4 Если RxPktRdy был установлен, ЦП должен прочитать данные из FIFO конечной точки 0, а затем сбросить RxPktRdy.

5 Если ожидаются дополнительные данные, ЦП должен повторить шаги 1 – 4.

6 Когда все данные были успешно получены, ЦП должен перейти на фазу состояния OUT контрольной транзакции.

22.3.2.3 *Фаза передачи данных OUT в качестве хоста*

Для фазы передачи данных OUT управляющей транзакции процессор, управляющий хост-устройством, должен:

1 Загрузить данные, которые нужно отправить в FIFO конечной точки 0

2 Установить бит TxPtRdy (CSR0.D1).

3 Затем контроллер USB должен отправить токен OUT, за которым последуют данные из FIFO в конечную точку 0 адресного устройства. Этот этап повторится при необходимости.

В конце попытки отправить данные USB контроллер должен сгенерировать прерывание конечной точки 0 (т.е. установить IntrTx.D0). Затем ЦП должен прочитать CSR0, чтобы узнать какой из бит RxStall (D2), Error (D4) или NAK Timeout (D7) был установлен.

Если RxStall установлен, то цель выдала ответ STALL.

Если установлен Error, то контроллер USB попытался отправить токен OUT и следующий пакет данных три раза, не получив ответа.

Если установлен NAK Timeout, то USB контроллер получил ответ NAK, для каждой попытки отправить токен OUT, за время превышающее установленное в регистре NAKLimit0. Затем USB контроллер может либо продолжить попытки отправки этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, очистив FIFO, прежде чем сбросить бит NAK Timeout.

Если ни один из бит RxStall, Error или NAK Limit не установлен, данные OUT были корректно получены.

4 Если необходимо отправить дополнительные данные, ЦП должен повторить шаги 1 – 3.

Когда все данные были успешно отправлены, ЦП должен перейти в фазу состояния IN контрольной транзакции.

22.3.2.4 *Фаза состояния IN в качестве хоста (следует за фазой SETUP или за фазой передачи данных OUT)*

Для фазы состояния IN контрольной транзакции, ЦП, управляющий хост-устройством, должен:

1 Установить StatusPkt и ReqPkt (биты CSR0.D6 и CSR0.D5).

Примечание – эти биты необходимо установить вместе.

2 Подождать, пока контроллер USB отправит токен IN и получит ответ от USB периферии.

3 Когда контроллер USB сгенерирует прерывание конечной точки 0 (то есть установит IntrTx.D0), ЦП должен прочитать CSR0, чтобы установить, какой из бит RxStall (D2), Error (D4), NAK Timeout (D7) или RxPktRdy (D0) был установлен.

Если RxStall установлен, то цель не смогла выполнить команду и поэтому выдала ответ STALL.

Если Error установлен, то контроллер USB попытался отправить требуемый токен три раза и не получил ответ.

Если NAK Timeout установлен, то USB контроллер получил ответ NAK для каждой попытки отправить токен IN за время, превышающее установленное в регистре NAKLimit0. Затем контроллер может либо продолжить отправку этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив ReqPkt и StatusPkt перед сбросом NAK Timeout.

4 Если RxPktRdy установлен, ЦП должен просто сбросить RxPktRdy.

22.3.2.5 *Фаза состояния OUT в качестве хоста (следует за фазой передачи данных IN)*

Для фазы состояния OUT управляющей транзакции, ЦП, управляющий хост-устройством, должен:

1 Установить StatusPkt и TxPktRdy (бит CSR0.D6 и CSR0.D1).

Примечание – Эти биты необходимо установить вместе.

2 Дождаться, пока контроллер USB отправит токен OUT и пакет DATA1 с нулевой длиной.

3 В конце попытки отправить данные контроллер USB сгенерирует прерывание конечной точки 0 (т.е. установит IntrTx.D0). Затем ЦП прочитает CSR0, чтобы узнать, какой из бит RxStall (D2), Error (D4) или NAK Timeout (D7) был установлен:

Если RxStall установлен, то цель не смогла выполнить команду и поэтому выдала ответ STALL.

Если Error установлен, то USB контроллер попытался отправить пакет STATUS и следующий пакет данных три раза, но не получил ответ.

Если NAK Timeout установлен, это означает, что контроллер USB получил ответ NAK для каждой попытки отправить токен IN за время большее, чем установлено в регистре NAKLimit0. Затем контроллер может либо продолжить попытки этой транзакции, сбросив бит NAK Timeout, либо прервать транзакцию, сбросив FIFO, прежде чем сбросить NAK Timeout.

4 Если ни один из RxStall, Error или NAK Timeout не установлен, фаза STATUS была проведена правильно.

22.4 Поточные (Bulk) передачи транзакций

22.4.1 Обработка поточных транзакций как периферийное устройство

22.4.1.1 *Поточные транзакции IN*

Четыре дополнительных параметра доступны для использования с конечной точкой Tx, используемой в периферийном режиме для поточных передач транзакций IN:

– Двухпакетная буферизация

За исключением случаев с использованием Dynamic FIFO, если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке, двойная буферизация пакетов будет активирована автоматически.

– DMA

Если DMA включена для конечной точки, запрос DMA будет генерироваться всякий раз, когда конечная точка сможет принять другой пакет в свой FIFO.

– AutoSet

Когда функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет установлен, после загрузки пакета размера TxMaxP байт в FIFO.

– Автоматическое разделение пакетов

Этот параметр конфигурации позволяет активировать запись больших пакетов данных в поточные конечные точки, которые затем разделяются на пакеты соответствующего (заданного) размера для передачи по шине USB.

Настройка

При настройке конечной точки Tx для поточных транзакций в регистр TxMaxP должен быть записан размер (в байтах) максимального размера пакета для конечной точки. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания в регистре IntrTxE должен быть установлен (если для этой конечной точки требуется прерывание), а старший байт регистра TxCSR должен быть установлен, как показано ниже (биты D9-D8 не используются):

D15	**AutoSet**	0/1	Установите значение 1, если требуется функция AutoSet
D14	**ISO**	0	Установите значение 0, чтобы включить поточный протокол
D13	**Mode**	1	Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только, если FIFO используется совместно с конечной точкой Rx)
D12	**DMAReqEnab**	0/1	Установите значение 1, если требуются запросы DMA. Примечание – Если установлено 1, также необходимо установить выбранный режим в DMAReqMode (TxCSR.D10)
D11	**FrcDataTog**	0	Установите значение 0, чтобы установить нормальное Data Toggle

Если конечная точка сконфигурирована впервые (при помощи команды SET_CONFIGURATION или SET_INTERFACE в конечной точке 0), бит ClrDataTog (D6) должен быть установлен в младший байт. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Также, если в FIFO есть какие-либо пакеты данных (установлен бит FIFONotEmpty (TxCSR.D1)), их следует удалить, установив бит FlushFIFO (TxCSR.D3).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

Процесс работы

Когда нужно передать данные по поточному каналу IN, пакет данных нужно загрузить в FIFO, и бит TxPktRdy (D0) нужно установить в регистр TxCSR. Когда пакет будет отправлен, контроллер сбросит бит TxPktRdy и сгенерирует прерывание, по которому можно загружать следующий пакет в FIFO. Если включена двойная буферизация пакетов, то после того, как первый пакет был загружен и установлен бит TxPktRdy, контроллер немедленно сбросит бит TxPktRdy и сгенерирует прерывание, по которому можно загружать второй пакет в FIFO. Программное обеспечение должно работать одинаково, загружая пакет, когда оно получает прерывание, независимо от того, включена ли двойная буферизация пакетов или нет.

В общем случае размер пакета не должен превышать размер, указанный нижними 11 битами регистра TxMaxP. Эта часть регистра определяет полезную нагрузку (размер пакета) для передачи по USB и требуется по спецификации USB для 8-, 16-, 32-, 64- (full-/high-speed) или 512-ти байт (high-speed). Если необходимо передать данные большего размера, то нужно отправить их как несколько USB-пакетов, которые должны быть размера TxMaxP [D10: D0], за исключением последнего пакета, размер которого неважен.

Если была выбрана опция автоматического разделения пакетов при настройке ядра (это можно определить по биту MPTxE (D6) регистра ConfigData), то в FIFO могут быть записаны пакеты в 32 раза превышающие размер, указанный в TxMaxP [D10:D0] (предполагается, что FIFO достаточно велик, чтобы принимать эти пакеты), которые затем разделятся ядром на пакеты нужного размера для передачи по USB. Размер пакетов, записанных в FIFO, задается как

m • полезная нагрузка,

где TxMaxP [D15: D11] = m - 1.

ПО должно установить соответствующие значения в регистр TxMaxP и в биты 10:0 поля wMaxPacketSize стандартного дескриптора конечной точки. Сам же процесс передачи больших пакетов ничем не отличается от процесса поточной передачи пакета стандартного размера.

Обработка ошибок

Если программное обеспечение хочет остановить поточный канал IN, оно должно установить бит SendStall (TxCSR.D4). Когда контроллер получит следующий токен IN, он отправит STALL на хост, установит бит SentStall (TxCSR.D5) и сгенерирует прерывание.

Если программное обеспечение получит прерывание с битом SentStall (TxCSR.D5), оно должно будет сбросить бит SentStall. Тем не менее оно должно оставить бит SendStall (TxCSR.D4) до тех пор, пока оно не будет готово повторно активировать поточный канал IN.

Примечание – Если хост не смог получить STALL по какой-либо причине, он отправит еще один токен IN, поэтому рекомендуется оставить бит SendStall установленным до тех пор, пока программное обеспечение не будет готово повторно активировать поточный канал IN. После включения канала, необходимо сбросить Data Toggle, установив бит ClrDataTog в регистре TxCSR (D6).

22.4.1.2 *Поточная транзакции OUT в качестве периферии*

Четыре дополнительных функции доступны для использования с конечной точкой Rx, используемой в периферийном режиме для поточных транзакций OUT.

– Двухпакетная буферизация

За исключением случая использования Dynamic FIFO, если значение, записанное в регистр RTxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке, то двойная буферизация пакетов будет активирована автоматически.

– DMA

Если DMA включена для конечной точки, запрос DMA будет генерироваться всякий раз, когда в FIFO конечной точки есть пакет данных.

– AutoClear

Когда функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сброшен, когда пакет размера RxMaxP байт будет выгружен из FIFO.

– Автоматическое объединение пакетов

Этот параметр конфигурации позволяет активировать чтение пакетов данных, которые затем объединяется в пакеты большего размера для последующего их чтения программным обеспечением.

Настройка

При настройке конечной точки Rx для поточных транзакций OUT, необходимо записать максимальный размер пакета (в байтах) для конечной точки в регистр RxMaxP. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания должен быть

установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание), а старший байт регистра RxCSR должен быть установлен, как показано ниже (биты D10 - D8 не используются/только для чтения):

D15	**AutoClear**	0/1	Установите значение 1, если требуется функция AutoClear
D14	**ISO**	0	Установите значение 0, чтобы включить поточный протокол передачи
D13	**DMAReqEnab**	0/1	Установите значение 1, если требуются запросы DMA. Примечание – Если установлено 1, также необходимо установить выбранный режим в DMAReqMode (RxCSR.D11)
D12	**DisNyet**	0	Установите значение 0, чтобы использовать нормальный контроль потока PING

Если конечная точка конфигурируется впервые (после команды SET_CONFIGURATION или SET_INTERFACE конечной точке 0), бит ClrDataTog (D7) должен быть установлен в младший байт RxCSR. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Также, если в FIFO уже есть какие-либо пакеты данных (указано в бите RxPktRdy (RxCSR.D0)), их следует удалить, установив бит FlushFIFO (RxCSR.D4).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

Процесс работы

После получения пакета данных поточной конечной точкой Rx, установится бит RxPktRdy (RxCSR.D0) и сгенерируется прерывание. Программное обеспечение должно прочитать регистр RxCount для конечной точки, чтобы определить размер пакета данных. Чтобы бит RxPktRdy сбросился, пакет данных следует прочитать из FIFO.

Полученные пакеты данных не должны превышать размер, указанный в регистре RxMaxP (так как это должно быть значение, указанное в поле wMaxPacketSize дескриптора конечной точки, отправленное на хост). Когда блок данных, размер которого превышает wMaxPacketSize, должен быть отправлен в функцию, он будет отправлен как несколько пакетов. Все пакеты будут размером wMaxPacketSize, за исключением последнего.

Если включена опция автоматического объединения пакетов (это можно определить по биту MPRxE (D7) в регистре ConfigData), то ядро может принимать до 32 пакетов одновременно и объединять их в один пакет в FIFO (предполагается, что FIFO достаточно велик, чтобы хранить более крупные пакеты). Размер пакетов, записанных в FIFO, определяется как

$$m \bullet wMaxPacketSize,$$

где RxMaxP [D15: D11] = m - 1.

Программное обеспечение, должно установить соответствующие значения в регистре RxMaxP и в биты 10:0 поля wMaxPacketSize дескриптора конечной точки.

Обработка ошибок

Если программное обеспечение хочет отключить поточный канал OUT, оно должно установить бит SendStall (RxCSR.D5). Когда USB контроллер получит следующий пакет, он отправит STALL на хост, установит бит SentStall (RxCSR.D6) и сгенерирует прерывание.

Когда программное обеспечение получит прерывание с битом SentStall (RxCSR.D6), оно должно сбросить его. Однако оно должно держать бит SendStall (RxCSR.D5) установленным до тех пор, пока оно не будет готово повторно активировать поточный канал OUT.

Примечание – Если хост не смог получить STALL по какой-либо причине, он отправит еще один пакет, поэтому рекомендуется оставить бит SendStall установленным до тех пор, пока программное обеспечение не будет готово повторно активировать поточный канал OUT. Когда поточный канал OUT снова включится, Data Toggle следует перезапустить, установив бит ClrDataTog в регистр RxCSR (D7).

22.4.2 Обработка поточных передач транзакций как хост

22.4.2.1 *Поточные передачи транзакции IN как хост*

Пять дополнительных функций доступны для использования с конечной точкой Rx, используемой в режиме хоста, для получения данных:

– Двухпакетная буферизация

Когда включена двойная буферизация пакетов, один пакет может быть принят во время чтения другого. За исключением случая, когда используется Dynamic FIFO, двойная буферизация пакетов будет автоматически включена, если значение, записанное в регистр RxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке.

– DMA

Если DMA включен для конечной точки, запрос DMA будет генерироваться всякий раз, когда в FIFO конечной точки есть пакет.

– AutoReq(uest)

Когда включена функция AutoReq(uest), бит ReqPkt (RxCSR.D5) будет автоматически устанавливаться при сбросе бита RxPktRdy.

– AutoClear

Если функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сброшен, когда пакет размера RxMaxP байт будет выгружен из FIFO.

– Автоматическое объединение пакетов

Этот параметр конфигурации позволяет активировать чтение пакетов данных, которые затем объединяется в пакеты большего размера для последующего их чтения программным обеспечением.

Настройка

Перед инициализацией любой поточной передачи транзакции IN в режиме хоста:

– Должен быть установлен адрес целевой функции.

В регистре RxType используемой конечной точкой должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 10 (для выбора режима поточной передачи) и в битах D3 - D0 указан номер конечной точки, указанный в дескрипторе соответствующей конечной точки IN, присвоенной в процессе перечисления устройств.

– В регистр RxMaxP конечной точки должен быть записан максимальный размер пакета для передачи (в байтах). Это значение должно совпадать со значением в поле MaxPacketSize стандартного дескриптора конечной точки.

– В регистр RxInterval должно быть записано значение NAK Limit (2 – 215 кадров/микрокадров) или ноль, если функция NAK Timeout не требуется.

– Соответствующий бит разрешения прерывания должен быть установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание)

– Биты регистра RxCSR должны быть установлены, как показано ниже:

D15	**AutoClear**	0/1	Установите значение 1, если требуется функция AutoClear
D14	**AutoReq**	0/1	Установите значение 1, если требуется функция AutoReq
D13	**DMAReqEnab**	0/1	Установите значение 1, если для этой конечной точки требуется запрос DMA. Примечание – Если бит установлен, также необходимо установить DMAReqMode (RxCSR.D11)
D12	**DisNyet**	0	Установите значение 0, чтобы обеспечить нормальный контроль потока PING

Если конечная точка конфигурируется впервые, то Data Taggle должен быть установлен на 0 либо с помощью битов Data Toggle Write Enable и Data Toggle (RxCSR.D10 и D9), либо путем установки бита ClrDataTog (D7) в младшем байте RxCSR. Также, если в FIFO уже есть какие-либо пакеты данных (определяется по биту RxPktRdy (RxCSR.D0)), их следует удалить, установив бит FlushFIFO (RxCSR.D4).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

Процесс работы

Когда потребуется совершить поточную передачу данных с периферийного устройства USB, ЦП должен установить бит ReqPkt в соответствующий регистр RxCSR (D5). Затем контроллер отправит токен IN в выбранную конечную точку периферийного устройства и будет ожидать возвращения данных.

Если данные принимаются правильно, устанавливается RxPktRdy (RxCSR.D0). Если периферийное устройство отвечает STALL-ом, устанавливается RxStall (RxCSR.D6). Если получен NAK, контроллер будет повторять отправку до тех пор, пока транзакция не передастся успешно или не будет установлен бит NAKLimit в регистре RxInterval. Если никакого ответа не получено вообще, контроллер предпримет еще две дополнительные попытки, после чего он сообщит об ошибке (установит RxCSR.D2).

Затем контроллер сгенерирует соответствующее прерывание конечной точки, после чего ЦП должен прочитать соответствующий регистр RxCSR, чтобы определить, какой бит из RxPktRdy, RxStall, Error или NAK Timeout установлен. Если установлен бит NAK Timout, контроллер либо продолжит пытаться отправить эту транзакцию, сбросив бит NAK Timeout, либо прервет передачу транзакции, сбросив ReqPkt, прежде чем сбросить бит NAK Timeout.

Полученные пакеты не должны превышать размер, указанный в регистре RxMaxP. Если необходимо отправить блок данных больший, чем RxMaxP, то он будет отправлен как несколько пакетов.

Если опция автоматического объединения пакетов была выбрана при настройке ядра (это можно определить по биту MPRxE (D7) в регистре ConfigData), то ядро может принимать до 32 пакетов одновременно и объединять их в один пакет в FIFO (предполагается, что FIFO достаточно велик, чтобы хранить более крупные пакеты). Размер пакетов, записанных в FIFO, определяется как m × wMaxPacketSize, где RxMaxP [D15: D11] = m - 1. ПО должно задать соответствующие значения в регистре RxMaxP и те же значение в биты 10:0 в поле wMaxPacketSize дескриптора конечной точки.

Все пакеты будут размером wMaxPacketSize, за исключением последнего.

Обработка ошибок

Если цель захочет закрыть поточный канал IN, она отправит ответ STALL на токен IN. Это приведет к установке бита RxStall (RxCSR.D6).

22.4.2.2 *Поточные транзакции out как хост*

Четыре дополнительных функции доступны для использования с конечной точкой Tx, используемой в режиме хоста, для поточной передачи данных:

– Двухпакетная буферизация

За исключением случая использования Dynamic FIFO, двойная буферизация пакетов будет активирована автоматически, если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. До двух пакетов может храниться в ожидании передачи на периферийное устройство.

– DMA

Если DMA включена для конечной точки, запрос DMA будет генерироваться всякий раз, когда конечная точка сможет принять пакет в своем FIFO.

– AutoSet

Если функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет установлен, когда пакет размера TxMaxP байт будет загружен в FIFO.

– Автоматическое разделение пакетов

Этот параметр конфигурации позволяет активировать запись больших пакетов данных в поточные конечные точки, которые затем разделяются на пакеты соответствующего (заданного) размера для передачи по шине USB.

Настройка

Перед началом любых поточных передач транзакций OUT необходимо:

– Установить адрес целевой функции.

В регистр TxType соответствующей конечной точки, должны быть записаны биты D7, D6, для выбора скорости работы, бит D5, D4 = 10 (для выбора режима поточной передачи) и в битах D3 - D0, указан номер конечной точки, указанный в дескрипторе конечной точки OUT, присвоенный в процессе перечисления устройств.

– В регистр TxMaxP конечной точки должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно быть таким же, как и в поле wMaxPacketSize стандартного дескриптора конечной точки.

– В регистр TxInterval должно быть записано значение NAK Limit (2 - 215 кадров/микрокадров) или ноль, если функция NAK Timeout не требуется.

– Бит разрешения прерывания должен быть установлен в регистре IntrTxE (если для этой конечной точки требуется прерывание).

– Следующие биты регистра TxCSR должны быть установлены, как показано ниже:

D15	**AutoSet**	0/1	Установите значение 1, если требуется функция AutoSet
D13	**Mode**	1	Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только в случае, если FIFO используется совместно с конечной точкой Rx)
D12	**DMAReqEnab**	0/1	Установите значение 1, если для этой конечной точки требуется запрос DMA. Примечание – Если установлено значение 1, также необходимо выбрать режим DMAReqMode (TxCSR.D10)
D11	**FrcDataTog**	0	Установите значение 0, чтобы разрешить нормальное Data Toggle

Если конечная точка конфигурируется впервые, то Data Toggle конечных точек должен быть установлен на 0 либо с помощью битов Data Toggle Write Enable и Data Toggle (TxCSR.D9 и D8), либо путем установки бита ClrDataTog (D6) в младший байт TxCSR. Это гарантирует, что Data Toggle (которое автоматически обрабатывается контроллером) запустится в правильном состоянии. Кроме того, если в FIFO есть какие-либо пакеты данных (определяется по биту FIFONotEmpty (TxCSR.D1)), их следует удалить, установив бит FlushFIFO (TxCSR.D3).

Примечание – Возможно, потребуется установить этот бит дважды, если включена двойная буферизация.

Процесс работы

Когда требуется совершить поточную передачу данных, ЦП должен записать первый пакет данных в FIFO (или два пакета, если используется двойной буфер), и установить бит TxPktRdy в соответствующий регистр TxCSR (D0). Затем контроллер отправит токен OUT в выбранную конечную точку периферийного устройства, а затем первый пакет данных из FIFO.

Если данные правильно приняты периферийным устройством, придет ACK, после чего контроллер сбросит TxPktRdy (TxCSR.D0). Если периферийное устройство USB ответит STALL-ом, то установится RxStall (TxCSR.D5). Если получен NAK, будет

повторять передачу до тех пор, пока транзакция не будет успешно завершена или не будет установлен NAKLimit в регистре TxInterval. Если никакого ответа вообще не получено, произойдет еще две дополнительные попытки отправить данные, а после контроллер сообщит об ошибке (TxCSR.D2).

Затем контроллер сгенерирует соответствующее прерывание конечной точки, после чего ЦП должен будет считать соответствующий регистр TxCSR, чтобы определить, установлены из бит RxStall (D5), Error (D2) или NAK Timeout (D7) установлен. Если установлен бит NAK Timeout, контроллер либо продолжит попытки отправки этой транзакции, сбросив бит NAK Timeout, либо прервет передачу, очистив FIFO, прежде чем сбросить бит NAK Timeout.

В общем случае размер пакета не должен превышать размер, указанный младшими 11-ю битами регистра TxMaxP (который должен был быть установлен в соответствии со значением, установленным в поле wMaxPacketSize соответствующего дескриптора конечной точки). Когда необходимо отправить блок данных с размером, превышающим TxMaxP, его нужно будет отправить как несколько пакетов. Эти пакеты должны быть размера TxMaxP [D10: D0], за исключением последнего пакета.

Если была выбрана опция автоматического разделения пакетов при конфигурации ядра (это можно определить по настройке бита MPTxE (D6) регистра ConfigData), то, пакеты с размером в 32 раза превышающим размер, указанный в TxMaxP [D10: D0], могут быть записаны в FIFO (предполагается, что FIFO достаточно велик, чтобы принимать большие пакеты), которые затем разобьются ядром на пакеты соответствующего размера для передачи по USB. Размер пакетов, записанных в FIFO, задается как m × USB-полезная нагрузка, где TxMaxP [D15: D11] = m - 1. ПО должно задать соответствующие значения в регистре TxMaxP.

Если общий размер блока данных кратен TxMaxP, хосту может потребоваться отправить нулевой пакет после отправки всех данных. Это можно сделать, установив TxPktRdy после получения последнего прерывания и не загружая каких-либо данных в FIFO.

Обработка ошибок

Если цель хочет выключить поточный канал OUT, она должна отправить ответ STALL. Это обозначается битом RxStall (TxCSR.D5).

22.5 Использование DMA

22.5.1 Использование DMA с поточной конечной точкой TX

Для конечных точек Tx линия DMA запросов активна, когда FIFO конечной точки способно принять пакет данных и неактивна, когда TxMaxP байт загружено в FIFO. В качестве альтернативы, линия станет неактивной, если бит TxPktRdy в TxCSR будет установлен.

Чтобы использовать DMA для отправки большого блока данных на USB хост через поточную конечную точку Tx, мы рекомендуем настроить контроллер DMA и USB контроллер следующим образом.

Контроллер DMA должен быть настроен для считывания пакетов с максимальным размером данных для конечной точки (512 байт для high-speed, 64 байта для full-speed), когда линия DMA запросов для конечной точки перейдет из неактивного состояния к активному. Контроллер должен продолжать выполнять записи при каждом активном состоянии DMA линии до тех пор, пока весь блок данных не будет передан. (Последний пакет может быть меньше максимального размера пакета). Затем он должен прервать процессор.

Контроллер USB следует запрограммировать с включенными режимами AutoSet и DMA Request Mode 1 путем установки бит AutoSet, DMAReqEnab и DMAReqMode в регистре TxCSR (бит D15, D12 и D10 соответственно).

Запрограммированный таким образом, USB контроллер будет реагировать на активную линию DMA, каждый раз, когда в его FIFO есть место, чтобы принять пакет. Кроме того, бит TxPktRdy будет автоматически установлен после того, как контроллер DMA загрузит в FIFO пакет максимального размера. Теперь пакет готов к отправке на хост. Когда последний пакет будет загружен контроллером DMA, он должен прервать процессор. Если последний загруженный пакет был меньше максимального размера, то бит TxPktRdy не был установлен, и поэтому его необходимо установить вручную (то есть с помощью ЦП), чтобы разрешить отправку последнего пакета. Бит TxPktRdy также должен быть установлен вручную, если последний пакет был максимального размера, и должен быть отправлен нулевой пакет для указания конца передачи.

22.5.2 Использование DMA с поточной конечной точкой RX

Поведение линии запросов DMA для конечной точки Rx зависит от режима запроса DMA, выбранного через регистр RxCSR (D11). В режиме 0 линия DMA запросов Rx активизируется, когда пакет данных доступен в конечной точке FIFO и становится неактивной либо при чтении последнего байта пакета данных, либо, когда бит RxPktRdy в RxCSR сбрасывается. Также будет сгенерировано соответствующее прерывание конечной точки Rx (если включено). В режиме 1 линия DMA запросов активизируется только, когда полученный пакет имеет максимальный размер (установленный в регистре RxMaxP). Если полученный пакет имеет другой размер, линия DMA запроса останется неактивной, в результате пакет останется в FIFO с установленным RxPktRdy. Это приведет к генерации прерывания конечной точки Rx (если включено).

22.6 Full-speed/low-bandwidth (низкопропускные) передачи по прерываниям

22.6.1 Передачи по прерываниям как периферия

Передача по прерываниям IN использует тот же протокол, что и поточная передача IN, и может использоваться таким же образом. Точно так же передача по прерываниям OUT использует почти тот же протокол, что и поточная передача OUT, и может использоваться таким же образом.

Тем не менее, следует отметить, что конечные точки Tx USB контроллера, который используется в качестве периферийного устройства, поддерживают одну

функцию передачи по прерываниям IN, которую они не поддерживают в поточных передачах IN – они поддерживают непрерывное переключение бита Data Toggle. Эта функция активируется установкой бита FrcDataTog в регистре TxCSR (D11). Когда этот бит установлен, контроллер посчитает пакет успешно отправленным и переключит data toggle, независимо от того, получен ли ACK от хоста или нет.

Другое отличие заключается в том, что конечные точки по прерываниям не поддерживают управление потоком PING. Это означает, что контроллер никогда не должен отвечать на пакет квитирования NYET, только на ACK / NAK / STALL. Для этого бит DisNyet в регистре RxCSR (D12) должен быть установлен, чтобы отключить передачу пакетов квитирования NYET в high-speed режиме.

22.6.2 Передачи прерываний как хост

Когда контроллер USB работает в качестве хоста, взаимодействие с конечной точкой по прерываниям на периферии USB обрабатываются почти так же, как поточные передачи, исключением того, что разрешены high-bandwidth передачи по прерываниям.

Основное различие заключается в том, что RxType [5: 4] и TxType [5: 4] необходимо установить равным 11 (для представления передачи Interrupt), а не 10.

Необходимый интервал опроса также необходимо установить в регистры RxInterval/TxInterval.

22.7 Full-speed/low-bandwidth (низкопропускные) изохронные передачи

22.7.1 Работа с изохронными передачами как с периферией

22.7.1.1 *Изохронные передачи IN*

Три дополнительных функции доступны для использования с конечной точкой Tx в периферийном режиме для изохронной передачи транзакций IN:

– Двойная буферизация пакетов

Если не используется Dynamic FIFO, двойная буферизация пакетов автоматически включится, если значение, записанное в регистр TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. Если включено, в FIFO может находиться до двух пакетов в ожидании передачи на хост.

– DMA

Если включена, запрос DMA будет генерироваться каждый раз, когда конечная точка будет иметь возможность принять другой пакет в своем FIFO.

– AutoSet

Когда функция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет автоматически устанавливаться, когда пакет размера TxMaxP байт будет загружаться в FIFO.

Настройка

При конфигурировании конечной точки Tx для изохронных передач IN, максимальный размер пакета (в байтах) должен быть записан в регистр TxMaxP. Это значение должно совпадать с полем wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, соответствующий бит разрешения прерывания должен быть

установлен в регистре IntrTxE (если требуется прерывание для этой конечной точки), а старший байт регистра TxCSR должен быть установлен, как показано ниже (биты D9 - D8 не используются):

D15	**AutoSet**	0/1	Установите значение 1, если требуется функция AutoSet
D14	**ISO**	1	Установите значение 1 для включения изохронного протокола передачи
D13	**Mode**	1	Установите значение 1, чтобы удостовериться, что FIFO доступен (необходимо если только FIFO используется совместно с конечной точкой Rx)
D12	**DMAReqEnab**	0/1	Установите значение 1, если для этой конечной точки требуется запрос DMA. Примечание – Если установлено значение 1, также потребуется установить режим DMAReqMode (TxCSR.D10)
D11	**FrcDataTog**	0	Игнорируется в изохронном режиме

Процесс работы

Изохронная конечная точка не поддерживает повторение попыток отправки данных, поэтому, если необходимо избежать переполнения данных отправляемых на узел, они должны загружать данные в FIFO до получения токена IN. Хост будет отправлять один токен для каждого кадра (или микрокадра в high-speed режиме), однако время в пределах кадра (или микрокадра) может меняться. Если токен IN принимается ближе к концу одного кадра, а затем в начале следующего кадра, то времени для перезагрузки FIFO будет мало. По этой причине обычно необходима двойная буферизация конечной точки.

Функция AutoSet может использоваться с изохронной конечной точкой Tx так же, как с поточной конечной точкой Tx. Однако, если данные не поступают из источника с постоянной скоростью, синхронизированной с часами кадра хоста, размер пакетов, отправленных хосту, должен увеличиваться или уменьшаться от кадра к кадру (или от микрокадра к микрокадру), чтобы соответствовать исходной скорости передачи данных. Это означает, что фактические размеры пакетов не всегда будут TxMaxP, что делает функцию AutoSet бесполезной.

Прерывание генерируется всякий раз, когда пакет отправляется на хост, и программное обеспечение может использовать это прерывание для загрузки следующего пакета в FIFO и установки бита TxPktRdy в регистр TxCSR (D0) так же, как для поточной конечной точки Tx. Поскольку прерывание может происходить почти в любое время внутри кадра (/микрокадра), в зависимости от того, когда хост запланировал передачу данных, это может привести к неправильным моментам времени подачи запросов на загрузку FIFO. Если источник данных для конечной точки поступает с какого-либо внешнего оборудования, может быть удобнее дождаться конца каждого кадра (/микрокадра) до следующей загрузки FIFO, поскольку это минимизирует потребность в дополнительной буферизации. Это можно сделать, используя прерывание SOF (IntrUSB.D3) или внешний SOF_PULSE-сигнал от USB контроллера, чтобы инициировать загрузку следующего пакета данных. SOF_PULSE генерируется один раз на кадр (/микрокадр), когда принимается пакет SOF. (контроллер также поддерживает внешний

счетчик кадров (/микрокадров), чтобы он мог генерировать SOF_PULSE, когда пакет SOF был потерян.) Прерывания все еще могут использоваться для установки бита TxPktRdy в TxCSR (D0).

Источником проблем может стать запуск изохронного IN канала с двойной буферизацией. Двойная буферизация требует, чтобы пакет данных не передавался до тех пор, пока кадр (/микрокадр) не будет загружен. Проблем нет, если функция загружает первый пакет данных, или по крайней мере, кадр (/микрокадр), прежде чем хост настроит канал (и, следовательно, начнет посылать токены IN). Но если хост уже начал посылать токены IN к моменту загрузки первого пакета, то возможно пакет должен быть передан в том же кадре (/микрокадре), в котором он был загружен, в зависимости от того, загружен ли он до или после получения IN токена. Эту потенциальную проблему можно избежать, установив бит ISO Update в регистр Power (D7). Когда этот бит установлен, любой пакет данных, загружаемый в FIFO изохронной конечной точки Tx, не будет передаваться до тех пор, пока не будет принят следующий пакет SOF, тем самым гарантируя, что пакет данных не будет отправлен слишком рано.

Обработка ошибок

Если конечная точка не имеет данных в своем FIFO, когда принимается токен IN, он отправит нулевой пакет данных на хост и установит бит UnderRun в регистре TxCSR (D2). Это свидетельствует о том, что программное обеспечение не обеспечивает доставку данных достаточно быстро для хоста.

Если программное обеспечение загружает один пакет на кадр (/микрокадр), и обнаруживает, что бит TxPktRdy в регистре TxCSR (D0) установлен, когда он хочет загрузить следующий пакет, это означает, что пакет данных не был отправлен (возможно, потому что токен IN от хоста был поврежден). ПО может либо сбросить неотправленный пакет, установив бит FlushFIFO в регистр TxCSR (D3) либо может пропустить этот пакет.

22.7.1.2 *Изохронные передачи OUT*

Три дополнительных функции доступны для конечной точки Rx, используемой в периферийном режиме для изохронных передач транзакций OUT:

– Двойная буферизация пакетов

За исключением использования Dynamic FIFO, двойная буферизация пакетов автоматически включается, когда значение, записанное в регистр RxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. При включении этой опции, в FIFO может быть сохранено до двух пакетов, ожидающих передачи на хост.

Примечание – Для предотвращения синхронных передач данных рекомендуется использовать двухбуквенную буферизацию, чтобы избежать ошибок при переполнении (см. «Процесс работы» ниже).

– DMA

Если включена, запрос DMA будет генерироваться каждый раз, когда у конечной точки есть пакет в FIFO.

– AutoClear

Когда функция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет автоматически сбрасываться, когда пакет размера RxMaxP байт будет выгружен из FIFO.

Настройка

При конфигурировании конечной точки Rx для изохронной передачи данных OUT, максимальный размер пакета (в байтах) для конечной точки должен быть записан в регистр RxMaxP. Это значение должно совпадать со значение в поле wMaxPacketSize стандартного дескриптора конечной точки. Кроме того, должен быть установлен соответствующий бит разрешения прерывания в регистре IntrRxE (если для этой конечной точки требуется прерывание), а старший байт регистра RxCSR должен быть установлен, как показано ниже (биты D10 – D8 не используются / только для чтения):

D15	**AutoSet**	0/1	Установите значение 1, если требуется опция AutoSet
D14	**ISO**	1	Установите значение 1 для включения изохронного протокола
D13	**Mode**	1	Установите значение 1, чтобы удостовериться, что FIFO доступен (необходимо если только FIFO используется совместно с конечной точкой Rx)
D12	**DMAReqEnab**	0/1	Установите значение 1, если для этой конечной точки требуется запрос DMA. Примечание – Если установлено значение 1, также потребуется установить режим DMAReqMode (RxCSR.D10)
D11	**FrcDataTog**	0	Игнорируется в изохронном режиме

Процесс работы

Изохронная конечная точка не поддерживает повторения попыток передачи данных, поэтому, если необходимо избежать переполнения данных, в FIFO должно быть место для приема пакета при его получении. Хост отправляет один пакет на кадр (или микрокадр в high-speed режиме), однако время в пределах кадра может меняться. Если пакет принят ближе к концу одного кадра (/микрокадра), а другой - к началу следующего кадра, времени для разгрузки FIFO будет мало. По этой причине обычно необходима двойная буферизация конечной точки.

Функция AutoClear может использоваться с изохронной конечной точкой Rx так же, как с поточной конечной точкой Rx. Однако, если приемник данных получает данные не с постоянной скоростью и синхронизируется с часами кадра хоста, размер пакетов, отправленных с хоста, должен увеличиваться или уменьшаться от кадра к кадру (или от микрокадра до микрокадра) до соответствия требуемой скорости передачи данных. Это означает, что фактические размеры пакетов не всегда будут равны RxMaxP, что делает функцию AutoClear бесполезной.

Обработка ошибок

Если в FIFO нет места для хранения пакета, полученного от хоста, будет установлен бит OverRun в регистре RxCSR (D2). Это свидетельствует о том, что программное обеспечение не выгружает данные достаточно быстро для хоста.

Если контроллер USB обнаружит, что принятый пакет имеет ошибку CRC, он все равно разместит пакет в FIFO и установит биты RxPktRdy (RxCSR.D0) и DataError (RxCSR.D3).

22.7.2 Работа с изохронными передачами как хоста

22.7.2.1 *Изохронные передачи IN*

Четыре дополнительных опции доступны для использования с конечной точкой Rx, используемой в режиме хоста, для получения этих данных:

– Двойная буферизация пакетов

Когда включена двойная буферизация пакетов, один пакет может быть принят, пока другой считывается. За исключением случая использования Dynamic FIFO, двойная буферизация пакетов будет активирована автоматически, если значение, записанное в регистр RxMaxP, будет меньше или равно половине размера FIFO, выделенного конечной точке.

Примечание – Двойная пакетная буферизация данных рекомендуется к использованию в синхронных передачах, чтобы избежать переполнения данных (см. «Процесс работы» ниже).

– DMA

Если опция включена, запрос DMA будет генерироваться каждый раз, когда у конечной точки будет пакет в FIFO.

– AutoClear

Если опция AutoClear включена, бит RxPktRdy (RxCSR.D0) будет сброшен, когда пакет из RxMaxP байт будет выгружен из FIFO.

– AutoReq(uest)

Если включена опция AutoReq (uest), бит ReqPkt (RxCSR.D5) будет автоматически устанавливаться, при сбросе бита RxPktRdy.

Настройка

Перед началом изохронных транзакций IN:

– Должен быть установлен адрес целевой функции.

– В регистре RxType используемой конечной точки контроллера должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 01 (для выбора режима изохронной передачи) и биты D3 - D0, хранящие значение номера конечной точки, содержащегося в дескрипторе конечной точки IN, полученном во время перечисления устройств.

– В регистр RxInterval конечной точки должен быть записан требуемый интервал передачи данных.

– В регистр RxMaxP конечной точки должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно совпадать со значением в поле wMaxPacketSize стандартного дескриптора конечной точки.

Соответствующий бит разрешения прерывания должен быть установлен в регистре IntrRxE (если для этой конечной точки требуется прерывание).

– Следующие биты регистра RxCSR должны быть установлены, как показано ниже:

D15	**AutoClear**	0/1	Установите значение 1, если требуется функция AutoClear
D14	**AutoReq**	0/1	Установите значение 1, если требуется функция AutoRequest
D13	**DMAReqEnab**	0/1	Установите значение 1, чтобы удостовериться, что FIFO доступно (необходимо если только FIFO используется совместно с конечной точкой Rx)
D12	**DisNyet**	0	Игнорируется в изохронном режиме

Процесс работы

Работа начинается с того, что ЦП устанавливает ReqPkt (RxCSR.D5). Это приводит к тому, что контроллер отправляет IN токен в цель.

Когда принимается пакет, генерируется прерывание, по которому программное обеспечение может выгружать пакет из FIFO и сбрасывать бит RxPktRdy в регистре RxCSR (D0) так же, как и для поточной конечной точки Rx. Поскольку прерывание может происходить почти в любое время внутри кадра (/микрокадра), время запросов на выгрузку FIFO, вероятно, будет непостоянным. Если поток данных конечной точки уходит на внешнее оборудование, возможно лучше свести к минимуму потребность в дополнительной буферизации, дожидаясь конца каждого кадра перед выгрузкой FIFO. Это можно сделать, используя сигнал SOF_PULSE из контроллера, чтобы вызвать выгрузку пакета данных. SOF_PULSE генерируется один раз на кадр (/микрокадр). Прерывания по-прежнему могут использоваться для сброса бита RxPktRdy в RxCSR.

Обработка ошибок

Если во время приема пакета возникает ошибка CRC или bit-stuff, пакет все равно будет размещен в FIFO, но будет установлен бит DataError (RxCSR.D3), чтобы указать, что данные могут быть повреждены.

22.7.2.2 *Изохронная передача OUT*

Три дополнительных опции доступны для использования с конечной точкой Tx, в режиме хоста, для изохронной передачи данных:

– Двойная буферизация пакетов

За исключением случая использования Dynamic FIFO, двойная буферизация пакетов включается автоматически, если значение, записанное в регистре TxMaxP, меньше или равно половине размера FIFO, выделенного конечной точке. Когда опция включена, до двух пакетов можно сохранить в FIFO для ожидания передачи на периферию.

– DMA

Если опция включена, запрос DMA будет генерироваться всякий раз, когда конечная точка будет иметь возможность принять еще один пакет в своем FIFO.

– AutoSet

Если опция AutoSet включена, бит TxPktRdy (TxCSR.D0) будет автоматически устанавливаться, когда пакет размера TxMaxP байт будет загружен в FIFO.

Настройка

Перед началом изохронной передачи OUT:

– Должен быть установлен адрес целевой функции.

– В регистр TxType конечной точки контроллера, должны быть установлены биты D7, D6 для выбора рабочей скорости, биты D5, D4 = 01 (для выбора изохронной передачи), а в биты D3 - D0 установлено значение номера конечной точки, содержащегося в дескрипторе конечной точки OUT, полученном во время перечисления устройств.

– В регистр TxInterval конечной точки должен быть записан требуемый интервал передачи (обычно одна транзакция для каждого кадра/микрокадра).

– В регистр TxMaxP конечной точки контроллера должен быть записан максимальный размер пакета (в байтах) для передачи. Это значение должно совпадать со значением в поле wMaxPacketSize стандартного дескриптора конечной.

– Должен быть установлен соответствующий бит разрешения прерывания в регистре IntrTxE (если для этой конечной точки требуется прерывание)

– Следующие биты регистра TxCSR должны быть установлены, как показано ниже:

D15	**AutoSet**	0/1	Установите значение 1, если требуется функция AutoSet
D14	**Mode**	0/1	Установите значение 1, чтобы гарантировать, что FIFO доступен (необходимо только в том случае, если FIFO используется совместно с конечной точкой Rx)
D13	**DMAReqEnab**	0/1	Установите значение 1, если для этой конечной точки требуется запрос DMA. Примечание – Если установлено значение 1, также потребуется выбирать режим DMAReqMode (TxCSR.D10)
D12	**FrcDataTog**	0	Игнорируется в изохронном режиме

Процесс работы

Работа начинается с того, что ЦП записывает данные в FIFO, а затем устанавливает TxPktRdy (TxCSR.D0). Это вынуждает USB контроллер отправить токен OUT, за которым последует первый пакет данных из FIFO.

Прерывание генерируется всякий раз, когда отправляется пакет, и ПО может по этому прерыванию загружать следующий пакет в FIFO и устанавливать бит TxPktRdy в регистре TxCSR (D0), аналогично поточной конечной точке Tx. Поскольку прерывание может происходить в любое время внутри кадра, то это может привести к неправильному времени запросов загрузки FIFO хоста при передаче. Если источник данных для конечной точки поступает с некоторого внешнего оборудования, может быть удобнее дождаться конца каждого кадра перед загрузкой FIFO, поскольку это минимизирует потребность в дополнительной буферизации. Это можно сделать, используя сигнал SOF_PULSE из контроллера, чтобы инициировать загрузку следующего пакета данных. SOF_PULSE генерируется один раз на кадр (/микрокадр). Прерывания могут использоваться для установки бита TxPktRdy в TxCSR.

Функция AutoSet может использоваться с изохронной конечной точкой Tx так же, как с поточной конечной точкой Tx. Однако, если данные поступают из источника не с постоянной скоростью, синхронизированной с часами фрейма, то размер пакетов будет увеличиваться или уменьшаться от кадра к кадру (или от микрокадра к микрокадру), чтобы соответствовать скорости источника данных. Это означает, что фактические размеры пакетов не всегда будут равны TxMaxP, что делает функцию AutoSet бесполезной.

22.8 High-bandwidth (высокопропускные) изохронные передачи/ передачи по прерываниям

Высокопропускные изохронные передачи и передачи по прерываниям используют те же самые протоколы, что и другие изохронные передачи, и передачи по прерываниям. Однако есть некоторые особенности проведения высокопропускных передач:

1 Высопропускные изохронные передачи и передачи по прерываниям могут проводиться только если ядро контроллера было сконфигурировано с поддержкой таких передач.

Ядро также необходимо настроить таким образом, чтобы конечные точки, используемые для передач данных с высокой пропускной способностью, имели FIFO достаточного размера для хранения данных по меньшей мере одного пакета высокой пропускной способности (от 1 Кбайт и до 3 Кбайт).

2 При настройке максимального размера пакета, обрабатываемого конечной точкой в регистре TxMaxP/RxMaxP, максимальное количество передач данных на микрокадр также должно быть установлено через биты D11 и D12 этого регистра.

Это максимальное количество передач (две или три) также представляет максимальное количество частей, на которые может быть разделен любой один «высокопропускной» пакет, что в свою очередь устанавливает максимальный размер пакета в два или три раза больше максимальной полезной нагрузки.

Примечание – Максимальная полезная нагрузка, которая может быть отправлена в любой транзакции, составляет 1 Кбайт.

3 При отправке пакетов бит TxPktRdy должен быть установлен программным обеспечением. Аналогично, при выгрузке пакетов из конечной точки RX, ПО должно очистить регистр RxPktRdy.

Функции AutoSet и AutoClear не могут использоваться для установки и очистки этих бит в транзакциях с высокой пропускной способностью.

4 Передача пакетов в виде набора частей приводит к возникновению еще одного типа ошибок – передаче неполных пакетов.

Для конечных точек Tx эта проблема возможна только, если контроллер находится в режиме периферийного устройства и возникает, когда контроллер не может получить достаточное количество токенов IN от хоста для отправки всех частей пакета данных. Когда это происходит, контроллер устанавливает бит IncompTx в регистре TxCSR (D7).

Для конечных точек Rx проблема возникает, когда PID из полученных частей пакета данных показывают, что одна или несколько частей пакета данных не были получены. Когда это происходит, контроллер устанавливает бит IncompRx в регистре RxCSR (D8). В основном этот бит будет устанавливаться только тогда, когда ядро работает в периферийном режиме, но его также можно установить в режиме хоста, если (и только если) устройство, с которым контроллер обменивается данными, не отвечает в соответствии с протоколом USB,

22.8.1 Подключение/Отключение

22.8.1.1 *В хост режиме*

Если контролер работает в хост режиме, ЦП начинает сессию установкой бита Session (DevCtl.D0). Затем питание подается на VBus, и ядро ожидает подключения устройства.

Когда устройство обнаружено, генерируется прерывание соединения (т.е. IntrUSB.D4 устанавливается). Скорость подключенного устройства может быть определена путем чтения регистра DevCtl, где бит FSDev (D6) будет установлен для устройства с full-/high-speed, а бит LSDev (D5) будет установлен для low-speed устройства. Затем ЦП должен перезагрузить устройство. Если установлены оба бита FSDev и HS Enab (Power.D5), USB контроллер будет пытаться согласоваться для high-speed работы. Сделает ли он это успешно, будет указано в бите HS Mode (Power.D4).

ЦП должен поддерживать бит Reset, установленным 20 мс, чтобы гарантировать сброс цели. Затем он может начать перечисление устройств.

Если устройство отключено во время сеанса, будет создано прерывание Disconnect (т.е. IntrUSB.D5 будет установлен).

22.8.1.2 *В режиме периферийного устройства*

Если USB контроллер работает в периферийном режиме, прерывание не возникнет, когда устройство подключается к хосту.

Однако прерывание Disconnect (IntrUSB.D5) сгенерируется, когда хост завершит сеанс.

22.8.2 Запрос сеанса OTG

Чтобы сэкономить энергию, дополнение USB On-The-Go позволяет VBus включаться только тогда, когда это необходимо, и отключаться, когда VBus не используется.

VBus всегда предоставляется устройством типа «A» на шине. Контроллер определяет, является ли оно устройством «A» или «B» типа, путем проверки сигнала входа IDDIG из UTMI + PHY. Этот сигнал установлен на низком уровне, если обнаружено устройство типа «A» (т.е. контроллер является устройством типа «A») и установлен на высоком уровне если обнаружено устройство типа «B» (т.е. контроллер является устройством типа «B»).

23.8.2.1 Начало сеанса

Когда требуется начать сеанс, ЦП необходимо установить бит Session в регистре DevCtl (D0). Затем USB контроллер активирует считывание PIN-кода. Это приведет к

тому, что на входе IDDIG установится сигнал низкого уровня, если обнаружено соединение типа A или высокого уровня, если обнаружено соединение типа B. Также установится бит B-Device в регистре DevCtl (D7).

Если USB контроллер является устройством типа «A»: он войдет в режим хоста (устройство «A» всегда является хостом по умолчанию), включит VBus и будет ждать, пока VBus не поднимется выше необходимого порога, что он определит по появлению положительному уровню VBUSVALID (это событие также приводит к тому, что биты Vbus [1:0] в регистре DevCtl (D4 - D3) установятся на 11b).

Далее контроллер будет ждать подключения периферии. Когда будет обнаружено периферийное устройство, сгенерируется прерывание Connect (IntrUSB.D4) (если включено), и либо бит FSDev, либо бит LSDev в регистре DevCtl (D6 / D5 соответственно) будет установлен в зависимости от того, будет ли периферия full-/high-speed или low-speed.

Затем ЦП должен перезагрузить это периферийное устройство. Если установлены биты FSDev и HSEnab (Power.D5), USB контроллер будет отслеживать сигнал LINESTATE во время сброса, чтобы узнать, получен ли высокоскоростной импульс из периферийного устройства или нет. Если импульс будет получен, контроллер ответит высокоскоростными импульсом и перейдет в режим high-speed.

Чтобы завершить сеанс, ЦП должен сбросить бит Session (DevCtl.D0).

Если USB контроллер является устройством типа «B»: он запросит сеанс, используя протокол Session Request, определенный в дополнении USB On-The-Go, то есть сначала установит DISCHRGVBUS чтобы сбросить VBus. Затем, когда VBus опустится ниже порога окончания сеанса (что определяется по высокому уровню входа SESSEND и битам VBus [1:0] в регистре DevCtl (D4 - D3), которые устанавливаются на 00b) – и состояние линии будет SE0 не менее 2 мс – контроллер отправит импульс на линию передачи данных, а затем импульс на VBus (установит высокий уровень CHRGVBUS).

В конце сеанса бит Session сбрасывается – обычно это делает контроллер, но он также может быть сброшен ЦП. Затем контроллер переключит TERMSEL, что приведет к тому, что PHY отключит подтягивающий резистор. Это укажет устройству «A» о завершении сеанса.

22.8.2.1 *Обнаружение активности*

Когда другое устройство OTG пожелает начать сеанс, оно либо повысит уровень VBus выше порога сеанса, если устройство типа «A» (на что указывает высокий уровень сигнала AVALID, и биты VBus [1: 0] в регистре DevCtl (D4 - D3), установленные на 10b), либо, если это устройство типа «B», оно сначала отправит импульс на линию передачи данных, а затем импульс на VBus. В зависимости от того, какое из этих действий происходит, контроллер сможет определить, является ли новое устройство «A» или «B» типа, и действовать соответственно следующим образом:

Если *VBus выше допустимого порога сеанса:* тогда USB контроллер является устройством типа «B». Контроллер установит бит Session в регистре DevCtl (D0). Когда на шине будет обнаружен сигнал сброса, сгенерируется прерывание Reset (IntrUSB.D2) (если включено), по которому ЦП должен начать сеанс.

Контроллер будет в периферийном режиме в этот момент, поскольку устройство «B» является периферийным по умолчанию.

В конце сеанса устройство «A» отключит питание VBus. Когда уровень VBus опустится ниже порогового значения (на что укажет понижение уровня входа AVALID, и установка бит VBus [1: 0] в регистре DevCtl (D4 - D3) на 01b), контроллер обнаружит это и сбросит бит Session чтобы показать, что сеанс закончился. Прерывание Disconnect (IntrUSB.D5) также будет сгенерировано (если включено).

Если *обнаружена передача данных/импульс VBus:* тогда USB контроллер является устройством типа «A». Он будет генерировать прерывание Session Request (IntrUSB.D6 - если включено), чтобы указать, что устройство «B» запрашивает начало сеанса. Затем ЦП должен начать сеанс, установив бит Session (DevCtl.D0).

22.8.3 Согласование роли хоста

Если USB контроллер является устройством типа «A» (IDDIG низкий, B-Device (DevCtl.D7) = 0), он автоматически перейдет в хост режим при запуске сеанса.

Если USB контроллер является устройством типа «B» (IDDIG высокий, B-Device (DevCtl.D7) = 1), он автоматически перейдет в режим периферийного устройства при запуске сеанса. Тем не менее ЦП может сделать запрос, чтобы контроллер стал хостом, установив бит Host Req в регистре DevCtl (D1). Этот бит можно установить одновременно с запросом на запуск сеанса, установив бит Session (DevCtl.D0) или в любой момент после начала сеанса.

Когда контроллер переходит в режим Suspend (нет активности на шине в течение 3 мс), то если бит Host Req установлен, он войдет в хост режим и начнет согласование хоста (как указано в дополнении USB On-The-Go) при помощи переключая TERMSEL, заставляя PHY отключить подтягивающий резистор на линии D. Это должно привести к тому, что устройство типа «A» переключится в режим периферийного устройства и подключит свой собственный подтягивающий резистор. Когда контроллер обнаружит это, он сгенерирует прерывание Connect (IntrUSB.D4), если оно включено. Он также установит бит Reset в регистре Power (D3), чтобы начать сброс устройства «A» (USB контроллер автоматически запускает эту последовательность сброса, чтобы гарантировать, что перезапуск начнется в течение 1 мс после того, как устройство «A», подключит свой подтягивающий резистор). ЦП должен подождать не менее 20 мс, а затем сбросить бит Reset и пронумеровать устройство «A».

Когда устройство типа «B» на основе USB контроллера закончило работу с шиной, ЦП должен перевести его в режим Suspend, установив бит Suspend Mode в регистр Power (D1). Устройство типа «A» должно обнаружить это и либо завершить сеанс, либо вернуться в режим хоста.

22.8.4 Действия Vbus

Спецификация USB On-The-Go определяет ряд пороговых значений:

– VBus активен (VBus Valid) (напряжение от 4,4 до 4,75 В)

– Сеанс активен (Session Valid) для устройства типа «A» (напряжение от 0,8 В до 2,1 В)

– Окончание сеанса (Session End) (напряжение от 0,2 В до 0,8 В) которым должны удовлетворять устройства, участвующие в двухточечных связях.

22.8.4.1 *Действия контроллера в роли устройства типа «А»*

Напряжение VBus больше напряжения активности VBus с сеансом, инициированным контроллером (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) = 11b и установлен бит Session (DevCtl.D0)). Когда VBus становится выше, чем уровень активности, USB контроллер входит в режим хоста и ожидает подключения устройства. После подключения он сгенерирует прерывание Connect (IntrUSB.D4). ЦП должен сбросить и перечислить подключенное устройство «В».

Напряжение VBus больше напряжения активности VBus с сеансом, инициированным устройством типа «В» (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) = 10b и сброшен бит сеанса (DevCtl.D0)). Когда VBus становится выше, чем уровень активности сеанса, контроллер сгенерирует прерывание Session Request (IntrUSB.D6). ЦП должен установить бит Session. Контроллер либо останется в режиме хоста, либо перейдет в режим периферии в зависимости от состояния подтягивающего резистора устройства «В». На выбранный режим будет указывать бит Host Mode (DevCtl.D2).

Напряжение VBus меньше напряжения активности VBus, но бит Session все еще установлен (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) ≠ 11b и установлен бит сеанса (DevCtl.D0)). Это указывает на проблему с уровнем мощности VBus. В этом случае контроллер автоматически прекратит сеанс и сгенерирует прерывание VBus Error (IntrUSB.D7).

22.8.4.2 *Действия контроллера в роли устройства типа «В»*

Напряжение VBus больше напряжения активности сессии (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) = 10b и сброшен бит Session (DevCtl.D0)). Это указывает на активность устройства типа «А». Контроллер установит бит Session и понизит уровень сигнал DPPULLDOWN, чтобы отключить подтягивающий резистор на линии D+.

Напряжение VBus меньше напряжения активности сессии, но бит Session все еще установлен (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) = 01b и установлен бит Session (DevCtl.D0)). Это означает, что устройство типа «А» потеряло питание (или отключилось). Контроллер сбросит бит Session (DevCtl.D0) и сгенерирует прерывание Disconnect (IntrUSB.D5). ЦП должен завершить сеанс.

Напряжение VBus меньше напряжения активности сессии и установленным битом Host Request (т.е. Vbus [1: 0] (DevCtl. [D4: D3]) = 00b и установлен бит DevCtl.D1). Это условие смены Хоста. После 3 мс без активности на шине контроллер войдет в режим хоста и отключит подтягивающий резистор, чтобы начать согласование для роли хоста. Когда устройство «А» подключит собственный подтягивающий резистор, контроллер сгенерирует прерывание Connect (IntrUSB.D4). Он также установит бит Reset (Power.D3), чтобы начать сброс устройства «А». ЦП должен выждать не менее 20 мс, а затем сбросить бит Reset и перечислить устройство «А».

22.8.5 Передача транзакций в качестве периферии

Примечание – Действия хоста показаны на белом фоне. Действия контроллера показаны – на голубом.

22.8.5.1 *Транзакции управления*

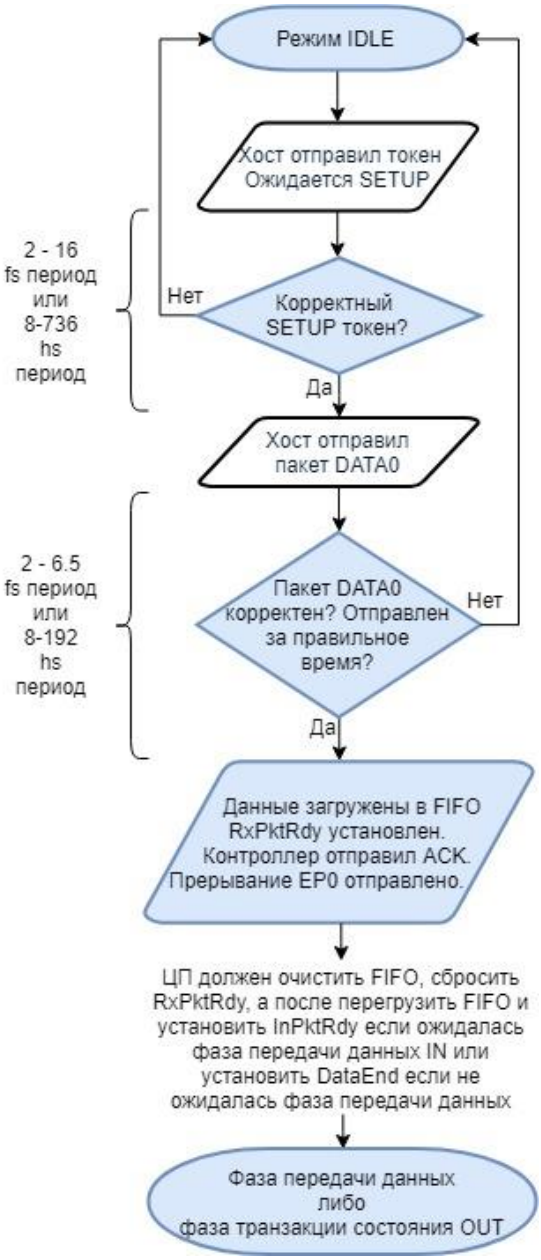


Рисунок 113 – Фаза установки

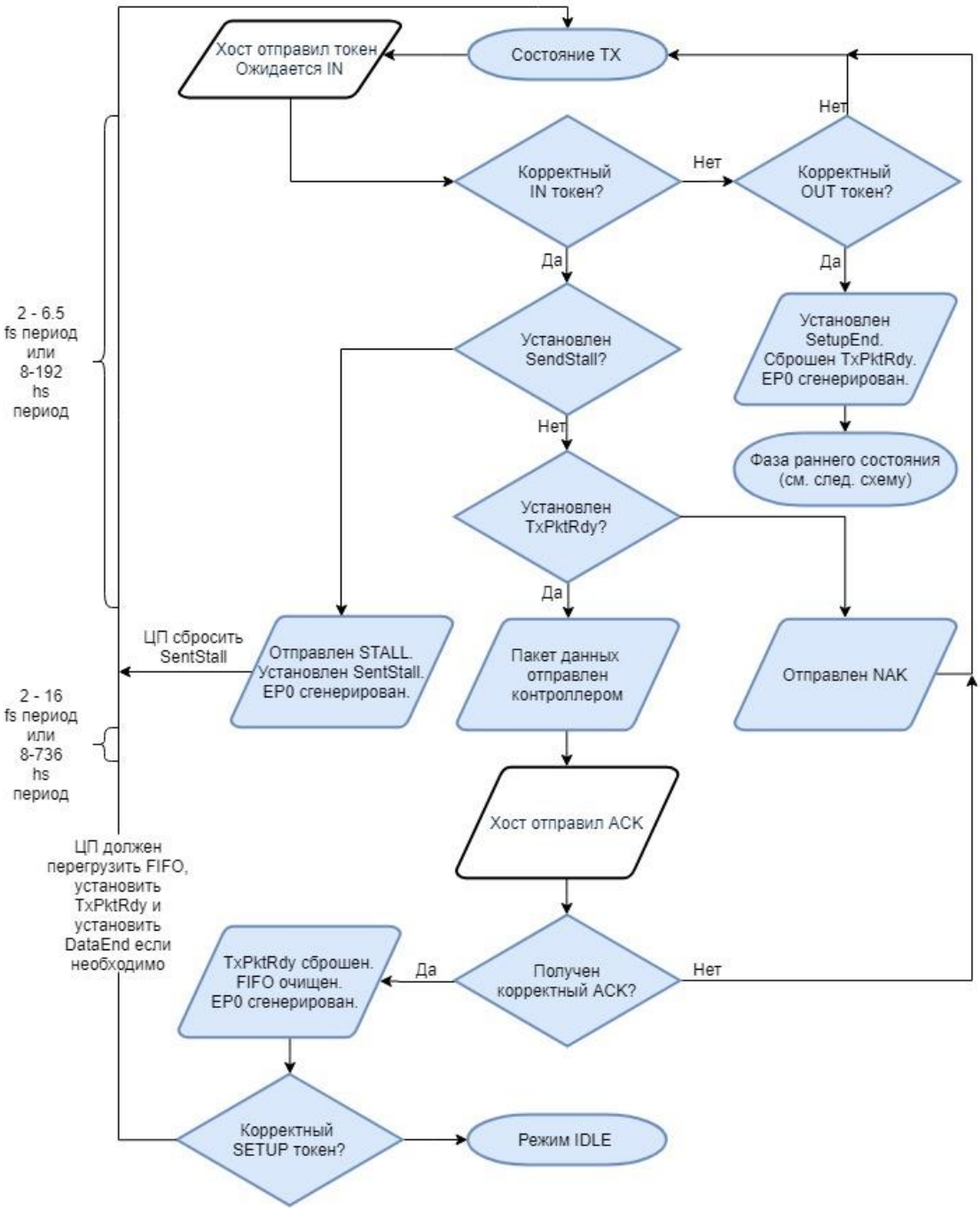


Рисунок 114 – Фаза передачи данных IN

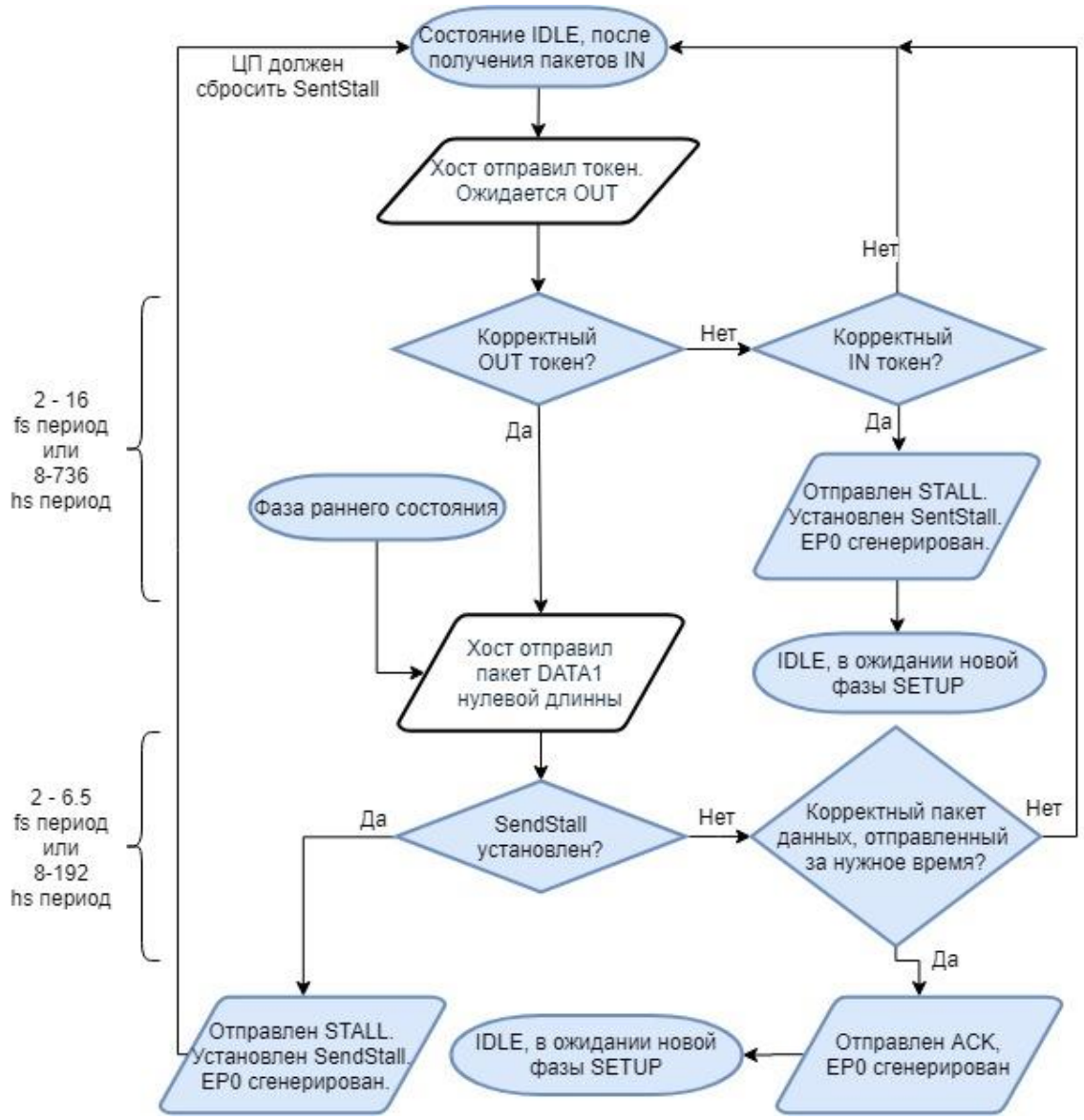


Рисунок 115 – Последующая фаза передачи состояния

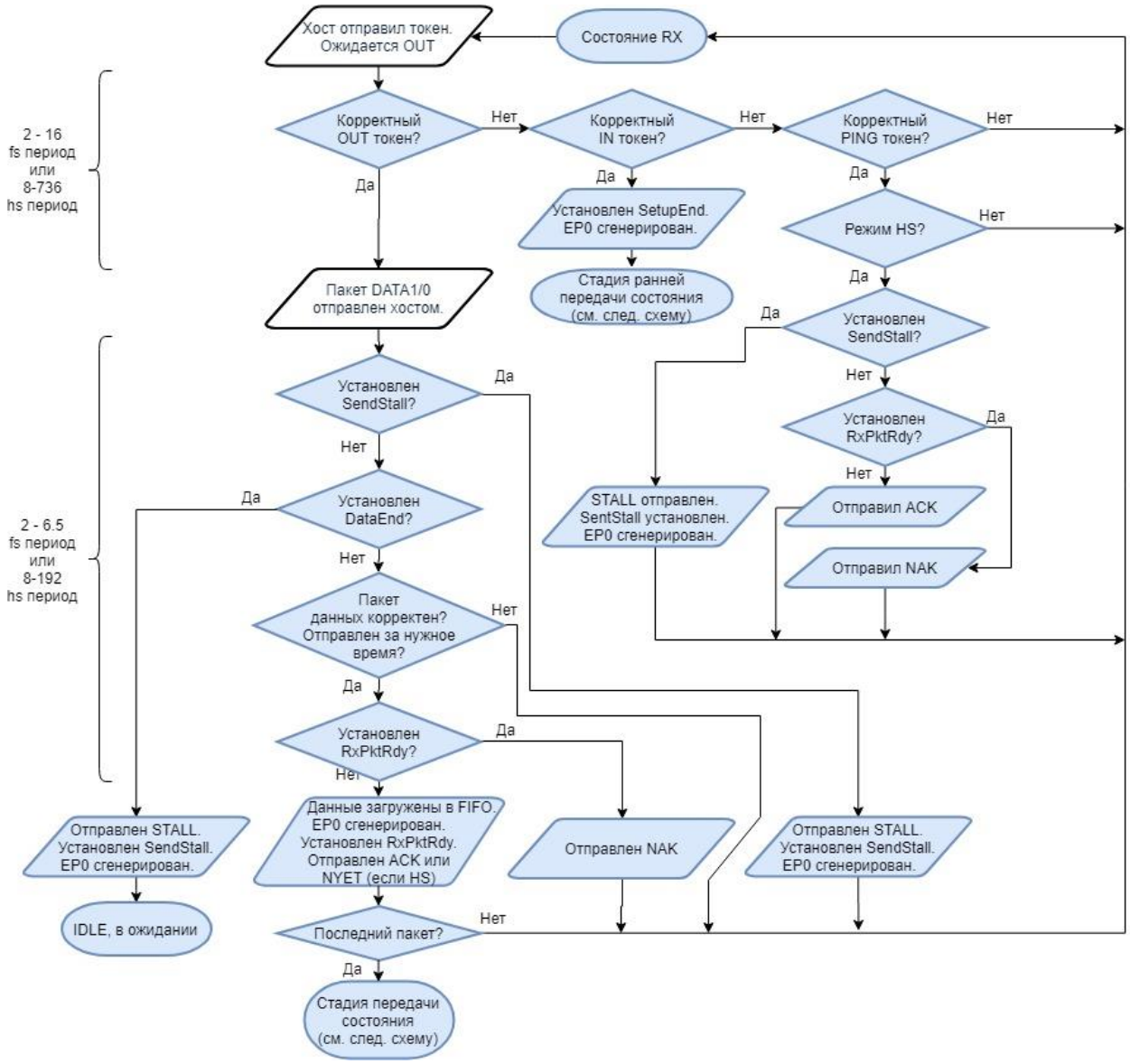


Рисунок 116 – Фаза передачи данных OUT

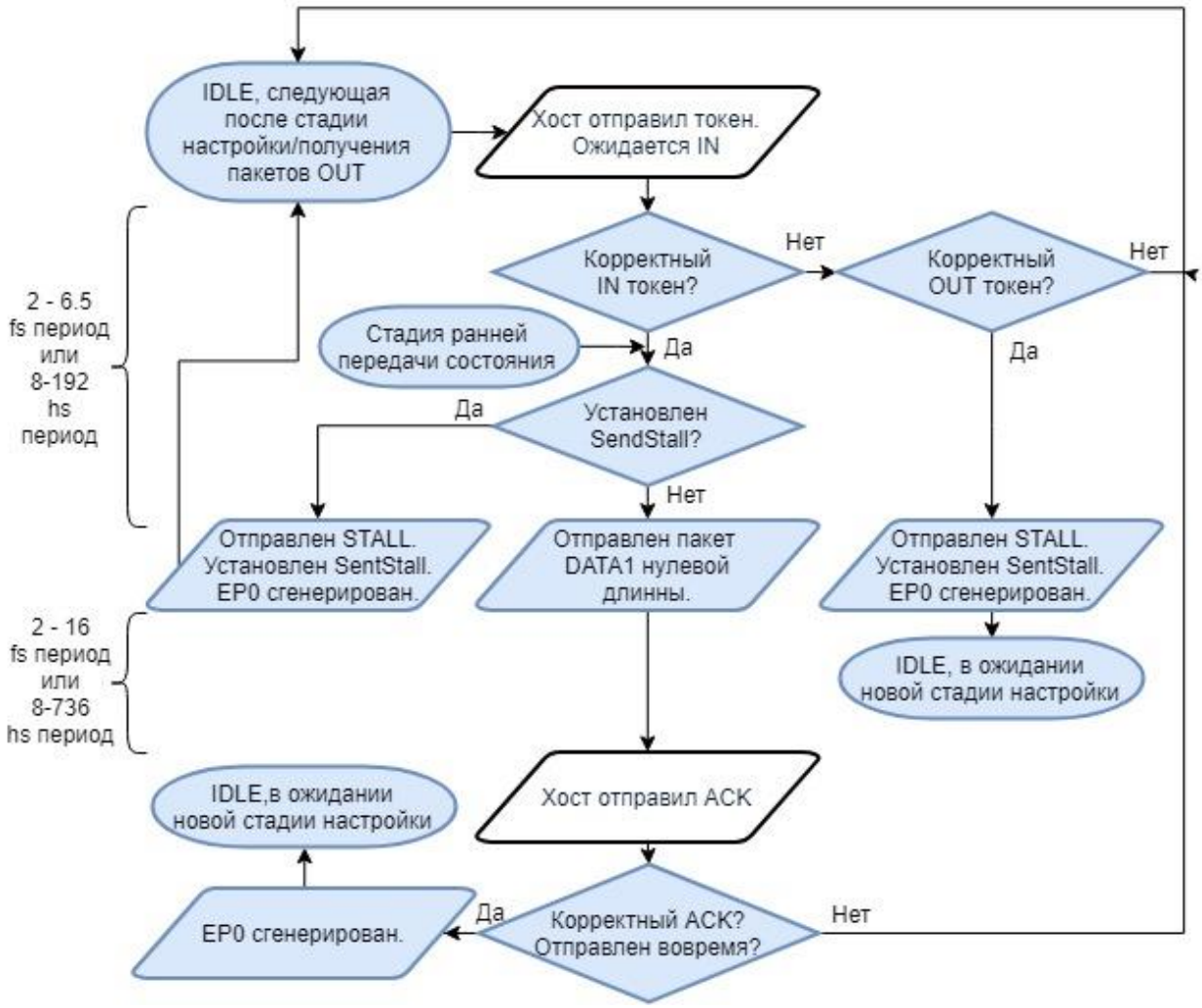


Рисунок 117 – Последующая фаза передачи состояния

22.9 Поточная передача/низкопропускная передача по прерываниям

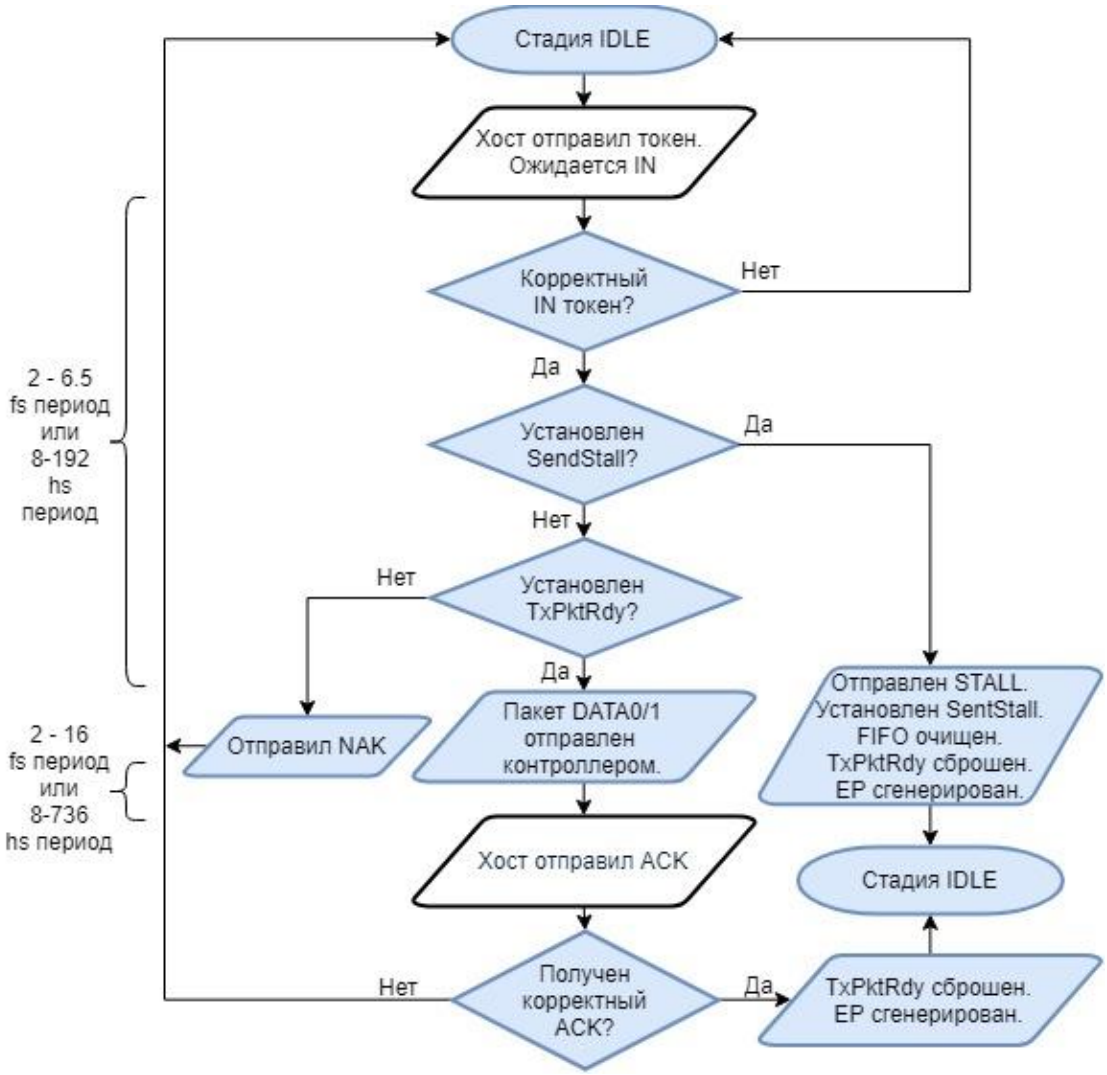


Рисунок 118 – Передача транзакций IN

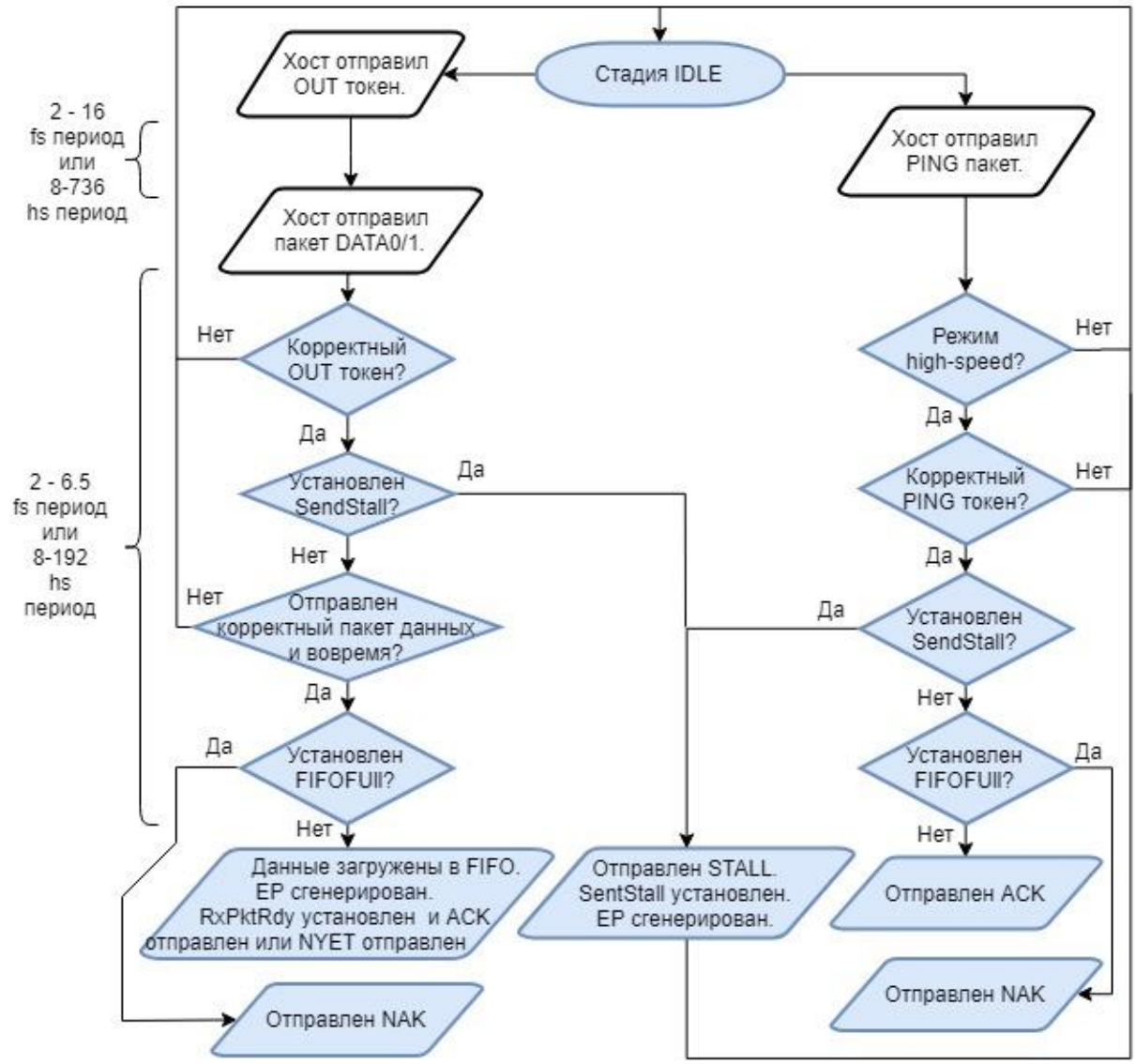


Рисунок 119 – Передача транзакций OUT

22.10 High-speed/low-bandwidth изохронные передачи

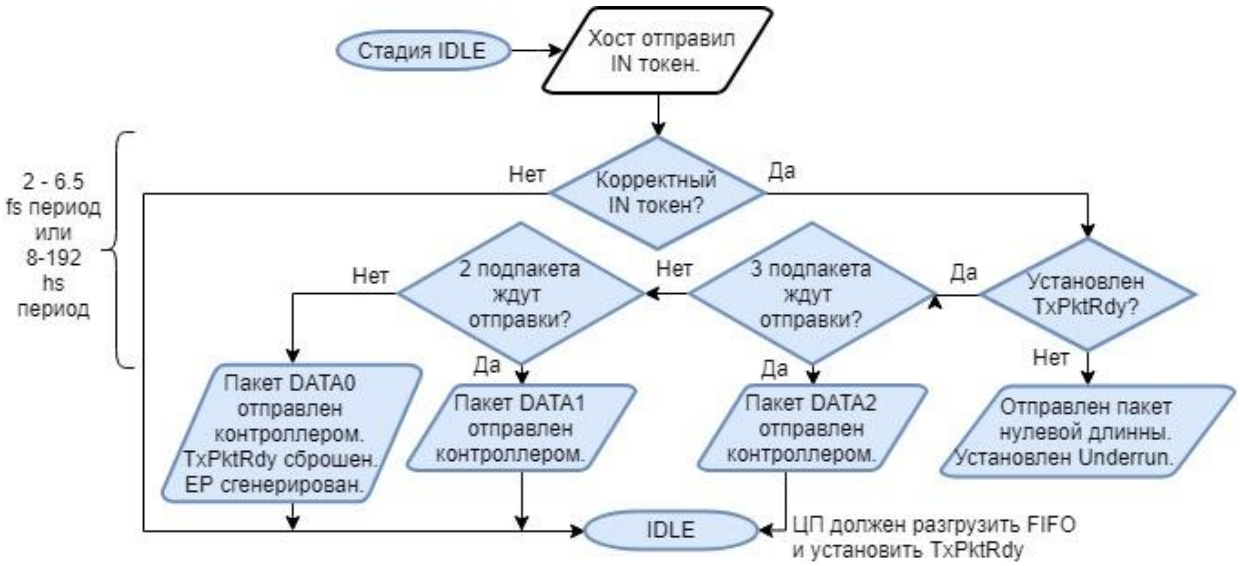


Рисунок 120 – Передача IN транзакций

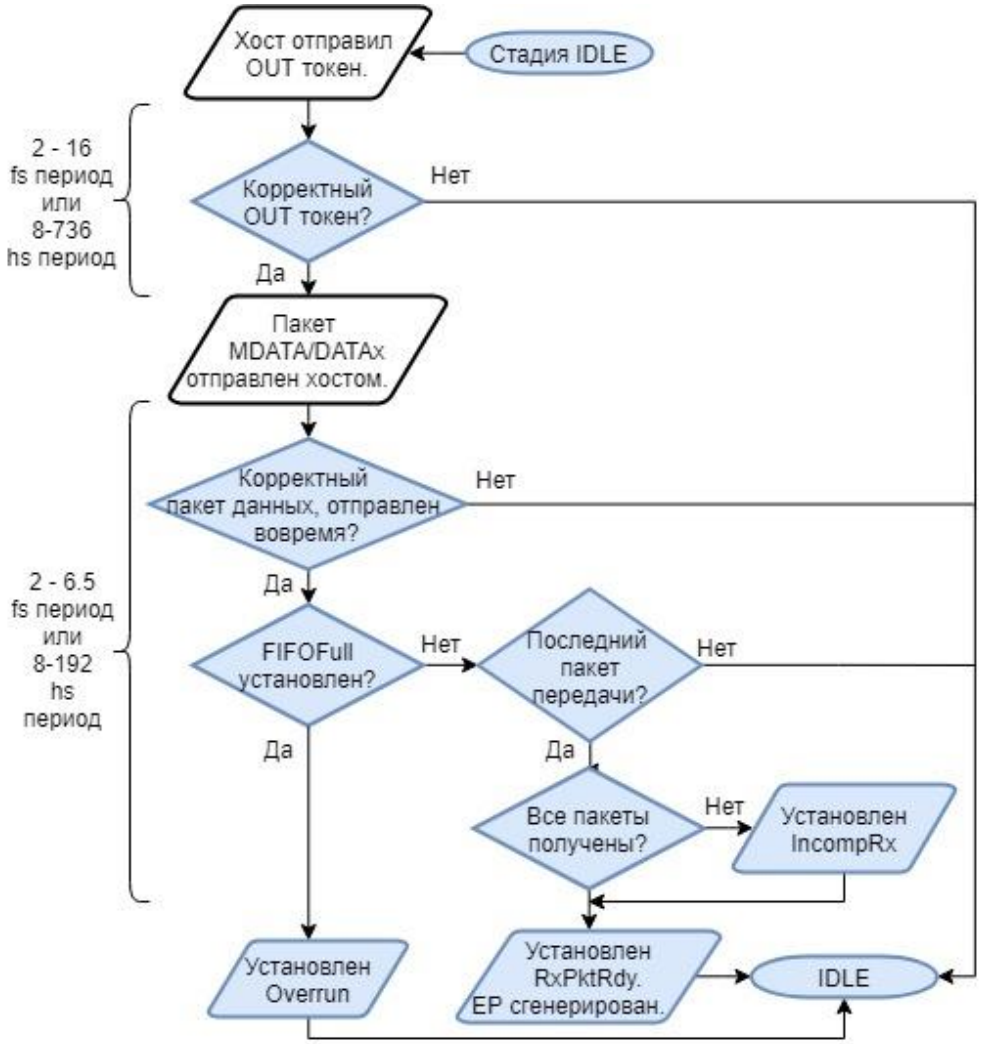


Рисунок 121 – Передача OUT транзакций

22.11 Высокопропускные передачи (изохронные/по прерываниям)

Рисунок 122 – Передача транзакций IN – слева, OUT –справа

22.12 Передача транзакций со стороны хоста

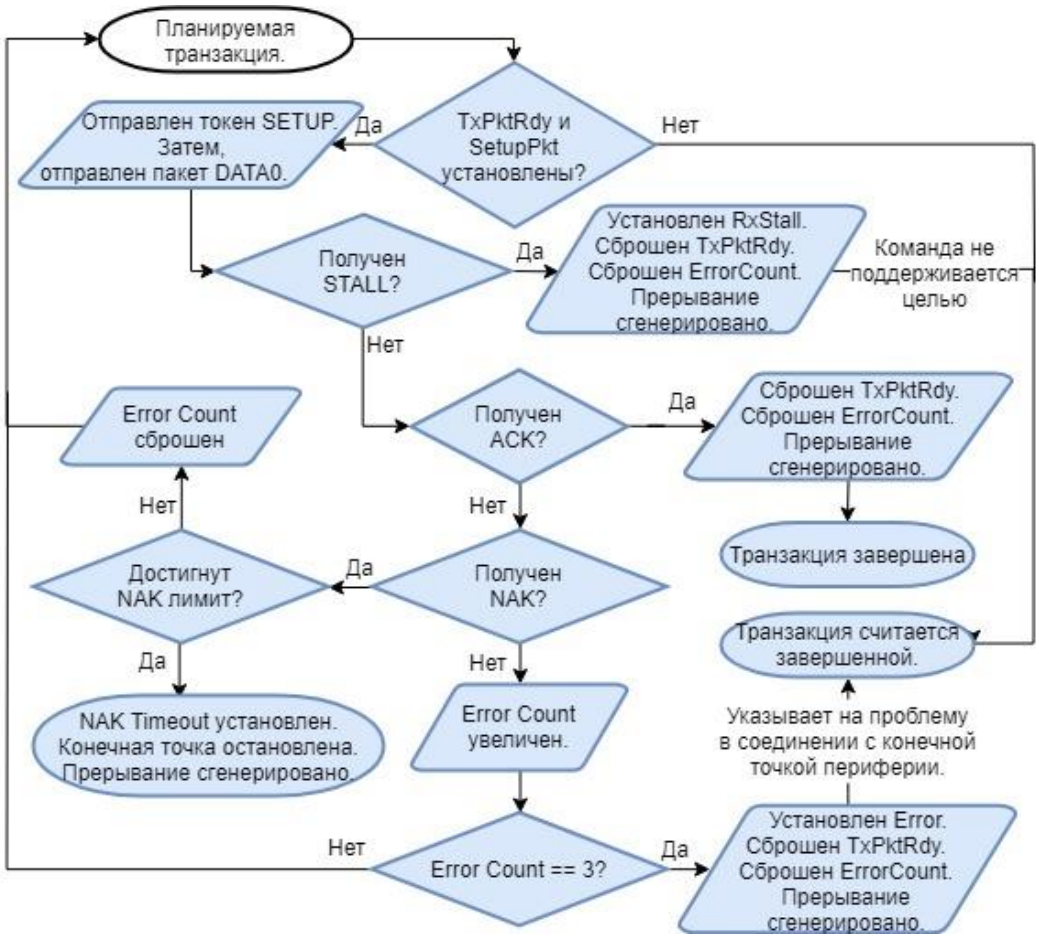


Рисунок 123 – Фаза установки

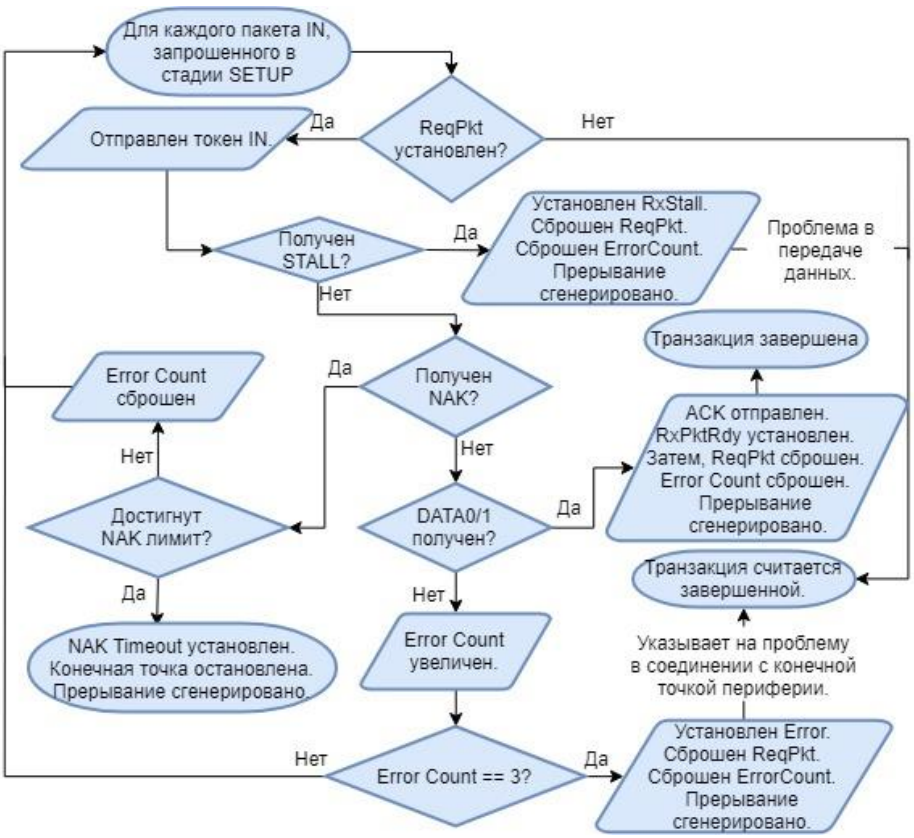


Рисунок 124 – Фаза передачи данных IN

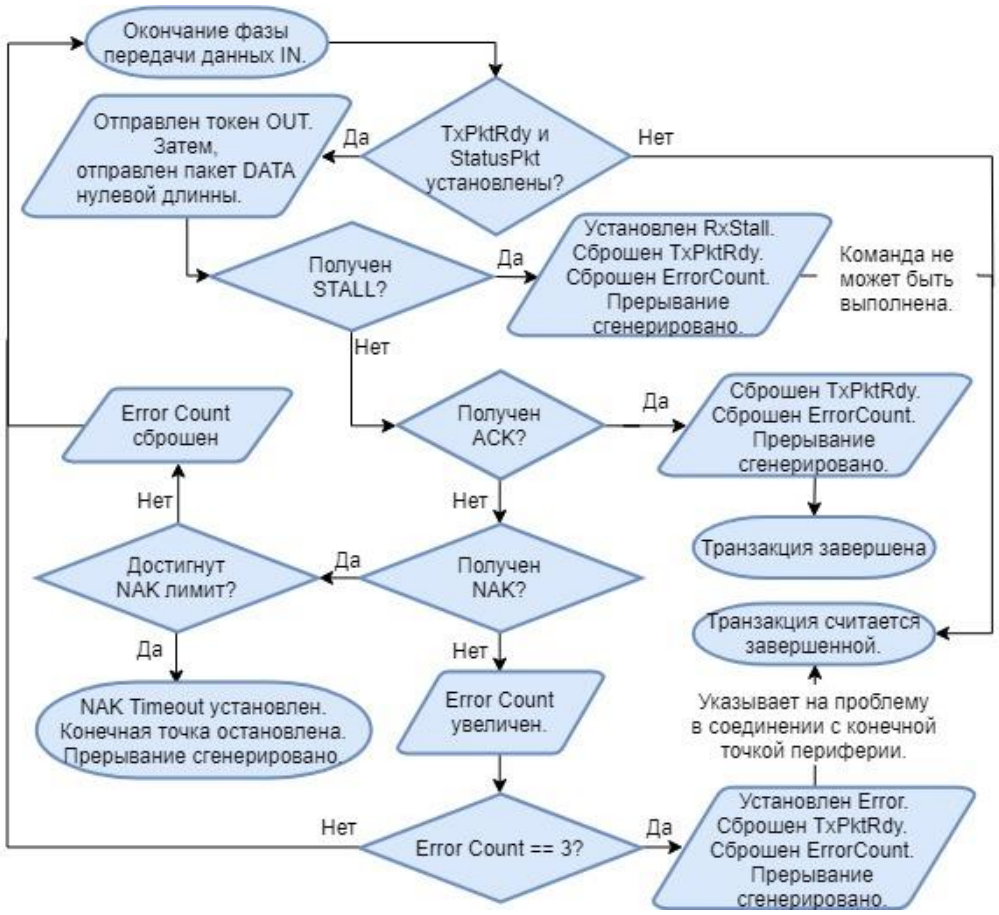


Рисунок 125 – Последующая фаза передачи состояния

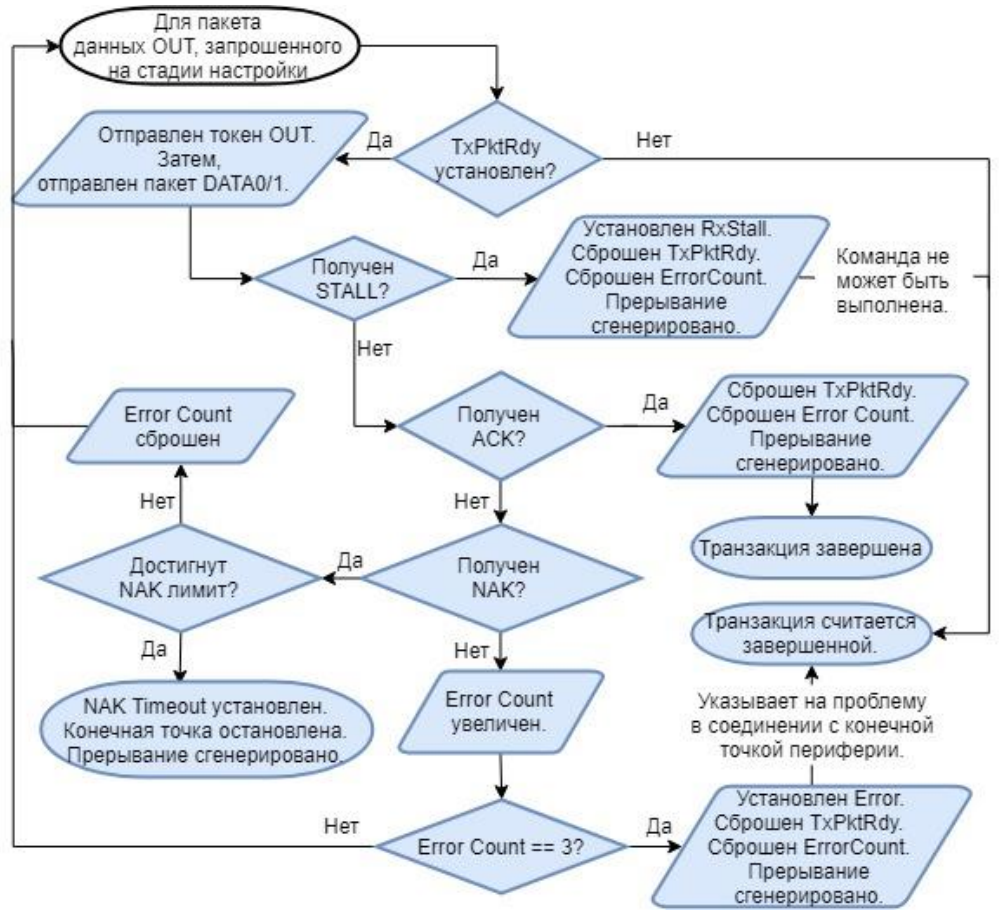


Рисунок 126 – Фаза передачи данных OUT

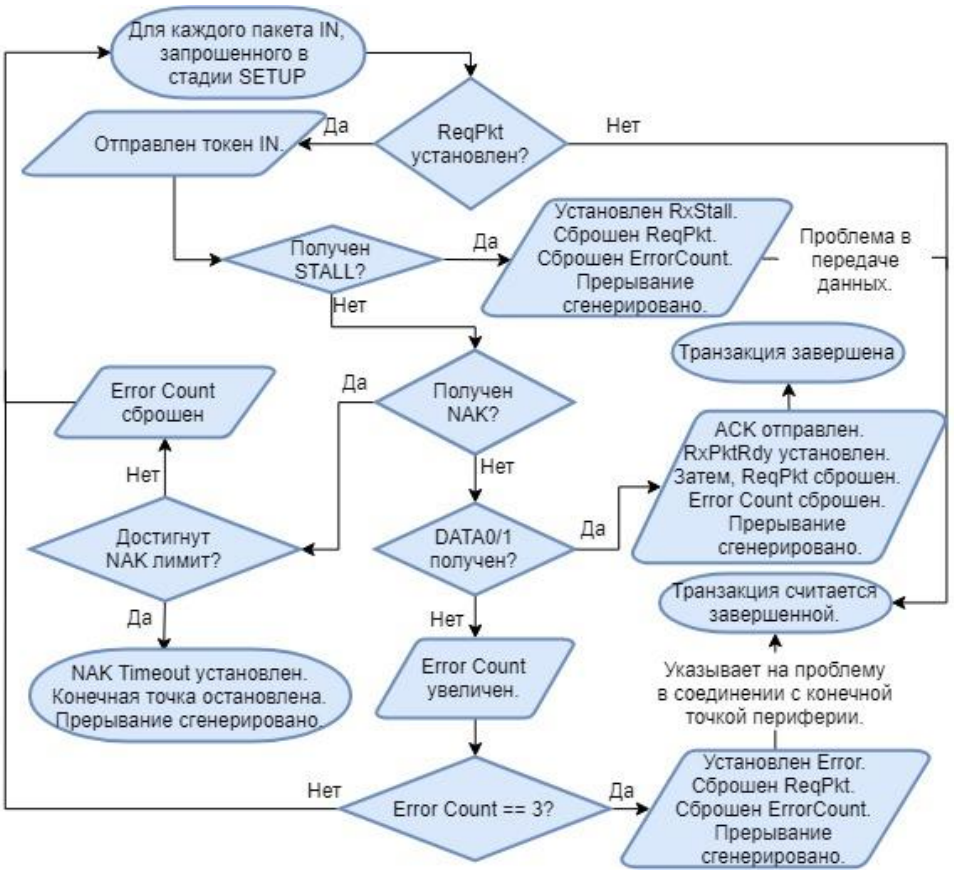


Рисунок 127 – Последующая фаза передачи состояния

22.13 Поточная передача/низкопропускная передача по прерываниям

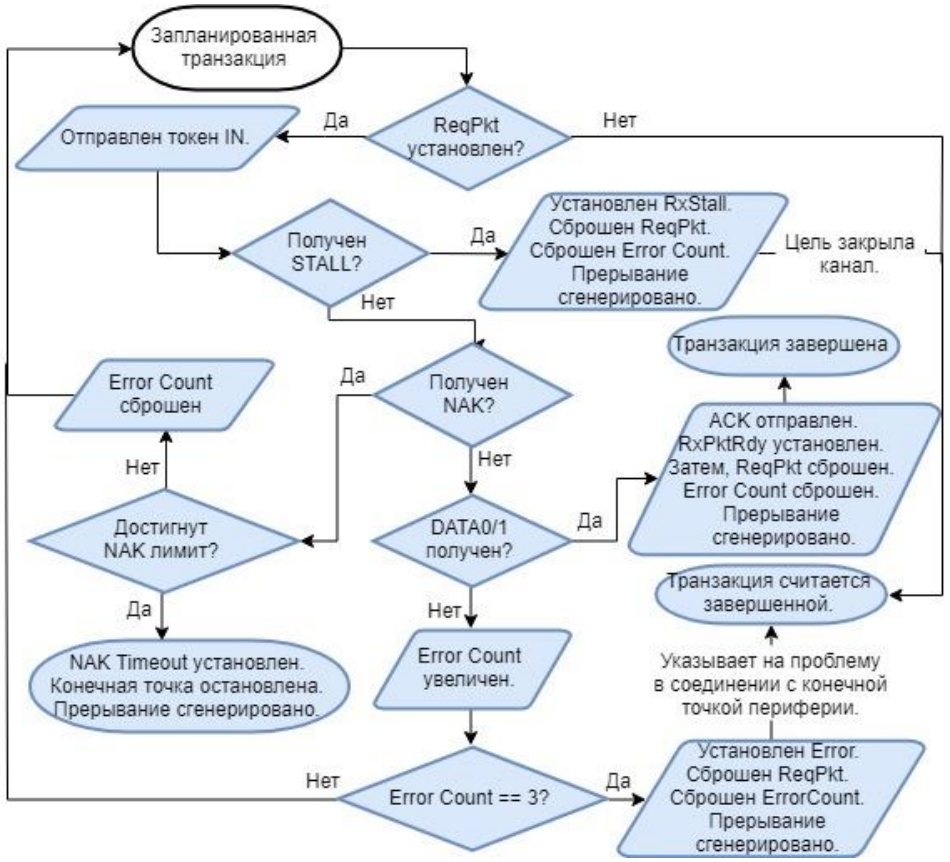


Рисунок 128 – Передача транзакций IN

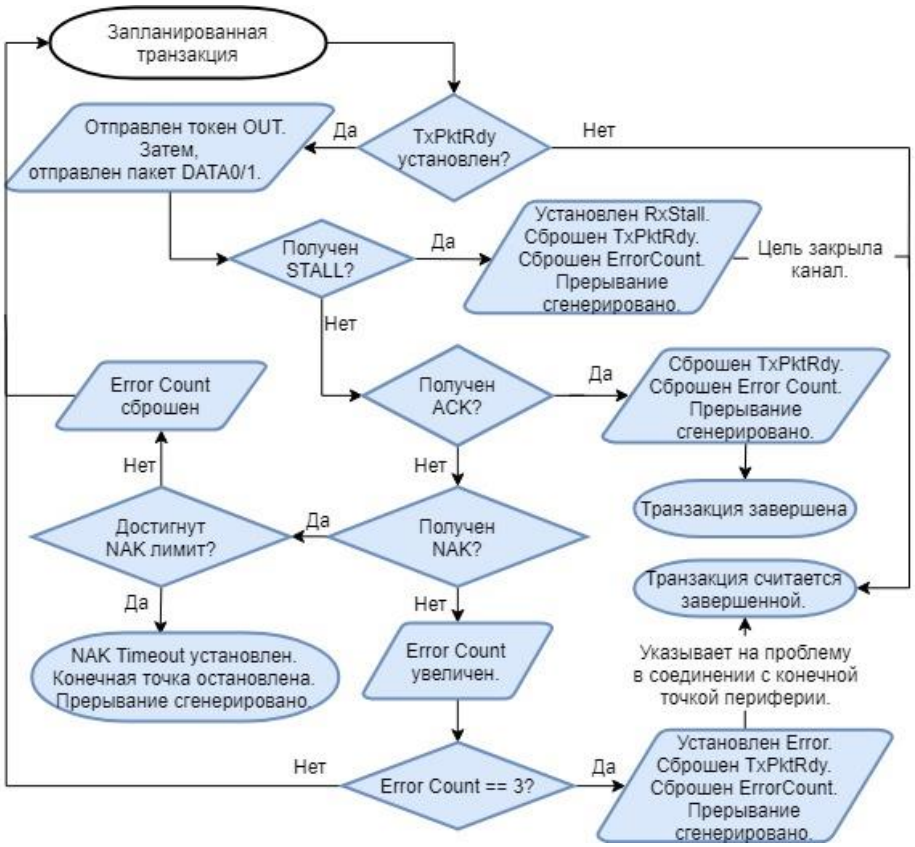


Рисунок 129 – Передача транзакций OUT

22.14 Full-speed/низкопропускные передачи по прерываниям

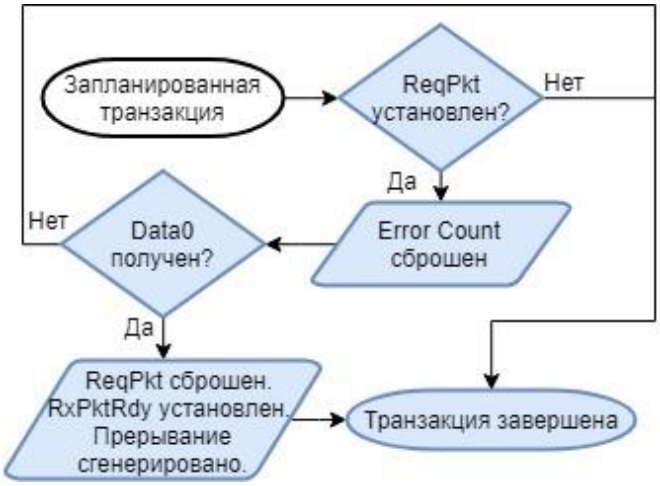


Рисунок 130 – Передача транзакций IN

Рисунок 131 – Передача транзакций OUT

22.15 Высокопропускные передачи (изохронные/по прерываниям)

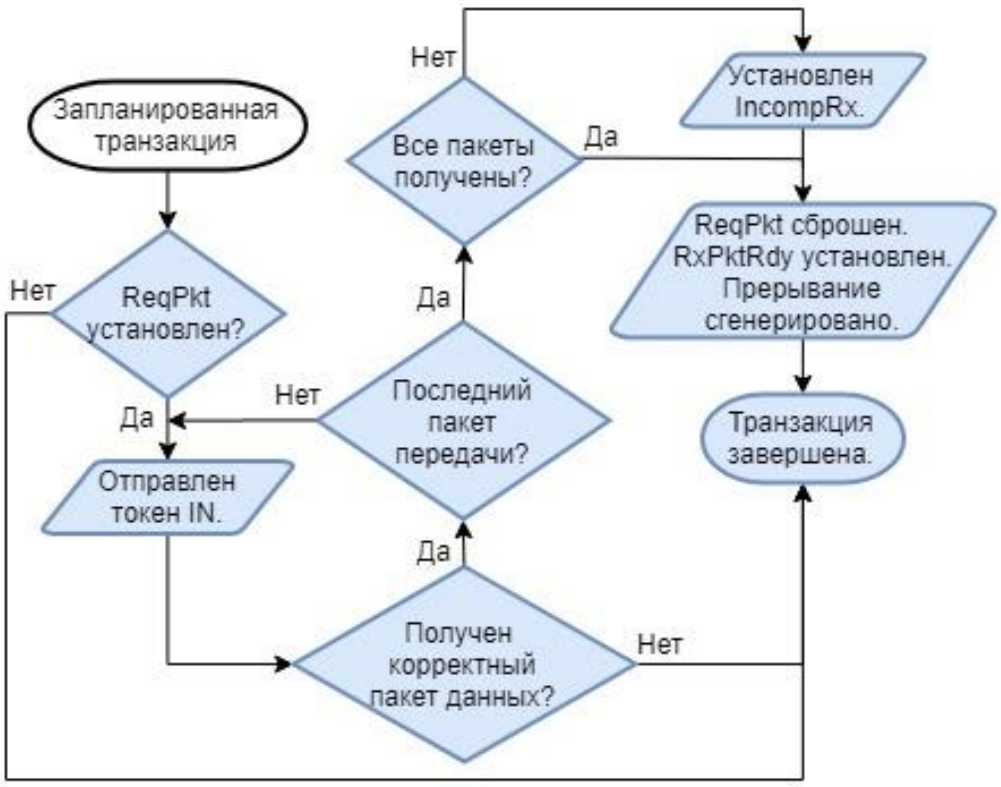


Рисунок 132 – Передача транзакций IN

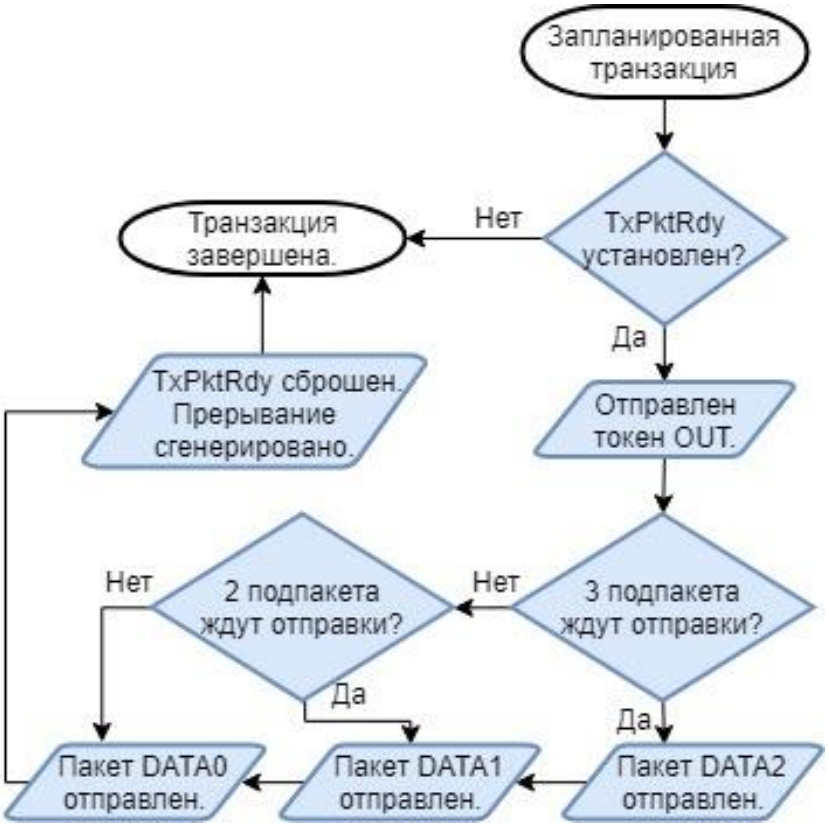


Рисунок 133 – Передача транзакций OUT

22.16 Режим тестирования

USB контроллер поддерживает четыре режима тестирования USB 2.0, определенных для high-speed функций.

Режимы тестирования устанавливаются путем настройки регистра TestMode (адрес 0Fh). Обычно тестовый режим запрашивается хостом, отправкой запроса SET_FEATURE в конечную точку 0. Когда ПО получает запрос, оно должен дождаться завершения передачи конечной точки 0 (прерывание конечной точки 0, указывающее на то, что фаза согласования завершена), а затем настроить регистр TestMode.

TEST SE0_NAK

Чтобы войти в тестовый режим Test_SE0_NAK, программное обеспечение должно установить бит Test_SE0_NAK, записав 8'h01 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он отвечает на любой действительный токен IN с помощью NAK.

TEST_J

Чтобы войти в тестовый режим Test_J, программное обеспечение должно установить бит Test_J, записав 8'h02 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он передает сигнал логического уровня J на шину.

TEST_K

Чтобы войти в тестовый режим Test_K, программное обеспечение должно установить бит Test_K, записав 8'h04 в регистр TestMode. Тогда контроллер перейдет в режим, в котором он передает сигнал логического уровня K на шину.

TEST_PACKET

Чтобы выполнить тест Test_Packet, программное обеспечение должно сначала записать стандартный тестовый пакет (показан ниже) в FIFO конечной точки 0 и установить бит InPktRdy в регистре CSR0 (D1). Затем он должен записать 8'h08 в регистр TestMode, чтобы войти в тестовый режим Test_Packet.

Далее представлен 53-байтовый тестовый пакет, который необходимо загрузить (все байты в шестнадцатеричном виде). Тестовый пакет должен быть загружен только один раз; контроллер продолжит повторную отправку тестового пакета без какого-либо дополнительного вмешательства со стороны программного обеспечения.

00 00 00 00 00 00 00 00
00 AA AA AA AA AA AA AA
AA EE EE EE EE EE EE EE
EE FE FF FF FF FF FF
FF FF FF FF FF FF 7F BF DF
EF F7 FB FD FC 7E BF DF E
F F7 FB FD 7E

Эта последовательность данных определена в спецификации универсальной последовательной шины Revision 2.0, раздел 7.1.20. Контроллер добавит DATA0 PID в начало последовательности и CRC в конец.

23 Интерфейс CAN FD (для микросхем с ревизии 3)

В процессоре реализовано два контроллера интерфейса CAN, каждый из которых является полнофункциональным узлом CAN, отвечающим требованиям к активным и пассивным устройствам CAN 2.0A и 2.0B и поддерживающим передачу данных на скорости не более 1 Мбит/сек.

Назначение внешних выводов микросхемы для CAN0, CAN1 приведено в таблице 202.

Таблица 202 – Назначение внешних выводов для CAN

Обозначение вывода	Назначение вывода для CAN#	Тип	Функциональное назначение
PC[0]	CAN0_TX	O	Интерфейс CAN0. Передатчик
PC[1]	CAN0_RX	I	Интерфейс CAN0. Приемник
PC[28]	CAN1_TX	O	Интерфейс CAN1. Передатчик
PC[29]	CAN1_RX	I	Интерфейс CAN1. Приемник

Для передачи выводов PC, указанных в таблице 202, под управление контроллеров интерфейса CAN, необходимо выбрать альтернативную функцию соответствующего GPIO и разрешить тактирование блока CAN# в регистре CFG8, подробнее см. подраздел 31.1 «Регистр конфигурации периферийных модулей CFG1».

Регистры интерфейса подключены к шине обмена периферийных устройств и имеют базовые адреса:

CAN0: 0x8000C000;

CAN1: 0x8000E000.

23.1 Обзор

Рисунок демонстрирует высокоуровневую архитектуру блока CAN FD и показывает возможность подключения интерфейсов.

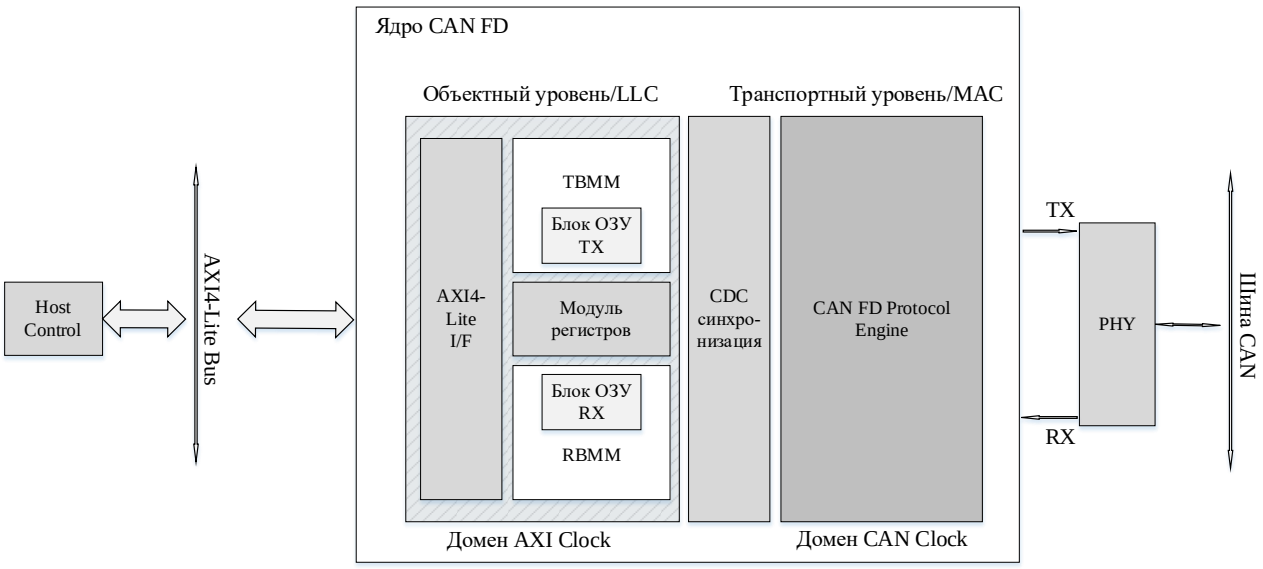


Рисунок 134 – Многоуровневая архитектура ядра CAN FD и возможность подключения

Примечание – блок должен быть подключен к внешнему блоку PHY для общения по шине CAN.

23.2 Описание ядра CAN FD

Основные функции разделены на два независимых уровня, как показано на рисунке 134. Объектный уровень взаимодействует с хостом через интерфейс APB и работает в домене синхросигнала apb_clk. Уровень передачи взаимодействует с внешним PHY и работает в домене синхросигнала can_clk. Обмен информацией между двумя слоями осуществляется через синхронизаторы CDC. Объектный слой CAN FD обеспечивает современный метод передачи и приема для управления буферами сообщений.

Ядро CAN FD соответствует спецификации стандарта ISO-11898-1/2015.

Объектный слой разделен на следующие подмодули:

– модуль Регистров – этот модуль дает доступ на чтение и запись в регистры через внешний хост-интерфейс.

– TX Модуль управления буфером передачи(TBMM) взаимодействует с механизмом протокола CAN FD, чтобы обеспечить следующий буфер для передачи по шине CAN. Он управляет доступом хоста к блоку ОЗУ TX.

– RX Модуль управления буфером приема(RBMM) взаимодействует с механизмом протокола CAN FD для обеспечения хранилища принимаемых сообщений от CAN-шины. Он управляет доступом хоста к блоку ОЗУ RX.

Транспортный слой обеспечивает следующие основные функции:

– инициирование процесса передачи после обнаружения незанятости шины (согласно межкадровому пространству):
  – сериализация кадра;
  – битовая начинка;
  – арбитраж и переход в режим приема в случае потери арбитража;
  – проверка подтверждения(ACK);
  – представление последовательного битового потока на PHY для передачи;
  – расчет CRC, включая неинформационные биты для кадров FD;
  – переключение битрейта.
– прием последовательного битового потока с физического уровня:
  – десериализация и перекомпиляция структуры кадра;
  – удаление неинформационных битов;
  – передача ACK;
  – переключение битрейта.
– функции битовой синхронизации;
– обнаружение ошибок и оповещение;
– распознавание состояния перегрузки и реакция на него.

23.3 Описание регистров

Для ядра CAN FD требуется 32 КБ отображаемого пространства памяти. Разделение этого адресуемого пространства внутри ядра показано в таблице 204.

Таблица 203 – Распределение адресного пространства

Адрес начала	Адрес конца	Секция	Пометки
0x0000	0x00FF	Регистры ядра	Это пространство реализовано с помощью триггеров
0x0100	0x1FFF	Отправка сообщений	Это пространство реализовано с помощью ОЗУ и обеспечивает хранение максимум 32 сообщений для отправки. Для режима буфера RX Sequential (FIFO режим), это также обеспечивает хранение 32 пар ID фильтр-маска. См. таблицу 230
0x2000	0x7FFF	Прием сообщений	Это пространство реализовано с помощью ОЗУ. Для режима буфера RX Sequential (режим FIFO) обеспечивает хранение двух принимающих FIFO на 64 сообщения. См. таблицы 237 и 238. Он обеспечивает хранение TX Event FIFO на 32 сообщения. См. таблицу 234. Для буферного режима почтового ящика он обеспечивает хранение максимум 48 принимаемых сообщений и соответствующего идентификатора маски. См. таблицу 244

Таблица 204 – CAN FD карта регистров ядра

Адрес начала	Наименование	Доступ	Описание	Пометки
0x0000	SRR	Чтение/Запись	Регистр программного сброса	Регистры представленные в обоих режимах работы: RX Mailbox, RX Sequential/FIFO
0x0004	MSR	Чтение/Запись	Регистр выбора режима	
0x0008	BRPR	Чтение/Запись	Регистр делителя скорости передачи этапа арбитража	
0x000C	BTR	Чтение/Запись	Регистр тактовой синхронизации этапа арбитража	
0x0010	ECR	Чтение	Счетчик ошибок	
0x0014	ESR	Чтение/Запись 1 для очистки	Регистр статуса ошибки	
0x0018	SR	Чтение	Регистр состояния	
0x001C	ISR	Чтение	Регистр состояния прерывания	
0x0020	IER	Чтение/Запись	Регистр разрешения прерывания	Регистры представленные в обоих

Адрес начала	Наименование	Доступ	Описание	Пометки
0x0024	ICR	Запись	Регистр очистки прерывания	режимах работы: RX Mailbox, RX Sequential/FIFO
0x0028	TSR	Чтение/Запись	Регистр отметки времени	
0x002C-0x0084	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	
0x0088	DP_BRPR	Чтение/Запись	Регистр делителя скорости передачи данных	
0x008C	DP_BTR	Чтение/Запись	Регистр тактовой синхронизации фазы данных	
0x0090	TRR	Чтение/Запись	Регистр запроса готовности буфера TX	
0x0094	IETRS	Чтение/Запись	Регистр разрешения прерывания запроса готовности буфера TX	
0x0098	TCR	Чтение/Запись	Регистр запроса отмены готовности буфера TX	
0x009C	IETCS	Чтение/Запись	Регистр разрешения прерывания запроса отмены готовности буфера TX	
0x00A0	TxE_FSR	Чтение/Запись	Регистр статуса TX Event FIFO	
0x00A4	TxE_WMR	Чтение/Запись	Регистр отметки TX Event FIFO	
0x00A8-0x00AC	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	
0x00B0	RCS0	Чтение/Запись	Регистр 0 Статуса контроля буфера приема	Регистры, представленные в режиме работы RX Mailbox, остальные зарезервированы
0x00B4	RCS1	Чтение/Запись		
0x00B8	RCS2	Чтение/Запись		
0x00BC	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	
0x00C0	IERBF0	Чтение/Запись	Регистр 0 прерывания при	Регистры, представленные в

Адрес начала	Наименование	Доступ	Описание	Пометки
			заполнении буфера приема	режиме работы RX Mailbox, остальные зарезервированы
0x00C4	IERBF1	Чтение/Запись	Регистр 1 прерывания заполнения буфера приема	
0x00C8-0x00DC	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	
0x00E0	AFR	Чтение/Запись	Регистр (Управления) Приемного фильтра	Регистры, представленные в режиме работы RX Sequential/FIFO, остальные зарезервированы
0x00E4	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	
0x00E8	FSR	Чтение/Запись	Регистр статуса RX FIFO	
0x00EC	WMR	Чтение/Запись	Регистр отметки RX FIFO	
0x00F0-0x00FF	Зарезервировано	-	Запись не имеет никакого смыла, чтение возвращает 0	

23.3.1 Регистр программного сброса

Запись в регистр программного сброса (SRR) переводит ядро в режим настройки. В режиме настройки ядро рецессивно находится на шине, не передает и не принимает сообщения. Во время включения питания биты CEN и SRST равны 0, а бит CONFIG в регистре состояния (SR) равен 1. Регистры конфигурации уровня передачи могут быть изменены только тогда, когда бит CEN в SRR равен 0. Биты регистра выбора режима (кроме SLEEP и SBR) могут быть изменены только тогда, когда бит CEN равен 0. Если бит CEN изменяется во время работы ядра, рекомендуется сбросить ядро, чтобы операция началась заново.

Таблица 205 – Регистр SRR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:2	-	-	0	Зарезервировано
1	CEN	R/W	0	CAN Enable. Это бит разрешения для блока. 1 – блок находится в состоянии Loopback, Sleep, Snoop или Normal, в зависимости от битов LBACK, SLEEP и SNOOP в регистре MSR;

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				0 – блок находится в состоянии настройки. Примечание – Если бит CEN очищается во время работы ядра, рекомендуется сбросить ядро, чтобы операция началась заново
0	SRST	WO	0	Reset. Это бит программного сброса ядра. 1 – сброс ядра. Если 1 записана в этот бит, все конфигурационные регистры, включая SSR, сбрасываются. Чтение данного бита всегда возвращает 0. Примечание – После выполнения программного или аппаратного сброса подождите 16 тактов apb_clk перед инициированием следующей транзакции APB

23.3.2 Регистр выбора режима

Запись в регистр выбора режима (MSR) позволяет ядру переходить в режимы Snoop, Sleep, Loopback, или Normal. В нормальном режиме ядро участвует в обычной коммуникации по шине. Если бит SLEEP установлен в 1, ядро переходит в спящий режим. Если бит LBACK установлено в 1, ядро переходит в режим Loopback. Если для режима SNOOP установлено значение 1, ядро входит в Режим Snoop и не участвует в обмене данными по шине, а только получает сообщения.

Важно: Биты LBACK, SNOOP, SLEEP не могут одновременно быть установлены в 1. В любой момент ядро может находиться в каком-то из режимов Loopback, Sleep или Snoop. Когда все три бита установлены в 0, ядро может перейти в режим Normal при соблюдении других условий.

Таблица 206 – регистр MSR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	-	-	0	Зарезервировано
7	ABR	R/W	0	Auto Bus-Off Recovery Request 1 – запрос на автоматическое восстановление отключенной шины; 0 – запроса нет. Если этот бит установлен, узел выполняет автоматическое восстановление при выключении шины независимо от установленного бита SBR в этом регистре. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0.
6	SBR	R/W	0	Start Bus-Off Recovery Request

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				1 – запрос на начало восстановления отключенной шины; 0 – запроса нет. Узел остается в состоянии Bus-Off до тех пор, пока бит SBR не будет установлен в 1 (это обеспечивает что бит ABR в этом регистре не установлен). Этот бит может быть записан только тогда, когда узел находится в состоянии Bus-Off. Этот бит автоматически сбрасывается после того, как узел завершает восстановление или покидает состояние Bus-Off из-за программного/ аппаратного сброса или при переходе CEN в 0
5	DPEE	R/W	0	Disable Protocol Exception Event Detection/Generation. 1 – отключить обнаружение/генерацию событий исключений протокола, приемник CAN FD, если бит «res» в кадре CAN FD определяется как 1. В этом случае приемник CAN FD генерирует ошибку формы. 0 – обнаружение/генерация PEE включена. Если CAN FD приемник определяет бит res как 1, он переходит в состояние интеграции с шиной (PEE_config) и ожидает состояния Bus Idle (11 последовательных номинальных рецессивных битов). Счетчик ошибок остается неизменным. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
4	DAR	R/W	0	Disable Auto-Retransmission. 1 – отключить автоматическую повторную передачу по шине CAN, чтобы обеспечить однократную передачу; 0 – автоматическая повторная передача включена. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
3	BRSD	R/W	0	CAN FD Bit Rate Switch Disable Override. 1 – заставляет ядро передавать кадры CAN FD только в номинальный битрейт (путем переопределения бита BRS элемента сообщения TX); 0 – заставляет ядро передавать кадры CAN FD в соответствии с битом BRS в TX-сообщении. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
2	SNOOP	R/W	0	SNOOP Mode Select/Request.

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				Это бит запроса режима Snoop. 1 – запрос на переход ядра в режим Snoop; 0 – нет такого запроса. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0. Убедитесь, что режим Snoop запрограммирован только после системного сброса или программного сброса. Чтобы ядро перешло в режим Snoop, биты LBACK и SLEEP в этом регистре должны быть установлены в 0. Особенности режима Snoop: – Ядро передает рецессивные биты на шину CAN; – Ядро принимает сообщения, которые передают другие узлы, но не подтверждает. Сохраняет полученные сообщения в блоке оперативной памяти RX на основе запрограммированного фильтра идентификаторов; – Счетчики ошибок отключены и сброшены в 0. Чтение счетчика ошибок возвращает ноль
1	LBACK	R/W	0	Loopback Mode Select/Request. Это бит запроса режима Loopback. 1 – запрос на переход ядра в режим Loopback; 0 – Нет такого запроса. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0. Для входа в режим Loopback, биты SLEEP и SNOOP в этом регистре должны быть установлен в 0
0	SLEEP	R/W	0	Sleep Mode Select/Request. Это бит запроса спящего режима. 1 – запрос на перевод ядра в спящий режим; 0 – Нет такого запроса. Этот бит сбрасывается, когда ядро выходит из спящего режима. Для перехода в спящий режим, биты LBACK и SNOOP в этом регистре должны быть установлены в 0

23.3.3 Регистр делителя скорости передачи во время арбитража

Тактовая частота can_clk делится на (запрограммированное значение делителя + 1) для генерации квантовых часов, необходимых для отбора и синхронизации.

Таблица 207 – Регистр BRPR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	-	-	0	Зарезервировано
7:0	BRP[7:0]	R/W	0	Arbitration Phase (Nominal) Baud Rate Prescaler. Делитель скорости передачи во время арбитража. Эти биты указывают значение делителя. Фактическое значение на единицу больше, чем значение, записанное в регистр. Эти биты могут быть записаны только тогда, когда бит CEN в SRR равен 0

23.3.4 Регистр тактовой синхронизации этапа арбитража

Таблица 208 – Регистр BTR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:23	-	-	0	Зарезервировано
22:16	SJW[6:0]	R/W	0	Скачок ресинхронизации. Указывает скачок ресинхронизации, как указано в стандарте для номинальной битовой синхронизации. Фактическое значение на единицу больше, чем значение, записанное в регистр. Эти биты могут быть записаны только тогда, когда бит CEN в SRR равен 0
15	-	-	0	Зарезервировано
14:8	TS2[6:0]	R/W	0	Сегмент времени 2. Указывает сегмент фазы 2, как указано в стандарте для номинальной битовой синхронизации. Фактическое значение на единицу больше, чем значение, записанное в регистр. Эти биты могут быть записаны только тогда, когда бит CEN в SRR равен 0
7:0	TS1[7:0]	R/W	0	Сегмент времени 1. Указывает сумму сегмента распространения и сегмента фазы 1 как указано в стандарте для номинальной битовой синхронизации. Фактическое значение на единицу больше, чем значение, записанное в регистр. Эти биты могут быть записаны только тогда, когда бит CEN в SRR равен 0

23.3.5 Регистр счетчика ошибок

ECR является регистром только для чтения. Запись в ECR не имеет никакого эффекта. Значения счетчиков ошибок в регистре отражают значения счетчиков ошибок передачи и приема в ядре. Следующие условия сбрасывают счетчики ошибок передачи и приема:

– Когда в бит SRST регистра SRR записывается 1.

– Когда в бит CEN регистра SRR записывается 0.

– Когда ядро переходит в состояние Bus-Off.

– Во время восстановления Bus-Off до тех пор, пока ядро не перейдет в состояние Error Active (после 128 повторений из 11 последовательных рецессивных битов).

Важно: При Bus-Off восстановлении счетчик ошибок приема увеличивается на 1, когда видна последовательность из 11 последовательных номинальных рецессивных битов.

Примечание – В режиме SNOOP счетчики ошибок выключены и сброшены к 0, чтение счетчиков возвращает 0.

Таблица 209 – Регистр ECR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:16	-	-	0	Зарезервировано
15:8	REC[7:0]	R	0	Счетчик ошибок приема. Показывает значение счетчика ошибок приема
7:0	TEC[7:0]	R	0	Счетчик ошибок отправки. Показывает значение счетчика ошибок отправки

23.3.6 Регистр состояния ошибки

Регистр состояния ошибки (ESR) указывает тип ошибки, возникшей на шине. Если происходит более одной ошибки, в этом регистре устанавливаются все соответствующие биты флага ошибок. Запись в этот регистр не устанавливает никаких битов, но очищает биты, которые установлены.

Таблица 210 – Регистр ESR

Биты	Обозначение	Значение по сбросу	Назначение
31:12	-	0	Зарезервировано
11	F_BERR	0	Битовая ошибка в фазе данных CAN FD: 1 – Указывает, что произошла битовая ошибка при передаче данных; 0 – указывает на отсутствие битовой ошибки при передаче данных после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его

Биты	Обозначение	Значение по сбросу	Назначение
10	F_STER	0	Ошибка стафинга в фазе данных CAN FD: 1 – Указывает, что произошла ошибка стафинга при передаче данных; 0 – указывает на отсутствие ошибки стафинга при передаче данных после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
9	F_FMER	0	Ошибка формата в фазе данных CAN FD: 1 – Указывает, что произошла ошибка формата при передаче данных; 0 – указывает на отсутствие ошибки формата при передаче данных после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
8	F_CRCER	0	Ошибка CRC в фазе данных CAN FD: 1 – Указывает, что произошла ошибка CRC при передаче данных; 0 – указывает на отсутствие ошибки CRC при передаче данных после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
7:5	-	0	Зарезервировано
4	ACKER	0	Ошибка подтверждения(ACK) в фазе данных CAN FD: 1 – Указывает, что произошла ошибка ACK при передаче данных; 0 – указывает на отсутствие ошибки ACK при передаче данных после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
3	BERR	0	Битовая ошибка. Показывает, что принятый бит отличается от отправленного бита во время общения по шине: 1 – Указывает, что произошла битовая ошибка; 0 – указывает на отсутствие битовой ошибки после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
2	STER	0	Ошибка стафинга. Показывает ошибку, если есть нарушение стафинга. 1 – Указывает, что произошла ошибка стафинга 0 – указывает на отсутствие ошибки стафинга после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его
1	FMER	0	Ошибка формата. Ошибка в одном из фиксированных полей кадра сообщения 1 – Указывает, что произошла ошибка формата 0 – указывает на отсутствие ошибки формата после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его

Биты	Обозначение	Значение по сбросу	Назначение
0	CRCER	0	Ошибка CRC. 1 – Указывает, что произошла ошибка CRC 0 – указывает на отсутствие ошибки CRC после последней записи в этот бит. Если этот бит установлен, запись 1 очищает его

Примечания

1 На этапе компенсации задержки передатчика любая ошибка определяется как ошибка F_BERR (передатчиком).

2 Ошибки, связанные с битами в фиксированных полях, определяются как ошибки формата.

3 В случае ошибки CRC и повреждения разделителя CRC устанавливается только бит FMER

23.3.6.1 *Регистр состояния*

Таблица 211 – Регистр SR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:23	-	-	0	Зарезервировано
22:16	TDCV[6:0]	R	0	Transmitter Delay Compensation Value. В этом поле указывается положение вторичной точки выборки (определяется как сумма TDCOFF и измеренной задержки для FDF до падающего фронта бита res от TX к RX в кадре CAN FD) в CAN часы. Это поле предназначено для статуса
15:13	-	-	0	Зарезервировано
12	SNOOP	R	0	Режим SNOOP. 1 – Указывает, что контроллер находится в режиме Snoop, при условии, что бит NORMAL также установлен
11	-	-	0	Зарезервировано
10	BSFR_CONFIG	R	0	Индикатор состояния Bus-Off Recovery: 1 – Указывает, что ядро находится в режиме восстановления Bus-Off (Состояние интеграции шины). Когда этот бит установлен, биты состояния BBSY и NORMAL в этом регистре ничего не значат
9	PEE_CONFIG	R	0	Индикатор состояния PEE: 1 – Указывает, что ядро находится в режиме PEE (состояние интеграции шины). Когда этот бит установлен, биты состояния BBSY и NORMAL в этом регистре ничего не значат

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
8:7	ESTAT[1:0]	R	0	Состояние ошибки Показывает состояние ошибки блока. 00 – режим конфигурации (CONFIG = 1). Состояние ошибки неопределенно; 01 – активное состояние ошибки; 11 – пассивное состояние ошибки; 10 – состояние Bus-Off
6	ERRWRN	R	0	Предупреждение об ошибке. Указывает, что, либо счетчик ошибок передачи, либо счетчик ошибок приема превысил значение 96: 1 – один или несколько счетчиков ошибок имеют значение 96; 0 – ни один из счетчиков ошибок не имеет значения 96
5	BBSY	R	0	Указывает состояние шины CAN: 1 – Указывает, что ядро, либо получает сообщение, либо передает сообщение; 0 – Указывает, что ядро находится, либо в режиме настройки, либо шина простаивает
4	BIDLE	R	0	Шина простаивает. Указывает состояние шины CAN: 1 – Указывает, что обмен данными по шине не осуществляется; 0 – Указывает, что ядро находится либо в режиме настройки, либо шина занята
3	NORMAL	R	0	Нормальный режим. Указывает, что ядро находится в нормальном режиме: 1 – Указывает, что ядро находится в нормальном режиме; 0 – Указывает, что ядро не находится в нормальном режиме
2	SLEEP	R	0	Режим Sleep: 1 – Указывает, что ядро находится в режиме Sleep; 0 – не находится в режиме Sleep
1	LBACK	R	0	Режим Loopback: 1 – Указывает, что ядро находится в режиме Loopback; 0 – не находится в режиме Loopback
0	CONFIG	R	1	Режим Настройки: 1 – Указывает, что ядро находится в режиме Настройки; 0 – не находится в режиме Настройки

23.3.7 Регистр состояния прерывания

Биты состояния прерывания в ISR могут быть очищены путем записи соответствующих бит в регистр очистки прерывания. Для всех битов в ISR, условие установки имеет больший приоритет, чем условие очистки, и бит продолжает оставаться в 1.

Таблица 212 – Регистр ISR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31	TXEWMFLL	R	0	Прерывание TX Event FIFO Watermark Full 1 – указывает, что FIFO Event TX заполнен основываясь на запрограммированную отметку Прерывание продолжает действовать до тех пор, пока уровень заполнения буфера FIFO Event TX больше отметки TX Event FIFO Full.
30	TXEOFLW	R	0	Прерывание TX Event FIFO Overflow. 1 – Указывает, что сообщение было потеряно. Это состояние возникает, когда ядро успешно передало сообщение, для которого запрашивается хранилище событий, но TX Event FIFO заполнен. Этот бит сбрасывается, когда в бит CEN в регистре SRR записывается 0.
29:24	RXBOFLW_BI	R	0	Индекс буфера RX для прерывания переполнения (режим почтового ящика). Дает индекс буфера RX, для которого генерируется событие переполнения. Это поле автоматически сбрасывается, если бит RXBOFLW очищен. В случае, если происходит более одного события переполнения (до того, как Host может очистить RXBOFLW), RXBOFLW_BI показывает индекс переполнения для последнего события. Это поле имеет значение, только если бит прерывания переполнения RXBOFLW установлен. Это поле также очищается при программном/ аппаратном сбросе или если записывается 0 в бит CEN в регистре SRR.
23:18	RXLRM_BI	R	0	Индекс буфера RX для последнего полученного сообщения (режим почтового ящика). Дает индекс буфера RX для последнего полученного сообщения. Это поле имеет значение только в том случае, если в этом регистре установлен бит RXOK. Это поле

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				очищается при аппаратном/программном сбросе или когда записывается 0 в бит CEN в SRR.
17	RXMNF	R	0	RX Match Not Finished. 1 – Указывает, что процесс сопоставления не был завершен до начала шестого бита в поле EOF и кадр был отброшен
16	RXBOFLW/ RXFWMFLL_1	R	0	RX Buffer Overflow Interrupt (режим почтового ящика) 1 –указывает, что сообщение было потеряно из-за переполнения буфера. Индекс буфера фиксируется в поле RXBOFLW_BI. RX FIFO 1 Watermark Full Interrupt (Последовательный/FIFO Режим). 1 – Указывает, что RX FIFO-1 заполнен основываясь на запрограммированной отметке. Примечание – Это прерывание доступно только при включенном RX FIFO-1. Прерывание продолжает действовать до тех пор, пока уровень заполнения RX FIFO-1 больше чем отметка RX FIFO-1 Full.
15	RXRBF/ RXFOFLW_1	R	0	RX Buffer Full Interrupt (режим почтового ящика) 1 – Указывает, что буфер приема получил сообщение и стал заполненым. RX FIFO-1 Overflow Interrupt (Последовательный/FIFO Режим). 1 – Указывает, что сообщение было потеряно. Это состояние возникает, когда получено новое сообщение с идентификатором, совпадающим с Receive FIFO 1, а Receive FIFO 1 заполнен. Примечание – Это прерывание доступно только при включенном RX FIFO-1. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
14	TXCRS	R	0	TX Cancellation Request Served Interrupt. 1 – Указывает, что запрос на отмену был очищен.

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
13	TXRRS	R	0	TX Buffer Ready Request Served Interrupt. 1 – Указывает, что запрос готовности буфера был очищен.
12	RXFWMFLL	R	0	RX FIFO-0 Watermark Full Interrupt (Последовательный/FIFO Режим). 1 – Указывает, что RX FIFO-0 заполнен основываясь на запрограммированной отметки. Прерывание продолжает действовать до тех пор, пока уровень заполнения RX FIFO-0 больше чем отметка RX FIFO-0 Full.
11	WKUP	R	0	Wake-Up Interrupt 1 – указывает, что ядро перешло в нормальный режим из режима Sleep. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
10	SLP	R	0	Sleep Interrupt. 1 – Указывает, что ядро перешло в режим Sleep. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
9	BSOFF	R	0	Bus-Off Interrupt. 1 – Указывает, что ядро перешло в режим Bus-Off. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
8	ERROR	R	0	Error Interrupt. 1 – Указывает, что произошла ошибка во время передачи/приема сообщения. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
7	-	-	0	Зарезервировано
6	RXFOFLW	R	0	RX FIFO-0 Overflow Interrupt (Последовательный/FIFO Режим). 1 – Указывает, что сообщение было потеряно. Это состояние возникает, когда получено новое сообщение с идентификатором, совпадающим с Receive FIFO 0, а Receive FIFO 0 заполнен. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
5	TSCNT_OFLW	R	0	Timestamp Counter Overflow Interrupt 1 – указывает, что регистр отметки времени переполнился (переход от 0xffff к 0x0)

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
4	RXOK	R	0	New Message Received Interrupt. 1 – Указывает, что сообщение было успешно принято и записано в RX FIFO-0 или RX FIFO-1 или в буфер RX Mailbox. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
3	BSFRD	R	0	Bus-Off Recovery Done Interrupt. 1 – Указывает, что ядро восстановилось из состояния Bus-Off. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
2	PEE	R	0	Protocol Exception Event Interrupt. 1 – Указывает, что ядро (приемник CAN FD) зафиксировало событие PEE. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
1	TXOK	R	0	Transmission Successful Interrupt. 1 – Указывает, что сообщение было успешно отправлено. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
0	ARBLST	R	0	Arbitration Lost Interrupt. 1 – Указывает, что арбитраж был потерян во время отправки сообщения. Это поле очищается когда записывается 0 в бит CEN в регистре SRR.
Примечание – В режиме Loopback оба бита RXOK и TXOK установлены. Бит RXOK устанавливается перед установкой бита TXOK				

23.3.8 Регистр разрешения прерываний

Биты регистра разрешения прерывания (IER) используются для включения прерывания, когда соответствующее событие происходит.

Таблица 213 – Регистр IER

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31	ETXEWMFLL	R/W	0	1 – Разрешает генерацию прерывания, если соответствующий бит в ISR поднят; 0 – Запрещает генерацию прерывания, если соответствующий бит в ISR поднят
30	ETXEOFLW	R/W	0	
17	ERXMNF	R/W	0	
16	ERXBOFLW/ ERXFWMFLL_1	R/W	0	

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
15	ERXRBF/ ERXFOFLW_1	R/W	0	
14	ETXCRS	R/W	0	
13	ETXRRS	R/W	0	
12	ERXFWMFLL	R/W	0	
11	EWKUP	R/W	0	
10	ESLP	R/W	0	
9	EBSOFF	R/W	0	
8	EERROR	R/W	0	
7	-	-	0	Зарезервировано
6	ERXFOFLW	R/W	0	1 – Разрешает генерацию прерывания, если соответствующий бит в ISR поднят; 0 – Запрещает генерацию прерывания, если соответствующий бит в ISR поднят
5	ETSCNT_OFLW	R/W	0	
4	ERXOK	R/W	0	
3	EBSFRD	R/W	0	
2	EPEE	R/W	0	
1	ETXOK	R/W	0	
0	EARBLOST	R/W	0	

23.3.9 Регистр очистки прерываний

Биты регистра очистки прерывания (ICR) используются для очистки бит состояния прерывания (ISR регистр)

Таблица 214 – Регистр ICR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31	CTXEWMFLL	W	0	1 – Очищает соответствующий бит состояния прерывания ISR. Запись 1 в данные биты очищают соответствующие биты в регистре ISR. Чтение всегда возвращает 0
30	CTXEOFLW	W	0	
17	CRXMNF	W	0	
16	CRXBOFLW/ CRXFWMFLL_1	W	0	
15	CRXRBF/ CRXFOFLW_1	W	0	
14	CTXCRS	W	0	
13	CTXRRS	W	0	
12	CRXFWMFLL	W	0	
11	CWKUP	W	0	
10	CSLP	W	0	
9	CBSOFF	W	0	
8	CERROR	W	0	
7	-	W	0	Зарезервировано
6	CRXFOFLW	W	0	1 – Очищает соответствующий бит состояния прерывания ISR
5	CTSCNT_OFLW	W	0	
4	CRXOK	W	0	
3	CBSFRD	W	0	

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
2	CPEE	W	0	Запись 1 в данные биты очищают соответствующие биты в регистре ISR. Чтение всегда возвращает 0.
1	CTXOK	W	0	
0	CARBLOST	w	0	

23.3.10 Регистр отметки времени

16-битный счетчик, работающий в автономном режиме, увеличивается один раз каждые 16 тактов CAN. Отметка времени захватывается после бита SOF; то есть, когда поле ID начинается на шине CAN. Это вносится в поле DLC элемента сообщения, когда кадр успешно получен или передан. Счетчик временных меток может быть сброшен программно. Нет регистрового бита для индикации переполнения счетчика.

Таблица 215 – Регистр TSR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:16	TIMESTAMP_CNT[15:0]	R	0	Текущее значение счетчика отметки времени. Это поле очищается когда записывается 0 в бит CEN в регистре SRR
15:1	-	-	0	Зарезервировано
0	CTS	W	0	Clear Timestamp Counter. Внутренний счетчик сбрасывается до 0, когда CTS = 1. Только один раз нужно записать 1 в данный бит, чтобы очистить счетчик. Чтение всегда возвращает 0

23.3.11 Регистр делителя скорости передачи данных Data phase.

Таблица 216 – Регистр DP_BRPR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:17	-	-	0	Зарезервировано
16	TDC	R/W	0	Transmitter Delay Compensation (TDC) Enable: 1 – Включает функцию TDC, как указано в стандарте CAN FD; 0 – TDC отключен. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
15:14	-	-	0	Зарезервировано
13:8	TDCOFF[5:0]	R/W	0	Transmitter Delay Compensation Offset Это смещение указывается в тактах CAN и добавляется к измеренному значению задержки

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				передатчика для размещения вторичной точки выборки (SSP) в соответствующую позицию (например, установите половину времени передачи данных в тактах CAN, чтобы поместить SSP в середину бита данных). Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
7:0	DP_BRP[7:0]	R/W	0	Data Phase Baud Rate Prescaler. Эти биты указывают значение делителя для тактовой синхронизации данных, как указано в стандарте CAN FD. Фактическое значение на единицу больше, чем значение, записанное в регистр. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0

Важно: На сумму измеренной задержки цикла и TDC Offset накладывается следующее граничное условие:

Измеренная задержка цикла + TDCOFF < 3 времени передачи бита в фазе данных

Убедитесь, что граничное условие соблюдается при программировании смещения и битрейта фазы данных. В случае, если эта сумма превышает 127 тактов CAN, максимальное значение 127 тактов CAN используются ядром для компенсации задержки передатчика.

Примечание – Если задержка цикла < 1 времени передачи бита фазы данных, то метод TDC/SSP не требуется.

23.3.12 Регистр тактовой синхронизации Data phase.

Таблица 217 – Регистр DP_BTR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:20	-	-	0	Зарезервировано
19:16	DP_SJW[3:0]	R/W	0	Data Phase Synchronization Jump Width. Указывает скачок ресинхронизации, как указано в стандарте CAN FD для тактовой синхронизации данных. Фактическое значение на единицу больше, чем значение, записанное в регистр. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
15:12	-	-	0	Зарезервировано

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
11:8	DP_TS2[3:0]	R/W	0	Data Phase Time Segment 2 Указывает фазовый сегмент 2, как указано в стандарте CAN FD для тактовой синхронизация данных. Фактическое значение на единицу больше, чем значение, записанное в регистр. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0
7:5	-	-	0	Зарезервировано
4:0	DP_TS1[4:0]	R/W	0	Data Phase Time Segment 1 Указывает сумму сегмента распространения и сегмента фазы 1 как указано в стандарте CAN FD для тактовой синхронизации данных. Фактическое значение на единицу больше, чем значение, записанное в регистр. Этот бит может быть записан только тогда, когда бит CEN в SRR равен 0

23.3.13 Регистр запроса готовности буфера TX

Таблица 218 – Регистр TRR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	RR31/RR8	R/W, Host устанавливает 1, а ядро очищает	0	Примечание – Эти биты зависят от количества буферов TX
7	RR7		0	TX Buffer_0 Ready Request Этот управляющий бит соответствует сообщению TB0 в блоке TX RAM. Host записывает 1, чтобы указать, что буфер готов к передаче. Ядро очищает этот бит, когда: – Передача буфера по шине CAN завершена. – Если ядро находится в режиме DAR, то после одной попытки передачи по шине CAN [либо успешно, либо безуспешно (то есть арбитраж проигран или ошибка)] – Если сообщение отменено из-за запроса на отмену – Любая комбинация вышеперечисленных трех. Записи хоста в этот бит игнорируются, если этот бит равен 1. Примечание – Этот регистр остается в состоянии сброса, когда включен режим SNOOP
6	RR6			
5	RR5			
4	RR4			
3	RR3			
2	RR2			
1	RR1			
0	RR0			

Примечания

1 Хост может устанавливать запросы на передачу для нескольких буферов за одну запись в этот регистр.

2 Запись любого значения в этот регистр запускает планировщик буфера, чтобы повторить раунд планирования, с целью найти выигрышный буфер. (исключения: когда уровень передачи находится в трехбитном межкадровом промежутке без заблокированного буфера или если предыдущий раунд планирования уже работает. В таких случаях вызов планировщика буфера откладывается до завершения события)

Важно: Ненужные записи в этот регистр могут снизить пропускную способность ядра на шине CAN. Обеспечьте, что регистр записывается только тогда, когда это требуется.

23.3.14 Регистр разрешения прерывания запроса готовности буфера TX

Таблица 219 – Регистр IETRS

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	ERRS31/ERRS8	R/W	0	Примечание – Эти биты зависят от количества буферов TX
7	ERRS7	R/W	0	TX Buffer_0 Ready Req Served/Clear Interrupt Enable: 1 – разрешает установку бита TXRRS в регистре ISR, когда бит RR0 в регистре TRR очищается; 0 – бит TXRRS в регистре ISR не устанавливается, если бит RR0 в регистре TRR очищается
6	ERRS6			
5	ERRS5			
4	ERRS4			
3	ERRS3			
2	ERRS2			
1	ERRS1			
0	ERRS0			

23.3.15 Регистр запроса отмены готовности буфера TX

Таблица 220 – Регистр TCR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	CR31/CR8	R/W, Host устанавливает 1, а ядро очищает	0	Примечание – Эти биты зависят от количества буферов TX
7	CR7		0	TX Buffer_0 Cancel Request Это бит запроса отмены соответствует биту RR0 в регистре TRR. Host записывает 1, чтобы указать запрос на отмену запроса готовности соответствующего буфера (то есть бит RR0 в регистре TRR). Ядро очищает этот бит, когда запрос на отмену выполнен.
6	CR6			
5	CR5			
4	CR4			
3	CR3			
2	CR2			
1	CR1			
0	CR0			

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
				Записи Host в этот бит игнорируются, если CR0 равен 1 или бит RR0 в регистре TRR равен 1 Если буфер уже заблокирован для отправки уровнем передачи то отмена выполняется в конце цикла передачи независимо от того, успешно ли передан кадр или нет. То есть, если сообщение не удалось отправить из-за потери арбитража или какой-либо ошибки, то сообщение отменяется (без попытки повторной передачи) и запрос на отмену очищается. Наряду с сбросом бита RR0. Если сообщение успешно передано, то бит RR0 сбрасывается и запрос на отмену очищается в любом случае. Примечание – Если выполняется раунд планирования внутреннего буфера, то рассмотрение запроса на отмену откладывается до его завершения

Примечание – Хост может установить запросы на отмену для нескольких буферов за одну запись в этот регистр.

Важно: Ненужные отмены буфера могут снизить пропускную способность ядра на шине CAN. Обеспечьте, что отмены буфера запрашиваются только тогда, когда это требуется.

23.3.16 Регистр разрешения прерывания запроса отмены буфера TX

Таблица 221 – Регистр IETCS

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:8	ECRS31/ECRS8	R/W	0	Примечание – Эти биты зависят от количества буферов TX.
7	ECRS7	R/W	0	TX Buffer_0 Transmission Served/Cleared Interrupt Enable: 1 – разрешает установку бита TXCRS в регистре ISR, когда бит CR0 в регистре TCR очищается; 0 – бит TXCRS в регистре ISR не устанавливается, если бит CR0 в регистре TCR очищается.
6	ECRS6			
5	ECRS5			
4	ECRS4			
3	ECRS3			
2	ECRS2			
1	ECRS1			
0	ECRS0			

23.3.17 Регистр статуса TX Event FIFO

Таблица 222 – Регистр TxE_FSR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:14	-	-	0	Зарезервировано
13:8	TXE_FL[5:0]	R	0	Fill Level(0-32). Количество сохраненных сообщений в буфере TX Event FIFO начиная с индекса RI, заданного в этом регистре. Например, если FL = 0x5 и RI = 0x3, то TX Event FIFO имеет пять сообщений, начиная с индекса чтения 3 (Начальный адрес 0x2018). FL сохраняется, если бит CEN сброшен. FL сбрасывается, если программный или аппаратный сброс
7	TXE_IRI	W	0	Increment Read Index by 1. Каждый хост записывает установку этого бита в 1, ядро увеличивает Индекс чтения (поле RI) на 1 и обновляет Fill Level (т. е. уменьшает на 1). Если FILL Level равен 0, установка этого бита не имеет никакого эффекта. FILL Level может оставаться неизменным, когда IRI установлен и ядро только завершает успешную передачу и увеличивает внутренний индекс записи. Этот бит всегда читается как 0
6:5	-	-	0	Зарезервировано
4:0	TXE_RI	R	0	Read Index (0 to 31) Каждый раз, когда устанавливается бит IRI, ядро увеличивает индекс чтения на + 1 (при условии, что уровень FILL не равен 0) и сохраняет его для хоста для доступа к следующему доступному сообщению. RI = 0x0 -> чтение следующего сообщения начинается с адреса = 0x2000 RI = 0x1 -> чтение следующего сообщения начинается с адреса = 0x2008 RI сохраняется, если бит CEN сброшен. RI сбрасывается, если программный или аппаратный сброс

23.3.18 Регистр отметки TX Event FIFO

Таблица 223 – Регистр TxE_WMR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:5	-	-	0	Зарезервировано
4:0	TXE_FWM	R/W	0xf	TX Event FIFO Full Watermark TX Event FIFO генерирует прерывание FULL на основе значение, запрограммированного в этом поле. Установите его в диапазоне (1-31). Прерывание TX FIFO Full Watermark в ISR регистре продолжается до тех пор, пока TX Event FIFO Fill Level превышает TX Event FIFO Full Watermark. Это поле может быть записано только тогда, когда бит CEN регистра SRR равен 0.

23.3.19 Регистр 0 Статуса контроля буфера приема

Таблица 224 – Регистр RCS0

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31	CSB15	Записать 1 для очистки	0	Core Status bit for RX Buffer*: 1 – указывает на то, что соответствующий буфер заполнен, то есть ядро получило сообщение в этот буфер; 0 – буфер не заполнен. Host очищает этот бит, записывая 1
:	:			
18	CSB2			
17	CSB1			
16	CSB0			
15	HCB15	R/W	0	Host Control bit for RX Buffer*: 1 – указывает на то, что соответствующий буфер активен, то есть поле ID буфера RB* и соответствующий регистр маски запрограммированы Хостом и этот буфер может получить сообщение; 0 – буфер неактивен. Хост может изменить любой бит в любой момент
:	:			
2	HCB2			
1	HCB1			
0	HCB0			

CSBx:HCBx = «00» -> Буфер неактивен (не учитывается в процессе сопоставления ID).

CSBx:HCBx = «01» -> Буфер активен (может принять сообщение, если идентификатор приема совпадает с идентификатором буфера).

CSBx:HCBx = «11» -> Буфер заполнен (получено сообщение)

CSBx:HCBx = «10» -> Буфер недействителен. Это состояние может произойти, когда ядро обновляет бит CS0, чтобы указать, что буфер заполнен, и в то же время Хост пытается сделать буфер неактивным.

Примечания

1 При изменении состояния буфера с активного в неактивное, хост должен проверить обновление путем чтения (чтобы проверить, не изменился ли статус буфера на Invalid из-за того, что ядро одновременно указало на то, что буфер заполнен).

2 Полный (11) -> Активный (01) или Неактивный (00) может учитываться или может не учитываться в текущем процессе сопоставления, если он запущен.

3 Неактивный (00) -> Активный (01) может учитываться или может не учитываться в текущем процессе сопоставления, если он запущен.

4 Недопустимые буферы не участвуют в процессе сопоставления ID.

23.3.20 Регистр 1 Статуса контроля буфера приема

Описание аналогично таблице 224.

Примечание – Это пространство зарезервировано для режима RX Sequential/FIFO или, когда количество буферов RX mailbox – 16/32. Когда зарезервировано, запись не действует, а чтение возвращает 0.

23.3.21 Регистр 2 Статуса контроля буфера приема

Описание аналогично таблице 224.

Примечание – Это пространство зарезервировано для режима RX Sequential/FIFO или, когда количество буферов RX mailbox – 16. Когда зарезервировано, запись не действует, а чтение возвращает 0.

23.3.22 Регистр 0 разрешения прерывания при заполнении буфера приема

Таблица 225 – Регистр IERBF0

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:16	ERBF31/ ERBF16	R/W	0	Примечание – Эти биты зависят от количества буферов TX
15	ERBF15	R/W	0	RX Buffer_15/0 Full Interrupt Enable: 1 – Разрешает установку бита RXBFL в ISR, когда RX Buffer* становится Полным; 0 – бит RXBFL в ISR не устанавливается, если RX Buffer* становится Полным.
:	:			
2	ERBF2			
1	ERBF1			
0	ERBF0			
Примечание – Это пространство зарезервировано для режима RX Sequential/FIFO. Когда зарезервировано, запись не действует, а чтение возвращает 0				

23.3.23 Регистр 1 разрешения прерывания при заполнении буфера приема

Таблица 226 – Регистр IERBF1

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:16	-	-	0	Зарезервировано
15:1	ERBF47/ ERBF33	R/W	0	RX Buffer_47/32 Full Interrupt Enable:

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
0	ERBF32			1 – Разрешает установку бита RXBFL в ISR, когда RX Buffer* становится Полным; 0 – бит RXBFL в ISR не устанавливается, если RX Buffer* становится Полным.
Примечание – Это пространство зарезервировано для режима RX Sequential/FIFO или, когда количество буферов RX mailbox – 16/32. Когда зарезервировано, запись не действует, а чтение возвращает 0				

23.3.24 Регистр управления приемного фильтра

В режиме буфера RX Sequential/FIFO регистр (управления) приемным фильтром (AFR) определяет какие фильтры приема использовать. Каждый регистр идентификатора фильтра приема (AFIR) и пара регистров маски фильтра (AFMR) связана с битом UAF.

– Когда бит UAF равен 1, для приема используется соответствующая пара приемных фильтров фильтрации. Когда бит UAF равен 0, соответствующая пара приемных фильтров, не используется для фильтрации приема.

– Чтобы изменить пару приемных фильтров в нормальном режиме, соответствующий бит UAF в этом регистр должен быть установлен в 0.

– После изменения приемного фильтра соответствующий бит UAF должен быть установлен в 1.

– Если во все биты UAF установлены 0, полученные сообщения не сохраняются в буферах RX Sequential/FIFO.

– Если биты UAF изменяются с 1 на 0 во время приема сообщения, то сообщение может быть сохранено, а может и не быть сохранено.

Таблица 227 – Регистр AFR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31 : 0	UAF31/ UAF0	R/W	0	Use Acceptance Filter Mask Pair 31/0 Включает использование Маски приемного фильтра*: 1 – указывает что используется для приемной фильтрации пара, состоящая Acceptance Filter Mask register* (AFMR* или M*) и Acceptance Filter ID register* (AFID* или F*); 0 – указывает, что пара AFMR* и AFID* не используется для приемной фильтрации
Примечание – Это пространство зарезервировано для режима RX Mailbox buffer. Когда зарезервировано, запись не действует, а чтение возвращает 0				

23.3.25 Регистр статуса RX FIFO

Таблица 228 – Регистр FSR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31	-	-	0	Зарезервировано
30:24	FL_1[6:0]	R	0	RX FIFO-1 Fill Level (0-64). Примечание – Это поле зарезервировано, если RX FIFO-1 не включен. Количество сохраненных сообщений в RX FIFO-1, начиная с индекс чтения (RI), задано в этом регистре. Например, если FL = 0x5 и RI = 0x2, то RX FIFO-1 имеет пять сообщения, начиная с индекса чтения 2 (начальный адрес 0x4196). FL сохраняется, если бит CEN сброшен. FL сбрасывается, если программный или аппаратный сброс
23	IRI_1	W	0	RX FIFO-1 Increment Read Index by 1. Примечание – Это поле зарезервировано, если RX FIFO-1 не включен. Каждый хост записывает установку этого бита в 1, ядро увеличивает Индекс чтения (поле RI) на 1 и обновляет Fill Level (т. е. уменьшает на 1). Если FILL Level равен 0, установка этого бита не имеет никакого эффекта. FILL Level может оставаться неизменным, когда IRI установлен и ядро только завершает успешную передачу и увеличивает внутренний индекс записи. Этот бит всегда читается как 0
22	-	-	0	Зарезервировано
21:16	RI_1[5:0]	R	0	RX FIFO-1 Read Index (0 to 63). Примечание – Это поле зарезервировано, если RX FIFO-1 не включен. Каждый раз, когда устанавливается бит IRI, ядро увеличивает индекс чтения на + 1 (при условии, что уровень FILL не равен 0) и сохраняет его для хоста для доступа к следующему доступному сообщению. RI – 0x0 –> чтение следующего сообщения начинается с адреса = 0x4100 RI – 0x1 –> чтение следующего сообщения начинается с адреса = 0x4148 RI сохраняется, если бит CEN сброшен. RI сбрасывается, если программный или аппаратный сброс
15	-	-	0	Зарезервировано

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
14:8	FL[6:0]	R	0	RX FIFO-0 Fill Level (0-64). Количество сохраненных сообщений в RX FIFO-0, начиная с индекс чтения (RI), задано в этом регистре. Например, если FL = 0x5 и RI = 0x2, то RX FIFO-0 имеет пять сообщения, начиная с индекса чтения 2 (начальный адрес 0x2196). FL сохраняется, если бит CEN сброшен. FL сбрасывается, если программный или аппаратный сброс
7	IRI	W	0	RX FIFO-0 Increment Read Index by 1. Каждый хост записывает установку этого бита в 1, ядро увеличивает Индекс чтения (поле RI) на 1 и обновляет Fill Level (т. е. уменьшает на 1). Если FILL Level равен 0, установка этого бита не имеет никакого эффекта. FILL Level может оставаться неизменным, когда IRI установлен и ядро только завершает успешную передачу и увеличивает внутренний индекс записи. Этот бит всегда читается как 0
6	-	-	0	Зарезервировано
5:0	RI	R	0	RX FIFO-0 Read Index (0 to 63). Каждый раз, когда устанавливается бит IRI, ядро увеличивает индекс чтения на + 1 (при условии, что уровень FILL не равен 0) и сохраняет его для хоста для доступа к следующему доступному сообщению. RI = 0x0 –> чтение следующего сообщения начинается с адреса = 0x2100 RI = 0x1 –> чтение следующего сообщения начинается с адреса = 0x2148 RI сохраняется, если бит CEN сброшен. RI сбрасывается, если программный или аппаратный сброс
Примечание – Это пространство зарезервировано для режима RX Mailbox buffer. Когда зарезервировано, запись не действует, а чтение возвращает 0				

23.3.26 Регистр отметки RX FIFO

Таблица 229 – Регистр WMR

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:21	-	-	0	Зарезервировано
20:16	RXFP	R/W	0xf	Receive Filter Partition. Полученные сообщения, соответствующие парам Фильтр-Маска от 0 до RXFP, хранятся в RX FIFO-0. Полученные сообщения, соответствующие парам Фильтр-Маска RXFP+1 и больше, сохраняются в RX FIFO-1. Примечание – Это поле доступно, только если включен RX FIFO-1. Это поле можно изменить, если CEN = 0
13:8	RXFWM_1	R/W	0xf	RX FIFO-1 Full Watermark. Примечание – Это поле доступно, только если включен RX FIFO-1. RX FIFO-1 генерирует прерывание FULL на основе значение, запрограммированного в этом поле. Установите его в диапазоне (1-63). Прерывание RX FIFO-1 Full Watermark в ISR регистре продолжается до тех пор, пока RX FIFO-1 Fill Level превышает RX FIFO-1 Full Watermark. Это поле может быть записано только тогда, когда бит CEN регистра SRR равен 0
7:6	-	-	0	Зарезервировано
5:0	RXFWM	R/W	0xf	RX FIFO-0 Full Watermark. RX FIFO-0 генерирует прерывание FULL на основе значение, запрограммированного в этом поле. Установите его в диапазоне (1-63). Прерывание RX FIFO-0 Full Watermark в ISR регистре продолжается до тех пор, пока RX FIFO-0 Fill Level превышает RX FIFO-0 Full Watermark. Это поле может быть записано только тогда, когда бит CEN регистра SRR равен 0
Примечание – Это пространство зарезервировано для режима RX Mailbox buffer. Когда зарезервировано, запись не действует, а чтение возвращает 0				

23.4 Описание регистров пространства сообщений CAN FD TX

Таблица 230 – CAN FD пространство сообщений TX

Адрес начала	Наименование	Доступ	Описание	Пометки
0x100	TB0-ID	Чтение/Запись	Регистр TB ID	Только необходимые место-положения DW нужно записать согласно полям FDF и DLC для данного сообщения
0x104	TB0-DLC	Чтение/Запись	Регистр TB DLC	
0x108	TB0-DW0	Чтение/Запись	Регистр TB DW0	
0x10C–0x144	TB0-DW1 – TB0-DW15	Чтение/Запись		
0x0148-0x033C	TB1-TB7	Чтение/Запись		
0x0340-0x057C	TB8-TB15	Чтение/Запись		Зарезервировано, если число буферов TX равно 8. В данном случае запись запрещена, чтение возвращает 0
0x0580-0x0894	TB16-TB26	Чтение/Запись		Зарезервировано, если число буферов TX равно 8 ил 16. В данном случае запись запрещена, чтение возвращает 0
0x0898-0x09FC	TB27-TB31	Чтение/Запись		Зарезервировано, если число буферов TX равно 8 или 16. В данном случае запись запрещена, чтение возвращает 0
0x0A00	M0/AFMR0	Чтение/Запись	Регистр маски фильтра приема	Пара 0 идентификатор-маска фильтра
0x0A04	F0/AFIR0	Чтение/Запись	Регистр ID фильтра приема	
0x0A08	M1/AFMR1	Чтение/Запись	Регистр маски фильтра приема	Пара 1 идентификатор-маска фильтра
0x0A0C	F1/AFIR1	Чтение/Запись		
0x0A10-0x0AFC	M2, F2-M31, F31	Чтение/Запись		Пары 2-31 идентификатор-маска фильтра

Примечания

1 Чтение из неинициализированного участка памяти может вернуть X или неверные данные. Программный или аппаратный сбросы не очищают блоки оперативной памяти;

2 Каждый TB связан с битом в регистре TRR. Ядро получает доступ к элементам TB внутри ОЗУ блока TX только в том случае, если соответствующий бит TRR установлен. После установки бита TRR хост не должен обращаться к элементам сообщения TX до тех пор, пока ядро не очистит TRR, чтобы избежать проблем конфликтов памяти

23.4.1 Регистр TB*-ID (Address Offset + 0x0100, 0x0148 …)

Таблица 231 – Регистр TB ID

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
31:21	ID[28:18]	Контроль	-	Standard Message ID. Часть идентификатора для стандартного кадра. Эти биты указывают идентификатор стандартного кадра. Это поле действует как для кадров CAN, так и для кадров CAN FD Standard и Extended
20	SRR/RTR/RRS	Контроль	-	Substitute Remote Transmission Request. Для расширенных кадров CAN и CAN FD этот бит передается в позиции SRR соответствующего кадра и должен быть установлен как 1. Для стандартного кадра CAN FD этот бит передается в позиции RRS и должен быть установлен как 0. Этот бит различает стандартные кадры данных CAN и стандартные удаленные кадры CAN, как показано ниже: 1 – Указывает, что кадр сообщения является стандартным удаленным кадром CAN. 0 – Указывает, что кадр сообщения является стандартным кадром данных CAN. Примечание – Удаленных кадров CAN FD нет
19	IDE	Контроль	-	Identifier Extension. Этот бит различает кадры, использующие стандартный идентификатор и те, которые использует расширенный идентификатор. Действительно как для кадров CAN, так и для CAN FD Standard и Extended 1 – указывает на использование расширенного идентификатора сообщения. 0 – указывает на использование стандартного идентификатора сообщения
18:1	ID[17:0]	Контроль	-	Extended Message ID. В этом поле указывается расширенный идентификатор. Действительно только для расширенных кадров CAN и CAN FD. Для стандартных кадров CAN и CAN FD запись в это поле должна быть 0

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
0	RTR/RRS	Контроль	-	Remote Transmission Request. Этот бит различает расширенные кадры данных CAN и расширенные удаленные кадры CAN. 1 – Указывает, что кадр сообщения является CAN Remote. 0 – Указывает, что кадр сообщения является кадром данных CAN. Для расширенного кадра CAN FD этот бит передается в позиции RRS и должен быть установлен как 0. Для стандартных кадров CAN и CAN FD запись в этот бит должна быть 0. Примечание – Удаленных кадров CAN FD нет

23.4.2 Регистр TB*-DLC (Address Offset + 0x0104, 0x014C …)

Таблица 232 – Регистр TB DLC

Биты	Обозначение	Контроль /статус	Значение по сбросу	Назначение
31:28	DLC[3:0]	Контроль	-	Data Length Code Это код длины данных поля управления кадра CAN или CAN FD
27	EDL/FDF	Контроль	-	Extended Data Length/FD Frame Format Этот бит различает формат кадра CAN и CAN FD. 1 – кадр формата CAN FD. 0 – кадр формата CAN.
26	BRS	Контроль	-	Bit Rate Switch Бит BRS определяет, переключается ли битрейт внутри кадра формата CAN FD или нет (при условии, что бит BRSD не установлен в регистре MCP). 1 – битрей переключается со стандартного битрейта Фазы арбитража до предварительно настроенного альтернативного битрейра фазы данных внутри кадра CAN FD. 0 – Битрейт не переключается внутри кадра CAN FD. Примечание – BRS не существует в кадрах формата CAN и должен быть установлен 0
25	Reserved	-	-	Зарезервировано
24	EFC	Контроль	-	Event FIFO Control. 0 – не хранить события TX Event. 1 – хранить события TX Event

Биты	Обозначение	Контроль /статус	Значение по сбросу	Назначение
23:16	MM	Контроль	-	Message Marker. Записывается ЦП во время настройки буфера TX. Скопировано в Элемент Tx Event FIFO для идентификации статуса сообщения TX
15:0	Reserved	-	-	Зарезервировано, запись в это поле должно быть 0

23.4.3 Регистр TB*-DW0 (Address Offset + 0x0108, …, 0x0150 …)

Таблица 233 – Регистр TB DW0

Биты	Обозначение	Значение по сбросу	Назначение
31:24	Data bytes0 [7:0]	-	Байт данных 0 Байт данных должен быть передан с кадром CAN или CAN FD на основе поля управления DLC
23:16	Data bytes1 [7:0]	-	Байт данных 1 Байт данных должен быть передан с кадром CAN или CAN FD на основе поля управления DLC
15:8	Data bytes2 [7:0]	-	Байт данных 2 Байт данных должен быть передан с кадром CAN или CAN FD на основе поля управления DLC
7:0	Data bytes3 [7:0]	-	Байт данных 3 Байт данных должен быть передан с кадром CAN или CAN FD на основе поля управления DLC
Примечание – В соответствии с полем FDF и DLC для данного сообщения должны быть записаны только требуемые местоположения DW			

23.4.4 Регистр статуса TX Event FIFO

Описание Регистра CAN FD TX Event FIFO

Таблица 234 – Пространство сообщений CAN FD TXE

Адрес начала	Наименование	Доступ	Описание
0x2000	TXE FIFO TB0-ID	Чтение	Регистр ID TXE FIFO TB
0x2004	TXE FIFO TB0-DLC	Чтение	Регистр TXE FIFO TB DLC
0x2008	TXE FIFO TB1-ID	Чтение	
0x200C	TXE FIFO TB1-DLC	Чтение	
	:		
0x20FC	TXE FIFO TB31-DLC	Чтение	
Примечания 1 Чтение из неинициализированного участка памяти может вернуть X или неверные данные. Программный или аппаратный сбросы не очищают блоки оперативной памяти; 2 Элемент буфера сообщений находится в ОЗУ блока TX. Хост должен соблюдать правила доступа для чтения, чтобы избежать проблем конфликтов памяти			

23.4.5 Регистр TXE FIFO TB* ID (Address Offset + 0x2000, 0x2008 …)

Таблица 235 – Регистр TXE FIFO TB ID

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
31:21	ID	Статус	-	Стандартный идентификатор сообщения. Часть идентификатора стандартного кадра Эти биты указывают идентификатор стандартного кадра. Это поле действительно как для кадров CAN, так и для CAN FD Standard и Extended
20	SRR/RTR/RRS	Статус	-	Substitute Remote Transmission Request. Для расширенных кадров CAN и CAN FD этот бит передается в позиции SRR соответствующего кадра и должен быть установлен как 1. Для стандартного кадра CAN FD этот бит передается в позиции RRS и должен быть установлен как 0. Этот бит различает стандартные кадры данных CAN и стандартные удаленные кадры CAN, как показано ниже: 1 – Указывает, что кадр сообщения является стандартным удаленным кадром CAN. 0 – Указывает, что кадр сообщения является стандартным кадром данных CAN. Примечание – Удаленных кадров CAN FD нет
19	IDE	Статус	-	Identifier Extension. Этот бит различает кадры, использующие стандартный идентификатор и те, которые использует расширенный идентификатор. Действительно как для кадров CAN, так и для CAN FD Standard и Extended 1 – указывает на использование расширенного идентификатора сообщения. 0 – указывает на использование стандартного идентификатора сообщения
18:1	ID[17:0]	Статус	-	Extended Message ID. В этом поле указывается расширенный идентификатор. Действительно только для расширенных кадров CAN и CAN FD. Для стандартных кадров CAN и CAN FD запись в это поле должна быть 0

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
0	RTR/RRS	Статус	-	Remote Transmission Request. Этот бит различает расширенные кадры данных CAN и расширенные удаленные кадры CAN. 1 – указывает, что кадр сообщения является CAN Remote. 0 – указывает, что кадр сообщения является кадром данных CAN. Для расширенного кадра CAN FD этот бит передается в позиции RRS и должен быть установлен как 0. Для стандартных кадров CAN и CAN FD запись в этот бит должна быть 0. Примечание – Удаленных кадров CAN FD нет

23.4.6 Регистр TXE FIFO TB* DLC (Address Offset + 0x2004, 0x200C …)

Таблица 236 – Регистр TXE FIFO TB DLC

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
31:28	DLC[3:0]	Статус	-	Data Length Code Это код длины данных поля управления кадра CAN или CAN FD
27	FDF	Статус	-	Extended Data Length/FD Frame Format Этот бит различает формат кадра CAN и CAN FD: 1 – кадр формата CAN FD; 0 – кадр формата CAN
26	BRS	Статус	-	Bit Rate Switch Бит BRS определяет, переключается ли битрейт внутри кадра формата CAN FD или нет (при условии, что бит BRSD не установлен в регистре MCP). 1 – битрейт переключается со стандартного битрейта Фазы арбитража до предварительно настроенного альтернативного битрейта фазы данных внутри кадра CAN FD. 0 – битрейт не переключается внутри кадра CAN FD. Примечание – BRS не существует в кадрах формата CAN и должен быть установлен 0
25:24	ET	Статус	-	Event Type: 11 – передано; 01 – Передано, несмотря на запрос отмены или режим передачи DAR; 00 – Зарезервировано; 10 – Зарезервировано

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
23:16	MM	Статус	-	Message Marker. Записывается ЦП во время настройки буфера TX. Скопировано в Элемент Tx Event FIFO для идентификации статуса сообщения TX
15:0	Timestamp	Статус	-	Отметка времени, полученная после бита SOF. Это пишет ядро для статуса успешно переданного сообщения

23.4.7 Пространство сообщений CAN FD RX (Sequential/FIFO Buffers-RX FIFO-0) Register Descriptions

Таблица 237 – Пространство сообщений CAN FD RX (Sequential/FIFO Buffers - RX FIFO-0)

Адрес начала	Наименование	Доступ	Описание	Примечание
0x2100	RB0-ID	чтение	Регистр RB ID	Только необходимые местоположения DW нужно записать согласно полям FDF и DLC для данного сообщения. **Важно**: Убедитесь, что не выполняются непреднамеренные записи от хост-интерфейса к блоку RX RAM message space (ядро не блокирует запись в блок RX RAM message space)
0x2104	RB0-DLC	чтение	Регистр RB DLC	
0x2108	RB0-DW0	чтение	Регистр RB DW0	
0x210C-0x2144	RB0-DW1 – RB0-DW15	чтение		
0x2148-0x218C	RB1	чтение		
:	:	:		
0x32B8-0x32FC	RB63	чтение		

Примечания

1 Чтение из неинициализированного участка памяти может вернуть X или неверные данные. Программный или аппаратный сбросы не очищают блоки оперативной памяти;

2 Элемент буфера сообщений находится в ОЗУ блока RX. Хост должен соблюдать правила доступа для чтения, чтобы избежать проблем конфликтов памяти

23.4.8 Пространство сообщений CAN FD RX (Sequential/FIFO Buffers-RX FIFO-1) Register Descriptions

Таблица 238 – Пространство сообщений CAN FD RX (Sequential/FIFO Buffers - RX FIFO-1)

Адрес начала	Наименование	Доступ	Описание	Примечание
0x4100	RB0-ID	чтение	Регистр RB ID	Только необходимые местоположения DW нужно читать согласно полям FDF и DLC для данного сообщения. **Важно**: Убедитесь, что не выполняются непреднамеренные записи от хост-интерфейса к блоку RX RAM message space (ядро не блокирует запись в блок RX RAM message space)
0x4104	RB0-DLC	чтение	Регистр RB DLC	
0x4108	RB0-DW0	чтение	Регистр RB DW0	
0x410C-0x4144	RB0-DW1 – RB0-DW15	чтение		
0x4148-	RB1	чтение		

Адрес начала	Наименование	Доступ	Описание	Примечание
0x418C				
:	:	:		
0x52B8-0x52FC	RB63	чтение		

Примечания

1 Чтение из неинициализированного участка памяти может вернуть X или неверные данные. Программный или аппаратный сбросы не очищают блоки оперативной памяти.

2 Элемент буфера сообщений находится в ОЗУ блока RX. Хост должен соблюдать правила доступа для чтения, чтобы избежать проблем конфликтов памяти

23.4.9 Регистр RB*-ID (Address Offset + 0x2100, 0x2148 …, 0x4100, 0x4148, …)

Таблица 239 – Регистр RB ID

Биты	Обозначение	Контроль /статус	Значение по сбросу	Назначение
31:21	ID[28:18]	Статус	-	Standard Message ID. Часть идентификатора для стандартного кадра. Эти биты указывают идентификатор стандартного кадра. Это поле действует как для кадров CAN, так и для кадров CAN FD Standard и Extended
20	SRR/RTR/RRS	Статус	-	Substitute Remote Transmission Request. Для расширенных кадров CAN и CAN FD этот бит получен в позиции SRR соответствующего кадра. Для стандартного кадра CAN FD этот бит передается в позиции RRS. Этот бит различает стандартные кадры данных CAN и стандартные удаленные кадры CAN, как показано ниже: 1 – указывает, что кадр сообщения является стандартным удаленным кадром CAN. 0 – указывает, что кадр сообщения является стандартным кадром данных CAN
19	IDE	Статус	-	Identifier Extension. Этот бит различает кадры, использующие стандартный идентификатор и те, которые использует расширенный идентификатор. Действительно как для кадров CAN, так и для CAN FD Standard и Extended 1 – указывает на использование расширенного идентификатора сообщения. 0 – указывает на использование стандартного идентификатора сообщения

Биты	Обозначение	Контроль /статус	Значение по сбросу	Назначение
18:1	ID[17:0]	Статус	-	Extended Message ID. В этом поле указывается расширенный идентификатор. Действительно только для расширенных кадров CAN и CAN FD. Для стандартных кадров CAN и CAN FD, это поле зарезервировано и не имеет смысла
0	RTR/RRS	Статус	-	Remote Transmission Request. Этот бит различает расширенные кадры данных CAN и расширенные удаленные кадры CAN. 1 – указывает, что кадр сообщения является CAN Remote. 0 – указывает, что кадр сообщения является кадром данных CAN. Для расширенного кадра CAN FD этот бит передается в позиции RRS и должен быть установлен как 0. Для стандартных кадров CAN и CAN FD, это поле зарезервировано и не имеет смысла

23.4.10 Регистр RB*-DLC (Address Offset + 0x2104, 0x214C …, 0x4104, 0x414C …)

Таблица 240 – Регистр RB DLC

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
31:28	DLC[3:0]	Статус	-	Data Length Code Это код длины данных поля управления кадра CAN или CAN FD
27	EDL/FDF	Статус	-	Extended Data Length/FD Frame Format Этот бит различает формат кадра CAN и CAN FD: 1 – кадр формата CAN FD; 0 – кадр формата CAN
26	BRS	Статус	-	Bit Rate Switch Бит BRS определяет, переключается ли битрейт внутри кадра формата CAN FD или нет 1 – битрейт переключается со стандартного битрейта Фазы арбитража до предварительно настроенного альтернативного битрейта фазы данных внутри кадра CAN FD. 0 – битрейт не переключается внутри кадра CAN FD. Этот бит не несет смысла, если получен кадр CAN

Биты	Обозначение	Контроль/ статус	Значение по сбросу	Назначение
25	ESI	Статус	-	Error State Indicator Бит ESI показывает статус ошибки отправителя/ передатчика сообщения. 1 – полученный кадр был отправлен пассивным к ошибкам передатчиком. 0 – полученный кадр был отправлен активным к ошибкам передатчиком. Этот бит не имеет значения, если получен кадр CAN
24:21	-	-	-	Зарезервировано
20:16	Matched_Filter _Index[4:0]	Статус	-	Это поле состояния записывается ядром в режиме RX FIFO для предоставления соответствующего индекса фильтра для полученного сообщения. Примечание – Это поле зарезервировано в режиме RX Mailbox и чтение из этого поля возвращает 0
15:0	Timestamp	Статус	-	Отметка времени, полученная после бита SOF. Это пишет ядро для статуса успешно полученного сообщения

23.4.11 Регистр RB*-DW0 (Address Offset + 0x0108, …, 0x0150 …)

Таблица 241 – Регистр TB DW0

Биты	Обозначение	Значение по сбросу	Назначение
31:24	Data bytes0 [7:0]	-	Байт данных 0 Байт данных который получен с кадром CAN или CAN FD на основе поля управления DLC
23:16	Data bytes1 [7:0]	-	Байт данных 1 Байт данных который получен с кадром CAN или CAN FD на основе поля управления DLC
15:8	Data bytes2 [7:0]	-	Байт данных 2 Байт данных который получен с кадром CAN или CAN FD на основе поля управления DLC
7:0	Data bytes3 [7:0]	-	Байт данных 3 Байт данных который получен с кадром CAN или CAN FD на основе поля управления DLC
Примечание – Только необходимые местоположения DW нужно читать согласно полям FDF и DLC для данного сообщения			

23.4.12 Фильтры приема

В режиме RX Sequential/FIFO имеется 32 приемных фильтра. Каждый фильтр приема состоит из Регистра маски фильтра приема и регистра идентификатора фильтра приема (который управляется битами регистра (управления) фильтром приема, описанные в таблице 25).

23.4.13 Фильтрация приема, когда RX FIFO-1 отсутствует или отключен

Приемочная фильтрация выполняется в такой последовательности:

1 Входящий идентификатор маскируется битами регистра маски фильтра приема;

2 Регистр идентификатора фильтра приема также маскируется битами регистра маски фильтра приема;

3 Сравниваются оба полученных значения;

4 Если оба эти значения равны, сообщение сохраняется в RX FIFO-0;

5 Фильтрация приема обрабатывается каждым из определенных фильтров. Если входящий идентификатор проходит через любой приемный фильтр, сообщение сохраняется в RX FIFO-0.

Примечание – RX FIFO-1 можно отключить (то есть остановить маршрутизацию сообщений в RX FIFO-1) путем программирования RXFP как 'd31 (в регистре отметки RX FIFO).

23.4.14 Фильтрация приема при включеном RX FIFO-1

В этом случае поле RXFP (в регистре отметки RX FIF) вместе с регистром (управления) фильтром приема определяет, сохраняются ли полученные сообщения в RX FIFO-0 или RX FIFO-1. В этом случае поле RXFP должно быть меньше 'd31. Входящий идентификатор маскируется битами регистра маски фильтра приема.

1 Регистр идентификатора фильтра приема также маскируется битами регистра маски фильтра приема;

2 Оба полученных значения сравниваются;

3 Если оба эти значения равны, а соответствующий индекс фильтра меньше или равен полю RXFP, сообщение сохраняется в RX FIFO-0;

4 В противном случае, если оба эти значения равны, а соответствующий индекс фильтра больше, чем поле RXFP, сообщение сохраняется в RX FIFO-1.

Процесс сопоставления идентификаторов является последовательным процессом. Он начинается с самого низкого включенного фильтра и останавливается на первом совпадении. Следовательно, если входящее сообщение удовлетворяет условию 8, но RX FIFO-0 переполнен, сообщение отбрасывается (независимо от статуса RX FIFO-1) и указывается переполнение RX FIFO-0.

Точно так же, если входящее сообщение удовлетворяет условию 9 и RX FIFO-1 заполнен, сообщение отбрасывается (независимо от статуса RX FIFO-0) и указывается переполнение RX FIFO-1 (см. рисунки 135 – 138).

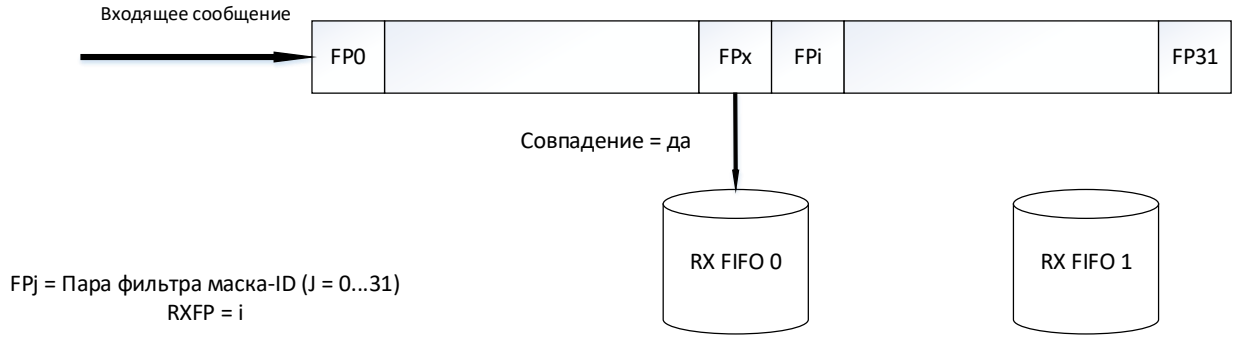


Рисунок 135 – нормальное функционирование RX FIFO 0

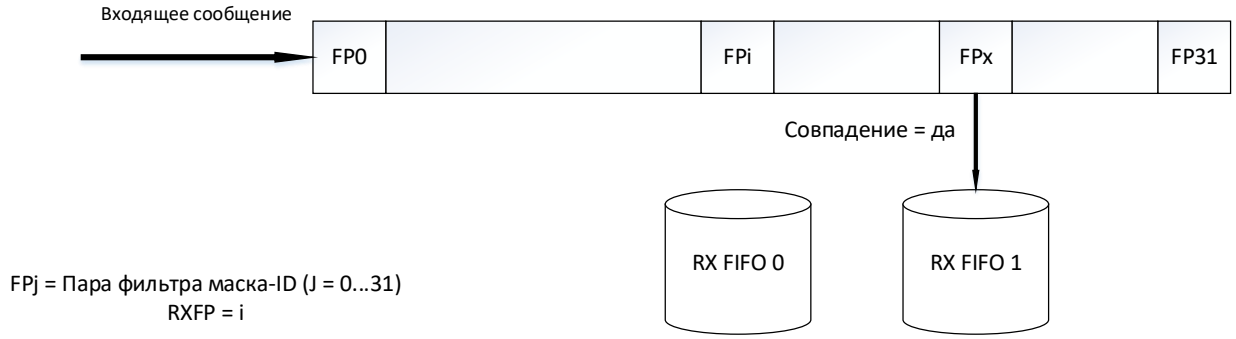


Рисунок 136 – нормальное функционирование RX FIFO 1

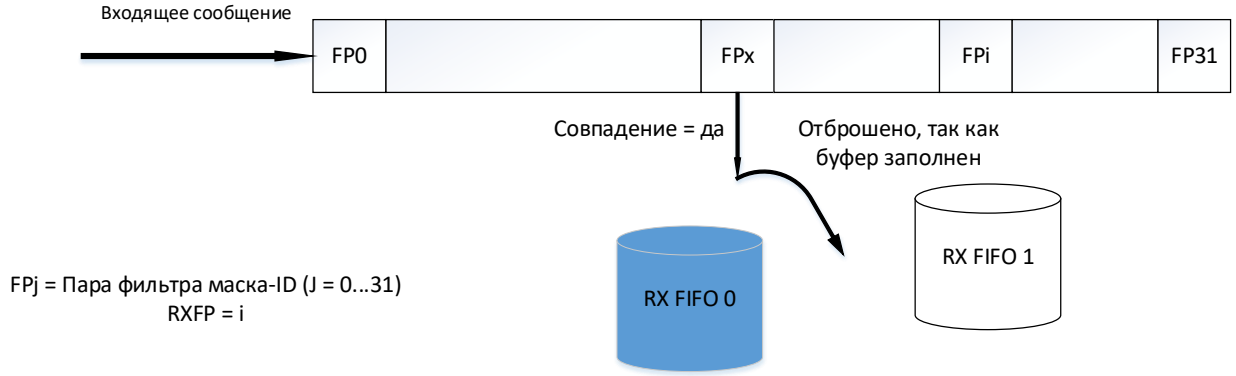


Рисунок 137 – отбрасывание сообщения (RX FIFO 0 заполнен, но совпадение)

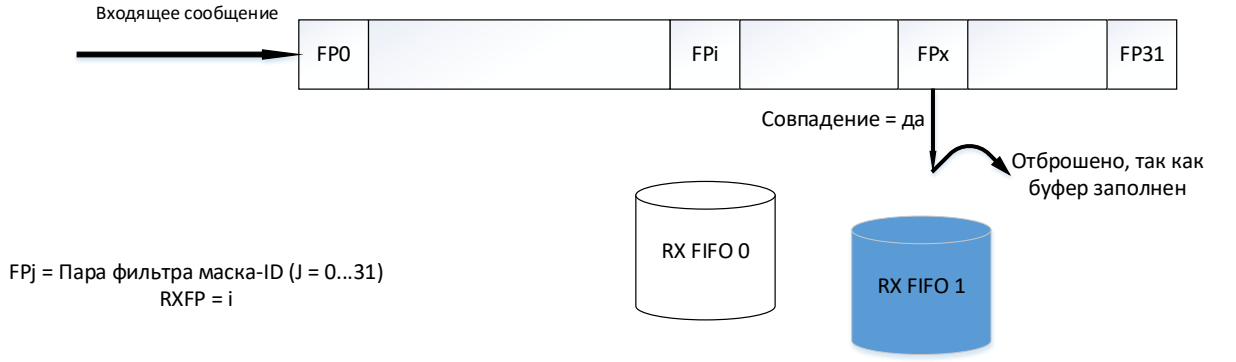


Рисунок 138 – отбрасывание сообщения (RX FIFO 1 заполнен, но совпадение)

Примечание – Если все биты UAF установлены на 0, полученные сообщения не сохраняются ни в одном RX FIFO.

Важно: Обеспечьте правильное программирование бита IDE для стандартных и расширенных кадров в регистре маски и регистре идентификатора. Если вы установите бит IDE в регистре маски равным 0, то будет выполнена только проверка идентификатора стандартного кадра, следовательно, если биты стандартного идентификатора входящего сообщения совпадают с соответствующими битами идентификатора фильтра (после применения маски), сообщение сохраняется.

23.4.15 Регистр AFMR* (Address Offset + 0x0A00, 0x0A08,…)

Регистры маски фильтра приема (AFMR) содержат биты маски, используемые для фильтрации приема. Часть идентификатора входящего кадра сообщения сравнивается с идентификатором сообщения, хранящийся в регистре идентификатора фильтра приема. Биты маски определяют, какие биты идентификатора, хранящиеся в регистре идентификатора фильтра приема, сравниваются с входящим идентификатор сообщения для кадров CAN или CAN FD.

Все битовые поля (AMID[28:18], AMSRR, AMIDE, AMID[17:0] и AMRTR) должны быть определены для Расширенных кадров. Только AMID[28:18], AMSRR и AMIDE должны быть определены для стандартных кадров. AMID[17:0] и AMRTR должны быть записаны 0 для стандартных кадров.

Таблица 242 – Регистр маски фильтра приема

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:21	AMID[28:18]	R/W	0	Standard Message ID Mask. Эти биты используются для маски идентификатора в стандартном кадре: 1 – указывает на то, что соответствующий бит в регистре идентификатора маски приема используется при сравнении входящего идентификатора сообщения; 0 – указывает, что соответствующий бит в регистре идентификатора маски приема не используется при сравнении идентификатора входящего сообщения
20	AMSRR	R/W	0	Substitute Remote Transmission Request Mask. Этот бит используется для маски бита RTR в стандартном кадре: 1 – указывает на то, что соответствующий бит в регистре идентификатора маски приема используется при сравнении входящего идентификатора сообщения; 0 – указывает, что соответствующий бит в регистре идентификатора маски приема не используется при сравнении идентификатора входящего сообщения

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
19	AMIDE	R/W	0	Identifier Extension Mask. Используется для маски бита IDE: 1 – указывает на то, что соответствующий бит в регистре идентификатора маски приема используется при сравнении входящего идентификатора сообщения; 0 – указывает, что соответствующий бит в регистре идентификатора маски приема не используется при сравнении идентификатора входящего сообщения. Если AMIDE равен 1 и бит AIIDE в соответствующем регистре идентификатора приема равен 0, эта маска применима только к стандартным кадрам. Если AMIDE равен 1 и бит AIIDE в соответствующем регистре идентификатора приема равен 1, эта маска применима только к расширенным кадрам. Если AMIDE равен 0, эта маска применима как к стандартному, так и к Расширенному кадрам
18:1	AMID[17:0]	R/W	0	Extended Message ID Mask. Эти биты используются для маски идентификатора в расширенном кадре: 1 – указывает на то, что соответствующий бит в регистре идентификатора маски приема используется при сравнении входящего идентификатора сообщения; 0 – указывает, что соответствующий бит в регистре идентификатора маски приема не используется при сравнении идентификатора входящего сообщения
0	AMRTR	R/W	0	Remote Transmission Request Mask. Эти биты используются для маски бита RTR в расширенном кадре: 1 – указывает на то, что соответствующий бит в регистре идентификатора маски приема используется при сравнении входящего идентификатора сообщения; 0 – указывает, что соответствующий бит в регистре идентификатора маски приема не используется при сравнении идентификатора входящего сообщения

23.4.16 Регистр AFIR* (Address Offset+ 0x0A04, 0x0A0C,…)

Регистры идентификатора фильтра приема (AFIR) содержат биты идентификатора, которые используются для фильтрации приема. Все битовые поля (AIID[28:18], AISRR, AIIDE, AIID[17:0] и AIRTR) должны быть быть определено для расширенных кадров.

Для стандартных кадров необходимо определить только AIID[28:18], AISRR и AIIDE. AIID[17:0] и AIRTR следует записать 0 для стандартных кадров.

Таблица 243 – Регистры идентификаторов фильтра приема

Биты	Обозначение	Права доступа	Значение по сбросу	Назначение
31:21	AIID[28:18]	R/W	0	Standard Message ID Mask. Стандартный идентификатор
20	AISRR	R/W	0	Substitute Remote Transmission Request Mask. Бит запроса удаленной передачи для стандартных кадров
19	AIIDE	R/W	0	Identifier Extension Mask. Бит различия стандартных и расширенных кадров
18:1	AIID[17:0]	R/W	0	Extended Message ID Mask. Расширенный идентификатор
0	AIRTR	R/W	0	Remote Transmission Request Mask. RTR бит для расширенного кадра

23.4.17 Пространство сообщений CAN FD RX (Mailbox Buffers) Register Descriptions

Таблица 244 – пространство сообщений CAN FD RX(Mailbox Buffers)

Адрес начала	Наимено-вание	Доступ	Описание	Примечание
0x2100	RB0-ID	Чтение/ Запись	Регистр RB ID	Только необходимые местополо-жения DW нужно читать согласно DLC для данного сообщения. **Важно**: Убедитесь, что не выполняются непреднамеренные записи от хост-интерфейса к блоку RX RAM message space (ядро не блокирует запись в блок RX RAM message space)
0x2104	RB0-DLC	Чтение/ Запись	Регистр RB DLC	
0x2108	RB0-DW0	Чтение/ Запись	Регистр RB DW0	
0x210C-0x2144	RB0-DW1 – RB0-DW15	Чтение/ Запись		
0x2148-0x257C	RB1-RB15	Чтение/ Запись		
0x2580-0x29FC	RB16-RB31	Чтение/ Запись		Зарезервировано, если количество RX буферов равно 16. В этом случай, ядро не позволяет запись по этому адресу пространства и чтение возвращает 0

Адрес начала	Наимено-вание	Доступ	Описание	Примечание
0x2A00-0x2E7C	RB32-RB47	Чтение/ Запись		Зарезервировано, если количество RX буферов равно 16 или 32. В этом случае, ядро не позволяет запись по этому адресу пространства и чтение возвращает 0
0x2F00	MRB0		Регистр маски фильтра приема. Маска для буфера RB0	MRB16-MRB47 действительны в зависимости от количества RX буферов. Когда кол-во буферов RX равно 16 или 32, ядро не дает доступ для записи вне соответствующего адресного пространства и чтение возвращает 0
0x2F04	MRB1		Маска для буфера RB1	
0x2F08-0x2FBC	MRB2-MRB47		Маска для буфера RB2-RB47	
0x2FC0-0x2FFF	-	-	Зарезервировано, запись не дает результата, чтение возвращает 0.	
Примечания 1 Чтение из неинициализированного участка памяти может вернуть X или неверные данные. Программный или аппаратный сбросы не очищают блоки оперативной памяти; 2 Элемент буфера сообщений находится в ОЗУ блока RX. Хост должен соблюдать правила доступа для чтения, чтобы избежать проблем конфликтов памяти				

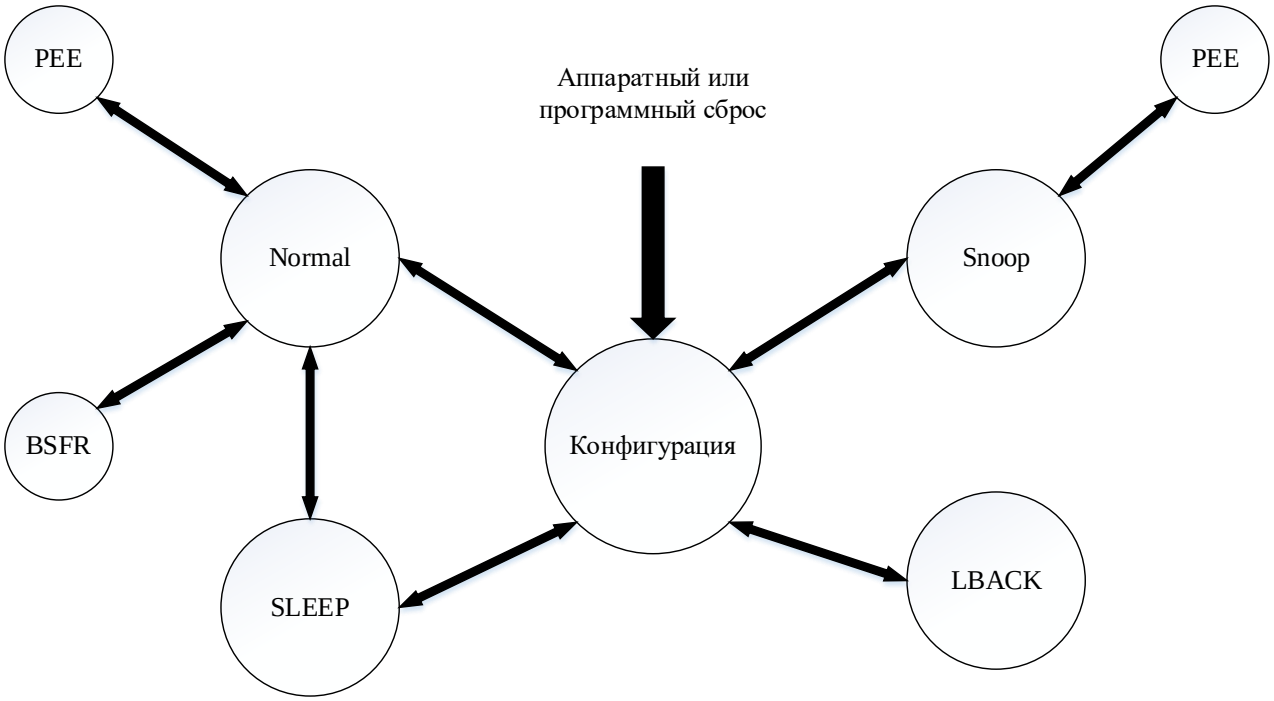
23.5 Работа с блоком

23.5.1 Режимы работы и состояния

Ядро CAN FD поддерживает следующие режимы и состояния:

- Конфигурация
- Обычный
- Loopback
- Sleep
- Snoop
- Состояние исключения протокола (PEE)
- Состояние восстановления при отключении шины (Bus-Off Recovery)

На рисунке 139 показаны переходы основных режимов работы, а в таблице 245 определены режимы операции с соответствующими битами управления и состояния.



После сброса, когда CEN = 1, ядро выходит из режима конфигурации после обнаружения 11 Последовательных рецессивных бит на шине CAN.

Рисунок 139 – Переходы основных режимов работы

Таблица 245 – Переходы режимов работы ядра CAN FD

Аппаратный сброс	Регистр SRR		Регистр MSR			Регистры SR (статус)							Режим работы
	SRST (прог. сброс)	CEN	LBACK	SLEEP	SNOOP	Config	BSFR_Config	PEE_Config	LBACK	SLEEP	NORMAL	SNOOP	
0 (сброс)	X	X	X	X	X	1	0	0	0	0	0	0	Ядро под сбросом
1	1 (сброс)	X	X	X	X	1	0	0	0	0	0	0	Ядро под сбросом
1	0	0	X	X	X	1	0	0	0	0	0	0	Конфигурация
		1	0	0	0	0	0	0	0	0	1	0	Обычный
			0	0	1	0	0	0	0	0	1	1	Snoop
			0	1	0	0	0	0	0	1	0	0	Sleep
			1	0	0	0	0	0	1	0	0	0	Loopback
			0	X	0	0	1	0	0	0	X	0	Bus-Off Recovery
			0	X	X	0	0	1	0	0	X	X	PEE
Примечания 1 X – Бит состояния ничего не значит;													

2 Переход в состояние Bus-Off зависит от значения счетчика ошибок передачи в соответствии со стандартной спецификацией. Восстановление из состояния Bus-Off зависит от настроек битов SBR и ABR в регистре MSR (согласно соответствующему описанию поведения битов). Bus-Off Recovery может отслеживаться через бит состояния BSFR_CONFIG в регистре SR и поле REC в регистре ECR. Вход и выход из состояния Bus-Off могут также генерировать прерывание;

3 Переход в состояние исключения протокола CAN FD (PEE) зависит от бита DPEE в регистре MSR. Ядро входит и выходит из PEE состояния согласно спецификации стандарта ISO, и это отражается битом состояния PEE_CONFIG в регистре SR. Вход в состояние PEE также может генерировать прерывание

23.5.1.1 *Режим настройки*

Ядро входит в режим конфигурирования, независимо от режима работы, после любого из следующих действий:

– Запись 0 в бит CEN в регистре SRR

– Запись 1 в бит SRST в регистре SRR.

– Подача 0 на вход сброса.

– После сброса ядро выходит из режима конфигурации после установки бита CEN и обнаружения 11 номинальных последовательных рецессивных бит на шине CAN.

Характеристика режима конфигурации

Ядро CAN FD имеет следующие характеристики режима конфигурации:

– Контроллер теряет синхронизацию с шиной CAN и подает постоянный рецессивный бит на линию TX.

– Регистр счетчика ошибок сбрасывается.

– Регистр состояния ошибки сбрасывается.

– Регистры BTR и BRPR могут быть изменены.

– Бит CONFIG в регистре состояния равен 1.

– Ядро не получает никаких новых сообщений.

– Ядро не передает никаких сообщений.

– Доступны все регистры конфигурации.

– Если есть сообщения, ожидающие передачи, когда CEN записан 0, они сохраняются (если не отменены) и передаются при возобновлении нормальной работы.

– Отмена сообщения разрешена.

– Новые сообщения могут быть добавлены для передачи (при условии, что бит SNOOP не установлен в регистре MCP).

– Если доступны новые полученные сообщения, они сохраняются до тех пор, пока хост не прочитает их.

– Биты регистра состояния прерывания ARBLST, TXOK, RXOK, RXOFLW, RXOFLW_1, ERROR, BSOFF, SLP и WKUP очищаются.

– Биты регистра состояния прерывания TXTRS и TXCRS могут быть установлены из-за отмены.

– Прерывания генерируются, если соответствующие биты в IER равны 1.

– Находясь в режиме конфигурации, контроллер остается в этом режиме до тех пор, пока бит CEN в регистре SRR равен 1.

– После того, как бит CEN установлен в 1, контроллер ожидает последовательность из 11 номинальных рецессивных бит перед выходом из режима конфигурации.

– CAN FD переходит в режимы Normal, Loopback, Snoop или Sleep из режима конфигурации, в зависимости от битов LBACK, SNOOP и SLEEP в регистре MSR.

23.5.1.2 *Нормальный режим*

Важно: Ядро может перейти в нормальный режим, только если биты Sleep, Loopback и Snoop равны 0 в регистре MSR.

В нормальном режиме ядро участвует в обмене данными по шине, передавая и получая сообщения.

Примечание – В обычном режиме ядро не хранит собственные переданные сообщения.

(Внутренний) Режим Loopback

Важно: Этот режим используется для диагностики.

В режиме Loopback ядро получает любые сообщения, которые оно передает по внутренней обратной связи к линии RX и подтверждает их. Полученные сообщения сохраняются в приемных буферах на основе результата совпадения ID. Ядро хранит собственные переданные сообщения (на основе совпадения идентификатора) в буферах Mailbox или sequential/FIFO буферах в режиме Loopback.

Оно не участвует в обычной коммуникации по шине и не получает никаких сообщений, передаваемых другими узлами CAN (внешняя линия TX игнорируется). Он управляет рецессивным битовым потоком на шине CAN (внешняя линия TX).

23.5.1.3 *Режим Sleep*

Ядро переходит в спящий режим из конфигурационного или нормального режима, когда бит SLEEP равен 1 в регистре MSR, шина CAN простаивает, ожидающих запросов на передачу нет. Ядро входит в режим конфигурации, когда удовлетворяется любое условие конфигурации. Ядро переходит в нормальный режим (очистка бита запроса SLEEP в регистре MSR, а также очистка соответствующего бита состояния) при следующих условиях (пробуждения):

– Всякий раз, когда бит SLEEP установлен на 0.

– Всякий раз, когда бит SLEEP равен 1 и обнаруживается активность шины.

– Всякий раз, когда есть новое сообщение для передачи.

Прерывания генерируются, когда ядро переходит в режим сна или выходит из режима сна.

23.5.1.4 *Режим Snoop (мониторинг шины)*

Важно: Этот режим используется для диагностики.

Особенности режима Snoop следующие:

– Ядро передает рецессивные биты на шину CAN.

– Ядро принимает сообщения, которые передаются другими узлами, но не ACK. Сохраняет полученные сообщения на основе запрограммированной фильтрации идентификаторов.

– Счетчики ошибок отключены и сброшены в 0. Считывание регистра счетчика ошибок, возвращает 0.

Рекомендуется: программировать режим Snoop только после перезагрузки системы или программного сброса.

23.5.1.5 *Состояние исключения протокола(PEE)*

CAN FD входит в состояние исключения протокола CAN FD (PEE), если он получает бит res, который должен быть рецессивным в кадре CAN FD (при условии, что бит DPEE не установлен в регистре MSR). Ядро выходит из этого состояния после обнаружения последовательности из 11 номинальных рецессивных битов на CAN шине и, в соответствии со спецификацией протокола, количество ошибок при передаче и приеме остается неизменным в этом состоянии.

23.5.1.6 *Состояние восстановления при отключении шины (Bus-Off Recovery)*

CAN FD переходит в состояние отключения шины, если счетчик ошибок передачи достигает или превышает его предел. Восстановление из состояния Bus-Off регулируется установкой бита автоматического восстановления (ABR) или ручного восстановления (SBR) в регистре MSR и выполняется в соответствии с спецификацией.

23.5.2 Программная модель

В этом разделе описаны различные этапы настройки, которые необходимо выполнить для программирования CAN FD для работы. В этом разделе подробно описаны следующие основные этапы настройки:

1 Программирование регистров конфигурации для инициализации CAN FD в зависимости от режима работы.

2 Передача, отмена и прием сообщений.

23.5.2.1 *Последовательность конфигурации регистров*

Ниже приведены шаги по настройке CAN FD при включении ядра или после системного, или программного сброса.

1 Выберите режим работы:

– Обычный — запишите 0 в биты LBACK, SNOOP и SLEEP в MSR. Запишите необходимые значение для полей BRS и DAR в регистре MSR.

– Sleep – Запишите 1 в бит SLEEP в MSR и 0 в биты LBACK и SNOOP в MSR. Запишите необходимое значение для полей BRS и DAR в регистр MSR.

– Loopback — запишите 1 в бит LBACK в MSR и 0 в биты SLEEP и SNOOP в MSR. Запишите необходимое значение для полей BRS в регистр MSR.

– Snoop – Запишите 1 в бит SNOOP в MSR и 0 в биты LBACK и SLEEP в регистре MSR.

2 Настройте регистры конфигурации транспортного уровня.

Важно: Для правильной работы убедитесь, что все узлы CAN FD в сети запрограммированы на одинаковые битрейт фазы арбитража, битрейт фазы данных, положение точки выборки фазы арбитража и положение точки выборки фазы данных.

– Запрограммируйте регистр делителя (номинального) битрейта фазы арбитража и регистр тактовой синхронизации этапа арбитража со значением, рассчитанным для конкретного битрейта фазы арбитража.

– Запрограммируйте регистр предделителя битрейта фазы данных и регистр тактовой синхронизации со значением для достижения желаемого битрейта фазы данных.

– Регистр синхронизации битов фазы данных также содержит поля управления TDC.

Примечание – Скорость передачи данных, настроенная для фазы данных, должна быть выше или равна скорости передачи данных, настроенной для фазы арбитража. Регистры конфигурации уровня передачи могут быть изменены только тогда, когда бит CEN в регистре SRR равен 0.

Важно: Шаг 3 предназначен только для режима RX Sequential/FIFO.

3 Настройте регистры фильтра приема (AFR, AFMR, AFIR) следующим образом:

– Запишите 0 в бит UAF в регистре, соответствующем паре регистров маска – ID фильтра приема, которые необходимо настроить.

– Запишите требуемую информацию о маске в регистр маски фильтра приема.

– Запишите требуемую информацию об идентификаторе в регистр идентификатора фильтра приема.

– Запишите 1 в бит UAF, соответствующий паре регистров маски – ID фильтра приема.

– Повторите шаги для каждой пары регистров маски – ID.

– Если вы хотите включить RX FIFO-1, вам необходимо настроить маску фильтра и регистр идентификатора согласно требованию. В поле RXFP в регистре отметки RX FIFO также необходимо установить значение меньше, чем 'd31.

Важно: Шаг № 4 предназначен только для режима RX Mailbox.

4 Настройте регистры маски (MRB) для буферов RX Mailbox. Настройте регистр RB-ID соответствующего буфера и установите бит управления в регистре RCS, чтобы сделать буфер Активным.

5 Запрограммируйте регистры разрешения прерывания в соответствии с требованиями.

6 Включите контроллер протокола, записав 1 в бит CEN в регистре SRR.

После появления 11 последовательных рецессивных битов CAN FD очищает бит CONFIG в регистре состояния на 0 и устанавливает соответствующий бит режима в регистре состояния.

Рекомендуется: если бит CEN очищается во время работы ядра, рекомендуется сбросить ядро, после чего работа начинается заново. Кроме того, биты LBACK, SLEEP и SNOOP никогда не должны быть устанавлины в 1 в одно и то же время.

23.5.2.2 *Передача, отмена и прием сообщений*

Передача

Все сообщения, записываемые в буфер TX, должны соответствовать требуемому формату сообщения для полей ID, DLC и DW описанных ранее. Каждый бит RR регистра запроса готовности буфера передачи (TRR) соответствует сообщению в ОЗУ блока TX.

TX – действия хоста:

1 Опросите регистр TRR, чтобы проверить текущие ожидающие запросы на передачу.

2 Если все биты регистра TRR установлены, новый запрос на передачу может быть добавлен только в том случае, если:

– Один или несколько запросов на передачу буфера отменены, или

– Одна или несколько передач завершены.

3 Если один или несколько битов регистра TRR не установлены/очищены, может быть отправлен новый запрос на передачу, по следующему принципу:

– Во-первых, подготовьте один или несколько сообщений в ОЗУ блока TX (путем записи правильных полей ID, DLC и DW каждого элемента сообщения соответствующего буфера TX). Если требуется протоколирование событий для этого элемента сообщения, установите бит EFC в поле DLC.

– Включите генерацию прерываний по мере необходимости.

– Установите соответствующие биты TRR, чтобы разрешить запросы готовности буфера. Хост может включить несколько запросов на передачу за одну запись в регистр TRR.

– Дождитесь прерывания (если разрешено) или опросите регистр TRR, чтобы получить статус запроса.

4 CAN FD очищает бит TRR, когда соответствующий запрос буфера завершен (либо из-за успешной передачи, или отмены, или из-за передачи в режиме DAR).

5 Хост может прочитать TX Event FIFO, чтобы узнать отметки времени и порядок передачи сообщений.

Примечание – Бит TXOK в ISR устанавливается после успешной передачи сообщения ядром. Бит ARBLST в ISR устанавливается, если CAN FD теряет арбитраж шины при передаче сообщения. Бит ERROR в ISR устанавливается, если при передаче сообщения произошла какая-либо ошибка.

TX — действия ядра:

1 CAN FD вычисляет буфер для передачи по наивысшему приоритету. Если два буфера имеют одинаковый ID, выбирается буфер с меньшим индексом.

2 Если включено, копирует поля ID и DLC в TX Event FIFO и добавляет отметку времени сообщению и тип события.

3 Он очищает соответствующий бит TRR, когда запрос на передачу обслужен (либо успешной передачей по CAN-шине, либо из-за отмены, либо из-за DAR передачи).

4 Если разрешено IETRS и IER, бит TXRRS устанавливается в регистре ISR и генерируется прерывание.

Примечание – CAN FD обращается к буферу в пространстве элементов сообщений блока TX ОЗУ, только если установлен соответствующий бит TRR. Хост должен соблюдать правила доступа, чтобы избежать конфликтов памяти, то есть после того, как Хост устанавливает запрос готовности буфера в регистре TRR, он не должен читать или записывать соответствующее пространство элемента сообщения до тех пор, пока соответствующий бит RR не будет очищен/установлен в 0.

Отмена передачи

Каждый бит CR регистра TX Buffer Cancel Request (TCR) соответствует элементу сообщения в блоке TX ОЗУ (и, следовательно, соответствует биту RR регистра TRR).

TX — действия хоста:

1 Опросить регистр TRR, чтобы проверить текущие ожидающие запросы на передачу.

2 Опросить регистр TCR, чтобы проверить текущие ожидающие запросы на отмену.

– Отмена передачи для буфера (TXB_i) может быть запрошена только при наличии соответствующего ожидающего запроса на передачу, установленного в регистре TRR.

– Если уже есть ожидающий запрос на отмену для TXB_i, никаких действий не требуется и Хост должен ждать (посредством опроса/прерывания), пока ядро не обслужит запрос на отмену для TXB_i.

3 Если в буфере TXB_i есть ожидающий запрос на передачу, но нет ожидающего запроса на отмену, то запрос на отмену передачи можно создать следующим образом:

– Включите генерацию прерываний, если требуется.

– Установите требуемый бит/биты CR регистра TCR. Хост может запросить отмену сразу нескольких буферов за одну запись в регистр TCR.

– Дождитесь прерывания или опросите регистр TCR, чтобы проверить статус отмены.

4 CAN FD очищает бит в регистре TCR, когда запрос на отмену соответствующего буфера выполнен.

5 CAN FD также очищает соответствующий бит в регистре TRR при совершенной отмене.

TX — действия ядра:

1 CAN FD немедленно выполняет отмену буфера, за исключением следующих случаев:

– Когда буфер заблокирован транспортным уровнем для передачи по шине CAN. В этом случае отмена выполняется в конце передачи независимо от того была ли передача успешной или неудачной (потеря арбитража или ошибка).

– Когда ядро выполняет раунд планирования, чтобы выбрать следующий буфер для передачи. В этом случае отмена выполняется после завершения раунда планирования.

2 CAN FD очищает соответствующие биты в регистрах TCR и TRR при выполненной отмене.

3 Если разрешено IETCS и IER, бит TXRCS устанавливается в регистре ISR (когда ядро очищает бит в регистре TCR) и генерируется прерывание.

Прием (режим Sequential Buffer/FIFO)

Всякий раз, когда новое сообщение (прошедшее требуемую фильтрацию) сохраняется в RX FIFO, ядро обновляет соответствующее поле Fill Level регистра FSR и устанавливает бит RXOK в регистре ISR.

RX — действия хоста:

1 В соответствии с требованием запрограммируйте регистр отметки RX FIFO для установки отметки заполнения и поле RXFP (Регистр отметки RX FIFO можно установить/изменить только при CEN = 0).

2 При необходимости включите генерацию прерываний RXOK и RX Overflow.

3 Доступность нового сообщения можно узнать путем опроса регистра FSR или по индикации прерывания Watermark Full.

4 Прочтите новое сообщение (из RX FIFO-0 или RX FIFO-1), начиная с соответствующего индекса местоположения чтения (указывается в поле регистра FSR).

5 После прочтения сообщения запишите регистр FSR, установив соответствующий бит IRI в 1. Это позволяет ядру увеличивать соответствующее поле индекса чтения на +1 и обновлять соответствующий уровень заполнения в регистре FSR. Если Fill level равен 0, установка бита IRI не имеет эффекта.

6 Повторяйте шаги с 3 по 5, пока все сообщения не будут прочитаны либо из RX FIFO-0, либо из RX FIFO-1.

RX —действия ядра:

1 Когда сообщение успешно получено, ядро записывает временную метку и поле индекса совпадающего фильтра полученного элемента сообщения.

2 CAN FD увеличивает уровень заполнения, соответствующего RX FIFO в регистре FSR на 1. после каждого успешного приема (без ошибок и успешной фильтрации сообщений).

3 Уровень заполнения также обновляется ядром после того, как хост записывает бит IRI соответствующего RX FIFO в регистре FSR.

Фильтрация

Каждая пара приемных фильтров имеет регистр маски фильтра приема и регистр ID фильтра приема. Каждая пара имеет соответствующий бит UAF для включения/отключения фильтра.

Фильтрация при отсутствии или отключённом RX FIFO-1:

Фильтрация выполняется в следующей последовательности:

1 Входящий идентификатор маскируется битами в соответствии с регистром маски приемного фильтра.

2 Регистр ID фильтра также маскируется битами регистра маски приемлемого фильтра.

3 Сравниваются оба полученных значения.

4 Если оба эти значения равны, то сообщение сохраняется в RX FIFO-0.

5 Фильтрация выполняется каждым из заданных фильтров. Если входящий идентификатор проходит через любой из фильтров сообщение сохраняется в RX FIFO-0.

Примечание – RX FIFO-1 можно отключить (то есть остановить маршрутизацию сообщений в RX FIFO-1) путем программирования RXFP как 'd31 (в регистре отметки RX FIFO). См. описание регистра для более подробной информации.

Фильтрация при включённом RX FIFO-1:

В этом случае, поле RXFP (в регистре RX FIFO Watermark) вместе с регистром (управления) фильтром определяет, сохраняются ли полученные сообщения в RX FIFO-0 или RX FIFO -1. В этом случае поле RXFP должно быть меньше 'd31.

1 Входящий идентификатор маскируется битами регистра маски приемного фильтра.

2 Регистр идентификатора фильтра также маскируется битами регистра маски фильтра.

3 Оба полученных значения сравниваются.

4 Если оба эти значения равны, а соответствующий индекс фильтра меньше или равен полю RXFP, сообщение сохраняется в RX FIFO-0.

5 В противном случае, если оба эти значения равны, а соответствующий индекс фильтра больше, чем поле RXFP, сообщение сохраняется в RX FIFO-1.

Процесс сопоставления идентификаторов является последовательным процессом. Он начинается с самого низкого включенного фильтра и останавливается на первом совпадении. Следовательно, если входящее сообщение удовлетворяет условию 4, но RX FIFO-0 заполнен, сообщение отбрасывается (независимо от статуса RX FIFO-1) и в RX FIFO-0 указывается переполнение.

Точно так же, если входящее сообщение удовлетворяет условию 5, но RX FIFO-1 заполнен, сообщение отбрасывается (независимо от статуса RX FIFO-0) и указывается переполнение в RX FIFO-1. Смотреть Рисунок 2-1 – Рисунок 2-4, для получения подробной информации.

Примечание – Если все биты UAF установлены на 0, полученные сообщения не сохраняются ни в одном RX FIFO.

Регистры пары фильтров хранятся в оперативной памяти блока. Хост должен убедиться, что каждая используемая пара фильтров правильно инициализирована. Программный или аппаратный сброс не очистить содержимое этих регистров.

Хост должен инициализировать/обновлять/изменять пару фильтров, только если соответствующий бит UAF равен 0.

Важно: Обеспечьте правильное программирование бита IDE для стандартных и расширенных кадров в регистре маски и регистре ID. Если вы установите бит IDE в регистре маски равным 0, то будет выполняться проверка стандартного ID и, следовательно, если биты стандартного ID входящего сообщения совпадают с битами стандартного ID фильтра (после применения битов регистра маски), сообщение сохраняется.

Прием (режим Mailbox):

Каждый буфер приема ОЗУ блока RX имеет два бита. Бит HCBx дает Хосту управление, чтобы делать буфер активным или неактивным, а бит CSBx дает состояние ядра (буфер полон). Вместе эти два бита определяют состояние буфера как неактивный, активный, полный или недействительный. Эти биты подробно описаны в регистрах состояния управления буфером приема (RCS). Каждый буфер приема также имеет один регистр маски ID в ОЗУ блока RX.

RX — действия хоста:

1 Хост подготавливает один или несколько приемных буферов для приема сообщений следующим образом:

– Если буфер неактивен:

– Запишите требуемый ID в поле ID элемента Receive Buffer_i в блоке памяти RAM.

– Запишите соответствующий регистр маски для элемента Receive Buffer_i в блоке RAM.

– Если буфер находится в активном/полном/недействительном состоянии, а хост хочет изменить его ID/маску:

– Измените состояние буфера на Неактивный.

– Запишите требуемый ID в поле ID элемента Receive Buffer_i в блоке памяти RAM.

– Запишите соответствующий регистр маски для элемента Receive Buffer_i в блоке RAM.

2 При необходимости включите генерацию прерываний.

3 Измените статус буфера с Неактивный на Активный. Хост может изменить статус с Неактивный на Активный для многих буферов за одну запись в регистр RCS.

4 Дождитесь прерывания или опросите регистры RCS, чтобы узнать состояние буфера.

5 Хост может читать сообщения из блока ОЗУ RX, имеющего статус Full. После чтение, хост может изменить статус буфера обратно на Активный (если вы хотите

продолжить с теми же ID/маска) или Неактивный (если вы хотите перепрограммировать ID/маску).

Примечание – CAN FD может считывать поле идентификатора полного буфера для текущего процесса сопоставления, поэтому Хост не должен изменять идентификатор буфера, когда он находится в состоянии «Полный».

При необходимости Хост может удалить сообщение, не читая его, изменив статус с полного на активный/неактивный.

Примечание – Всякий раз, когда новое сообщение успешно получено и записано в любой из приемных буферов, бит RXOK устанавливается в регистре ISR, а поле RXLRM_BI фиксирует индекс почтового ящика, в котором сообщение было сохранено. Предусмотрено получение прерывания для любого выборочного буфера или буферов. В случае переполнения в ISR устанавливается бит RXOFLW. Поле RXOFLW_BI в регистре ISR фиксирует индекс переполнения, соответствующий последнему событию переполнения, и поддерживает его до тех пор, пока не будет очищен бит RXOFLW.

RX — действия ядра:

1 CAN FD ищет активные буферы, начиная с самого нижнего индекса, которое соответствует ID входящего сообщения. Когда в активных пустых буферах не найдено совпадений, но совпадение находится в полном буфере, генерируется состояние переполнения и соответствующий индекс буфера фиксируется в ISR. Если совпадения также нет и в полных буферах, сообщение отбрасывается без указания.

2 CAN FD изменяет состояние буфера на «Полный», когда сообщение получено без ошибок и имеет отметку времени. В случае ошибок (например, ошибка CRC) состояние буфера остается активным.

3 Если разрешено IERBF и IER, бит RXRBF устанавливается в регистре ISR (когда ядро изменяет состояние буфера RCS на «Полный») и генерируется прерывание.

Примечания

1 CAN FD обращается к буферу пространства элементов сообщений RAM блока RX на основе состояние буфера.

– Активный: доступ на чтение для идентификатора и маски. Доступ на запись для полученного сообщения. Доступ на чтение и запись для отметки времени.

– Полный: доступ на чтение для идентификатора и маски, чтобы найти условие переполнения.

2 Хост должен соблюдать правила доступа, чтобы избежать конфликтов памяти. Например:

– Если состояние буфера Активный, не обращайтесь к соответствующему пространству ОЗУ.

– Если статус буфера Полный, не изменяйте соответствующие ID и Маску.

Важно: Поскольку регистры идентификатора и маски находятся в ОЗУ блока, программный или аппаратный сбросы не очищает содержимое этих регистров. Хост должен правильно инициализировать их перед использованием.

Примечания к процессу сопоставления идентификаторов

1 Ожидается, что тактовая частота APB достаточно высока, чтобы процесс сопоставления завершался до завершения приема кадра по шине CAN.

2 Если ядро не может завершить процесс сопоставления до шестого бита EOF, оно устанавливает бит RX MNF в регистре ISR.

3 Логика управления RX MAilbox на объектном уровне ожидает сигнала транспортного уровня RXOK (шестой бит EOF) перед установкой соответствующего бита RCSx(i), чтобы указать, что RX буфер заполнен.

4 Шина CAN может столкнуться с ошибкой после поля данных текущего кадра (шестой бит поля EOF). В этой ситуации логика управления Mailbox не установливает бит RCSx(i), чтобы указать, что буфер RX заполнен. Сопоставленный элемент блока ОЗУ RX может отображать частичное или полное обновление данных.

5 Идентификатор, полученный с сообщением, записывается в поле ID буфера Mailbox.

Пример 1:

Запрограммированный хостом идентификатор и маска:

ID: 0x1234_5678

Маска: 0xFFFF_FF00

Входящие идентификаторы 0x1234_56xx будут соответствовать этому буферу Mailbox.

Входящее сообщение имело ID: 0x1234_56AB.

Тогда после получения сообщения, ID будет таким:

ID: 0x1234_56AB

Маска: 0xFFFF_FF00

Пример 2:

Запрограммированный хостом идентификатор и маска:

ID: 0xABCD_1234

Маска: 0000_0000

Поскольку маска равна 0x0, любые входящие ID будут соответствовать этому буферу.

Входящее сообщение имело ID: 0x5678_4321.

Тогда после получения сообщения, ID будет таким:

ID: 0x5678_4321

Маска: 0x0000_0000

23.5.3 Тактирование

CAN FD имеет два тактовых сигнала: can_clk и apb_clk. Требуется, чтобы apb_clk имел большую частоту, чем часы can_clk.

- Тактовая частота can-clk может составлять от 8 до 80 МГц.
- Тактовая частота apb_clk может составлять от 8 до 200 МГц.

Примечание – При аппаратной реализации протокола рекомендуется использовать тактовую частоту CAN 20, 40 или 80 МГц. Для получения дополнительной информации см. Robustness of a CAN FD Bus System – AboutOscillator Tolerance and Edge Deviations.

Важно: can_clk должен соответствовать диапазону допуска генератора, указанному в соответствующих стандартах.

23.5.4 Сбросы

CAN FD можно сбросить с помощью входного порта системного (аппаратного) сброса или через управляемый программным обеспечением сброс, обеспечиваемый битом SRST в регистре SRR. Оба системный и аппаратный сброс сбрасывают все ядро CAN FD (то есть как объектный уровень, так и транспортный уровень).

Транспортный уровень остается в состоянии сброса до тех пор, пока бит CEN (разрешение CAN) в регистре SRR равен 0 (то есть бит CEN является третьим источником сброса для транспортного уровня). Если бит CEN очищается во время работы ядра, рекомендуется сбросить ядро, чтобы начать работу сначала.

Объектный уровень сбрасывается синхронно относительно двух упомянутых выше источников. (то есть внутреннее начало и окончание сброса на объектном уровне выполняется синхронно с тактовым сигналом apb_clk).

Транспортный уровень сбрасывается асинхронно по отношению к вышеупомянутым трем источникам (то есть внутреннее начало сброса на транспортном уровне является асинхронным, тогда как окончание сброса достигается синхронно по отношению к часам CAN). Транспортный уровень сбрасывается, ядро теряет синхронизацию с CAN-шиной и гонит рецессивные биты на линии TX.

23.5.4.1 *Системный (аппаратный) сброс*

Системный (аппаратный) сброс можно включить, установив 0 на входной порт сброса. Все регистры конфигурации сбрасываются до значений по умолчанию. Транзакции чтения/записи не могут быть выполнены, когда вход сброса равен 0. Когда применяется системный сброс, текущие транзакции APB могут резко завершиться. Как правило, импульс системного сброса должен быть дольше, чем по крайней мере два такта CAN.

Важно: Поскольку транспортный уровень сбрасывается асинхронно, убедитесь, что линия сброса работает без сбоев.

23.5.4.2 *Программный сброс*

Программный сброс можно включить, записав 1 в бит SRST в регистре SRR. Когда установлен программный сброс, все регистры конфигурации, включая бит SRST в регистре SRR сбрасываются до значений по умолчанию. Транзакции чтения/записи могут быть выполнены начиная с следующего допустимого окна транзакции (которое начинается после шестнадцати тактов apb_clk после программного сброса).

Важно: Содержимое блоков ОЗУ TX и RX не очищается при любом применяемом сбросе.

23.5.5 Прерывания

Ядро имеет один выход прерывания для индикации прерывания. Прерывания обозначаются активацией линии ip2bus_intrevent (переход линии с логического 0 на логическую 1). Начало и окончание прерывания синхронно с apb_clk.

Такие события, как ошибки на линии шины, передача и прием сообщений, а также различные другие условия могут генерировать прерывания. При включении питания на линии прерывания устанавливается низкий уровень.

Регистр состояния прерывания (ISR) указывает биты состояния прерывания. Эти биты устанавливаются и очищаются независимо от состояния соответствующего бита в регистре разрешения прерывания (IER). IER отвечает за функцию разрешения прерываний. Очистка бита состояния в ISR обрабатывается записью 1 в соответствующий бит в регистре очистки прерываний (ICR).

Два условия, которые вызывают активацию линии прерывания:

– Если бит в ISR равен 1 и соответствующий бит в IER равен 1.

– Изменение бита IER с 0 на 1, когда соответствующий бит в ISR уже равен 1.

Два условия, которые вызывают сброс линии прерывания:

– Очистка бита в ISR (путем записи 1 в соответствующий бит в регистре ICR при условии, что соответствующий бит в IER равен 1).

– Изменение бита IER с 1 на 0, когда соответствующий бит в ISR равен 1.

Когда оба условия активации и сброса приходят одновременно, линия сброса сначала сбрасывается, а затем активируется, если условие активации осталось истинным.

24 Интерфейс I2C

Интерфейс I2C ©Philips использует для обмена две линии:
– SDA (данные);
– SCL (синхронизация).

Назначение внешних выводов микросхемы для I2C приведено в таблице 246.

Таблица 246 – Назначение внешних выводов для I2C

Обозначение вывода	Назначение вывода для I2C	Тип	Функциональное назначение
PC[11]	I2C_SDA	I/O	Интерфейс I2C. SDA
PC[12]	I2C_SCL	I/O	Интерфейс I2C. SCL

Подключение интерфейса I2C к внешним выводам определяется регистром CFG1, подробнее см. подраздел 31.1 «Регистр конфигурации периферийных модулей CFG1».

Для передачи выводов PC, указанных в таблице 246, под управление интерфейса I2C требуется только включить интерфейс битом I2C_ON регистра CR при выключенной альтернативной функции выводов порта PC.

Регистры интерфейса подключены к шине обмена периферийных устройств и имеют базовый адрес **0x800002E0**. Краткое описание регистров приведено в таблице 247.

Таблица 247 – Регистры интерфейса I2C

Смещение	Название	Выполняемая функция регистра
0	CR	Управление
1	SR	Состояние
2	DR	Данные
3	AR	Адрес
4	VR	Делитель клока
5	CR_set	Установка бит регистра управления
6	CR_clear	Сброс бит регистра управления
7	AXR	Расширение адреса
8	INFO	Информация о внутреннем состоянии интерфейса
9	PR	Значение внешних линий SCL и SDA

Интерфейс использует в качестве базовой частоты частоту обмена по шине периферийных устройств SCLK (частота SOC-шины, равная половине процессорной частоты). Реальная частота обмена получается после деления базовой частоты на 11-разрядный коэффициент деления. Коэффициент деления задается в регистре VR. Таким образом:

$$\mathrm{SCL\_freg} = \frac{\mathrm{SCLK}}{4 \cdot \mathrm{VR}[11:0]}. \quad (9)$$

Предположим, что частота ядра процессора 200 МГц, частота периферийной шины SCLK 100 МГц. Нам необходимо обеспечить частоту интерфейса SCL_freq равной

100 кГц. Коэффициент деления получаем делением 100 МГц на 100 кГц и дополнительным делением на четыре. В результате получаем 250. В регистр VR записываем константу 250.

Интерфейс реализован таким образом, что выполнение обмена предполагает тесное взаимодействие процессора и аппаратной части интерфейса. При этом алгоритм работы зависит от режима, в котором используется интерфейс, а именно:

– работает интерфейс как мастер или как подчиненный;

– используется стандартная семибитовая адресации или расширенная 10-битовая;

– работает ли интерфейс как единственный мастер на линиях или в мультимастерной системе.

Все особенности, приведенные выше, будут влиять на разработку программной части алгоритма обмена. В общих чертах алгоритм работы интерфейса заключается в следующем:

– пользовательская программа записывает в регистр управления некоторую команду;

– интерфейс выполняет требуемые действия и после их завершения устанавливает флаг прерывания и код состояния;

– пользовательская программа обнаруживает запрос прерывания, считывает регистр состояния и выполняет необходимые действия, после чего очищает флаг запроса прерывания;

– после очистки флага прерывания интерфейс выполняет действия, соответствующие его новому состоянию.

Более подробно особенности различных режимов работы будут описаны в соответствующих подразделах.

24.1 Регистры интерфейса

24.1.1 Регистр управления CR

Регистр задает основные конфигурационные и управляющие параметры. Описание разрядов регистра приведено в таблице 248.

Таблица 248 – Регистр CR

Бит	Имя	Назначение
0	ABRT	Флаг аварийного (преждевременного) завершения работы интерфейса в режиме мастера. *Устанавливается аппаратно и программно, сбрасывается программно*
1	SLVE	Флаг завершения работы интерфейса в режиме подчиненного. Устанавливается в «1» при работе интерфейса в режиме подчиненного в момент получения условия “stop” при установленном бите SLVE_EN. *Устанавливается аппаратно и программно, сбрасывается программно*
2	AA	Когда равен 1: – разрешает выдачу бита подтверждения в фазе приема данных; – разрешает сравнение адреса в фазе приема адреса

Бит	Имя	Назначение
3	SI	Запрос прерывания. Аппаратная установка флага в «1» сопровождается записью некоторого кода в регистр состояния. *Устанавливается аппаратно и программно, сбрасывается программно*
4	STO	Формирование «STOP» условия при работе в режиме мастера. *Устанавливается программно, сбрасывается аппаратно и программно*
5	STA	Формирование «START» условия при работе в режиме мастера
6	I2C_ON	Разрешение работы I2C интерфейса. Если бит равен нулю, происходит сброс интерфейса в начальное состояние. Установка бита в «1» разрешает работу интерфейса
7	M10	Режим работы интерфейса при расширенной (10 бит) адресации. Имеет значение только при работе в режиме подчиненного: 0 – используется адрес регистра AR; 1 – кроме адреса AR используется дополнительный адрес AXR
8	SLVE_EN	Разрешение генерации события при обнаружении на линиях интерфейса ситуации «СТОП». Имеет смысл только в случае, когда интерфейс работает в режиме подчиненного и активен: 0 – при работе интерфейса ситуация «СТОП» не вызывает генерации запроса прерывания к процессору; 1 – если интерфейс будучи активным подчиненным устройством (к нему в данный момент обращается мастер) обнаруживает на линиях состояние «СТОП», то происходит установка флагов IRQ и SLVE, и запись соответствующего кода в регистр состояния
9	ABN_EN	Разрешение детектирования конфликтов на линии данных при работе в мультимастерной системе: 1 – разрешено; 0 – запрещено
10	TLOW_EN	1 – разрешение контролировать параметр T_low при выполнении мастером процедуры «restart»; 0 – пользователь самостоятельно контролирует соблюдение параметра T_low при выполнении «restart»
11	IRQ	Запрос прерывания. Доступен по чтению и записи. Устанавливается аппаратно при аппаратной установке флагов SI, ABRT, SLVE. Сбрасывается программно
12	FREE_EN	1 – разрешена генерация прерывания если линии интерфейса готовы для обмена; 0 – запрещена
13	-	
14	ABN_S_EN	Разрешение детектирования конфликтов на линиях интерфейса при работе в мультимастерной системе: 1 – разрешает анализ нарушений при генерации события «рестарт» и генерацию прерывания; 0 – анализ запрещен

Бит	Имя	Назначение
15	ABN_P_EN	Разрешение детектирования конфликтов на линиях интерфейса при работе в мультимастерной системе: 1 – разрешает анализ нарушений при генерации события «стоп» и генерацию прерывания; 0 – анализ запрещен
31-16	-	Не используются. При чтении всегда ноль

Регистр управления CR доступен для записи по трем адресам (смещение относительно базы). При обращении по адресу 0 возможны чтение и загрузка регистра управления. При записи по адресу 5 возможна установка бит регистра управления, а при записи по адресу 6 – очистка бит. Установка или очистка бита происходит, если соответствующий ему бит загружаемого значения равен единице.

Бит I2C_ON управляет включением интерфейса. Запись «1» в данный бит приводит к разрешению работы интерфейса. Если необходимо вернуть интерфейс в исходное состояние, то данный бит нужно сбросить в 0.

Бит STA управляет формированием состояния «СТАРТ» на линиях обмена интерфейса. Установка данного бита приводит к изменениям на линиях SDA и SCL, которые соответствуют состоянию «СТАРТ». После формирования этого состояния на внешних линиях, интерфейс устанавливает флаг прерывания и записывает соответствующий код в регистр состояния. После этого интерфейс переходит в состояние «мастер». При работе в мультимастерной среде установка данного бита в «1» не гарантирует, что интерфейс захватит шину и станет мастером. Другой интерфейс может опередить, и тогда данный интерфейс будет находиться в состоянии ожидания готовности линий интерфейса для начала генерации состояния «СТАРТ». Если процессор получил запрос прерывания с кодом «start», необходимо сбросить бит STA до момента записи данных для передачи.

Бит STO управляет формированием состояния «СТОП» на линиях обмена интерфейса. Установка данного бита приводит к изменениям на линиях SDA и SCL, которые соответствуют состоянию «СТОП». После формирования этого состояния на внешних линиях интерфейс устанавливает флаг прерывания и записывает соответствующий код в регистр состояния, а также автоматически сбрасывает бит STO. Установка данного бита должна производиться, только если интерфейс работает в состоянии «мастер».

Бит AA выполняет две функции:

– во время обмена данными между мастером и подчиненным выполняется функция разрешения выдачи подтверждения (acknowledge);

– во время фазы приема адреса выполняются функции управления разрешением сравнения собственного адреса с принятым и выдачи бита подтверждения.

Бит SI является флагом прерывания, который может устанавливаться аппаратурой интерфейса. Значение бита «1» свидетельствует о том, что интерфейсом завершена очередная фаза обмена с кодом в регистре состояния и ожидается реакция со стороны пользователя.

Бит ABN_EN расширяет возможности интерфейса по анализу ситуаций сбоя при работе в мультимастерной системе. В такой системе есть риск захвата шины одновременно несколькими мастерами. В этом случае каждый из мастеров выполняет мониторинг передаваемой информации на линии и реальной информации с линии. В случае несовпадения данных, мастер автоматически отключается и устанавливает бит ABRT. Обязательная процедура анализа распространяется на передачу первого байта адреса. Если два мастера обращаются к различным устройствам, то один из них обязательно отключится. Если возможно обращение нескольких мастеров к одному устройству (адрес одинаковый), то необходимо сравнение передаваемых данных. Это сравнение является дополнительной опцией и включается установкой бита ABN_EN в «1». Сравниваются только данные при выдаче. Выдаваемые биты подтверждения при чтении не сравниваются.

Бит M10 может быть использован только в режиме подчиненного при условии, что обмен происходит с использованием расширенной 10-битовой адресации. Бит остается неизменным в течение всего обмена. Бит, установленный в «1», обязывает интерфейс рассматривать первый байт данных после стартового байта адреса, как дополнительный байт адреса, который будет сравниваться с регистром AXR. Только при их совпадении интерфейс перейдет в режим работы подчиненного и сформирует запрос прерывания к процессору. При сравнении второго байта адреса используется анализ бита направления передачи, заданного в первом принятом адресном байте. Сравнение выполняется, если бит указывает на запись. При выполнении операции чтения в режиме M10, мастер предварительно передает первый байт адреса и признак записи, затем передается второй байт адреса. После этого выполняется рестарт и передача первого байта адреса, но уже с признаком чтения. Подчиненный отслеживает тот факт, что чтению предшествовала процедура идентификации. В противном случае подчиненный не отвечает на первый байт адреса, хотя он и совпадает.

Бит TLOW_EN может быть использован при выполнении процедуры «restart». В этом случае после записи некоторого байта в подчиненное устройство, процессор установит бит STA и будет ожидать прерывания по событию «restart». В случае если скорость обмена по шине низкая, а реакция на прерывания у процессора быстрая, то может возникнуть проблема, что перед выполнением «restart» на линии SCL не выдержан низкий уровень требуемой длительности. Установка данного бита в «1» позволяет на аппаратном уровне гарантировать соблюдение данного временного параметра. Бит необходимо устанавливать только в случае реальной необходимости. Если процедура обработки прерывания тратит достаточное время на обработку прерывания (больше времени T_low), использование данного бита только замедлит генерацию события «restart».

Бит IRQ – запрос прерывания. Бит доступен для чтения и записи. Дополнительно бит устанавливается аппаратно при установке флагов SI, ABRT и SLVE. При обслуживании прерывания данный бит должен быть сброшен как можно раньше. Бит SI выполняет дополнительную функцию по удержанию линии SCL в нуле и поэтому должен быть сброшен только после обслуживания интерфейса.

24.1.2 Регистр состояния SR

Регистр отражает информацию о событии (код события), которое вызвало запрос прерывания к процессору. Описание разрядов регистра приведено в таблице 249.

Таблица 249 – Регистр SR

Биты	Значение	Описание
4:0	0x1F	Значение после сброса
	0x01	Мастер сформировал условие «stop». Интерфейс завершил работу в режиме мастера
	0x02	Мастер сформировал условие «start». Интерфейс начал работу в режиме мастера
	0x03	Мастер сформировал условие «restart». Интерфейс продолжает работу в режиме мастера
	0x04	Подчиненный обнаружил со стороны мастера обращение для записи по адресу 0. Режим широковещательной записи во все подчиненные устройства
	0x05	Подчиненный обнаружил условие «stop». Интерфейс закончил работу в режиме подчиненного
	0x09	Интерфейс, работая в состоянии «подчиненный», обнаружил на шине «свой» адрес, а также запрос на запись, после чего ответил битом подтверждения ACK = 1
	0x0B	Интерфейс, работая в состоянии «подчиненный», обнаружил на шине «свой» адрес, а также запрос на чтение, после чего ответил битом подтверждения ACK = 1
	0x0C	Мастер отослал первый байт, получил в ответ бит ACK=0 и ждет записи данных для передачи
	0x0D	Мастер отослал первый байт, получил в ответ бит ACK=1 и ждет записи данных для передачи
	0x0E	Мастер отослал первый байт, получил в ответ бит ACK=0 и ждет сброса флага прерывания для начала чтения данных из внешнего устройства
	0x0F	Мастер отослал первый байт, получил в ответ бит ACK=1 и ждет сброса флага прерывания для начала чтения данных из внешнего устройства
	0x10	«Подчиненный» принял байт данных и ответил битом подтверждения ACK = 0
	0x11	«Подчиненный» принял байт данных и ответил битом подтверждения ACK = 1
	0x12	«Подчиненный» отослал байт данных и принял бит подтверждения ACK = 0
	0x13	«Подчиненный» отослал байт данных и принял бит подтверждения ACK = 1
	0x14	Мастер отослал байт данных и принял бит ACK=0
	0x15	Мастер отослал байт данных и принял бит ACK=1
	0x16	Мастер принял байт данных и отослал бит ACK=0
	0x17	Мастер принял байт данных и отослал бит ACK=1
	0x1E	Шина свободна

Биты	Значение	Описание
5	-	Не используются. При чтении всегда ноль
6	ABRT	Бит 0 регистра управления. Информирует об аварийном (преждевременном) завершении работы мастера
7	SLVE	Бит 1 регистра управления. Информирует об обнаружении ситуации “stop” при работе подчиненного. Дополнительно в битах 4:0 устанавливается код 5
31:8	-	Резерв. При чтении всегда ноль

Значение регистра состояния устанавливается только аппаратно. Невозможно изменить это значение программно. Биты регистра доступны только по чтению. Если интерфейс выключен, значение регистра состояния равно 0x1F.

24.1.3 Регистр данных DR

Посредством этого регистра происходит запись данных для передачи и чтение данных после приема. Однако необходимо помнить, что запись новых данных в регистр возможна только тогда, когда значение регистра состояния указывает на ожидание этого действия. То же самое относится и к чтению данных. Регистр является непосредственно тем сдвиговым регистром, из которого на линию SDA передаются (старший бит первым) и принимаются (принятый бит поступает в младший бит регистра) данные.

24.1.4 Регистр адреса AR

Регистр содержит адрес устройства для работы в режиме подчиненного. При работе в расширенном режиме 10-битовой адресации регистр содержит два старших бита адреса и специальную кодовую комбинацию. Описание разрядов регистра приведено в таблице 250.

Таблица 250 – Регистр AR

Бит	Имя	Описание
0	AZE	Разрешение доступа к «подчиненному» по нулевому адресу (когда бит равен 1)
7:1	SA	Адрес устройства в режиме «подчиненный»
31:8	-	Резерв. При чтении всегда ноль

Биты SA регистра хранят адрес устройства при функционировании интерфейса I2C в режиме подчиненного. В этом случае обращение к процессору посредством I2C интерфейса возможно только при совпадении адреса в регистре AR с принятым адресом.

Бит AZE дает дополнительную возможность при работе в режиме подчиненного. Если бит равен единице, возможно обращение к интерфейсу по адресу равному нулю.

При работе в режиме 10-битовой адресации регистр должен хранить двоичный код 1111_0aaz.

Где биты «aa» это старшие биты 10-битового адреса. Младшие восемь бит адреса должны храниться в регистре AXR.

24.1.5 Регистр делителя частоты VR

Регистр коэффициента деления внутреннего делителя частоты. Формула для получения рабочей частоты была приведена выше. Разряды регистра описаны в таблице 251.

Таблица 251 – Регистр VR

Бит	Имя	Описание
11:0	DIV	Коэффициент деления для формирования частоты SCL
15:12	-	При чтении всегда 0
23:16	DSUP	Количество тактов предустановки бита данных относительно SCL. Реальное время предустановки будет равно DSUP+2
31:24	DHOLD	Количество тактов удержания бита данных относительно среза SCL. Реальное время предустановки будет равно DSUP+4 (для микросхем с ревизии 3)

24.1.6 Регистр состояния линий интерфейса PR

Регистр позволяет прочитать текущее состояние линий обмена интерфейса. Регистр доступен только по чтению. Описание разрядов регистра приведено в таблице 252.

Таблица 252 – Регистр PR

Бит	Имя	Описание
0	SDA_p	Значение вывода SDA
1	SCL_p	Значение вывода SCL
31:2	-	Всегда 0

24.1.7 Регистр адреса AXR

Регистр содержит младшие восемь бит адреса устройства для работы в режиме подчиненного и в режиме 10-битовой адресации. Доступны для чтения и записи только 8 младших бит регистра. Старшие биты при чтении всегда равны нулю.

24.1.8 Регистр INFO

Регистр доступен только для чтения и отражает состояние внутренней аппаратуры интерфейса. Биты могут быть полезны при проверке работы интерфейса, а также (возможно) и при реализации алгоритмов обмена. Описание разрядов регистра приведено в таблице 253.

Таблица 253 – Регистр INFO

Бит	Имя	Описание
0	I2C_ready	Линии SDA и SCL находятся в высоком уровне. 0 – на одной из линий интерфейса 0
1	SCLм	Логический уровень внутреннего сигнала синхронизации, который мастер выдает на линию SCL

Бит	Имя	Описание
2	SDA_mg	Значение линии SDA. Берется мажоритарно по трем последним отсчетам
3	SCL_mg	Значение линии SCL. Берется мажоритарно по трем последним отсчетам
4	Master	1 – интерфейс работает в режиме мастера
5	Slave	1 – интерфейс работает в режиме подчиненного
6	ST_BY	1 – интерфейс выполняет передачу (если мастер) или прием (если подчиненный) первого байта сообщения (адрес). Бит устанавливается в «1» при обнаружении ситуации «start». Сбрасывается в ноль после завершения работы с первым байтом
7	R_W	Бит указывает на направление передачи данных при обмене: 0 – в режиме мастера указывает на передачу данных; в режиме подчиненного – на прием; 1 – обратное направление передачи. Значение бита достоверно только после приема-передачи первого байта сообщения
11:8	BIC	Внутренний счетчик, указывающий на текущий номер бита передаваемого на линиях байта сообщения
12	BUSY	1 – интерфейс занят. Бит устанавливается событием «start» и сбрасывается событием «stop»
13	STX_BY	Бит устанавливается в «1» только при режиме работы с 10-битовой адресацией. 1 – интерфейс в режиме подчиненного выполняет прием второго байта сообщения (расширенный адрес). Бит устанавливается в «1» при обнаружении ситуации совпадения принятого адреса с регистром AR. Сбрасывается в ноль после завершения приема второго байта адреса
14	SLV_EXE	1 – интерфейс работает в режиме подчиненного. Отличие данного бита от флага Slave состоит в том, что данный флаг сбрасывается в ноль при выполнении ***чтения*** данных из подчиненного при получении бита подтверждения ACK=0. Флаг Slave сбросится только при получении ситуации «stop»
15	SLV_XM	Бит устанавливается в «1» только при режиме работы с 10-битовой адресацией. 1 – интерфейс в режиме подчиненного принял полный 10-битовый адрес и он совпал с адресом интерфейса. Бит сбрасывается в ноль при получении ситуации «stop» или когда бит M10 регистра управления равен нулю
16	M_EXE	1 – интерфейс работает в режиме мастера. Отличие от флага master состоит в том, что данный флаг не сбрасывается перед выполнением события «restart»
31:17	-	Всегда 0

24.2 Синхронизация интерфейса

Интерфейс использует для своей синхронизации частоту шины периферийных устройств. Для получения требуемых стандартных частот передачи на линии SCL используется программируемый делитель. Регистр VR задает коэффициент деления

частоты. Формула расчета была приедена ранее (см. стр. 365). Делитель частоты используется только при работе в режиме мастера и формирует требуемую частоту на линии SCL. Информация с линий SCL и SDA не используется интерфейсом напрямую, а защелкивается в буфере, тактируемым частотой периферийной шины. Имеется ограничение на минимальное значение коэффициента деления, он должен быть не менее трех. При значении меньше трех интерфейс в режиме мастера не работает. При минимальном значении итоговый коэффициент деления частоты равен 12. Таким образом, теоретически максимально достижимая частота обмена:

$$\mathrm{SCL\_max} = \frac{\mathrm{SCLK}}{12}. \qquad (10)$$

На практике достижение такой частоты обмена зависит от номинала внешнего резистора на линиях SDA и SCL. Корректная работа на максимально возможной частоте возможна, только если длительность фронта перехода из «0» в «1» на линиях SCL и SDA не превышает длительности периода частоты SCLK. В противном случае коэффициент деления необходимо увеличивать. Также имеет значение допустимая скорость обмена других устройств на шине. При необходимости подчиненное устройство должно успевать тормозить мастера.

При выдаче информации на линию SDA интерфейс гарантирует минимальное время удержания предыдущего значения данных после изменения линии SCL из «1» в «0». Для микросхем ревизии 2 это время равно четырем тактам частоты синхронизации периферийной шины SCLK. Для микросхем с ревизии 3 – исчисляется тактами частоты синхронизации периферийной шины SCLK и равно значению поля DHOLD регистра VR плюс 4.

Время предустановки данных на линии SDA до перехода линии SCL из «0» в «1» зависит от того, какой бит данных выдается:

– Если выдается *не первый бит*, время предустановки будет равно половине периода SCL минус четыре такта SCLK;

– Если выдается *первый бит*, процессор для выдачи данных сначала запишет байт данных в регистр данных DR для передачи. Через такт SCLK с момента записи эти данные появятся на линии SDA. Переход линии SCL из нуля в единицу после записи данных зависит от нескольких факторов. Например, при работе в режиме подчиненного линия SCL будет удерживаться подчиненным до момента очистки бита SI регистра управления. Если процессор оперативно среагировал на прерывание, быстро произвел запись новых данных и сбросил флаг SI, оставшееся время (с момента сброса SI до момента, когда мастер начнет выдавать «1» на линию SCL) будет составлять время предустановки данных. Если же процессор поздно среагировал на прерывание (или частота обмена высокая), и мастер уже готов выдать на SCL единицу, но подчиненный его удерживает, время предустановки будет равно количеству тактов от момента записи данных до момента сброса флага SI (минус 1). Например, если записать данные и сразу же за этим сбросить флаг, время предустановки будет равно нулю.

Возможны два способа управления временем предустановки первого бита данных:

– программный;
– аппаратный.

При программном способе рекомендуется после записи данных произвести чтение какого-нибудь регистра интерфейса, а затем сбросить флаг SI. Это даст как минимум пять тактов предустановки. Поскольку пользователь чаще всего не может знать, сколько времени прошло с момента генерации запроса прерывания, он всегда самостоятельно должен обеспечивать требуемое время предустановки первого выдаваемого бита. Например, стандарт I2C в типовом режиме работы требует минимальное время предустановки равное 250 нс. Если процессор работает на частоте 200 МГц, чтобы обеспечить требование стандарта, необходимо после записи данных для выдачи сбросить бит SI не ранее, чем через 250нс / 5нс = 50 тактов (или 25 тактов SCLK).

Возможен также аппаратный способ управления временем предустановки первого выдаваемого на линию SDA бита. Регистр VR имеет 8-разрядное поле DSUP. Если значение этого поля не равно нулю – включается механизм автоматического обеспечения времени предустановки. Интерфейс (в режиме мастер или подчиненный) отслеживает факт того, что флаг SI установлен и произведена запись в регистр данных для передачи. Затем интерфейс начинает обратный отсчет задержки, величина которой задана в поле DSUP. Если бит SI был сброшен раньше, чем истек интервал времени удержания, интерфейс притормозит линию SCL. Если бит SI будет сброшен после истечения заданного интервала времени, линия SCL будет освобождена сразу же после сброса SI. Данный механизм позволяет освободить пользователя от необходимости следить за соблюдением времени предустановки данных.

При работе в режиме подчиненного интерфейс после приема (или передачи) «своего» байта данных сформирует для процессора запрос прерывания и будет удерживать линию SCL в нуле. Линия будет удерживаться до тех пор, пока процессор не обслужит интерфейс и не сбросит бит SI. Отметим, что случайная установка пользователем бита SI при работе интерфейса в режиме подчиненного немедленно приведет к переводу линии SCL в ноль.

При работе в режиме мастера интерфейс самостоятельно задает синхронизацию на линии SCL. При этом программируемый делитель интерфейса постоянно отслеживает, не удерживает ли какое-либо из устройств линию SCL в низком уровне. В этом случае делитель останавливается. Таким образом, подчиненное устройство может притормаживать мастера при передаче любого бита сообщения.

24.3 Режимы работы интерфейса

Интерфейс может работать в различных режимах:

– мастер в одномастерной системе;
– мастер в мультимастерной системе;
– мастер в системах с семи или 10-битовой адресацией;
– подчиненный с режимом адресации семь бит;
– подчиненный с режимом адресации 10 бит.

Для любого режима работы перед началом использования блока необходимо соответствующим образом назначить выводы общего назначения на выводы SCL и SDA посредством регистра CFG1 (см. раздел 31 «Модуль управления синхронизацией и энергопотреблением»).

В каждой из данных ситуаций будет свой особенный алгоритм работы. Поскольку интерфейс предполагает тесное программное управление, часть задачи по выполнению обмена на шине ложится на процессор.

Рассмотрим возможные примеры алгоритмов работы в различных режимах. Действия интерфейса будут выделены курсивом. Предположим, что в регистре VR заранее запрограммирован коэффициент деления частоты для получения рабочей частоты интерфейса, поэтому данный регистр далее не берется во внимание.

24.3.1 Мастер в одномастерной системе с режимом адресации 7 бит. Запись

Действия пользователя для реализации процедуры обмена данными с некоторым устройством могут быть следующими:

1 Включение интерфейса посредством установки бита I2C_ON регистра управления. Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

2 Анализ того, что линии SCL и SDA имеют высокий уровень. Установка бита STA для генерации события “start” на линиях интерфейса. Ожидание прерывания.

Интерфейс выполняет процедуру генерации события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x02. Формируется запрос на прерывание к процессору.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло событие «start». Сброс бита STA.

Запись в регистр данных DR значения адреса устройства, к которому будет выполняться обращение. Бит 0 регистра должен быть равен нулю, что указывает на запись в устройство.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x0C или 0x0D. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

4 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=0, обнаружена ошибка, и устройства нет. Если АСК=1, устройство есть и можно передавать данные для записи.

Запись в регистр данных DR значения байта данных для передачи.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x14 или 0x15. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

5 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=0, устройство больше не может принять данные. Если АСК=1, устройство может принять следующие данные.

Предположим, что больше не нужно передавать данные. Производим установку бита STO для генерации события «stop».

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет генерацию события «stop». Устанавливается бит SI, а в регистр состояния записывается код 0x01. Формируется запрос на прерывание к процессору.

6 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Выключение интерфейса. Завершение алгоритма работы.

Каждый раз, когда устанавливается бит SI, линия синхронизации SCL удерживается в низком уровне. Как только бит SI сбрасывается, выдача синхросигнала возобновляется.

24.3.2 Мастер в одномастерной системе с режимом адресации 7 бит. Чтение

Действия пользователя для реализации процедуры обмена данными с некоторым устройством могут быть следующими:

1 Включение интерфейса посредством установки бита I2C_ON регистра управления. Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

2 Анализ того, что линии SCL и SDA имеют высокий уровень.

Установка бита STA для генерации события “start” на линиях интерфейса.

Ожидание прерывания.

Интерфейс выполняет процедуру генерации события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x02. Формируется запрос на прерывание к процессору.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло событие «start». Сброс бита STA.

Запись в регистр данных DR значения адреса устройства, к которому будет выполняться обращение. Бит 0 регистра должен быть равен единице, что указывает на чтение данных из устройства.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x0E или 0x0F. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

4 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит ACK=0, обнаружена ошибка, и устройства нет. Если ACK=1, устройство есть и можно читать данные из устройства.

Устанавливаем бит AA в регистре управления, если необходимо будет прочитать более, чем один байт. Иначе бит AA = 0.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет чтение восьми бит данных из устройства и отсылает в устройство бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x16 (AA=0) или 0x17(AA=1). Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

5 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит ACK=0, дальнейшее чтение данных не нужно. Если ACK=1, интерфейс должен принять следующий байт данных.

Читаем принятые данные из регистра DR.

Если необходимо еще прочитать данные – сбрасываем бит SI и переходим к ожиданию прерывания. Перед чтением последнего байта обязательно нужно очистить бит AA.

Предположим, что дальнейшее чтение данных не нужно. Производим установку бита STO для генерации события «stop».

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет генерацию события «stop». Устанавливается бит SI, а в регистр состояния записывается код 0x01. Формируется запрос на прерывание к процессору.

6 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Выключение интерфейса. Завершение алгоритма работы.

Каждый раз, когда устанавливается бит SI, линия синхронизации SCL удерживается в низком уровне. Как только бит SI сбрасывается, выдача синхросигнала возобновляется.

В ряде случаев при обмене с внешним устройством возникает необходимость сначала записать данные в устройство (к примеру, внутренний указатель адреса), а затем перейти к чтению данных. При этом между записью и последующим чтением не должно быть состояния «СТОП». Эта задача решается следующим образом. После записи последнего байта, когда интерфейс установил флаг прерывания SI, пользователь должен установить бит STA в регистре управления и сбросить флаг прерывания SI. Подобная последовательность действий приводит к возникновению ситуации «РЕСТАРТ» и после нее можно снова передать адрес устройства с битом направления соответствующим чтению данных.

24.3.3 Подчиненный с режимом адресации 7 бит. Прием данных

Действия пользователя для реализации процедуры работы процессора как подчиненного устройства на шине I2C могут быть следующими:

1 Программирование адреса устройства в регистре AR. Установка бита AA в регистре управления для разрешения выдачи подтверждения при обнаружении совпадения адреса.

Включение интерфейса посредством установки бита I2C_ON регистра управления.

Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

Ожидание прерывания.

Интерфейс ожидает возникновения события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает внутренний бит BUSY, ST_BY и ожидает приема первого адресного байта. После приема байта интерфейс выполняет сравнение принятого адреса с адресом в регистре AR. В случае совпадения, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x09. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

2 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло обращение к процессору со стороны мастера по записи.

Если необходимо принять только один байт, пользователь сбрасывает бит AA в ноль. Чтобы принять столько байт данных, сколько может передать мастер, бит AA должен оставаться всегда в 1.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру передать байт данных, после этого отвечает мастеру битом подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x10 или 0x11. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Чтение из регистра данных DR значения байта принятых данных.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру передать следующий байт данных.

Процесс может повторяться многократно.

Для прекращения передачи данных, пользователю необходимо сбросить бит AA, и после приема очередного байта интерфейс ответит ACK=0. Мастер в ответ должен сформировать событие «stop». Если мастер передал последний байт данных, событие «stop» генерируется независимо от значения бита ACK. Интерфейс в состоянии подчиненного не сопровождает наличие события “stop” генерацией прерывания процессору. Чтобы узнать о завершении обмена не по инициативе пользователя, следует при включении интерфейса установить бит SLVE_EN. Обмен завершен, если на линиях было событие «stop». В противном случае, узнать о завершении текущего обмена можно только при следующем обращении к подчиненному устройству. Чтобы точно знать момент завершения обмена при использовании бита SLVE_EN, пользователю необходимо после получения прерывания по событию «stop» обработать прерывание и очистить флаг SLVE.

Каждый раз, когда подчиненный устанавливает бит SI, линия синхронизации SCL удерживается в низком уровне. Как только бит SI сбрасывается, разрешение синхросигнала возобновляется. Рекомендуется стремиться к максимально быстрой реакции на прерывание, чтобы не удерживать линию SCL слишком долго. Так как длительное удерживание может быть расценено мастером, как отказ устройства, и повлечет сброс всей системы.

24.3.4 Подчиненный с режимом адресации 7 бит. Выдача данных

Действия пользователя для реализации процедуры работы процессора как подчиненного устройства на шине I2C могут быть следующими:

1 Программирование адреса устройства в регистре AR. Установка бита AA в регистре управления для разрешения выдачи подтверждения в случае обнаружения совпадения адреса.

Включение интерфейса посредством установки бита I2C_ON регистра управления.

Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

Ожидание прерывания.

Интерфейс ожидает возникновения события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает внутренний бит BUSY, ST_BY и ожидает приема первого адресного байта. После приема байта интерфейс выполняет сравнение принятого адреса с адресом в регистре AR. В случае совпадения, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x0B. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

2 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло обращение к процессору со стороны мастера для чтения данных. Пользователь должен знать, какой байт данных хочет прочитать мастер. Нужное значение записывается в регистр данных DR для передачи.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру прочитать байт данных. После приема байта, мастер отвечает подчиненному битом подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x12 или 0x13. Значение кода зависит от бита подтверждения. Если бит подтверждения равен 0, интерфейс сбрасывает внутренний флаг SLV_EXE и перестает контролировать выдачу данных на линию SDA. Это позволяет мастеру сформировать событие «stop». Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=1, процессор записывает в регистр данных DR следующий байт для передачи. Если бит АСК равен нулю, происходит завершение алгоритма. События «stop» можно не дожидаться.

Производим сброс бита SI регистра управления. Ожидание прерывания, если передача не завершена.

Интерфейс освобождает линию SCL и позволяет мастеру принять следующий байт данных (если чтение продолжается) или сформировать событие «stop».

24.3.5 Мастер в одномастерной системе с режимом адресации 10 бит. Запись

Данный режим работы идентичен базовому семибитному режиму. Отличие заключается в том, что за первым адресным байтом необходимо передать второй байт адреса. Для пользователя это будет некоторая модификация алгоритма, для аппаратной части интерфейса второй байт адреса будет выглядеть, как первый байт передаваемых данных.

24.3.6 Мастер в одномастерной системе с режимом адресации 10 бит. Чтение

Процедура чтения в 10-битовом режиме адресации имеет ряд отличий от стандартной процедуры. Также имеются некоторые особенности.

Например, необходимо прочитать два байта данных из устройства, использующего 10 бит адреса. Последовательность действий будет следующей:

1 Включение интерфейса посредством установки бита I2C_ON регистра управления. Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

2 Анализ того, что линии SCL и SDA имеют высокий уровень.

Установка бита STA для генерации события «start» на линиях интерфейса.

Ожидание прерывания.

Интерфейс выполняет процедуру генерации события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x02. Формируется запрос на прерывание к процессору.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло событие “start”. Сброс бита STA.

Запись в регистр данных DR значения адреса устройства, к которому будет выполняться обращение. Особенность адреса в том, что в старших 7-ми битах он содержит код 1111_0aa. Биты «aa» это два старших бита 10-разрядного адреса устройства. Бит направления передачи должен быть равен нулю, что указывает на запись в устройство. Это необходимо для того чтобы передать второй байт адреса.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x0C или 0x0D. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

4 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит ACK=0, обнаружена ошибка, и устройства нет. Если ACK=1, устройство есть и можно передавать второй байт адреса. Отметим, что бит ACK=1 могут выставлять несколько устройств использующих 10-битовую адресацию, т.к. старшие два бита адреса у многих устройств могут быть одинаковыми.

Запись в регистр данных DR значения второго байта адреса для передачи.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код

0x14 или 0x15. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

5 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=0, устройства с таким адресом нет и нужно прекратить обмен. Если АСК=1, устройство есть и с ним можно выполнять обмен данными.

Предположим, что получили АСК=1 и необходимо прочитать данные. Для этого нужно изменить направление передачи данных.

Установка бита STA для генерации события “restart” на линиях интерфейса.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет процедуру генерации события «restart» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x03. Формируется запрос на прерывание к процессору.

6 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло событие «restart». Сброс бита STA.

Запись в регистр данных DR значения первого байта адреса устройства, к которому будет выполняться обращение. Значение адреса должно быть тем же. Особенность в том, что бит направления передачи должен быть равен единице, что указывает на чтение из устройства. В данном случае должно ответить то устройство, для которого на предыдущих стадиях была выполнена процедура выбора.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет передачу восьми бит данных и принимает от устройства бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x0E или 0x0F. Значение кода зависит от бита подтверждения. Формируется запрос на прерывание к процессору.

7 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=0, обнаружена ошибка, и устройства нет. Если АСК=1, устройство есть и можно читать данные из устройства.

Устанавливаем бит AA в регистре управления, если необходимо будет прочитать более чем один байт. Иначе бит AA = 0. В примере необходимо прочитать два байта, поэтому AA=1.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет чтение восьми бит данных из устройства и отсылает в устройство бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x17. Формируется запрос на прерывание к процессору.

8 Получение запроса на прерывание. Сброс бита IRQ. Анализ кода состояния.

Читаем принятые данные из регистра DR.

Сбрасываем бит AA, т.к. следующий прочитанный байт должен быть последним.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет чтение восьми бит данных из устройства и отсылает в устройство бит подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x16. Формируется запрос на прерывание к процессору.

9 Получение запроса на прерывание. Сброс бита IRQ. Анализ кода состояния.

Читаем принятые данные из регистра DR. Больше данные приниматься не будут, т.к. передан бит ACK=0.

Производим установку бита STO для генерации события «stop».

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс выполняет генерацию события «stop». Устанавливается бит SI, а в регистр состояния записывается код 0x01. Формируется запрос на прерывание к процессору.

10 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Выключение интерфейса. Завершение алгоритма работы.

Каждый раз, когда устанавливается бит SI, линия синхронизации SCL удерживается в низком уровне. Как только бит SI сбрасывается, выдача синхросигнала возобновляется.

24.3.7 Подчиненный с режимом адресации 10 бит. Прием данных

В данном случае процессор на шине I2C – подчиненное устройство, к которому возможно обращение по адресу. Для идентификации устройства должен использоваться 10-битовый адрес. Алгоритм работы устройства, в случае записи в него одного байта данных, следующий:

1 Программирование адреса устройства в регистре AR. При использовании стандартной адресации в AR должен быть код 1111_0aa0, где биты «aa» это два старших бита нашего 10-битового адреса.

Запись в регистр AXR младших восьми бит адреса.

Установка бита M10 для задания режима работы с адресом 10 бит.

Установка бита AA в регистре управления для разрешения выдачи подтверждения в случае обнаружения совпадения адреса.

Установка бита SLVE_EN, если необходимо получить прерывание после завершения обмена при возникновении события «stop».

Включение интерфейса посредством установки бита I2C_ON регистра управления.

Разрешение обработки запроса прерывания от интерфейса в контроллере прерываний.

Ожидание прерывания.

Интерфейс ожидает возникновения события «start» на линиях SCL и SDA. Когда событие происходит, интерфейс устанавливает внутренний бит BUSY, ST_BY и ожидает приема первого адресного байта. После приема байта интерфейс выполняет сравнение принятого адреса с адресом в регистре AR. В случае совпадения первого байта адреса, интерфейс отвечает битом подтверждения ACK=1, устанавливает признак slave, а также устанавливает бит STX_BY, указывающий на то, что интерфейс ожидает второй байт адреса.

При получении следующего байта данных интерфейс проверяет принятый байт на совпадение с регистром AXR. Если адрес не совпадает, интерфейс переходит в исходное состояние. В случае совпадения, интерфейс устанавливает бит SI и в регистр состояния записывает код 0x09. Также устанавливается внутренний флаг SLV_XM. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

2 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло обращение к процессору со стороны мастера по записи.

В данном случае принимается только один байт, поэтому бит AA сбрасываем в ноль. Если нужно принять столько байт данных, сколько может передать мастер, то бит AA должен оставаться всегда в «1».

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру передать байт данных, после этого отвечает мастеру битом подтверждения (в нашем примере ACK=0). Устанавливается бит SI, а в регистр состояния записывается код 0x10. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле. Т.к. интерфейсом передан бит ACK=0, данные больше приниматься не будут. Интерфейс сбрасывает внутренний флаг SLV_EXE, хотя флаг slave будет оставаться в «1».

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Чтение из регистра данных DR значения байта принятых данных.

Производим сброс бита SI регистра управления.

Т.к. мастеру уже передана информация о завершении обмена, можно не анализировать другие события. Но если установлен бит SLVE_EN, можно узнать, когда мастер сформирует событие «stop». В этом случае необходимо перейти к ожиданию прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру завершить обмен. Мастер может пытаться передавать еще байты данных, однако интерфейс будет отвечать ACK=0, не будет генерировать прерывание к процессору и не будет тормозить SCL линию. Если бит SLVE_EN установлен, интерфейс будет ждать события «stop» и сформирует запрос на прерывание.

Можно отметить следующую особенность использования бита SLVE_EN. При работе в режиме подчиненного бит SLVE_EN лучше устанавливать в момент, когда обнаружено обращение к нужному устройству по записи. Даже если на прием байта последовал ответ ACK=1, мастер может прекратить обмен по своей инициативе. Тогда мы узнаем быстрее, что текущий обмен завершен. Также существует вероятность, что мастер прервет обмен в любой момент (даже не закончив передачи байта). Бит SLVE_EN можно сбросить в ноль после приема байта данных, на который последовал ответ ACK=0 (т.е. мастер проинформирован о завершении обмена). Если бит SLVE_EN используется для того, чтобы точно знать момент завершения обмена, то после получения прерывания по событию «stop», пользователь должен обработать прерывание и очистить флаг SLVE.

24.3.8 Подчиненный с режимом адресации 10 бит. Выдача данных

Начало процесса чтения данных из подчиненного, использующего для адресации 10 бит, аналогично началу процесса записи, описанному в предыдущем пункте. Интерфейс отследит обращение к нему со стороны мастера и сформирует для процессора запрос прерывания. Код состояния будет равен 0x09. Дальнейшие действия должны быть следующими:

1 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло обращение к процессору со стороны мастера по записи.

Если не известно, будет мастер записывать данные или считывать, то бит AA должен быть установлен в «1». Мастер может, как передать байт данных, так и сгенерировать событие «restart». Устанавливаем AA в «1».

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру выполнить следующие действия. При выполнении мастером операции чтения, он сформирует на линиях интерфейса событие «restart». Интерфейс сбросит флаги slave, SLV_EXE, но останется установленным флаг SLV_XM, что будет свидетельствовать о выполненной недавно процедуре выбора подчиненного. Интерфейс примет первый байт адреса, определит совпадение, установит, что бит направления указывает на чтение. При выполнении мастером операции чтения, интерфейс проверит бит SLV_XM и убедится, что процедура выбора предшествовала чтению. После этого интерфейс ответит ACK=1 и перейдет в состояние «подчиненый_чтение». Интерфейс устанавливает бит SI и в

регистр состояния записывает код 0x0B. Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

2 Получение запроса на прерывание. Сброс бита IRQ.

Анализ того, что произошло обращение к процессору со стороны мастера для чтения данных. Пользователю необходимо знать, какой байт данных хочет прочитать мастер. Нужное значение записывается в регистр данных DR для передачи.

Производим сброс бита SI регистра управления. Ожидание прерывания.

Интерфейс освобождает линию SCL и позволяет мастеру прочитать байт данных. После приема байта, мастер отвечает подчиненному битом подтверждения. Устанавливается бит SI, а в регистр состояния записывается код 0x12 или 0x13. Значение кода зависит от бита подтверждения. Если бит подтверждения равен 0, интерфейс сбрасывает внутренний флаг SLV_EXE и перестает контролировать выдачу данных на линию SDA. Это позволяет мастеру сформировать событие «stop». Формируется запрос на прерывание к процессору. Интерфейс удерживает линию SCL в нуле.

3 Получение запроса на прерывание. Сброс бита IRQ.

Анализ кода состояния. Если бит АСК=1, процессор записывает в регистр данных DR следующий байт для передачи. Если бит АСК равен нулю, происходит завершение алгоритма. События «stop» можно не дожидаться.

Производим сброс бита SI регистра управления. Ожидание прерывания, если передача не завершена.

Интерфейс освобождает линию SCL и позволяет мастеру принять следующий байт данных (если чтение продолжается) или сформировать событие «stop».

Нужно отметить, что мастер может прекратить работу в любой момент времени при записи данных в подчиненное устройство (мастер контролирует линии SCL и SDA, и может сформировать «stop» во время передачи бита данных). При чтении данных из подчиненного устройства линию SDA контролирует подчиненное устройство. Однако, если подчиненный выдает бит данных равный единице, мастер может сформировать «stop» в этом случае. Подчиненный все время должен быть готов, что при возможности мастер может преждевременно (аварийно) завершить обмен. Задача интерфейса, корректно обработать ситуацию, перейти в исходное состояние, не отвлекать процессор, если для него данные события не имеют значения. При желании контролировать аномальное завершение обмена пользователь должен устанавливать бит SLVE_EN. При использовании бита SLVE_EN после получения прерывания по событию «stop», пользователь должен обработать прерывание и очистить флаг SLVE.

24.3.9 Мастер в многомастерной системе

Стандарт обмена I2C допускает работу нескольких ведущих (мастеров) устройств на одной шине. Особенность разработки алгоритма обмена в данной ситуации состоит в необходимости учитывать непредсказуемое поведение интерфейса. Например, интерфейс может работать как мастер и в тоже время имеет адрес на шине и может работать как подчиненный. В этом случае пользователь, желая выполнить обмен с некоторым подчиненным устройством в режиме мастер, может установить бит STA, ожидать возникновения события “start”, но в ответ получить прерывание с кодом, который указывает на то, что к интерфейсу производится обращение как к подчиненному устройству.

При работе в мультимастерной системе установка бита STA не гарантирует возможность захватить шину и стать мастером. Но даже после установки бита STA получение от интерфейса прерывания с кодом события «start» не гарантирует, что в это же время другой мастер одновременно не сформировал аналогичное событие. Поэтому, став мастером и начав обмен, интерфейс во время передачи первого байта адреса всегда сравнивает то, что он выдает на линию SDA и то, что реально присутствует на линии. Сравнение происходит в момент фронта (из «0» в «1») линии SCL. Если выдаваемое значение и значение на линии не совпадают, интерфейс принимает решение об аварийном завершении и отключается от линий обмена. При этом бит ABRT устанавливается в «1», и формируется запрос прерывания к процессору. Пользователь должен проанализировать данный бит и принять решение о повторной попытке обмена. В данном случае имеет место попытка нескольких мастеров одновременно обратиться к устройствам с различными адресами.

В случае, когда несколько мастеров одновременно обращаются к одному устройству, все биты адреса и бит направления совпадут. Тогда определить факт одновременного наличия нескольких мастеров на шине можно только, сравнивая выдаваемые биты данных. Сравнение данных включается специальным битом ABN_EN. Если он установлен, интерфейс будет анализировать выдаваемые биты данных, и, в случае аномалий, произведет отключение мастера и установится бит ABRT. В случае чтения, данные выдает подчиненное устройство, а мастера могут выдавать бит подтверждения. Нарушение бита подтверждения не контролируется интерфейсом, однако, получив прерывание, пользователь сам может определить, какой бит он выдавал и какой получил с линии. *В данной ситуации обмена проблема может возникнуть, если два мастера читают разное количество данных из подчиненного устройства. Мастер, который первым прочитал нужное ему количество данных, ответит неактивным битом подтверждения (сбросит AA перед чтением последнего байта). Однако другой мастер ответит активным битом подтверждения. В этом случае первый мастер получит код прерывания, в котором будет противоположное значение бита подтверждения. Данный факт рассматривается как наличие второго мастера, и первый мастер должен освободить интерфейс. В этом случае, при сбросе бита SI необходимо сбросить и бит включения интерфейса. Это приведет интерфейс в исходное состояние. Если интерфейс не выключать, то он продолжит работу как мастер и будет*

принимать данные до тех пор, пока последний мастер на шине не сформирует АСК равный нулю. В принципе, и такой алгоритм можно рассматривать как рабочий.

При выдаче сигнала синхронизации SCL мастер всегда контролирует факт удержания подчиненным линии SCL в нуле. Поэтому, если два мастера начали одновременную работу, и их частоты не совпадают, итоговое значение линии SCL будет равно «проводному И» двух выдаваемых сигналов. Если факт задержки установки высокого уровня SCL это нормальное событие для мастера, то факт более ранней выдачи нуля на линии SCL может рассматриваться как присутствие другого мастера. В этом случае мастер, который раньше выдал на SCL ноль, может выключить других, более «медленных» мастеров. Произойдет это по следующей причине. Выдача новых данных на линию SDA привязана к реальному уровню на линии SCL, а сравнение данных внутри конкретного мастера привязано к его внутреннему клоку SCLm. Более быстрый мастер переключит линию SCL из «1» в «0», все мастера выдадут на линию новый бит данных. Но более медленный мастер будет по-прежнему сравнивать ранее выдаваемый бит с новым значением линии SDA. Поэтому, если при выдаче есть смена значения бита данных, то медленный мастер отключится, даже если оба мастера выдают одно и то же значение. Поскольку значение 0 и значение 0xFF адреса маловероятно, то разноскоростные мастера с большой долей вероятности обнаружат аномалию еще при выдаче адреса.

Для разрешения обнаружения конфликтов на линиях в момент генерации события «restart» используется бит ABN_S_EN. Конфликт может произойти, если два мастера одновременно обращаются к одному и тому же устройству, но один мастер для записи, а другой для чтения данных. В данном случае мастер, который пытается сгенерировать событие «restart», освобождает линии данных и синхронизации, что позволяет другому мастеру свободно осуществлять выдачу данных.

Для разрешения обнаружения конфликтов на линиях в момент генерации события «stop» используется бит ABN_P_EN. Конфликт может произойти, например, если два мастера одновременно обращаются к одному и тому же устройству по записи, но один мастер желает завершить обмен (не смотря на бит подтверждения подчиненного), а другой пытается продолжить обмен. В данном случае мастер, который пытается сгенерировать событие «stop», удерживает линию данных в нуле, но освобождает линию синхронизации. Здесь возможны два варианта:

1 Первому мастеру удалось сформировать событие «стоп» (если второй мастер выдавал бит данных 1), а второй мастер получит преждевременное прерывание по окончанию обмена.

2 Если второй мастер выдает бит данных 0, первый мастер отключается от шины, т.к. генерирует внутреннее событие “stop” и не проверяет, что такое же событие произошло на линиях, а второй мастер благополучно продолжает обмен.

Остаются проблемными ситуации, описанные в стандарте:

– Один мастер «restart», а другой выдает бит данных. В этом случае очень важна скорость обмена устройств и то, какой бит данных выдает мастер. Наиболее вероятен сценарий, при котором мастер, формирующий рестарт, обнаружит аномалию на шине и

отключится. Но не исключен вариант, что ошибку в выдаваемом бите данных обнаружит второй мастер.

– Один мастер «stop», а другой выдает бит данных. В данном случае возможны как преждевременная генерация события стоп для второго мастера, так и благополучное завершение операций для обоих мастеров.

– Один мастер «restart», а другой «stop». В данной ситуации сформируется ситуация стоп, т.к. линия данных будет удерживаться в нуле, что позволит первому мастеру обнаружить конфликт.

Если при работе в многомастерной системе возможны аномалии, которые приводят к переводу интерфейса в непредсказуемое состояние, необходимо использовать системный таймер в качестве генератора максимального интервала ожидания события от интерфейса. Если запрос от таймера придет раньше, чем ответ от интерфейса, ситуацию можно рассматривать как аномальное явление. После этого необходимо выключить интерфейс, произвести анализ состояния линий интерфейса, при необходимости сбросить подчиненные устройства и затем повторить попытку обмена. Возможный алгоритм поведения следующий:

1 Обнаружение потери управления шиной (бит ABRT установлен).

2 Установка бита FREE_EN и программирование таймера на заданный интервал времени.

3 Если первым пришел запрос прерывания от интерфейса и по флагам интерфейса видно, что обмен не был завершен корректно (отсутствовал «стоп»), генерация на линиях события «старт», а затем «стоп».

4 Если первым пришел запрос на прерывание от таймера, выполняется программный анализ линий интерфейса. Если линия SCL удерживается в нуле, выход только в аппаратном сбросе устройств. Если линия SDA удерживается в нуле, программная генерация синхросигнала на линии SCL, до момента, когда SDA равно единице. После этого генерация события «stop».

24.3.10 Одновременная работа в режиме мастера и подчиненного

Возможна ситуация, когда в многомастерной системе необходимо передавать сообщения другим мастерам, а также принимать от них сообщения. В этом случае интерфейс должен самостоятельно определить, в каком режиме он должен работать в текущий момент времени. Рассмотрим пример системы, в которой имеется несколько процессоров, все они подключены посредством своих аппаратных интерфейсов к общей I2C линии и периодически обмениваются сообщениями. При включении интерфейса не задается его режим работы. Если пользователь установил бит STA и интерфейс сформировал событие «start», то интерфейс будет работать в режиме мастера. Если же интерфейс получил с линий «чужое» событие “start”, то он – подчиненный. При этом у него должен быть запрограммирован адрес подчиненного устройства в соответствующих регистрах.

Если возникнет ситуация, когда одновременно несколько интерфейсов становятся на шине мастерами, интерфейс А (адрес подчиненного 0x20) может обращаться к интерфейсу Б (адрес подчиненного 0x22), а интерфейс Б к интерфейсу А. При передаче

адреса интерфейс Б обнаружит, что передан не его адрес и отключит режим мастера (будет установлено прерывание и флаг ABRT). Однако интерфейс Б продолжит принимать данные с шины и определит, что выполняется обращение к нему как к подчиненному устройству. В данном случае пользователь интерфейса Б, начав обмен по шине как мастер должен быть готов к тому, что он потеряет управление шиной и к нему может быть выполнено обращение как к подчиненному устройству. Если такая ситуация произошла, пользователь интерфейса Б должен обработать запрос подчиненного устройства и затем повторить попытку обмена.

Отметим, что данный вариант обмена допустим только при семибитовой адресации подчиненных устройств. При 10-битовом режиме адресации, в случае одновременного старта нескольких мастеров, они могут пропустить обращение к ним как к подчиненному устройству. При этом интерфейс мастера, который первым обнаружил аномалию на шине и отключился, может не смочь в данной транзакции ответить, как подчиненное устройство. Мастер, который обращается к интерфейсу с 10-битовой адресаций может не получить активный бит подтверждения. Этот факт должен рассматриваться лишь как аномальное событие на шине. Реакцией на такое событие должна быть повторная попытка обмена. Однако начало повторной попытки для обоих мастеров лучше всего сделать через случайно сгенерированный интервал времени. Это позволит избежать одновременного захвата шины несколькими интерфейсами.

Отметим, что подобной системе лучше всего программировать скорости интерфейсов, отличающиеся друг от друга на некоторую величину. В этом случае вероятность одновременного захвата шины будет очень низкой.

24.4 Некоторые особенности работы интерфейса

В данном подразделе будут рассмотрены некоторые моменты в реализации аппаратной части интерфейса и сформулировать рекомендации для разработки более надежных алгоритмов обмена.

24.4.1 Запрос прерывания

Прерывание от I2C интерфейса анализируется по уровню. Поэтому при обработке прерывания необходимо обязательно сбрасывать бит IRQ регистра управления. Нужно гарантировать надежный интервал времени между сбросом IRQ и выходом из обработчика прерывания, чтобы исключить возможность повторной генерации прерывания.

24.4.2 Включение интерфейса

Если процессор является единственным мастером на шине I2C, включение интерфейса не представляет собой никаких проблем. Однако если процессор подключен к шине, на которой находится несколько мастеров, включение интерфейса в работу может произойти во время обмена по шине. Аппаратура интерфейса, по возможности, обеспечивает исключение возникновения различных непредвиденных ситуаций. Хотя может понадобиться и анализ состояния линий интерфейса перед включением.

24.4.3 Бит STA

Данный бит устанавливается в «1» при необходимости сформировать на линиях интерфейса событие «start» или «restart». После получения прерывания, необходимо не забывать сбрасывать данный бит в ноль. Аппаратно данный бит не сбрасывается. После генерации события «start» или «restart» бит STA удерживает линию SDA в нуле. Чтобы обеспечить необходимое время предустановки данных, бит STA должен быть сброшен до момента записи данных для передачи. Обязательно бит STA должен быть сброшен до сброса флага SI. Если нужно сформировать событие “restart” (при установленном бите SI), необходимо убедиться, что бит STA сброшен. Затем установить бит STA и только после этого сбросить бит SI. Это гарантирует время предустановки линии SDA до фронта SCL. Возможно, для соблюдения параметра t_LOW линии SCL, необходимо будет использовать бит TLOW_EN.

Перед установкой бита STA проверка готовности линий интерфейса не обязательна. Аппаратура интерфейса самостоятельно анализирует готовность линий для генерации события.

24.4.4 Бит SLE_EN

Интерфейс, обнаружив «stop», выдаст запрос на прерывание путем установки бит IRQ и SLVE. После этого интерфейс переходит в состояние ожидания и может детектировать факт обращения к подчиненному до того, как процессор обработает предыдущее событие «stop». Если по каким-то причинам процессор не успел обработать событие «stop» от подчиненного до момента приема стартового адресного байта следующего обмена, то в коде состояние будет отражен факт приема стартового байта, но в тоже время будет присутствовать и установленный бит SLVE.

При предсказуемом поведении мастера и заранее оговоренном формате обмена, нет необходимости в использовании флага SLVE. При чтении из подчиненного АСК формирует мастер, и он обязан будет дать АСК = 0, чтобы информировать о завершении обмена.

Иначе подчиненный не отпустит линию SDA. В этом случае нет необходимости контролировать «стоп». При записи в типичных случаях всегда известна длина записываемых данных, и подчиненный может сам управлять окончанием обмена, выдавая АСК=0. Поэтому опять "стоп" контролировать не обязательно. В результате можно сделать вывод, что данный флаг и бит его разрешения можно использовать только для контроля каких-либо аномалий на шине обмена (если они возможны). Например, после активизации подчиненный выполняет прием или выдачу байта данных. Перед обменом байта устанавливаем SLV_EN. Байт данных передается-принимается корректно, и подчиненный обрабатывает событие. Если подчиненный видит, что этот байт был последний, он больше не контролирует шину и снимает SLV_EN (сбрасывая одновременно и другие флаги). Если вместо события об окончании приема-передачи данных будет получен флаг SLVE, рассматриваем это как аномальное завершение и отмечаем факт несостоявшегося обмена. Таким образом, бит SLVE (SLV_EN) выступает как бы предохранителем возможного непредсказуемого поведения мастера. Только в случае приема подчиненным неизвестного заранее объема блока данных, может

возникнуть потребность в контроле события "стоп", чтобы узнать, что блок данных передан.

24.4.5 Мастер. Делитель клока

Интерфейс имеет 12-разрядный счетчик DIVC, который используется для задания частоты синхронизации SCL при работе в режиме мастера. Счетчик задает четверть периода частоты синхронизации и ведет счет, начиная от величины DIV (регистр VR) до единицы. Затем счет повторяется. Каждый раз, когда счетчик перезагружается, специальный двухразрядный счетчик CNTC увеличивает свое значение на единицу. Старший бит счетчика CNTC и есть внутренний сигнал синхронизации SCLm, который выдается на линию SCL. Таким образом, в одном периоде SCL можно выделить четыре фазы t0, t1, t2, t3. Счетчик может останавливать свой счет при следующих условиях:

– Работая в режиме мастера и находясь в интервале t1, при значении DIVC равном двум интерфейс проверяет, установлен ли бит SI или истек ли период предустановки данных. Если бит SI установлен или период предустановки не закончился, интерфейс останавливает счет DIVC.

– Работая в режиме мастера и находясь в интервале t2 (SCLm стал равен единице), интерфейс через четыре такта после начала интервала проверяет значение линии SCL. Если линия в нуле, счетчик DIVC останавливается. Это означает, что подчиненное устройство удерживает мастера. Как только линия SCL перейдет в высокий уровень, счетчик возобновит счет.

– Работая в режиме мастера и находясь в интервале t2 (SCLm равен единице), интерфейс сканирует линию SCL, и как только на ней установится высокий уровень, интерфейс установит внутренний флаг POS_F. Установка данного флага означает, что линия SCL была переведена в высокий уровень. На счетчик DIVC не влияет ситуация, когда в течение интервалов t2 или t3 на линии будет короткий импульс SCL=0 (длительность не более периода SCLK). Но, если на линии SCL преждевременно (до окончания интервала t3) установится нулевой уровень, то счетчик DIVC перезагрузится, а счетчик CNTC сбросится в ноль. Счет продолжится.

24.4.6 Подключение на плате

Для устойчивой работы интерфейса требуется на выводе SCL поставить RC-фильтр. Номинал резистора 100 Ом, конденсатора – 30 ÷ 56 пФ. Такие номиналы позволяют работать до скорости обмена в 400 кГц.

24.5 Примеры программирования

В данном подразделе приведены несколько примеров программирования различных действий интерфейса при реализации алгоритмов обмена.

24.5.1 Мастер. Обработка события “start”

Предположим, что интерфейс включен: запрограммирована скорость обмена, установлен бит включения. Перед установкой бита STA пользователю следует записать в регистр векторов прерываний адрес обработки события «start», разрешить прерывание от

I2C, в некоторую структуру info_I2C записывается адрес устройства, к которому будет обращение, после этого установить бит STA. Далее можно выполнять другие задачи или ожидать прерывание. При наступлении прерывания выполниться процедура:

```
.align 4;
do_restart:     q[j27+4] = k3:0;
                k2 = k31 + base_I2C;;

                        q[j27+8] = j3:0;
                                k3 = 0x820;;
                        [k2+I2C_CRC] = k3;; // clear int & STA

                        j1 = [k2+I2C_SR];;  // get status
                        k0 = [k31 + info_I2C+0];; // device address
                        [k2+I2C_DR] = k0;;

                        comp(j1,2); k0 = k31+8;;      // check start code
                        if njeq, jump abnormal_start (NP); k1 = restart_service;;

                        [j31+base_INTV+26] = k1;;
                        [k2+I2C_CRC] = k0;   // clear SI

                        j3:0  = q[j27+8];;
                        rti (ABS) (NP); k3:0  = q[j27+4];;
```

В данной процедуре следует сохранить на стеке несколько регистров, загрузить в свободные регистры базовый адрес интерфейса и нужную информацию. Необходимо сбросить бит запроса прерывания и бит STA. Из структуры извлечь адрес устройства и записать его в регистр данных для передачи. Далее необходимо прочитать регистр состояния и убедиться, что произошло ожидаемое событие (что не является обязательным, т.к. других событий интерфейс сгенерировать не может). В регистр векторов прерываний следует записать адрес следующего обработчика, который будет обрабатывать факт передачи адреса и ответ устройства (адрес `service_Address`). После это необходимо сбросить бит SI и разрешаем интерфейсу продолжить работу. Далее следует восстановить регистры и выйти из обработчика.

Данный обработчик тратит 42 такта процессора от момента установки запроса прерывания в регистре управления, до момента сброса флага SI.

Если ожидается прерывание от I2C, а не выполняется иная работа, то вся необходимая информации (базовые адреса, нужные константы) может находиться в регистрах. В этом случае не нужно ничего сохранять на стеке, а затем восстанавливать, и время обработки может быть меньше. Например, следующая процедура:

```
.align 4;
do_xrestart:    [k2+I2C_CRC] = j2;; // clear int & STA
                j1 = [k2+I2C_SR];;   // get status
                        [k2+I2C_DR] = k0;  j2 = j31+8;; // write address
                        [j31+base_INTV+26] = k1;;// setup new vector
                        rti (ABS) (NP); [k2+I2C_CRC] = j2;; // clear SI
```

От момента установки запроса прерывания, до момента сброса флага SI, задержка 28 тактов. Особое внимание обратим на команду `j1 = [k2+I2C_SR]`. В данном случае нет необходимости в чтении регистра состояния, но можно использовать это чтение как задержку. Такая задержка нужна, чтобы гарантированно снялся запрос прерывания до момента выхода из прерывания. Если эту задержку убрать, будет сформирован ложный повторный запрос, т.к. процессор успеет выйти из прерывания до момента, когда еще не снят запрос прерывания (из-за более медленной шины периферийных устройств).

Запись нового вектора прерываний должна быть раньше, чем сброс флага SI. Иначе нет гарантии, что новое прерывание сработает по новому вектору.

25 Контроллер ARINC

Контроллер интерфейса ARINC содержит в своем составе восемь приемников и четыре передатчика по ГОСТ 18977-79 (далее ARINC).

Каждый приемник поддерживает функцию распознавания меток (или адресов). Для каждого приемника может быть запрограммировано до 256 меток. Помимо этого, фильтрация входных данных может осуществляться не только на базе меток, но и на базе двух бит Источник/Приемник.

Каждый передатчик поддерживает однонаправленную передачу 32-разрядных слов по двухпроводной витой паре, используя формат кодирования RZ. Доступна возможность запрограммировать 32-й бит как данные или как бит паритета. В случае формирования бита паритета, программируется его четность или нечетность.

Каждый приемник и передатчик использует собственный буфер FIFO для хранения данных. Размеры буфера FIFO варьируются от 32x32 до 256x32. Статус наполненности FIFO определяется на основе соответствующих бит статуса для каждого FIFO.

Контроллер поддерживает различные скорости приема и передачи данных. Для настройки скорости приема/передачи данных предусмотрено несколько делителей:

– делитель частоты SOCCLK (поле DIV[7:0] в регистрах RX_CR и CONTROL), который задает опорную частоту работы контроллера;

– делитель опорной частоты (бит CLK в регистрах RX_CR и CONTROL), который задает скорость приема/передачи данных.

При установке опорной частоты равной 1 МГц делитель опорной частоты реализует стандартные скорости приема/передачи данных – 100 кГц (делитель опорной частоты равен 10) и 12,5 кГц (делитель опорной частоты равен 80). Другие скорости приема/передачи данных могут быть заданы при установке опорной частоты, отличной от 1 МГц.

Контроллер позволяет обнаруживать ошибки в скорости приема/передачи данных, а также в паузах между сообщениями.

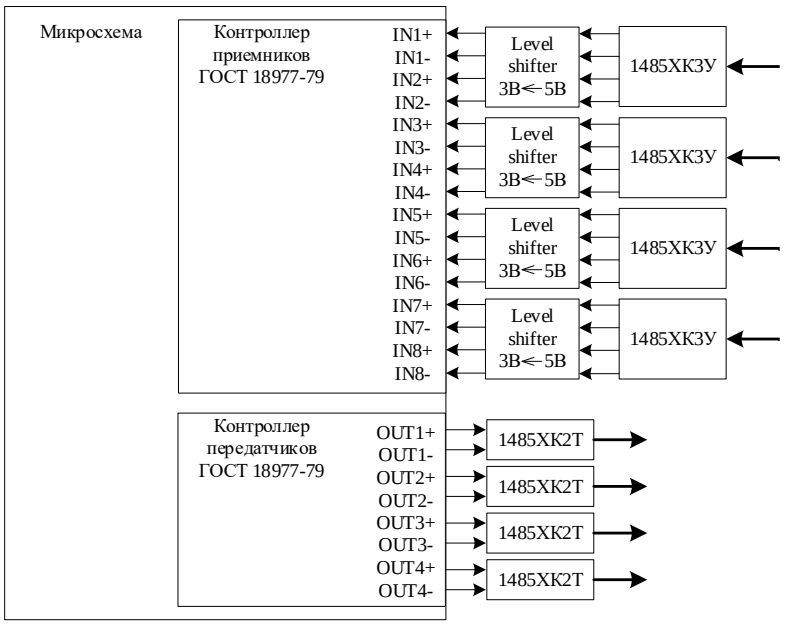


Рисунок 140 – Структурная схема контроллера интерфейса ARINC

Особенности:

– симплексный режим приема/передачи со скоростями 12,5 кГц или 100 кГц при установке опорной частоты равной 1 МГц;

– фильтрация входных данных на базе 256 меток и двух бит Источник/Приемник для каждого приемника;

– возможность передачи 32 бита, как данных, так и паритета;

– выбор четности/нечетности бита паритета;

– размеры буферов FIFO передатчиков: одно 256x32, три 64x32;

– размеры буферов FIFO приемников: два 256x32, четыре 64x32, два 32x32;

– возможность формирования прерываний при разных статусах наполненности буферов FIFO и при возникновении ошибок скорости передачи слова и паузы между словами;

– маскирование прерываний.

Назначение внешних выводов контроллера приведено в таблице 254.

Таблица 254 – Назначение внешних выводов контроллера

Обозначение вывода	Назначение вывода контроллера	Тип	Функциональное назначение
PB[4]	AR_IN1P	I/O	Интерфейс ARINC. Вход приемника 1 (+)
PB[5]	AR_IN1N	I/O	Интерфейс ARINC. Вход приемника 1 (-)
PB[6]	AR_IN2P	I/O	Интерфейс ARINC. Вход приемника 2 (+)
PB[7]	AR_IN2N	I/O	Интерфейс ARINC. Вход приемника 2 (-)
PB[8]	AR_IN3P	I/O	Интерфейс ARINC. Вход приемника 3 (+)
PB[9]	AR_IN3N	I/O	Интерфейс ARINC. Вход приемника 3 (-)
PB[10]	AR_IN4P	I/O	Интерфейс ARINC. Вход приемника 4 (+)
PB[11]	AR_IN4N	I/O	Интерфейс ARINC. Вход приемника 4 (-)
PB[12]	AR_IN5P	I/O	Интерфейс ARINC. Вход приемника 5 (+)
PB[13]	AR_IN5N	I/O	Интерфейс ARINC. Вход приемника 5 (-)
PB[14]	AR_IN6P	I/O	Интерфейс ARINC. Вход приемника 6 (+)
PB[15]	AR_IN6N	I/O	Интерфейс ARINC. Вход приемника 6 (-)
PB[16]	AR_IN7P	I/O	Интерфейс ARINC. Вход приемника 7 (+)
PB[17]	AR_IN7N	I/O	Интерфейс ARINC. Вход приемника 7 (-)
PB[18]	AR_IN8P	I/O	Интерфейс ARINC. Вход приемника 8 (+)
PB[19]	AR_IN8N	I/O	Интерфейс ARINC. Вход приемника 8 (-)
PB[20]	AR_OU1P	I/O	Интерфейс ARINC. Выход передатчика 1 (+)
PB[21]	AR_OU1N	I/O	Интерфейс ARINC. Выход передатчика 1 (-)
PB[26]	AR_OU2P	I/O	Интерфейс ARINC. Выход передатчика 2 (+)
PB[27]	AR_OU2N	I/O	Интерфейс ARINC. Выход передатчика 2 (-)
PB[28]	AR_OU3P	I/O	Интерфейс ARINC. Выход передатчика 3 (+)
PB[29]	AR_OU3N	I/O	Интерфейс ARINC. Выход передатчика 3 (-)
PB[30]	AR_OU4P	I/O	Интерфейс ARINC. Выход передатчика 4 (+)
PB[31]	AR_OU4N	I/O	Интерфейс ARINC. Выход передатчика 4 (-)

25.1 Формат слова

Слова в интерфейсе ARINC всегда 32-разрядные, и включают в себя пять полей: паритет, SSM, данные, источник/приемник, метка. Биты передаются младшими разрядами вперед, за исключением метки, которая передается старшими разрядами вперед. В результате можно описать порядок следования бит по шине ARINC следующим образом:

8, 7, 6, 5, 4, 3, 2, 1, 9, 10, 11, 12, 13…32.

32	31	30	29	11	10	9	8	1
P	SSM		DATA		SDI		LABEL	
			MSB	LSB				

Старший разряд всегда бит паритета. Стандартом установлено, что бит паритета должен дополнять слово до нечетного. Таким образом, количество единиц в 32-разрядном слове должно быть нечетным. Например, если биты 1-31 содержат четное количество единиц, бит паритета должен быть установлен в единицу, с другой стороны, если биты 1-31 содержат нечетное количество единиц, то бит паритета должен быть сброшен в ноль.

Биты 31 и 30 содержат знак или статус. В контроллере эти биты рассматриваются как обычные данные и помещаются в FIFO вместе с полем данных без изменений и дополнительной обработки.

Как пример биты 31 и 30 могут кодировать следующие характеристики, представленные в таблице 255.

Таблица 255 – Биты 31 и 30

Бит		Значение
31	30	
0	0	Плюс, север, восток, справа, к, выше
0	1	Невычислительные данные
1	0	Функциональный тест
1	1	Минус, юг, запад, слева, от, ниже

Биты 10 и 9 позволяют распознать Источник/Приемник данных. Это применяется при нескольких приемниках на шине ARINC, чтобы определить, для кого из них предназначаются данные. В системе со сложной структурой эти биты могут также использоваться, чтобы определить источник передачи. В остальных случаях эти разряды используются как данные. Следует отметить, что в интерфейсе ARINC на одной витой паре может быть один передатчик и до 20 приемников. Если включена функция проверки этих бит, при их несовпадении с битами, заданными программно в контроллере, сообщение не будет помещено в FIFO.

Биты с первого по восьмой позволяют идентифицировать тип данных оставшейся части слова, следовательно, методы преобразования, применяемые к данным. Помимо этого, в контроллере метки используются для фильтрации входных данных, то есть если метка в принятом сообщении не соответствует ни одной из меток определенной в памяти меток приемного канала, то данные не помещаются в FIFO. Это может служить аналогом

того, что приемник не может интерпретировать метод обработки этих данных, следовательно, эти данные предназначены для другого приемника.

В случае если приемник принимает данные с неправильным битом паритета, они также не будут помещены в FIFO.

25.2 Структурная схема канала приема

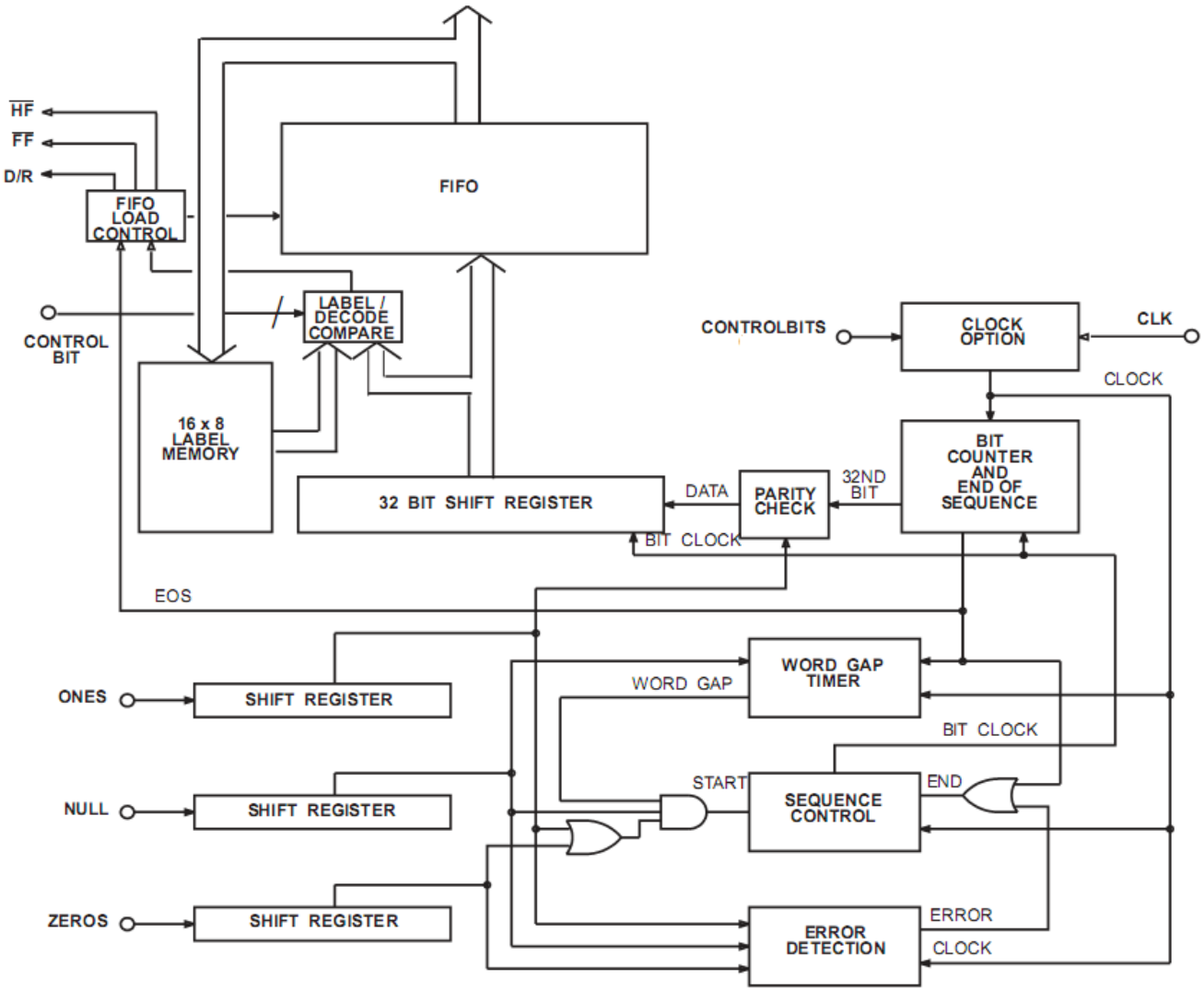


Рисунок 141 – Структурная схема канала приема

Если канал приемника работает на опорной частоте 1 МГц, то ошибка обнаружения бита в линии не будет составлять более 0,1 %.

Сдвиговые регистры длиной 10 бит, предназначенные для обнаружения в линии трех последовательностей единиц (Ones), нулей (Zeros) и отсутствие сигнала (Null), позволяют считать данные действительными. В дополнении к этому для бит данных, One или Zero в верхних битах сдвигового регистра должны сопровождаться Null в нижних битах в переделах битового интервала. В пределах паузы между сообщениями, три последовательных бита Null должны быть сэмплированы в верхней и нижней части сдвигового регистра Null. В этом случае гарантируется минимальная ширина импульса данных.

Каждый бит данных должен быть обнаружен в пределах от восьми до 12 сэмплов. В этом случае скорость передачи считается верной.

Таймер паузы между сообщениями сэмплирует сдвиговый регистр Null каждые 10 тактов опорной частоты при CLK = 0 (или 80 тактов опорной частоты при CLK = 1) после последнего полученного бита данных. Если Null обнаружен, таймер инкрементируется. Значение таймера равное трем разрешает следующий прием.

Схема паритета считает количество принятых единиц, включая бит паритета. Если результат нечетный, на выходе схемы формируется сигнал равный нулю.

После того как приняты все 32 бита логика приемника формирует сигнал конец последовательности (EOS). В зависимости от состояния бит LB_EN, SD_EN, SDI1, SDI2 регистра управления принимается решение о загрузке принятых данных в FIFO. Если в принятом слове биты 9 и 10 не соответствует правилам или не совпала метка, слово не загружается в FIFO. Случаи, в которых происходит загрузка FIFO принятыми данными, перечислены в таблице 256.

Таблица 256 – Загрузка FIFO принятыми данными

LB_EN	Результат сравнения слова ARINC с меткой	SD_EN	Результат сравнения бит 9,10 слова ARINC с SDI1, SDI2	FIFO
0	X	0	X	Загружается
1	не совпала	0	X	Игнорируются
1	совпала	0	X	Загружается
0	X	1	не совпали	Игнорируются
0	X	1	совпали	Загружается
1	совпала	1	не совпали	Игнорируются
1	не совпала	1	совпали	Игнорируются
1	не совпала	1	не совпали	Игнорируются
1	совпала	1	совпали	Загружается

Если хотя бы одно слово загружено в FIFO, устанавливается в единицу сигнал DR, что отражается в регистре статуса контроллера. Флаг остается в неизменном состоянии, пока последнее слово не будет прочитано из FIFO и оно не будет пустым. Помимо этого, применяются еще два сигнала характеризующие состояние FIFO, а именно HF означает, что FIFO наполовину полно и FF означает, что FIFO полно. Установка этих сигналов также отражается в регистре статуса. Каждый из этих флагов может быть источником прерывания, в случае если оно разрешено соответствующим битом маскирования регистра управления.

25.3 Структурная схема канала передачи

Если флаг TX_R в состоянии логической единицы, это значит, что FIFO пусто и в него могут быть загружены 31- или 32-битные данные. Количество слов данных определяется размером FIFO для выбранного канала передачи. Если флаг TX_R в состоянии логического нуля, тогда только в доступные в FIFO ячейки можно загрузить данные. Если FIFO заполнено полностью, флаг FFT установлен в единицу, то FIFO

игнорирует дальнейшие попытки записи в него. FIFO наполовину пусто, если установлен флаг HFT, в этом случае можно загрузить данными оставшуюся половину буфера FIFO.

В нормальном режиме работы 32 бит передаваемых данных является битом паритета. Четность или нечетность выбирается битом ODD регистра управления. Если бит разрешения паритета (EN_PAR) сброшен в ноль, тогда 32 бит передается, как бит данных из FIFO.

Если бит CH_EN установлен в единицу и FIFO передачи не пусто, начинается передача слов данных из FIFO до тех пор, пока FIFO не будет пусто или не будет сброшен бит CH_EN.

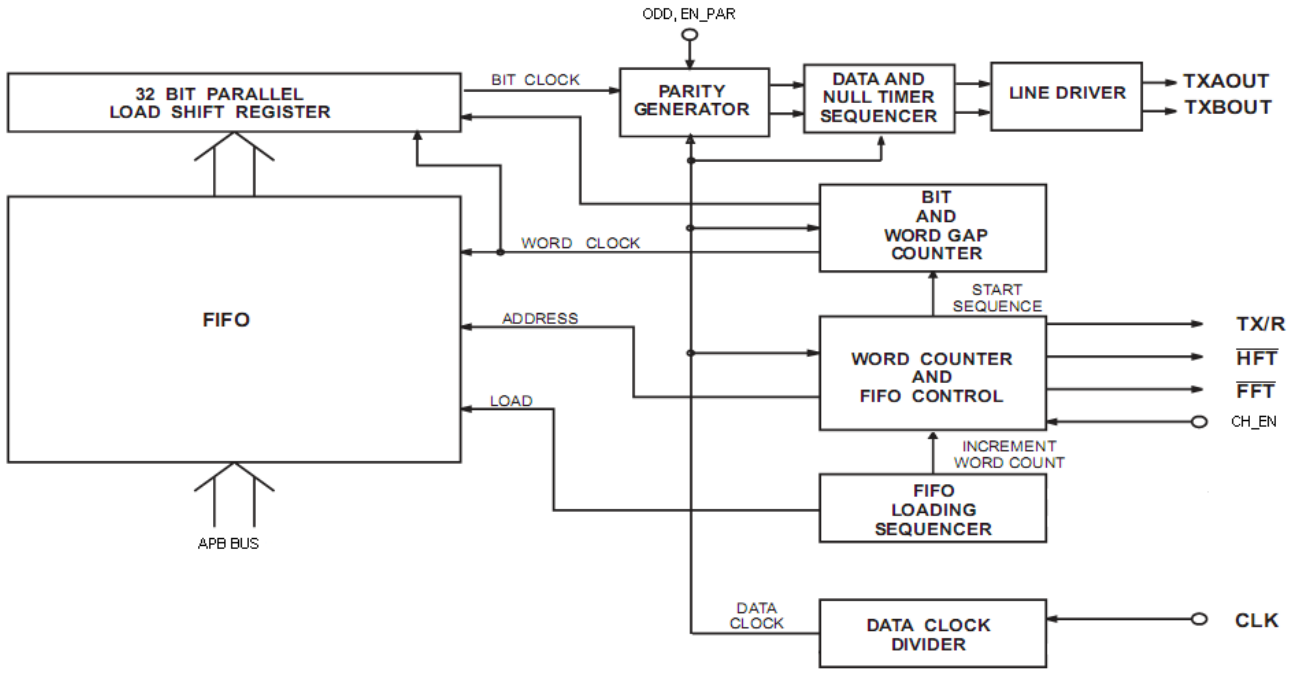


Рисунок 142 – Структурная схема канала передачи

25.4 Описание регистров приемника

Таблица 257 – Регистры приемника

Базовый адрес	Номер	Название	Описание
0x8000_2000		ARINC429R	
+0			Канал 0
	0	RX_CR	Регистр управления приемника
	1	-	-
	2	DATA_R	FIFO принимаемых данных
	3	STATUS	Регистр состояния приемника
+4			Канал 1
+8			Канал 2
+12			Канал 3
+16			Канал 4
+20			Канал 5
+24			Канал 6
+28			Канал 7

Базовый адрес	Номер	Название	Описание
0x8000_2400		Память приемников	Прямой доступ в память приемников
	0x0000-0x00FF	FIFO_D0	Буфер приемника канала 0
	0x0100-0x01FF	FIFO_D1	Буфер приемника канала 1
	0x0200-0x023F	FIFO_D2	Буфер приемника канала 2
	0x0240-0x027F	FIFO_D3	Буфер приемника канала 3
	0x0280-0x02BF	FIFO_D4	Буфер приемника канала 4
	0x02C0-0x02FF	FIFO_D5	Буфер приемника канала 5
	0x0300-0x031F	FIFO_D6	Буфер приемника канала 6
	0x0320-0x033F	FIFO_D7	Буфер приемника канала 7
	0x0340-0x03BF	-	-
	0x03C0-0x03FF	LABELS	Буфер меток. Для каждого канала 0-7 используется восемь 32-разрядных слов, образующих одно 256 битовое кодовое слово метки. Каждый бит соответствует одному из 256 возможных значений метки. Номер слова – три младших бита адреса. Номер канала – биты 5..3 адреса

Доступ к буферу данных возможен как прямым, так и косвенным путем (через указатель чтения). Доступ к меткам только прямым путем.

25.5 Регистр управления приемника RX_CR

Таблица 258 – Биты регистра управления приемника RX_CR

Бит	Имя	Значение	Описание
0	CH_EN		Разрешение работы канала: 1 – прием по каналу разрешен; 0 – канал приема находится в состоянии сброса
1	CLK		Скорость приема данных: 1 – частота приема данных = опорная частота/80 (12,5 кГц при опорной частоте 1 МГц); 0 – частота приема данных = опорная частота/10 (100 кГц при опорной частоте 1 МГц)
2	LB_EN		Разрешение обнаружения меток: 1 – разрешено обнаружение меток в первых 8 принятых битах; 0 – обнаружение отключено, все принятые данные помещаются в FIFO
3	SD_EN		Разрешение декодирования бит данных 9 и 10: 1 – разрешено сравнение бит данных 9 и 10 со значением бит SDI1 и SDI2 соответствующего канала; 0 – декодирование отключено, все принятые данные помещаются в FIFO

Бит	Имя	Значение	Описание
4	DA		Бит прямого доступа: 1 – память приема канала работает не в режиме FIFO, доступ к ней осуществляется в диапазоне адресов в зависимости от номера канала; 0 – обычный режим работы FIFO. При приеме данных из канала занесение их в память происходит в соответствии с адресом в первых восьми битах сообщения
5	SDI1		Бит сравнения SDI1. Значение бита сравнивается с битом 9 принимаемых данных, если установлен бит SD_EN соответствующего канала
6	SDI2		Бит сравнения SDI2. Значение бита сравнивается с битом 10 принимаемых данных, если установлен бит SD_EN соответствующего канала
7	ENSYNC		Разрешение работы входов приемника в режиме данных и синхросигнала: 1- разрешено; 0 – запрещено. При установленном бите ENSYNC для соответствующего канала вход IN_A работает как данные (D), вход IN_B как синхросигнал (SYN)
8	ENPAR		Разрешение 32 бита паритета для канала: 1 – разрешена передача 32-м битом бита паритета; 0 – разрешена передача 32-м битом бита данных. Запрещено сбрасывать этот бит в ноль в штатном режиме работы контроллера
9	ODD		Выбор четности или нечетности бита паритета: 1 – бит паритета формируется как дополнение до нечетности (если сумма всех разрядов данных по модулю 2 равно нулю, бит паритета устанавливается в «1», в противном случае в «0»); 0 – бит паритета формируется как дополнение до четности (если сумма всех разрядов данных по модулю 2 равна единице, бит паритета устанавливается в «1», в противном случае в «0»)
11, 10	-		
12	INTEDR		Разрешение прерывания наличие данных в FIFO: 1 – разрешено прерывание, если FIFO приема данных не пусто; 0 – прерывание запрещено

Бит	Имя	Значение	Описание
13	INTEER		Разрешение прерывания ошибка приема: 1 – разрешено прерывания при возникновении ошибки в скорости приема или во времени паузы 4T между сообщениями (для сброса ошибки необходимо сбросить канал битом CH_EN); 0 – прерывание запрещено
14	INTEFF		Разрешение прерывания FIFO полно: 1 – разрешено прерывание при полном заполнении FIFO данных; 0 – прерывание запрещено
15	INTEHF		Разрешение прерывания FIFO наполовину полно 1 – разрешено прерывание, если FIFO наполовину или более полно; 0 – прерывание запрещено
23..16	DIV[7:0]		Делитель частоты SOCCLK, формирующий опорную частоту. Содержит значение, на которое необходимо поделить частоту SOCCLK, чтобы получить опорную частоту. Значение частоты не может быть более 255 МГц
31:24			

25.5.1 Регистр состояния приемника STATUS

Таблица 259 – Биты регистра состояния приемника STATUS

Бит	Имя	Значение	Описание
0	DR		Бит наличия данных в FIFO: 0 – FIFO пусто; 1 – FIFO содержит данные
1	ERR		Бит ошибки: 0 – нет ошибок; 1 – возникла ошибка приема
2	FF		Бит полноты FIFO: 0 – FIFO не полно; 1 – FIFO полно
3	HF		Бит наполненности FIFO: 0 – FIFO не наполнено до половины; 1 – FIFO заполнено на половину и более
4	IRQ_DR		Запрос прерывания при наличии данных в FIFO: 0 – нет запроса; 1 – есть запрос
5	IRQ_ERR		Запрос прерывания при наличии ошибки: 0 – нет запроса; 1 – есть запрос
6	IRQ_FF		Запрос прерывания при полном заполнении FIFO: 0 – нет запроса; 1 – есть запрос

Бит	Имя	Значение	Описание
7	IRQ_HF		Запрос прерывания при заполнении FIFO наполовину или более: 0 – нет запроса; 1 – есть запрос
8	IRQ_RX_CH0		Запрос прерывания от канала 0: 0 – нет запроса; 1 – есть запрос
9	IRQ_RX_CH1		Запрос прерывания от канала 1: 0 – нет запроса; 1 – есть запрос
10	IRQ_RX_CH2		Запрос прерывания от канала 2: 0 – нет запроса; 1 – есть запрос
11	IRQ_RX_CH3		Запрос прерывания от канала 3: 0 – нет запроса; 1 – есть запрос
12	IRQ_RX_CH4		Запрос прерывания от канала 4: 0 – нет запроса; 1 – есть запрос
13	IRQ_RX_CH5		Запрос прерывания от канала 5: 0 – нет запроса; 1 – есть запрос
14	IRQ_RX_CH6		Запрос прерывания от канала 6: 0 – нет запроса; 1 – есть запрос
15	IRQ_RX_CH7		Запрос прерывания от канала 7: 0 – нет запроса; 1 – есть запрос
23..16	CHAN_WP		Указатель на запись данных
31..24	DATA_RP		Указатель на чтение данных

Все приемники имеют один вход запроса прерывания в контроллере прерываний. Дополнительная информация о конкретном источнике прерываний может быть получена из регистра состояний.

25.5.2 LABEL

Буфер меток, с которыми сравниваются первые восемь принимаемых бит, если установлен LB_EN бит соответствующего канала. Размер буфера для каждого канала 256x1. Это означает, что каждой из 256 кодовых комбинаций метки поставлен в соответствие один бит буфера. Процесс сравнения выполняется путем чтения соответствующего бита сразу после приема кода метки. К моменту приема всего сообщения бит из буфера должен быть получен и, если его значение равно единице, это соответствует приему сообщения с заданной меткой.

25.5.3 DATA_R

FIFO принимаемых данных

В FIFO помещаются 32-разрядные данные, принимаемые из соответствующего канала. Размер FIFO для каждого канала разный:

- канал 1 – 256x32;
- канал 2 – 256x32;
- канал 3 – 64x32;
- канал 4 – 64x32;
- канал 5 – 64x32;
- канал 6 – 64x32;
- канал 7 – 32x32;
- канал 8 – 32x32.

Наличие или отсутствие данных в FIFO контролируется битами статуса DR, HF, FF соответствующего канала. При чтении данных указатель чтения автоматически инкрементируется. При записи данных в FIFO после приема указатель записи инкрементируется. Все FIFO каналов приема физически размещены в одном модуле синхронной статической памяти. Буфер каждого канала имеет смещение относительно начала памяти в соответствии с размером предыдущих буферов. Об этом нужно помнить при прямом доступе в память приемников.

Доступ к памяти

Все буферы приемников и меток расположены физически в одном модуле памяти. Поэтому при попытке обращения к буферу со стороны приемников и процессора может возникать конфликт. Он разрешается следующим образом. Если есть запрос от процессора, он имеет самый высокий приоритет и обслуживается первым. Для выбора обслуживаемого канала используется трехбитный счетчик, который циклически выбирает канал для обслуживания. Таким образом, в течение восьми тактов любой канал может быть обслужен. Запрос на доступ к памяти канала выставляется при приеме сообщения или при приеме метки. После приема сообщения у канала есть 32 такта времени ожидания на обслуживание. При приеме метки остается 22 такта ожидания. Под тактом понимается время приема одного бита сообщения. Поскольку частота шины при этом должна быть как минимум в 10 раз выше (минимальный делитель опорной частоты равен 10), то при работе на 1 МГц частоты шины (SOCCLK) мы имеем 220 тактов частоты шины, чтобы обслужить максимум 16 запросов (восемь сообщений и восемь меток). Приостановку обслуживания запросов может вызывать доступ к памяти со стороны процессора. Однако один цикл чтения памяти процессором занимает минимум три такта (в действительности больше), из которых только один есть обращение к памяти, а два других могут быть использованы для обслуживания каналов. Задержку обслуживания может вызвать обращение к памяти по записи со стороны процессора. Запись может выполняться каждый такт шины. Это нужно учитывать при работе с интерфейсом и не выполнять непрерывно записи в буфер приемника большого блока данных.

25.6 Описание регистров передатчика

Таблица 260 – Регистры передатчика

Базовый адрес	номер	Название	Состояние после сброса	Описание
0x8000_3000		ARINC429T0		Контроллер интерфейса передатчиков ARINC429
+0				Канал 0
	0	CONTROL		Регистр управления передатчика
	1	-		
	2	DATA_T		Регистр передаваемых данных
	3	STATUS		Регистр состояния передатчика
+4				Канал 1
+8				Канал 2
+12				Канал 3
0x8000_3400	0x000-0x1FF	Память передатчиков		Прямой доступ в память передатчиков
	0x000	TX0_base		Базовый адрес передатчика 0
	0x100	TX1_base		Базовый адрес передатчика 1
	0x140	TX2_base		Базовый адрес передатчика 2
	0x180	TX3_base		Базовый адрес передатчика 3
	0x1c0	FREE_base		Неиспользуемая область памяти

25.6.1 Регистр управления передатчиком CONTROL

Таблица 261 – Биты регистра управления передатчиком CONTROL

Бит	Имя	Значение	Описание
0	CH_EN		Разрешение работы канала: 1 – передача по каналу разрешена; 0 – канал передачи находится в состоянии сброса
1	CLK		Скорость передачи данных: 1 – частота передаваемых данных = опорная частота/80 (12,5 кГц при опорной частоте 1 МГц); 0 – частота передаваемых данных = опорная частота/10 (100 кГц при опорной частоте 1 МГц)
2	EN_PAR		Разрешение 32 бита паритета: 1 – разрешена передача 32-м битом бита паритета; 0 – разрешена передача 32-м битом бита данных
3	ODD		Выбор четности или нечетности бита паритета: 1 – бит паритета формируется как дополнение до нечетности (если сумма всех разрядов данных по модулю 2 равно нулю, бит паритета устанавливается в «1», в противном случае в «0»);

Бит	Имя	Значение	Описание
			0 – бит паритета формируется как дополнение до четности (если сумма всех разрядов данных по модулю 2 равна единице, бит паритета устанавливается в «1», в противном случае в «0»)
4	INTE_TXR		Разрешение прерывания при опустошении буфера FIFO: 1 – разрешено прерывание FIFO передачи данных пусто; 0 – прерывание запрещено
5	INTE_FFT		Разрешение прерывания при полном заполнении буфера FIFO: 1 – разрешено прерывание при полном заполнении FIFO данных; 0 – прерывание запрещено
6	INTE_HFT		Разрешение прерывания при заполнении наполовину или менее буфера FIFO: 1 – разрешено прерывание FIFO наполовину или менее полно; 0 – прерывание запрещено
7	ENSYNC		Разрешение работы выходов передатчика в режиме данных и синхросигнала: 1 – разрешено; 0 – запрещено. При установленном бите ENSYNC для соответствующего канала выход OUT_A работает как данные (D), выход OUT_B как синхросигнал (SYN)
15..8	DIV[7:0]		Делитель частоты SOCCLK, формирующий опорную частоту. Содержит значение, на которое необходимо поделить частоту SOCCLK, чтобы получить опорную частоту

25.6.2 Регистр состояния передатчика STATUS

Таблица 262 – Биты регистра состояния передатчика STATUS

Бит	Имя	Значение	Описание
0	TX_R		Флаг наличия данных в FIFO: 1 – FIFO пусто; 0 – FIFO содержит данные
1	BUSY		1 – передатчик выполняет передачу данных; 0 – передатчик в ожидании новых данных
2	FFT		Флаг полноты FIFO: 1 – FIFO полно; 0 – FIFO не полно
3	HFT		Флаг наполненности FIFO канала: 1 – FIFO наполнено до половины или менее; 0 – FIFO наполнено более чем на половину
4	IRQ_TXR		Запрос прерывания если FIFO пусто: 0 – нет запроса; 1 – есть запрос

Бит	Имя	Значение	Описание
5	-		
6	IRQ_FFT		Запрос прерывания при полном заполнении FIFO: 0 – нет запроса; 1 – есть запрос
7	IRQ_HF		Запрос прерывания при заполнении FIFO наполовину или менее: 0 – нет запроса; 1 – есть запрос
8	IRQ_TX_CH0		Запрос прерывания от канала 0: 0 – нет запроса; 1 – есть запрос
9	IRQ_TX_CH1		Запрос прерывания от канала 1: 0 – нет запроса; 1 – есть запрос
10	IRQ_TX_CH2		Запрос прерывания от канала 2: 0 – нет запроса; 1 – есть запрос
11	IRQ_TX_CH3		Запрос прерывания от канала 3: 0 – нет запроса; 1 – есть запрос
15...12	-		
23…16	DATA_WP		Значение указателя на запись данных
31…24	CHAN_RP		Значение указателя на чтение данных

Все передатчики имеют один вход запроса прерывания в контроллере прерываний. Дополнительная информация о конкретном источнике прерываний может быть получена из регистра состояний.

25.6.3 DATA_T

FIFO передаваемых данных

FIFO может содержать данные объемом:

– 256x32 для передачи по каналу 1;
– 64x32 для передачи по каналу 2;
– 64x32 для передачи по каналу 3;
– 64x32 для передачи по каналу 4.

Наличие или отсутствие данных в FIFO контролируется битами статуса TX_R, HFT, FFT.

FIFO доступно по чтению и записи, однако только чтение вызывает инкремент указателя чтения.

Все FIFO каналов передачи физически размещены в одном модуле синхронной статической памяти. Буфер каждого канала имеет смещение относительно начала памяти в соответствии с размером предыдущих буферов. Об этом нужно помнить при прямом доступе в память передатчиков.

26 Контроллер МКПД по ГОСТ Р 52070-2003

В микросхеме имеется два независимых контроллера МКПД по ГОСТ Р 52070-2003 (далее 1553), каждый из которых содержит необходимую логику и память для обработки и хранения командных слов и слов данных одного полного сообщения 1553. Каждый контроллер содержит два канала для приема/передачи сообщений 1553: основной и резервный. В один момент времени может работать только один из каналов – основной или резервный. Одновременная работа двух каналов не предусмотрена. Контроллер может работать как в режиме контролера шины, так и в режиме оконечного устройства. Для хранения входящих и исходящих командных и статусных слов, а также команд управления используются 16-разрядные регистры. Для хранения данных используется шестнадцатиразрядная память, в которой данные хранятся в области памяти, соответствующей субадресу командного слова. В каждом субадресе можно хранить только одно полное сообщение 1553. При передаче сообщения данные в память можно заносить как на «лету», так и до начала передачи. При приеме сообщения, данные можно считывать из памяти, как на «лету», так и после установки флага VALMESS.

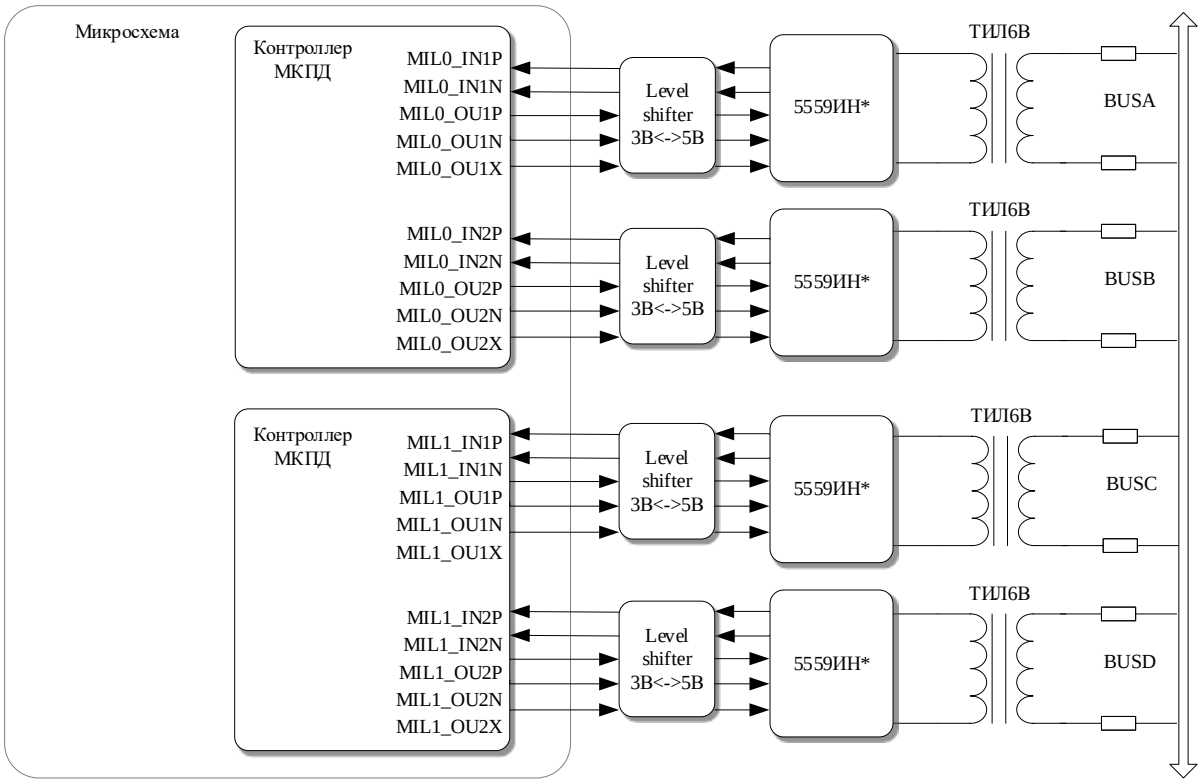


Рисунок 143 – Структурная схема контроллера интерфейса по ГОСТ Р 52070-2003

Особенности:

– поддержка основных (формат 1 – формат 6) и групповых (формат 7 – формат 10) форматов сообщений;

– поддержка режимов работы: контроллер шины, оконечное устройство, монитор;

– скорость передачи данных 1 Мбит/с в полудуплексном режиме;

– поддержка двух каналов связи: основного и резервного;

– двухпортовая память принимаемых данных 1К×16;

– двухпортовая память передаваемых данных 1К×16;

– возможность формирования прерываний при успешном приеме и при возникновении ошибок на шине;

– маскирование прерываний.

Назначение внешних выводов контроллера приведено в таблице 263.

Таблица 263 – Назначение внешних выводов контроллера (интерфейс 0, интерфейс 1)

Обозначение вывода	Назначение вывода	Тип вывода	Функциональное назначение
PA[26]	MIL0_OU1P	O	Интерфейс МКПД0. Выход основного канала (+)
PA[27]	MIL0_OU1N	O	Интерфейс МКПД0. Выход основного канала (-)
PA[28]	MIL0_OU1X	O	Интерфейс МКПД0. Разрешение передачи основного канала
PA[29]	MIL0_OU2P	O	Интерфейс МКПД0. Выход резервного канала (+)
PA[30]	MIL0_OU2N	O	Интерфейс МКПД0. Выход резервного канала (-)
PA[31]	MIL0_OU2X	O	Интерфейс МКПД0. Разрешение передачи резервного канала
PB[0]	MIL0_IN1P	I	Интерфейс МКПД0. Вход основного канала (+)
PB[1]	MIL0_IN1N	I	Интерфейс МКПД0. Вход основного канала (-)
PB[2]	MIL0_IN2P	I	Интерфейс МКПД0. Вход резервного канала (+)
PB[3]	MIL0_IN2N	I	Интерфейс МКПД0. Вход резервного канала (-)
PD[16]	MIL1_IN1P	I	Интерфейс МКПД1. Вход основного канала (+)
PD[17]	MIL1_IN1N	I	Интерфейс МКПД1. Вход основного канала (-)
PD[18]	MIL1_OU1P	O	Интерфейс МКПД1. Выход основного канала (+)
PD[19]	MIL1_OU1N	O	Интерфейс МКПД1. Выход основного канала (-)
PD[20]	MIL1_OU1X	O	Интерфейс МКПД1. Разрешение передачи основного канала
PD[21]	MIL1_IN2P	I	Интерфейс МКПД1. Вход резервного канала (+)
PD[22]	MIL1_IN2N	I	Интерфейс МКПД1. Вход резервного канала (-)
PD[23]	MIL1_OU2P	O	Интерфейс МКПД1. Выход резервного канала (+)
PD[24]	MIL1_OU2N	O	Интерфейс МКПД1. Выход резервного канала (-)
PD[25]	MIL1_OU2X	O	Интерфейс МКПД1. Разрешение передачи резервного канала

26.1 Режимы работы

Контроллер поддерживает три режима работы:

– контроллера шины (КШ);

– оконечного устройства (ОУ);

– неадресуемого монитора (М).

26.1.1 Контроллер шины

В этом режиме контроллер передает команды в магистраль, участвует в пересылке слов данных, принимает и контролирует ответную информацию о состоянии ОУ. Помимо этого, КШ реализует все команды управления. Для того чтобы реализовать передачу командного слова в магистраль, используется регистр CommandWord1. А для сообщений формата 3 и 8, помимо этого, применяется регистр CommandWord2. Ответная информация о состоянии ОУ после приема из магистрали хранится в регистре StatusWord1. А для сообщений формата 3 и 8, помимо этого, применяется регистр StatusWord2. Для передачи и приема слов данных команд управления (КУ), форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой бита BCMODE и сбросом RTMODE.

26.1.2 Оконечное устройство

В этом режиме контроллер осуществляет проверку достоверности командных слов, поступающих к нему от КШ. Командное слово считается достоверным, если не возникло ошибок в магистрали при его приеме, или если поле «Адрес ОУ» соответствует коду собственного адреса ОУ или коду 11111 (групповая команда). Если командное слово определено как достоверное, то ОУ посылает в линию ответное слово (ОС) и в зависимости от поля «Прием/Передача» принимает или передает число данных, соответствующее полю «Число СД/Код КУ». Если же происходит прием от КШ команды управления, то ОУ реагирует в соответствии с форматами сообщений команд управления. Принятое из магистрали командное слово помещается в регистр CommandWord1, а для сообщений формата 3 и 8 принятое второе командное слово помещается в регистр CommandWord2. Ответное слово ОУ для передачи в магистраль помещается в регистр StatusWord1. Помимо этого, для сообщения формата 3, этот регистр содержит принятое ответное слово от другого ОУ. Для передачи и приема слов данных команд управления, форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой бита RTMODE и сбросом BCMODE.

26.1.3 Монитор

В этом режиме осуществляется прослушивание магистрали и отбор необходимой информации для проведения: технического обслуживания, регистрации эксплуатационных параметров, анализа решаемых задач или обеспечения информацией резервного КШ. Монитор пассивно прослушивает выбранную шину и захватывает весь трафик на шине, но никогда не передает информацию на шину. Принятое из магистрали командное слово помещается в регистр CommandWord1, а для сообщений формата 3 и 8 принятое второе командное слово помещается в регистр CommandWord2. Ответное слово ОУ, принятое из магистрали, помещается в регистр StatusWord1. А для сообщений формата 3 и 8, помимо этого, применяется регистр StatusWord2. Для приема слов данных команд управления, форматы сообщений 5, 6 и 10, применяется регистр ModeData. Выбор этого режима работы осуществляется в регистре CONTROL установкой битов RTMODE и BCMODE. Для быстрой расшифровки сообщений можно применить регистр MSG.

Каждому формату сообщения на магистрали соответствует определенный код в этом регистре.

26.2 Форматы сообщений

Сообщения, передаваемые по информационной магистрали, имеют формат, соответствующий форматам основных или групповых сообщений. Любые другие типы сообщений, не соответствующие ГОСТ Р 52070-2003, не поддерживаются.

Форматы основных сообщений, приведенные на рисунке 144, используются для передачи информации, предназначенной одному ОУ, и предусматривают выдачу ОС. В данном случае КС – командное слово, СД – слово данных, ОС – ответное слово. Времена t_1 и t_2 формируются аппаратно и не могут быть изменены программно. Пауза t_2 между сообщениями, формируемая КШ, – не менее 4 мкс, а пауза перед передачей ОС, формируемая ОУ, – в пределах от 4 до 12 мкс. Если после ожидания 14 мкс так и не поступило ОС от ОУ, то фиксируется отсутствие ОС от ОУ, и формируется соответствующий признак ошибки.

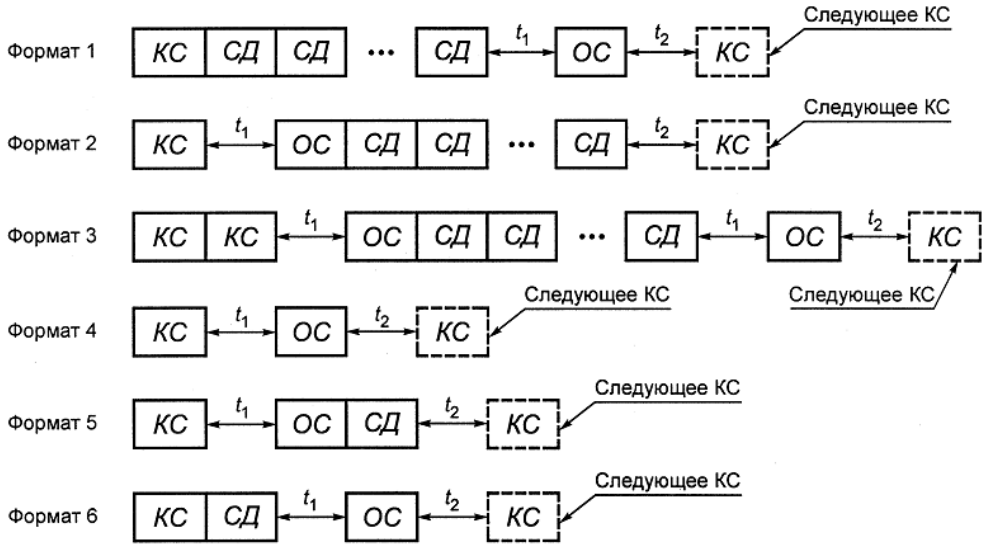


Рисунок 144 – Форматы сообщений

Время непрерывной передачи данных в линию не превышает 660 мкс, что соответствует командному слову и 32-v словам данных.

Формат 1 – передача данных от КШ к ОУ.

Формат 2 – передача данных от ОУ к КШ.

Формат 3 – передача данных от ОУ к ОУ.

Формат 4 – передача КУ.

Формат 5 – передача КУ и прием СД от ОУ.

Формат 6 – передача КУ и СД оконечному устройству.

Групповые сообщения, приведенные рисунке 145, начинающиеся с передачи КШ групповой команды с кодом адреса 11111, используются для передачи информации одновременно нескольким ОУ без выдачи ими ОС.

Формат 7 – передача данных (в групповом сообщении) от КШ к оконечным устройствам.

Формат 8 – передача данных (в групповом сообщении) от оконечного устройства к оконечным устройствам.

Формат 9 – передача групповой команды управления.

Формат 10 – передача групповой команды управления со словом данных.

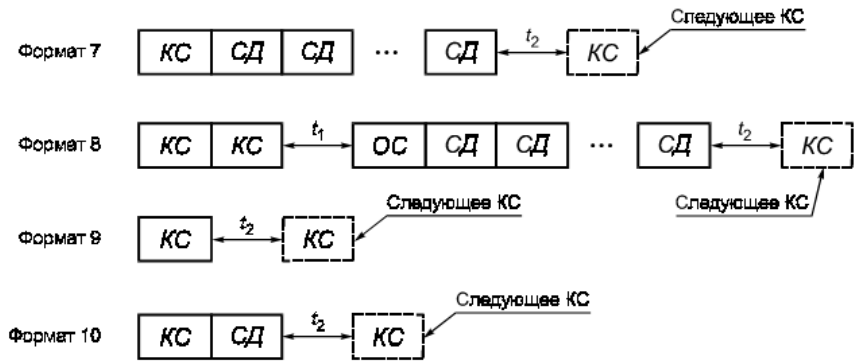


Рисунок 145 – Форматы групповых сообщений

Если ОУ в формате сообщения ОУ-ОУ получило достоверное командное слово на прием информации, то первое СД должно быть им принято через паузу не более (57±3) мкс, в противном случае формируется соответствующий признак ошибки.

26.3 Формат слов

Каждое слово начинается с сигнала пословной синхронизации (с синхросигнала) и имеет 17 информационных разрядов, включая разряд контроля по четности. Форматы слов приведены на рисунке 146.

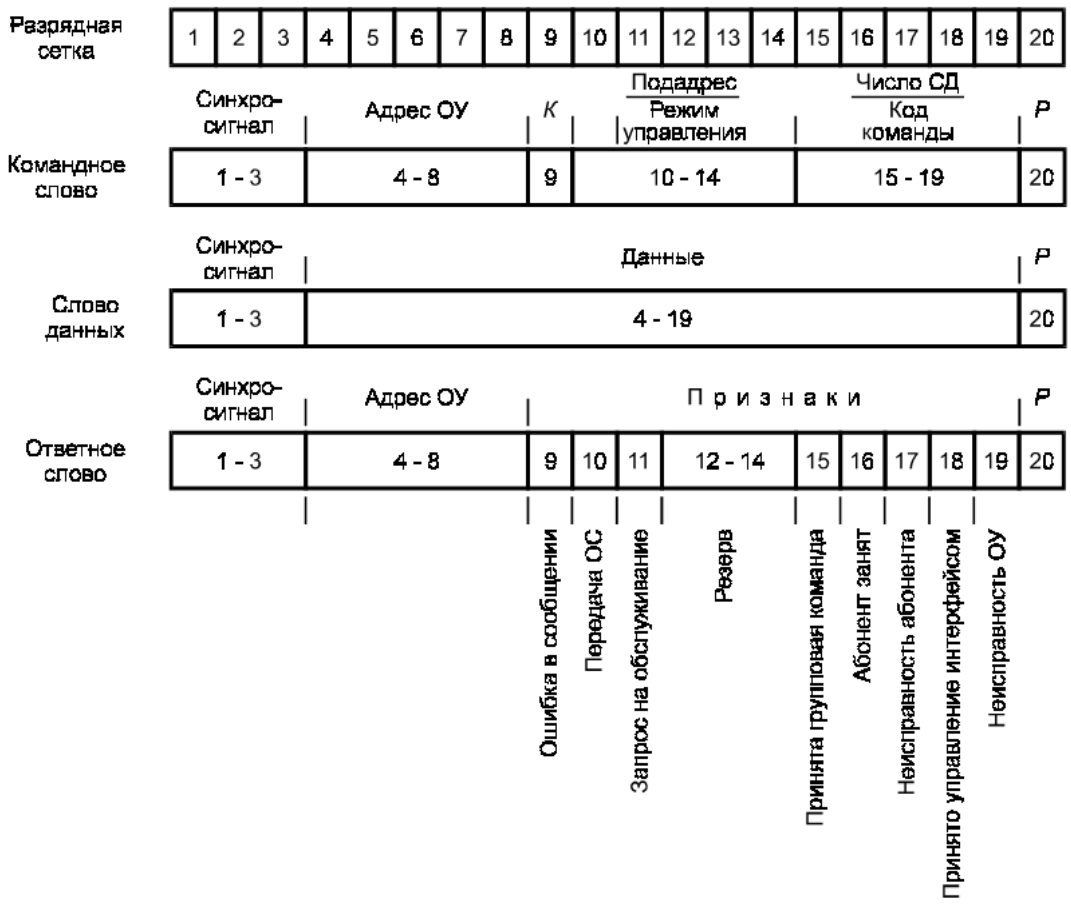


Рисунок 146 – Форматы слов

Командное слово содержит:

– синхросигнал;
– поле «Адрес ОУ»;
– разряд «Прием/Передача»;
– поле «Подадрес/Режим управления»;
– поле «Число СД/Код КУ»;
– разряд контроля по четности.

Синхросигнал имеет длительность, составляющую три интервала времени передачи одного двоичного разряда. Полярность первой половины сигнала положительная, а второй – отрицательная.

Адрес ОУ содержит код адреса из диапазона кодов 00000 – 11110, которому предназначено КС. КС с кодом адреса 11111 называется групповой командой, а сообщение, содержащее групповую команду – групповым.

Разряд «Прием/Передача» указывает на действие, которое должно выполнить ОУ (принимать или передавать СД). Логический нуль означает, что ОУ должно принимать СД, а логическая единица – передавать СД.

Поле «Подадрес/Режим управления» содержит код подадреса ОУ или код признака режима управления 00000 или 11111.

Поле «Число СД/Код КУ» содержит код числа слов данных, которые должны быть переданы или приняты ОУ в связи с приемом адресованного ему КС, или код КУ. В одном сообщении может быть передано не более 32 СД. Числовое значение двоичных кодов, обозначающих число СД, соответствует их десятичным эквивалентам, за исключением кода 00000, который соответствует числу 32.

Разряд контроля по четности используется для контроля по четности предшествующих ему 16 разрядов КС. Разряд принимает такое значение, чтобы сумма значений всех 17 информационных разрядов слова (включая контрольный разряд) была нечетной.

Слово данных содержит:

– синхросигнал;
– данные;
– разряд контроля по четности.

Синхросигнал имеет длительность, составляющую три интервала времени передачи одного двоичного разряда. Полярность первой половины сигнала отрицательная, а второй – положительная.

Поле данных содержит передаваемые данные, а разряд контроля по четности формируется так же, как в командном слове.

Ответное слово содержит:

– синхросигнал;
– поле «Адрес ОУ»;
– поле разрядов признаков состояния: ошибка в сообщении, передача ОС, запрос на обслуживание, принята групповая команда, абонент занят, неисправность абонента, принято управление интерфейсом, неисправность ОУ;

– разряд контроля по четности.

Синхросигнал аналогичен синхросигналу КС. Поле «Адрес ОУ» содержит собственный адрес ОУ. Поле разрядов признаков состояния ОУ отображает текущее состояние ОУ. Разряд контроля по четности формируется так же, как в командном слове.

26.4 Структурная схема в режиме КШ

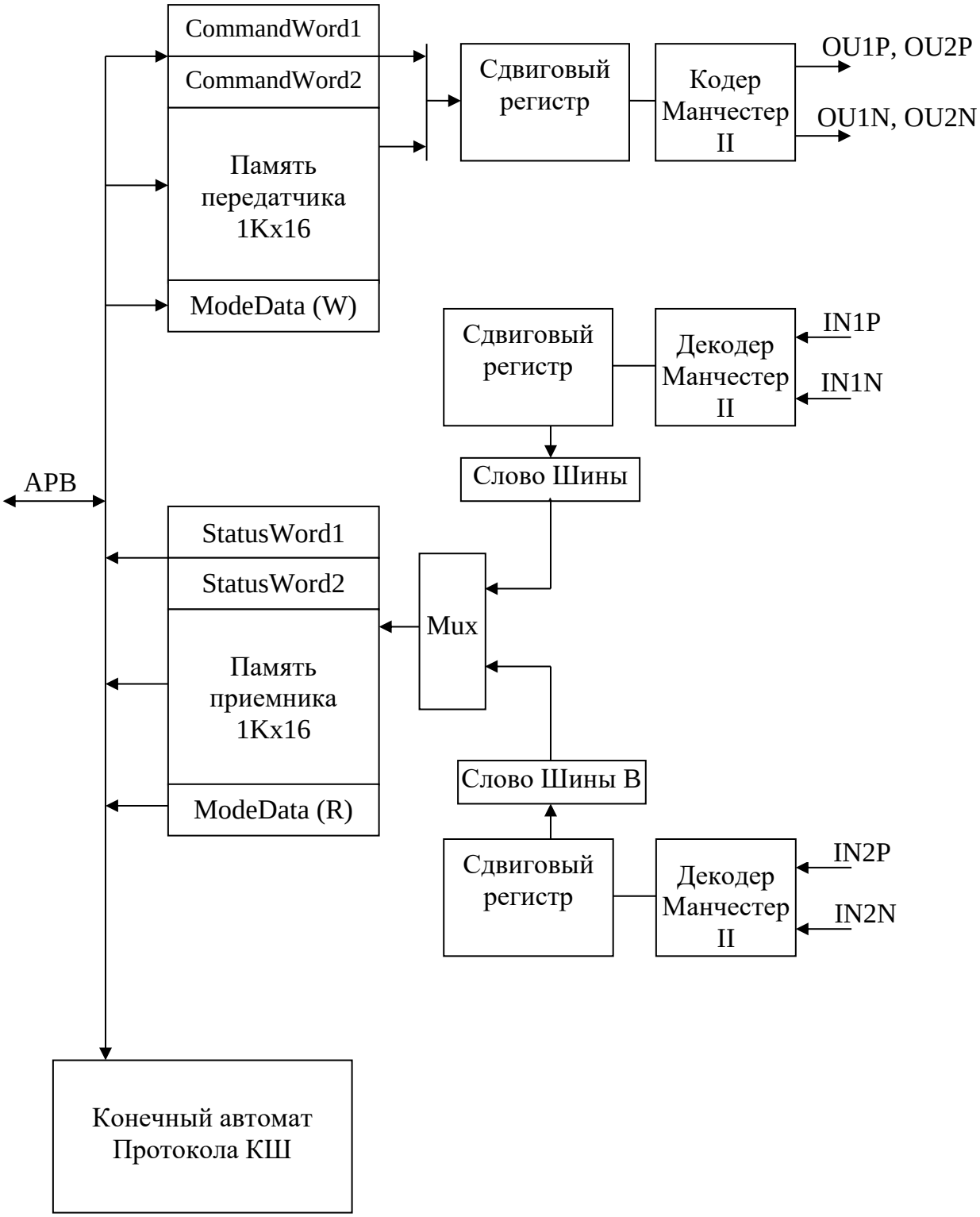


Рисунок 147 – Структурная схема работы в режиме КШ

26.5 Структурная схема в режиме ОУ

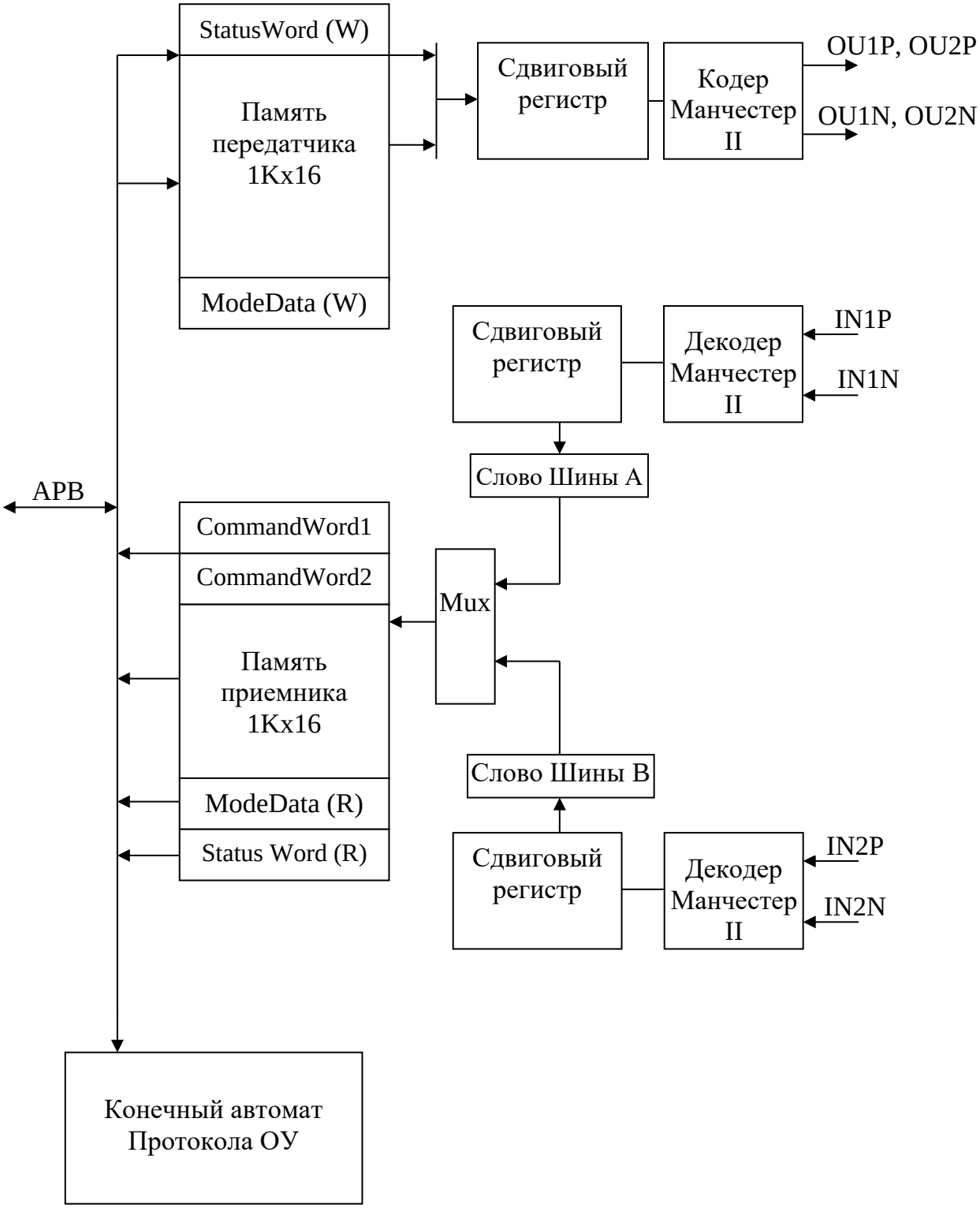


Рисунок 148 – Структурная схема работы в режиме ОУ

26.6 Структурная схема в режиме М

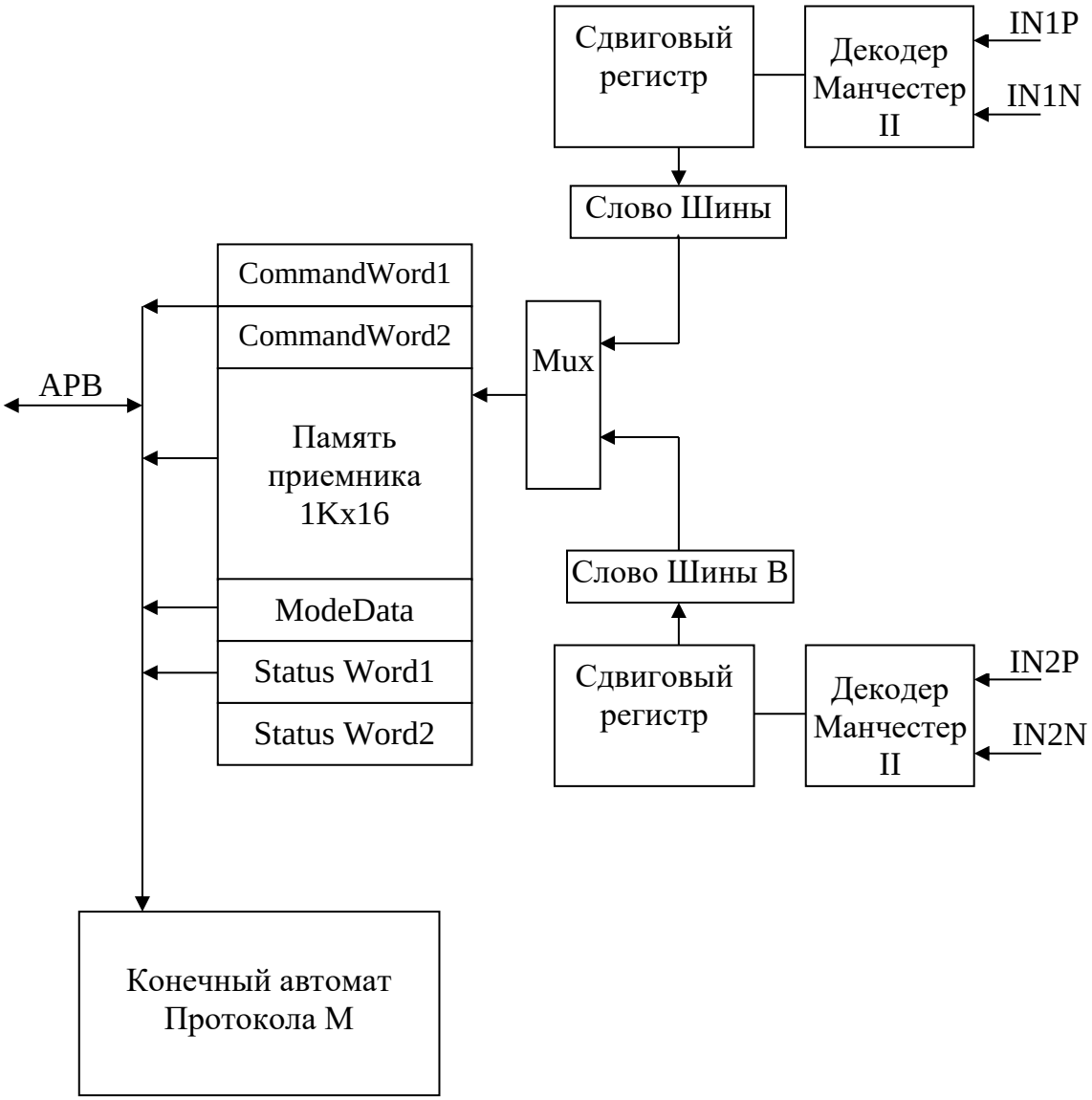


Рисунок 149 – Структурная схема работы в режиме М

26.7 Инициализация

Перед началом работы с контроллером в первую очередь необходимо сбросить контроллер, чтобы очистить все регистры сообщений. Это достигается установкой бита MR регистра CONTROL в логическую единицу. Затем бит необходимо сбросить в нуль. Далее нужно задать в регистре CONTROL значение делителя частоты DIV таким образом, чтобы при делении частоты SOC-шины (SOCCLK) на это значение получить опорную частоту блока контроллера 1 МГц. После этого с помощью бит RTMODE и BCMODE выбирается соответствующий режим работы ОУ или КШ.

Для того чтобы выбрать, какой канал будет использован для передачи данных, основной или резервный, устанавливается соответствующий бит (TRA – основной канал, TRB – резервный канал). В режиме КШ командные слова будут передаваться только по тому каналу, который выбран. В режиме ОУ необходимо установить оба бита, так как ОУ

не может выбирать, по какому каналу ему передавать СД и ОС, и поэтому их передача происходит по тому каналу, по которому было принято КС.

Для режима ОУ в битах RTA4 – RTA0 регистра CONTROL задается адрес ОУ, который должен соответствовать адресу в поле «Адрес ОУ» командного слова, если идет обращение к этому ОУ.

Пример инициализации ОУ
MIL_STD_1553B0->CONTROL=0x00000001; //установка бита MR=1
MIL_STD_1553B0->CONTROL=0x00014078;
//RTMODE=1, TRB=TRA=1, RTA=1, DIV=40

Пример инициализации КШ
MIL_STD_1553B0->CONTROL=0x00000001; //установка бита MR=1
MIL_STD_1553B0->CONTROL=0x00014014;
//BCMODE=1, TRA=1, TRB=0, RTA=0,DIV=40

В обоих случаях значения делителя частоты DIV = 40, что соответствует частоте SOC-шины 40 МГц, и для получения опорной частоты контроллера необходимо 40 МГц/DIV = 1 МГц.

26.8 Прием и передача в режиме ОУ

Для того чтобы настроить контроллер в режиме ОУ, необходимо выполнить все пункты, описанные в подразделе 26.7 «Инициализация». После этого необходимо задать ответное слово для КШ с помощью регистра StatusWord1. В режиме ОУ регистр по записи содержит предназначенное для передачи КШ ответное слово, а по чтению содержит ответное слово, полученное от передающего ОУ в транзакции ОУ-ОУ.

Пример записи ответного слова ОУ
MIL_STD_1553B0->StatusWord1=0x00000800;

В данном случае в регистр заносятся только старшие 5 разрядов, соответствующие адресу ОУ. Остальные разряды признаки состояния ОУ можно оставить в нуле. Но в процессе работы может возникать необходимость изменять эти биты. Для этого необходимо программно устанавливать и сбрасывать эти биты, так как аппаратно они не изменяются.

Для того чтобы обеспечить формат сообщения 5, необходимо задавать слово данных, передаваемое КШ в команде управления. Для этих целей используется регистр ModeData.

Пример записи слова данных команды управления
MIL_STD_1553B0->ModeData=0x000055AA;

После того как проведена инициализация, заданы ответное слово и слово данных команды управления, ОУ сразу готово к работе и может отвечать на все возможные форматы сообщений.

Так как в процессе работы ОУ в каждый момент времени требуется передача определенных СД и СД команды управления, то программно необходимо обновлять те области памяти, которые содержат эти данные. Если эти области не обновляются, то при запросе данных КШ будут переданы те данные, которые были последний раз записаны в эти области памяти. Поэтому при написании программы следует помнить и обновлять данные и слова данных команды управления.

Для хранения СД применяется адресное пространство 0x000-0x3FF (относительно базового адреса периферийного блока). Данные шестнадцатиразрядные, но обращение к ним должно быть выровнено по границе 32-разрядного слова.

Пример инициализации данных для подадреса 1

```
addon=0x20;
for(i=1;i<=32;i++)
{
   MIL_STD_1553B0->DATA[addon]=i;
   addon++;
}
```

Из примера становится ясно, что стартовый адрес памяти СД для подадреса 1 – 0x20, для последующих подадресов: n•0x20, где n- номер подадреса (n=1-31).

При приеме СД или слова данных команды управления, признаком обновления их значений является флаг VALMESS. После того как флаг установлен, можно считать новые данные или слово данных команды управления. Но следует учитывать то, что этот флаг автоматически сбрасывается через 4 мкс после его установки, поэтому желательно применять прерывания по установке сигнала VALMESS.

Пример чтения слова данных команды управления

```
i=MIL_STD_1553B0->ModeData;
```

В переменную i будет прочитано значение слова данных команды управления.

Пример чтения данных для подадреса 1

```
addon=0x20;
for(i=0;i<32;i++)
{
   mas[i]=MIL_STD_1553B0->DATA[addon];
   addon++;
}
```

Для упрощения декодирования команд управления КШ в режиме ОУ доступен регистр кодов MSG полученных сообщений. Каждому формату сообщения на магистрали соответствует определенный код в этом регистре. Чтение и последующая дешифрация этого кода упрощает процедуру декодирования сообщения и уменьшает время обработки сообщений. При использовании регистра экономится время на чтение двух командных регистров и разбор значений бит этих регистров.

26.9 Прием и передача в режиме КШ

В отличие от режима ОУ, в режиме КШ необходимо задавать не ответное слово, а командное слово или два командных слова в режиме работы ОУ-ОУ. Помимо этого, необходимо инициировать процедуру приема или передачи данных установкой бита BCSTART. После завершения транзакции на шине этот бит автоматически сбрасывается в ноль. Поэтому для инициирования новой транзакции нужно повторно устанавливать этот бит.

Пример записи командных слов и бита BCSTART
```
MIL_STD_1553B0->CommandWord1=0x00000820;      //Командное слово 1
MIL_STD_1553B0->CommandWord2=0x00000000;      //Командное слово 2
MIL_STD_1553B0->CONTROL=0x00014016;  //Регистр управления
```

Как видно из примера, в командном слове 1 задается код слов данных 00000, что соответствует 32 СД. Данные будут передаваться от контроллера шины оконечному устройству с адресом 1 из подадреса 1. Второе командное слово задается равным нулю и никак не влияет на транзакцию. В регистре управления устанавливается бит BCMODE, что соответствует режиму работы КШ, а также устанавливается бит BCSTART, что инициирует начало транзакции, выбирается канал А для передачи (TRA=1), а также устанавливается делить частоты 40, что соответствует частоте SOC-шины 40 МГц.

Для того чтобы инициировать прием в этом примере, необходимо только установить бит 10 равным единице в командном слове 1.

Если транзакция завершена успешно (признак VALMESS установился в единицу), то полученные СД или слово данных команды управления могут быть прочитаны. В противном случае устанавливается один из флагов ошибки. Сброс этих флагов осуществляется установкой битом MR или инициированием новой транзакции битом BCSTART.

В режиме работы ОУ-ОУ, форматы сообщений 3 и 8, КШ принимает из магистрали СД в подадрес, указанный во втором командном слове, регистр CommandWord2.

26.10 Прерывания

Для уменьшения потерь времени программы на опрос флагов контроллера, введено одно прерывание, генерируемое при установке любого из флагов контроллера. Прерывание может генерировать установка одного из четырех флагов:

- флаг ошибки;
- флаг успешного завершения транзакции в канале;
- флаг приема достоверного КС, ОС или слова данных команды управления;
- флаг неактивности контроллера.

Каждый из флагов может быть маскирован битами разрешения прерывания по какому-либо флагу.

26.11 Описание регистров

Таблица 264 – Регистры контроллера МКПД (MIL_STD)

Базовый адрес	Смещение	Название	Описание
0x8000_6000		MIL_STD_1553B0	
0x8000_7000		MIL_STD_1553B1	
	0x000-0x3FF	DATA	Память передаваемых/принимаемых СД
	0x400	CONTROL	Регистр управление контроллером
	0x401	STATUS	Регистр состояния контроллера
	0x402	ERROR	Регистр ошибок контроллера
	0x403	CommandWord1	Регистр командного слова 1
	0x404	CommandWord2	Регистр командного слова 2
	0x405	ModeData	Слово данных команды управления
	0x406	StatusWord1	Регистр ответного слова 1
	0x407	StatusWord2	Регистр ответного слова 2
	0x408	INTEN	Регистр разрешения прерываний
	0x409	MSG	Регистр декодирования сообщений

26.11.1 Регистр управления CONTROL

Таблица 265 – Регистр управления контроллером CONTROL

Номер	31….24	23	22	21	20	19	18	17	16
Доступ	U	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс		0	0	0	0	0	0	0	0
	-	RT_HW	INPINV	DIV7	EN_FLT	INVTR	RERR	DIV6	DIV5

Номер	15	14	13	12	11	10	9	8
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	0
	DIV4	DIV3	DIV2	DIV1	DIV0	RTA4	RTA3	RTA2

Номер	7	6	5	4	3	2	1	0
Доступ	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0	0	1
	RTA1	RTA0	TRB	TRA	RTMODE	BCMODE	BCSTART	MR

Таблица 266 – Описание бит регистра CONTROL

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31…24	-	Зарезервировано
23	RT_HW	Бит аппаратной поддержки ГОСТ 51765-2001 для режима ОУ: 1 – аппаратная поддержка разрешена; 0 – аппаратная поддержка запрещена. При установке этого бита достаточно программной реализации циклического подадреса, установки и сброса признаков: абонент занят, неисправность абонента, неисправность ОУ, запрос на обслуживание, а также задание адреса ОУ и бита паритета посредством внешнего соединения
22	INPINV	Бит инверсии входов приемника INxP и INxN: 1 – инверсия разрешена; 0 – аппаратная поддержка запрещена. Вход INxP инвертируется и коммутируется на отрицательный вход декодера манчестерского кода. Вход INxN инвертируется и коммутируется на положительный вход декодера манчестерского кода. Применение этого бита актуально для приемопередатчиков с принудительной установкой выхода приемника в состояние логической «1», например 5559ИН74Т (HI1574)
21	DIV7	Делитель частоты SOCCLK до 1 МГц. Содержит старший разряд делителя, на который необходимо поделить частоту SOCCLK, чтобы получить 1 МГц
20	EN_FLT	Включение фильтрации импульсных помех: 1 – фильтрация включена; 0 – фильтрация выключена. Рекомендуется устанавливать в ноль
19	INVTR	Разрешение инверсии сигналов управления шинными формирователями (сигнал EN для 5559ИН1У) OU1X, OU2X: 1 – инверсия; 0 – прямой выход
18	RERR	Сброс ошибок в режиме ОУ и М: 1 – ошибки могут быть сброшены только битом MR; 0 – сброс ошибок происходит автоматически, после установки бита IDLE
17…11	DIV6-DIV0	Делитель частоты SOCCLK до 1 МГц. Содержит значение, на которое необходимо поделить частоту SOCCLK, чтобы получить 1 МГц. Частота SOCCLK обязательно должна быть не более 120 МГц и кратна восьми. Если SOCCLK не кратна восьми, то DIV[6:3]=(SOCCLK/8)+1, DIV[2:0]=0, но стабильность приема не гарантируется.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
10…6	RTA4-RTA0	Адрес оконечного устройства. Содержит адрес, который присвоен устройству, если контроллер работает в режиме оконечного устройства RTMODE=1; BCMODE=0
5	TRB	Блокировка передатчика резервного канала: 1 – передатчик разблокирован; 0 – передатчик заблокирован
4	TRA	Блокировка передатчика основного канала: 1 – передатчик разблокирован; 0 – передатчик заблокирован
3…2	RTMODE BCMODE	Выбор режима работы контроллера: 10 – режим оконечного устройства; 01 – режим контроллера шины; 11 – режим неадресуемого монитора
1	BCSTART	Инициирует передачу сообщения в канал в режиме КШ: 1 – старт сообщения.; 0 – стоп сообщения. Сбрасывается в ноль автоматически по завершению сообщения
0	MR	Сброс контроллера: 1 – контроллер сбрасывается в исходное состояние; 0 – разрешение работы контроллера

26.11.2 Регистр состояния STATUS

Таблица 267 – Регистр состояния STATUS

Номер	31…11	10	9	8	7	6
Доступ	U	RO	RO	RO	RO	RO
Сброс		0	0	0	0	0
	-	RCVB_stick	RCVA_stick	ERR_stick	VALMESS_stick	RFLAGN_stick

Номер	5	4	3	2	1	0
Доступ	RO	RO	RO	RO	RO	RO
Сброс	0	0	0	0	0	1
	RCVB	RCVA	ERR	VALMESS	RFLAGN	IDLE

Таблица 268 – Описание бит регистра STATUS

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...11	-	Зарезервировано
10	RCVB_stick	Признак активности резервного канала: 0 – канал неактивен; 1 – канал активен. Устанавливается при установке бита RCVB. Сбрасывается программной записью нуля, если RCVB=0.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету
9	RCVA_stick	Признак активности резервного канала: 0 – канал неактивен; 1 – канал активен. Устанавливается при установке бита RCVA. Сбрасывается программной записью нуля, если RCVA=0. В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету
8	ERR_stick	Ошибка в сообщении: 0 – нет ошибок; 1 – в последней транзакции возникла ошибка. Устанавливается при установке бита ERR. Сбрасывается программной записью нуля, если ERR=0. В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету
7	VALMESS_stick	Успешное завершение транзакции в канале: 0 – транзакция завершена с ошибкой, либо транзакции нет в канале; 1 – транзакция завершена успешно. Устанавливается при установке бита VALMESS. Сбрасывается программной записью нуля, если VALMESS =0. В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету
6	RFLAGN_stick	Получено достоверное слово из канала: 0 – нет достоверных слов в канале; 1 – в режиме КШ получено достоверное ответное слово; 1 – в режиме ОУ или М получено достоверное командное слово, ответное слово или слово данных в команде управления. Устанавливается при установке бита RFLAGN. Сбрасывается программной записью нуля, если RFLAGN =0. В установленном состоянии не формирует прерывание. Необходимо следить за своевременным сбросом, иначе бит может относиться не к своему пакету
5	RCVB	Признак активности резервного канала: 0 – канал неактивен; 1 – канал активен
4	RCVA	Признак активности основного канала: 0 – канал неактивен; 1 – канал активен
3	ERR	Ошибка в сообщении: 0 – нет ошибок; 1 – в последней транзакции возникла ошибка.

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
		В режиме ОУ и М, если сброшен бит RERR, автоматически сбрасывается не менее чем через 4 мкс после установки.
2	VALMESS	Успешное завершение транзакции в канале. 0 – транзакция завершена с ошибкой, либо транзакции нет в канале; 1 – транзакция завершена успешно. В режиме ОУ и М автоматически сбрасывается не менее чем через 4 мкс после установки.
1	RFLAGN	Получено достоверное слово из канала: 0 – нет достоверных слов в канале; 1 – в режиме КШ получено достоверное ответное слово; 1 – в режиме ОУ или М получено достоверное командное слово, ответное слово или слово данных в команде управления. Между сообщениями бит автоматически сбрасывается в ноль.
0	IDLE	Состояние контроллера: 1 – контроллер в неактивном состоянии; 0 – контроллер в состоянии обмена сообщениями

26.11.3 Регистр ошибок ERROR

Таблица 269 – Регистр ошибок ERROR

Номер	31...7	6	5	4	3	2	1	0
Доступ	U	RO	RO	RO	RO	RO	RO	RO
Сброс	0	0	0	0	0	0	0	0
	-	PROERR	CONERR	GAPERR	CSYCERR/ SEQERR	DSYCERR/ SYNCERR	MANERR	NORCV

Таблица 270 – Описание бит регистра ERROR

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...7	-	Зарезервировано
6	PROERR	Ошибка в протоколе в режиме КШ: 1 – недопустимое слово обнаружено на шине во время обмена сообщениями; 0 – нет ошибок
5	CONERR	Ошибка непрерывности сообщения: 1 – передача сообщения не непрерывная; 0 – нет ошибок
4	GAPERR	Недопустимая активность на шине: 1 – обнаружена активность на шине в интервале 4 мкс после успешного завершения сообщения; 0 – нет ошибок

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
3	CSYCERR/ SEQERR	Ошибка синхронизации команды в режиме КШ (CSYCERR): 1 – ожидался синхроимпульс команды, а получен синхроимпульс данных; 0 – ошибок нет; Ошибка после приема команды в режиме ОУ (SEQERR). 1 – обнаружена пауза после приема командного слова с битом;10 равным нулю или обнаружены слова данных после приема командного слова с битом 10 равным единице; 0 – ошибок нет; Ошибка в режиме М (SEQERR). 1 – обнаружено отсутствие ожидаемых данных в сообщении или пауза при приеме первого слова; 0 – ошибок нет
2	DSYCERR/ SYNCERR	Ошибка синхронизации данных в режиме КШ (DSYCERR). 1 – ожидался синхроимпульс данных, а получен синхроимпульс команды; 0 – ошибок нет. Ошибка синхронизации в режиме ОУ и М (SYNCERR). 1 – ожидался синхроимпульс команды, а получен синхроимпульс данных или наоборот; 0 – ошибок нет
1	MANERR	Ошибка декодирования NRZ кода: 1 – ошибка в количестве принятых бит или ошибка в бите контроля четности; 0 – ошибок нет
0	NORCV	Ошибка приема: 1 – не получено ответное слово в интервале 14 мкс или не получены ожидаемые данные; 0 – ошибок нет

26.11.4 Регистр команды 1 CommandWord1

Таблица 271 – Регистр команды 1 CommandWord1

Номер	31...16	15…11	10	9…5	4…0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	Адрес ОУ	Прием / Передача	Подадрес / Режим управления	Число СД / Код команды

Таблица 272 – Описание бит регистра CommandWord1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...11	Адрес ОУ	Адрес оконечного устройства, которому предназначено командное слово

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
10	Прием/передача	Бит приема/передачи: 1 – режим работы ОУ-КШ; 0 – режим работы КШ-ОУ
9..5	Подадрес / Режим управления	Содержит подадрес, по которому в памяти располагаются принимаемые или передаваемые СД. В случае передачи команды, содержит код 00000 или 11111
4..0	Число СД / Код команды	Содержит количество принимаемых или передаваемых слов данных. В случае передачи команды содержит код команды из таблицы 1 ГОСТ Р52070-2003
Примечание – В режиме ОУ и М регистр доступен только на чтение, в режиме КШ – только на запись		

26.11.5 Регистр команды 2 CommandWord2

Таблица 273 – Регистр команды 2 CommandWord2

Номер	31...16	15…11	10	9…5	4…0
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	
	-	Адрес ОУ	Прием/ передача	Подадрес / Режим управления	Число СД / Код команды

Таблица 274 – Описание бит регистра CommandWord2

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...11	Адрес ОУ	Адрес оконечного устройства, которому предназначено командное слово
10	Прием/передача	Бит приема/передачи: 1 – режим работы ОУ-ОУ; 0 – командное слово не используется
9..5	Подадрес	Содержит подадрес, по которому в памяти располагаются принимаемые или передаваемые СД
4..0	Число СД	Содержит количество принимаемых или передаваемых слов данных
Примечание – В режиме ОУ и М регистр доступен только на чтение и содержит второе командное слово транзакции ОУ-ОУ. В режиме КШ регистр доступен только на запись и используется для транзакции ОУ-ОУ, если установлен в единицу бит 10		

26.11.6 Слово данных команды управления ModeData

Таблица 275 – Регистр слова данных команды управления ModeData

Номер	31...16	15…0
Доступ	U	R/W
Сброс	0	0
	-	Слово данных команды управления

Таблица 276 – Описание бит регистра ModeData

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..16	-	Зарезервировано
15..0	ModeData	Содержит принятое или передаваемое слово данных в команде управления

26.11.7 Ответное слово 1 StatusWord1

Таблица 277 – Регистр ответного слова 1 StatusWord1

Номер	31...16	15...11	10	9	8
Доступ	U	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0
	-	Адрес ОУ	Ошибка в сообщ.	Пер.ОС	Запр. На обсл.

Номер	7…5	4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W	R/W
Сброс	0	0	0	0	0	0
	-	Прин ГК	Абонзан.	Неисп. Абон.	Прин упр. Инт.	Неисп ОУ

Таблица 278 – Описание бит регистра StatusWord1

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...11	Адрес ОУ	Адрес ОУ, от которого принято ответное слово в режиме КШ. Адрес ОУ, которое передает ответное слово в режиме ОУ
10	Ошибка в сообщ.	Ошибка в сообщении
9	Пер.ОС	Передача ответного слова
8	Запр. На обсл.	Запрос на обслуживание
7...5	-	Зарезервировано
4	Прин. ГК	Принята групповая команды
3	Абон. Зан.	Абонент занят
2	Неисп. Абон.	Неисправность абонента
1	Прин. Упр. Инт.	Принято управление интерфейсом
0	Неисп. ОУ	Неисправность ОУ

Примечания

1 Для режима КШ по чтению регистр содержит первое принятое ответное слово, а в случае транзакции ОУ-ОУ по чтению содержит ОС, принятое от принимающего ОУ (второе ОС на шине).

2 Для режима М по чтению регистр содержит первое ОС в транзакции на шине.

3 Для режима ОУ по чтению регистр содержит ОС, принятое от второго ОУ в транзакции ОУ-ОУ (это может быть как принимающее, так и передающее ОУ)

26.11.8 Ответное слово 2 StatusWord2

Таблица 279 – Регистр ответного слова 2 StatusWord2

Номер	31…16	15…11	10	9	8	7..5	4	3	2	1	0
Доступ	U	RO	RO	RO	RO	U	RO	RO	RO	RO	RO
Сброс	0	0	0	0	0	0	0	0	0	0	0
	-	Адрес ОУ	Ошибка в сообщ.	Пер. ОС	Запр. На обсл.	-	ПринГК	Абон зан.	Неисп. абон.	Прин упр. Инт.	Неисп ОУ

Таблица 280 – Описание бит регистра StatusWord2

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31...16	-	Зарезервировано
15...11	Адрес ОУ	Адрес ОУ, передающего данные в транзакции ОУ-ОУ
10	Ошибка в сообщ.	Ошибка в сообщении
9	Пер. ОС	Передача ответного слова
8	Запр. На обсл.	Запрос на обслуживание
7…5		Зарезервировано
4	Прин. ГК	Принята групповая команды
3	Абон. Зан.	Абонент занят
2	Неисп. Абон.	Неисправность абонента
1	Прин. Упр. Инт.	Принято управление интерфейсом
0	Неисп. ОУ	Неисправность ОУ

Примечания

1 Для режима КШ доступен только на чтение и в случае транзакции ОУ-ОУ содержит ОС, принятое от передающего ОУ (первое ОС на шине).

2 Для режима М доступен только на чтение и содержит второе ОС в транзакции ОУ-ОУ на шине.

3 В режиме ОУ регистр не используется

26.11.9 Регистр разрешения прерываний INTEN

Таблица 281 – Регистр разрешения прерываний INTEN

Номер	31…4	3	2	1	0
Доступ	U	R/W	R/W	R/W	R/W
Сброс		0	0	0	0
	-	ERRIE	VALMESSIE	RFLAGNIE	IDLEIE

Таблица 282 – Описание бит регистра INTEN

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31..4		Зарезервировано
3	ERRIE	Прерывание при возникновении ошибки в сообщении: 0 – прерывание маскировано; 1 – прерывание разрешено, это позволяет генерировать прерывание при возникновении ошибок в сообщении

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
2	VALMESSIE	Прерывание при успешном завершении транзакции в канале: 0 – прерывание маскировано; 1 – прерывание разрешено, это позволяет генерировать прерывание при успешном завершении обмена данными в канале
1	RFLAGNIE	Прерывание при приеме достоверного слова: 0 – прерывание маскировано; 1 – прерывание разрешено, это позволяет генерировать прерывание при приеме достоверного ОС в режиме КШ или достоверного КС, ОС или слова данных команды управления в режиме ОУ
0	IDLEIE	Прерывание неактивности контроллера: 0 – прерывание маскировано; 1 – прерывание разрешено, это позволяет генерировать прерывание по переходу контроллера в неактивное состояние

26.11.10 Регистр декодирования сообщений MSG

Таблица 283 – Регистр декодирования сообщений MSG

Номер	15..14	13..0
Доступ	U	RO
Сброс		0
	-	Код сообщения
Примечание – Регистр содержит код сообщения, полученного в режиме ОУ или М, и доступен только на чтение. В режиме КШ регистр не используется. Регистр обновляется каждый раз при получении нового достоверного КС		

Таблица 284 – Коды сообщений регистра MSG

Код сообщения	CommandWord1	CommandWord2
Команды обмена данными	15:11 10 9:5 4:0	15:11 10 9:5 4:0
0001 Команда приема КШ-ОУ, не групповая	RTA 0 00001-11110 XXXXX	
0080 Команда приема КШ-ОУ, групповая	11111 0 00001-11110 XXXXX	
0004 Команда приема ОУ-ОУ, не групповая	RTA 0 00001-11110 XXXXX	XXXXX 1 00001-11110 XXXXX
0100 Команда приема ОУ-ОУ, групповая	11111 0 00001-11110 XXXXX	не RTA 1 00001-11110 XXXXX
0402 Команда передачи ОУ-КШ	RTA 1 00001-11110 XXXXX	
1008 Команда передачи ОУ-ОУ, не групповая	не F 0 00001-11110 XXXXX	RTA 1 00001-11110 XXXXX
0200 Команда передачи ОУ-ОУ, групповая	11111 0 00001-11110 XXXXX	RTA 1 00001-11110 XXXXX
Команды управления		
0410 Код 0-15 К=1 нет данных, не групповая	RTA 1 00000 11111 0XXXX	
0400 Код 0-15 К=1 нет данных, групповая	11111 1 00000 11111 0XXXX	
2420 Код 16-31 К=1 с данными, не групповая	RTA 1 00000 11111 1XXXX	
0040 Код 16-31 К=0 с данными, не групповая	RTA 0 00000 11111 1XXXX	
0800 Код 16-31 К=0 с данными, групповая	1111 0 00000 11111 1XXXX	

26.11.11 Память принимаемых/передаваемых данных DATA

Память принимаемых/передаваемых данных**.**

Примечание – Данные читаются из памяти или записываются в память в соответствии с подадресом (биты с 9 по 5) достоверного командного слова. Каждому подадресу соответствует 32×16 ячеек памяти на прием и 32x16 ячеек памяти на передачу. Общий объем памяти данных 2К×16.

27 Модуль цифрового смесителя

Модуль цифрового смесителя (далее ЦС) предназначен для приема информации от внешнего RF-приемника сигналов систем GPS/Glonass, предварительной обработки принятых данных и передачи данных во внутреннюю память процессора. ЦС может принимать сигналы от двух внешних RF-приемников. В качестве внешней микросхемы RF-приемника может быть использована, например, gp2015 фирмы Zarlink Semiconductors Inc. Внутренняя структура ЦС представляет собой набор из 16 идентичных каналов обработки, которые могут работать параллельно.

Назначение внешних выводов модуля ЦС приведено в таблице 285.

Таблица 285 – Назначение внешних выводов модуля ЦС

Обозначение вывода	Назначение вывода	Тип вывода	Функциональное назначение
PC[24]	GPS0_CLKO	O	Интерфейс GPS0. Выходной синхросигнал
PC[25]	GPS0_MAG	I	Интерфейс GPS0. Амплитуда
PC[27]	GPS0_SIGN	I	Интерфейс GPS0. Знак
PC[28]	GPS1_CLKO	O	Интерфейс GPS1. Выходной синхросигнал
PC[29]	GPS1_MAG	I	Интерфейс GPS1. Амплитуда
PC[31]	GPS1_SIGN	I	Интерфейс GPS1. Знак
L0DATIP[6]	GPS0_CLKIP	I	GPS0. Вход синхросигнала (+)
L0DATIN[6]	GPS0_CLKIN	I	GPS0. Вход синхросигнала (-)
L0DATIP[7]	GPS1_CLKIP	I	GPS1. Вход синхросигнала (+)
L0DATIN[7]	GPS1_CLKIN	I	GPS1. Вход синхросигнала (-)

Внутренний интерфейс модуля ЦС представлен в таблице 286. Описанные сигналы и шины представляют собой интерфейс внутреннего модуля в системе на кристалле.

Таблица 286 – Интерфейс ЦС

Сигнал	Тип	Функция
PCLK	Вход	Синхросигнал периферийной шины
PRESET	Вход	Сброс (активный 0)
PSEL	Вход	Выбор модуля ЦС (активная 1)
PADDR [11:0]	Вход	Адрес регистра
PWRITE	Вход	Признак записи
PWDATA [31:0]	Вход	Шина данных для записи
PRDATA [31:0]	Выход	Шина данных для чтения
DCLK0	Вход	Синхросигнал RF-приемника 0
SAT0[1:0]	Вход	Данные (sign+magn) приемника 0
DCLK1	Вход	Синхросигнал RF-приемника 1

Сигнал	Тип	Функция
SAT1[1:0]	Вход	Данные (sign+magn) приемника 1
DMA_WE	Выход	Строб записи данных в контроллер DMA
DMA_busy	Вход	Признак занятости DMA
DMA_WA [19:0]	Выход	Адрес для DMA
DMA_WD [127:0]	Выход	Данные для DMA
INT	Выход	Запрос прерывания от модуля ЦС

Для сигналов, описанных в таблице, внешними сигналами микросхемы являются только входы от RF-приемников. Регистры модуля доступны посредством интерфейса периферийной шины. Обработанные данные поступают в канал 12 контроллера DMA, далее они направляются в буфер, запрограммированный каналом. Канал 12 DMA задает базовый адрес буфера приема. Смещение внутри буфера формируется каналом ЦС и передается в DMA вместе с данными. Модуль ЦС может формировать запрос прерывания в контроллер прерываний.

27.1 Регистры ЦС

Список регистров ЦС приведен в таблице 287. Регистры доступны со стороны шины периферийных устройств и имеют базовый адрес 0x80005000.

Таблица 287 – Регистры ЦС

Смещение	Имя	Описание	Сброс
0-7	CH0	Регистры канала 0	
8-15	CH1	Регистры канала 1	
…	…	Регистры канала 2--14	
0x78-0x7F	CH15	Регистры канала 15	
0x80	CR[31:0]	Регистр управления	0
0x81	SR[31:0]	Регистр состояния	0
0x82	LEN[7:0]	Регистр количества (биты 7:0) принимаемых данных. Задает количество квадрослов данных, которое каждый из каналов выдает в контроллер DMA до момента переключения в другой режим. Одинаково для всех каналов. Каждое квадрослово содержит восемь пар (два числа по восемь бит) отсчетов.	
0x83	TST[31:0]	Регистр тестовых данных (биты 31:0). Запись в этот регистр вызывает подачу на корреляторы восьми пар (двух бит) данных. Биты 31:16 канал 1, биты 15:0 канал 0	0

Смещение	Имя	Описание	Сброс
0x84	IRQ[15:0]	Регистр запросов прерываний (биты 15:0). Информирует о переключении канала из одного состояния в противоположное. Биты могут быть сброшены записью первого значения.	
0x85	CFG[3:0]	Регистр конфигурации модуля ЦС	

Каждый из каналов может работать с одним из двух буферов. После выгрузки заданного количества квадрослов входных отсчетов (задается регистром LEN) канал переключается на работу с другим буфером. При этом он использует другой набор регистров.

Соответственно, при работе с другим буфером канал может выбрать другой источник сигнала, значение задержки, а также значения шагов.

Длина выгружаемых данных для всех каналов одинакова.

При переключении канала вырабатывается флаг прерывания. Канал может быть обслужен и ему можно задать новые параметры.

27.1.1 Регистр управления CR

Регистр позволяет задать основные режимы работы модуля. Назначение бит регистра приведено в таблице 288.

Таблица 288 – Регистр управления

Бит	Название	Функция
0	EN0	Разрешение работы канала 0 (1)
…		
15	EN15	Разрешение работы канала 15 (1)
16	IE0	Разрешение прерывания от канала 0 (1 – разрешено)
…		
31	IE15	Разрешение прерывания от канала 15 (1 – разрешено)

После сброса значение регистра управления равно 0.

27.1.2 Регистр состояния SR

Позволяет анализировать состояние модуля. Назначение бит регистра приведено в таблице ниже.

Таблица 289 – Регистр состояния

Бит	Название	Функция
0	PPS0	1 – канал 0 работает с буфером 1; 0 – канал 0 работает с буфером 0
…	…	..
15	PPS15	1 – канал 15 работает с буфером 1; 0 – канал 15 работает с буфером 0
16	Err0	1 – ошибка в работе канала 0 (переполнение); 0 – нет ошибок

Бит	Название	Функция
..	…	..
31	Err15	1 – ошибка в работе канала 15(переполнение); 0 – нет ошибок

27.1.3 Регистр количества LEN

Позволяет задавать количество отгружаемых каналом квадрослов данных до момента переключения в другой режим. Назначение бит регистра приведено в таблице ниже.

Таблица 290 – Регистр LEN

Бит	Название	Функция
7:0	LEN	Количество квадрослов данных минус 1
31:8	-	Не используются. При чтении всегда ноль

27.1.4 Регистр конфигурации CFG

Позволяет задавать режим работы модуля ЦС. Назначение бит регистра приведено в таблице ниже.

Таблица 291 – Регистр TSM

Бит	Название	Функция
0	TESTM	Разрешение тестового режима: 1 – разрешено; 0 – запрещено. Нормальный режим работы
1	NPS0	Выбор активного фронта/среза синхросигнала приемника 0: 0 – фронт (переход из 0 в 1); 1 – срез (переход их 1 в 0)
2	NPS1	Выбор активного фронта/среза синхросигнала приемника 1: 0 – фронт (переход из 0 в 1); 1 – срез (переход их 1 в 0)
3	-	Бит общего назначения
31:4	-	Не используются. При чтении всегда ноль

27.1.5 Регистр тестовых данных TST

Позволяет задавать данные для работы модуля ЦС без использования внешних приемников. Регистр имеет 32 разряда, и младшие 16 разрядов используются как данные от приемника 0, а старшие – как данные от приемника 1. Как известно, разрядность данных приемника равна двум битам. В модуле ЦС эти данные поступают на линию задержки, которая имеет восемь стадий. Использование 16 разрядов данных тестового регистра позволяет задать состояние всей линии задержки конкретного приемника. Каждый канал обрабатывает двухбитные входные данные с конкретной стадии линии задержки. Номер стадии программируется.

27.1.6 Регистр запросов прерываний IRQ

Отражает состояние запросов прерываний от каждого из каналов ЦС. Модуль ЦС имеет один общий запрос прерывания в контроллере прерываний, поэтому для определения номера канала, вызвавшего запрос прерывания может использоваться регистр IRQ. Биты с 0 по 15 регистра соответствуют каналам с 0 по 15 модуля ЦС. Старшие 16 бит не используются и при чтении равны 0. Регистр запросов доступен по чтению. При записи возможен только сброс бит регистра. Если при записи какой-то из младших 16 разрядов данных равен единице, то соответствующий ему разряд регистра IRQ будет сброшен в ноль. Нулевое значение разряда данных не влияет на бит регистра.

27.2 Регистры канала

Таблица 292 – Регистры канала

Смещение	Имя	Описание	Сброс
0	SC0_CNT	27 бит регистр-счетчик таблицы синуса-косинуса	
1	SC0_STEP	32 бит регистр приращения счетчика SC0_CNT	
2	DZ0_CNT	27 бит регистр-счетчик дециматора	
3	DZ0_STEP	32 бит регист приращения счетчика DZ0_CNT	
4	SC1_CNT	27 бит регистр-счетчик таблицы синуса-косинуса	
5	SC1_STEP	27 бит регист приращения счетчика SC1_CNT	
6	DZ1_CNT	27 бит регистр-счетчик дециматора	
7	DZ1_STEP	27 бит регист приращения счетчика DZ1_CNT	

27.2.1 Регистры SCx_CNT и SCx_STEP

Регистр SC_CNT содержит значение указателя на таблицу синусов-косинусов. Начальное значение указателя можно задать путем записи в регистр. Каждый раз после чтения константы из таблицы (обработка одного принятого значения), значение указателя наращивается на значение в регистре SC_STEP, т.е. выполняется операция SC_CNT=SC_CNT + SC_STEP[26:0]. Таблица значений синусов-косинусов имеет 8 элементов. Для выбора одного из восьми значений используются только биты 26:24 регистра SC_CNT. Значения синусов и косинусов приведены в таблице ниже.

Таблица 293 – Значение таблицы синусов косинусов

SC_CNT [26:24]	SIN (SATi)				COS(SATi)			
	00	10	01	11	00	10	01	11
0	1	-1	3	-3	2	-2	6	-6
1	2	-2	6	-6	1	-1	3	-3
2	2	-2	6	-6	-1	1	-3	3
3	1	-1	3	-3	-2	2	-6	6
4	-1	1	-3	3	-2	2	-6	6
5	-2	2	-6	6	-1	1	-3	3
6	-2	2	-6	6	1	-2	3	-3
7	-1	1	-3	3	2	-1	6	-6

Поскольку только 27 младших бит используются для значения приращения, оставшиеся биты используются для задания других функций канала. Бит SC_STEP[31] выбирает источник сигнала (0 – приемник 0, 1 – приемник 1). Биты SC_STEP[30:28] представляют собой трех битовое значение, которое используется для выбора стадии линии задержки, с которой выполняется прием данных для обработки. Пара регистров 0 используется для работы в текущем режиме, а пара 1 для работы в следующем режиме.

27.2.2 Регистрs DZ_CNT и DZ_STEP

Данные регистры используются для задания коэффициента децимации. Каждое входное значение с приемника сигнала, после обработки с использований значений синуса и косинуса, поступает в аккумулятор сигналов, который выполняет накапливание поступающих значений. Значение регистра DZ_CNT позволяет задать количество значений, которые поступают на аккумулятор, для формирования итоговой суммы. Счетчик после каждого такта накопления увеличивает свое значение на приращение DZ_STEP, т.е. выполняется операция DZ_CNT = DZ_CNT + DZ_STEP. Момент переполнения счетчика DZ_STEP, т.е. загрузка в DZ_CNT значения которое имеет старший бит (26-й) равным нулю в момент, когда значение бита DZ_CNT[26] равно единице, вызывает окончание текущего накопления, формирование двух отсчетов и начало новых накоплений. Накопленная пара отсчетов поступает в выходной буфер. Когда будут сформированы 8-мь пар отсчетов (128-бит квадрослово), канал выполнит запрос к контроллеру DMA и передаст подготовленное значение. При этом канал также подготовит значение смещение адреса, по которому должна быть выполнена запись данных. Формирование смещения происходит следующим образом. После старта канала его внутренний счетчик QW_CNT начинает отсчет переданных квадрослов. Значение счетчика изменяется от нуля до значения, заданного в регистре LEN, т.е. максимальный диапазон значений от 0 до 255. Состав смещения показан в таблице 294.

Таблица 294 – Структура смещения канала

Биты	Значение	Описание
1:0	00	Адрес слова в квадрослове
9:2	QW_CNT	Номер квадрослова
13:10	CH_N	Номер канала
14	PPS	Выбор одного из двух буферов. Значение, с которым работает буфер, задается в регистре состояния
19:15	00000	Всегда нули

Таким образом, зная номер канала и номер буфера с которым он работает, а также зная максимальное количество формируемых отсчетов, можно определить местонахождение данных в оперативной памяти и выполнить дальнейшую обработку.

После выгрузки заданного значения отсчетов канал переключается на работу со вторым буфером и первый становится доступным для обработки. При этом для второго буфера канал использует новый набор регистров управления.

Факт окончания выгрузки данных в текущий буфер отражается в регистре запросов прерывания и может использоваться для выполнения процессором действий по обслуживанию канала.

Все сформированные данные передаются во внутреннюю оперативную память через канал DMA 12. В канале 12 задается базовый адрес массива данных модуля ЦС. Скорость работы канала должна быть достаточной для обслуживания всех 16 каналов ЦС. Если в процессе обслуживания какой-то канал не сможет передать свои данные, будет сформирован признак ошибки.

28 Модуль цифровой обработки UP/DOWN

Модуль цифровой обработки UP/DOWN может быть сконфигурирован для работы в двух режимах:

– режим децимации входного сигнала (DOWN);

– режим интерполяции выходного сигнала (UP).

В режиме DOWN модуль (см. рисунок 150):

– Принимает информацию от приемника LINK-порта и помещает ее во входной буфер. Входная информация представляет собой 16-разрядные числовые данные в дополнительном коде, упакованные в 128-разрядные слова.

– Извлекает информацию из буфера, производит перемножение данных на значения синусов и косинусов Ti_s, Ti_c из таблицы. Каждое 16-разрядное число умножается на два коэффициента и получается одно комплексное число с 16-разрядными мнимой и действительной частями.

– Перемноженные данные (комплексное число) поступают на входной фильтр (filter_ADD). На данном этапе производится накопление данных.

– После входного фильтра данные поступают на выходной фильтр (filter_SUB).

– Результат выходного фильтра поступает на сдвигатель, где из 86-разрядного входного данного формируется 16-разрядный результат. Два 16-разрядных результата образуют одно комплексное число, которое помещается в выходной буфер. В буфере числа упаковываются в 128-разрядные слова.

– Данные с выходного буфера могут быть выданы на периферийную шину обмена при чтении с использованием процессора либо контроллера DMA. Выходной буфер имеет возможность формирования запроса на обслуживание к контроллеру DMA.

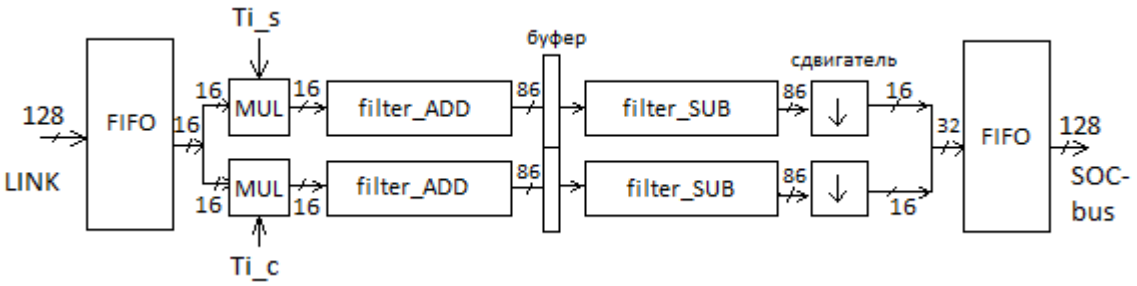


Рисунок 150 – Структура модуля в режиме DOWN

Структура входного и выходного фильтров для режима DOWN приведена на рисунке 151. На рисунке представлен случай длины фильтра равной трем.

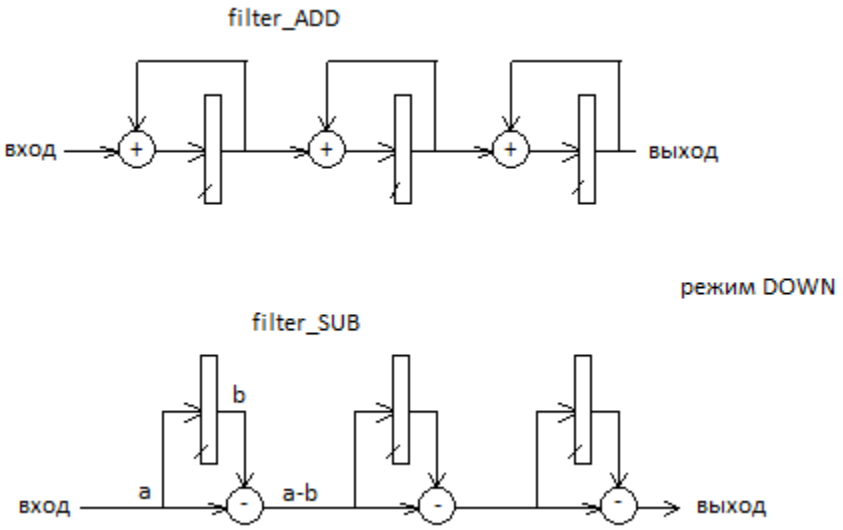


Рисунок 151 – Структура фильтров для режима DOWN

Вообще, итоговая передаточная характеристика каскада фильтров H(z) имеет следующий вид

$$\left(\frac{1}{M}\cdot\frac{1-Z^{-M}}{1-Z^{-1}}\right)^{N}, \qquad (11)$$

где M – коэффициент децимации (см. определение поля Kdelay регистра CR);
N – порядок фильтра, определяемый полем FLEN регистра CR.

Фильтр filter_ADD тактируется синхросигналом процессорного ядра и за один такт может производить обработку двух отсчетов. В связи с этим коэффициент децимации, задаваемый параметром Kdelay, может быть только четным. Фильтр filter_SUB тактируется синхросигналом периферийной шины. Выдача одного отсчета производится при передаче одного результата с первичной стадии обработки (от фильтра filter_ADD). Например, если значение Kdelay равно четырем, то каждые пять тактов процессорной частоты будет происходить обработка 10 входных отсчетов и передача одного результата на фильтр filter_SUB. Соответственно, каждые пять тактов процессорной частоты фильтр filter_SUB будет посылать в выходной буфер одно комплексное 32-разрядное число и каждые 20 процессорных тактов (пять тактов • четыре числа) будет формироваться одно квадрослово данных, и посылаться запрос к контроллеру DMA на пересылку данных. Приемник порта связи передает в модуль обработки данные упакованными 128-разрядными словами. При обработке данных со скоростью два 16-разрядные отсчета за один процессорный такт, скорость поступления данных от приемника порта связи не должна превышать одного квадрослова за четыре процессорных такта. При работе с восьми разрядной шиной данных, порт связи за каждый такт частоты приемника принимает один 16-разрядный отсчет. Таким образом, за восемь тактов частоты приема порт связи сформирует одно квадрослово. Частота приемника LINK-порта, в этом случае, не должна превышать частоту процессора более, чем в два раза. При работе с 16-разрядной шиной данных частота приемника порта связи

не должна превышать частоту процессорного ядра, т.к. в противном случае, может произойти переполнение входного буфера модуля обработки.

В режиме UP модуль (рисунок 152):

– Принимает информацию с шины периферийных устройств (при выполнении операции записи процессором или контроллером DMA) и помещает ее во входной буфер. Информация представлена в виде 32-разрядного комплексного числа, мнимая и действительная части разрядностью 16 бит. Числа упакованы в 128-разрядные слова.

– Извлекает информацию из буфера и подает их на входной фильтр (filter_SUB). Каждый такт извлекается и обрабатывается одно комплексное число.

– После входного фильтра данные поступают на выходной фильтр (filter_ADD).

– Результат выходного фильтра предварительно сдвигается вправо на заданное число разрядов, с целью выделения значащей 16-разрядной части.

– Результаты работы сдвигателя перемножаются на коэффициенты из таблицы, суммируются и помещаются в выходной буфер. В выходной буфер помещаются 16-разрядные результаты в дополнительном коде, которые упаковываются в 128-разрядные слова.

– Данные с выходного буфера могут быть выданы в передатчик LINK-порта. При этом инициатором чтения данных из буфера является передатчик LINK-порта.

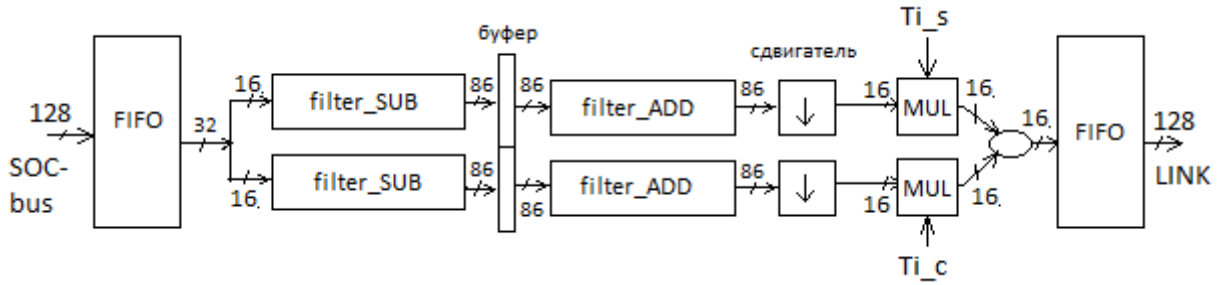


Рисунок 152 – Структура модуля в режиме UP

Структура фильтров filter_SUB и filter_ADD для режима UP аналогична структуре этих же фильтров для режима DOWN.

1 При готовности данных во входном FIFO и готовности промежуточного буфера принять данные, фильтр filter_SUB формирует один отсчет и помещает его в буфер.

2 Получив данные от первичного фильтра filter_SUB, вторичный фильтр filter_ADD формирует несколько выходных отсчетов в соответствии с заданным коэффициентом интерполяции Kdelay.

3 Каждый отсчет (комплексная и мнимая части) поступают на сдвигатель для выделения 16-разрядного результата, затем умножается на коэффициенты синусов и косинусов Ti_s, Ti_c из таблицы синусов и косинусов.

4 Результаты перемножения частей комплексного числа складываются, и получается один 16-разрядный выходной отсчет.

5 Выходные отсчеты упаковываются в 128-разрядные квадрослова.

6 Как только полное 128-разрядное число готово, оно может быть считано передатчиком порта связи и передано во внешнее устройство.

Фильтр filter_SUB работает на частоте периферийной шины, фильтр filter_ADD работает на частоте процессорного ядра и может формировать два выходных отсчета за один такт. Таким образом, каждые четыре такта процессорного клока в выходное FIFO поступает одно квадрослово данных для передачи. Скорость считывания из выходного FIFO не должна превышать скорости записи. В противном случае будет передано предыдущее значение. Скорость продвижения конвейера обработки определяется степенью наполнения выходного буфера и наличием данных во входном буфере. Если в выходном буфере имеется свободное место, модуль производит обработку и формирует выходной отсчет. Естественно, при наличии входных данных от процессора во входном буфере.

Таблицы синусов и косинусов для DOWN и UP преобразований зашиты в ПЗУ. Таблица SC_table содержит значения синуса-косинуса в одном квадранте (по 256 16-битных значений). Таблица CORR_table – коррекция (линейная аппроксимация). 14-битное значение текущего указателя угла ugol[13:0] формирует итоговое значение синуса-косинуса в соответствии со следующим алгоритмом:

– биты ugol[13:6] выбирают грубое значение в квадранте из таблицы SC_table: coarse_sin = SC_table(ugol[13:6]);

– эти же биты выбирают значение угла наклона аппроксимирующей прямой: corr_tan = CORR_table(ugol[13:6]);

– результирующее значение синуса-косинуса будет: sinus = coarse_sin + corr_tan•ugol[5:0];

– само 16-битное значение текущего указателя угла ugol формируется из предыдущего прибавлением значения регистра STEP.

28.1 Регистры модуля

В таблице 295 приведен список всех регистров контроллера. Регистры контроллера доступны со стороны шины периферийных устройств и имеют базовый адрес 0x80000280, 0x80000260, 0x800002A0, 0x800002C0 для модулей UP/DOWN0, -1, -2, -3, соответственно).

Таблица 295 – Регистры модуля

Смещение	Имя	Тип	Значение по сбросу	Описание
0	CR	R/W	0	Регистр управления
1	SR	R	0	Регистр состояния
2	STEP	R/W	0	Регистр шага по таблице коэффициентов
3	-	-	-	Зарезервировано
7-4	DR	R/W	-	Регистр данных
8	RCNT	R/W	0	Счетчик отсчетов
9	XCR	R/W	0	Регистр управления инициализацией от внешних выводов
10	RCNT_STEP	R	0	Составной регистр формата {RCNT_buf, 0x0000, STEP}

28.1.1 Регистр управления CR

Регистр позволяет задать основные режимы работы модуля. Назначение бит регистра приведено в таблице 296.

Таблица 296 – Регистр управления

Бит	Название	Значение по сбросу	Функция
0	EN	0	Разрешение работы: 1 – работа разрешена; 0 – работа запрещена
1	LINK	0	Источник данных в режиме DOWN: 0 – прием данных с шины APB; 1 – прием данных с порта LINK
2	ROUND	0	1 – округление выходного результата; 0 – нет округления
3	SAT	0	1 – насыщение для выходного результата; 0 – нет насыщения
4	-	-	Зарезервировано
5	TBD	0	0 – первый 16-бит отсчет находится в младших битах 128-бит слова; 1 – в старших битах
6	IQ_QI	0	Выдача результата из двух 16-разрядных половин: 1 – формат выдачи данных {I[15:0]}, Q[15:0]}; 0 – формат выдачи данных {Q[15:0]}, I[15:0]}
7	DAM	0	Режим DOWN либо UP: 0 – DOWN; 1 – UP
16:8	Kdelay	0	Коэффициент передачи между фильтрами. Определяет коэффициент интерполяции/децимации, который равен (Kdelay+1)•2. Допустимые значения Kdelay > 0
23:17	SHFR	0	Сдвиг вправо выходного 86-разрядного данного для получения 16-разрядного результата SHFR = FLEN•log2((Kdelay+1)•2), где FLEN – это количество ступеней обработки (3, 5, 7)
24	INT_BLK	0	Блокировка прерывания при обнулении RCNT: 1 – прерывание запрещено; 0 – прерывание разрешено
27:25	-	0	Зарезервировано
29:28	FLEN	0	Длина фильтров: 00 – три ступени; 01 – пять ступеней; 10 – семь ступеней
30	RCNT_ON	0	Разрешение работы счетчика: 1 – работа разрешена; 0 – работа запрещена
31	LNKUSE	0	Номер LINK-порта для приема (DOWN) или передачи данных (UP)

После сброса значение регистра управления равно нулю. Назначение бит регистра зависит от выбранного режима (DOWN или UP).

28.1.1.1 *Режим DOWN*

Режим работы DOWN задается, если бит DAM равен нулю. В этом случае модуль принимает данные и производит децимацию. Необходимо задать источник данных. Для выбора Link-порта необходимо установить бит LINK. В этом случае входное FIFO находится под управлением приемника Link-порта и способно принимать 128-разрядные данные.

В процессоре имеется два Link-порта. Бит LNKUSE позволяет выбрать нулевой (если «0») или первый (если «1») порт. В модуле имеются фильтры обработки данных, для которых можно задать требуемую длину. Биты FLEN позволяют выбрать количество ступеней обработки для фильтров. В фильтре filter_ADD происходит накопление информации. Параметр Kdelay позволяет задать количество накоплений, которые произведет первый фильтр до того, как полученный результат передаст во второй фильтр. Данный параметр является коэффициентом децимации.

Принятые 16-разрядные данные (упакованные в 128-разрядные слова) представляют собой действительные числа в формате fractional, т.е. знаковые дробные числа K со значением в диапазоне $-1 < K < 1$. Принятое число перемножается на 16-разрядные коэффициенты из таблицы. Один коэффициент представляет собой значение синуса, а второй – значение косинуса. При перемножении получается 32-разрядное число MUL, но итоговый результат 16-разрядный, округленный до ближайшего четного числа

При извлечении 16-разрядного числа из упакованного 128-разрядного порядок следования отсчетов (начиная со старших слов или с младших) в 128-разрядном слове определяет бит TBD.

При выполнении обработки чисел с помощью встроенных фильтров итоговый результат имеет разрядность 86 бит. Для того чтобы преобразовать конечный результат обратно в формат 16-разрядного числа используется сдвиговое устройство, которое выполняет сдвиг входного 86-разрядного результата вправо на значение, заданное полем SHFR.

После сдвига для выходного результата можно задать режим округления с помощью бита ROUND:

1 если бит равен нулю, то в качестве результата используется 16 младших бит результата сдвига;

2 если бит равен единице – используется округление. Тип округления – biased, т.е. к старшему выдвинутому биту при округлении добавляется значение 1. Если установлен в «1» бит SAT, то при формировании результата учитывается факт его переполнения:

– если число больше, чем максимальное 16-разрядное положительное число, то итоговый результат равен 0x7FFF;

– если число меньше максимального отрицательного числа, то итоговый результат равен 0x8000.

Два полученных 16-разрядных результата образуют одно 32-разрядное комплексное число, которое упаковывается в 128-разрядные слова и записывается в выходной буфер. При этом существует возможность с помощью бита IQ_QI задать расположение мнимой и действительной частей в 32-разрядном слове. Выходной буфер формирует запрос к контроллеру DMAна обслуживание.

В режиме DOWN можно задать бит источника данных LINK равным нулю. В этом случае данные могут быть загружены с шины периферийных устройств процессором.

После определения всех параметров модуль необходимо включить, установив бит EN в «1».

28.1.1.2 *Режим UP*

Режим работы UP задается, если бит DAM равен единице. В этом случае модуль принимает данные от процессора (nDMA), производит процедуру интерполяции и выдает данные в передатчик Link-порта. Назначение бит в основном аналогично назначениям при работе в режиме DOWN. Параметр Kdelay в режиме UP является коэффициентом интерполяции. Значение коэффициента определяет количество отсчетов, сформированных выходным фильтром (filter_ADD) до момента обновления его входного буфера результатом из входного фильтра. Бит TBD также позволяет определить, с какого 32-разрядного слова, упакованного в 128-разрядное квадрослово, нужно начинать обработку. Бит выбора Link-порта LNKUSE задет номер порта, в который осуществляется выдача информации.

28.1.2 Регистр состояния SR

Регистр состояния SR позволяет анализировать состояние модуля. Назначение бит регистра приведено в таблице 297.

Таблица 297 – Регистр состояния

Бит	Название	Функция
0	T_req	Флаг готовности модуля к приему данных: 1 – модуль готов принять новые данные; 0 – модуль не готов принять новые данные
1	R_req	Флаг готовности модуля выдать данные: 1 – модуль готов выдать данные; 0 – модуль не готов выдать данные
2	Ferr	Флаг состояния выходного FIFO: 1 – переполнение; 0 – нет переполнения
3	overf	Флаг состояния входного FIFO: 1 – переполнение; 0 – нет переполнения

Флаг T_req формируется модулем в случае, когда входное FIFO может принять входные данные в режиме UP. В ответ на запрос канал DMA может выполнить пересылку нового квадрослова в приемный буфер.

Флаг R_req формирует запрос к контроллеру DMA в случае, когда в его выходном FIFO (в режиме DOWN) имеются данные. Это позволяет каналу DMA, прочитать данные и отослать их в память процессора.

Флаг Ferr указывает на то, что при работе в режиме DOWN выходное FIFO переполнилось, т.е. скорость потока обработанных данных выше, чем скорость считывания данных контроллером DMA.

Флаг overf устанавливается, если поток данных от приемника Link-порта превышает скорость обработки данных в модуле децимации.

28.1.3 Регистр шага STEP

Регистр шага STEP – 16-разрядный регистр, который содержит приращение счетчика для выбора констант таблицы коэффициентов. Начальное значение регистра равно нулю. Значение регистра модифицируется как записью непосредственно в него, так и записью в регистр RCNT (см. ниже).

Рекомендуемое значение STEP = 65536•$F_{ПЧ}/F_{SC}$,

где $F_{ПЧ}$ – промежуточная частота;

F_{SC} – частота дискретизации.

28.1.4 Регистр счетчика отсчетов RCNT

Запись в регистр RCNT всегда осуществляется 64-разрядным словом. Младшие 16 бит слова записываются в регистр STEP, а старшие 32 разряда записываются в регистр RCNT. Регистр RCNT имеет буферный регистр и счетный регистр. При записи по адресу 8, значение записывается сразу в RCNT_cnt и RCNT_buf. Счетчик RCNT_cnt может осуществлять вычитание единицы из своего значения по сигналу модуля. Вычитание происходит в следующих случаях:

– при работе в режиме DOWN в момент, когда в выходное FIFO записывается 128-разрядное слово;

– при работе в режиме UP в момент считывания 128-разрядного слова из входного FIFO для обработки.

В момент, когда значение счетчика становится равным нулю, происходит формирование запроса на прерывание к процессору. Таким образом, модуль может проинформировать процессор о том, что заданное количество квадрослов принято или отправлено. Если бит RCNT_ON регистра управления установлен, после достижения счетчиком значения ноль счетчик перезагружается начальным значением из RCNT_buf и начинает обратный отсчет. Поскольку модуль всегда работает с каналом DMA, можно также использовать прерывание от контроллера DMA для информирования процессора о передаче заданного количества данных.

При чтении по адресу 8 будет возвращено значение {RCNT, 0x0000, STEP}.

28.1.5 Регистр XCR

Назначение бит регистра приведено в таблице 298.

Таблица 298 – Регистр XCR

Бит	Название	Значение по сбросу	Функция
0	SYNC_SEL	0	Выбор канала синхронизации и инициализации модулятора: 0 – сигналы L0ACKI, L0BCMPI; 1 – сигналы L1ACKI, L1BCMPI
1	EN_COG_SYNC	0	Разрешение внешней синхронизации модулятора от вывода L#ACKI: 1 – синхронизация разрешена; 0 – синхронизация запрещена
2	EN_SYNC_RES	0	Разрешение сброса интеграторов/дециматоров от вывода L#BCMPI: 1 – сброс разрешен; 0 – сброс запрещен
3	EN_COG_PACK	0	Разрешение работы канала DMA: 1 – DMA разрешен; 0 – DMA запрещен
4	EN_INT_ACK	0	Разрешение прерывания по входу L#ACKI: 1 – прерывание разрешено; 0 – прерывание запрещено
5	EN_INT_BCMP	0	Разрешение прерывания по входу L#BCMPI: 1 – прерывание разрешено; 0 – прерывание запрещено
6	EN_INT_RFIN	0	Разрешение прерывания по обнулению счетчика RCNT: 1 – прерывание разрешено; 0 – прерывание запрещено
7	-	0	Зарезервирован
8	SC_CLR_ACK	0	Разрешение сброса цифрового смесителя по L#ACKI
9	ADD_CLR_ACK	0	Разрешение сброса сумматора по L#ACKI
10	SUB_CLR_ACK	0	Разрешение сброса фильтра по L#ACKI
11	-	0	Зарезервирован
12	SC_CLR_BCMP	0	Разрешение сброса цифрового смесителя по сигналу L#BCMPI
13	ADD_CLR_BCMP	0	Разрешение сброса сумматора по сигналу L#BCMPI
14	SUB_CLR_BCMP	0	Разрешение сброса фильтра по сигналу L#BCMPI
15	EN_LOAD_BCMP	0	Разрешение загрузки начального значения счетчика RCNT внешним сигналом: 1 – L#BCMPI или L#ACKI; 0 – L#ACKI

Регистр предоставляет возможность синхронизировать несколько модулей UP/DOWN при построении системы, включающей в себя несколько АЦП или даже процессоров. Используя внешние выводы L0ACKI, L0BCMPI, L1ACKI, L1BCMPI можно инициировать захват данных одновременно на нескольких процессорах в нужное время.

Для использования этой возможности необходимо установить в «1» бит EN_COG_PACK и выбрать нужную пару сигналов L#ACKI, L#BCMPI посредством бита SYNC_SEL и один из сигналов L#ACKI, L#BCMPI посредством бита EN_LOAD_BCMPI. Далее необходимо в нужное пользователю время подать импульсы длительностью не менее одного такта SOC-шины, что приведет, в зависимости от значения регистра XCR, к загрузке начальных значений в счетчик RCNT, указатель коэффициентов смесителя, обнулению сумматоров и дифференциаторов фильтров. Следует иметь в виду, что при этом указатель в буфере порта связи не обнуляется, поэтому данные для обработки на DOWN-конвертеры выровнены с точностью до одного квадрослова.

28.1.5.1 *Механизм синхронизации модулей UP/DOWN (для микросхем с ревизии 3)*

Схему синхронизации модулей см. на рисунке 153. Добавлен бит 7 регистра LRCTL# и регистр CFG12 Модуля управления синхронизацией и энергопотреблением. Также добавлен аппаратный модуль синхронизации L#SYNC, который вместе со сбросом регистров модуля UP/DOWN обеспечивает синхронный сброс и регистров портов связи, обеспечивая таким образом истинную когерентность нескольких микросхем.

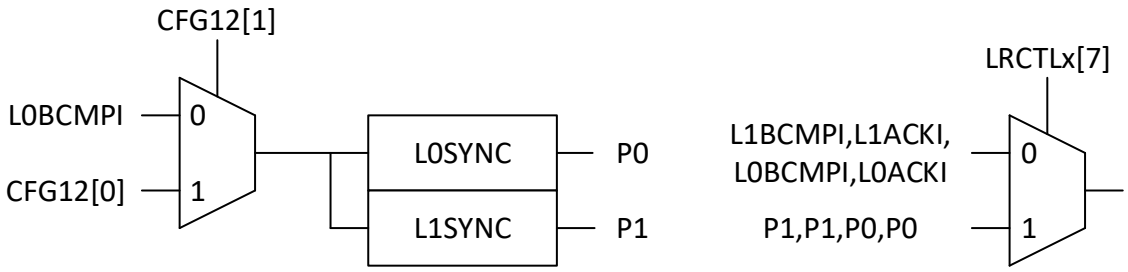


Рисунок 153 – Схема синхронизации модулей UP/DOWN

В связи с этими изменениями описание регистра XCR будет следующим:

Таблица 299 – Регистр XCR

Бит	Название	Значение по сбросу	Функция
0	SYNC_SEL	0	Выбор канала синхронизации и инициализации модулятора: 0 – сигнал P0; 1 – сигнал P1
1	EN_COG_SYNC	0	Разрешение внешней синхронизации модулятора от сигнала P#: 1 – синхрониация разрешена; 0 – синхронизация запрещена
2	EN_SYNC_RES	0	Разрешение сброса интеграторов/дециматоров от сигнала P#: 1 – сброс разрешен; 0 – сброс запрещен

Бит	Название	Значение по сбросу	Функция
3	EN_COG_PACK	0	Разрешение работы канала DMA: 1 – DMA разрешен; 0 – DMA запрещен
4	EN_INT_ACK	0	Разрешение прерывания по сигналу P#: 1 – прерывание разрешено; 0 – прерывание запрещено
5	EN_INT_BCMP	0	Разрешение прерывания по сигналу P#: 1 – прерывание разрешено; 0 – прерывание запрещено
6	EN_INT_RFIN	0	Разрешение прерывания по обнулению счетчика RCNT: 1 – прерывание разрешено; 0 – прерывание запрещено
7	-	0	Зарезервирован
8	SC_CLR_ACK	0	Разрешение сброса цифрового смесителя по сигналу P#
9	ADD_CLR_ACK	0	Разрешение сброса сумматора по сигналу P#
10	SUB_CLR_ACK	0	Разрешение сброса фильтра по сигналу P#
11	-	0	Зарезервирован
12	SC_CLR_BCMP	0	Разрешение сброса цифрового смесителя по сигналу P#
13	ADD_CLR_BCMP	0	Разрешение сброса сумматора по сигналу P#
14	SUB_CLR_BCMP	0	Разрешение сброса фильтра по сигналу P#
15	EN_LOAD_BCMP	0	Не имеет смысла

28.2 Подключение модулей UP/DOWN в системе

С помощью модуля можно принимать данные и производить предварительную обработку. Входные данные могут быть получены только с использованием приемников Link-портов. В процессоре имеется четыре модуля UP/DOWN. Структура системы при работе в режиме DOWN представлена на рисунке 154.

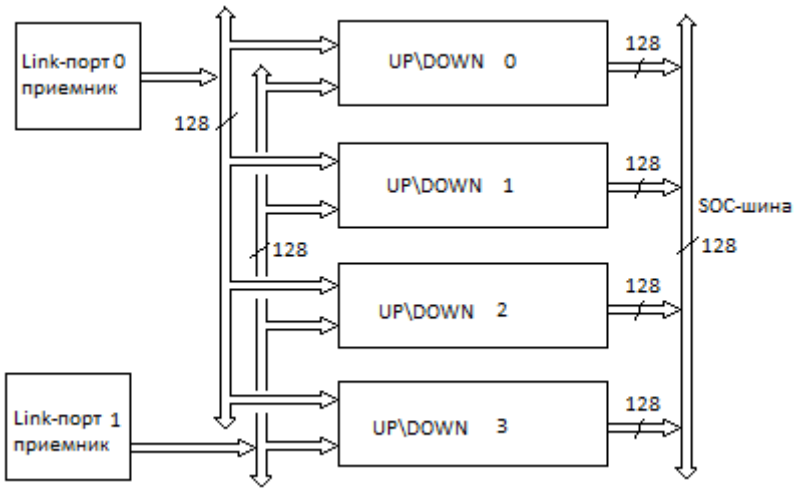


Рисунок 154 – Подключение модуля в режиме DOWN

Как следует из рисунка 152, каждый из приемников Link-портов может передавать данные модулю для обработки. Выбор конкретного порта осуществляется программированием регистра управления CR. При этом сам модуль Link-порта настраивается на прием данных не в собственный буфер, а в модуль UP/DOWN.

При включении системы желательна следующая последовательность действий:

– выполнение настроек и включение модуля UP/DOWN;
– программирование канала DMA на работу с модулем UP/DOWN;
– включение приемника Link-порта.

Модуль обработки находится в постоянном ожидании данных от приемника Link-порта. Как только приемник включается и начинает принимать данные, модуль начинает обработку принятых данных. При этом с одним и тем же приемником могут работать несколько модулей UP/DOWN, которые могут иметь различные настройки.

При работе модуля в режиме UP структура связей (рисунок 155) отражает прием данных от контроллера DMA, обработку данных и передачу в передатчик порта связи.

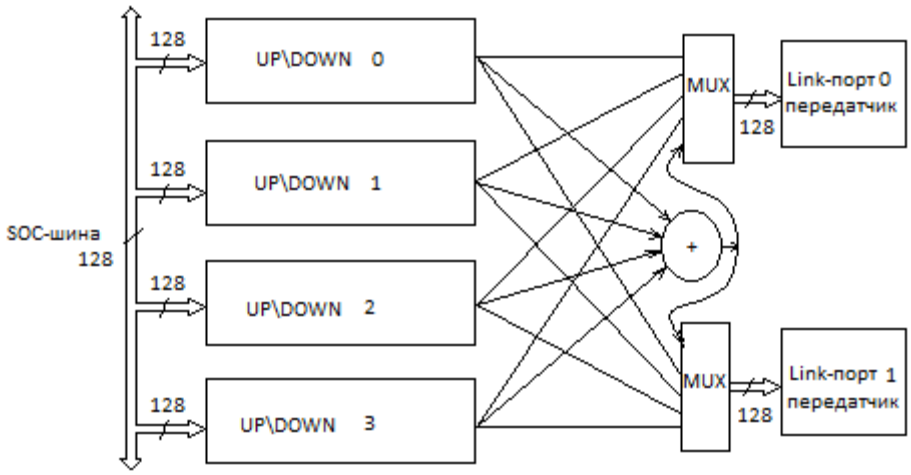


Рисунок 155 – Подключение модуля в режиме UP

Каждый из четырех модулей может быть подключен к передатчику одного из портов связи. Выбор номера порта связи выполняется в регистре управления. Одновременно к передатчику может быть подключен только один модуль. Если ошибочно несколько модулей сконфигурированы для подключения к одному и тому же передатчику, преимущество в подключении имеет модуль 0, затем 1, 2 и 3. Особым образом обрабатывается случай, когда все четыре модуля одновременно подключены к одному и тому же передатчику. В этом случае на передатчик подается сумма выдаваемых значений из всех четырех модулей. Значение суммы округляется и восемь 16-разрядных значений подаются в передатчик выбранного порта. Непосредственное подключение данных от модулей к передатчику выполняется при программировании Link-порта.

29 Интерфейс Ethernet 10/100/1000 (для микросхем с ревизии 3)

29.1 Структурная схема контроллера интерфейса

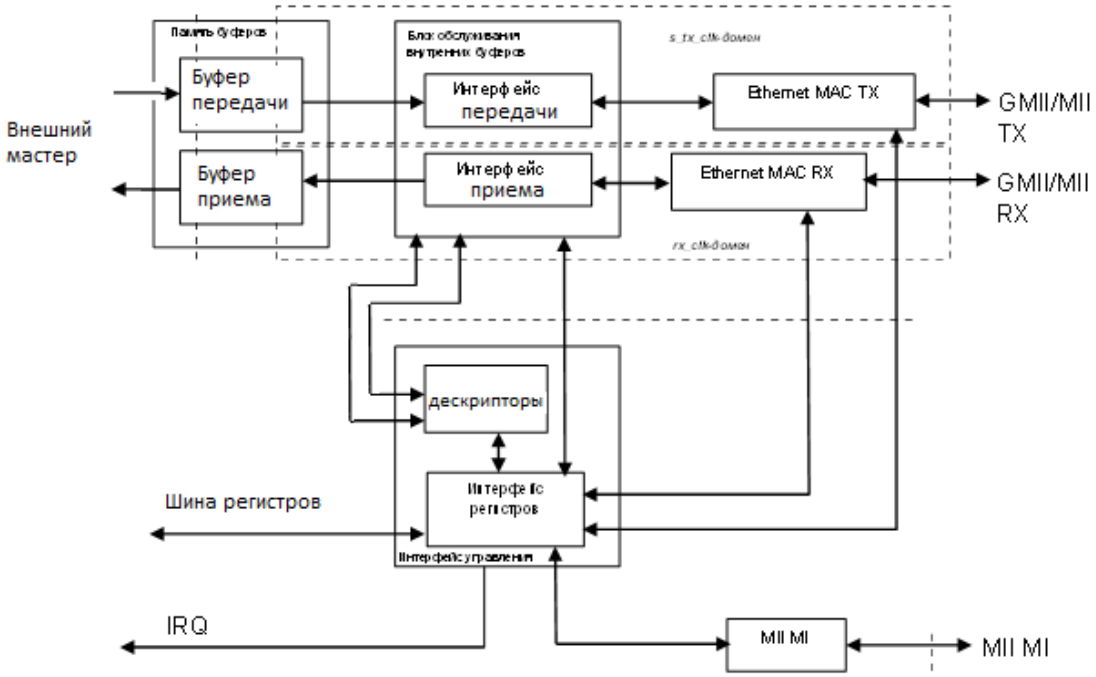


Рисунок 156 – Структурная схема контроллера интерфейса

Контроллер Ethernet MAC состоит из следующих подблоков:

– память буферов;
– интерфейс управления;
– обслуживание внутренних буферов;
– Ethernet MAC TX (функция передачи пакетов);
– Ethernet MAC RX (функция приёма пакетов);
– MII MI (интерфейс MII Management Interface для настройки PHY).

Подблок памяти буферов состоит из двух однопортовых блоков памяти типа SRAM. Данный подблок обеспечивает временное хранилище фреймов внутри блока Ethernet MAC (для TX и RX раздельно). Каждый модуль памяти имеет размер 16 Кбайт и логически разделен на два подмодуля по 8К байт.

Подблок обслуживания внутренних буферов постоянно сканирует регистры внутренних дескрипторов на предмет появления в них новых команд отправки фреймов по TX и команд разрешения получения фреймов по RX. Подблок обслуживания внутренних буферов инициирует процесс передачи фреймов по TX из буферов TX, и инициирует процесс получения фреймов по RX в буферы RX.

Подблок интерфейса управления использует периферийную шину для чтения и записи регистров и дескрипторов со стороны центрального процессора. Интерфейс управления также предоставляет сигнал прерывания для контроллера прерываний процессора, а также сигналы запроса на обслуживание для системного DMA контроллера.

Подблок Ethernet MAC TX генерирует исходящие байтовые потоки данных типа Ethernet 1000BASE-T GMII или исходящие полубайтовые потоки данных

типа Ethernet 10BASE-T/100BASE-TX MII на основе фреймов, предоставляемых интерфейсом данных TX. Подблок Ethernet MAC TX осуществляет требуемые процедуры отложенной передачи (Deference) и процедуры временного останова передачи (BackOff), самостоятельно генерирует межпакетные символы (inter-packet gaps), самостоятельно генерирует контрольные суммы пакетов (FCS) и осуществляет контроль за физической средой передачи (мониторинг GMII/MII сигналов Carrier Sense и Collision).

Подблок Ethernet MAC RX воспринимает входящие байтовые потоки данных типа Ethernet 1000BASE-T GMII или входящие полубайтовые потоки данных типа Ethernet 10BASE-T/100BASE-TX MII, передавая данные принятых пакетов подблоку интерфейса данных RX. Подблок Ethernet MAC RX ожидает начала передачи пакета по маркеру SFD (Start Frame Delimiter) в начале пакета, проверяет контрольные суммы пакетов FCS, обнаруживает лишние полубайты в потоке приёма фрейма (dribble nibbles) и определяет другие типовые ошибки Ethernet-передачи.

Назначением подблока MII MI является контроль и управление за состоянием PHY или других совместимых с MII Management Interface устройств (например, RGMII PHY). Подблок MII MI доступен через регистры подблока Register and Descriptor Interface.

Подключение к внешним микросхемам физического уровня PHY реализовано посредством интерфейса RGMII. Блок управления RGMII находится снаружи блока MAC.

29.2 Регистры интерфейса.

Для пользователя модуль Ethernet MAC представляет собой набор программно-адресуемых регистров и модулей памяти. При инсталляции в процессор К1967ВН044, модуль имеет карту памяти, представленную в таблице 300.

Таблица 300 – Карта памяти

Начальный адрес	Конечный адрес	Описание
MAC 0		
0x0300_0000	0x0300_07FF	Буфер 0 передатчика (8К байт)
0x0300_0800	0x0300_0FFF	Буфер 1 передатчика (8К байт)
0x0300_1000	0x0300_3FFF	зарезервировано
0x0300_4000	0x0300_7FFF	Буфер FIFO передатчика TX0_FIFO_DATA
0x0300_8000	0x0300_87FF	Буфер 0 приемника (8К байт)
0x0300_8800	0x0300_8FFF	Буфер 1 приемника (8К байт)
0x0300_9000	0x0300_BFFF	зарезервировано
0x0300_C000	0x0300_FFFF	Буфер FIFO приемника RX0_FIFO_DATA
0xC000_7800	0xC000_78FF	Регистры управления и статуса
0xC000_7900	0xC000_79FF	Дескрипторы передатчика
0xC000_7A00	0xC000_7AFF	Дескрипторы приемника
MAC 1		
0x0301_0000	0x0300_07FF	Буфер 0 передатчика (8К байт)

Начальный адрес	Конечный адрес	Описание
0x0301_0800	0x0300_0FFF	Буфер 1 передатчика (8К байт)
0x0301_1000	0x0300_3FFF	зарезервировано
0x0301_4000	0x0300_7FFF	Буфер FIFO передатчика TX1_FIFO_DATA
0x0301_8000	0x0300_87FF	Буфер 0 приемника (8К байт)
0x0301_8800	0x0300_8FFF	Буфер 1 приемника (8К байт)
0x0301_9000	0x0300_BFFF	зарезервировано
0x0301_C000	0x0300_FFFF	Буфер FIFO приемника RX1_FIFO_DATA
0xC000_7C00	0xC000_7CFF	Регистры управления и статуса
0xC000_7D00	0xC000_7DFF	Дескрипторы передатчика
0xC000_7E00	0xC000_7EFF	Дескрипторы приемника

Приемник и передатчик имеют по два буфера. В текущий момент времени приемник или передатчик может работать с одним буфером, в то время как со вторым буфером может работать внешний мастер. К буферам возможен как прямой доступ, так и доступ в режиме FIFO. В режиме FIFO используется специальная область адресного пространства, при обращении в которую специальный внутренний указатель выбирает, с каким из буферов (0 или 1) в текущий момент будут производиться действия. Этот механизм позволяет упростить работу с интерфейсом.

Таблица 301 содержит названия и адреса всех программно-доступных регистров Ethernet MAC. Поле «Адрес» указывает относительный адрес регистра в шестнадцатеричной форме. Полный адрес регистра Ethernet MAC рассчитывается как (базовый адрес + относительный адрес), где базовый адрес равен 0xC000_7800(MAC0) либо 0xC000_7C00(MAC1). Ширина всех программно-доступных регистров блока Ethernet MAC составляет 32 разряда.

Таблица 301 – Карта регистров Ethernet MAC

Название регистра	Адрес	Доступ	Общее описание регистра
MODE	0x000	R/W	Регистр задания режима работы
STATUS	0x001	R/O	Регистр статуса
THIS_MAC_STATION_ ADDRESS_0 THIS_MAC_STATION_ ADDRESS_1	0x002 0x003	R/W	Регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 содержат MAC-адрес в каноническом формате
MULTICAST_ADDRESS_ HASH_TABLE_0 до MULTICAST_ADDRESS_ HASH_TABLE_15	0x004 до 0x013	R/W	Регистры от MULTICAST_ADDRESS_HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15 содержат таблицу хешей multicast-адресов
MII_MI_MODE	0x014	R/W	Регистр режима MII Management Interface

Название регистра	Адрес	Доступ	Общее описание регистра
MII_MI_ADDR	0x015	R/W	Регистр адреса MII Management Interface
MII_MI_TX_DATA	0x016	R/W	Регистр данных для передачи по MII Management Interface
MII_MI_COMMAND	0x017	W/O	Командный регистр MII Management Interface
MII_MI_STATUS	0x018	R/O	Регистр статуса MII Management Interface
MII_MI_RX_DATA	0x019	R/O	Регистр принятых данных по MII Management Interface
BACKOFF_RANDOM_SEED	0x01a	R/W	Регистр случайного начального заполнения для процедуры BackOff
INTERRUPT_SOURCE	0x01b	R/W	Регистр источников прерывания
INTERRUPT_MASK	0x01c	R/W	Регистр маски прерывания
-	-	-	Зарезервировано
SOFTWARE_RESET	0x01E	W/O	Регистр программного сброса
LATE_COLLISIONS_COUNTER	0x01F	R/O	Регистр счётчика Late Collisions Counter
MAC_CONTROL	0x020	R/W	Регистр MAC Control
-	-	-	Зарезервировано
MAC_FIFO_MODE	0x030	R/W	Разрешение прерываний от FIFO интерфейса
MAC_FIFO_MODE_SET	0x031	W/O	Установка бит регистра: установка в 1; 0 – без изменений
MAC_FIFO_MODE_CLR	0x032	W/O	Сброс бит регистра: сброс в 0; 0 – без изменений
SGMII_CTRL	0x033	R/W	Регистр управления интерфейсом SGMII
MAC_FIFO_STATUS	0x034	R/O	Состояние FIFO интерфейса
MAC_FIFO_IRQ	0x035	R/O	Состояние запросов прерываний
Rx_STATUS	0x036	R/O	Регистр состояния FIFO приемника
TX_RX_CNT	0x37	R/O	Счетчик количества фреймов
RX_DFIFO_info	0x38	R/O	Значение указателей буфера данных приемника
QW_CNT	0x39	R/O	Счетчик количества квадрослов
-	-	-	Зарезервировано

Название регистра	Адрес	Доступ	Общее описание регистра
Rx_STATUS_POP_ACT	0x3E	R/O	Регистр состояния FIFO приемника. При чтении счетчик фреймов увеличивается на 1 и также активизируется бит ACT соответствующего (RXP) дескриптора приемника
Rx_STATUS_POP	0x3F	R/O	Регистр состояния FIFO приемника. При чтении счетчик фреймов увеличивается на 1
-	-	-	Зарезервировано
DESC_TX0	0x100	R/W	Дескриптор 0 передатчика
DESC_TX1	0x101	R/W	Дескриптор 1 передатчика
DESC_TX_FIFO	0x180	R/W	Дескриптор FIFO передатчика
-	-	-	Зарезервировано
DESC_RX0	0x200	R/W	Дескриптор 0 приемника
DESC_RX1	0x201	R/W	Дескриптор 1 приемника
DESC_RX_FIFO	0x280	R/W	Дескриптор FIFO приемника
Обозначения: – R/W – чтение и запись; – R/O – только чтение (запись не имеет эффекта); – W/O – только запись (читается как 0)			

Каждый из двух буферов передатчика и приемника имеет ассоциированный с ним регистр-дескриптор. Аналогично буферной памяти, имеется специальный адрес, позволяющий обращаться к регистру дескриптора в режиме FIFO. Наличие доступа в режиме FIFO позволяет упростить работу интерфейса с системным контроллером DMA. Интерфейс передатчика (либо приемника) можно представить, как буфер FIFO имеющий 2 входа (записи). Вход представляет собой буфер памяти и соответствующий ему дескриптор.

При симуляции и синтезе проекта в файле конфигурации config.v необходимо установить параметры:

```
`define P_NUMBER_OF_OUTER_TX_DESCRIPTORS 11'd2
`define P_OUTER_TX_DESC_MEM_ADDR3_SIZE 1

`define P_NUMBER_OF_OUTER_RX_DESCRIPTORS 11'd2
`define P_OUTER_RX_DESC_MEM_ADDR3_SIZE 1
```

29.2.1 Регистр MODE

Относительный адрес: 0x00.
Доступ: R/W.
Значение после сброса: 0.

Регистр MODE (таблица 302) предназначен для задания режимов работы блока Ethernet MAC.

После записи данных в этот регистр, данные из этого регистра начинают передаваться в другие части системы Ethernet MAC, а разряд 2 регистра STATUS (индикатор окончания конфигурации регистра) будет иметь значение 1 до окончания передачи данных из этого регистра. Пока значение разряда 2 регистра STATUS не перейдёт в значение 0, повторная запись данного регистра не будет иметь эффекта.

Таблица 302 – Регистр MODE

Биты	Назначение
31:13	Зарезервировано
12:9	TX: InterPacketGapPart1[3:0] (значение в байтах), должно содержать число от 0 до 8. Для InterPacketGapPart1 > 8, будет записано значение 8
8:7	TX и RX: Режим скорости: 00 – 1000 Мбит/с; 01 – 10 Мбит/с; 10 – 1000 Мбит/с; 11 – 100 Мбит/с
6	TX и RX: разряд включения режима Half Duplex: 1 – включено; 0 – выключено
5	RX: разряд включения режима PassReceiveFCSMode: 1 – включено; 0 – выключено
4	RX: разряд включения режима Promiscuous Mode: 1 – включено; 0 – выключено
3	RX: разряд включения режима Multicast: 1 – включено; 0 – выключено
2	RX: Receive Enable: 1 – включается получение последующих входящих фреймов по Ethernet MAC RX; 0 – текущая передача по Ethernet MAC RX будет завершена (если она идёт в данный момент), и после этого получение передач по RX будет отключено
1	TX: разряд разрешения режима Burst Mode: 1 – разрешено; 0 – запрещено
0	TX: Transmit Enable: 1 – включается отправка последующих исходящих фреймов по Ethernet MAC TX; 0 – текущая передача по Ethernet MAC TX будет завершена (если она идёт в данный момент), и после этого отправка последующих исходящих фреймов по TX будет отключена

Разряды 0 «TX: Transmit Enable» и 2 «RX: Receive Enable» могут быть установлены в значение 0 в любое время при необходимости. Чтобы изменить любые другие поля регистра MODE или регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1, необходимо выполнить следующую последовательность действий.

После сброса Ethernet MAC:

1 Изменить значение регистров THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 и после этого изменить значение регистра MODE;

2 Установить поля 0 «TX: Transmit Enable» и 2 «RX: Receive Enable» в значение 1, после чего можно начинать отправку и приём новых фреймов по TX и RX соответственно.

Во время активности Ethernet MAC (поля 0 «TX: Transmit Enable» и 2 «RX: Receive Enable» в значении 1):

1 Прекратить добавлять новые фреймы на отправку по TX;

2 Установить поля 0 «TX: Transmit Enable» и 2 «RX: Receive Enable» в значение 0;

3 Подождать до тех пор, пока передачи по TX и RX не завершатся: пока поля 0 и 1 регистра STATUS не будут равны 0;

4 Изменить значение регистров THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 и после этого изменить значение регистра MODE;

5 Установить поля 0 «TX: Transmit Enable» и 2 «RX: Receive Enable» в значение 1, после чего можно начинать отправку и приём новых фреймов по TX и RX соответственно.

29.2.2 Регистр STATUS

Относительный адрес: 0x01.

Доступ: R/O.

Значение после сброса: 0x0.

Регистр STATUS (таблица 303) предназначен для отображения статуса блока Ethernet MAC.

После записи некоторого регистра Ethernet MAC, один из индикаторов, соответствующий этому регистру (биты 2-23 регистра STATUS) устанавливается в значение 1 и держится в значении 1 до тех пор, пока данный регистр не будет окончательно сконфигурирован, после чего значение данного разряда сбрасывается в 0. В то время, пока значение этого разряда равно 1, повторная запись в такой регистр не будет иметь эффекта.

Таблица 303 – Регистр STATUS

Биты	Назначение
31:24	Зарезервировано
23	Индикатор окончания конфигурации регистра MAC_CONTROL: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи

Биты	Назначение
22	Индикатор окончания конфигурации регистра DATA_BUS_CONFIG: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
21	Индикатор окончания конфигурации регистра BACKOFF_RANDOM_SEED: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
20	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_15: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
19	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_14: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
18	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_13: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
17	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_12: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
16	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_11: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
15	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_10: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
14	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_9: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи

Биты	Назначение
13	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_8: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
12	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_7: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
11	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_6: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
10	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_5: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
9	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_4: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
8	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_3: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
7	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_2: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
6	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_1: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
5	Индикатор окончания конфигурации регистра MULTICAST_ADDRESS_HASH_TABLE_0: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи

Биты	Назначение
4	Индикатор окончания конфигурации регистра THIS_MAC_STATION_ADDRESS_1: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
3	Индикатор окончания конфигурации регистра THIS_MAC_STATION_ADDRESS_0: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи
2	Индикатор окончания конфигурации регистра MODE: 1 – регистр в процессе конфигурации и очередная запись в этот регистр не будет иметь эффекта; 0 – этот регистр готов к перезаписи. Следует иметь в виду, что при использовании интерфейса RGMII, бит будет установлен в 0, только когда в системе присутствует, в том числе и синхросигнал приемника
1	RX Busy: 1 – Ethernet MAC RX в работе; 0 – Ethernet MAC RX в режиме ожидания
0	TX Busy: 1 – Ethernet MAC TX в работе; 0 – Ethernet MAC TX в режиме ожидания

29.2.3 Регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1

Относительные адреса:

– THIS_MAC_STATION_ADDRESS_0: 0x02;

– THIS_MAC_STATION_ADDRESS_1: 0x03.

Доступ: R/W.

Значения после сброса: 0x0 (для обоих регистров).

Регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 содержат MAC-адрес в каноническом формате (таблица 304).

После записи данных в один из регистров этой группы, данные из этого регистра начинают передаваться в другие части системы Ethernet MAC, а один из разрядов регистра STATUS (соответствующий индикатор окончания конфигурации регистра, разряды 3 и 4 регистра STATUS) будет иметь значение 1 до окончания передачи данных из этого регистра. Пока значение этого разряда регистра STATUS не перейдёт в значение 0, повторная запись данного регистра не будет иметь эффекта.

Таблица 304 – Регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1

Регистры и их поля битов	Назначение
THIS_MAC_STATION_ADDRESS_1[31:16]	Зарезервировано

THIS_MAC_STATION_ADDRESS_1[7:0]	MAC-адрес, байт 5
THIS_MAC_STATION_ADDRESS_1[15:8]	MAC-адрес, байт 4
THIS_MAC_STATION_ADDRESS_0[7:0]	MAC-адрес, байт 3
THIS_MAC_STATION_ADDRESS_0[15:8]	MAC-адрес, байт 2
THIS_MAC_STATION_ADDRESS_0[23:16]	MAC-адрес, байт 1
THIS_MAC_STATION_ADDRESS_0[31:24]	MAC-адрес, байт 0

В стандарте IEEE 802.3 байты (октеты) посылаются по линии связи слева направо, начиная с младшего байта. Например, MAC-адрес в каноническом формате 12-34-56-78-9A-BC (байт 0 крайний слева) будет отправлен по линии связи как разряды 01001000 00101100 01101010 00011110 01011001 00111101. Если по линиям отправляется по байту за раз, то передача будет в следующем порядке: 12, 34, 56, 78, 9A, BC. TX отправляет (и RX принимает) MAC-адрес начиная с байта 0 и заканчивая байтом 5.

Правила изменения регистров THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 описаны в пункте 29.2.1 «Регистр MODE».

29.2.4 Регистры MULTICAST_ADDRESS_HASH_TABLE_0 – MULTICAST_ADDRESS_HASH_TABLE_15

Относительные адреса:
– MULTICAST_ADDRESS_HASH_TABLE_0: 0x4;
– MULTICAST_ADDRESS_HASH_TABLE_1: 0x5;
– MULTICAST_ADDRESS_HASH_TABLE_2: 0x6;
– MULTICAST_ADDRESS_HASH_TABLE_3: 0x7;
– MULTICAST_ADDRESS_HASH_TABLE_4: 0x8;
– MULTICAST_ADDRESS_HASH_TABLE_5: 0x9;
– MULTICAST_ADDRESS_HASH_TABLE_6: 0xA;
– MULTICAST_ADDRESS_HASH_TABLE_7: 0xB;
– MULTICAST_ADDRESS_HASH_TABLE_8: 0xC;
– MULTICAST_ADDRESS_HASH_TABLE_9: 0xD;
– MULTICAST_ADDRESS_HASH_TABLE_10: 0xE;
– MULTICAST_ADDRESS_HASH_TABLE_11: 0xF;
– MULTICAST_ADDRESS_HASH_TABLE_12: 0x10;
– MULTICAST_ADDRESS_HASH_TABLE_13: 0x11;
– MULTICAST_ADDRESS_HASH_TABLE_14: 0x12;
– MULTICAST_ADDRESS_HASH_TABLE_15: 0x13.

Доступ: R/W.

Значения после сброса: 0x0 (для всех MULTICAST_ADDRESS_ HASH_TABLE_*).

Регистры MULTICAST_ADDRESS_HASH_TABLE_0 – MULTICAST_ ADDRESS_HASH_TABLE_15 предназначены для функции фильтрации

Multicast-адресов. В этих регистрах содержатся части таблицы хешей multicast-адресов (multicast_address_hash_table[511:0], таблица).

После записи данных в один из регистров этой группы, данные из этого регистра начинают передаваться в другие части системы Ethernet MAC, а один из разрядов регистра STATUS (соответствующий индикатор окончания конфигурации регистра, разряды 5-20 регистра STATUS) будет иметь значение 1 до окончания передачи данных из этого регистра. Пока значение этого разряда регистра STATUS не перейдёт в значение 0, повторная запись данного регистра не будет иметь эффекта.

Таблица 305 – Регистры MULTICAST_ADDRESS_HASH_TABLE_0 – MULTICAST_ADDRESS_HASH_TABLE_15

Регистр	Назначение
MULTICAST_ADDRESS_HASH_TABLE_0	multicast_address_hash_table[31:0].
MULTICAST_ADDRESS_HASH_TABLE_1	multicast_address_hash_table[63:32].
MULTICAST_ADDRESS_HASH_TABLE_2	multicast_address_hash_table[95:64].
MULTICAST_ADDRESS_HASH_TABLE_3	multicast_address_hash_table[127:96].
MULTICAST_ADDRESS_HASH_TABLE_4	multicast_address_hash_table[159:128].
MULTICAST_ADDRESS_HASH_TABLE_5	multicast_address_hash_table[191:160].
MULTICAST_ADDRESS_HASH_TABLE_6	multicast_address_hash_table[223:192].
MULTICAST_ADDRESS_HASH_TABLE_7	multicast_address_hash_table[255:224].
MULTICAST_ADDRESS_HASH_TABLE_8	multicast_address_hash_table[287:256].
MULTICAST_ADDRESS_HASH_TABLE_9	multicast_address_hash_table[319:288].
MULTICAST_ADDRESS_HASH_TABLE_10	multicast_address_hash_table[351:320].
MULTICAST_ADDRESS_HASH_TABLE_11	multicast_address_hash_table[383:352].
MULTICAST_ADDRESS_HASH_TABLE_12	multicast_address_hash_table[415:384].
MULTICAST_ADDRESS_HASH_TABLE_13	multicast_address_hash_table[447:416].
MULTICAST_ADDRESS_HASH_TABLE_14	multicast_address_hash_table[479:448].
MULTICAST_ADDRESS_HASH_TABLE_15	multicast_address_hash_table[511:480].

Во время приёма фреймов по RX модуль IEEE 802.3 CRC32 рассчитывает контрольную сумму начиная с первого принятого байта. Так как поле Destination Address составляет первые 6 байт любого фрейма, то значение CRC32 после шестого байта есть не что иное как CRC32 от поля Destination Address, в котором может быть адрес типа Multicast. Разряды [8:0] от этого значения CRC32 являются адресом таблицы хешей multicast-адресов multicast_address_hash_table[511:0]. Если адресуемое однорязрядное значение поля таблицы содержит 1, значит данный адрес Destination Address типа Multicast является разрешённым (иначе – запрещённым).

Чтобы рассчитать значения таблицы multicast_address_hash_table, нужно для каждого разрешённого адреса Destination Address типа Multicast нужно рассчитать

CRC32, взять разряды [8:0] от этих 32-разрядных значений, и по всем адресам, равным полученным 9-разрядным значениям, записать значение 1 в поле таблицы multicast_address_hash_table. Значение 0 в поле таблицы хешей означает, что адрес типа Multicast, соответствующий полю в этой таблице будет отфильтрован.

Нужно отметить, что разряды [8:0] от полученных CRC32[31:0] могут совпадать для нескольких различных адресов типа Multicast, поэтому данный способ фильтрации является неточным: некоторые запрещённые адреса могут генерировать такие же значения адреса таблицы хешей, что и некоторые разрешённые адреса. В случае наличия таких совпадений драйвер Ethernet MAC должен дополнительно проверять адреса типа Multicast.

Регистры MULTICAST_ADDRESS_HASH_TABLE_0 – MULTICAST_ ADDRESS_HASH_TABLE_15 используются только в Ethernet MAC RX. Чтобы поменять значения разрядов в этих регистрах, необходимо выполнить следующие шаги.

После сброса Ethernet MAC:

1 Изменить значение регистров MULTICAST_ADDRESS_HASH_TABLE_*.

2 Установить поле 2 «RX: Receive Enable» регистра MODE в значение 1, после чего можно начинать приём новых фреймов по RX.

Во время активности Ethernet MAC (поле 2 «RX: Receive Enable» регистра MODE в значении 1):

1 Установить поле 2 «RX: Receive Enable» регистра MODE в значение 0;

2 Подождать до тех пор, пока передачи по RX не завершатся: пока поле 1 регистра STATUS не будет равно 0;

3 Изменить значение регистров MULTICAST_ADDRESS_HASH_TABLE_*.

4 Установить поле 2 «RX: Receive Enable» регистра MODE в значение 1, после чего можно начинать приём новых фреймов по RX.

29.2.5 Регистр MII_MI_MODE

Относительный адрес: 0x14.

Доступ: R/W.

Значение после сброса: 0x64.

Регистр MII_MI_MODE (таблица 306) предназначен для задания режима работы подблока MII MI.

Таблица 306 – Регистр MII_MI_MODE

Биты	Назначение
31:10	Зарезервировано
9	mdc_enable: разрешение генерации MDC: 0 – генерация запрещена; 1 – генерация разрешена
8	no_preamble: 1 – преамбула не используется; 0 – 32-разрядная преамбула используется во время передач по интерфейсу MII MI

Биты	Назначение
7:0	clock_divider: Делитель частоты для линии MDC

Поле clock_divider (делитель частоты) должно иметь чётное значение (разряд 0 равен 0) и должно быть больше 1. При задании делителя частоты разряд 0 всегда читается как 0, и значение делителя будет считано как 2, в случае, если поле clock_divider[7:0] равно 0 или 1.

Эффективная частота MDC будет иметь значение

(SOCCLK_frequency / clock_divider) МГц.

Эффективная частота MDC не должна превышать 2,5 МГц согласно стандарту IEEE 802.3, однако некоторые PHY разрешают и большие частоты.

Значение поля clock_divider после сброса – 100, что обеспечивает частоту MDC 2,5 МГц для частоты SOCCLK_frequency 250 МГц.

Значение поля no_preamble после сброса: 0.

Значение поля mdc_enable после сброса 0.

29.2.6 Регистр MII_MI_ADDR

Относительный адрес: 0x15.

Доступ: R/W.

Значение после сброса: 0x0.

Регистр MII_MI_ADDR (таблица 307) предназначен для задания адресов фрейма MII MI.

Таблица 307 – Регистр MII_MI_ADDR

Биты	Назначение
31:10	Зарезервировано
9:5	REGAD (адрес регистра MII MI)
4:0	PHYAD (адрес PHY)

29.2.7 Регистр MII_MI_TX_DATA

Относительный адрес: 0x16.

Доступ: R/W.

Значение после сброса: 0x0.

Регистр MII_MI_TX_DATA (таблица 308) предназначен для задания поля DATA фрейма MII MI.

Таблица 308 – Регистр MII_MI_TX_DATA

Биты	Назначение
31:16	Зарезервировано
15:0	Поле данных DATA

29.2.8 Регистр MII_MI_COMMAND

Относительный адрес: 0x17.

Доступ: W/O.

Значение после сброса: 0x0.

Регистр MII_MI_COMMAND (таблица 309) предназначен для задания команды чтения или записи для подблока MII MI.

Таблица 309 – Регистр MII_MI_COMMAND

Биты	Назначение
31:2	Зарезервировано
1	Отправить фрейм чтения
0	Отправить фрейм записи

При записи регистра MII_MI_COMMAND: если разряды [1:0] равны:

01 или 11 – отправить фрейм записи;

10 – отправить фрейм чтения;

00 – нет эффекта.

При чтении данного регистра всегда возвращается значение 0.

Если подблок MII MI находится в состоянии работы (MII_MI_STATUS: разряд 0 равен 1), запись в этот регистр не будет иметь эффекта. Перед отправкой очередной команды необходимо проверить регистр MII_MI_STATUS: если MII_MI_STATUS: разряд 0 равен 0, то команда будет отправлена.

Перед отправкой новой команды записи (отправка фрейма записи), следующие регистры должны содержать правильные значения:

– MII_MI_MODE;

– MII_MI_TX_DATA;

– MII_MI_ADDR.

Перед отправкой новой команды чтения (отправка фрейма чтения), следующие регистры должны содержать правильные значения:

– MII_MI_MODE;

– MII_MI_ADDR.

После окончания выполнения запроса на отправку фрейма чтения данные, которые были прочитаны в результате этой операции, будут доступны в регистре MII_MI_RX_DATA.

29.2.9 Регистр MII_MI_STATUS

Относительный адрес: 0x18.

Доступ: R/O.

Значение после сброса: 0x0.

Регистр MII_MI_STATUS (таблица 310) предназначен для отображения статуса подблока MII MI.

Таблица 310 – Регистр MII_MI_STATUS

Биты	Назначение
31:1	Зарезервировано
0	MII MI Busy: 1: MII MI в работе, 0: MII MI готово получить новую команду

29.2.10 Регистр MII_MI_RX_DATA

Относительный адрес: 0x19.

Доступ: R/O.

Значение после сброса: 0x0.

Регистр MII_MI_RX_DATA (таблица 311) предназначен для хранения принятых в процессе чтения данных фрейма MII MI (поле DATA).

Таблица 311 – Регистр MII_MI_RX_DATA

Биты	Назначение
31:16	Зарезервировано
15:0	Полученное поле DATA

Содержимое регистра MII_MI_RX_DATA сохраняется до отправки следующей команды чтения.

29.2.11 Регистр BACKOFF_RANDOM_SEED

Относительный адрес: 0x1A.

Доступ: R/W.

Значение после сброса: 0x0.

Регистр BACKOFF_RANDOM_SEED (таблица 312) предназначен для задания разрядов [9:1] случайного начального заполнения для генератора псевдослучайных чисел на базе регистра сдвига с линейной обратной связью для процедуры BackOff. Разряд [0] случайного начального заполнения всегда равен 1.

После записи данных в этот регистр, данные из этого регистра начинают передаваться в другие части системы Ethernet MAC, а разряд 21 регистра STATUS (индикатор окончания конфигурации регистра) будет иметь значение 1 до окончания передачи данных из этого регистра. Пока значение разряда 21 регистра STATUS не перейдёт в значение 0, повторная запись данного регистра не будет иметь эффекта.

Таблица 312 – Регистр BACKOFF_RANDOM_SEED

Биты	Назначение
31:9	Зарезервировано
8:0	Разряды [9:1] случайного начального заполнения для генератора случайных чисел на базе регистра сдвига с линейной обратной связью для процедуры BackOff

29.2.12 Регистр INTERRUPT_SOURCE

Относительный адрес: 0x1B.

Доступ: R/W.

Значение после сброса: 0x0.

Регистр INTERRUPT_SOURCE (таблица 313) предназначен для отображения источника прерывания внутри блока Ethernet MAC.

Таблица 313 – Регистр INTERRUPT_SOURCE

Биты	Назначение
31:18	Зарезервировано
17:11	Поле Latest Receive Descriptor Done. По чтению: номер последнего дескриптора TX, который вызвал прерывание: это номер от 0 до 1. По записи: записываемое значение сравнивается с текущим содержимым этого поля [17:11]; если записываемое число такое же, как и текущее значение этого поля [17:11], тогда сигнал запроса на прерывание, вызванное дескриптором TX, может быть сброшен в 0 и поля [3:2] могут быть очищены; если записываемое число не совпадает с текущим значением этого поля [17:11], тогда сигнал запроса на прерывание, вызванного дескриптором TX, не будет сброшен в 0, и поля [3:2] и [17:11] не будут записаны
10:4	Поле Latest Transmit Descriptor Done. По чтению: номер последнего дескриптора RX, который вызвал прерывание: это номер от 0 до 1. По записи: записываемое значение сравнивается с текущим содержимым этого поля [10:4]; если записываемое число такое же, как и текущее значение этого поля [10:4], тогда сигнал запроса на прерывание, вызванное дескриптором RX, может быть сброшен в 0 и поля [3:2] могут быть очищены; если записываемое число не совпадает с текущим значением этого поля [10:4], тогда сигнал запроса на прерывание, вызванного дескриптором RX, не будет сброшен в 0, и поля [3:2] и [17:11] не будут записаны
3	Поле индикатора frame_receive_error. Индикатор принятого по RX фрейма с ошибкой. Очищается при записи 1 в это поле
2	Поле индикатора frame_received. Индикатор принятого по RX фрейма без ошибки. Очищается при записи 1 в это поле
1	Поле индикатора frame_transmit_error. Индикатор отправленного по TX фрейма с ошибкой. Очищается при записи 1 в это поле
0	Поле индикатора frame_transmitted indicator. Индикатор отправленного по TX фрейма без ошибки. Очищается при записи 1 в это поле

Выходной сигнал interrupt_request устанавливается в значение 1, если хотя бы один из разрядов поля [3:0] регистра INTERRUPT_SOURCE равен 1 и при этом ему соответствующий разряд регистра INTERRUPT_MASK равен 1. Как только все разряды поля [3:0] регистра INTERRUPT_SOURCE становятся равными 0, тогда сигнал interrupt_request принимает значение 0.

Регистр INTERRUPT_SOURCE содержит информацию только о последнем дескрипторе TX и последнем дескрипторе RX, которые вызвали прерывание.

29.2.13 Регистр INTERRUPT_MASK

Относительный адрес: 0x1C.

Доступ: R/W

Значение после сброса: 0x0.

Регистр INTERRUPT_MASK (таблица 314) предназначен для хранения маски разрешения прерываний для регистра INTERRUPT_SOURCE.

Таблица 314 –Регистр INTERRUPT_MASK

Биты	Назначение
31:4	Зарезервировано
3	Разрешение установки сигнала прерывания в 1 при включении индикатора frame_receive_error
2	Разрешение установки сигнала прерывания в 1 при включении индикатора frame_received
1	Разрешение установки сигнала прерывания в 1 при включении индикатора frame_transmit_error
0	Разрешение установки сигнала прерывания в 1 при включении индикатора frame_transmitted

29.2.14 Регистр SOFTWARE_RESET

Относительный адрес: 0x1E.

Доступ: W/O.

Значение после сброса: 0x0.

Регистр SOFTWARE_RESET (таблица 315) предназначен для обеспечения возможности сброса блока Ethernet MAC со стороны программного интерфейса.

Таблица 315 – Регистр SOFTWARE_RESET

Биты	Назначение
31:3	Зарезервировано
2	Программный сброс (когда 1) логики FIFO приемника
1	Программный сброс (когда 1) логики FIFO передатчика
0	Программный сброс: значение 1 по записи начинает программный сброс всего блока Ethernet MAC (включая логику FIFO)

После записи 1 в разряд 0 регистра SOFTWARE_RESET весь блок Ethernet MAC начинает процесс сброса. Запрещается инициировать любые операции чтения или записи с регистрами Ethernet MAC в течение 10 тактов шины. Биты 1 и 2 позволяют дополнительно выполнить сброс логики управления буферами FIFO раздельно для приемника и передатчика, независимо от основной части интерфейса.

29.2.15 Регистр LATE_COLLISIONS_COUNTER

Относительный адрес: 0x1F.

Доступ: R/O.

Значение после сброса: 0x0.

Регистр LATE_COLLISIONS_COUNTER (таблица 316) предназначен для счётчика aLateCollisions для функции Layer Management. Счётчик aLateCollisions является несбрасываемым счётчиком с переполнением.

Таблица 316 – Регистр LATE_COLLISIONS_COUNTER

Биты	Назначение
31:0	Счётчик aLateCollisions (для Layer Management)

29.2.16 Регистр MAC_CONTROL

Относительный адрес: 0x20.

Доступ: R/W.

Значение после сброса: 0x0.

Регистр MAC_CONTROL (таблица 317) предназначен для обеспечения функциональности IEEE 802.3 MAC Control, а именно для поддержки фреймов типа MAC Control PAUSE.

После записи данных в этот регистр, данные из этого регистра начинают передаваться в другие части системы Ethernet MAC, а разряд 23 регистра STATUS (индикатор окончания конфигурации регистра) будет иметь значение 1 до окончания передачи данных из этого регистра. Пока значение разряда 23 регистра STATUS не перейдёт в значение 0, повторная запись данного регистра не будет иметь эффекта.

Таблица 317 – Регистр MAC_CONTROL

Биты	Назначение
31:19	Зарезервировано
18	Поле MAC Control Enable. Если значение поля MAC Control Enable = 1, то по принятию фрейма типа MAC Control PAUSE по RX такой фрейм не будет зафиксирован как принятый в дескрипторе RX, а сам блок Ethernet MAC обработает этот фрейм соответствующим образом: TX будет переведён в режим паузы автоматически, на определённое запрошенное время запрещая передачу новых фреймов по TX. Если значение поля MAC Control Enable = 0, то по принятию фрейма типа MAC Control PAUSE по RX такой фрейм будет зафиксирован как принятый в дескрипторе RX как обычный фрейм с данными

Биты	Назначение
17	Поле pause_status. Индикация для процедуры PAUSE, которая была запрошена удалённым MAC: если по RX был получен корректный фрейм типа MAC Control PAUSE, то TX может быть автоматически установлен в режим паузы на запрошенное время. Пока TX находится в состоянии паузы, поле pause_status остаётся в значении 1, иначе оно имеет значение 0. Запись в этот разряд не имеет эффекта
16	Поле отправки MAC Control PAUSE. Если поле «TX и RX: разряд включения режима Half Duplex» в регистре MODE установлено как 0 (режим half duplex не используется), то запись значения 1 в поле отправки MAC Control PAUSE начнёт процедуру отправки фрейма типа MAC Control PAUSE по TX в автоматическом режиме. Поле отправки MAC Control PAUSE очищается автоматически после отправки одного фрейма типа MAC Control PAUSE. Если поле «TX и RX: разряд включения режима Half Duplex» в регистре MODE установлено как 1 (режим half duplex используется), то запись значения 1 в поле отправки MAC Control PAUSE не будет иметь никакого эффекта. Пока поле отправки MAC Control PAUSE находится в значении 1, запись в весь регистр MAC_CONTROL не будет иметь эффекта
15:0	Поле pause_time. Задаёт операнд pause_time, который помещается в отправляемый по TX фрейм типа MAC Control PAUSE

Регистр MAC_CONTROL используется для автоматической отправки одиночного фрейма типа MAC Control PAUSE.

Запрос на автоматическую отправку фрейма типа MAC Control PAUSE имеет наивысший приоритет в очереди отправки фреймов по TX. Для корректной автоматической отправки фрейма типа MAC Control PAUSE регистры THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 должны содержать правильное значение MAC-адреса. Фрейм типа MAC Control PAUSE может быть отправлен даже в том случае, если TX в данный момент находится в состоянии паузы. После отправки фрейма типа MAC Control PAUSE поле отправки MAC Control PAUSE меняет значение с 1 на 0, а по окончании отправки такого фрейма прерывание не генерируется.

Внимание: принятие фрейма типа MAC Control PAUSE с адресом типа Multicast 01:80:C2:00:00:01 возможно только в том случае, если соответствующее поле таблицы таблицы хешей multicast-адресов (регистры MULTICAST_ADDRESS_HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15) будет установлено в 1.

29.2.17 Регистр MAC_FIFO_MODE

Относительный адрес: 0x30.

Доступ: R/W.

Значение после сброса: 0x0.

Таблица 318 – Регистр MAC_FIFO_MODE

Биты	Назначение
31:8	Поле зарезервировано.
7	Разрешение генерации запроса к DMA от FIFO передатчика (DMA_REQ_EN): 0 – нет разрешения; 1 – разрешено
6	Режим генерации на обслуживание FIFO статуса приемника (R_REQ_M): 0 – после передачи всех данных; 1 – во время передачи последних восьми квадрослов
5	Управление дескрипторами приемника (Smart_Rx): 1 – прямое; 0 – косвенное
4	Режим работы FIFO данных приемника (HD_mode): 0 – дуплексный; 1 – полудуплексный
3	Режим работы буфера приемника(RX_FFM): 0 – память; 1 – FIFO
2	Выбор типа прерывания: 0 – по уровню; 1 – импульсное
1	Разрешение прерывания (когда 1) при готовности FIFO приемника
0	Разрешение прерывания (когда 1) при готовности FIFO передатчика

Регистр определяет дополнительные режимы взаимодействия с модулем Ethernet MAC. Буферная память и дескрипторы приемника и передатчика могут быть доступны в режиме FIFO. В этом случае логика FIFO имеет возможность формировать запросы прерываний к процессору и запросы на обслуживание к системному DMA в дополнение (или вместо) запросов прерывания, формируемых с помощью регистра INTERRUPT_MASK. Если для обслуживания передатчика используется канал DMA, то после выполнения всех подготовительных работ (включение интерфейса, программирования канала DMA) необходимо установить бит DMA_REQ_EN в 1. Это разрешит генерацию запросов к DMA от FIFO передатчика.

Буфер данных приемника имеет дополнительный режим работы. Если бит RX_FFM равен нулю, то доступ к текущему буферу выполняется на основании текущего указателя FIFO (буфер 0 или буфер 1). Вычитка данных должна производится только после полного приема фрейма, т.е. только после того как в FIFO статуса будет записано управляющее слово. Если же бит RX_FFM равен 1, то вычитка принимаемых в буфер данных может производиться одновременно с приемом. Ввиду наличия некоторых ограничений на прием данных, запрос к каналу DMA формируется только если принято не менее 128 байт данных (HD_mode равен 0) либо не менее 512 байт данных

(HD_mode равен 1). Если бит RX_FFM установлен в 1, FIFO статуса формирует запрос к каналу DMA только если все принятые данные фрейма были перемещены в системную память. При этом бит R_REQ_M позволяет формировать запрос к DMA немного раньше. Если данный бит установлен в 1, то запрос на чтение FIFO статуса будет сформирован уже во время начала чтения последних восьми квадрослов принятых данных. Это позволяет DMA более оперативно произвести вычитку статусной информации. Если бит R_REQ_M равен 1, то необходимо следить, чтобы запрос к статусному FIFO использовал тот же канал EDMA контроллера, что и запрос к данным. В противном случае теоретически возможна ситуация, когда чтение статуса может опередить вычитку последнего квадрослова данных.

При обслуживании FIFO статуса можно использовать функцию автоматической активации дескриптора обслуживаемого буфера приемника. Если бит RX_FFM равен 1 и бит Smart_Rx равен 0, то при выполнении записи статуса принятого фрейма в FIFO статуса возможна автоматическая активация следующего дескриптора. Для этого необходимо выполнение следующих условий:

– в буферной памяти приемника достаточно места для приема еще одного фрейма максимальной длины.

– в FIFO статуса имеется место для приема статуса следующего фрейма.

Если же в FIFO данных либо в FIFO статуса уже нет места для приема следующего фрейма, то после приема некоторого количества фреймов может создаться ситуация, когда оба дескриптора приемника имеют неактивное значение и прием остановлен. В этом случае (бит RX_FFM равен 1 и бит Smart_Rx равен 0) если в процессе обслуживания приемника появляется возможность принять следующий фрейм, то текущий по очереди дескриптор автоматически устанавливается в активное значение. Имеется специальный регистр Rx_STATUS_POP_ACT чтения статусного FIFO, который автоматически устанавливает бит активности обслуженного дескриптора. Однако в режиме RX_FFM==1 и Smart_Rx==0, чтение данного регистра вызовет автоматическую активацию дескриптора только если вычитываемый статус был последним в FIFO. Если бит Smart_Rx равен 1, то автоматическая активация дескриптора возможна только при чтении регистра Rx_STATUS_POP_ACT, т.е. пока фрейм не будет полностью передан в системную память, дескриптор активировать нельзя. В этом случае невозможно в буферной памяти и FIFO статуса одновременного нахождения более двух необслуженных фреймов.

Основное назначение регистра MAC_FIFO_MODE состоит в создании для DMA контроллера максимальных удобств при обслуживании интерфейса. Это в свою очередь снижает нагрузку на процессор.

29.2.18 Регистр MAC_FIFO_STATUS

Относительный адрес: 0x34.

Доступ: R/W.

Значение после сброса: 0x0.

Таблица 319 – Регистр MAC_FIFO_STATUS

Биты	Назначение
31:10	Всегда 0
9	RF_REQ – запрос (когда 1) на обслуживание FIFO данных приемника
8	R_REQ – запрос (когда 1) на обслуживание FIFO статуса приемника
7	HW_ERR: 1 – аппаратная ошибка приемника; 0 – нет ошибок
6	Rx_status_full. FIFO статуса приемника: 0 – заполнено полностью; 1 - имеет свободное место
5	RX_fifo_RP: номер (0 или 1) текущего буфера принятого фрейма для обработки внешним мастером
4	TX_fifo_XP: номер (0 или 1) текущего буфера (дескриптора) взятого передатчиком для передачи.
3	TX_fifo_WP: значение (0 или 1) указателя записи FIFO передатчика, т.е. номер буфера (дескриптора) в который будет произведена следующая запись.
2	TX_EMPTY: 0 – в передающем FIFO имеются данные; 1 – передающее FIFO пустое
1	RX_nEMPTY: 1 – в FIFO приемника имеется фрейм данных для обработки; 0 – нет запросов для обслуживания
0	TX_nFULL: 1 – в FIFO передатчика имеется свободное место для передачи следующего фрейма; 0 – FIFO передатчика занято либо передатчик выключен.

Бит TX_fifo_WP используется при работе с регистром DESC_TX_FIFO. В этом случае он указывает на номер дескриптора (0 или 1) передатчика к которому выполняется доступ в режиме FIFO. Бит TX_fifo_RP используется при работе с регистром DESC_RX_FIFO. В этом случае он указывает на номер дескриптора (0 или 1) приемника к которому выполняется доступ в режиме FIFO. Бит TX_fifo_XP указывает на номер буфера (дескриптора) взятого передатчиком для обслуживания либо уже обслуженного.

29.2.19 Регистр MAC_FIFO_IRQ

Относительный адрес: 0x35.

Доступ: R/W.

Значение после сброса: 0x0.

Таблица 320 – Регистр MAC_FIFO_IRQ

Биты	Назначение
31:4	Всегда 0
3	RX_IRQ: 1 – запрос прерывания от приемника INTERRUPT_MASK
2	TX_IRQ: 1 – запрос прерывания от передатчика INTERRUPT_MASK
1	RX_FIFO_IRQ: 1 – запрос прерывания при готовности данных в FIFO приемника
0	TX_FIFO_IRQ: 1 – запрос прерывания при готовности FIFO передатчика принять следующий фрейм данных

Имеется принципиальное отличие в сигналах запросов прерываний, генерируемыми с помощью регистра INTERRUPT_MASK и регистра MAC_FIFO_IRQ. В первом случае мы получаем прерывание, информирующее нас о том, что фрейм передан либо принят. Во втором случае мы получаем прерывание о том, что фрейм можно переместить в буфер передатчика для отправки либо, что принятый в буфер фрейм можно переместить в системную память. Например, получив прерывание по TX_FIFO_IRQ, мы размещаем фрейм в буфере передатчика и записываем активное значение дескриптора. После этого фрейм будет передан и после завершения передачи мы получим прерывание TX_IRQ. В случае приема ситуация немного сложнее. В начале мы получим прерывание RX_FIFO_IRQ сообщающее нам информацию о том, что данные и статус принятого фрейма находятся в буфере приемника. Только после вычитки данных и статуса мы получим прерывание RX_IRQ. Прерывания от логики FIFO могут быть импульсные либо по уровню. Прерывание от INTERRUPT_MASK только по уровню. Пользователь сам может себе выбрать наиболее удобный источник прерываний. Для обслуживания передатчика (если используется только процессор) достаточно прерывания TX_FIFO_IRQ. Каждый раз, когда мы получаем это прерывание мы имеем информацию о том, что очередной фрейм может быть размещен в буфере передатчика. Как только мы записываем активный дескриптор для подтверждения отправки фрейма, мы может получить следующее TX_FIFO_IRQ прерывание, если в буфере имеется место для следующего фрейма. В противном случае мы получим новое TX_FIFO_IRQ прерывание только когда очередной фрейм будет передан и в буфере появится свободное место. Для обслуживания приемника достаточно прерывания RX_FIFO_IRQ. После вычитки статуса новое RX_FIFO_IRQ прерывание будет сформировано, если в FIFO статуса имеется новый принятый фрейм.

29.2.20 Регистр Rx_STATUS

Относительный адрес: 0x36.
Доступ: R/W.
Значение после сброса: 0x0.

Таблица 321 – Регистр Rx_STATUS

Биты	Назначение
31:20	Всегда 0
19:17	Статус принятого фрейма RX_DN_ST
16	Номер активного буфера (0 или 1) RXP
15:14	Всегда 0
13:0	Размер данных в буфере принятого фрейма (количество байт) RX_SIZE

Регистр статусного FIFO имеет несколько адресов доступа. Чтение по различным адресам вызывает различные побочные действия. При чтении Rx_STATUS считывается текущее значение статуса без каких-либо дополнительных действий. Если мы прочитаем регистр Rx_STATUS_POP, то дополнительно к прочитанной информации указатель на чтение будет увеличен на 1. Таким образом, мы переходим к анализу следующего фрейма. При чтении регистра Rx_STATUS_POP_ACT, выполняются аналогичные действия, но дополнительно активизируется бит ACT соответствующего (RXP) дескриптора приемника. Таким образом, посредством чтения, мы можем автоматически активировать обслуженный буфер для приема нового фрейма.

Биты RX_SIZE указывают (в байтах) на количество принятых в буфер данных. Бит RXP указывает на один из двух буферов в котором находятся принятые данные. При доступе к буферу по адресам RX_FIFO_DATA, бит RXP определяет к какому конкретно буферу выполняется обращение. Биты RX_DN_ST содержат код ошибки принятого сообщения. Данная информация идентична информации, которая записывается приемником в регистр дескриптора после завершения приема фрейма.

29.2.21 Регистр TX_RX_CNT

Относительный адрес: 0x37.
Доступ: R/O.
Значение после сброса: 0x0.

Таблица 322 – Регистр TX_RX_CNT

Биты	Назначение
31:24	Всегда 0
23:16	Количество фреймов принятых приемником и обработанных внешним мастером RX_CNT
15:8	Всегда 0
7:0	Количество фреймов, переданных передатчику TX_CNT

Данный регистр позволяет отследить количество переданных в FIFO фреймов (но не обязательно уже отправленных), а также количество принятых фреймов приемником и обслуженных внешним мастером. Иными словами, счетчик TX_CNT ведет подсчет количества операций push (с битом ACT равным 1) в передающее FIFO, а счетчик RX_CNT подсчет количества операций pop из приемного FIFO статуса. После значения 255 счетчик принимает значение 0. Сброс значений счетчиков выполняется при записи специального значения в регистр программного сброса (бит 1 или бит 0 равны 1).

29.2.22 Регистр RX_DFIFO_info

Относительный адрес: 0x38.
Доступ: R/O.
Значение после сброса: 0x0.

Таблица 323 – Регистр RX_DFIFO_info

Биты	Назначение
31	RX_D_FULL: 1 – буфер заполнен полностью; 0 – в буфере имеется свободное место
30:28	Всегда 0
27:18	RX_D_RP – текущее значение указателя чтения FIFO данных приемника (указатель на квадрослово)
17:12	Всегда 0
11:0	RX_D_WP – текущее значение указателя записи в FIFO данных приемника (с точностью до слова)

Данный регистр имеет справочный характер и предоставляет информацию о работе буфера данных приемника в случае, когда бит RX_FFM равен 1. В этом режиме работы приемник использует для записи в память принятых данных не абсолютный адрес, а указатель RX_D_WP. При чтении данных из буфера используется не абсолютный адрес, а значение указателя RX_D_RP. Вместе с этим, при вычитке порции из восьми квадрослов данных, внешний мастер должен использовать изменяющийся адрес. Биты 4:2 указателя на слово в буфере должны изменять свое значение от 0 до 7. Данный факт позволяет отследить чтение последнего слова порции данных.

29.2.23 Регистр QW_CNT

Относительный адрес: 0x39.
Доступ: R/O.
Значение после сброса: 0x0.

Таблица 324 – Регистр QW_CNT

Биты	Назначение
31:0	Число квадрослов данных, отправленных из FIFO приемника во внешнюю память. Это число со знаком. Отрицательное значение указывает на то, что принятый фрейм еще не отправлен в основную память. Если ноль или положительное число, значит текущий фрейм полностью передан в основную память.

Каждый раз, когда из буфера данных приемника (работающего в режиме FIFO) считывается квадрослово данных, QW_CNT увеличивает свое значение на 1. Если мы считываем информацию из FIFO статуса приемника, то значение QW_CNT уменьшается на величину QW_LEN, которая рассчитывается как длина принятого фрейма в байтах, выравненная к числу квадрослов кратных 8 в большую сторону. Например, принято сообщение длиной в 150 байт (128+22). Значение QW_LEN будет равно 16. При корректной работе приемника, после вычитки данный и последующей вычитке статуса, в регистре QW_CNT должен быть ноль.

29.3 Дескрипторы TX

Дескрипторы TX содержат информацию о соответствующих буферах TX. Интерфейс содержит два дескриптора TX.

Таблица 325 – Дескриптор TX (контроль и статус)

Номер	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	-		LENGTH													

Номер	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACT	-			FPR	-		DEF	CER	RETRY_ATTEMPT					TX_DN_ST	

Таблица 326 – Описание полей дескриптора TX (контроль и статус)

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31:30	-	Зарезервировано
29:16	LENGTH	Размер данных (в байтах), который был записан для передачи. Должен быть ≥ 1. При записи 0 в это поле, блок Ethernet MAC перезапишет 1 в это поле для избежания блокировки. В случае, если поддержка jumbo-фреймов отключена (не объявлено «`define JUMBO_FRAMES_SUPPORT» в файле config.v), разряды 29:27 поля LENGTH являются зарезервированными

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
15	ACT	Активный статус. При записи 1 – дескриптор TX устанавливается в активный режим, после чего данный дескриптор будет либо ожидать очереди на отправку, либо уже сейчас находится в процессе отправки. Если ACT равен 1, то запись в это поле не будет иметь эффекта. Запись в это поле возможна только если значение этого поля равно 0. При чтении 0 – дескриптор TX не активен; если дескриптор ранее был установлен как активный. ACT равный 0 означает, что передача по TX была завершена, а поля дескриптора теперь содержат информацию о статусе передачи
14:12	-	Зарезервировано
11	FPR	1 – FCS и Padding включены в данные передачи; 0 – FCS и Padding должны быть добавлены блоком Ethernet MAC автоматически
10:9	-	Зарезервировано
8	DEF	Deferred. Если этот бит равен 1, тогда фрейм был задержан (deferred) перед передачей. Это поле используется для счётчика aFramesWithDeferredXmissions (Layer Management)
7	CER	Collision error. Если этот бит равен 1, значит при передаче фрейма возникла ошибка типа aCarrierSenseErrors
6:2	RETRY_ ATTEMPT	Это поле показывает число попыток повтора (retry attempts), после которых передача была завершена со статусом TX_DN_ST. Полное число повторов рассчитывается как (RETRY_ATTEMPT + 1) если (TX_DN_ST == 01 – TRANSMIT_OK); если (TX_DN_ST == 11 – LATE_COLLISION_ERROR) или (TX_DN_ST == 10 – EXCESSIVE_COLLISION_ERROR), то число повторов рассчитывается как (RETRY_ATTEMPT). При (RETRY_ATTEMPT == attemptLimit == 16) в поле TX_DN_ST генерируется статус: 10 – EXCESSIVE_COLLISION_ERROR
1:0	TX_DN_ST	TX Done Status – статус окончания передачи по TX: 00 – зарезервировано; 01 – TRANSMIT_OK; 10 – EXCESSIVE_COLLISION_ERROR; 11 – LATE_COLLISION_ERROR

В блоке Ethernet MAC процесс отправки фреймов связан с буферами и дескрипторами TX, причём к каждому буферу TX приставлен свой дескриптор шириной

в 32 разряда. Размер области памяти, выделенной для одного буфера фрейма, составляет 8192 байта. Число внутренних буферов TX равно двум.

Начальный относительный адрес дескрипторов TX в модуле регистров интерфейса равен 0x100. Общая формула расчёта адреса дескриптора TX:

Относительный адрес дескриптора TX (контроль и статус) = 0x100 + (номер дескриптора TX • 1).

Доступ к регистрам дескрипторов возможен только в режиме 32-разрядной записи или чтения. Возможно прямое обращение к регистрам дескриптора (по их адресу), а также косвенное обращение посредством регистра DESC_TX_FIFO. Данный регистр может быть использован при доступе к модулю передачи в режиме FIFO. Запись или чтение по адресу DESC_TX_FIFO будет вызывать обращение к 0-му или к 1-му дескриптору согласно текущего FIFO указателя (бит TX_fifo_WP регистра MAC_FIFO_STATUS). В этом случае нет необходимости отслеживать с каким конкретно дескриптором работает модуль передачи. При сбросе модуля начальное значение указателей равно нулю.

Для того, чтобы сконфигурировать блок Ethernet MAC на отправку фрейма по TX, необходимо сначала записать фрейм в один из буферов TX, а затем соответствующим образом активировать регистр закреплённого за этим буфером дескриптора TX. Для активации дескриптора TX со стороны центрального процессора в дескрипторе TX бит ACT записывается в значение 1. По окончании отправки фрейма по TX блок Ethernet MAC самостоятельно записывает статус окончания операции в исходный дескриптор и сбрасывает в этом дескрипторе разряд ACT в 0.

29.4 Дескрипторы RX

Дескрипторы RX содержат информацию о соответствующих буферах RX. Интерфейс содержит два дескриптора RX.

Таблица 327 – Дескриптор RX (контроль и статус)

Номер	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
	-		LENGTH													

Номер	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	ACT	-												RX_DN_ST		

Таблица 328 – Описание полей дескриптора RX (контроль и статус)

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
31:30		Зарезервировано

Номер бита	Функциональное имя бита	Расшифровка функционального имени бита, краткое описание назначения и принимаемых значений
29:16	LENGTH	Длина (в байтах) принятого фрейма, который был помещён в соответствующий буфер RX. В случае, если поддержка jumbo-фреймов отключена (не объявлено «`define JUMBO_FRAMES_SUPPORT» в файле config.v), разряды 29:27 поля LENGTH являются зарезервированными
15	ACT	Активный статус. При записи 1 – дескриптор RX устанавливается в активный режим, после чего данный дескриптор будет ожидать приёма фрейма по RX. Если ACT равен 1, то запись в это поле не будет иметь эффекта. Запись в это поле возможна только если значение этого поля равно 0. При чтении 0 – дескриптор RX не активен; если дескриптор ранее был установлен как активный. ACT равный 0 означает, что приём по RX был завершён, а поля дескриптора теперь содержат информацию о статусе принятого фрейма
14:3		Зарезервировано
2:0	RX_DN_ST	RX Done Status – статус принятого по RX фрейма: 000 – зарезервировано; 001 – RECEIVE_OK; 010 – LENGTH_ERROR; 011 – FRAME_CHECK_ERROR; 100 – FRAME_TOO_LONG; 101 – ALIGNMENT_ERROR; 110 – зарезервировано; 111 – зарезервировано

В блоке Ethernet MAC процесс приёма фреймов связан с буферами и дескрипторами RX, причём к каждому буферу RX приставлен свой дескриптор шириной в 32 разряда. Размер области памяти, выделенной для одного буфера фрейма, составляет 8192 байта. Число внутренних буферов RX равно двум.

Начальный относительный адрес дескрипторов RX в модуле регистров интерфейса равен 0x200. Общая формула расчёта адреса дескриптора RX:

Относительный адрес дескриптора RX (контроль и статус) = 0x200 + (номер дескриптора RX • 1).

Доступ к регистрам дескрипторов возможен только в режиме 32-разрядной записи или чтения. Возможно прямое обращение к регистрам дескриптора (по их адресу), а также косвенное обращение посредством регистра DESC_RX_FIFO. Данный регистр может быть использован при доступе к модулю в режиме FIFO. Запись или чтение по адресу DESC_RX_FIFO будет вызывать обращение к 0-му или к 1-му дескриптору согласно текущего FIFO указателя (бит RX_fifo_RP регистра MAC_FIFO_STATUS). В этом случае нет необходимости отслеживать с каким конкретно дескриптором работает модуль приема. При сбросе модуля начальное значение указателей равно нулю.

Для того, чтобы сконфигурировать блок Ethernet MAC на приём фрейма по RX, необходимо сначала высвободить один из буферов RX, после чего соответствующим образом активировать регистр дескриптора RX, соответствующий этому буферу RX. Для активации дескриптора RX со стороны центрального процессора в дескрипторе RX бит ACT записывается в значение 1. По окончании приёма фрейма по RX блок Ethernet MAC выполняет следующие действия:

– Записывает статус окончания операции в FIFO статуса;

– Записывает аналогичный статус окончания операции в исходный дескриптор и сбрасывает в этом дескрипторе разряд ACT в 0. (возможны варианты).

29.5 Использование MII Management Interface

Интерфейс MII Management Interface (MII MI) – это простой двухпроводной серийный интерфейс между одним главным устройством типа Ethernet MAC и, возможно, многими подчинёнными устройствами типа Ethernet PHY. Интерфейс MII Management Interface используется для доступа к регистрам управления и статуса внутри PHY. Физический интерфейс MII MI состоит из двунаправленной линии MDIO (Management Data Input/Output) и однонаправленной линии синхросигнала MDC (Management Data Clock). Передачи по линии MDIO синхронны с MDC. При операции записи Ethernet MAC отправляет 32-разрядную преамбулу (опционально) и 32-разрядный фрейм записи по линии MDIO, последние 16 разрядов которого – записываемое поле с данными DATA для адресуемого регистра. При операции чтения Ethernet MAC отправляет 32-разрядную преамбулу (опционально) и 16-разрядный полуфрейм чтения по линии MDIO, а затем Ethernet MAC освобождает линию MDIO, а подчинённый PHY перехватывает линию MDIO и отправляет по ней 16 разрядов поля DATA с содержимым адресуемого регистра.

Для отделения цифровой и аналоговой частей интерфейса MII MI, внешний интерфейс Ethernet MAC MII MI состоит из четырёх однонаправленных сигналов:

– MDC;
– MDI (Management Data Input);
– MDO (Management Data Output);
– MDOEN (Management Data Output Enable).

Однонаправленные линии MDI, MDO и MDOEN подключаются к буферу с тремя состояниями для линии MDIO соответствующим образом, показанным на рисунке 157.

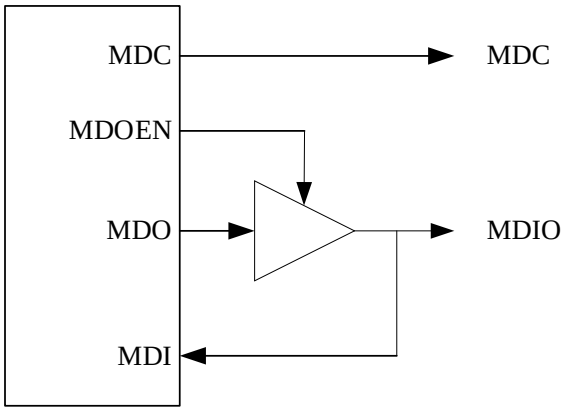


Рисунок 157 – Подключение линий MDI, MDO и MDOEN к буферу с тремя состояниями

Интерфейс MII MI в блоке Ethernet MAC поддерживает две команды:

– Команда отправки фрейма записи;

– Команда отправки фрейма чтения.

Для отправки фрейма чтения или фрейма записи по линии MDIO центральным процессором должен быть соответствующим образом записан регистр MII_MI_COMMAND и все связанные с ним регистры (см. раздел 1.4.9 Регистр MII_MI_COMMAND).

Интерфейс MII MI описан в разделе спецификации IEEE 802.3 SECTION TWO «22. Reconciliation Sublayer (RS) and Media Independent Interface (MII)», в подразделе «22.2.4 Management functions».

29.6 Режимы работы интерфейса

Возможны различные режимы работы интерфейса:

– Обслуживание с помощью процессора;

– Обслуживание с помощью процессора и системного DMA.

Возможно обслуживание интерфейса с использованием DMA, с привлечением процессора только в случае обнаружения ошибок или окончания приема либо передачи.

29.6.1 Передача

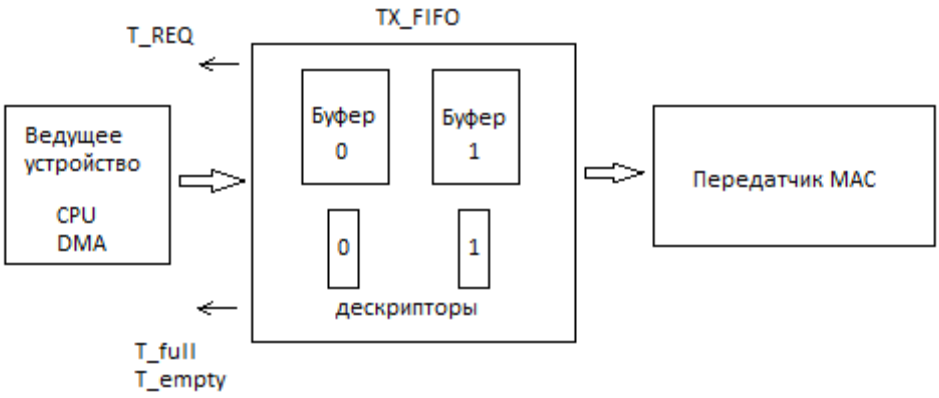


Рисунок 158 – Структурная схема передатчика

Для того, чтобы начать передачу фреймов по Ethernet MAC TX, необходимо завершить несколько предварительных этапов настройки.

Для разрешения прерываний необходимо записать регистр INTERRUPT_MASK и (или) регистр MAC_FIFO_MODE.

Предварительная запись регистра BACKOFF_RANDOM_SEED необходима для TX для функции определения времени повторной попытки передачи (BackOff). Для корректной работы функции BackOff необходимо, чтобы у двух сообщающихся блоков Ethernet MAC не совпадало содержимое регистров BACKOFF_RANDOM_SEED (рекомендуется генерировать содержимое регистра BACKOFF_RANDOM_SEED с использованием числа уникального MAC-адреса).

Далее необходимо записать все поля регистра MODE в нужные значения, в том числе поле «TX: Transmit Enable» в значение 1.

После этих предварительных шагов Ethernet MAC TX готов к конфигурированию фреймов на отправку. Для отправки фреймов необходимо:

– Записать данные фрейма в буфер TX.
– Записать соответствующий буферу дескриптор TX.

Число дескрипторов TX равно двум. После сброса Ethernet MAC TX ожидает готовности дескриптора TX номер 0. Для начала передачи необходимо поместить фрейм в буфер TX номер 0, а затем записать дескриптор TX номер 0, причём поле ACT должно быть установлено в 1, а указатель дескриптора TX должен содержать достоверную информацию. После этого Ethernet MAC TX начнёт отправлять буфер 0. Если есть очередной фрейм на отправку, то процессор может записать буфер TX номер 1 и дескриптор TX номер 1 аналогичным образом. После того, как буфер 0 будет отправлен, Ethernet MAC TX начнёт ожидать готовности буфера 1. Если буфер 1 уже установлен на отправку, Ethernet MAC TX сразу же начнёт его отправку. После отправки буфера 1 Ethernet MAC TX начнёт ожидать буфер 0 на готовность (обращение номеров по кругу). Таким образом осуществляется круговое обращение буферов TX в Ethernet MAC TX.

В интерфейсе реализован доступ к буферам и дескрипторам посредством метода FIFO. В этом случае пользователю нет необходимости знать с каким конкретно номером буфера (0 или 1) ему необходимо работать. Для доступа к указанным ресурсам используются специальные адреса (FIFO-области), дополнительно использующие внутренние FIFO указатели, выбирающий текущий номер буфера.

Активация дескриптора TX означает его блокировку: после того, как поле ACT записано в 1, возможность записи в этот дескриптор блокируется до тех пор, пока соответствующий буфер не будет отправлен. Возможна ситуация, при которой все буферы TX установлены на отправку, и не осталось ни одного свободного буфера TX (это соответствует случаю, когда FIFO передатчика заполнено полностью). В таком случае необходимо ожидать высвобождения ближайшего буфера TX для отправки нового фрейма. Нужно отметить, что буферы TX загружаются процессором (DMA) и высвобождаются блоком Ethernet MAC TX в порядке возрастания номеров буферов, с обращением к номеру 0 после последнего номера буфера TX.

На рисунке 159 показан типовой алгоритм регистрации окончания отправок фреймов по Ethernet MAC TX. Данный алгоритм предложен как один из возможных вариантов использования Ethernet MAC в процессе регистрации окончания отправок фреймов по TX.

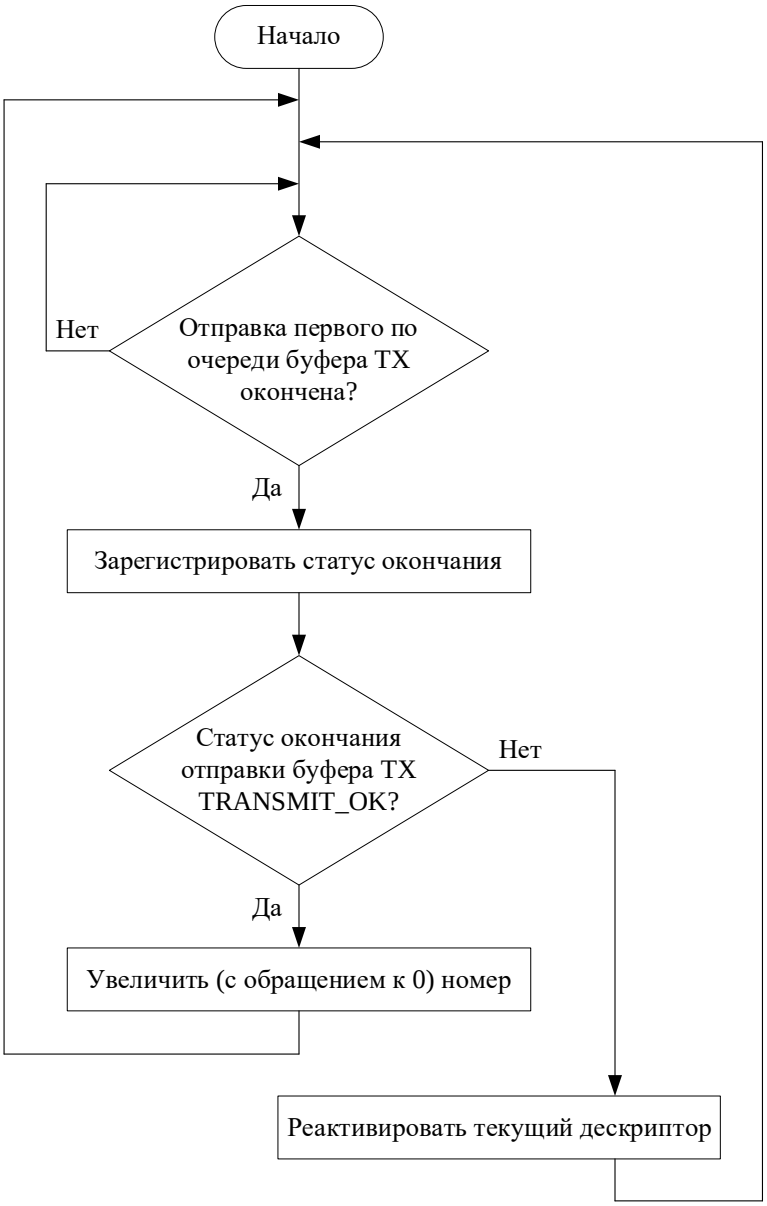


Рисунок 159 – Типовой алгоритм регистрации окончания отправок фреймов по Ethernet MAC TX

Когда Ethernet MAC TX заканчивает отправку фрейма, поле ACT соответствующего дескриптора TX очищается в значение 0, в данный дескриптор TX помещается информация о статусе окончания передачи, указатель чтения FIFO передатчика увеличивается на 1 (переход к следующему фрейму), возможна (если разрешена) установка сигнала прерывания в значение 1.

Чтобы определить, что отправка фрейма по TX была окончена, процессор может либо регулярно опрашивать бит TX_EMPTY регистра MAC_FIFO_STATUS, либо может ожидать сигнал прерывания от Ethernet MAC TX, который будет установлен в 1 после деактивации очередного дескриптора TX. В последнем случае информация о последнем дескрипторе TX, который вызвал прерывание, будет содержаться в регистре INTERRUPT_SOURCE.

После того, как дескриптор TX деактивируется, Ethernet MAC TX переводит внутренний указатель на следующий по очереди буфер TX: если следующий буфер TX

установлен как активный, то сразу же начнётся его отправка, иначе Ethernet MAC TX будет ожидать активации этого дескриптора со стороны процессора.

Если статус окончания передачи TRANSMIT_OK, то Ethernet MAC TX переводит внутренний указатель на следующий по очереди буфер TX: если следующий буфер TX установлен как активный, то сразу же начнётся его отправка, иначе Ethernet MAC TX будет ожидать активации этого дескриптора со стороны процессора. Таким образом, в случае статуса TRANSMIT_OK у деактивированного дескриптора TX необходимо увеличить на 1 номер следующего ожидаемого к деактивации дескриптора TX, после чего нужно начать ожидание деактивации этого дескриптора.

В случае, если статус окончания передачи не TRANSMIT_OK (есть код ошибки), то Ethernet MAC TX не переводит внутренний указатель на следующий по очереди буфер TX, и отправка фреймов полностью останавливается до вмешательства со стороны управляющего процессора. В данной ситуации для продолжения отправки фреймов нужно либо реактивировать дескриптор TX, который был отправлен с ошибкой, либо сделать полный сброс Ethernet MAC. Такое поведение при возникновении кода ошибки передачи обусловлено тем, что в такой ситуации существует несколько возможных вариантов поведения для управляющего процессора (драйвера), и выбор такого поведения ложится на драйвер.

Возможен режим прямого доступа к регистрам дескрипторов и буферам передатчика, а также режим FIFO, когда нет необходимости отслеживать текущий номер буфера(дескриптора) доступного в данный момент для работы. В режиме FIFO флаг TX_nFULL регистра состояния MAC_FIFO_STATUS информирует о том, есть в FIFO передатчика свободное место или нет. Если свободное место имеется (флаг равен 1) то необходимо произвести запись данных фрейма в область адресов TX_FIFO_DATA, а затем записать активное значение дескриптора фрейма в регистр DESC_TX_FIFO. Внутренняя логика самостоятельно отслеживает, с каким из двух дескрипторов необходимо выполнить действия. После записи дескриптора указатель записи FIFO увеличивается на 1. Указатель чтения FIFO увеличивается на 1 после того как фрейм будет передан.

Таким образом, процесс обслуживания передатчика заключается в выполнении следующих действий:

– загрузка данных фрейма в буфер TX_FIFO_DATA передатчика;

– запись активного значения дескриптора в регистр DESC_TX_FIFO для разрешения передачи фрейма;

– чтение статуса переданного фрейма.

Обслуживание передатчика может выполняться как с помощью процессора, так и с помощью соответствующего канала DMA. В случае обслуживания с помощью DMA, передатчик генерирует специальный запрос T_REQ, в соответствии с которым DMA выполняет заданную последовательность операций. Поскольку для обслуживания необходимо выполнение трех разнородных действий, то канал DMA активизирует цепочку из трех операций:

– чтение регистра DESC_TX_FIFO в массив статуса передачи в системной памяти;

– чтение массива данных из системной памяти в буфер TX_FIFO_DATA передатчика;

– чтение из массива дескрипторов в системной памяти очередного дескриптора и запись по адресу DESC_TX_FIFO.

Первоначальная вычитка статуса переданного фрейма необходима, если до текущего момента два или более фрейма уже были отправлены. DMA может генерировать прерывание по окончании передачи каждого или некоторого количества фреймов. При организации передачи с использованием DMA, в системной памяти должен быть определен список заданий для передачи необходимого количества фреймов.

При обслуживании посредством процессора, могут использоваться специальные флаги регистра состояния, отражающие текущее состояние буфера передатчика. Также может быть использован запрос прерывания, который формируется каждый раз, когда в буфере передатчика имеется свободное место. При обслуживании передатчика с помощью процессора (без привлечения DMA) пользователь сам определяет удобный для себя алгоритм работы (прямой доступ к регистрам дескрипторов и буферной памяти либо косвенный доступ посредством режима FIFO).

Вычитываемый из регистра DESC_TX_FIFO статус содержит информацию об особенностях передачи фрейма. Необходимость в сохранении статуса переданного фрейма заключается в том, что во время передачи возможно наличие ошибки. В этом случае фрейм не будет передан, в регистр дескриптора будет записан код ошибки, а передатчик перейдет к обслуживанию следующего фрейма. При обнаружении ошибок передачи возможны следующие варианты поведения:

– разрешение генерации специального прерывания,

– завершение передачи всех фреймов и последующий анализ информации.

В последнем случае, после анализа статусной информации возможно повторение передачи некоторых фреймов либо какие-то дополнительные действия. Использование специального аппаратного прерывания позволяет более оперативно отреагировать на ошибку передачи.

Системный DMA может выполнить три описанные ранее действия с помощью цепочки операций. Действия должны выполняться строго в приведенном порядке. Первое действие всегда ожидает флага готовности FIFO передатчика. Последнее действие (запись активного дескриптора) приводит к увеличению указателя FIFO и означает готовность фрейма к отправке. Уменьшение указателя FIFO выполняется сигналом окончания передачи фрейма. Указатель FIFO передатчика увеличивает свое значение при любой записи в регистр дескриптора. Однако вместе с этим он запоминает бит ACT записываемого дескриптора. Если бит равен 1, то указатель чтения будет изменяться сигналом завершения передачи фрейма. Если бит равен 0, то указатель чтения изменится автоматически. Данная функция реализована с целью упрощения вычитки статуса дескриптора после завершения передачи (вычитка статуса последних двух переданных фреймов).

29.6.1.1 *Подготовка данных для передачи*

Перед отправкой фрейма (кадра) необходимые данные должны быть размещены в буфере передачи. Общая структура одного фрейма MAC приведена на рисунке 160.

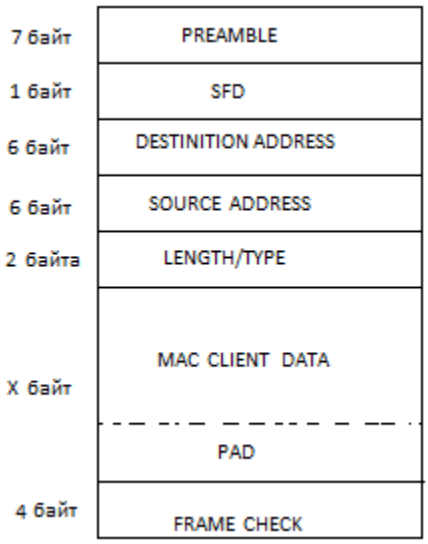


Рисунок 160 – Общая структура одного фрейма MAC

При передаче каждого фрейма, блок передачи автоматически вставляет в кадр восемь начальных байт (Preamble + SFD). Каждый байт поля Preamble имеет значение 0x55, а байт поля SFD имеет значение 0xD5. Таким образом, в буфере передатчика, обязательно начиная с нулевого адреса, необходимо расположить данные, которые должны начинаться с поля Destinition address, т.е. с адреса приемника. Размер полезных данных (MAC Client data) может быть любой, но не должен превышать величину 8192 - 14 (либо -16). Четыре байта кода проверки (Frame check) могут отсутствовать. В этом случае должен быть задан режим автоматического формирования контрольной суммы передатчиком.

Как отмечалось ранее, каждый буфер имеет ассоциированный с ним дескриптор. Поле LENGTH дескриптора должно быть равно общей длине данных, записанных в буфер, т.е. с учетом размера всех присутствующих полей (кроме Preamble и SFD). Передатчик передает число байт заданное в поле LENGTH дескриптора. Если объем передаваемых данных менее необходимого, то передатчик может автоматически дополнять кадр до необходимого минимального размера.

Как отмечалось ранее, при использовании DMA, необходимые для передачи данные должны быть подготовлены в системной памяти в виде фреймов, выравненных на границе квадрослова. При этом, каждый фрейм должен иметь соответствующую структуру. При наличии ограниченного размера свободной памяти, использование промежуточных буферов, для подготовки фреймов к отправке, может быть затруднительным. В этом случае передачу некоторого объёма данных можно организовать следующим образом:

– разбить необходимый для передачи массив данных на равные порции;

– в буферах передатчика заполнить значения полей адреса приемника, адреса источника и поля длины необходимой информацией;

– запрограммировать DMA на пересылку данных непосредственно из исходного массива в буфер передачи (начиная с адреса поля MAC Client data).

При таком подходе начальный адрес источника и (или) приемника может быть не выровнен на границе квадрослова. Однако это не помешает DMA эффективно выполнить передачу, т.к. он поддерживает доступ по не выровненным адресам (нужен доп тест проверки).

29.7 Прием

На Рисунке 7 показан типовой алгоритм конфигурирования Ethernet MAC RX и активации возможности приёма фреймов по Ethernet MAC RX. Данный алгоритм предложен как один из возможных вариантов использования Ethernet MAC для приёма фреймов по RX.

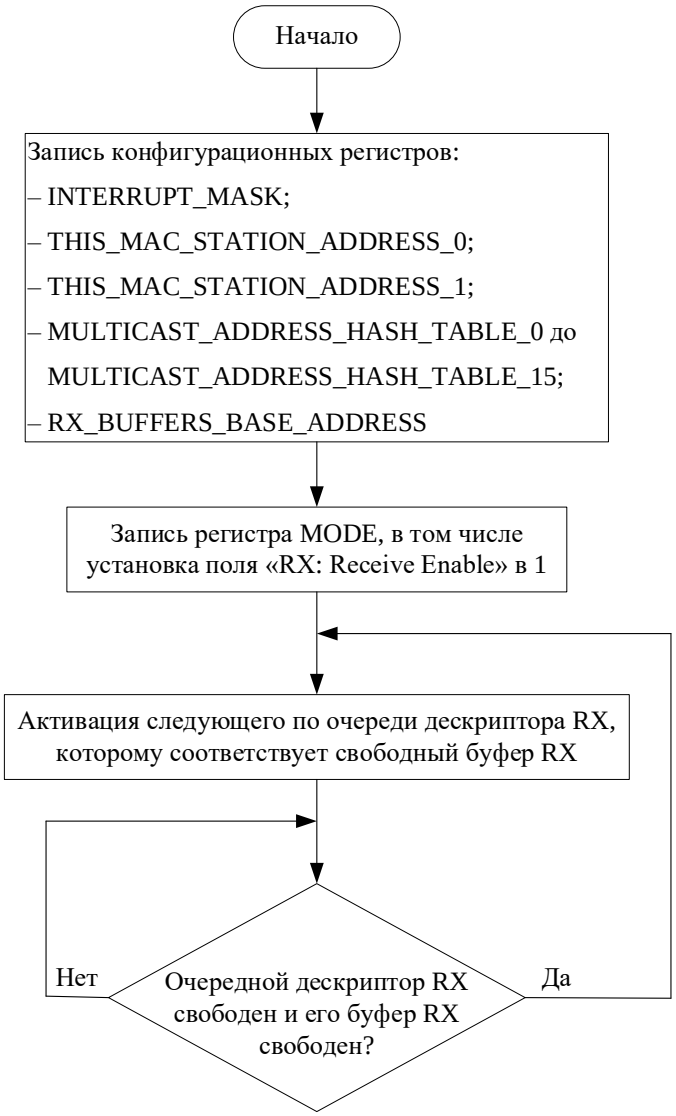


Рисунок 161 – Алгоритм конфигурирования Ethernet MAC RX и активации возможности приёма фреймов по Ethernet MAC RX

Для того, чтобы начать приём фреймов по Ethernet MAC RX, необходимо завершить несколько предварительных этапов настройки.

Для разрешения прерываний необходимо записать регистр INTERRUPT_MASK и (или) регистр MAC_FIFO_MODE.

Предварительная запись регистров THIS_MAC_STATION_ADDRESS_0 и THIS_MAC_STATION_ADDRESS_1 и регистров от MULTICAST_ADDRESS_ HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15 необходима RX для функции фильтрации адресов.

Далее необходимо записать все поля регистра MODE в нужные значения, в том числе поле «RX: Receive Enable» в значение 1.

После этих предварительных шагов Ethernet MAC RX готов к конфигурированию дескрипторов RX на приём. Для возможности приёма фреймов необходимо записать активное значение в дескрипторы RX (значение 0x8000). Биты ACT дескрипторов выполняют вспомогательную роль оперативного управления разрешением приема фреймов.

Число дескрипторов RX равно 2 (номера от 0 до 1). После сброса Ethernet MAC RX ожидает готовности дескриптора RX номер 0 на приём. Для возможности приёма фрейма в буфер RX номер 0 необходимо записать дескриптор RX номер 0, причём поле ACT должно быть установлено в 1. Затем дескриптор RX номер 1 может быть записан аналогичным образом, и буфер RX номер 1 также будет доступен для приёма фрейма. После того, как в буфер 0 будет принят фрейм, дескриптор 0 будет автоматически деактивирован (ACT равен 0) и записан со статусом приёма фрейма, а Ethernet MAC RX начнёт ожидать готовности буфера RX номер 1. Если буфер 1 уже установлен на приём, Ethernet MAC RX сразу же сможет осуществить приём в буфер 1. После обслуживания буфера 1 Ethernet MAC RX начнёт ожидать буфер 0 на готовность (обращение номеров по кругу). Таким образом осуществляется круговое обращение буферов RX в Ethernet MAC RX.

Нужно отметить, что активация дескриптора RX означает его блокировку: после того, как поле ACT записано в 1, возможность записи в этот дескриптор блокируется до тех пор, пока соответствующий буфер не будет принят и соответствующий дескриптор не будет автоматически деактивирован. Если все дескрипторы RX деактивированы, то возможность приёма очередных фреймов по RX отсутствует. В такой ситуации необходимо как можно скорее высвободить (вычитать) все принятые фреймы из буферов RX, и затем реактивировать дескрипторы RX. Нужно отметить, что дескрипторы RX активируются управляющим процессором и деактивируются блоком Ethernet MAC RX в порядке возрастания номеров буферов, с обращением к номеру 0 после последнего номера буфера RX.

Когда Ethernet MAC RX заканчивает приём фрейма, он записывает справочную (статусную) информацию в FIFO статуса. Данный статус содержит информацию о размере принятых данных, номере буфера в который были приняты данные, а также код завершения приема. Запись в FIFO статуса может формировать запрос на прерывание к процессору, либо запрос на обслуживание к каналу DMA. Любой из этих запросов означает необходимость вычитки данных из буфера данных и вычитки информационного слова из FIFO статуса. Как только вычитка данных будет завершена, поле ACT соответствующего дескриптора RX очищается в значение 0, а также в данный

дескриптор RX помещается информация о статусе окончания приёма (копия информации, записанной ранее в FIFO статуса). Данное событие может сопровождаться установкой сигнала прерывания в значение 1. Информация о последнем дескрипторе RX, который вызвал прерывание, будет содержаться в регистре INTERRUPT_SOURCE.

После того, как дескриптор RX деактивируется, Ethernet MAC RX переводит внутренний указатель на следующий по очереди буфер RX: если следующий буфер RX установлен как активный, то сразу же будет доступна возможность приёма фрейма, иначе Ethernet MAC RX будет ожидать активации этого дескриптора со стороны процессора. Таким образом, после приёма фрейма по RX необходимо высвободить соответствующий буфер RX и реактивировать соответствующий дескриптор RX. Реактивация обслуженного буфера(дескриптора) может быть выполнена автоматически при помощи чтения регистра RX_STATUS_POP_ACT. В данном случае чтение указанного регистра выполняет две функции: чтение справочной информации о принятом фрейме и активация дескриптора установкой бита ACT.

Таким образом, с точки зрения пользователя, процесс обслуживания приемника заключается в выполнении следующих действий:

– Чтение принятых данных из буфера приемника.

– Чтение статуса принятого фрейма.

– Запись активного значения дескриптора (0x8000) для разрешения приема следующего фрейма.

Возможно несколько вариантов обслуживания приемника. Самый простой заключается в обслуживания с помощью процессора. В этом случае структурная схема приемника может быть представлена, как на рисунке 162.

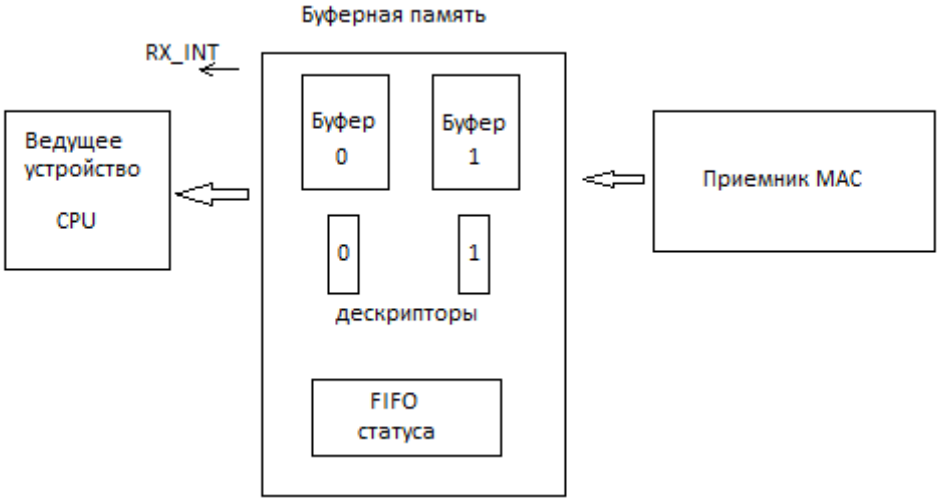


Рисунок 162 – Структурная схема приемника

После того как приемник включен и в регистры дескрипторов записано активное значение, возможен прием сообщений.

Во время приема данные записываются в один из двух буферов. Завершение приема сообщения сопровождается записью статусной информации в FIFO статуса. Статусное FIFO может генерировать прерывание к процессору. После получения прерывания процессор должен:

– вычитать соответствующую информацию из регистра Rx_STATUS и определить размер принятых данных (RX_SIZE);

– вычитать данные из соответствующего буфера RX_FIFO_DATA. Вычитка данных должна производиться посредством чтения квадрослов. При вычитке из буфера последнего квадрослова принятых данных в приемник MACa передается импульс, в соответствии с которым он записывает статусную информацию в регистр дескриптора DESC_RXi;

– записать новое активное значение дескриптора DESC_RX_FIFO;

– сбросить прочитанное ранее значение Rx_STATUS с вершины FIFO посредством чтения регистра Rx_STATUS_POP.

После этого, обслуженный буфер будет готов к приему нового сообщения. Отметим особенность процедуры удаления обслуженного сообщения из FIFO статуса. Здесь возможны следующие варианты:

– после вычитки данных, записываем активное значение в регистр дескриптора DESC_RX_FIFO и затем делаем чтение регистра Rx_STATUS_POP;

– после вычитки данных, делаем чтение регистра Rx_STATUS_POP_ACT. Это приводит не только к продвижению указателя чтения FIFO, но и к автоматической активизации соответствующего дескриптора приемника и нет необходимости делать в него специальную запись. Отметим необходимость строго соблюдать последовательность действий. Информация на вершине FIFO статуса определяет номер текущего обслуживаемого буфера (дескриптора), поэтому её преждевременное удаление приведет к ошибке.

Таким образом, оптимальной последовательностью действий обслуживания приемника, можно считать:

– реакция на прерывание после записи информации в FIFO статуса (либо сканирование флага RX_nEMPTY регистра состояния);

– вычитка статусной информации из Rx_STATUS и определение размера принятых данных;

– вычитка всех данных из буфера RX_FIFO_DATA приемника;

– сброс FIFO статуса с помощью чтения регистра Rx_STATUS_POP_ACT с автоматической активизацией следующего приема в обслуженный буфер.

Описанную последовательность действий можно выполнить и с помощью канала DMA. В данном случае канал DMA должен выполнить цепочку из двух последовательных действий:

– после записи в FIFO статуса последнее вырабатывает запрос к DMA. Выполняется чтение буфера данных. Канал DMA всегда программируется на чтение максимально возможного объема данных, которые может принять приемник. При вычитке данных каналом DMA, буфер приемника специальным сигналом интерфейса информирует о вычитке последнего слова, что приводит к автоматическому завершению работы канала;

– следующим действием DMA является чтение регистра Rx_STATUS_POP_ACT, что приводит к сохранению принятого статуса и активизации обслуженного буфера.

При приеме фрейма возможно обнаружение ошибок. Алгоритм работы приемника реализован таким образом, что принятые данные всегда помещаются в буфер, а в дескриптор заносится соответствующая информация. О том, что обнаружена ошибка при приеме, можно узнать, если в регистре маски прерываний разрешить прерывание при наличии ошибочной ситуации. Также о наличии ошибки можно узнать при анализе слова состояния принятого фрейма.

Системный DMA может принимать фреймы с ошибками аналогично тому, как принимаются корректные фреймы. Описанные действия могут выполняться с помощью цепочки операций. Действия должны выполняться строго в приведенном порядке. Первое действие всегда ожидает флага готовности FIFO приемника. Последнее действие (чтение Rx_STATUS_POP_ACT) приводит к уменьшению указателя FIFO и означает готовность буфера (дескриптора) к приему нового фрейма. При обслуживании MACa с помощью DMA самым удобным вариантом является генерация прерывания к процессору каналом DMA после обслуживания текущего буфера. В этом случая вся информация уже размещена в оперативной памяти.

Описанный выше алгоритм работы имеет ряд недостатков:

– процессор (или DMA) должны ожидать пока в буфер данных не будет записано всё принятое сообщение и сформировано статусное слово. Только после этого можно производить вычитку данных;

– при работе с DMA заранее невозможно определить размер принятых данных и поэтому в системной памяти необходимо резервировать место для максимально возможного сообщения (8 Кбайт);

– очень «неудобным» случаем для приемника становится ситуация, когда длинные сообщения чередуются с короткими. В этом случае возможна потеря сообщений.

Для ускорения обслуживания приемника, реализован вариант, в котором буферная память приемника (16 Кбайт) рассматривается как единое FIFO данных. Статусное FIFO имеет глубину в 8 слов. Данный режим включается посредством установки бита RX_FFM в регистре MAC_FIFO_MODE. Структурную схему приемника можно представить в виде, как на рисунке 163.

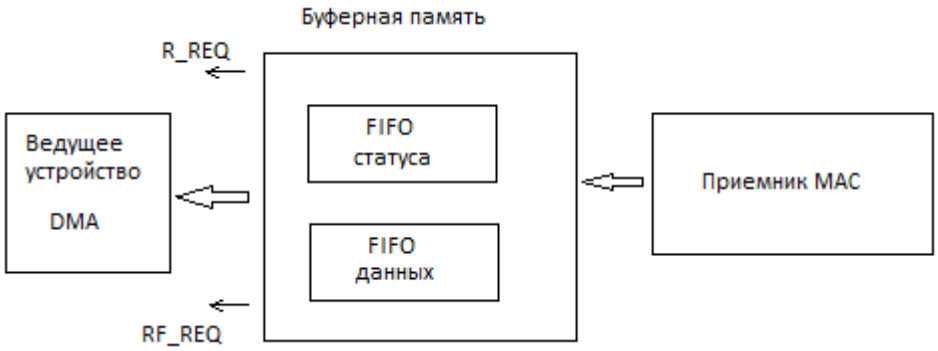


Рисунок 163 – Структурная схема приемника

В данном варианте приемник формирует два запроса к DMA на обслуживание – от FIFO данных и от FIFO статуса. Для обслуживания каждого запроса в DMA имеется индивидуальный канал. Особенность в данном режиме работы состоит в том, что запрос

от FIFO данных активизируется, когда принято 128 или более байт данных, т.е. восемь или более квадрослов. Соответствующий канал DMA в ответ на запрос всегда (должен) делает вычитку 8 квадрослов данных. Таким образом, к моменту завершения приема фрейма останется вычитать максимум 8 квадрослов данных. Если размер сообщения не кратен 128 байтам, то при завершении приема, т.е. при записи статуса в соответствующее FIFO, происходит автоматическое увеличение указателя FIFO данных до значения восьми квадрослов. Как только канал DMA завершает вычитку последних данных принятого фрейма, формируется запрос на чтение статусного FIFO. Другой канал DMA читает регистр Rx_STATUS_POP_ACT, что автоматически сбрасывает прочитанный статус и повторно активизирует только что обслуженный буфер для нового приема.

Данный подход имеет еще одно преимущество. Если канал DMA на некотором интервале времени не успевает оперативно обслуживать приемник (имеются более высокоприоритетные пересылки или другие обстоятельства), то буфер приемника способен принять до 8-ми сообщений, прежде чем прием будет остановлен. Каждый раз, когда в FIFO статуса записывается новое информационное слово, приемник проверяет, осталось ли в буфере данных достаточно места для приема следующего сообщения. Если место имеется, соответствующий дескриптор автоматически активизируется. В связи с тем, что приемник может принимать фреймы с максимальным размером до 8192 байта, приходится постоянно отслеживать, чтобы в буфере было достаточно места для приема самого длинного фрейма.

Имеются некоторые особенности в данном режиме при работе приемника в режиме полудуплекса. В этом случае необходимо иметь возможность отбрасывать фреймы, которые имеют размер менее 512 байт. Если установлен полудуплексный режим либо установлен бит HD_MODE в регистре MAC_FIFO_MODE, поведение FIFO данных приемника немного меняется. В этом случае запрос к DMA каналу формируется только в случае, когда:

- принято 512 или более байт данных,
- обнаружено завершение фрейма (запись статуса в FIFO).

Как и прежде, DMA вычитывает информации порциями в 8 квадрослов. Принятый фрейм данных также выравнивается до границы в 128 байт. Единственным неудобным моментом в данном случае является ожидание накопления большего количества данных для начала обслуживания.

При работе буфера приемника в режиме FIFO все принятые фреймы записываются непрерывно в одну выделенную область памяти. Каждый фрейм выровнен до границы в 128 байт. Полезная длина фрейма определяется по информации из статуса фрейма.

29.7.1 Структура принятых данных

Ранее была приведена общая структура фрейма передаваемых данных. Приемник обнаруживает начало фрейма по преамбуле и SFD, после чего начинает прием данных. В буфер приема начинается запись данных, начиная с адреса приемника. Правда только в том случае, если указанный адрес детектируется приемником как свой. Общая длина

принятых в буфер данных содержится в поле RX_SIZE слова состояния. Поле контрольной суммы не учитывается в этом размере. Если буфер приемника работает в режиме FIFO и используется канал DMA для вычитки данных, то вне зависимости от количества принятых в буфер приемника данных, итоговый размер данных, переданный с помощью DMA в системную память будет выровнен (в большую сторону) до границы кратной 128 байтам. Например, поле RX_SIZE равно 129. Реально в системную память будет записано 256 байт. Данное обстоятельство необходимо учитывать при расчете адреса местонахождения заданного фрейма в списке, относительно начального адреса буфера.

29.8 Общие замечания

Возможны другие варианты обслуживания интерфейса с различной степенью привлечения процессора и контроллера DMA. Самый простой вариант может использовать буферную память интерфейса для формирования фрейма процессором. Обработка принятого фрейма также может выполняться прямо в буфере без дополнительного копирования в промежуточный буфер. Использование системного DMA предполагает формирование задания на обработку в виде цепочки операций описывающих необходимую последовательность действий. DMA может выполнять все либо часть необходимых операций.

При работе на максимальной частоте, максимальная скорость передачи достигается в случае, когда к моменту окончания передачи текущего фрейма, следующий фрейм уже готов для передачи. В случае запаздывания готовности, возникают простои передатчика и таким образом снижается скорость передачи.

При работе на максимальной частоте, максимальная скорость приема достигается в случае, когда к моменту окончания приема текущего фрейма, предыдущий фрейм уже извлечен из буфера приемника и буфер готов для приема следующего фрейма. В случае запаздывания готовности следующего буфера, принимаемый фрейм может быть потерян, что потребует его повторной передачи и приема, и таким образом снижается скорость приема.

Высокоскоростной прием данных является наиболее трудноразрешимой задачей в случае наличия одного процессора и при одновременной его занятости выполнением каких-либо других операций. В этом случае может возникать проблема с достаточной скоростью реакции на прерывание. Для снижения требований к времени реакции на прерывание, может использоваться DMA с заранее запрограммированной цепочкой операций на прием некоторого числа фреймов без участия процессора. В любом случае, обработка некоторого числа фреймов процессором должна выполняться быстрее приема такого же числа следующих фреймов, в случае необходимости обеспечения максимальной скорости приема информации (без потерь и повторной передачи).

Работа передатчика не выдвигает никаких-либо определенных требований по наличию некоторого количества свободной памяти в системе. Как отмечалось выше, фрейм для передачи может формироваться прямо в буфере передатчика. Работа приемника может выдвигать определенные требования к наличию некоторого объема

буферной памяти. Если скорость реакции на прерывание не позволяет процессору самому эффективно принимать информацию, то может быть привлечен канал DMA с некоторым количеством выделенной буферной системной памяти. Размер этой памяти определяется минимальным количеством буферов, используемых DMA для приема сообщений. Размер каждого буфера определяется максимально возможной длиной принимаемого сообщения. Единственными требованиями к буферу (для достижения высокой скорости обмена) является его выравненность на границе квадрослова. Максимальный объем буфера зависит от максимального размера фрейма передаваемых данных, но не может превышать 8 Кбайт. Предпочтительно для буфера выделять непрерывную область памяти, т.к. в случае наличия разрывов необходим дополнительный элемент в цепочке операций.

Отмечу, что выполнение приема-передачи с помощью DMA предполагает:

- предварительную подготовку фреймов для передачи в промежуточном буфере;
- последующую обработку принятых в промежуточный буфер фреймов.

Таким образом, процессор может подготовить все данные для отправки, не привязываясь к скорости работы передатчика и выполнить обработку принятых данных в удобное время.

В зависимости от производительности процессора, характера потока приема-передачи, доступного объема оперативной памяти, наличия дополнительных фоновых процессов, возможен выбор различных вариантов работы с интерфейсом.

29.9 Примеры

Пример обслуживания MAC-интерфейса процессором посредством обработки прерываний от приемника и передатчика.

Этап 1. Инициализация интерфейса.

```
#define rmac_irq_n  78 // RX
#define tmac_irq_n  77 // TX

test_irq_MAC: j5 = base_MAC;;
        k0 = 1;;
                        [j5+30]  =k0;; // -- программный сброс
                        k1 = [j5+30];; // - задержка


        k0 = 2;;
                        [j5+28]  =k0;; // -- ПРЕРЫВАНИЕ ТОЛЬКО ПРИ ОШИБКЕ ПЕРЕДАЧИ


                        k1 = 0x00009abc;;
                        k0 = 0x12345678;;
                        [j5+2]  =k0;; // mac address
                        [j5+3]  =k1;;
```

```
k0 = 0x1e1;;
                [j5+26]  =k0;;// значение SEED

k0 = 0x00000040;;
                [j5+4]  =k0;;// hash значение

k0 = 0x110f;;
                [j5+0]  =k0;;          // TX-RX on


lc0 = 32;; // 20000;;
mac_i4_dly:   if nlc0e, jump mac_i4_dly;;// задержка при включении


k0 = 0x8000;;// активизация буферов приемника
                [j5+0x200]  =k0;;    // RX desc 0
                [j5+0x201]  =k0;;    // RX desc 1
```

Выполняется программный сброс. Далее разрешаются прерывания только при обнаружении ошибки при передаче. Задается собственный адрес интерфейса. Включается передатчик и приемник. Все дескрипторы приемника программируются как готовые к приему.

Этап 2. Подготовка контроллера прерываний.

```
J4 = MAC_RX_ISR;;

k0 = rmac_irq_n;;
[j31 + base_Core0 + base_L2_INSL + 15] = k0;;   // 15-е прерывание настраивается на
                                                //обработку запроса от приемника

[j31 + base_Core0 + base_L2_INTV + 15] = j4;;// задается вектор обработки
                                             //прерывания

k0 = 1;;
[j31 + base_Core0 + base_L2_IMSK + 15] = k0;; //  15-е прерывание разрешается
j4 = MAC_TX_ISR;;

k0 = tmac_irq_n;;
[j31 + base_Core0 + base_L2_INSL + 14] = k0;; // 14-е прерывание настраивается на
                                                //обработку запроса от передатчика


[j31 + base_Core0 + base_L2_INTV + 14] = j4;;// задается вектор обработки
                                             //прерывания
```

```
k0 = 1;;
[j31 + base_Core0 + base_L2_IMSK + 14] = k0;; //  14-е прерывание разрешается в
                                                  //контроллере прерываний.


SQCTLST = SQCTL_GIE;; // разрешаются прерывания в процессоре

k7 = 3+2;;     // регистр используется для задания количества передаваемых фреймов (3)
               //плюс два служебных

k0 = 0x103;;
[j31+base_MAC + 0x30] = k0;; // разрешаются прерывания от FIFO приемника и
                                 //передатчика в MAC модуле. Прерывания импульсные
```

В контроллере прерываний определяется вектор обработки прерывания для приемника и передатчика. Приемнику назначается 15-е прерывание, а передатчику 14-е. Разрешаются прерывания в процессоре. Далее разрешаются прерывания от FIFO передатчика и приемника в импульсном режиме. Тест запрограммирован на передачу и прием трех фреймов. В тесте выход передатчика соединен со входом приемника.

Этап 3. Обработка прерываний.

Как только мы разрешили прерывания, пойдет процесс генерации запросов на прерывание и их обслуживание. Простенький пример обработчика передатчика.

```
.align_code 4;
MAC_TX_ISR:          // tx service

j27 = j27-12(NF);;

q[j27+0] = j3:0;;
                q[j27+4] = xr3:0;;
                [j27+8]  = j31;;
                [j27+9]  = k31;; // сохраняем используемые регистры

j0 = [j31+base_MAC + 0x180];; // извлекаем информацию из текущего дескриптора

comp(k7,2); j1 = 0;;// анализ того, что осталась извлечь только статус последних двух
                      //переданных фреймов
                if keq, jump skip_tx_data (NP); comp(k7,1);;
                if keq, jump skip_tx_data (NP);;

                lc0 = (24/4);;
j3 = base_TX_fifo;;
                j2 = txa_data;;// передаем в буфер FIFO заданный фрейм. В тесте он
                                 //всегда один и тот же.
```

```
rep_itx_ou_data:
                    xr3:0 =  q[j2+=4];;
                    if nlc0e, jump rep_itx_ou_data;  q[j3+=4] = xr3:0;;

                    j1 =  0x00488000;;  // дескриптор с длиной и битом ACT равным 1
skip_tx_data:
                    k7 = k7 - 1;;// алализ того что переданы все фреймы. После чего запрос
                                 //прерывания от передатчика будет выключен.
                    j2 = 0x102;;  // disable TX
                    if keq; do, [j31+base_MAC + 0x30] = j2;;

                    [j31+base_MAC + 0x180] = j1;; // запись текущего дескриптора

 // восстановление контекста
                    j3:0 = q[j27+=4];;
                    xr3:0 = q[j27+=4];;
                    j31   = [j27+=1];;
                    k31   = [j27+=3];;

        rti (abs)(np);;
```

Передатчик запрограммирован на передачу 3-х фреймов. Вначале вычитывает статус, затем пересылает заранее подготовленные данные (в примере одни и те же) и в конце записывает активный дескриптор. Попутно проверяется , что 3 фрейма переданы и дополнительно будут произведены две записи в регистр дескриптора с битом ACT равным нулю. Это упрощает процесс вычитки статуса переданных фреймов.

Пример обработчика приемника

```
.align_code 4;
MAC_RX_ISR:          // rx service

        j27 = j27-12(NF);;

        q[j27+0] = j3:0;;
                    q[j27+4] = xr3:0;;
                    [j27+8]  = j31;; // сохраняем используемые регистры

        j2 = [j31+base_MAC+0x36];; // извлекаем информацию о длине фрейма
                    j3 = base_RX_fifo;;
                    j2 = j2 and 0x3FF0;;
                    j2 = lshiftr j2 by 4;;  // определяем количество  квадрослов которые
                                            //необходимо вычитать из буфера приемника

get_rx_in_data:
                    xr3:0 =  q[j3+=4];;// извлекаем данные. В этом цикле должна быть
```

```
                                    //добавлена обработка принятых данных. Как
                                    //минимум – пересылка в другой буфер.
                        if njeq, jump get_rx_in_data; j2 = j2 - 1;;


        j0 = [j31 + base_MAC + 0x3E];; // вычитываем текущий дескриптор и активизируем
                                    //буфер на прием следующего фрейма

// восстановление контекста

                         j3:0 = q[j27+=4];;
                        xr3:0 = q[j27+=4];;
                        j31   = [j27+=4];;

              rti (abs)(np);;
```

Вычитывается размер принятых данных. Вычитываются данные, затем продвигается статусное FIFO и устанавливается активное значение дескриптора приемника. При вычитке размера доступна также информация о статусе принятого фрейма. Если фрейм принят с ошибкой и нет необходимости вычитывать ошибочные данные, то стадию вычитки данных можно пропустить. Однако, интерфейс работает так, что без вычитки последнего квадрослова принятого фрейма, не выполнится разблокировка дескриптора. Поэтому к базовому адресу буфера необходимо добавить длину фрейма (количество слов) и произвести фиктивное чтение. Это вызовет разблокировку дескриптора и его затем можно активизировать.

Аналогичный пример задания можно выполнить и с помощью системного контроллера DMA. Для этого необходимо сформировать задание в виде цепочки операций.

Исходное задание

```
// массив подготовленных для передачи дескрипторов
.var  tx_desc[3] ={ 0x004c8800,// дескриптор 0
                    0x00488000,// дескриптор 1
                    0x00000000 }; // доп дескриптор

// область памяти для сохранения статуса переданных фреймов
.var  tx_status[2] = {0,0};
// область памяти для сохранения статуса принятых фреймов
.var  rx_status[2] = {0,0};

.align  4;
.var txa_data[24] ={0x78563412,
    0x3412bc9a,
     .. подготовленный для передачи фрейм 0};
.align  4;
.var txb_data[24] ={0x78563412,
```

```
    0x3412bc9a,
     .. подготовленный для передачи фрейм 1};

#define  Address_base  0x100000  // память L3
#define EDMA_mac0_Tx  32
```

Цепочка операций для передачи

В данном примере будут переданы и приняты два фрейма. Передатчику MAC интерфейса выделен канал номер 32 контроллера прямого доступа. Конфигурирование канала выполняется при помощи следующей последовательности действий.

Канал передатчика

```
  xr0 = 0x00c00000 + (62<<12) + (6<<8) + (6<<4);;
  xr1 = Address_base+tx_mac_chain;;  // src
  xr2 = (8<<16) + 32;; // b a cnt
  xr3 = base_PaRAM+(62*8);; // dst

  xr4 =0x00000008;; // B  D.S IDX
  xr5 =0x0000ffff;; // link to dummy
  xr6 =0x00000000;; // C  D.S IDX
  xr7 =0x00000001;; // ccnt

     q[j31+base_PaRAM+(EDMA_mac0_Tx*8)]   =xr3:0;;
     q[j31+base_PaRAM+(EDMA_mac0_Tx*8)+4] =xr7:4;;
```

Канал передатчика (32-й) программируется на загрузку задания в дополнительный канал 62. Задания для передатчика находятся по адресу tx_mac_chain. В примере канал запрограммирован на выполнение 8 заданий. После загрузки в 62-й канал активного задания, канал 62 автоматически активируется на выполнение. Как только 62-й канал выполнит первое задание, он тут же активирует 32-й канал, который подзагружает в канал 62 новое задание. После выполнения 3-го задания, от канала 62 к каналу 32 не поступает сигнала для активирования. Вместо этого канал 32 ожидает импульса активации от FIFO передатчика.

Цепочки операций должны быть подготовлены до момента запуска каналов.

```
.align 4;
.var tx_mac_chain[80] = {
// buffer 0
        0x00400000 + (EDMA_mac0_Tx<<12) + (1<<11) + (6<<8) + (6<<4) + (1<<3),
   Address_base+txa_data, 0x00010000+80, base_TX_fifo, // src, cnt, dst
   0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

   0x00000000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3),
   Address_base+tx_desc, 0x00010004, base_MAC0+0x180, // src, cnt, dst
   0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt
```

```
// buffer 1
      0x00400000 + (EDMA_mac0_Tx<<12) + (1<<11) + (6<<8) + (6<<4) + (1<<3), //
  Address_base+txb_data, 0x00010000+80, base_TX_fifo, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static I=>I
  Address_base+tx_desc+1, 0x00010004, base_MAC0+0x180, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

// last 2 desc

  0x00400000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //
  base_MAC0+0x180, 0x00010004, Address_base+tx_status+2, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static I=>I
  Address_base+tx_desc+2, 0x00010004, base_MAC0+0x180, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00400000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //
  base_MAC0+0x180, 0x00010004, Address_base+tx_status+3, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Tx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static I=>I
  Address_base+tx_desc+3, 0x00010004, base_MAC0+0x180, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1}; // offs,link, ccnt
```

Для передачи одного фрейма необходимо в общем случае три отдельные задания:

- вычитка статуса ранее переданного фрейма;
- подзагрузка данных нового фрейма;
- запись активного дескриптора нового фрейма.

При передаче первых двух фреймов нет необходимости в вычитке статуса. Однако после передачи последнего фрейма есть необходимость в вычитке статуса двух последних переданных фреймов. Делается это при помощи фиктивной записи неактивного (нулевого) значения дескриптора.

Цепочка операций для приема

Последовательность программирования каналов DMA при приеме данных зависит от режима работы буфера данных приемника. Рассмотрим случай, когда RX_FFM равен нулю, т.е. режим «память». В данном случае программирование канала DMA приемника напоминает программирование аналогичного канала для передатчика. Приемник имеет фиксированный канал DMA ассоциированный с FIFO статуса

#define EDMA_mac0_Rx 30

Данный канал программируется следующим образом (пример).

```
xr0 = 0x00c00000 + (61<<12) + (6<<8) + (6<<4);;
xr1 = Address_base+rx_mac_chain;;  // src
xr2 = (6<<16) + 32;; // b a cnt
xr3 = base_PaRAM+(61*8);; // dst

xr4 =0x00000008;; // B  D.S IDX
xr5 =0x0000ffff;; // link to dummy
xr6 =0x00000000;; // C  D.S IDX
xr7 =0x00000001;; // ccnt

    q[j31+base_PaRAM+(EDMA_mac0_Rx*8)]   =xr3:0;;
    q[j31+base_PaRAM+(EDMA_mac0_Rx*8)+4] =xr7:4;;
```

Используется вспомогательный канал номер 61 в который загружается очередное задание. Базовый список заданий находится по адресу rx_mac_chain. Рассмотрим пример такого списка заданий

```
.align 4;
.var rx_mac_chain[6*8] = {

      0x00400000 + (EDMA_mac0_Rx<<12) + (1<<11) + (6<<8) + (6<<4) + (1<<3), //
  base_RX_fifo, 0x00010000+2000, 0x00140000, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Rx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static
  base_MAC0+mac_Rx_status, 0x00010004, Address_base+rx_status, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00400000 + (EDMA_mac0_Rx<<12) + (1<<11) + (6<<8) + (6<<4) + (1<<3), //   static
  base_RX_fifo, 0x00010000+2000, 0x00140100, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Rx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static
  base_MAC0+mac_Rx_status, 0x00010004, Address_base+rx_status+1, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt


      0x00400000 + (EDMA_mac0_Rx<<12) + (1<<11) + (6<<8) + (6<<4) + (1<<3), //
  base_RX_fifo, 0x00010000+2000, 0x00140200, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1, // offs,link, ccnt

  0x00000000 + (EDMA_mac0_Rx<<12) + (1<<11) + (2<<8) + (2<<4) + (1<<3), //   static
  base_MAC0+mac_Rx_status, 0x00010004, Address_base+rx_status+2, // src, cnt, dst
  0x00000000, 0xFFFF, 0, 1}; // offs,link, ccnt
```

В данном примере описан прием 3-х фреймов. Каждый фрейм требует выполнения двух заданий. Загрузка статусного слова в FIFO статуса приемника активизирует запрос R_REQ, который вызывает срабатывание канала 30. Данный канал подзагружает в канал номер 61 первое задание. Его функция состоит в копировании принятых данных из текущего буфера в оперативную память. Канал программируется на максимальную длину фрейма, однако буфер приемника отслеживает чтение последнего квадрослова и информирует DMA канал о преждевременном завершении. После завершения копирования данных, канал 61 активизирует канал 31. Канал 31 считывает следующее задание и загружает в канал 61. Следующее задание выполняет чтение регистра статусного FIFO. В качестве адреса берется значение

```
#define mac_Rx_status   0x3E  // pop ACT =1
```

Это позволяет прочитать статус принятого фрейма, продвинуть указатель и автоматически активировать обслуженный буфер для нового приема.

В цепочке операций можно указать (при необходимости) моменты времени, когда можно генерировать прерывание для процессора. Описанная последовательность заданий предполагает, что в оперативной памяти определен массив rx_status для приема статусной информации, а также зарезервирован буфер памяти достаточного размера для приема данных (в приведенном примере начальные адреса буферов 0x00140000, 0x00140100, 0x00140200.

Рассмотрим случай организации приема в режиме, когда RX_FFM равен 1, т.е. режим «FIFO». В этом случае буфер приемника формирует два аппаратных запроса к DMA: первый от FIFO статуса и второй от FIFO данных.

```
#define EDMA_mac0_fRx  1
```

Программируются два канала DMA. Канал статусного FIFO

```
xr0 = 0x00000000 + (1<<OPT_TCINTEN) + (EDMA_mac0_Rx<<12) + (2<<8) + (2<<4);;
xr1 = base_MAC0+mac_Rx_status;;  // src
xr2 = (8<<16) + 4;; // b a cnt
xr3 = Address_base+frx_stat;; // dst

xr4 = (1<<16) + 0;; // B  D.S IDX
xr5 =0x0000ffff;; // link to dummy
xr6 =0x00000000;; // C  D.S IDX
xr7 =0x00000001;; // ccnt

   q[j31+base_PaRAM+(EDMA_mac0_Rx*8)]   =xr3:0;;
   q[j31+base_PaRAM+(EDMA_mac0_Rx*8)+4] =xr7:4;;
```

Канал предназначен для чтения статуса принятого фрейма и записи его значения в массив frx_stat. Каждая активация канала вызывает одну пересылку.

Канал FIFO данных определяется как

```
xr0 = 0x00000000 + (EDMA_mac0_fRx<<12) + (6<<8) + (6<<4);;
xr1 = base_RX_MAC;;  // src
xr2 = (200<<16) + (16*8);; // b a cnt
xr3 = 0x0011a000;; // dst

xr4 =(32<<16) + 0;; // B  D.S IDX
xr5 =0x0000ffff;; // link to dummy
xr6 =(0<<16) + 0;; // C  D.S IDX
xr7 =0x00000001;; // ccnt

    q[j31+base_PaRAM+(EDMA_mac0_fRx*8)]   =xr3:0;;
    q[j31+base_PaRAM+(EDMA_mac0_fRx*8)+4] =xr7:4;;
```

Канал программируется на однократную вычитку 128 байт (восемь квадрослов) данных, т.е. каждая активация запроса от FIFO данных приемника к каналу 1 DMA будет вызывать пересылку восьми квадрослов данных. Можно задать максимальный объем принимаемых данных (для некоторого количества фреймов). По завершении приема запрограммированного объема данных канал может перезагрузиться новым значением.

В данном алгоритме каналы DMA работают более оперативно, т.к. им не нужно подзагружать задания. Они всегда готовы к работе. Канал DMA статусного FIFO может генерировать прерывание после приема каждого статусного слова, что будет означать наличие принятого фрейма в оперативной памяти. Данные всех принятых фреймов располагаются в одном непрерывном буфере. Каждый фрейм выровнен до границы в 8 квадрослов. Реальное местонахождение фрейма можно определить на основании статусной информации и базового адреса буфера приема.

29.10 Поддержка jumbo-фреймов

Jumbo-фрейм (англ. jumbo frame) — Ethernet-фрейм, в котором можно передать данные, по размеру превышающие 2000 байт. Поддержка jumbo-фреймов выходит за рамки стандарта IEEE 802.3 Ethernet. Как правило, размер jumbo-фрейма не превышает 9000 байт, однако существуют различные вариации этого размера. Данный блок Ethernet MAC поддерживает jumbo-фреймы размером до 8192 байт, что является удобным размером для внутренних двухпортовых SRAM.

Включение поддержки jumbo-фреймов влияет на размер каждого внешнего и внутреннего буфера TX и RX (8192 байта вместо 2048 байт), а также на эффективную длину полей LENGTH в дескрипторах TX и RX (см. подразделы 29.3 «Дескрипторы TX» и 29.4 «Дескрипторы RX»).

29.11 Поддержка функциональности Layer Management

29.11.1 Общие положения Layer Management

Спецификация Layer Management стандарта IEEE 802.3 Ethernet описывает средства, составляющие запросы различных атрибутов (статистик) слоя MAC и запросы на выполнение определённых действий по управлению слоем MAC (подмножество DTE Management, класс oMACEntity Managed Object Class). Поддержка функциональности

Layer Management является необязательной для Ethernet MAC. Различные атрибуты и действия класса oMACEntity Managed Object Class сгруппированы в пакеты функциональности. Данный блок Ethernet MAC поддерживает два обязательных пакета функций класса oMACEntity Managed Object Class (Basic Package и Mandatory Package), а также один необязательный пакет – Recommended Package (рекомендуемый пакет). Таким образом, в данном блоке Ethernet MAC присутствует поддержка функциональности Layer Management на рекомендуемом уровне функциональности.

29.11.2 Использование функциональности Layer Management

В данном подразделе в табличной форме предоставлена информация о поддержке атрибутов и действий класса oMACEntity Managed Object Class по трём поддерживаемым пакетам (см таблицы 329 – 331). В данном блоке Ethernet MAC поддержка функций Layer Management организована таким образом, что информация, необходимая для поддержки Layer Management выводится в дескрипторы и регистры Ethernet MAC, а драйвер Ethernet MAC использует эту информацию для генерации требуемых статистик MAC и завершения требуемых действий по управлению слоем MAC. Таким образом, поддержка Layer Management для данного блока Ethernet MAC обеспечивается как аппаратными, так и программными средствами.

Полное описание атрибутов и действий Layer Management, поддерживаемых в данном блоке Ethernet MAC, расположено в разделе спецификации IEEE 802.3 SECTION TWO «30. Management», а именно в подразделе «30.3.1 MAC entity managed object class».

Таблица 329 – Поддержка атрибутов и действий пакета Basic Package

Атрибут или действие	Описание способа поддержки атрибута или действия
aMACID	Данный атрибут расположен на стороне драйвера
acInitializeMAC	Для инициализации блока Ethernet MAC используется программный сброс, доступный с применением регистра SOFTWARE_RESET

Таблица 330 – Поддержка атрибутов и действий пакета Mandatory Package

Атрибут или действие	Описание способа поддержки атрибута или действия
aFramesTransmittedOK	Если при окончании передачи значение поля TX_DN_ST дескриптора TX равно TRANSMIT_OK, то драйвер должен увеличить свой счётчик aFramesTransmittedOK на 1
aSingleCollisionFrames	Для обновления счётчика aSingleCollisionFrames драйвером используется поле RETRY_ATTEMPT дескриптора TX
aMultipleCollisionFrames	Для обновления счётчика aMultipleCollisionFrames драйвером используется поле RETRY_ATTEMPT дескриптора TX.

Атрибут или действие	Описание способа поддержки атрибута или действия
aFramesReceivedOK	Если при окончании приёма значение поля RX_DN_ST дескриптора RX равно RECEIVE_OK, то драйвер должен увеличить свой счётчик aFramesReceivedOK на 1
aFrameCheckSequenceErrors	Если при окончании приёма значение поля RX_DN_ST дескриптора RX равно FRAME_CHECK_ERROR, то драйвер должен увеличить свой счётчик aFrameCheckSequenceErrors на 1
aAlignmentErrors	Если при окончании приёма значение поля RX_DN_ST дескриптора RX равно ALIGNMENT_ERROR, то драйвер должен увеличить свой счётчик aAlignmentErrors на 1
aMACCapabilities	Данный блок Ethernet MAC поддерживает как режим full duplex, так и режим half duplex. Драйвер должен сообщать об этом факте в атрибуте aMACCapabilities
aDuplexStatus	В атрибуте aDuplexStatus должно содержаться значение поля «TX и RX: разряд включения режима Half Duplex» регистра MODE
aRateControlAbility	Данный блок Ethernet MAC не поддерживает как функцию Rate Control, поэтому атрибут aRateControlAbility всегда должен читаться как «false»
aRateControlStatus	Данный блок Ethernet MAC не поддерживает как функцию Rate Control, поэтому атрибут aRateControlStatus по чтению и записи должен отражать этот факт
aDeferControlAbility	Данный блок Ethernet MAC самостоятельно контролирует расстояние между соседними фреймами, поэтому атрибут aDeferControlAbility должен всегда читаться как «true»
aDeferControlStatus	Данный блок Ethernet MAC самостоятельно контролирует расстояние между соседними фреймами и не может отключить эту функцию, поэтому атрибут aDeferControlStatus по чтению и записи должен отражать этот факт

Таблица 331 – Поддержка атрибутов и действий пакета Recommended Package

Атрибут или действие	Описание способа поддержки атрибута или действия
aOctetsTransmittedOK	Если значение поля TX_DN_ST дескриптора TX равно TRANSMIT_OK, то драйвер должен увеличить счётчик aOctetsTransmittedOK согласно значениям, хранящимся в поле LENGTH и в поле FPR. Нужно отметить, что размер фрейма не может быть меньше 64 байт, и в случае малого размера фрейма TX автоматически дополняет фрейм до 64

Атрибут или действие	Описание способа поддержки атрибута или действия
	байт (padding), и такие дополнительные байты также учитываются в счётчике aOctetsTransmittedOK
aFramesWithDeferredXmissions	При обновлении этого счётчика учитывается поле RETRY_ATTEMPT и поле DEF дескриптора TX. Если фрейм отправлен с первой попытки (в режиме half duplex), тогда если поле DEF = 1, то драйвер должен увеличить значение счётчика aFramesWithDeferredXmissions на 1
aLateCollisions	Значение атрибута aLateCollisions содержится в регистре LATE_COLLISIONS_COUNTER
aFramesAbortedDueToXSColls	Если при окончании передачи значение поля TX_DN_ST дескриптора TX равно EXCESSIVE_COLLISION_ ERROR, то драйвер должен увеличить свой счётчик aFramesAbortedDueToXSColls на 1
aFramesLostDueToIntMACXmitError	Данный блок Ethernet MAC не внедряет никаких специфических для него (implementation-specific) статусов ошибки передачи, поэтому значение данного счётчика aFramesLostDueToIntMACXmitError никогда не должно быть отличным от нуля
aCarrierSenseErrors	При обновлении этого счётчика учитывается поле CER дескриптора TX. Если значение этого поля 1, то драйвер должен увеличить значение счётчика aCarrierSenseErrors на 1
aOctetsReceivedOK	Если значение поля RX_DN_ST дескриптора RX равно RECEIVE_OK, то драйвер должен увеличить счётчик aOctetsReceivedOK согласно значению, хранящемуся в поле LENGTH
aFramesLostDueToIntMACRcvError	Данный блок Ethernet MAC не внедряет никаких специфических для него (implementation-specific) статусов ошибки приёма, поэтому значение данного счётчика aFramesLostDueToIntMACRcvError никогда не должно быть отличным от нуля
aPromiscuousStatus	Данный блок Ethernet MAC поддерживает функцию LayerMgmtRecognizeAddress, и поле «RX: разряд включения режима Promiscuous Mode» соответствует атрибуту aPromiscuousStatus
aReadMulticastAddressList	Данный блок Ethernet MAC поддерживает функцию RecognizeAddress. Адреса типа multicast хранятся в неявном виде в таблице хешей, которая расположена в регистрах от MULTICAST_ADDRESS_HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15

Атрибут или действие	Описание способа поддержки атрибута или действия
acAddGroupAddress	Данный блок Ethernet MAC поддерживает функцию RecognizeAddress. Адреса типа multicast хранятся в неявном виде в таблице хешей, которая расположена в регистрах от MULTICAST_ADDRESS_HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15
acDeleteGroupAddress	Данный блок Ethernet MAC поддерживает функцию RecognizeAddress. Адреса типа multicast хранятся в неявном виде в таблице хешей, которая расположена в регистрах от MULTICAST_ADDRESS_HASH_TABLE_0 до MULTICAST_ADDRESS_HASH_TABLE_15

29.12 Интерфейс RMII/RGMII

Для подключения к внешней микросхемы физического уровня служит интерфейс RMII/RGMII. Выбор требуемого инетрфейса осуществляется посредством бита 27 RMII_EN регистра CFG1. Распредение выводов интерфейсов на выводы микросхемы приведено в таблицах 643 и 332.

Таблица 332 – Назначение выводов интерфейса RMII

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PA[19]	REF_CLK	I	Вход. Опорный синхросигнал 50 МГц
PA[20]	-	-	Не используется
PA[21]	RXB[0]	I	Вход данных, бит 0
PA[22]	RXB[1]	I	Вход данных, бит 1
PA[23]	RX_ER	I	Ошибка приема
PA[24]	CRS_DV	I	Служебная информация. Carrier Sense/Receive data valid
PA[25]	-	-	Не используется
PA[26]	-	-	Не используется
PA[27]	TXD[0]	O	Выход данных, бит 0
PA[28]	TXD[1]	O	Выход данных, бит 1
PA[29]	TX_EN	O	Разрешение передачи
PA[30]	-	-	Не используется

Таблица 333 – Назначение выводов интерфейса RGMII

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PA[19]	RXC	I	Вход синхросигнала
PA[20]	RX_CTL	I	Вход служебных сигналов
PA[21]	RD[0]	I	Вход данных, бит 0

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PA[22]	RD[1]	I	Вход данных, бит 1
PA[23]	RD[2]	I	Вход данных, бит 2
PA[24]	RD[3]	I	Вход данных, бит 3
PA[25]	TXC	I	Выход синхросигнала
PA[26]	TX_CTL	I	Выход служебных данных
PA[27]	TD[0]	I	Выход данных, бит 0
PA[28]	TD[1]	I	Выход данных, бит 1
PA[29]	TD[2]	I	Выход данных, бит 2
PA[30]	TD[3]		Выход данных, бит 3

Таблица 334 – Назначение выводов интерфейса MDIO

Обозначение вывода (основное)	Обозначение вывода контроллера	Тип вывода	Функциональное назначение
PC[11]	MDIO	I/O	Линия данных
PC[12]	MDC	O	Синхросигнал

Блок полностью автономный и работа с контроллером MAC не требует никакой настройки. Единственное, что может понадобиться, это скорректировать временные параметры приема и выдачи сигналов RMII/RGMII независимыми линиями задержки. Величина задержки подбирается через регистры CFG14, CFG15 Модуля управления синхронизацией и энергопотреблением.

30 Порт JTAG и интерфейс отладки

Процессор поддерживает IEEE Standard 1149.1 Joint Test Action Group (JTAG) порт для тестирования системы. Этот стандарт определяет метод последовательного сканирования состояния входов-выходов для каждого компонента системы. Последовательный порт JTAG также используется эмулятором для получения доступа к встроенным возможностям отладки процессора.

Описываемые ниже функции реализованы в аппаратной части процессора для того, чтобы позволить программе-отладчику и операционным системам выполнять более сложные задачи. Функциональность разбита на группы по нескольким уровням. Эти уровни выстроены друг над другом, что позволяет отладчику или ядру ОС осуществлять более качественный контроль и анализ процессора.

Отладчик может осуществлять контроль процессора двумя путями:

– используя отладочный монитор (работающий в режиме супервизора);

– используя встроенный эмулятор (ICE).

При отладке с помощью программного монитора в режиме супервизора, некоторые системные ресурсы используются только монитором (например, память, прерывание, флаги и т. п.).

При отладке с использованием встроенного эмулятора, используется резервный коммуникационный канал (порт доступа к тестированию JTAG) для контроля за процессором.

Порт JTAG является подчиненным устройством на SOC-шине. Многие отладочные функции являются востребованными как при использовании программного метода отладки, так и при аппаратном.

Функции и возможности процессора для поддержки отладки программ включают:

– Точки наблюдения

Позволяет пользователю указывать конкретный диапазон адресов точки наблюдения и условия, которые останавливают процессор. После остановки, пользователь может проанализировать состояние процессора, восстановить прежнее состояние и продолжить выполнение команд. Программные точки останова и одношаговые операции также выполняются с помощью точки наблюдения.

– Запись истории

Специальный буфер трасс позволяет проследить за программным счетчиком с целью восстановления хода программы. Буфер трассировки на кристалле сохраняет последние 32 перехода (все дискретные значения счетчика программ) выполненные устройством управления ядра, позволяя восстановить последний программный путь.

– Счетчик тактов

Обеспечивает функцию подсчета количества процессорных тактов для всех функций программного кода.

– Мониторинг производительности

Позволяет осуществлять мониторинг некоторых внутренних событий ядра процессора при выполнении программного кода. Отладчик может точно указать, какое из внутренних событий нуждается в мониторинге.

– Доступ к защищенным регистрам

Защищенные регистры доступны только в режиме супервизора или ICE. К ним нет доступа в режиме пользователя.

– Счетчики точки наблюдения

Позволяет пользователю искать **n**-е возникновение события до остановки программы.

– Исключение

Событие, которое отменяет выполнение всех последующих команд.

30.1 Рабочие режимы

Процессор работает в одном из трех режимов: режим эмулятора, супервизора или пользователя. В режимах пользователя и супервизора все команды выполняются стандартно. Тем не менее, в режиме пользователя ограничен доступ к регистрам. В режиме эмуляции имеются ограничения на использование некоторых типов команд.

30.2 Ресурсы отладки

В процессоре предусмотрены несколько ресурсов отладки, включая:

– специальные команды;

– точки наблюдения;

– буфер трассировки адреса команды.

30.2.1 Специальные команды

Процессор поддерживает специальные команды, которые используются для помощи в отладке системы. Данные команды нужны для выполнения программных остановов в отладчиках и вызовов операционной системы.

30.2.2 Точки наблюдения

Точки наблюдения адреса позволяют пользователю проверять состояние процессора во всех случаях, когда происходит выполнение условий доступа к памяти, определенной пользователем. Существует три точки наблюдения, которые работают параллельно. Во время работы каждой точки наблюдения пользователь может определять условия срабатывания при доступе к шине и действия, которые должен выполнить процессор.

Точка наблюдения имеет два адресных регистра: WPiL и WPiH. Когда точка наблюдения запрограммирована на один адрес, используется только регистр с младшим адресом (WPiL). Если точка наблюдения запрограммирована на диапазон адресов, в WPiL содержится младший адрес, а в WPiH – старший адрес. Необходимо установить регистры указателя адреса до того, как установлен регистр управления WPiCTL, который переводит точку наблюдения в активное состояние. Адресные регистры точки наблюдения не могут быть изменены, пока точка наблюдения активна.

Работа точки наблюдения определяется ее регистром управления WPiCTL, (где: i – это 0, 1 или 2 в зависимости от номера точки наблюдения).

Следующая информация должна быть запрограммирована в регистрах до начала поиска:

– адрес или диапазон адресов;

– отсчет.

Когда регистр управления устанавливается в активное состояние, поле отсчета устанавливается в начало отсчета. С этого момента аппаратная часть точки наблюдения отслеживается внутренние шины с целью совпадения адресов передач, что указывается в регистрах точек наблюдения.

При каждом совпадении значение счетчика уменьшается. После того, как счетчик достигает значение нуля, инициируется исключение: программное исключение или вызов эмулятора. Выбор действий определяется разрядами 3-2 регистра WPiCTL.

С этого момента регистр состояния указывает на завершение работы точки наблюдения. Чтобы начать новый поиск, пользователю необходимо снова записать регистр управления. Три точки наблюдения связаны с тремя различными шинами.

Точка наблюдения 0 используется для мониторинга на I-шине. Точка наблюдения 0 поддерживает пошаговую опцию (SSTP), которая используется при эмуляции для выполнения одношаговых операций.

Точка наблюдения 1 используется для наблюдения за доступом к данным через J- и K-шины. При этом мониторинг может выполняться для одной из шин или для обеих сразу. Точка наблюдения 1 поддерживает также выбор типа доступа: только чтение, только запись, чтение и запись.

Точка наблюдения 2 используется для наблюдения за доступом к данным через S-шину. Точка наблюдения 2 поддерживает также выбор типа доступа: только чтение, только запись, чтение и запись.

Регистр WPiSTAT указывает на текущее состояние точки наблюдения. Он хранит текущее значение счетчика поиска и режима работы точки наблюдения. Описание разрядов регистра состояния приведено в таблице 335.

Таблица 335 – Описание разрядов регистра WPiSTAT

Разряды	Имя	Опиание
15–0	VALUE	Значение точки наблюдения
		Определяет текущее значения счетчика точки наблюдения
17–16	EX	Разряд выполнения
		Выполнение точки наблюдения: 00 – точка наблюдения запрещена; 01 – поиск совпадений; 11 – окончание работы счетчика точки наблюдения
31–18	Reserved	Резервные

30.2.3 Буфер трассировки адреса команды (TBUF)

Поскольку программный счетчик недоступен вне кристалла в реальном времени в процессе выполнения команд, то буфер трассировки используется для помощи в отладке программы.

Буфер трассировки включает в себя 32 регистра, в которых хранится история последних от 16 до 32 переходов, выполненных устройством управления ядра процессора, что дает возможность пользователю полностью восстановить путь выполнения программы в течение последних переходов.

Переходы циклично записываются в регистры буфера трассировки. Первый записывается в TRCB0, следующий в TRCB1, и т. д. до TRCB31. Тридцать второй переход снова записывается в TRCB0, и так далее. Для того, чтобы определить какой из регистров был записан последним, пользователь должен обратиться к регистру указателя буфера трассировки, который указывает на последний записанный вход.

Буфер трассировки всегда запоминает адрес команды, вызвавшей переход. Если адрес перехода вычислялся с использованием дополнительного регистра (не только PC и смешение в коде команды), то кроме адреса команды перехода в буфере трасс запоминается и адрес перехода.

Для того, чтобы различать какой тип информации записан в линии буфера трасс, используется регистр маски TRCBMASK. Каждый разряд в RCBMASK связан с соответствующим регистром TRCBi. Если он сброшен, TRCBi хранит адрес команды перехода. Если же он установлен, TRCBi содержит адрес перехода.

30.3 Мониторинг производительности

Целью мониторинга производительности является оптимизация функционирования рабочих приложений. Аппаратные возможности процессора обеспечивают подсчет числа циклов, которые были затрачены на выполнение кода программы или подсчет числа заданных событий при выполнении программы. Мониторинг обеспечивается с помощью трех регистров: счетчик циклов, маска монитора производительности и счетчик монитора производительности.

Счетчики циклов и монитора производительности должны быть установлены пользователем в ноль перед тем, как начнется определенный период работы. Соотношение между общим счетчиком и счетчиком монитора производительности и дает требуемое значение.

Когда счетчик монитора производительности переполняется, это вызывает исключительную ситуацию.

30.3.1 Регистр маски монитора производительности (PRFM)

Регистр маски монитора производительности (PRFM) определяет, какие события в процессоре отслеживаются и подсчитываются счетчиком монитора производительности (PRFCNT). Значением сброса регистра PRFM является 0x0000 0000. Подробное описание разрядов регистра приведено в таблице 336.

Таблица 336 – Биты регистра PRFM

Бит	Имя	Назначение
7:0	-	Резерв
8	Jexe	Подсчет команд, выполненных на линии конвейера J: 1 – включен; 0 – выключен
9	Kexe	Подсчет команд, выполненных на линии конвейера К: 1 – включен; 0 – выключен
10	Xexe	Подсчет команд, выполненных на линиях конвейера X1 и X2: 1 – включен; 0 – выключен
11	Yexe	Подсчет команд, выполненных на линиях конвейера У1 и У2: 1 – включен; 0 – выключен
12	Sexe	Подсчет команд, выполненных на линии конвейера S: 1 – включен; 0 – выключен
15:13	-	резерв
16	STALL	Подсчет команд остановов конвейера вызванных ожиданием данных или другими блокирующими конвейер ситуациями: 1 – включен; 0 – выключен
17	BTB_true	Подсчет количества правильных предсказаний буфера переходов: 1 – включен; 0 – выключен
18	ABORT	Подсчет количества программных исключительных ситуаций: 1 – включен; 0 – выключен
19	Uexe	Подсчет количества линий команд, выполненных в режиме пользователя: 1 – включен; 0 – выключен
20	BTB_false	Подсчет количества ошибочных предсказаний буфера переходов: 1 – включен; 0 - выключен
21	BTB_load	Подсчет количества загрузок в буфер переходов: 1 – включен; 0 – выключен
31:22	-	Резерв

Все события, которые отслеживаются монитором производительности суммируются по «ИЛИ» и при их наступлении происходит увеличение счетчика PRFM на единицу. Логичным является разрешение подсчета одного условия из многих.

30.3.2 Регистр счетчика монитора производительности (PRFCNT)

Назначение счетчика производительности – увеличивать свое значение на единицу каждый раз, когда происходит отслеживаемое событие. Отслеживаемое событие определяется регистром маски монитора производительности (PRFM). Счет прекращается, когда процессор переходит в режим эмуляции. Регистр очищается после сброса.

30.3.3 Регистры счетчика циклов (CCNTx)

Длина счетчика циклов равна 64 разряда, но доступен он как два универсальных 32-разрядных регистра – CCNT0 и CCNT1. После сброса значение счетчика равно нулю.

В связи с тем, что счетчик 64 бита, а доступ возможен только к частям регистра, необходимо соблюдение специальной процедуры чтения и записи регистра. При чтении сначала считывается нижняя часть (CCNT0), а затем верхняя часть (CCNT1). При чтении нижней части верхняя копируется в буфер и это гарантирует ее корректное значение. При записи сначала пишется нижняя часть (она попадает в буфер), а затем верхняя. При записи верхней полное 64-битное значение попадает в счетчик.

Счет прекращается, когда процессор переходит в режим эмуляции.

30.3.4 Регистр данных точки наблюдения 1 (WPDR)

Точка наблюдения 1 имеет дополнительные возможности по мониторингу обмена данными ядра процессора с памятью. Выше описывался механизм срабатывания исключительной ситуации в случае совпадения адресов точки наблюдения с адресами шин J и K. Однако имеется возможность дополнить сравнение адресов еще и сравнением данных. При чтении имеется возможность контролировать то, какие данные считывает процессор. Если дополнительно к совпадению адресов происходит совпадение и прочитанных данных – точка срабатывает. Также можно контролировать и процесс записи данных. Необходимые для контроля данные записываются в регистр WPDR. Необходимо отметить, что шина чтения или записи имеет разрядность 128 бит. Разрядность регистра данных только 32 бита. Поэтому сравнение выполняется только для 32-разрядного слова шины данных определяемого младшими битами шины адреса, независимо от того выполняется чтение или запись 64- или 128-разрядного слова.

30.3.5 Регистр маски точки наблюдения 1 (WPMR)

При использовании шины данных в описании точки наблюдения 1, не все биты шины данных могут понадобиться при анализе читаемых или записываемых данных. Регистр маски позволяет установить, какие биты регистра данных WPDR должны участвовать в сравнении. Если бит установлен в «1», соответствующий ему бит данных сравнивается с битом шины данных. После сброса регистр маски равен нулю и данные не участвуют в работе точки наблюдения.

30.4 Интерфейс JTAG

Процессор поддерживает порт доступа к средствам тестирования, соответствующий стандарту IEEE 1149.1.

30.4.1 Выводы порта JTAG

Таблица 337 описывает выводы процессора, используемые в аппаратной части эмуляции JTAG. Подчеркнутые снизу названия выводов и выводы с чертой над названием указывают на активные низкие сигналы. Для устойчивой работы интерфейса JTAG необходимо, чтобы частота TCK была как минимум в два раза ниже частоты SOCCLK (минимум в четыре раза меньше частоты CCLK).

Таблица 337 – Выводы порта связи и эмуляции устройства ввода/вывода

Сигнал	Тип	Описание
nTRESET	Вход	Сброс JTAG. Сбрасывает интерфейс JTAG. Сигнал nTRESET должен быть активирован после включения питания, чтобы убедиться в правильной работе JTAG. Вход nTRESET имеет внутренний резистор доопределения до питания
TCK	Вход	Синхронизация JTAG. Обеспечивает тактовый сигнал для работы JTAG. Имеет внутренний резистор доопределения до питания
TDI	Вход	Входной сигнал данных JTAG. Имеет внутренний резистор доопределения до питания
TDO	Выход	Выходной сигнал данных JTAG
TMS	Вход	Выбор режима тестирования JTAG. Управляет машиной состояний тестирования. Имеет внутренний резистор доопределения до питания
EMU	Выход	Эмуляция. Признак перехода процессора в режим эмуляции

В дополнение к стандартным функциям JTAG, выводы TMS и EMU выполняют следующие функции отладки:

– Вход TMS (когда разряд TEME установлен в регистре EMUCTL) выполняет функцию запроса к процессору на переход в режим эмуляции. Исключение эмуляции формируется при каждом фронте/срезе сигнала TMS. Данная функция является дополнительной к функциональности TMS в JTAG.

– Выходной сигнал EMU является сигналом с открытым стоком, который активизируется, только если разряд EMUOE установлен в регистре EMUCTL. Вывод EMU указывает на следующее:

– Произошло срабатывание точки наблюдения, но регистр управления имеет в поле типа (поле EXTYPE в WPiCTL) значение ноль. Уровень сигнала EMU в этом случае снижается на 20 циклов CCLK.

– Каждый раз, когда процессор находится в режиме эмуляции и готов к выполнению новой линии команд, подаваемой в регистр EMUIR, уровень сигнала на выводе EMU снижается до тех пор, пока такая команда добавляется.

30.4.2 Группа регистров эмулятора JTAG

Базовый адрес регистров 0x8000_03A0. Список доступных регистров приведен в таблице 338.

Таблица 338 – Группа регистров JTAG

Имя	Описание	Смещение	Значение по умолчанию
EMUCTL	Управление	0x00	0x0000 0000
EMUSTAT	Состояние	0x01	0x0000 0002
EMUDAT	Данные	0x02	Не определено
IDCODE	Код ID	0x03	0x427A F0CB (для микросхем ревизии 2)
			0x3008 C001 (для микросхем ревизии 3)
EMUIR:0	Команда 0	0x04	0x33C0 0000
EMUIR:1	Команда 1	0x05	0x33C0 0000
EMUIR:2	Команда 2	0x06	0x33C0 0000
EMUIR:3	Команда 3	0x07	0xB3C0 0000
-	-	0x08	-
OSPID	OSPID	0x09	Не определено
-	-	0x0A– 0x1F	-

30.4.2.1 *Регистр управления (EMUCTL)*

Регистр EMUCTL устанавливает параметры эмуляции. Значением сброса регистра EMUCTL является 0. Большинство разрядов регистра EMUCTL используются эмулятором и не должны модифицироваться программой. Единственными разрядами, предназначенными для использования программой, являются SWRST (программный сброс) и BOOTDIS (запрет загрузки). Подробное описание разрядов регистра приведено в таблице 339.

Таблица 339 – Регистр EMUCTL

Бит	Имя	Назначение
0	EMEN	Включение эмулятора: 1 – включен; 0 – выключен
1	TEME	Разрешение генерации запроса к процессору по входу TMS: 1 – разрешено; 0 – запрещено
2	EMUOE	Разрешение активной выдачи информации на выход TDO: 1 – разрешено; 0 – запрещено
3	-	Бит общего назначения
4	SWRST	Запрос программного сброса: 1 – сброс; 0 – рабочий режим
5	BOOTDSBL	Запрещение загрузки 1 – запрещено;

Бит	Имя	Назначение
		0 – разрешено Установка данного бита запрещает загрузку из внешнего EPROM.
31:6	-	Всегда ноль

30.4.2.2 *Регистр состояния эмуляции (EMUSTAT)*

Регистр EMUSTAT отражает состояние эмуляции. Значением сброса регистра EMUCTL является 0x0000 0002. Подробное описание разрядов регистра приведено в таблице 340.

Таблица 340 – Регистр EMUSTAT

Бит	Имя	Назначение
0	EMUMOD	1 – процессор в состоянии эмуляции; 0 – нормальный режим работы
1	IRFREE	Флаг готовности команды в регистре EMUIR: 1 – команда готова; 0 – нет
2	INRESET	Флаг сброса эмулятора
31:3	-	Всегда ноль

30.4.2.3 *Регистр типа кристалла и кода ID (IDCODE)*

Регистр IDCODE определяет ID модели процессора и типа кристалла. Регистр доступен только по чтению и имеет значение 0x427AF0CB.

30.4.3 Регистр команды JTAG

Регистр команды JTAG имеет длину, равную пяти разрядам, и позволяет контроллеру доступа порта (TAP) выбрать любой из сканируемых регистров данных. Младший значащий бит первым выдвигается на TDO. Кодирование команд приведено в таблице 341.

Таблица 341 – Декодирование команды доступа порта JTAG

Если значение IR4–0 равно...	Соответствующая команда...	Выполнить...
00000	EXTEST	Выполнить EXTEST
10000	Sample/Preload	Выполнить сэмплирование и предзагрузку
01000	EMUIR	Сканирование в регистре EMUIR
10100	EMUDAT	Сканирование в регистре EMUDAT
01110	IDCODE	Сканирование кода регистра ID
11110	SAMPLEPC	Сэмплирование ПК, к которому подключен процессор
10001	EMUTRAP	Выполнить перехват эмуляции
10011	OSPID	Сканирование регистра OSPID
00100	EMUCTL	Сканирование регистра EMUCTL

Если значение IR4–0 равно...	Соответствующая команда...	Выполнить...
01100	EMUSTAT	Сканирование регистра EMUSTAT
11111	BYPASS	Параллельный режим

30.4.4 Регистры данных

Регистры данных выбираются через регистр команд. Как только соответствующее значение регистра данных записывается в регистр команд, и состояние TAP изменяется на SHIFT-DR, данные выбранного регистра начинают выдвигаться на TDO и приниматься с TDI. Больше деталей в спецификации стандарта IEEE 1149.1.

Регистры данных:

– Обход (Байпас)

Регистр, отвечающий требованиям IEEE 1149.1, является одноразрядным регистром.

– Пограничные регистры

Регистры, используемые многими командами JTAG. Все три команды JTAG соответствуют требованиям IEEE 1149.1

– EMUIR

Команда JTAG загружает 48 разрядов, из которых 32 наименьших значащих разряда загружаются в регистр EMUIR. EMUIR является 128-разрядным регистром. В режиме эмуляции регистр загружается четыре раза последовательно (одной командой JTAG) или загружается до тех пор, пока в загружаемом слове не установится разряд EX (тогда остальные слоты EMUIR считаются командами NOP). Каждый раз, когда данные загружаются в EMUIR, сначала загружаются разряды 31–0, затем загружаются разряды 63–32 и так далее.

Когда четыре загрузки EMUIR завершены (или линия команд была завершена установленным разрядом EX) команды, загруженные в JTAG, передаются в устройство управления ядра и выполняются.

Когда счетверенное слово загружается в EMUIR, оно должно включать одну полную линию команд. Линия команд не должна продолжаться в следующем счетверенном слове. Лишние слоты должны быть заполнены NOP командами.

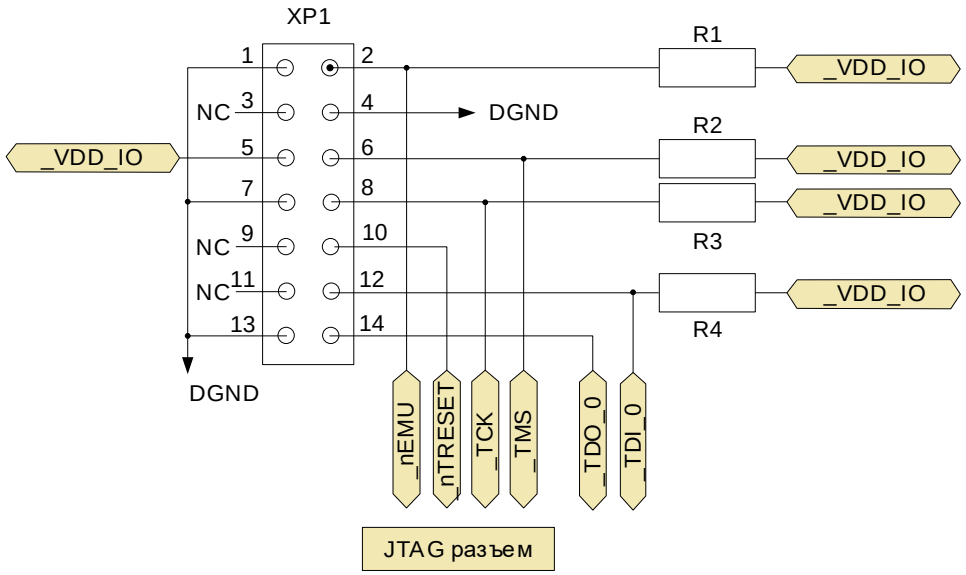
Запись в EMUIR является недопустимой операцией, когда процессор не находится в режиме эмуляции.

– EMUDAT

Это 48-разрядный регистр, в котором 32 старших значащих разряда используются для сканирования Ureg с таким же именем. Эмулятор использует этот регистр для чтения и записи всех остальных регистров в процессоре.

Эмулятор также использует данный регистр для доступа к памяти. Должна быть команда, использующая IALU для генерации адреса доступа к памяти, назначением или источником которой является EMUDAT. Когда эмулятор выполняет серию доступов к памяти, процессор должен вставить циклы удержания для того, чтобы завершить все доступы (как во время нормальной работы).

30.5 Схема подключения отладчика JEM-LYNX



R1 – R4 – резисторы сопротивлением 47 кОм ± 5 %

Рисунок 164 – Схема подключения отладчика JEM-LYNX

Назначение выводов разъема на рисунке 164 приведено в таблице 342.

Таблица 342 – Описание выводов разъема

Номер вывода	Обозначение	Назначение
1	GND	Общий
2	nEMU	Сигнал готовности процессора к приему отладочной инструкции
3	NC	Не используется
4	GND	Общий
5	VDDIO	Проверка наличия питания и согласование уровней сигналов
6	TMS	Изменение состояния TAP-контроллера
7	GND	Общий
8	TCK	Тактовая частота TAP-контроллера
9	nRESET	Синхронный аппаратный сброс процессоров
10	nTRESET	Сброс TAP-контроллера
11	JG_MX	Управление JG_MX
12	TDI	Запись команд и данных в TAP-контроллер
13	GND	Общий
14	TDO	Чтение данных из TAP-контроллера

31 Модуль управления синхронизацией и энергопотреблением

Процессор содержит специальный модуль (CMU) для задания различных режимов работы системы. Возможно индивидуальное задание частоты работы периферийных устройств, частоты процессора, частоты шин обмена. Возможна реализация режимов с пониженным энергопотреблением.

Данный модуль подключен к шине периферийных устройств и имеет базовый адрес **0x800001C0**.

Набор регистров управления/состояния представлен в таблице 343.

Таблица 343 – Регистры модуля CMU

Номер	Обозначение	Разрядность	Тип	Назначение
0	CFG1	32	R/W	Регистр конфигурации периферийных модулей
1	CFG2	16	R/W	Управление синтезатором частот CPU_PLL
2	CFG3	16	R/W	Управление синтезатором частот BUS_PLL
3	CFG4	2	R/W	Управление выбором частот процессора
4	CFG5	32	R/W	Управление синтезатором частот LINK_PLL
5	SYS_STS	16	R/W	Флаги состояния / Сброс флагов состояния
7-6	-			
10-8	CFG8	32	R/W	Управление синхронизацией периферийных устройств
11	CFG11	16	R/W	Коэффициент деления для синхросигнала интерфейчас CAN (для микросхем с ревизии 3)
12	CFG12	9	R/W	Регистр синхронизации портов связи (для микросхем с ревизии 3)
13	CFG13	1	R/W	Регистр тестового режима USB (для микросхем с ревизии 3)
14	CFG14	24	R/W	Регистр задержек приемника RGMII (для микросхем с ревизии 3)
15	CFG15	24	R/W	Регистр задержек передатчика RGMII (для микросхем с ревизии 3)

Процессор использует для своей работы внешний вход тактовой частоты OSCI. Данная частота используется блоком RTC в качестве рабочей частоты, а также в качестве опорной для внутренних умножителей частоты CPU_PLL, BUS_PLL и LINK_PLL. CPU_PLL формирует рабочую частоту для ядра процессора, которая далее делится, образуя частоту SOC-шины и периферийных устройств, а также частоту работы контроллеров. BUS_PLL формирует частоту работы внешнего интерфейса (SRAM, SDRAM). LINK_PLL формирует рабочую частоту для передатчиков портов связи (LINK-портов). Следует иметь в виду, что для достижения желаемой выходной частоты работы

LINK-портов FLINK следует запрограммировать LINK_PLL на выдачу частоты 2•FLINK. Алгоритм программирования всех PLL одинаков и более подробно описан в подразделе 31.2.1 «Регистры конфигурирования PLL CFG2, CFG3, CFG5».

Структура формирования частот процессора показана на рисунке 165. Значение коэффициента умножения для каждой PLL задается посредством программирования соответствующих регистров конфигурации. Периферийные устройства используют в качестве базовой частоты частоту SOCCLK. Далее внутри периферийного устройства с помощью регистров конфигурации может быть получена требуемая частота работы устройства.

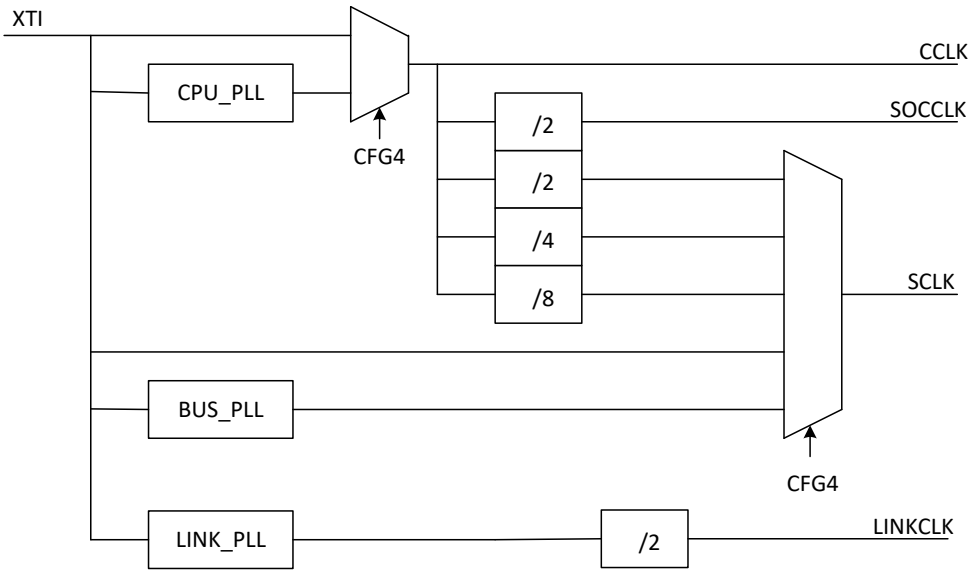


Рисунок 165 – Источники тактовой частоты

После сброса модули PLL отключены, и процессор начинает работу на частоте OSCI. Далее программа должна задать необходимые параметры частоты работы для PLL, выждать требуемое время необходимое для стабилизации выходной частоты и затем переключиться на частоту PLL.

Таким образом, процессор имеет несколько частотных доменов, т.е. определенных модулей, функционирующих на собственной частоте:

- процессорное ядро;
- шина периферийных устройств, работающая всегда на половине частоты ядра;
- интерфейс внешней памяти, работающий на собственной частоте;
- часы реального времени, работающие от внешней базовой частоты OSCI;
- порты связи, работающие на программируемой частоте.

31.1 Регистр конфигурации периферийных модулей CFG1

Регистр позволяет задавать различные параметры работы периферийных модулей процессора.

Таблица 344 – Регистр CFG1

Биты	Обозначение	Значение по сбросу	Назначение
1, 0	-	0	
2	LINK_BW	0	Ширина шины линк-портов: 0 – ширина шины задается битами регистра управления порта связи; 1 – ширина шины порта связи равна четырем битам
3	SYS_WE	0	Разрешение многократной записи регистров SYSCON, SDRCON: 0 – однократная запись; 1 – многократная запись
4	ARINC_T0_EN	0	Подключение передатчика 1 контроллера ARINC к внешним контактам: 1 – подключено
5	ARINC_T1_EN	0	Тоже для передатчика 2
6	ARINC_T2_EN	0	Тоже для передатчика 3
7	ARINC_T3_EN	0	Тоже для передатчика 4
8	MIL_DIS	0	Отключение передатчика МКПД от внешних контактов: 1 – отключен
11:9	-	0	
12	DMA_HP_EN	0	Разрешение высокоприоритетному каналу DMA иметь преимущество перед процессором при доступе к внешней памяти: 1 – разрешено; 0 – процессор всегда имеет высший приоритет
13	H_OFF	0	Запрещение работы хост-интерфейса: 0 – хост-интерфейс разрешен; 1 – хост интерфейс запрещен
14	SPI1_EN	0	Разрешение работы интерфейса SPI1: 0 – интерфейс запрещен; 1 – интерфейс разрешен
15	SPI2_EN	0	Разрешение работы интерфейса SPI2: 0 – интерфейс запрещен; 1 – интерфейс разрешен
16	I2C_ALT	0	Коммутация интерфейса I2C: 0 – SCL = PC[12], SDA = PC[11]; 1 – интерфейс I2C отключен
18. 17	NAND_ALT	00	Альтернативное подключение контроллера NAND Flash на порты общего назначения PA: 00 – NAND не подключен; 01 – запрещенная комбинация; 10 – подключен, но пользоваться нельзя. Интерфейсы SPI могут работать. NAND удерживает неактивными сигналы выборки и чтения-записи;

Биты	Обозначение	Значение по сбросу	Назначение
			11 – подключен и может использоваться
19	GTMR0_EN	0	Разрешение работы таймера 0 с функцией Захвата/ШИМ: 0 – таймер запрещен; 1 – таймер разрешен
20	GTMR1_EN	0	Разрешение работы таймера 1 с функцией Захвата/ШИМ: 0 – таймер запрещен; 1 – таймер разрешен
21	ETH_EN	0	Разрешение подключения выводов контроллера Ethernet (при включенной альтернативной функции GPIO и GRMR0_EN=0) (для микросхем с ревизии 3): 0 – контроллер Ethetnet отключен; 1 – контроллер Ethernet подключен
22	USB_EN	0	Разрешение подключения выводов контроллера USB (при включенной альтернативной функции GPIO) (для микросхем с ревизии 3): 0 – контроллер USB отключен; 1 – контроллер USB подключен
23	UART2_EN	0	Разрешение подключения выводов контроллера UART2 (при включенной альтернативной функции GPIO) (для микросхем с ревизии 3): 0 – контроллер UART2 отключен; 1 – контроллер UART2 подключен
24	UART3_EN	0	Разрешение подключения выводов контроллера UART3 (при включенной альтернативной функции GPIO) (для микросхем с ревизии 3): 0 – контроллер UART3 отключен; 1 – контроллер UART3 подключен
25	CAN0_EN	0	Разрешение подключения выводов контроллера CAN0 (при включенной альтернативной функции GPIO) (для микросхем с ревизии 3): 0 – контроллер CAN0 отключен; 1 – контроллер CAN0 подключен
26	CAN1_EN	0	Разрешение подключения выводов контроллера CAN1 (при включенной альтернативной функции GPIO) (для микросхем с ревизии 3): 0 – контроллер CAN1 отключен; 1 – контроллер CAN1 подключен
27	RMII_EN	0	Выбор интерфейса Ethernet PHY (для микросхем с ревизии 3): 0 – RGMII; 1 – RMII

31.2 Регистры управления внутренними PLL

Микросхема имеет синтезатор тактовой частоты CPU_PLL для формирования синхросигналов процессорного ядра, синтезатор тактовой частоты BUS_PLL для формирования синхросигнала контроллера внешней памяти, синтезатор тактовой частоты LINK_PLL для формирования синхросигнала портов связи. Для микросхем с ревизии 3 микросхема также имеет синтезатор тактовой частоты ETH_PLL для формирования синхросигнала блока Ethernet. Генераторы используют базовую частоту синхронизации OSCI.

31.2.1 Регистры конфигурирования PLL CFG2, CFG3, CFG5, CFG6

Регистры доступны по чтению и записи, позволяют задавать различные параметры работы синтезаторов частоты CPU_PLL, BUS_PLL, LINK_PLL, ETH_PLL. Запись в регистры разрешена только при нулевом значении бит CPPL_SEL, BPLL_SEL и LPLL_SEL, соответственно (регистр CFG4).

Таблица 345 – Регистр CFG2/ CFG3/ CFG5/ CFG6*

Биты	Обозначение	Значение по сбросу	Назначение
3:0	DIVR	0	Вход, задающий коэффициент деления делителя опорной частоты: 0000 – коэффициент деления 1; 0001 – коэффициент деления 2; 1111 – коэффициент деления 16
10:4	DIVF	0	Вход, задающий коэффициент деления делителя обратной связи: 0000000 – коэффициент деления 1; 0000001 – коэффициент деления 3; 0000010 – коэффициент деления 4; 1111111 – коэффициент деления 129
13:11	DIVQ	0	Вход, задающий коэффициент деления выходного делителя: 000 – коэффициент деления 1; 001 – коэффициент деления 2; 010 – коэффициент деления 4; 011 – коэффициент деления 8; 100 – коэффициент деления 16; 101 – коэффициент деления 32
16:14	RANGE	0	Вход подстройки фильтра в зависимости от частоты ЧФД: 001 – 10-16 МГц; 010 – 16-25 МГц; 011 – 25-40 МГц; 100 – 40-65 МГц; 101 – 65-100 МГц
19:17	-	0	Зарезервировано
20	bypass	0	Вход включения режима «bypass». При значении «1» на выход PLLOUT подается сигнал со входа REF

Биты	Обозначение	Значение по сбросу	Назначение
22, 21	S	0	Вход управления выходным мультиплексором, передающим сигнал на выход MUXOUT: 00 – подается сигнал с выхода PLL; 01 – подается сигнал с выхода делителя опорной частоты; 10 – подается сигнал с выхода делителя в обратной связи
* Доступен в микросхемах с ревизии 3			

Структурная схема PLL приведена на рисунке 166. Внешний вход частоты OSCI подключен ко входу REF.

Значения коэффициентов подбираются таким образом, чтобы частота f_{VCO} находилась в диапазоне 800 – 1600 МГц, а частота f_{IN} была равна частоте f_{FB} и находилась в диапазоне, определяемом значением RANGE.

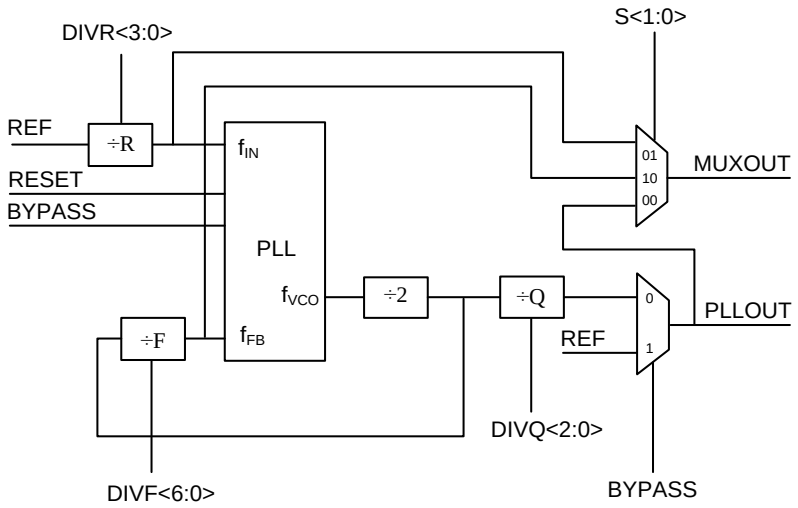


Рисунок 166 – Структура PLL

Процедура работы с CPU_PLL может быть следующей:

1 После сброса системы, синтезатор CPU_PLL выключен, и процессор работает на входной частоте OSCI.

2 Посредством регистра CFG2 пользователь производит установку рабочих параметров PLL.

3 После задержки, необходимой для стабилизации частоты PLL, пользователь устанавливает соответствующий бит в регистре CFG4 и переключается на частоту PLL.

4 При необходимости изменить параметры рабочей частоты, пользователь выполняет переключение на частоту OSCI. Далее выполняются шаги 2 и 3.

Аналогичные действия должны выполняться и при изменениях частоты внешней шины.

31.2.2 Регистр выбора источников синхронизации CFG4

Регистр позволяет задавать источник частоты различных модулей процессора.

Таблица 346 – Регистр CFG4

Биты	Обозначение	По сбросу	Назначение
0	CPLL_SEL	0	Задание частоты ядра процессора: 0 – источник частоты ядра – вход OSCI. Разрешение записи в регистр CFG2; 1 – источник частоты ядра – выход PLLOUT CPU_PLL. Запрещение записи в регистр CFG2
1	BPLL_SEL	0	Задание частоты интерфейса внешней шины: 0 – источник частоты внешней шины – OSCI. Разрешение записи в регистр CFG3; 1 – источник частоты внешней шины – сигнал BCLK (см. биты CFG4[5:4]). Запрещение записи в регистр CFG3
2	DIS_CC	0	1 – останов синхронизации ядра
3	DIS_BC	0	1 – останов синхронизации внешнего интерфейса
5:4	BCLK_SEL	00	Выбор источника синхронизации BCLK внешнего интерфейса: 00 – частота PLLOUT BUS_PLL; 01 – частота PLLOUT /2 CPU_PLL; 10 – частота PLLOUT /4 CPU_PLL; 11 – частота PLLOUT /8 CPU_PLL
6	-	0	
7	LPLL_SEL	0	Задание частоты портов связи: 0 – разрешение записи в регистр CFG5; 1 – запрещение записи в регистр CFG5

Переключение процессора с одной частоты на другую требует выполнения определенных действий. Подробно процесс переключения излагается в подразделе 31.10 «Переключение частоты процессора».

Также в процессоре предусмотрена возможность останова частот синхронизации с целью обеспечения реализации режима пониженного энергопотребления. Подробно процедура реализации данного механизма описана в подразделе 31.11 «Реализация «спящего режима»».

31.3 Регистр состояния системы SYS_STS

Регистр доступен только по чтению. Описание битов регистра состояния SYS_STS приведено в таблице 347.

Таблица 347 – Регистр SYS_STS

Бит	Обозначение	Назначение
0	BOOT_0	Состояние входов BOOT[2:0] (они же PC[6:4]) в момент завершения сигнала внешнего сброса или внутреннего сброса по включению питания. Хранят вариант начального старта процессора
1	BOOT_1	
2	BOOT_2	
3	-	

Бит	Обозначение	Назначение
4	L0BCMPO	Состояние входа PC[26] в момент завершения сигнала внешнего сброса или внутреннего сброса по включению питания.
5	L1BCMPO	Состояние входа PC[30] в момент завершения сигнала внешнего сброса или внутреннего сброса по включению питания. Вход используется для задания режима работы контактной площадки внешнего осциллятора
6	RST	Флаг устанавливается по сигналу внешнего сброса, сброса по включению питания или сброса от сторожевого таймера. Флаг может быть сброшен записью любого значения в регистр SYS_STS
7	POR	Флаг устанавливается по сигналу внешнего сброса или сброса по включению питания. Флаг может быть сброшен записью любого значения в регистр SYS_STS
31:8	-	Всегда 0

После сброса регистр состояния позволяет определить, был ли сброс вызван внешним источником или включением питания (RST==1, POR==1) или сброс был вызван срабатыванием сторожевого таймера (RST==1, POR==0). В регистре состояния отражается вариант начального старта, который используется внутренним ПЗУ для определения алгоритма загрузки. Если бит L1BCMPO равен единице, в качестве опорной частоты используется внешний вход OSCI. Если значение бита L1BCMPO равно нулю, в качестве опорной частоты используется встроенный внутренний осциллятор.

31.4 Регистр управления синхронизацией периферийных устройств CFG8

Регистр может использоваться для разрешения или останова синхронизации различных периферийных устройств. Нулевое значение бита разрешает синхронизацию соответствующего периферийного устройства. Единичное значение останавливает синхронизацию.

Таблица 348 – Регистр CFG8

Биты	Обозначение	Значение по сбросу	Разрешение синхронизации для модуля
0	I2S0_en	0	Интерфейс I2S 0
1	I2S1_en	0	Интерфейс I2S 1
2	VCAM_en	0	VCAM
3	SPI_en	0	Интерфейс SPI
4	NAND_en	0	Контроллер NAND
5	ARINC_en	0	ARINC
6	MIL0_en	0	МКПД 0
7	MIL1_en	0	МКПД 1
8	ADDA0_en	0	UP\DOWN 0
9	ADDA1_en	0	UP\DOWN 1
10	ADDA2_en	0	UP\DOWN 2
11	ADDA3_en	0	UP\DOWN 3

Биты	Обозначение	Значение по сбросу	Разрешение синхронизации для модуля
12	GPS0_en	0	Цифровой коррелятор
13	GPS1_en	0	Цифровой коррелятор
14	LCD_en	0	Контроллер LCD
15	UART0_en	0	UART0
16	UART1_en	0	UART1
17	SPI1_en	0	Интерфейс SPI1
18	SPI2_en	0	Интерфейс SPI2
19	GTMR0_en	0	Таймер 0 с функцией Захвата/ШИМ
20	GTMR1_en	0	Таймер 1 с функцией Захвата/ШИМ
21	I2C_en	0	Интерфейс I2C
22	UART2_en	0	UART2 (для микросхем с ревизии 3)
23	UART3_en	0	UART3 (для микросхем с ревизии 3)
24	CAN0_en	0	CAN0 (для микросхем с ревизии 3)
25	CAN1_en	0	CAN1 (для микросхем с ревизии 3)
26	ETH_en	0	Интерфейс Ethernet (для микросхем с ревизии 3)
27	USB_en	0	Интерфейс USB (для микросхем с ревизии 3)
31:28		0	Зарезервировано

Регистр может использоваться для задания режима пониженного энергопотребления системы. В случае если устройство в данный момент не используется, можно блокировать его частоту синхронизации. После блокировки, доступ к устройству невозможен. После сброса значение регистра равно нулю и на все периферийные устройства поступает сигнал синхронизации.

Примечание – Тактирование таймеров с функцией Захвата/ШИМ осуществляется, когда в регистре CFG8 установлено разрешение синхронизации сразу двух таймеров: GTMR0_en и GTMR1_en. См. описание ошибки 0007 в документе «Errata Notice».

31.5 Регистр формирования частоты интерфейса CAN CFG11 (для микросхем с ревизии 3)

Регистр используется для задания коэффициента деления частоты SOC-шины для формирования частоты работы интерфейса CAN.

Таблица 349 – Регистр CFG11

Биты	Обозначение	Значение по сбросу	Назначение
15:0	CAN_DV	0	Коэффициент деления частоты SOC-шины
31:16		0	Зарезервировано

31.6 Регистр синхронизации приемников портов связи CFG12 (для микросхем с ревизии 3)

Регистр используется для управления механизмом синхронизации приемников портов связи.

Таблица 350 – Регистр CFG12

Биты	Обозначение	Значение по сбросу	Разрешение синхронизации для модуля
0	INT_RX_SYNC	0	Строб синхронизации приема приемников портов связи
1	INT_RX_SYNC_EN	0	Разрешение синхронизации по биту INT_RX_SYNC (вместо L0BCMPI): 0 – для синхронизации используется вывод L0BCMPI; 1 – для синхронизации используется бит INT_RX_SYNC
2	EXT_PULSE	0	Строб синхронизации внешних микросхем
3	CLK_SEL	0	Выбор источника синхронизации строба внешних микросхем (бита EXT_PULSE): 0 – выход приемника синхросигнала порта связи 0; 1 – выход приемника синхросигнала порта связи 1
4	EXT_PULSE_EN	0	Разрешение выдачи строба синхронизации через площадку ввода-вывода OSCO: 0 – выдача сигнала EXT_PULSE запрещена; 1 – выдача сигнала EXT_PULSE разрешена
5	EXT_PULSE_SEL	0	Выбор источника строба синхронизации внешних микросхем: 0 – бит CFG12[2] EXT_PULSE; 1 – вход L1BCMPI (PC[31])
9:6	DEL_SEL	0	Выбор значения задержки строба синхронизации
31:10	-		Зарезервировано

Бит INT_RX_SYNC выполняет роль импульса синхронизации – альтернатива сигналу с входа L0BCMPI. Выбор того или иного источника синхронизации осуществляется битом INT_RX_SYNC_EN. Подробнее об алгоритме синхронизации по событию см. в подразделе 15.19 «Режим работы с синхронизацией по внешнему или внутреннему событию (для микросхем с ревизии 3)».

Бит EXT_PULSE может служить стробом синхронизации для микросхем всей системы в составе одного модуля обработки данных. Наряду с этим битом источником синхронизации может быть сигнал, поступающий на вход L1BCMPI. Выбор того или иного источника определяется битом CFG12[5] EXT_PULSE_SEL. Результирующий сигнал пересинхронизируется на один из синхросигналов, определяемых битом CFG12[3] CLK_SEL и выходит их микросхемы через вывод OSCO. На плате этот сигнал заводится на входы L0BCMPI всех процессоров К1967ВН044, входящих в состав системы. Для управления площадкой ввода-вывода служит бит EXT_PULSE_EN. При использовании описанной функции генерации внешнего импульса вначале нужно установить бит EXT_PULSE_EN = 1, а затем производить запись в регистр EXT_PULSE либо ожидать появления импульса на L1BCMPI.

Сигнал EXT_PULSE при выходе из микросхемы может иметь фазу, которая не будет гарантировать синхронный захват этого сигнала внутри остальных процессоров.

Для подбора нужной фазы перед выходом из микросхемы сигнал EXT_PULSE будет проходить через линию задержки, величина которой определяется полем DEL_SEL. Задержка линейная, код управления прямой двоичный. Величина единичной задержки порядка 100 пс.

Все сказанное выше о функции генерации внешнего импульса поясняется на рисунке 167.

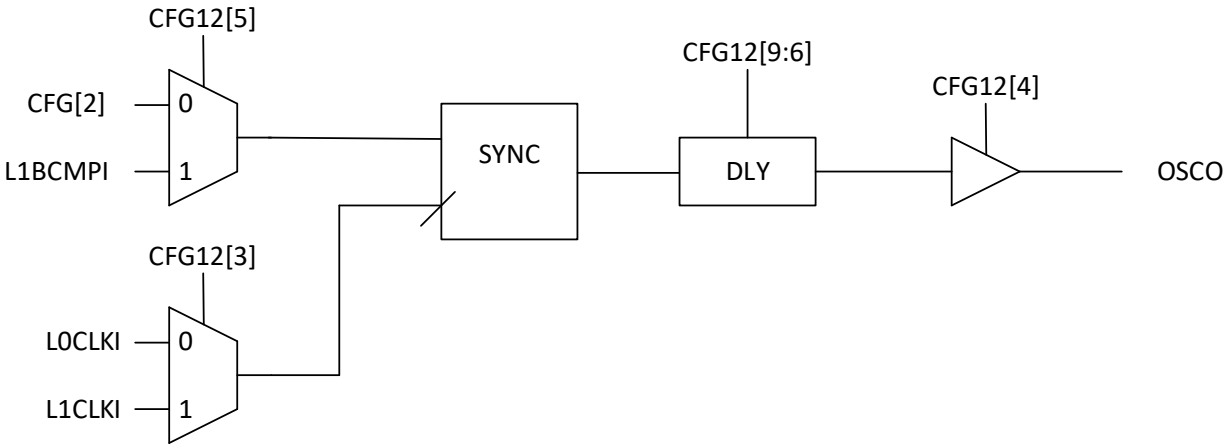


Рисунок 167 – Схема формирования внешнего импульса синхронизации

31.7 Регистр тестового режима USB CFG13 (для микросхем с ревизии 3)

Таблица 351 – Регистр CFG13

Биты	Обозначение	Значение по сбросу	Назначение
0	USB_TEST_SPEEDUP	0	Бит управления начальным значением счетчиков контроллера USB. Применяется только на этапе разработки микросхемы для уменьшения времени моделирования: 0 – счетчики при сбросе загружаются рабочими значениями; 1 – счетчики при сбросе загружаются уменьшенными значениями
31:1		0	Зарезервировано

31.8 Регистр задержек приемника RMII/RGMII CFG14 (для микросхем с ревизии 3)

Таблица 352 – Регистр CFG14

Биты	Обозначение	Значение по сбросу	Назначение
3:0	RGRXC_DEL_SEL	0	Выбор значения задержки сигнала RGRXC
7:4	RGRXCTL_DEL_SEL	0	Выбор значения задержки сигнала RGRXCTL
11:8	RGRXD0_DEL_SEL	0	Выбор значения задержки сигнала RGRXD[0]
15:12	RGRXD1_DEL_SEL	0	Выбор значения задержки сигнала RGRXD[1]
19:16	RGRXD2_DEL_SEL	0	Выбор значения задержки сигнала RGRXD[2]

Биты	Обозначение	Значение по сбросу	Назначение
23:20	RGRXD3_DEL_SEL	0	Выбор значения задержки сигнала RGRXD[3]
31:24	-	0	Зарезервировано

Для интерфейса RGMII в микросхеме реализованы внутренние линии задержки для коррекции разбежки сигналов при разводке платы. Величина единичной задержки – порядка 100 пс.

Итоговая задержка = 300 + RGRX*_DEL_SEL • 100 (пс).

31.9 Регистр задержек передатчика RMII/RGMII CFG15 (для микросхем с ревизии 3)

Таблица 353 – Регистр CFG15

Биты	Обозначение	Значение по сбросу	Назначение
3:0	RGTXC_DEL_SEL	0	Выбор значения задержки сигнала RGTXC
7:4	RGTXCTL_DEL_SEL	0	Выбор значения задержки сигнала RGTXCTL
11:8	RGTXD0_DEL_SEL	0	Выбор значения задержки сигнала RGTXD[0]
15:12	RGTXD1_DEL_SEL	0	Выбор значения задержки сигнала RGTXD[1]
19:16	RGTXD2_DEL_SEL	0	Выбор значения задержки сигнала RGTXD[2]
23:20	RGTXD3_DEL_SEL	0	Выбор значения задержки сигнала RGTXD[3]
31:24	-	0	Зарезервировано

Для интерфейса RGMII в микросхеме реализованы внутренние линии задержки для коррекции разбежки сигналов при разводке платы. Величина единичной задержки – порядка 100 пс.

Итоговая задержка = 300+ RGTX*_DEL_DEL • 100 (пс).

31.10 Переключение частоты процессора

В процессоре предусмотрена возможность гибкого выбора необходимой частоты функционирования. После сброса процессор всегда работает от входного источника синхронизации OSCI. С помощью имеющихся модулей PLL можно задать требуемую для выполнения задачи частоту процессорного ядра и переключиться на заданную частоту.

Переключение частот выполняется с помощью регистра конфигурации CFG4. Биты регистра номер 7, 1 и 0 выполняют выбор источника синхронизации для портов связи, внешней шины и ядра соответственно. Переключение частот для портов связи и внешнего порта выполняется простой установкой или сбросом соответствующих бит регистра. Главное требование при смене данных частот это обеспечение отсутствия в момент переключения каких-либо действий в передатчиках порта связи или во внешнем интерфейсе.

Перечень необходимых подготовительных действий для смены, например, частоты внешнего интерфейса зависит от конфигурации внешней шины. Если, например,

подключена внешняя динамическая память, нужно не забывать о периоде регенерации динамической памяти. Если смена частоты происходит после сброса процессора и динамическая память еще не инициализирована или нет необходимости обеспечивать сохранность информации в данной памяти, то переключение частоты можно выполнять без подготовительных работ. Если же процессор уже использовал динамическую память и в ней хранится информация, смена частот (например, с высокой на среднюю) будет требовать следующих действий:

– перевод динамической памяти в режим саморегенерации;

– переключение частоты внешней шины на входную частоту OSCI;

– подготовка новой частоты в модуле BUS_PLL;

– корректировка параметров периода регенерации динамической памяти и других временных параметров SDRAM;

– переключение на частоту BUS_PLL с помощью бита CFG4[1];

– выход SDRAM из режима саморегенерации.

Возможны и другие действия в зависимости от подключенных внешних периферийных устройств и их требований в обслуживании.

Процесс переключения частоты ядра также требует выполнения целого комплекса работ в зависимости от ситуации, в которой требуется переключение частот. Рассмотрим самый простой случай, когда процессор переключается на заданную частоту сразу после сброса. В этом случае должны быть выполнены следующие действия:

– с помощью регистра CFG2 задается требуемая конфигурация CPU_PLL для получения необходимой частоты. Выдерживается необходимое время для стабилизации выхода PLL.

– процессор переводится в режим останова синхронизации с помощью установки бита CFG4[2].

– сразу же за установкой бита CFG4[2] выполняется установка бита CFG4[0]. Процессор переключает частоту в момент, когда частоты ядра, периферийных устройств и внешней шины блокированы.

– через заданный интервал времени частота процессорного ядра разрешается и процессор продолжает выполнение программы с новой частотой.

При переключении частоты ядра используется функция останова синхронизации («спящий режим»). Подробно реализация данного режима и его особенностей приведена в подразделе 31.11 «Реализация «спящего режима»».

Возможность переключения частоты процессора позволяет гибко выбирать необходимую производительность процессора в зависимости от решаемой задачи и таким образом сокращать потребляемую мощность системы.

31.11 Реализация «спящего режима»

В процессоре возможна реализация режима, в котором частоты ядра, периферийной шины и внешнего интерфейса могут быть остановлены. Данная функция введена для обеспечения режима наименьшего потребления в случае ожидания внешнего или внутреннего события. Ниже показан формирователь частот с учетом функции

спящего режима (рисунок 168). Специальные gate-модули G0, G1, G2 выполняют блокировку выбранных частот.

Установка бита CFG4[3] приведет к останову синхронизации на внешней контактной площадке SCLK и к отсутствию синхронизации внешнего интерфейса. Данная функция может быть использована, когда выводы внешнего интерфейса выполняют GPIO функции, и нет необходимости выполнять адресуемые обмены по внешней шине. Возобновление синхронизации внешнего интерфейса выполняется программно установкой данного бита в ноль.

Останов синхронизации ядра реализован немного сложнее. При установке бита CFG4[2] в единицу произойдет блокировка прохождения частоты на ядро, шину периферийных устройств и на внешний интерфейс (хотя на внешнем контакте SCLK частота будет присутствовать). После останова синхронизации ядра никакие программы уже невозможно выполнять и выход из данного режима можно выполнить только аппаратно. При этом источники, формирующие сигнал пробуждения wake_up, должны использовать другие источники синхронизации.

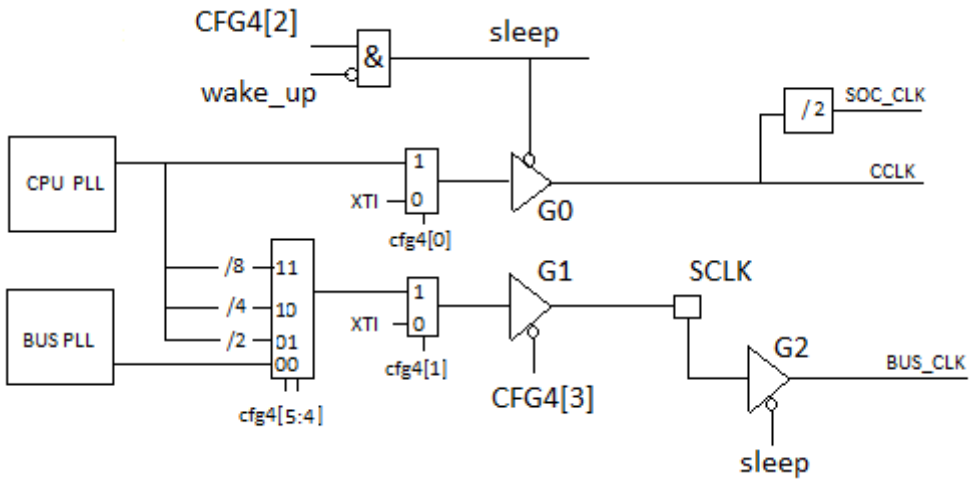


Рисунок 168 – Схема блокировки синхронизации

В качестве событий, которые могут вывести процессор из спящего режима, могут использоваться:

– Запрос прерывания по любому из внешних входов портов общего назначения PA, PB, PC. Прерывания должны быть запрограммированы с выбором функции «по уровню».

– Запрос от события «тик-импульс» часов реального времени.

– Запрос от события «прерывание от сторожевого таймера» часов реального времени.

– Запрос от события «будильник» часов реального времени.

Модуль часов реального времени использует для своей работы вход внешней частоты OSCI. В связи с этим при останове синхронизации ядра, модуль RTC продолжает работу и позволяет формировать требуемые сигналы.

При переходе в спящий режим можно запрограммировать выход из состояния сна с помощью необходимого пользователю события.

События wake_up, выводящие процессор из состояния сна, можно охарактеризовать двумя параметрами: периодом следования и длительностью активного уровня. Если выбран внешний источник порта общего назначения, период следования и длина импульса неизвестны. Если выбран один из источников модуля RTC, параметры данных импульсов известны.

После сброса тик-импульсы формируются с периодом 256 тактов внешней частоты и имеют длительность периода внешней частоты. Период следования можно поменять при помощи значения регистра TIC_VAL.

Счетчик сторожевого таймера после сброса имеет значение 15, поэтому сторожевой таймер после сброса сформирует первый свой импульс wake_up через 15 периодов тик-импульса. Дальше период следования импульсов пробуждения от сторожевого таймера будет равен 256 периодам тик-импульса (если сторожевой таймер не обслуживается).

Будильник часов реального времени будет формировать свой wake_up запрос каждый раз, когда значение счетчика секунд RTC_CNT будет совпадать со значением регистра RTC_MR. Длительность импульса будет равна одной псевдосекунде.

Длительность импульса wake_up имеет значение, т.к. после пробуждения процессор не сможет опять вернуться в режим сна, если импульс пробуждения не завершился.

Запросы на выход из режима сна от портов общего назначения могут быть блокированы путем записи в регистр маскирования прерываний значения ноль. Запросы от источников модуля RTC могут быть блокированы с помощью разрядов 10, 9, 8 регистра управления RTC_CR.

Из рисунка видно, что процессор может выйти из режима сна, используя запрограммированную частоту, т.е. может возобновить свою работу на частоте OSCI или частоте CPU_PLL. Дополнительное выключение модулей PLL для обеспечения пониженного энергопотребления также может быть использовано. Источник внешней синхронизации OSCI не имеет функции выключения.

Каждый раз при переходе в режим сна должны обеспечиваться сохранность информации во внешней динамической памяти (если она используется), отсутствие работающих периферийных устройств (их синхронизация будет остановлена), а также должен быть предусмотрен способ выхода из режима сна.

32 Начальный старт процессора

Разряды порта PC[6:4] (они же BOOT[2:0]) во время сброса процессора задают режим начального старта процессора. Процессор может выполнить один из 8-ми вариантов начального старта. В любом из выбранных вариантов процессор начинает работу с выполнения кода программы, размещенного во внутреннем загрузочном ПЗУ. Описание режимов загрузки приведено в таблице 354.

Таблица 354 – Режимы загрузки

BOOT[2:0]	Режим загрузки
000	Старт загрузки по адресам, выбираемым одним из входов nIRQ. Срабатывание по уровню. 16-разрядная внешняя шина. nIRQ[0] –> 0x30000000 (MS0); nIRQ[1] –> 0x38000000 (MS1); nIRQ[3] –> 0x00000000 (internal memory)
001	Старт загрузки по адресам, выбираемым одним из входов nIRQ. Срабатывание по уровню. 32-разрядная внешняя шина. nIRQ[0] –> 0x30000000 (MS0); nIRQ[1] –> 0x38000000 (MS1); nIRQ[3] –> 0x00000000 (internal memory)
010	Старт из параллельной восьмиразрядной FLASH-памяти
011	Старт из последовательной SPI FLASH-памяти (выборка по CS0)
100	Загрузка из NAND FLASH-памяти (выводы PD)
101	Загрузка через LINK-порты (любой) в однобитовом режиме
110	Загрузка из NAND FLASH-памяти (альтернативные выводы PA)
111	Зарезервировано. Выполнение служебной программы

Процессор считывает код старта из регистра состояния и выполняет переход на одну из меток, соответствующую номеру начального старта. Далее будут подробно рассмотрены действия, которые выполняются при выборе того или иного варианта старта. Файл def1967VN044.h с параметрами, используемыми в программе начального старта, является частью поставки среды разработки программ CM-LYNX.

```
#include "def1967VN044.h"

.section program;
.global _boot_start;

_boot_start:
        /*
                Default clock configuration:
                        osci = input
                        core clk (cclk) = osci
                        SOC bus clock (sclk) = osci/2
                        External bus (memory) clock = osci
                NAND operating frequency is configured to the slowest by hardware after reset
```

```
            DSP starts in SV mode, GIE bit is set, all interrupts are disabled (masks = 0)
        */

        // Enable multiple writes to SYSCON/SDRCON
        j9 = base_CMU;;
        j0 = 8;;
        cjmp = 0x00;;

        [j9 + 0] = j0;; // CMU_CFG1_LOC

// Define boot mode
        j1 = [j9 + 5];;// CMU_SYS_STS_LOC
        j1 = j1 and 7;;

        comp(j1, 7);;
        if jeq, jump start_XROM(np);;
        comp(j1, 6);;
        if jeq, jump start_xNAND(np);;
        comp(j1, 5);;
        if jeq, jump start_LINK1(np);;
        comp(j1, 4);;
        if jeq, jump start_NAND(np);;
        comp(j1, 3);;
        if jeq, jump start_SPIF(np);;
        comp(j1, 2);;
        if jeq, jump start_XPF(np);;
        comp(j1, 1);;
        if jeq, jump start_IRQ(np);          j10 = 0x00;;   // 32-bit mode
        j10 = SYSCON_MEM_WID64;;                            // 16-bit mode


        //-------------------------------------------------------//
        // Passive nIRQ boot
        //
        //-------------------------------------------------------//
start_IRQ:
        // Initialize SYSCON register
        j0 = j10 or SYSCON_MS0_WT3                  |
                    SYSCON_MS0_SLOW                 |
                    SYSCON_MS0_IDLE                 |

                    SYSCON_MS1_WT3                  |
                    SYSCON_MS1_SLOW                 |
                    SYSCON_MS1_IDLE;;
        syscon = j0;;

        // Setup GPIO
```

```
        // Alternate function for nBMS, nRD, MS1 and MS0 signals
        j0 = (PF_ALT << 20) | (PF_ALT << 21) | (PF_ALT << 18) | (PF_ALT << 19);;
        [j31 + GPC_ALT_LOC] = j0;;

        // Alternate function for DATA and ADDR bus
        j0 = PX_ALT_PDB0 | PX_ALT_PDB1;;                    // data [15:0]
        comp(j10, 0x00);;
        if jeq; do, j0 = j0 or PX_ALT_PDB2 | PX_ALT_PDB3;;  // data [31:16]
        j0 = j0 or PX_ALT_PAB0 | PX_ALT_PAB1 | PX_ALT_PAB2;; // addr
        [j31 + PX_ALT_LOC] = j0;;

        // Setup vectors
        j8 = 0x30000000;;
        IVIRQ0 = j8;;
        j8 = 0x38000000;;
        IVIRQ1 = j8;;
        j8 = 0x00000000;;
        IVIRQ3 = j8;;

        j8 = 0x0F;;
        intctl = j8;;    // nIRQ3:0 by level

        // Enable interrupts by external IRQs and vector interrupt (for host)
        j8 = INT_IRQ0 | INT_IRQ1 /*| INT_IRQ2*/ | INT_IRQ3 | INT_VIRPT;;
        IMASKH = j8;;

        // SQCTL[GIE] bit is set by reset, so we're done!
stay_idle:
        nop;; nop;; nop;; nop;;
        idle;; nop;; nop;; nop;;
        jump stay_idle(np);;


        //-------------------------------------------------------//
        // Serial SPI FLASH at #CS0
        //
        //-------------------------------------------------------//
start_SPIF:

        // Setup SPI
        j0 = (199 << 16) | SPCR0_DSS_8BIT;;    // 8-bit mode, SPI clock = (SCLK_in / 2) / 200
        [j31 + SPCR0_LOC] = j0;;
        j0 = (1<<SPCR1_HOLDCS_P) | (1<<SPCR1_SPE_P);;  // Enable SPI, hold #CS
        [j31 + SPCR1_LOC] = j0;;

        // Setup GPIO
        j0 = (PF_ALT << 4) | (PF_ALT << 5) | (PF_ALT << 6) | (PF_ALT << 7);;
```

```
        [j31 + GPA_ALT_LOC] = j0;;

        k0 = SPDR_LOC;;
        k1 = SPSR_LOC;;
        xr9 = 0x0808;;              // Initial FDEP arg
        xr10 = 0x0800;;             // FDEP arg modifier
        j10 = 0x00000000;;          // target address
        j11 = 0x00;;
        lc0 = 256;;

        // Issue READ command
        xr0 = 0x03;;
        [k0 + 0] = xr0;         xr1 = 0x00;;
        [k0 + 0] = xr1;;        // A[23:16]
        [k0 + 0] = xr1;;        // A[15:8]
        [k0 + 0] = xr1;;        // A[7:0]

loop_SPIF:
        // Write 4 bytes
        [k0 + 0] = xr1;;
        [k0 + 0] = xr1;;
        [k0 + 0] = xr1;;
        [k0 + 0] = xr1;;

        // Wait 4 bytes to be read
wait_SPIF:
        j0 = [k1 + 0];;
        j0 = j0 and (1<<SPSR_RFS_P);;
        if jeq, jump wait_SPIF;;

        // Read received data and compose 32-bit word
        xr8 = xr9;;
        xr4 = [k0 + 0];;
        xr0 = [k0 + 0];;
        xr4 += fdep r0 by r8;;
        xr0 = [k0 + 0];             xr8 = r8 + r10;;
        xr4 += fdep r0 by r8;;
        xr0 = [k0 + 0];             xr8 = r8 + r10;;
        xr4 += fdep r0 by r8;;

        // First read is dummy due to command and address shift
        [j10 += j11] = xr4;;
        if nlc0e, jump loop_SPIF;   j11 = 0x01;;

        // There are 4 more bytes in the FIFO - read them
        comp(j10, 0xFF);;
        if jeq, jump wait_SPIF(np); lc0 = 0x01;;
```

```
        // Done! Release #CS0
        j0 = /*(1<<SPCR1_HOLDCS_P) |*/ (1<<SPCR1_SPE_P);;
        [j31 + SPCR1_LOC] = j0;;

        // Jump to target address
        cjmp(abs)(np);;


    //-------------------------------------------------------//
        // Parallel 8-bit boot EEPROM
        //
        //-------------------------------------------------------//
start_XPF:

        // Setup GPIO
        // Alternate function for nBMS and nRD signals
        j0 = (PF_ALT << 20) | (PF_ALT << 21);;
        // Alternate function for MS1 and MS0 signals
        j0 = j0 or (PF_ALT << 18) | (PF_ALT << 19);;
        [j31 + GPC_ALT_LOC] = j0;;

        // Alternate function for DATA and ADDR bus
        j0 = PX_ALT_PDB0;;                          // data (8-bit)
        j0 = j0 or PX_ALT_PAB0 | PX_ALT_PAB1;;      // addr (16-bit)
        [j31 + PX_ALT_LOC] = j0;;

        // Setup DMA
        xr0 = 0x00;;
        xr1 = (256 << 16) | 4;;
        xr2 = 0x00;;
        xr3 = TCB_EPROM | TCB_NORMAL;;

        xr4 = 0x00;;
        xr5 = (256 << 16) | 1;;
        xr6 = 0x00;;
        xr7 = TCB_INTMEM | TCB_NORMAL;;

        DCS0    = xr3:0;;
        DCD0    = xr7:4;;

wait_XPF:
        j1:0 = DSTAT;;
        j0 = j0 and 0x7;;
        comp(j0, DSTAT_DONE);;
        if njeq, jump wait_XPF;;
```

```
        // Jump to target address
        cjmp(abs)(np);;

    //-------------------------------------------------------//
        // Link Ports slave boot
        //
        //-------------------------------------------------------//
start_LINK1:

        j0 = LRX_DSIZE_1BIT;;

start_LINK:
        j1 = j31 + j0;;
        j0 = j0 or LRX_EN | LRX_CLK_L0;;
        j1 = j1 or LRX_EN | LRX_CLK_L1;;

        // Setup GPIO
        // Alternate function for LINK0 and LINK1 control lines
        j4 = (PF_ALT<<24) | (PF_ALT<<25) | (PF_ALT<<26) | (PF_ALT<<27) | \
                (PF_ALT<<28) | (PF_ALT<<29) | (PF_ALT<<30) | (PF_ALT<<31);;
        [j31 + GPC_ALT_LOC] = j4;;

        // Enable LINK clock PLL bypass
        j4 = (1<<PLL_BYPASS_P);;
        [j31 + PLL_LINK_CFG_LOC] = j4;;

        // Setup DMA
        xr0 = 0x00;;
        xr1 = (256 << 16) | 0x04;;
        xr2 = 0x00;;
        xr3 = TCB_INTMEM | TCB_QUAD;;

        // Enable LINK port receivers
        LRCTL0 = j0;;
        LRCTL1 = j1;;

        // Start DMA
        DC8 = xr3:0;;
        DC9 = xr3:0;;

        // Wait for DMA (any of two channels)
wait_LINK:
        j1:0 = DSTAT;;
        j2 = j1 and (0x7 << 0);;
        j3 = j1 and (0x7 << 3);;
        comp(j2, DSTAT_DONE << 0);;
        if jeq, jump wait_LINK_done(np);;
```

```
        comp(j3, DSTAT_DONE << 3);;
        if jeq, jump wait_LINK_done(np);;
        jump wait_LINK;;

wait_LINK_done:
        // Jump to target address
        cjmp(abs)(np);;


start_XROM:  cjmp = 0x80008000;;
          cjmp(abs)(np);;

   //-----------------------------------------------------//
        // NAND FLASH boot
        //
        //-----------------------------------------------------//

start_xNAND:  j1 = 0x60008;; // bit 18-17
                        xr0 = 0x6FF7C;;
                   [j9+0] = j1;;
            [j31+GPA_ALT_LOC] = xr0;;
                        jump  boot_xNAND (NP);;

start_NAND:

        // Setup GPIO
        // Alternate function for DATA and ADDR bus
        j0 = PX_ALT_PDB2 | PX_ALT_PDB3;;      // PXD[31:16] = alternate functions
        j0 = j0 or PX_ALT_PDB23F0;;           // PXD[31:16] = NAND FLASH
        [j31 + PX_ALT_LOC] = j0;;

        // NAND controller is active after reset. It's settings are:
        //        - page size = 2K
        //        - 3 bytes to select a page (row)
        //        - 2 bytes to select a column
        //        - 5 address bytes and 2 read commands

boot_xNAND:

        // Setup number of 32-bit words to read
        lc0 = 256;;
        [j31 + NAND_CNTR_LOC] = lc0;;
        j4 = 0x00;;             // start address

read_NAND:
        j0 = [j31 + NAND_SR_LOC];;
        j0 = j0 and NAND_SR_EMPTY;;
```

```
if njeq, jump read_NAND;;
xr0 = [j31 + NAND_DR_LOC];;
[j4 += 0x01] = xr0;;
if nlc0e, jump read_NAND;;

// Jump to target address

cjmp(abs)(np);;
nop;;nop;;nop;;nop;;
nop;;nop;;nop;;nop;;


_boot_start.end:
```

32.1 Загрузка из внешней NAND флэш-памяти

Код программы выполняет загрузку 256 слов данных из внешней NAND флэш-памяти во внутреннюю память, начиная с адреса 0. После завершения загрузки управление передается на адрес 0.

32.2 Загрузка EPROM

Выбор данного варианта соответствует загрузке процессора из внешней параллельной памяти типа EPROM. При выборе режима загрузки EPROM процессор инициализирует канал 0 DMA для передачи 256 32-разрядных слов кода из загрузочного EPROM в ячейки 0x00-0xFF блока 0 внутренней памяти процессора. После инициализации канала DMA процессор переходит в состояние проверки завершения передачи блока данных из внешней памяти во внутреннюю. Как только передача завершится, процессор передает управление по адресу 0.

32.3 Загрузка с использованием порта связи

Процессор программирует два канала DMA для приема данных из портов связи. Каналы DMA портов связи инициализируются для передачи блока 256 слов в адреса внутренней памяти с 0 по 255. После программирования передачи, процессор выполняет ожидание завершения пересылки и затем передает управление по адресу 0.

32.4 Старт по запросу прерывания

В данном варианте старта процессор определяет ширину внешней шины в 32 бита (вариант 1) или 16 бит (вариант 0), конфигурирует контроллер прерываний для обслуживания запросов прерываний по входам IRQ0, IRQ1, IRQ3, предварительно определив для них значения векторов прерываний 0x3000_0000, 0x3800_0000 и 0. Также разрешается векторное прерывание. Определив конфигурацию, процессор переходит в

режим ожидания прерывания. При возникновении одного из определенных ранее прерываний, процессор перейдет на его обработку. В данном режиме старта внешний хост через последовательный интерфейс может загрузить код программы во внутреннюю память и, воспользовавшись векторным прерыванием, запустить процессор на выполнение загруженной программы.

32.5 Выбор источника синхросигнала

Вывод общего назначения PC[30] во время сброса определяет способ подачи внешнего синхросигнала на процессор. Если во время сброса на этом выводе удерживается низкий уровень, то к выводам OSCI, OSCO должна быть подключена схема запуска внутреннего генератора с кварцевым резонатором. В противном случае – высокий уровень на PC[30] или неподключенный вывод PC[30] (за счет внутреннего резистора доопределения до «питания») требуют подключения генератора синхросигнала на вывод OSCI.

33 Часы реального времени RTC

Модуль RTC представляет собой набор счетчиков, которые используют для своей работы входную частоту процессора OSCI. В связи с отсутствием частоты 32,768 кГц, обычно используемой для часов реального времени, в данном модуле используется предварительный делитель входной частоты OSCI, с помощью которого можно задать опорную частоту, выполняющую такие же функции, что и частота 32,768 кГц.

Модуль RTC включает набор регистров, позволяющих управлять работой модуля, а также отслеживать его состояние. Регистры подключены к шине обмена периферийных устройств и имеют базовый адрес **0x800001E0**. Краткое описание регистров приведено в таблице 355.

Таблица 355 – Регистры модуля RTC

Номер	Название	Бит	Сброс	Регистр
0	RTC_CNT	32	0	Счетчик секунд
1	RTC_MR	32	max	Регистр сравнения
2	WDT_CNT	8	15	Текущее значение сторожевого счетчика
3	RTC_TDIV	20	0	Текущее значение делителя тик-импульса
4	RTC_CR	8	0	Регистр управления
5	RTC_SDIV	16	0	Текущее значение делителя секунды
6	TIC_VAL	20	255	Период тик-импульса минус 1
7	SEC_VAL	16	3	Количество тик-импульсов в секунде минус 1
8	RTC_BUSY	1	0	Флаг занятости интерфейса

Счетчики модуля RTC также могут быть использованы как дополнительные таймерные модуля, используемые для генерации различных временных интервалов.

33.1 Формирователь «тик-импульсов»

Входная частота OSCI поступает на счетчик RTC_TDIV, который по каждому импульсу входного синхросигнала увеличивает свое значение на единицу. Значение счетчика RTC_TDIV непрерывно сравнивается со значением регистра TIC_VAL. Если значение счетчика RTC_TDIV становится равным значению регистра TIC_VAL, то с приходом следующего импульса OSCI значение счетчика RTC_TDIV становится равным нулю и к контроллеру прерывания посылается запрос прерывания – «тик-импульс». Регистр TIC_VAL задает период следования тик-импульсов и содержит значение реального периода, уменьшенное на единицу. Тик-импульсы могут быть использованы операционной системой в качестве базового периода для работы программ в системе разделения времени. Период формирования тик-импульсов импульсов равен

$$\text{T\_тик} = \frac{\text{OSCI}}{\text{TIC\_VAL} + 1}. \qquad (12)$$

Полученные тик-импульсы используются модулем RTC для генерации импульсов секунд (точнее – псевдосекунд) и для работы сторожевого таймера.

33.2 Формирователь импульсов секунд

Модуль RTC содержит счетчик RTC_SDIV, используемый для формирования импульсов секунд. Счетчик тактируется с помощью синхросигнала OSCI, однако увеличивает значение RTC_SDIV на единицу, только в момент прихода тик-импульса. Значение регистра RTC_SDIV постоянно сравнивается со значением регистра SEC_VAL, который задает период работы счетчика RTC_SDIV в терминах тик-импульсов. Если значение счетчика RTC_SDIV равно значению регистра SEC_VAL, то с приходом следующего тик-импульса будет сформирован импульс секунды. Значение счетчика RTC_SDIV станет равным нулю, и он начнет повторный счет. Таким образом генератор секунд формирует импульсы секунд согласно формуле

$$\text{T\_сек} = \frac{\text{T\_тик}}{\text{SEC\_VAL} + 1}. \quad (13)$$

Секундные импульсы поступают на счетчик секунд RTC_CNT.

33.3 Счетчик секунд RTC_CNT

В данном счетчике содержится время в секундах прошедшее с момента включения часов. Счетчик может быть скорректирован. Счетчик тактируется синхросигналом OSCI, однако увеличивает свое значение на единицу только в момент прихода импульса секунд. Регистр RTC_CNT имеет 32 разряда. Его значение непрерывно сравнивается со значением регистра RTC_MR. В случае если значение счетчика RTC_CNT равно значению регистра RTC_MR, то в момент прихода следующего секундного импульса будет сформирован запрос прерывания ALARM к процессору. Регистр RTC_MR позволяет реализовать функции будильника и должен содержать значение времени срабатывания в секундах (минус единица от действительного значения).

33.3.1 Регистр сравнения RTC_MR

Регистр сравнения используется для реализации функций будильника. Его значение непрерывно сравнивается со значением регистра счетчика секунд. В момент равенства значений устанавливается соответствующий флаг и вырабатывается сигнал прерывания.

33.3.2 Регистр управления RTC_CR

Регистр управляет включением или выключение различных функций часов реального времени. Биты регистра кратко описаны в таблицу 356. По сигналу внешнего сброса все биты регистра устанавливаются в ноль.

Таблица 356 – Регистр управления

Бит	Имя	Тип	Назначение
1:0			
2	WDT_EN	R/W	Разрешение работы сторожевого таймера: 0 – выключен; 1 – включен
3	FREEZ	R/W	Блокировка сторожевого таймера. После установки этого бита невозможно выключить сторожевой таймер.
6:4	WDT_SEL	R/W	Выбор значения загружаемого в счетчик сторожевого таймера: 0 – 1; 1 – 2; 2 – 4; 3 – 8; 4 – 16; 5 – 32; 6 – 64; 7 – 128
7	LOCK	R/W	Управление доступом к модулю RTC: 0 – чтение-запись; 1 – только чтение (для RTC_CR – чтение-запись)
8	DIS_ATIC	R/W	Когда 1 запрещает генерацию асинхронного прерывания от импульса «тик»
9	DIS_ADOG	R/W	Когда 1 запрещает генерацию асинхронного прерывания от сторожевого таймера
10	DIS_AMR	R/W	Когда 1 запрещает генерацию асинхронного прерывания от регистра сравнения

Функции регистра управления заключаются в задании параметров работы модуля RTC. Биты регистра управления позволяют включить сторожевой таймер (WDT_EN), при необходимости заблокировать возможность его случайного выключения(FREEZ). С помощью разрядов WDT_SEL можно задать значение константы, которое будет загружаться в сторожевой таймер при его инициализации. Также возможно задание блокировки доступа к регистрам RTC для защиты от случайной записи (бит LOCK). В этом случае ко всем регистрам RTC (кроме регистра управления) возможен доступ только по чтению. В процессоре реализован режим пониженного энергопотребления (спящий режим), в котором все системные частоты могут быть остановлены. Активной может быть только частота OSCI. В этом случае для выхода из спящего режима могут быть использованы асинхронные запросы прерываний от тик-импульса, будильника или сторожевого таймера. Соответствующие биты регистра управления разрешают эти функции.

33.3.3 Регистр счетчика сторожевого таймера WDT_CNT

Этот восьмиразрядный регистр осуществляет вычитание 1 из своего содержимого при поступлении тик-импульса. При достижении значения WDT_CNT==1 и в момент прихода следующего тик-импульса, сторожевой таймер вырабатывает запрос на

прерывание. При этом значение WDT_CNT становится равным нулю. Запрос прерывания от сторожевого таймера должен рассматриваться как запрос на повторную инициализацию сторожевого таймера. Если значение WDT_CNT равно нулю и приходит очередной тик-импульс, сторожевой таймер сформирует сигнал сброса системы длительностью в один период тик-импульса. Косвенным образом можно записать новое значение в счетчик сторожевого таймера. Для этого вначале нужно записать соответствующее значение в биты WDT_SEL регистра управления, а затем записать константу 0x04072013 по адресу счетчика сторожевого таймера. В регистре WDT_CNT автоматически установится требуемое значение.

33.3.4 Регистр занятости интерфейса часов RTC_BUSY

Это однобитный регистр, доступный только по чтению. Проблема в работе с модулем RTC заключается в том, что все его внутренние регистры тактируются частотой OSCI. В тоже время процессор обращается к модулю через периферийную шину, частота которой может значительно превосходить частоту OSCI. В связи с этим в модуле реализован согласующий буфер между двумя доменами синхронизации. Любая запись в часы приводит к записи в буфер. Значение из буфера в дальнейшем передается в другой домен и там происходит запись. Пока вся процедура не выполнена, флаг занятости равен единице и новая запись невозможна. А именно новая запись приведет к ошибке. Чтение регистров реализовано следующим образом. При чтении регистра модуля часов происходит копирование значения регистра в буфер периферийной шины. В этом же такте старое значение буфера передается в процессор. Таким образом, чтобы прочитать текущее значение какого-то регистра необходимо выполнить два последовательных чтения.

Отметим, что при чтении регистра ***RTC_BUSY*** считывается непосредственное значение флага, и нет необходимости выполнять чтение повторно.

33.4 Ограничения модуля RTC

Модуль использует для своей работы входную тактовую частоту OSCI. При генерации запросов прерывания от тик-импульсов, будильника, сторожевого таймера, импульсы запросов синхронизируются частотой SOC-шины. Для надежной генерации прерывания необходимо чтобы частота SOC-шины была выше внешней входной частоты OSCI.

33.5 Состояние модуля RTC после сброса

Процессор может быть сброшен в исходное состояние следующими сигналами:

– внутренний сигнал сброса от модуля контроля включения питания процессора nPOR;

– активным уровнем по внешнему входу nRESET;

– сигналом от сторожевого таймера WDT_RES.

Часть ресурсов модуля RTC сбрасывается сигналом nPOR только при включении питания процессора. Этими ресурсами являются регистры: RTC_CNT, RTC_MR, RTC_TDIV, RTC_SDIV, TIC_VAL, SEC_VAL. Все другие регистры и флаги модуля

сбрасываются любым из трех вышеназванных источников. Таким образом, если пользователь определил конфигурацию часов реального времени, значение часов будет корректным до тех пор, пока питание процессора в норме. Никакие внешние события или внутренний сброс от сторожевого таймера не изменяют значение часов. В модуле CMU имеется флаг, который позволяет различить сброс сторожевого таймера от сброса из других источников. Однако нет возможности различить сброс по включению питания от внешнего сброса. Между тем факт внешнего сброса может быть вычислен косвенным образом. Например, если после сброса значение счетчика секунд не равно нулю, значит произошел сброс от внешнего сигнала и программа не должна выполнять процедуру настройки часов повторно.

Регистр управления модулем RTC_CR сбрасывается в любом из трех случаев.

34 Схема формирования внутреннего сброса по включению питания

В микросхеме реализована схема формирования внутренних сигналов сброса по включению питания как по домену U_{CCIO} – сигнал por_b_33, так и по домену U_{CC} – сигнал por_b_12. Низкий уровень сигнала por_b_33 удерживает в выключенном состоянии приемопередатчики портов связи, высокий уровень разрешает их работу. Низкий уровень сигнала por_b_12 приводит все триггеры микросхемы в начальное состояние и удерживает схему в сбросе. Высокий уровень разрешает работу микросхемы. Алгоритм формирования этих сигналов следующий:

1 В начальный момент значение сигналов por_b_12, por_b_33 равно нулю;

2 При достижении обоими напряжениями питания U_{CCIO} и U_{CC} значений пороговых уровней U_{CCIO_th} и U_{CC_th} соответственно начинает работать внутренний счетчик;

3 Через промежуток времени T_{ON} внутренним счетчиком устанавливаются сигналы por_b_12 и por_b_33 в состояния U_{CC} и U_{CCIO} соответственно.

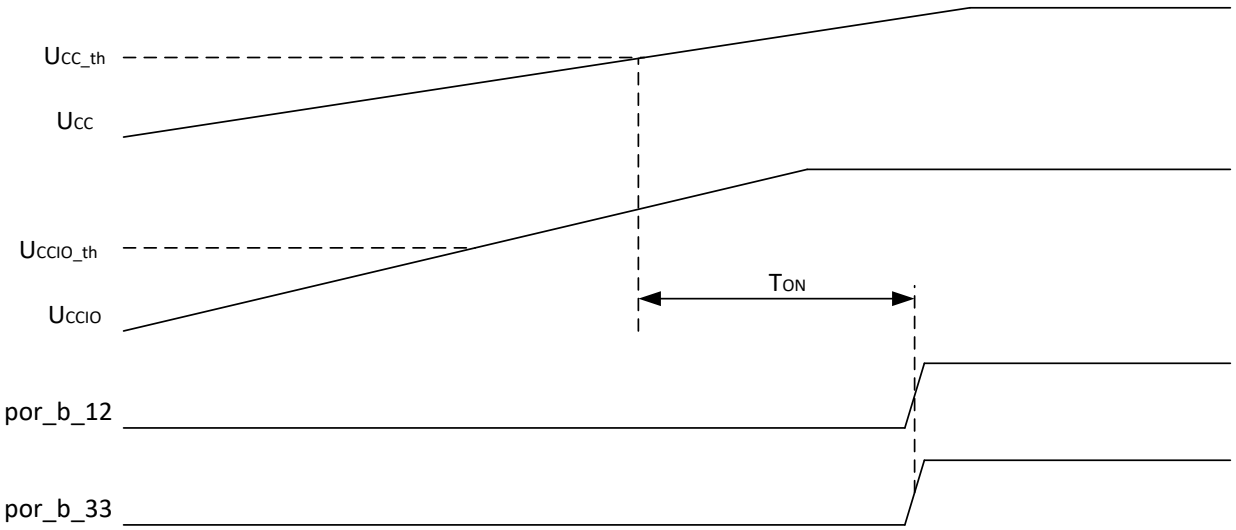
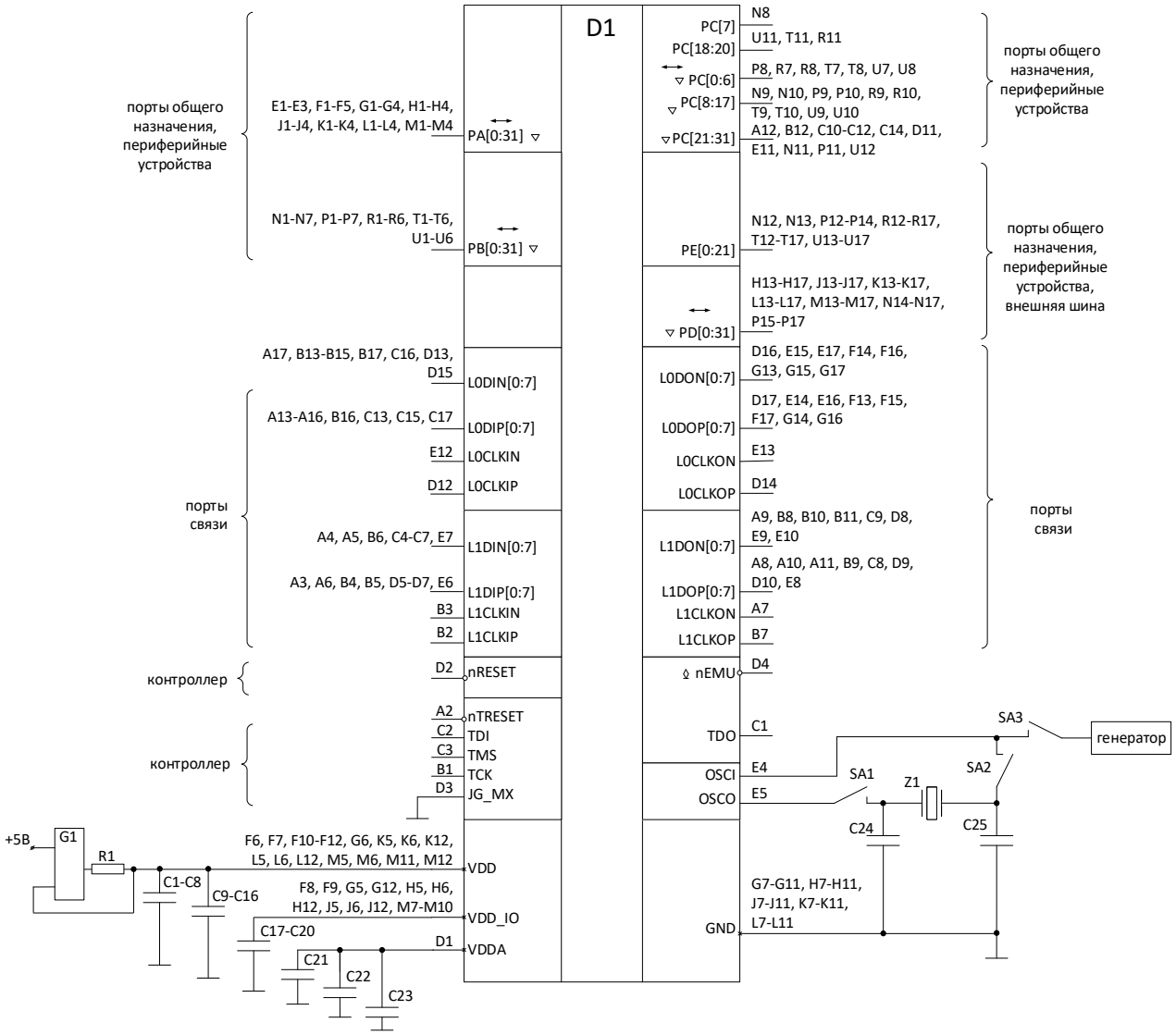


Рисунок 169 – Формирование внутренних сигналов сброса по включению питания

Таблица 357 – Параметры формирования внутренних сигналов сброса

Буквенное обозначение параметра, единица измерения	Норма параметра		Наименование параметра
	не менее	не более	
U_{CCIO_th}, В	2,75	2,81	Пороговое значение питания U_{CCIO}
U_{CC_th}, В	0,98	1,0	Пороговое значение питания U_{CC}
T_{ON}, мс	12	28	Задержка включения схемы

35 Типовая схема включения



C1 ÷ C8, C17 ÷ C21 – конденсаторы емкостью 100 нФ;
C9 ÷ C16, C22 – конденсаторы емкостью 10 нФ;
C23 – конденсатор емкостью 100 пФ;
C24, C25 – конденсатор емкостью 15 пФ;
D1 – включаемая микросхема;
G1 – генератор питания;
R1 – резистор сопротивлением не менее 2,1 Ом;
SA1 ÷ SA3 – переключатели;
Z1 – кварцевый резонатор.

Рисунок 170 – Типовая схема включения микросхем

36 Типовые зависимости

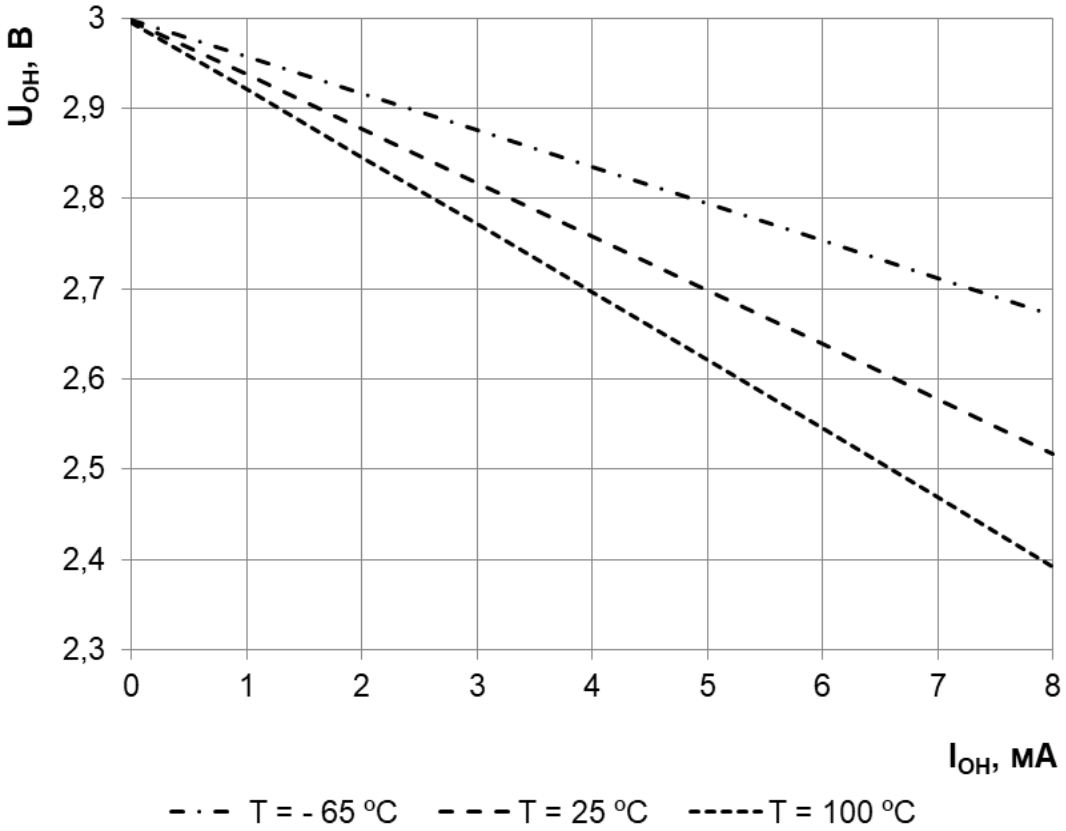


Рисунок 171 – Зависимость выходного напряжения высокого уровня U_{OH} от выходного тока высокого уровня при U_{CC} = 1,08 В, U_{CCIO} = 3,0 В

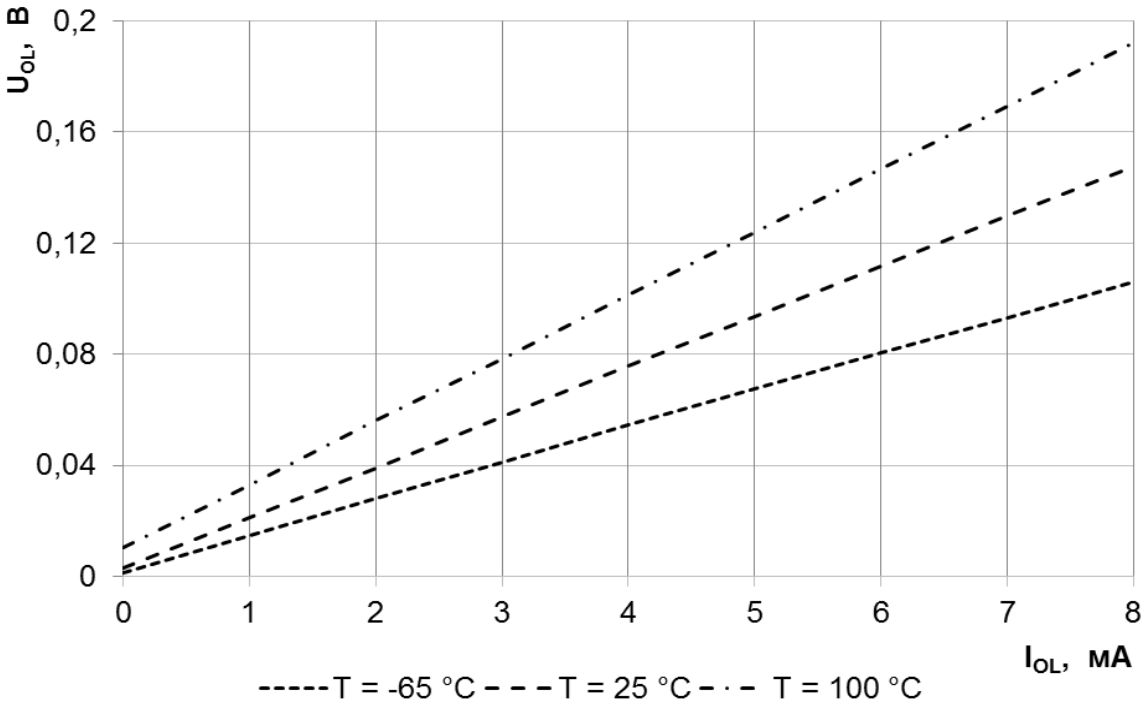


Рисунок 172 – Зависимость выходного напряжения низкого уровня U_{OL} от выходного тока низкого уровня при U_{CC} = 1,08 В, U_{CCIO} = 3,0 В

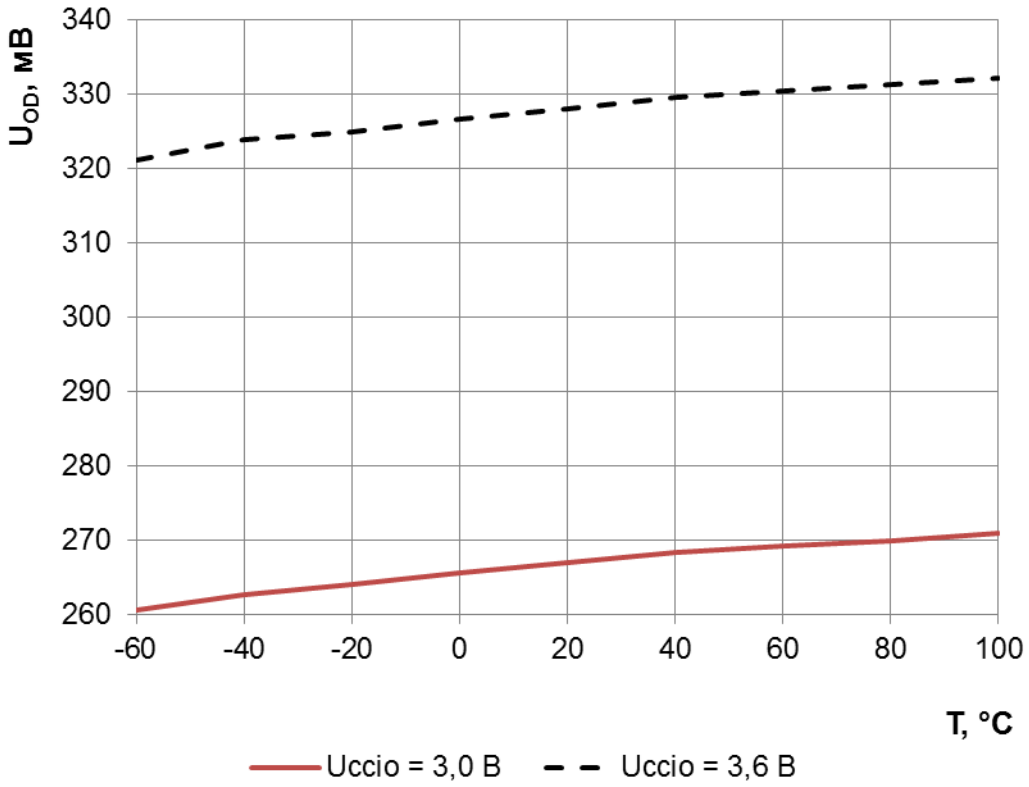


Рисунок 173 – Зависимость дифференциального выходного напряжения LINK-портов U_{OD} от температуры при R_L=100 Ом

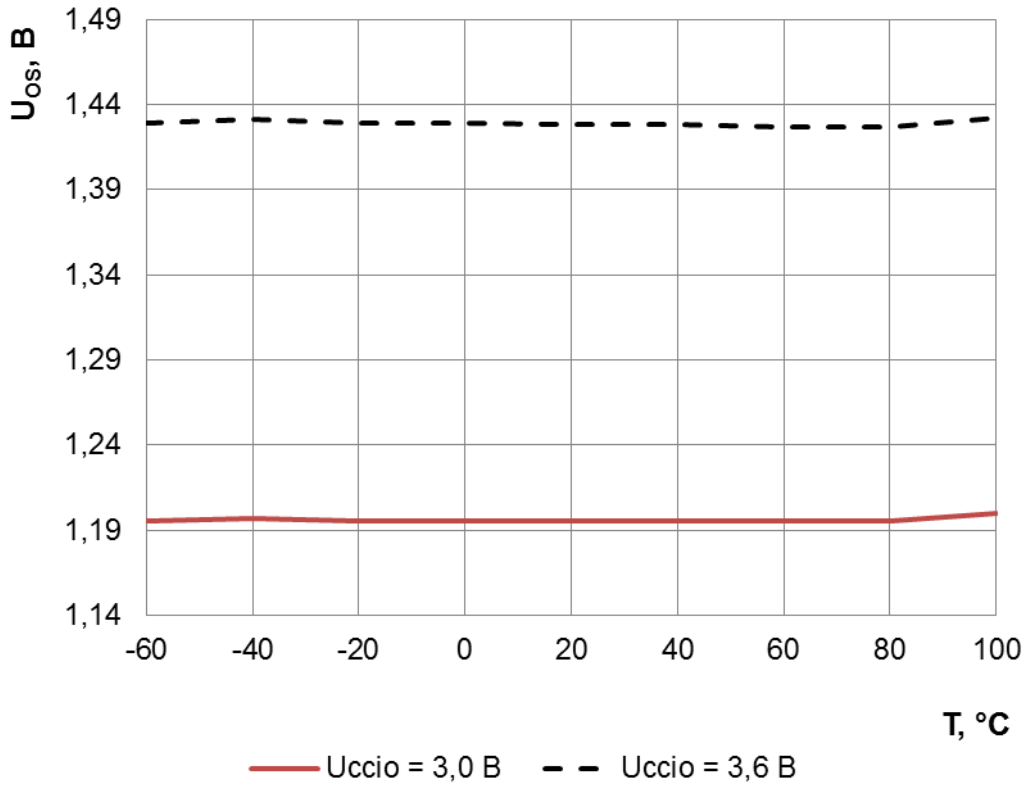


Рисунок 174 – Зависимость синфазного выходного напряжения LINK-портов U_{OS} от температуры при R_L=100 Ом

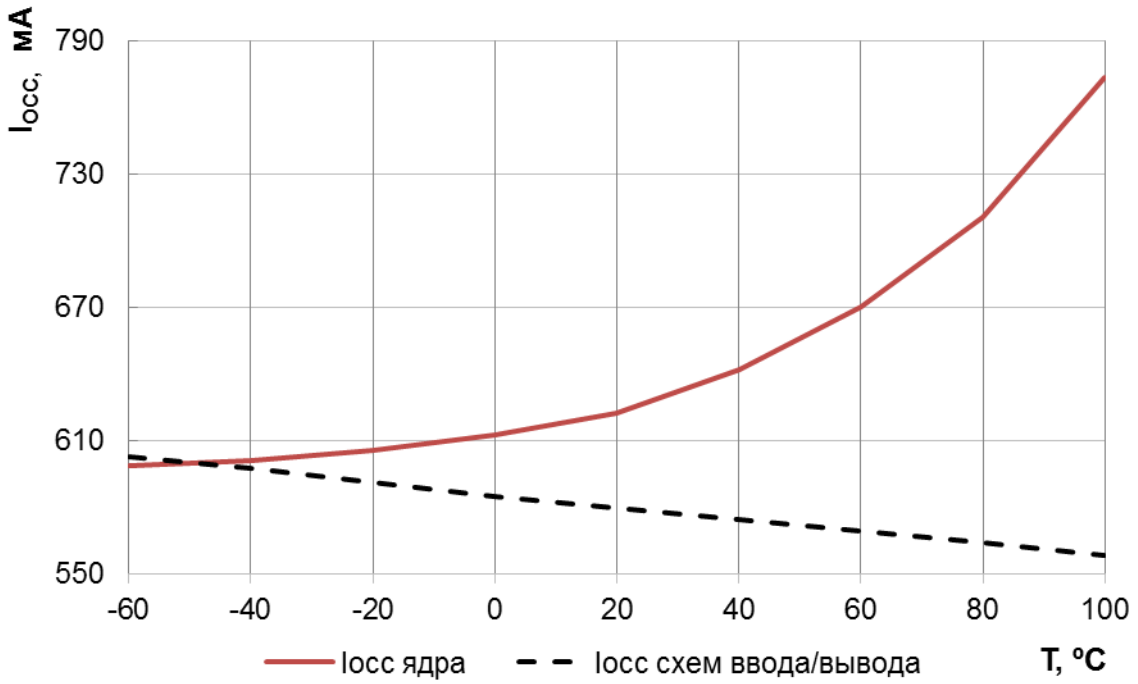


Рисунок 175 – Зависимость динамического тока потребления I_{OCC} ядра и схем ввода/вывода от температуры при U_{CC} = 1,32 В, U_{CCIO} = U_{CCA} =3,6 В, f_C = 230 МГц

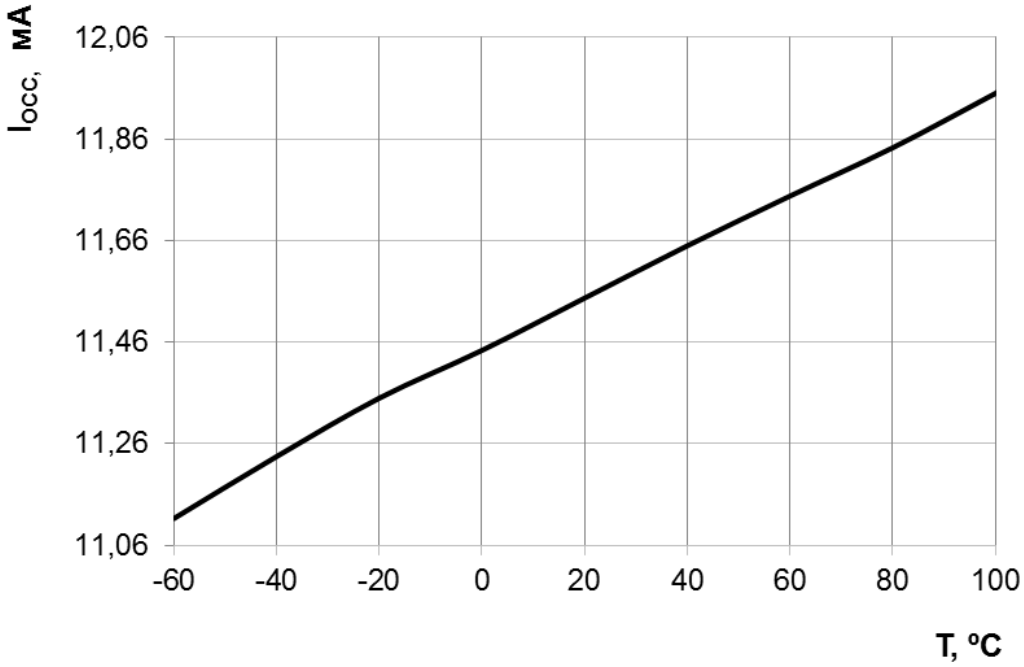


Рисунок 176 – Зависимость динамического тока потребления I_{OCC} аналоговых блоков от температуры при U_{CC} = 1,32 В, U_{CCIO} = U_{CCA} =3,6 В

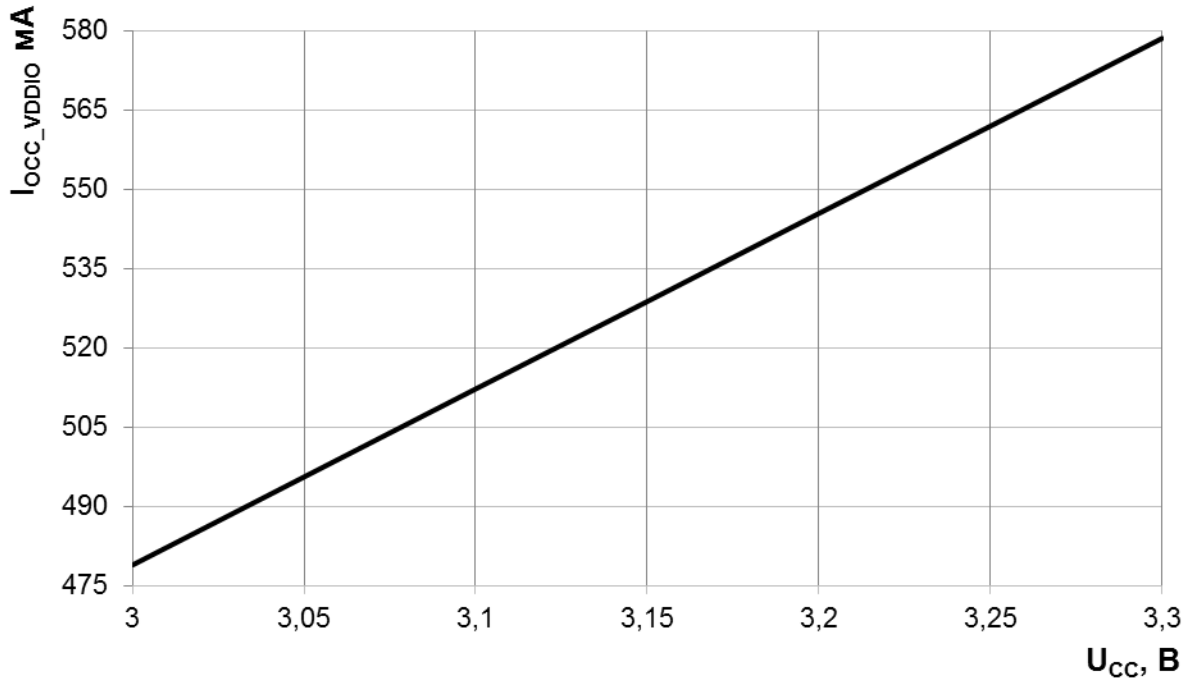


Рисунок 177 – Зависимость динамического тока потребления I_{OCC} схем ввода/вывода от напряжения питания при f_C = 230 МГц, T = 25 ºC

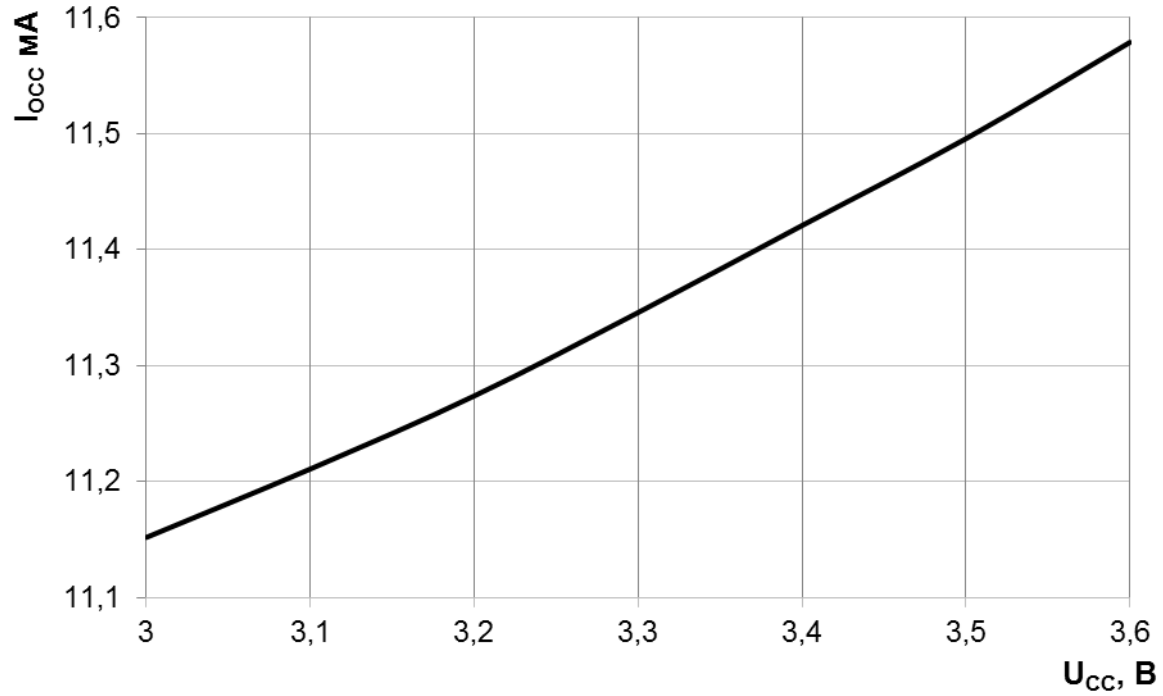


Рисунок 178 – Зависимость динамического тока потребления I_{OCC} аналоговых блоков от напряжения питания при f_C = 230 МГц, T = 25 ºC

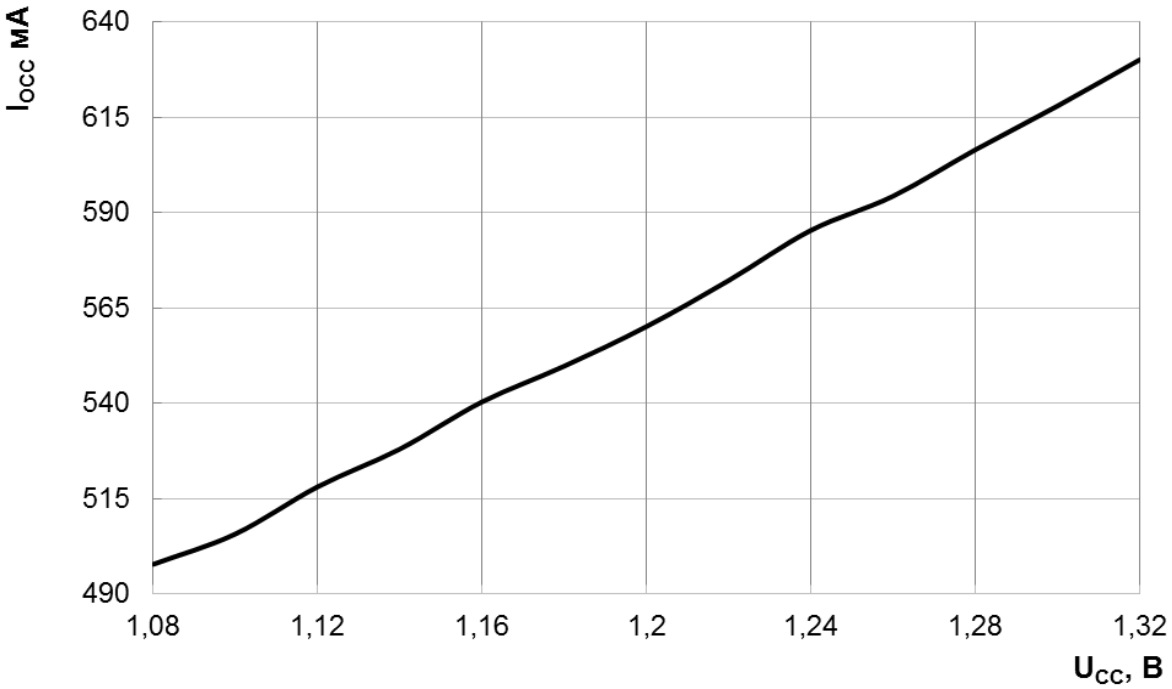


Рисунок 179 – Зависимость динамического тока потребления I_{OCC} ядра от напряжения питания при f_C = 230 МГц, T = 25 ºC

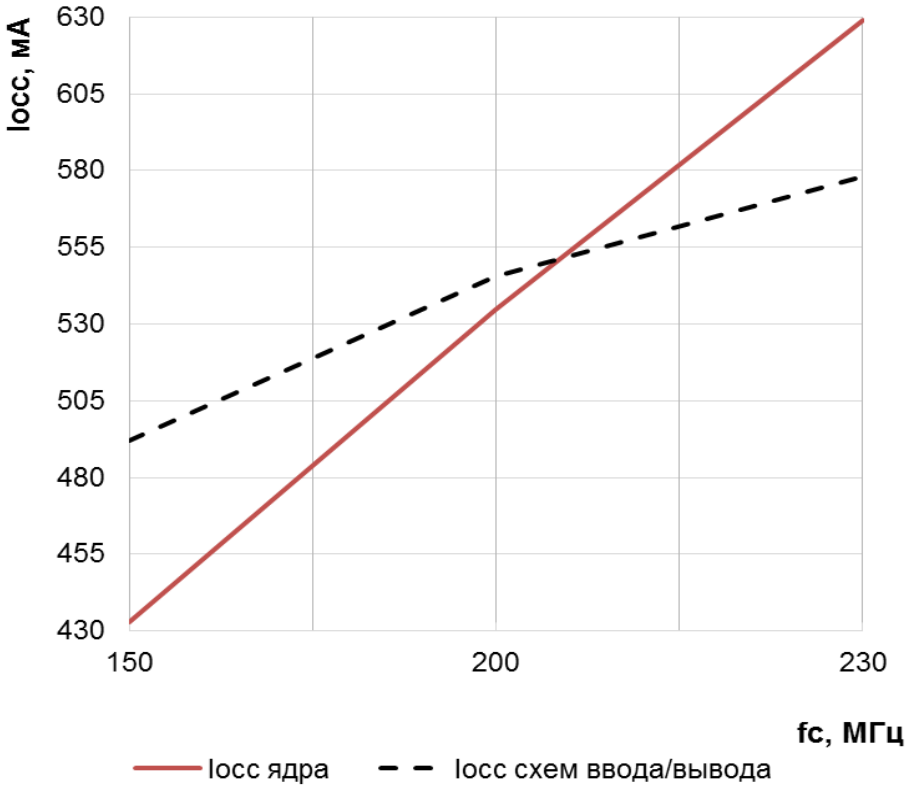


Рисунок 180 – Зависимость динамического тока потребления I_{OCC} ядра и схем ввода/вывода от тактовой частоты процессора f_C при U_{CC} = 1,32 В, U_{CCIO} = U_{CCA} = 3,6 В, T = 25 ºC

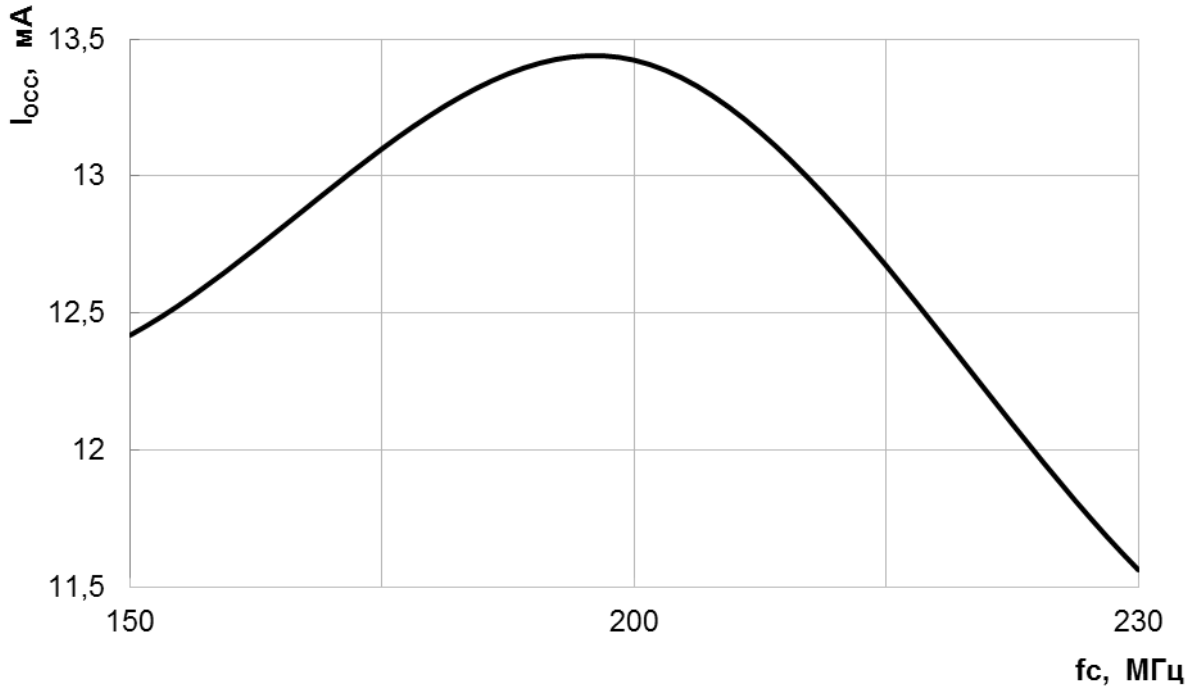


Рисунок 181 – Зависимость динамического тока потребления аналоговых блоков I_{OCC} от тактовой частоты процессора f_C при U_{CC} = 1,32 В, U_{CCIO} = U_{CCA} = 3,6 В, T = 25 ºC

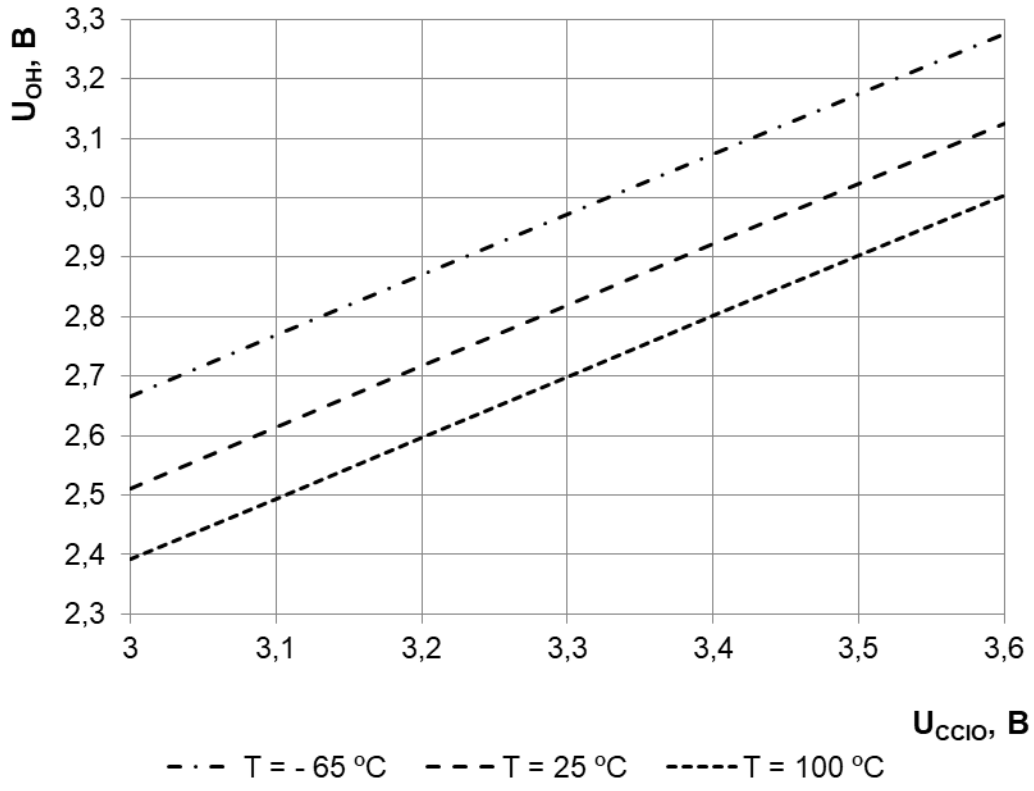


Рисунок 182 – Зависимость выходного напряжения высокого уровня U_{OH} от напряжения питания схем ввода/вывода U_{CCIO} при U_{CC} = 1,08 В, I_{OH} = – 4 мА

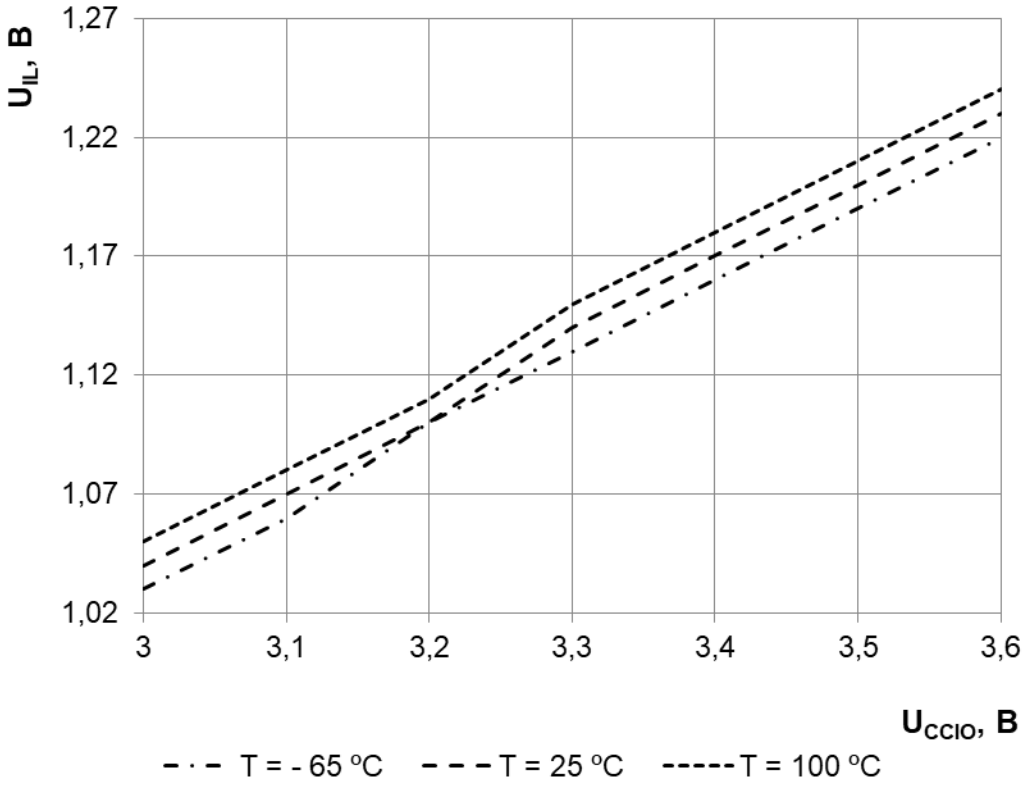


Рисунок 183 – Зависимость входного напряжения низкого уровня U_{IL} от напряжения питания схем ввода/вывода U_{CCIO} при U_{CC} = 1,08 В

37 Предельно-допустимые режимы

Таблица 358 – Предельно-допустимые режимы эксплуатации и предельные электрические режимы микросхем

Наименование параметра, единица измерения	Буквенное обозначение параметра	Предельно-допустимый режим		Предельный режим	
		не менее	не более	не менее	не более
Напряжение питания, В:					
– ядра на выводах VDD;	U_{CC}	1,08	1,32	–	1,4
– схем ввода/вывода на выводах VDD_IO;	U_{CCIO}	3,0	3,6	–	4,0
– аналоговых блоков на выводе VDDA	U_{CCA}	3,0	3,6	–	4,0
Входное напряжение высокого уровня, В	U_{IH}	2,0	U_{CCIO}	–	U_{CCIO} + 0,3
Входное напряжения низкого уровня, В					
- на всех выводах, кроме OSCI;	U_{IL}	0	0,8	– 0,3	0
- на выводе OSCI		0	0,4	– 0,3	0
Входное напряжение дифференциальное LINK-портов, В, на парах выводов L0DIN[0:7]-L0DIP[0:7], L0CLKIN- L0CLKIP, L1DIN[0:7], L1DIP[0:7], L1CLKIN, L1CLKIP	U_{ID}	0,2	1,2	–	–
Входное напряжение синфазное LINK-портов, В, на парах выводов L0DIN[0:7]-L0DIP[0:7], L0CLKIN-L0CLKIP, L1DIN[0:7]-L1DIP[0:7], L1CLKIN- L1CLKIP	U_{IS}	0,6	1,8	–	–
Напряжение, прикладываемое к выходу микросхемы в состоянии «Выключено», В	U_{OZ}	0	U_{CCIO}	– 0,3	U_{CCIO} + 0,3
Выходной ток высокого уровня, мА, на выводах портов A, B, C, D, E	I_{OH}	– 8*	0	– 24	–
Выходной ток низкого уровня, мА, на выводах портов A, B, C, D, E	I_{OL}	0	8	–	24
Тактовая частота процессора, МГц	f_C	–	230	–	–
Тактовая частота LINK-портов, МГц	f_{LINK}	–	180	–	–
Входная частота PLL, МГц	f_{C_PLL}	10	100	–	–
Емкость нагрузки каждого выхода, пФ	C_L	–	15	–	–

Примечания

1 *Суммарный выходной ток высокого уровня должен быть не более 800 мА;

2 Допускается использование разных источников для питания схем ввода/вывода и аналоговых блоков, при этом разница между напряжениями питания не должна превышать 200 мВ;

3 Не допускается одновременное воздействие двух и более предельных режимов

38 Электрические параметры

Таблица 359 – Электрические параметры микросхем при приемке и поставке

Наименование параметра, единица измерения, режим измерения	Буквенное обозначение параметра	Норма параметра		Температура среды, °С
		не менее	не более	
Выходное напряжение высокого уровня, В - при I_{OH} = – 8 мА;	U_{OH}	2,0	–	25, 85, – 40
- при I_{OH} = – 4 мА		2,4	–	
Выходное напряжение низкого уровня, В	U_{OL}	–	0,4	
Дифференциальное выходное напряжение LINK-портов уровней логического «0» и логической «1», мВ, R_L* = 100 Ом	U_{OD}	220	650	
Синфазное выходное напряжение LINK-портов уровней логического «0» и логической «1», В, R_L = 100 Ом	U_{OS}	1,10	1,55	
Ток утечки высокого уровня на входе, мкА	I_{ILH}	– 20	20	
Ток утечки низкого уровня на входе, мкА	I_{ILL}	– 20	20	
Ток утечки высокого уровня на входе с внутренним резистором доопределения до питания, мкА, $U_{IH} = U_{CCIO}$	I_{ILH_U}	– 20	20	
Ток утечки низкого уровня на входе с внутренним резистором доопределения до нуля, мкА, U_{IL} = 0 В	I_{ILL_D}	– 20	20	
Входной ток высокого уровня на выводе с внутренним резистором доопределения до нуля, мкА	I_{IH_D}	20	150	
Входной ток низкого уровня на выводе с внутренним резистором доопределения до питания, мкА	I_{IL_U}	– 150	– 20	
Выходной ток высокого уровня в состоянии «Выключено», мкА, $U_{OZ} = U_{CCIO}$	I_{OZH}	– 20	20	
Выходной ток низкого уровня в состоянии «Выключено», мкА, U_{OZ} = 0 В	I_{OZL}	– 150	– 20	
Динамический ток потребления, мА: – ядра, при f_C = 230 МГц;	I_{OCC}	–	1000	
– аналоговых блоков;		–	20	
– схем ввода/вывода		–	800	
* R_L – резистор, подключаемый между выходами дифференциальной пары				

Микросхемы устойчивы к воздействию статического электричества с потенциалом не менее 2 000 В.

39 Справочные данные

Значение собственной резонансной частоты не менее 3,4 кГц.
Тепловое сопротивление кристалл-корпус не более 1,7 °С/Вт.
Предельная температура p-n перехода кристалла 150 °С.

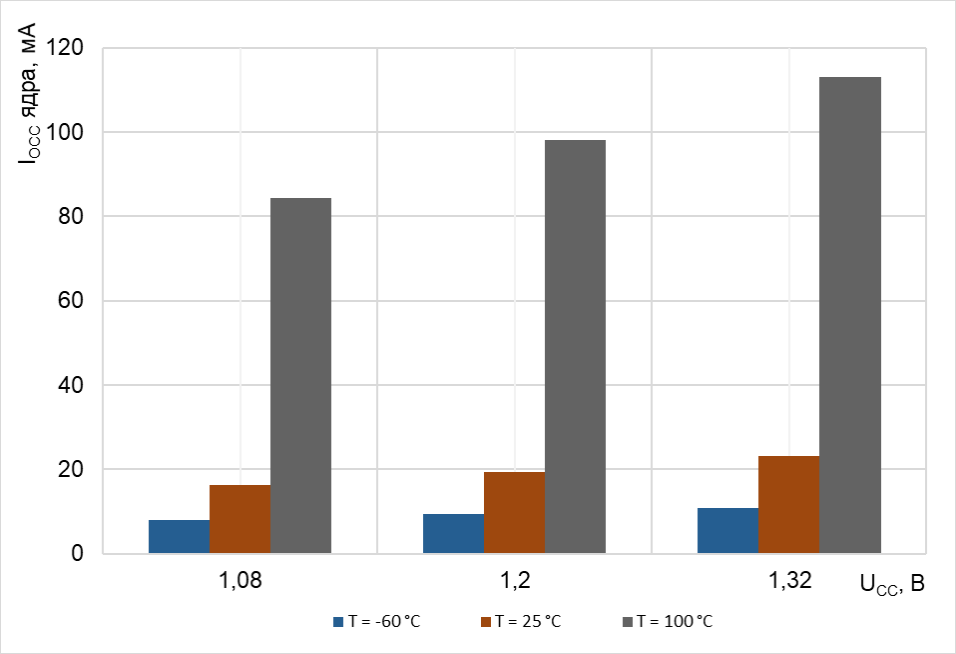


Рисунок 184 – Зависимость тока потребления I_{OCC} ядра от температуры и напряжения U_{CC} в режиме SLEEP

40 Габаритный чертеж микросхемы

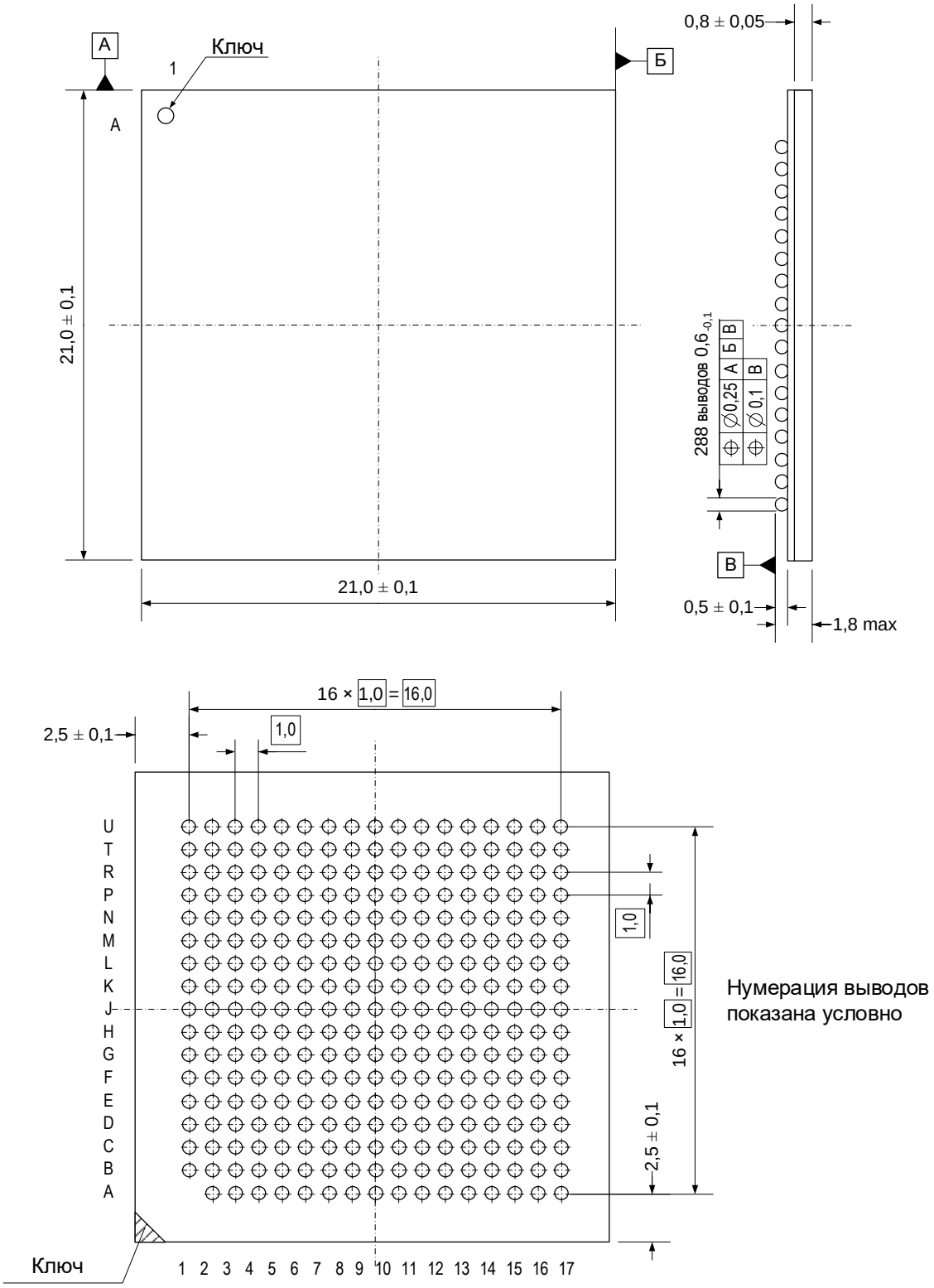


Рисунок 185 – Микросхема в корпусе BGA288

41 Информация для заказа

Обозначение	Маркировка	Тип корпуса	Температурный диапазон, °С
К1967ВН04BG	MDR1302	BGA288	от – 40 до 85
Примечание – Маркировка микросхем ревизии 2 – MDR32S2BG			

Лист регистрации изменений

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
1	03.08.2021	2.0.0	Введена вновь	
2	10.09.2021	2.0.1	Термин «пластиковый» заменен на «пластмассовый»	1
3	21.09.2021	2.1.0	Добавлена временная диаграмма интерфейса видеокамеры	275
4	26.05.2023	2.2.0	Раздел 4 – добавлен материал шариковых выводов и рекомендации по пайке Раздел 9 скорректирован и дополнен Пункты 13.1.3, 13.2.4, 13.2.5 Рисунок 64 скорректирован Таблицы 102-104 – названия столбцов скорректированы Таблица 105 – описание битов 13:4 скорректировано «Выбор размера страницы SDRAM» – добавлена формула Таблица 123 – смещение регистров LTSTATC0 и LTSTATC1 скорректировано Подраздел 15.9 – добавлена информация о 16-разрядной шине данных Таблица 127 – описание бита RDSIZE дополнено Таблица 128 – описание бита TDSIZE дополнено Подраздел 15.20 обновлен Подраздел 16.3 дополнен Таблица 142, рисунок 91 – SPI_EN –> SPI_CS Рисунки 143, 147-149 скорректированы Подразделы 26.7-26.9 – примеры скорректированы Таблицы 265, 266 – описание битов 23:20 скорректировано Раздел 26.9 –информация о режиме работы ОУ-ОУ дополнена Таблицы 267, 268 – описание битов Подпункт 28.1.1.1 –информация о бите ROUNDM исключена Подраздел «Схема подключения отладчика JEM-LYNX» перенесен в раздел 29 Раздел «Справочные данные» добавлен Исправление опечаток	36 105-140 225, 252, 253 241 243-244 245 247 274 276 285 286 291 295 302, 305 523, 529-531 531-534 535, 536 534 537 558 634 675 По тексту
5	21.12.2023	2.3.0	Добавлена информация о микросхемах ревизии 3 с маркировкой MDR1302 Таблица 1 дополнена, описание выводов PA[23], PA[24] скорректировано Таблица 2 дополнена Пункт 6.8.1 добавлен	По тексту 19 54 66

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
			Раздел 9 обновлен	105-140
			Таблица 67 – добавлено описание бита 34 PiDS	142
			Таблица 68 – добавлено описание бита 7 PE_DS и бита 11 PD_DS	146
			Таблицы 72, 73, 75- 78 обновлены	153-158, 169
			Таблицы 83, 84 обновлены	180, 181
			Раздел 13 – информация о восьмиразрядной шине данных добавлена	220
			Таблица 100 – описание бита 22 MS0_PIPE и бита 23 MS1_PIPE добавлено	228
			Таблица 120 – столбец «Тип вывода» обновлен	267
			Таблица 123 – регистры LIM_VAL(0,1) добавлены	274
			Подраздел 15.9 дополнен	276
			Пункт 15.12.3 скорректирован	284
			Таблица 127 – описание битов 5:4 RDSIZE, 15 ADC_DDR скорректировано, бита 7 RX_EXT_EN, 17 EVEN_ODD – добавлено	285
			Таблица 128 – описание битов 5:4 TDSIZE скорректировано	286
			Подразделы 15.15, 15.19 добавлены	287, 290
			Таблица 133 обновлена	293
			Подраздел 16.3 обновлен	295
			Таблица 145 – описание битов SRC обновлено	304
			Формула (7) – примечание добавлено	305
			Таблица 162 – описание бита 16 EN_CORR и бита 17 WT_ESRV добавлено	325
			Таблица 183 добавлена	349
			Таблица 184 скорректирована	349
			Раздел 22 «Контроллер USB канального уровня» добавлен	365
			Раздел 23 «Интерфейс CAN FD» добавлен	418
			Таблица 251 – описание битов 31:24 скорректировано	485
			Таблицы 263, 285 – столбец «Тип вывода» обновлен	
			Пункт 28.1.1 – исключена информация, что режим DOWN – тестовый	524, 546 557
			Пункт 28.1.5 – дополнен	
			Раздел 29 «Интерфейс Ethernet 10/100/1000» добавлен	560
			Таблицы 343, 344, 348 дополнены	565
			Подраздел 31.2 дополнен	
			Подразделы 31.5 – 31.9 Регистры CFG(11-15) добавлены	635, 637, 642 639
			Исправление опечаток	643-646 По тексту
6	27.03.2024	2.4.0	УГО – обозначение микросхемы исправлено	18

№ п/п	Дата	Версия	Краткое содержание изменения	№№ изменяемых листов
			Раздел 25 скорректирован Таблицы 258, 261 – описание битов CLK и DIV[7:0] скорректировано	509 515, 520
7	06.08.2024	2.5.0	Подраздел 17.6 – исправлена пунктуационная ошибка Таблица 338 – для IDCODE добавлено значение по умолчанию для ревизии 3	310 631