USER MANUAL

Multicellular processors:

MCp042P200102-LQ256

MCp042R100102-LQ256

Abstract

MCp042P200102/MCp042R100102 microprocessor includes a multicellular CPU core, which is the first processor core with a radically new (post-Neumann) multicellular architecture developed in Russia. Multicellular processor is designed for a wide range of control tasks and digital signal processing in applications that require minimum power consumption and high performance. Given multicellular processor consists of 4 cells (coherent processing units) combined by intellectual commutation environment. MCp042R100102 has dynamically reconfigurable core. Cells can be distributed among tasks according with algorithm inside firmware. Wherein any part of code can be processed by any number of cells at any combinations. Core of MCp042P200102 can process only one task at one time.

Features:

- Cell amount — 4
- Processor word — 32/64 бита
- DM — 128KiB (4*4K*64)
- PM — 128KiB (4*4K*64)
- FPU SP/DP (in each cell)
- Clock frequency — 150 MHz
- Processor performance — 24 MFLOPS/MHz;

General characteristics:

- Package — LQFP-256
- Operation environment – (-60 ...+125°C)
- Maximum power consumption:
- core — 0.6Wt at 100MHz
- GPIOs — 80мВт
- Supply voltage (separate): cores — 1,8V, peripheral — 3,3V

Peripherals:

- 2 SPI interfaces with slave devices' selector (in "master" mode)
- 4 UART with FIFO for receive-transmit
- 2 I2C interfaces (one "master" and one "slave")
- I2S interface (one "master", receive data)
- Ethernet MAC controller 10/100Mb/s
- USB 2.0 FS (device), external interface ULPI
- RTC with calendar
- 4 GP timers
- 6 GPIO, total amount of input-output ports – 166
- PWM controller with 4 channels
- WDT
- ADC(48ksps, 16bit, 8ch)
- DAC(125ksps, 12bit, 2ch)

Contents

1 Type codes and abbreviations **5**
1.1 List of abbreviations 5
1.2 conventional Type Codes 5

2 Description **6**
2.1 General technical characteristics 6
2.2 MP structure 7

3 Central processor unit **9**
3.1 Registers 10

4 Memory organization **12**
4.1 Memory map 12
4.2 Memory switch 15
4.3 Data transaction controller (DTC) 17

5 Clock system **18**

6 Debugging system **19**

7 Peripheral devices **20**
7.1 Input-output port (GPIO) 20
7.1.1 Brief characteristics 20
7.1.2 GPIO functioning 20
7.1.3 Register description 22
7.2 External memory bus controller (MCTRL0) 24
7.2.1 Short description 24
7.2.2 Интерфейс PROM, I/O 25
7.2.3 Интерфейс SDRAM 27
7.3 UART(UARTx) interface 28
7.3.1 Brief characteristics 28
7.3.2 Data transmission 28
7.3.3 Data receipt 29
7.3.4 Transmission speed settings 30
7.3.5 Self-test modes 30
7.3.5.1 Self-test mode on interface line level 30

7.3.5.2 Self-test mode on data level . . . 30
7.3.6 Interrupt formation . . . 30
7.3.6.1 Receiver pending interrupt mode . . . 31
7.3.7 Registers' description . . . 32
7.4 SPI interface (SPIx) . . . 35
7.4.1 General characteristics . . . 35
7.4.2 3-wire operation . . . 37
7.4.3 Data receive and transmit . . . 37
7.4.4 SCK clock signal . . . 37
7.4.5 Operation in «master» mode . . . 37
7.4.6 Operation in «slave» mode . . . 38
7.4.7 Descriptiom of registers . . . 39
7.5 Interface I^2C«master» (I2C0) . . . 42
7.5.1 Brief characteristics . . . 42
7.5.2 General description of receive-transmit protocol . . . 43
7.5.3 Carrier frequency generation . . . 44
7.5.4 Interface operation algorythm . . . 44
7.5.4.1 Data record . . . 44
7.5.4.2 Data reading . . . 45
7.5.5 Description of registers . . . 47
7.6 Interface I^2C «slave» (I2C1) . . . 50
7.6.1 Brief characteristics . . . 50
7.6.2 General description of receive-transmit protocol . . . 51
7.6.3 Carrier frequency generation . . . 52
7.6.4 Interface operation algorythm . . . 52
7.6.4.1 Data receipt from «master» . . . 52
7.6.4.2 Data transmission to «master» . . . 52
7.6.5 description of registers . . . 54
7.7 Controller I^2S(I2Sx) . . . 57
7.7.1 Brief characteristics . . . 57
7.7.2 Bus general description I^2S . . . 57
7.7.3 Description of registers . . . 59
7.8 general purpose timer (GPTIMx) . . . 61
7.8.1 Brief characteristics . . . 61
7.8.2 Work alogorythm . . . 61
7.8.3 Register description . . . 63
7.9 Controller Ethernet(Ethernet0) . . . 65

7.9.1 Brief description . . . 65
7.9.2 Clocking . . . 66
7.9.3 Access to internal FIFO receive-transmit buffers. . . . 66
7.9.4 Transmitter DMAC . . . 66
7.9.4.1 Decriptor setting . . . 66
7.9.4.2 Data preparation for transmission . . . 67
7.9.4.3 Data transmission . . . 67
7.9.4.4 Descriptor operation after data transmission completion . . . 67
7.9.5 Receiver DMAC . . . 68
7.9.5.1 Descriptor setting . . . 68
7.9.5.2 Data receipt . . . 68
7.9.5.3 Descriptor processimg after date transmission is finished . . . 69
7.9.6 Description of registers . . . 70
7.10 PWM controller (PWMx) . . . 73
7.10.1 Brief characteristics . . . 73
7.10.2 PWM initialization . . . 73
7.10.3 PWM operation modes . . . 74
7.10.4 PWM Interrupts . . . 74
7.10.5 PWM pulse duration . . . 74
7.10.6 Description of registors . . . 75

8 Processor's pinmap assignation 78
8.1 Processor input/output table QFP256 . . . 78
8.2 Processor output diagram in QFP256 package . . . 84

9 Electric characteristics 85
9.1 Electrical characteristics of input-output ports . . . 85

1 Type codes and abbreviations

1.1 List of abbreviations

MP — microprocessor;

SW — software;

MSb — most significant bit;

MSB — most significant bit;

LSb — least significant bit;

LSB — least significant bit;

RAM — random access memory;

PM — program memory;

DM — data memory;

DMA — direct memory access;

GPR— general purpose register(s);

1.2 conventional Type Codes

'1', '0'— state of logical item, logical zero, accordingly;

REG(BIT) — this record is used for pointing bit in register, where REG is register's name, and BIT is an indication of bit group in it. For example, "bit I2CxCR(EN)" signifies that there is a designation at EN bit of I2CxCR register, and "I2CxPSC(PSC)" signifies bit group PSC. To find out more details about designated registers and bits, see these registers' description

PUx, BLOCKx, PUxREG — In the notations of registers, PU and MP names sumbol "x" can be used. This is digit replacement, for example, there are several identical peritheral UART blocks possesing digits 0,1 and so on. For UART0 "x" it is 0. If there is I2CxCR register then register for I2C0 is being called I2C0CR

2 Description

2.1 General technical characteristics

Notation	MCp042P200102/MCp042R100102
General purpose of function	64-bit processor
Multicellular core	4 cells 32/64bit
FPU(ieee754)	single/double precision
Peak performance	24 MFLOPS/МГц
Internal SRAM, KiB	512
External memory bus, bit	32
USB 2.0 FS device	1
UART	4
SPI	2
I2C	1 "master", 1 "slave"
I2S	1 "master"/"slave" (receiver/transceiver)
PWM	1, 4 channels
Ethernet 10/100	1
DAC, 12bit, 125ksps,	2 channels
ADC, 16bit, 48ksps,	8 channels
GPIO pins	166

2.2 MP structure

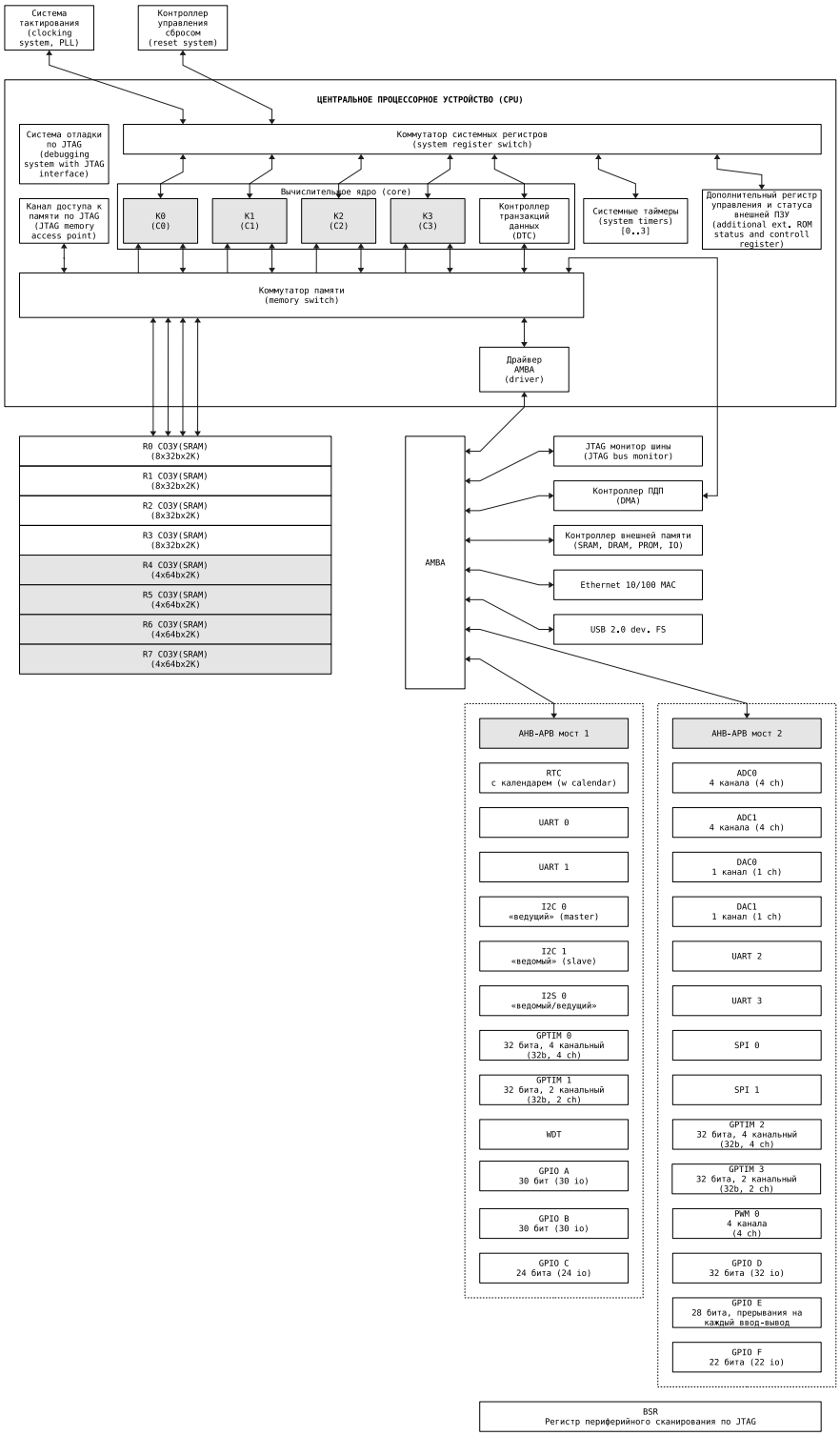

Figure 1: General structure of MCp042P200102/MCp042R100102

MP consist of:

- CPU;
- Peripheral devices. Peripheral bus is AMBA 2.0;
- SRAM.

3 Central processor unit

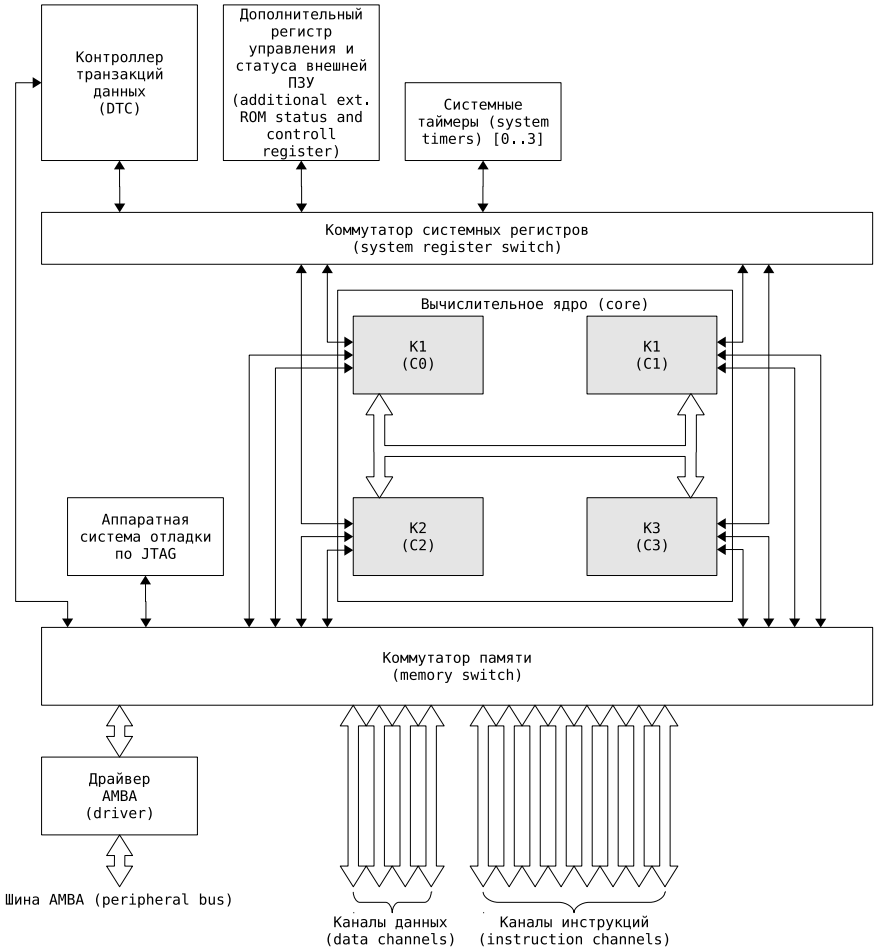

Figure 2: CPU block scheme

CPU consist of:

- 4 cellular processing unit;
- memory switch;
- system peripherals;
- system peripherals switch;
- debugging system.

Block scheme of CPU present on pic. 2.

3.1 Registers

All registers exept system are 64bit width. General purpose (GPR), index and control registers have copies in each cell. It's necessary for MCp042R100102 , because when we reconfigure CPU we need to see registers in each group of cells. System peripherals registers are situated in system peripherals like PLL, System timers etc. There are shared resouces.

#	Register	Description
GPR, index registers (copies in each cell)		
0-7	нет	GPR
8-15	нет	index registers
System peripherals registers		
16	PLLCR	control register of PLL
17	PLLSTR	status register of PLL
18	PLLTMCR	controll register of test block of PLL
19	MCTRL_CH0_CPR	switch ch 0 control
20	MCTRL_CH1_CPR	switch ch 1 control
21	MCTRL_CH2_CPR	switch ch 2 control
22	MCTRL_CH3_CPR	switch ch 3 control
23	ST0PRDR	system timer 0 period
24	ST1PRDR	system timer 1 period
25	ST2PRDR	system timer 2 period
26	ST3PRDR	system timer 3 period
27	ST0CR	system timer 0 control
28	ST1CR	system timer 1 control
29	ST2CR	system timer 2 control
30	ST3CR	system timer 3 control
31	*reserved*	
32	ST0VAL	curent value of system timer 0
33	ST1VAL	curent value of system timer 1
34	ST2VAL	curent value of system timer 2
35	ST3VAL	curent value of system timer 3
36-39	*reserved*	
40	DTC_CTRL	DTC control register
41	DTC_ST	DTC status register
42	DTC_IMASK0	DTC mask 0 register
43	DTC_IMASK1	DTC mask 1 register
44	DTC_DATA	DTC data register
45	DTC_S_ADDR	DTC source address
46	DTC_D_ADDR	DTC destination address
47	ROM_CTRL	external PROM status and control
Control registers (copies in each cell)		
48	PSW	processor state register
49	INTR	interrupts
50	MSKR	interrupts mask
51	ER	errors and exceptions
52	IRETADDR	return address
53	FORCE	force interrupts
54	*reserved*	
55	IHOOKADDR	interrupt handler address
56	INTNUM	current pending interrupt number
57	REGMOD	index registers modification
58	LASTADDR	address of previous paragraph
59	CURRENTADDR	address of curent paragraph
60	NEWADDR	address of next paragraph
61	LASTPSW	last value of PSW
62	*reserved*	
63	ICR	reconfiguration control

Table 2: MP registers

4 Memory organization

Program model of memory of MCp042P200102 / MCp042R100102 looks like solid massive of bytes. Address range: $[0, 2^{32}]$. Memory space is common for code and data, but there is some restriction for their placement.

4.1 Memory map

We can see some regions of the memory (pic.**??**):

- on chip memory;
- external RAM;
- external ROM;
- I/O space;
- image of 0x00000000-0x40000000 address range;
- peripheral bus.

On chip memory. This is SRAM.

There are 4 regions in 0x00000000-0x00040000 address range. There are 32 physical blocks 33×2^{11}, that provide access to neighbor addresses at the same time. Bit 33 used by debugging system only (ch.**??**). These regions are specialized for code. You can place here code and data. Only these regions provide unlimited number hardware brakepoints for programs.

Block number you can calculate with a next expression:

$$n = \left\lceil \frac{A_b}{2^{16}} \right\rceil + \left\lceil \frac{A_b \ mod \ 32}{8} \right\rceil, \ A_b = [0, 2^{18} - 1]$$

Address in bytes, result is a minimal integer number after division.

There are 4 regions in 0x00040000-0x00080000 address range. There are 16 physical blocks 64×2^{11}, that provide access to neighbor addresses at the same time. Don't use these regions for code that you whant to execute. Because it's possible that cells will request information from the same physical block at the same time and it will cause of delay. But you can store code here.

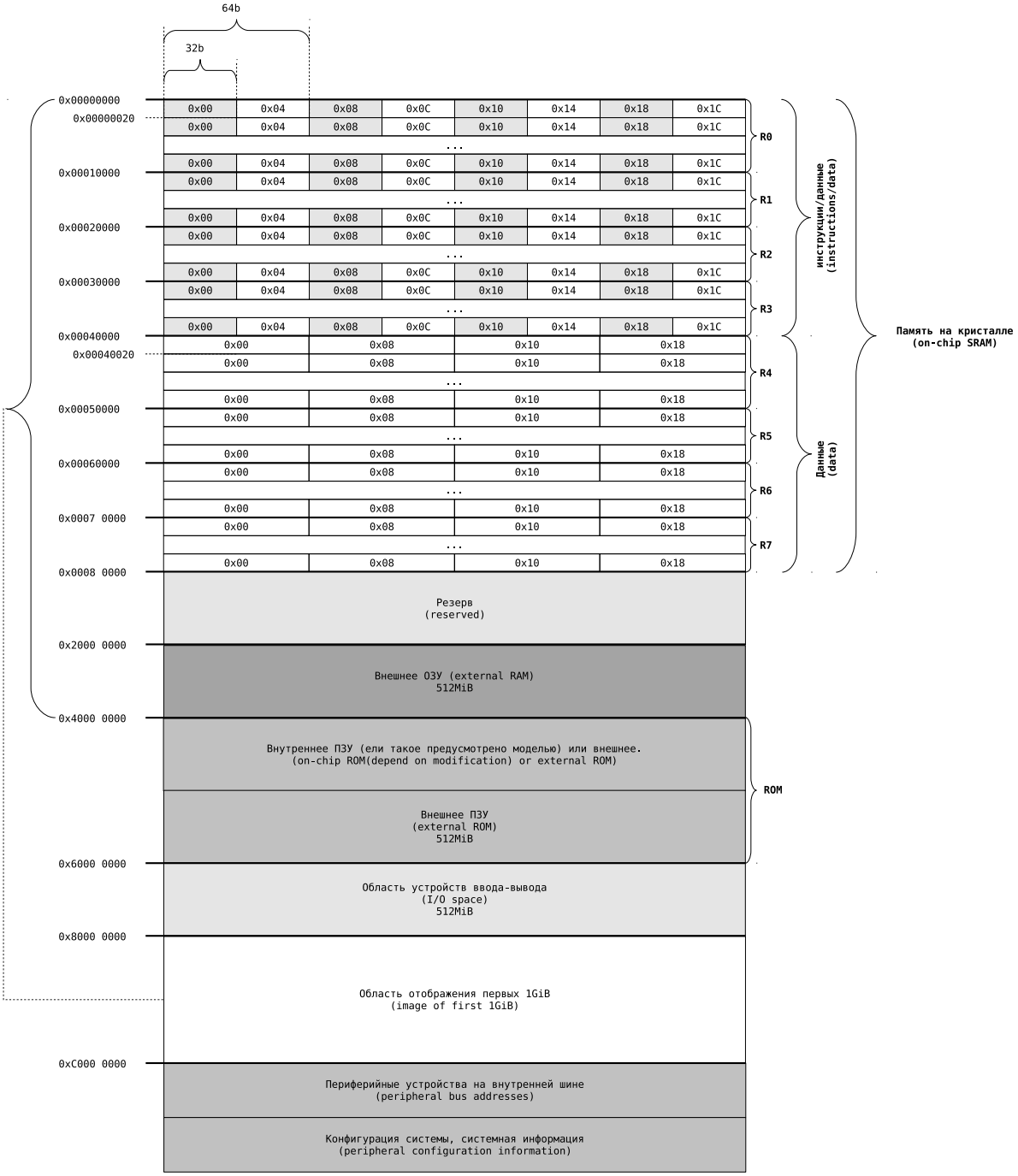

Figure 3: Memory map

Block number you can calculate with a next expression:

$$n = \left\lceil \frac{A_b}{2^{16}} \right\rceil + \left\lceil \frac{A_b \ mod \ 32}{4} \right\rceil, \ A_b = [2^{18}, 2^{19} - 1]$$

Address in bytes, result is a minimal integer number after division.

External RAM, external ROM, I/O space There is special interface that provide access to these regions of memory. (see ch.7). You can srore data and code, execute programm in these regions. But it's slower than you will use on-chip memory.

Image of 0x00000000-0x40000000 address range. This region used by peripheral DMA controller for access to first 1GiB address space. (ch.7).

Peripheral bus. Peripheral bus addresses.

4.2 Memory switch

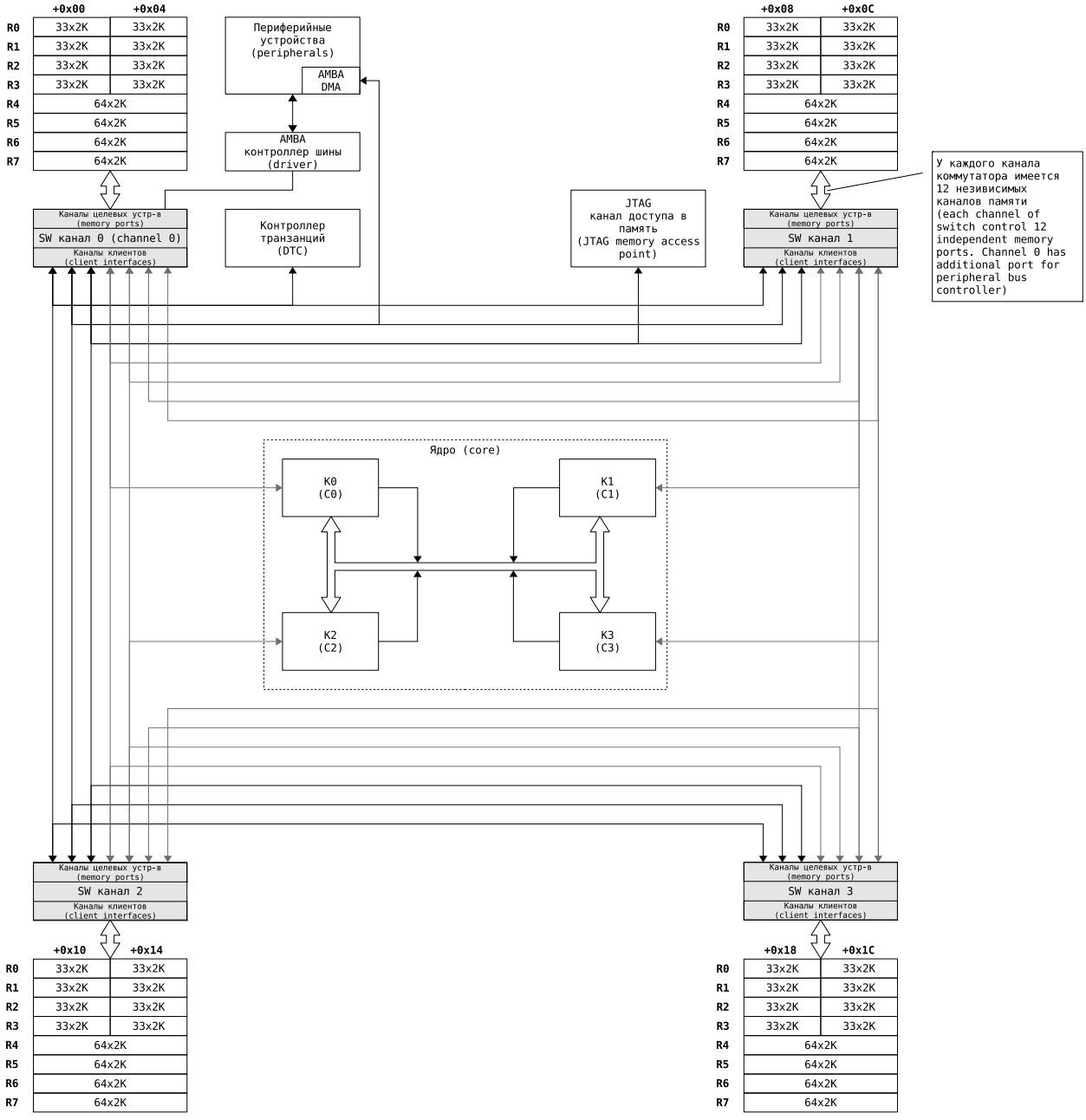

Figure 4: Switch structure

Switch has 4 independent channels, that controll dedicated memory blocks. Each channel has 15 clients access channels and 12 independent memory ports. Channel «0» has additional peripheral bus channel.

Switch channel number ypu can calculate with next expression:

$$n = \left\lceil \frac{A_b \ mod \ 32}{4} \right\rceil$$

Address in bytes, result is a minimal integer number after division.

Client sends request to all switch channel at the same time. If channel can service this request then it will reply with data after some time. If clients requested the same address then they will be served according priority.

#	Client ports	Description
0	C0_I_0	Instruction channels 32bit data width
1	C0_I_1	
2	C1_I_0	
3	C1_I_1	
4	C2_I_0	
5	C2_I_1	
6	C3_I_0	
7	C3_I_1	
8	C0_D	Data channels
9	C1_D	
10	C2_D	
11	C3_D	
12	DTC_INDEX	DTC channel
13	AMBA_DMA_INDEX	peripheral DMA channel
14	JTAG_MEM_INDEX	Debugging system channel

Table 3: Clients of switch channel

#	Memory ports	Description
0	R0_0	Region 0-3 of memory 32bit data width
1	R0_1	
2	R1_0	
3	R1_1	
4	R2_0	
5	R2_1	
6	R3_0	
7	R3_1	
8	R4	Region 4-7 of memory 64bit data width
9	R5	
10	R6	
11	R7	
12	AMBA	Address space after R7 region

Table 4: Memory ports of switch channel

Switch channel has priority scheme. Client channels devided by 4 types:

- instruction channel;
- data channel;

- AMBA(peripheral) channel;
- DTC channel.

4.3 Data transaction controller (DTC)

5 Clock system

6 Debugging system

7 Peripheral devices

7.1 Input-output port (GPIO)

7.1.1 Brief characteristics

- MP contains 2 32-bit ports, 1 24-bit ports and 1 16-bit;
- Each port bit can be individually configured at inputs or outputs and can optionally generate interrupts;
- interrupt request can be formed in accordance with signal level or edge (front/back);
- port inputs-outputs can be switched to alternative function;

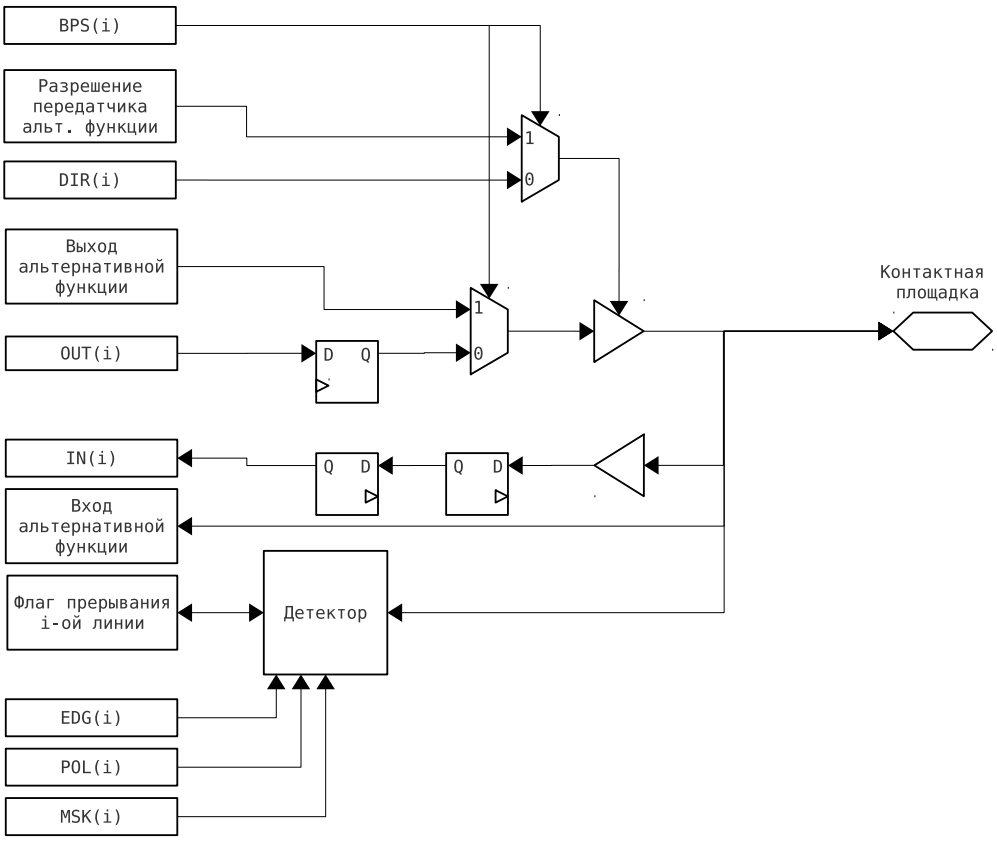

Figure 5: Block diagramm of i line GPIO

7.1.2 GPIO functioning

Input-output ports are implemented as bi-directional buffers with programmable output definition. Each buffer's input is synchronized by means of two series-connected flip-flops

to prevent the possibility of metastability. The synchronized value may be read from data receiving register (GPIOxIN) of input-output port. The output definition is controlled by transmission permit register (GPIOxDIR). Logical unit ('1') in the given bit of input-output port configures its corresponding line to the output. The output value is taken from the register of the transmitted data (GPIOxOUT) of input-output port.

Each input-output port line can be configured to form an interrupt. Formation of interrupt is controlled by three registers: interrupt mask register (GPIOxMSK), register of interrupt event settings, signal polarity register (GPIOxPOL), and signal component register (GPIOxEDG). To enable the interrupt corresponding interrupt mask bit has to be set (one). If signal component register is reset (zero), an interrupt is generated by signal level. If signal polarity register is reset (zero), an interrupt occurs in case of active low level signal, if signal polarity register is set (one), an interrupt occurs in case of active high level signal. If signal component register is set (one), an interrupt is generated by signal edge. If signal polarity register is reset (zero), an interrupt occurs in case of front signal edge. If signal polarity register is set (one), an interrupt occurs in case of back signal edge.

Each input-output port can be shared for other signal types that perform alternate functions. To enable the task of some port line's alternate functions task corresponding bit (one) is to be set in alternative functions permit register (GPIOxBPS) A description of all inputs-outputs in MP find in Section **??**

7.1.3 Register description

General purpose input-output ports:

GPIOA: basic address - 0xC01F 0000; port width - 32 bits.
GPIOB: basic address - 0xC01F 0100; port width - 32 bits.
GPIOC: basic address - 0xC01F 0200; port width - 24 bits.
GPIOD: basic address - 0xC01F 0300; port width - 16 bits.

To get the real address of the register the register address's offset should be added to base (initial) address on the bus.

Bits from 0 to 31 are significant for ports A, B , bits from 0 to 23 are significant for port C, bits from 0 to 15 are significant for port D. Reading the bits from 24 to 31 for port C and from 16 to 31 for the port D will give a zero result and the record will be ineffectual.

Регистр	Address displacement	Access	Description
GPIOxIN	00h	*R*	Receiving data register.
GPIOxOUT	04h	*RW*	Transmitted data register.
GPIOxDIR	08h	*RW*	Transmission enabling register.
GPIOxMSK	0Ch	*RW*	Mask interrupt register.
GPIOxPOL	10h	*RW*	Event interrupt setting register, signal polarity.
GPIOxEDG	14h	*RW*	Event interrupt setting register, signal component.
GPIOxBPS	18h	*RW*	Alternative functions enabling register

GPIOxIN	Receiving data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Receiving data register. Each regiter bit corresponds with each port line.

GPIOxOUT	Receiving data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Receiving data register. Each register bit corresponds with each port line.

GPIOxDIR	Transmission enabling register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Transmission enabling register. Each register bit corresponds with each port line. If any register bit is set in '1', then in relevant port line data trasmission is allowed, if bit is set in '0'- transmission is forbidden.

GPIOxMSK	Interrupt mask register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Interrupt mask register. Each register bit corresponds with each port line. If any register bit is set in '1', then in relevant port line data event interrupt is allowed, if bit is set in '0'- event interrupt is forbidden.

GPIOxPOL	Event interrupt setting register, signal polarity																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Event interrupt setting register, signal polarity. Each register bit corresponds with each port line. If any register bit is set in '1', then relevant port line forms interruption under leading edge/high level, if bit is set in '0'- forms interrution under trailing edge/low level. Edge or level choice depends on settings of relevant bit in register GPIOxEDG.

GPIOxEDG	Event interrupt setting register, signal component																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Event interrupt setting register, signal component. Each register bit corresponds with each port line. If any register bit is set in '1', then relevant port line forms interruption under edge, if bit is set in '0'- it forms interruption under level.

GPIOxBPS	Alternative functions enabling register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Description	DATA																															

0-31 DATA Alternative functions enabling register. Each register bit corresponds with each port line. If any register bit is set in '1', then relevant port line is allowed to imlement alternative function, if bit is set in '0'- imlementation of alternative function is forbidden.

7.2 External memory bus controller (MCTRL0)

7.2.1 Short description

- поддерживает работу с памятью типа: PROM, DRAM, SRAM;
- автоматическое регенерация SDRAM;
- внешняя шина данных до 32 бит;
- работа с устройствами ввода/вывода (I/O);

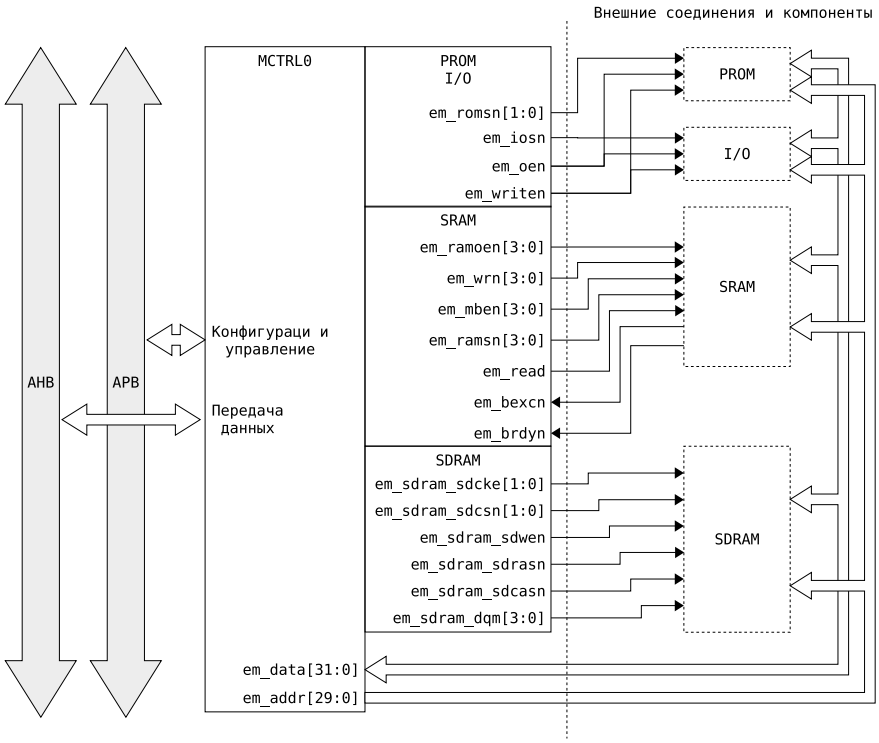

Figure 6: Контроллер шины внешней памяти

Контроллер может управлять 2-мя банками PROM, одним банком I/O, пятью банками SRAM и двумя банками SDRAM.

7.2.2 Интерфейс PROM, I/O

Ниже приведены диаграммы работы интерфейса.

Диаграммы чтения:

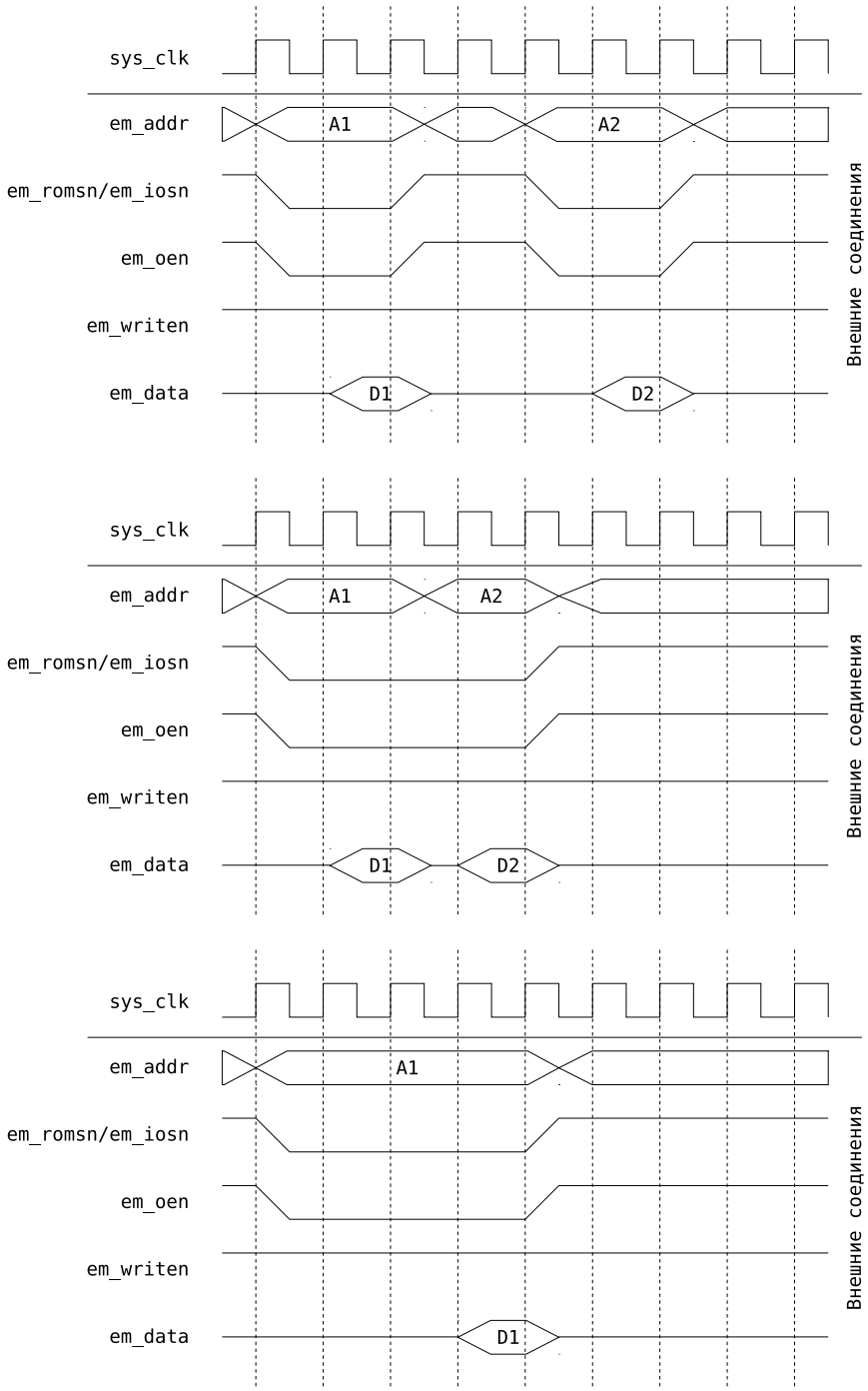

Figure 7: Диаграммы чтения PORM, I/O (независимые чтения, непрерывное чтение и чтение с 2-мя циклами ожидания)

Диграммы записи:

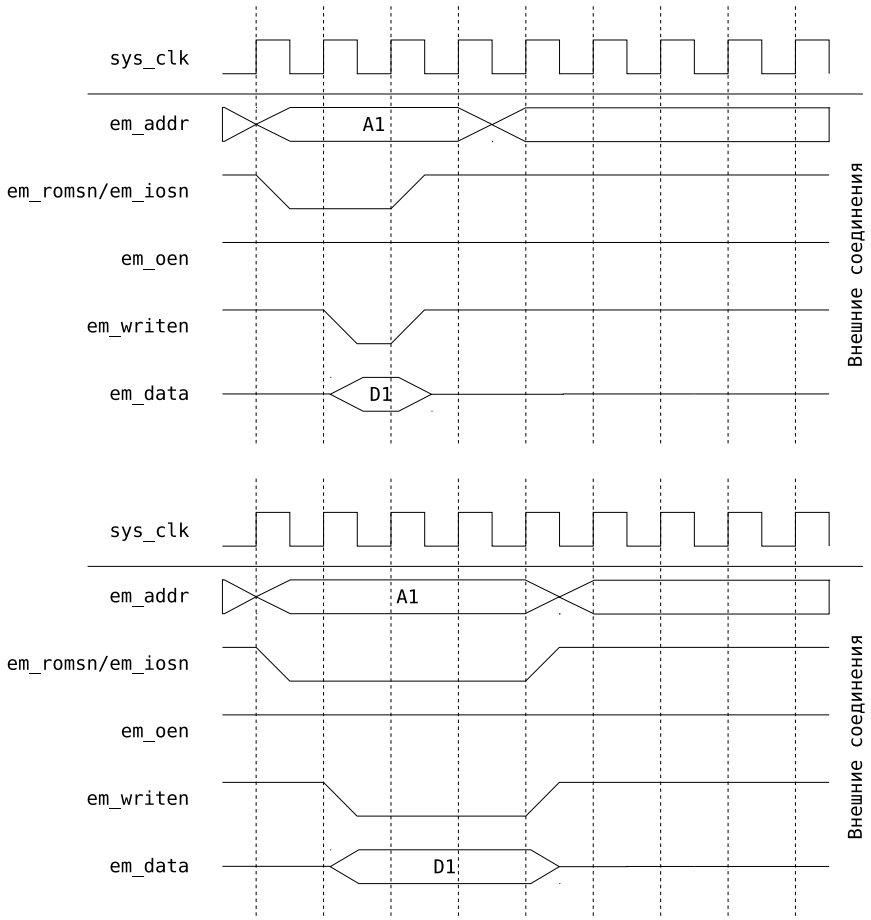

Figure 8: Диаграммы записи PORM, I/O (без циклов ожидания и с 2-мя циклами ожидания)

7.2.3 Интерфейс SDRAM

Ниже приведены диаграммы работы интерфейса.

7.3 UART(UARTx) interface

7.3.1 Brief characteristics

- full duplex mode;
- separate FIFO buffers 32 in depth for receive-transmit;
- data word - 8 bit, fixed;
- adjustable parity check;
- 1 stop bit;
- built-in data flow check (CTS, RTS);

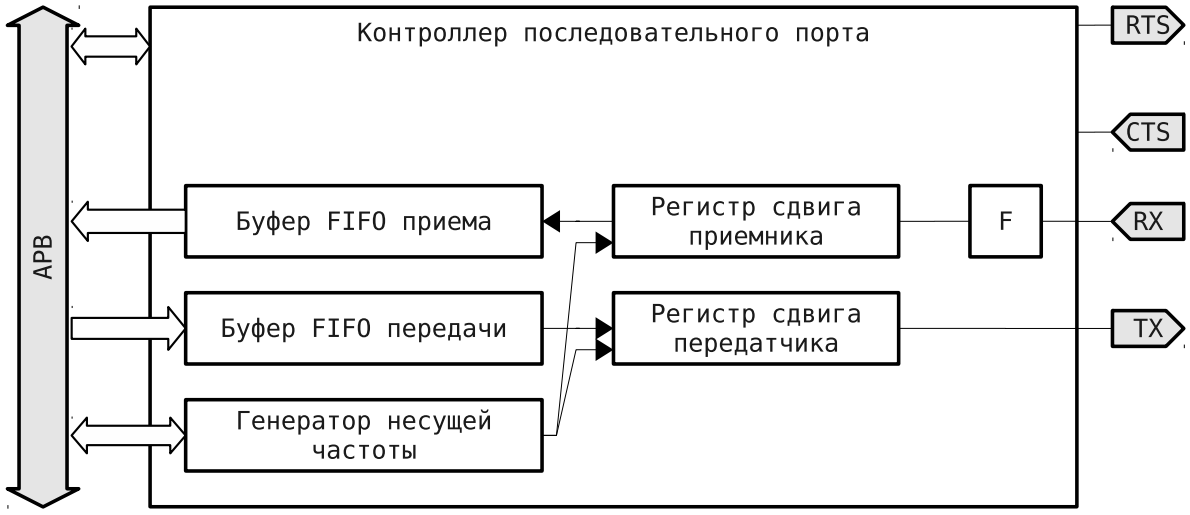

Figure 9: Block diagram UARTx

Fig. 9 is a block diagram of serial UART interface controller.

UART controller automatically generates a parity check bit during transmission and generates parity check when receiving data. Parity check mode turning on/off and setting is implemented in UARTxCR.

7.3.2 Data transmission

Transmitter turning and enabling is implemented by UARTxCR(TE). Data for transmission is written in FIFO transmit buffer. Only FIFO input is available for user. Reference to it is

implemented through UARTxDATA. Buffer width - 8 bit, depth - 32.

From the FIFO buffer data is sent to the shift register from which they bitwise (LSB) appear at the output TX. Start bit, stop bit and parity check bit (if enabled) are generated automatically. Fig. 10 shows possible transmitted data format, for given controller.

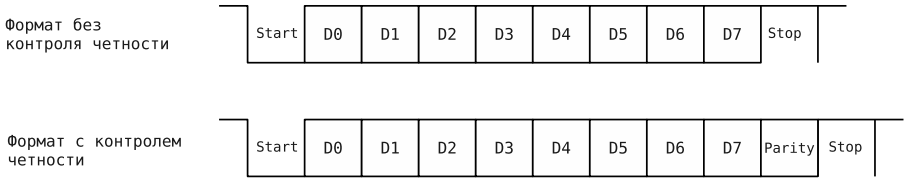

Figure 10: Data burst format

After transmission completion of the last data word from the buffer FIFO, after stop bit formation TX output is set into logic state '1' empty shift register sign is set to UARTxST (TS). After this bit is automatically set to a logic '0 ' once the data appear in FIFO buffer. If the FIFO is empty, bit UARTxST(TE) is set. Bits UARTxST (TF) signalize if buffer is full-the buffer is full and UARTxST (TH) - less than half of the buffer is full. UARTxST(TCNT) spinner is aimed to control if FIFO buffer is full. UARTxCR(TF) bit controls interrupt requests under buffer events.

When trasmitter operation is forbidden, transmission stops immediately, current data word transmission from shift register is being interrupted as well.

If in-built flow control is enabled, the data from the shift register will be transmitted only if CTS in logic state '0'. If to set signal to logic '1'during the transmission, then the transmission will stop, and is resumed only when CTS is brought back to logic '0'.

7.3.3 Data receipt

Bit UARTxCR(RE) switches and enables receiver operation. The received data are being written in FIFO receive buffer. Only FIFO output is available for user in general mode. Addressing it is implemented through UARTxDATA. Buffer width - 8 bit, depth - 32. The received signal passes the digital lowpass filter.

The receiver monitors input signal state and in case of signal transfer from logical state '1 'to state '0' starts receiving data. Through $\frac{T_{UART}}{2}$ input signal state is fixed, where T_{UART} - the period of one data word bit. If start bit is not fixed, the receiver will be brought back to sleep mode. If start bit is received, then the remaining data bit words and overhead bits will be received. When last bit is received, data is placed in FIFO buffer, UARTxST(DR) bit is

set. Receipt error bits are set in UARTxST(DR), if such are fixed Error bits are cleansed only by software.

In case when received data are placed in shift register, and FIFO buffer is full, and start bit is fixed at receiver input, shift register data will be lost. Meanwhile UARTxST(OV) bit will be set.

Bits UARTxST(RF) signalize that buffer is full, UARTxST(RH) signalize that less than half of the buffer is full. To control filling of FIFO buffer there is UARTxST(RCNT) spinner. Interrupt request definition under buffer events is controlled by UARTxCR(RF) bit.

If data flow built-in control is enabled and FIFO buffer is full, then RTS transfers into logic state '1'. Once at least one data word is read from the buffer, RTS automatically transfers into logic state '0'.

7.3.4 Transmission speed settings

To set data transmission speed there is a system frequency predivision, which division ratio is set in the register UARTxBDR (formula shown below).

$$BRDIV = \frac{F_{sys}}{8 \cdot F_{UART} - 1};$$

7.3.5 Self-test modes

In self-test modes all controller outputs are transfered into inactive mode.

7.3.5.1 Self-test mode on interface line level In this mode, UART transmitter output commutes with receiver input inside the chip, and CTS signal commutes with RTS. Enabling this mode is implemented by means of UARTxCR(LB) bit setting.

7.3.5.2 Self-test mode on data level This mode allows recording in FIFO receive buffer and reading from FIFO transmit buffer. Reading and writing are carried out through UARTxFIFODBG register. Enabling this mode is done by setting UARTxCR(LB) bit.

7.3.6 Interrupt formation

Interrupt requests are formed in the following cases:

From transmitter's shift register:

- transmitter operation is enabled: UARTxCR(TE) bit is set;
- transmitter interruptions are enabled: UARTxCR(TI) bit is set.

From FIFO transmit buffer:

- transmitter operation is enabled: UARTxCR(TE) bit is set;
- transmitter interruptions are enabled: UARTxCR(TF) bit is set;

From receiver's shift register:

- transmitter operation is enabled: UARTxCR(RE) bit is set;
- transmitter interruptions are enabled: UARTxCR(RI) bit is set.

From FIFO transmit buffer:

- transmitter operation is enabled: UARTxCR(RE) bit is set;
- transmitter interruptions are enabled: UARTxCR(RF) bit is set;

7.3.6.1 Receiver pending interrupt mode The mode is enabled by setting UARTxCR(DI) bit. Interrupt from the receiver is formed only in the case of formation of a pause after the last data word receipt. Pause time is equal to 4.5 data word receipt. If the interrupt is enabled from FIFO receive buffer, then an interrupt from shift register will be cleansed. Only buffer interrupts will be active.

Note: The definition of this mode does not affect the formation of the request interruption when receiving transaction completion sign.

7.3.7 Registers' description

Base address UART0 - 0xC000 0100
Base address UART1 - 0xC000 0200
Base address UART2 - 0xC010 0100
Base address UART2 - 0xC010 0200

In order to receive real register address add register address diplacement to base (initial) address on the bus.

Register	Address displacement	Access	Description
UARTxDATA	00h	RW	Data register (FIFO)
UARTxST	04h	R	State register
UARTxCR	08h	RW	Control register
UARTxDBR	0ch	RW	Clock frequency division ratio register

UARTxDATA	Data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	—																															
Description	—																								DATA							

8-31 — *reserved*
0-7 DATA Data register. During writing is an input of 32-byte buffer of FIFO transmitter. During reading is an input of 32-byte buffer of FIFO receiver.

UARTxST	State register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	—						—						—									—	—	—	—	—	—	—	—	—	—	—
Description	RCNT						TCNT						*reserved*									RF	TF	RH	TH	FE	PE	OV	BR	TE	TS	DR

26-31	RCNT	FIFO receiver data spinner
20-25	TCNT	FIFO transmitter data spinner
11-19	—	*reserved*
10	RF	FIFO receiver is full
9	TF	FIFO transmitter is full
8	RH	FIFO receiver is half and more full
7	TH	FIFO transmitter is half and more full
6	FE	Received data format error
5	PE	Received data parity check register
4	OV	One or more received data character is lost due to overflow
3	BR	Special exchnge completion character received (BREAK)
2	TE	Transmitter FIFO is empty
1	TS	Transmitter shift register is empty
0	DR	New characters are fixed in receiver register

UARTxCR	UART control register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	—	—																—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Description	FA	*reserved*																SI	DI	BI		RF	TF		LB	FL	PE	PS	TI	RI	TE	RE

31	FA	FIFO receiver and transmitter usage enabling ('0'- forbidden, '1'- permitted)
15-30	—	*reserved*
14	SI	Interrupt enabling under empty transmitter shift register ('0'– forbidden, '1'– permitted)
13	DI	Receiver pending interrupt (4 characters + 4 bits and no new character) ('0'– forbidden, '1'– permitted)
12	BI	Interrupt enabling under BREAK character receipt ('0'– forbidden, '1'– permitted)
11	—	*reserved*
10	RF	Interrupt enabling under FIFO receiver ('0'– forbidden, '1'– permitted)
9	TF	Interrupt enabling under FIFO transmitter ('0'– forbidden, '1'– permitted)
8	—	*reserved*
7	LB	Internal outside loop switch for self test ('0'– switched off, '1'– switched on)
6	FL	Data flow control enabling (CTS/RTS) ('0'– forbidden, '1'– permitted)
5	PE	Parity check enabling ('0'– forbidden, '1'– permitted)
4	PS	Parity check type selection ('0'– for parity, '1'– for oddness)
3	TI	Transmitter interrupt enabling under character transmit completion ('0'– forbidden, '1'– permitted)
2	RI	Receiver interrupt enabling under character receipt ('0'– forbidden, '1'– permitted)
1	TE	Transmitter operation enabling ('0'– forbidden, '1'– permitted)
0	RE	Transmitter operation enabling ('0'– forbidden, '1'– permitted)

UARTxBDR	Clock frequency division ratio register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	—																				—											
Description	*reserved*																				BRDIV											

12-31	—	*reserved*
0-11	BRDIV	System frequency division ratio for required data exchange speed formation BRDIV = Fsys/BR*8, где Fsys в Гц, а BR в бит/с

7.4 SPI interface (SPIx)

7.4.1 General characteristics

- able to operate in «master» or «slave» modes;
- all SPI modes are supported, as well as three-wire mode, where bidirectional data line is used;
- adaptive data word length;
- separate receive-transmit FIFO buffers 32 in depth;
- selector for 3 slave devices;
- user-set data word format — LSB or MSB;
- user-set CPOL polarity and clock signal CPHA phase;
- user-set rate of data exchange.

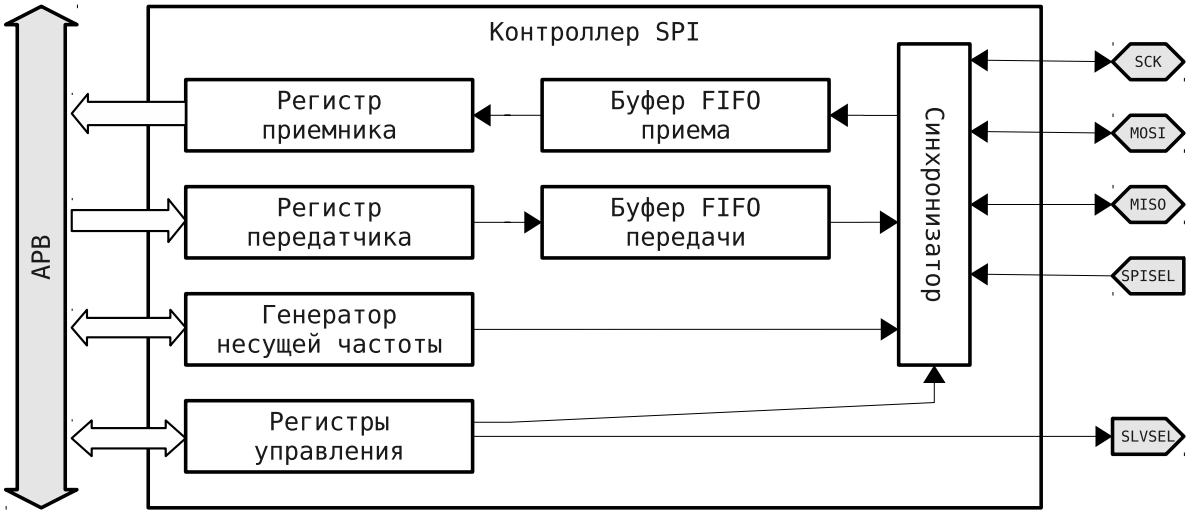

Figure 11: Block diagram SPIx

The SPI interface is a full-duplex. The transmission starts as soon as «master» translates SLVSEL signal of corresponding «slave» driven to the active state, SCK as derived from the inactive state.

The data is transmitted by «master» through MOSI line, received through MISO.

System with one «master» and one «slave» is allowed to be not controlled by means of SLVSEL signal, it can always be activated.

In the system having several «masters», each of them monitors SPISEL signal to avoid conflicts with other «master». If active level occured in SPISEL input, then receiving «master» switches off.

In the process of receiving or transmitting, data change when the state of SCK changes. Initial state and SCK active edge values determine SPI operation mode. Fig. 12 shows diagrams with SPI operation mode in the process of transmission 0x55 in MSB mode. It should be noted that the data should be available in the transmit buffer to the first SCK state change.

When operating in «slave» mode data transmission through MISO line will be delayed so as to synchronize the transmitter. See the description in Section 7.4.6.

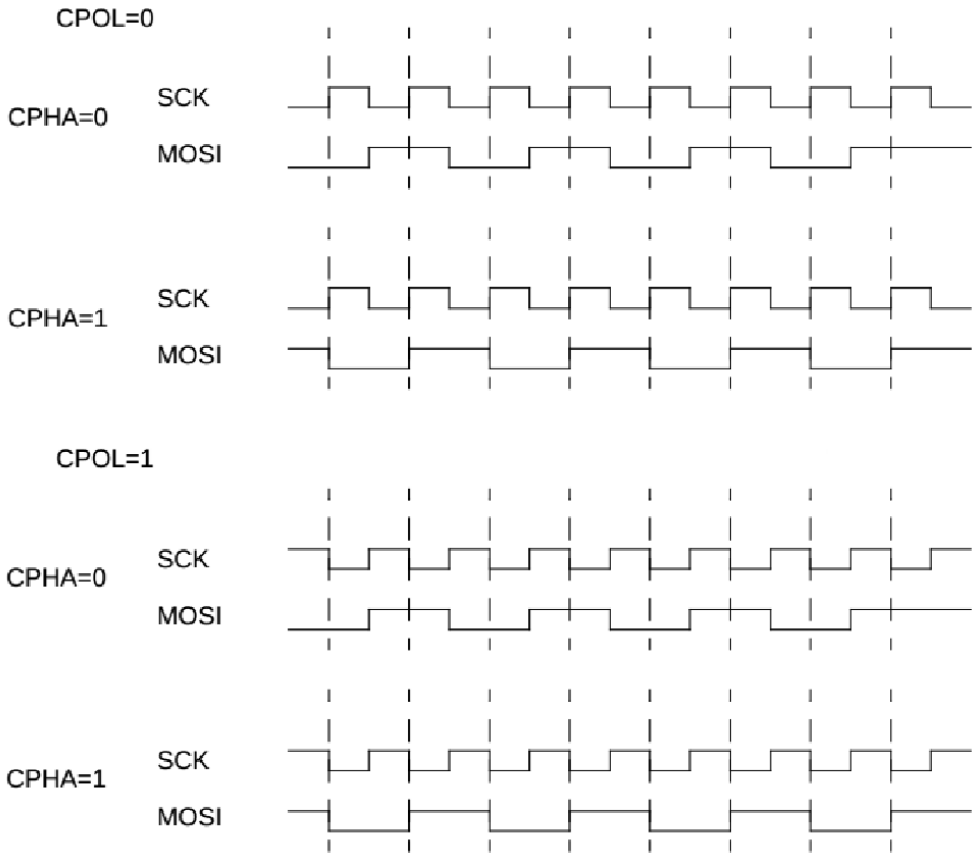

Figure 12: SPI operation mode

7.4.2 3-wire operation

Interface controller can be configured to operate in three-wire mode (see SPIxCR(TWEN) register). Operation will be implemented in half-duplex mode. This mode uses one bidirectional data receive-transmit line instead of two unidirectional for receive-transmit.

MOSI line is used for data exchange , MISO in this mode is not activated. The direction of data exchange is selected in SPIxCR(TTO) register.

Changing the data in receive-transmit is implemented in accordance with clock signal polarity and phase as in four-wire operation mode.

7.4.3 Data receive and transmit

Interface Controller has separate FIFO registers and buffers for receiving and transmitting. FIFO buffer capacity - 32 bit, depth - 32. Data word capacity is defined in SPIxCR(TWEN). If transmit buffer is not full, then bit is set in SPIxST(NF) register, data can be sent into buffer. If the receive buffer contains at least one fully received word, then bit is set in SPIxST(NE) register. If there is a situation that more than 33 or data words are received then bit is set in SPIxST(OV) register. When operating in «slave» mode interface controller may detect the situation when it was chosen by «master» (SPISEL took a logic '0' value) and there is no data in transmission buffer. In this case, bit is set in SPIxST(UN) register.

7.4.4 SCK clock signal

Interface controller may generate SCK signal only in «master». SCK generator parameters are set in SPIxCR register.

$$F_{SCK} = \frac{F_{sys}}{(4 - (2 \cdot FACT))(PM + 1))}, \quad DIV16 = 0;$$

$$F_{SCK} = \frac{F_{sys}}{16(4 - (2 \cdot FACT))(PM + 1))}, \quad DIV16 = 1;$$

7.4.5 Operation in «master» mode

In this mode, as soon as data are available in transmit FIFO buffer, they are immediately transferred. If the data are transmitted and the transmit buffer is empty, SCK is not received.

If, during operation in this mode, SPISEL signal takes logic '0' value interface controller stops data transmission and sets bit in SPIxST(MME) register. Enable bit in the «master» mode in SPIxCR(MS) register is reset.

Interface controller behaviour in case SPISEL signal change is determined in SPIxCR(IGSEL) register.

7.4.6 Operation in «slave» mode

In this mode, interface controller does not control interface lines until SPISEL signal will not accept logic value '0 ' from «master». Once this has occurred, MISO is configured as an output and this output has the status relevant to the first bit of transmission FIFO data buffer. If interface controller operates in three-wire mode, word receive completion is expected on MOSI line and then MOSI is configured as an output. If transmit buffer is empty, the transmission line has logic status '1'.

SCK frequency in this mode must satisfy the following condition:

$$F_{SCK} \leq \frac{F_{sys}}{8};$$

Interface controller transmitter is synchronized from external SCK, so that new data on MISO line will appear only after 2 periods F_{sys} after SCK front.

Interface controller can also be used for internal SCK filter, this is controlled by SPIxCR(PM) register. SPIxCR(PM) determines which time, expressed in F_{sys} periods SCK signal must be stable. With each PM increase on 1 next data putout delay on the MISO line is increased by 2 F_{sys} periods. It is also necessary to increase SCK period to the same value as calculated from the conditions mentioned above.

7.4.7 Descriptiom of registers

Base address SPI0 - 0xC010 2000
Base address SPI1 - 0xC010 2100
Base address SPI2 - 0xC010 2200.

In order to receive real register address add register address displacement to base (initial) address on the bus.

Register	Address displacement	Access	Description
SPIxCFG	00h	RW	Configuration setting register
SPIxCR	20h	RW	Control register
SPIxST	24h	RW	State register
SPIxMSK	28h	RW	Mask register
SPIxCMD	2Ch	RW	Command register
SPIxTX	30h	W	Transmit data register
SPIxRX	34h	R	Receive data register
SPIxSS	38h	RW	Slave device selection register

SPIxCFG	Configuration register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0x3								0				0	0		0	0x20								0							
Description	SSSZ								MAXWLEN				TWEN	-		SSEN	FDEPTH								—							

24-31	SSSZ	slave device selection line amount
20-23	MAXWLEN	maximum data word supported length (0-32)
19	TWEN	three-wire mode enabling ('1' - permitted, '0' - forbidden)
17-18	—	*reserved*
16	SSEN	slave device selection signal enabling ('1' - permitted, '0' - forbidden)
8-15	FDEPTH	Depth FIFO RX, TX
0-7	—	*reserved*

SPIxCR	Control register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0	0	0	0	0	0	0	0				0				0	0	0	0	0					0			0	0		
Description	-	LOOP	CPOL	CPHA	DIV16	REV	MS	EN	LEN				PM				TWEN	-	FACT	-	CG					—			TTO	—		

31	—	*reserved*
30	LOOP	self-testing mode ('1' - permitted, '0' - forbidden)
29	CPOL	SCK state in wait mode ('1' - logic 1 , '0' - logic 0)
28	CPHA	clock phase setting ('1' - data will be read on the second SCK state transition, '0' - data will be read on the first SCK state transition)
27	DIV16	division by 16 enabling(only in master mode) ('1' - permitted, '0' - forbidden)
26	REV	transit direction ('1' - MSB, '0' - LSB)
25	MS	mode selection ('1' - master, '0' - slave)
24	EN	operation permit ('1' - permitted, '0' - forbidden)
20-23	LEN	data word length 0x0 - word length 32 bit 0x1-0x2 - nonaccepted value 0x3-0xf - 4-16 bit accordingly
16-19	PM	predivision mode (only in master mode): if DIV16 - 0: $F_{sck} = \frac{F_{sys}}{(4-2 \cdot FACT \cdot (PM-1))}$ if DIV16 - 1: $F_{sck} = \frac{F_{sys}}{(16 \cdot (4-2 \cdot FACT \cdot (PM-1)))}$

15	TWEN	three-wire mode ('1' - permitted, '0' - forbidden)
14	—	*reserved*
13	FACT	frequency predivision mode (1 - compatibility c MCP83xx):
12	—	*reserved*
7-11	CG	SCK signal receiving turning-off after each word transition data on N priods(only in master mode)
4-6	—	*reserved*
3	TTO	transmission procedure when operating through three-wire line ('1' - slave delivers first, '0' - master delivers first)
0-2	—	*reserved*

SPIxST — **State register**

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0																0	0	0	0	0	0	0	0							
Description	TIP	—																LT	-	OV	UN	MME	NE	NF	—							

For all register bits: ('1' - attribute presence, '0' - attribute absence)

31	TIP	data word is being transmitted
15-30	—	*reserved*
14	LT	last data word transmitted: transmission buffer is empty or in SPIxCMD LST bit is written (bit is being cleansed by record '1')
13	—	*reserved*
12	OV	receiver buffer is empty , new data are being ignored (bit is being cleansed by record '1')
11	UN	data for transition are absent in the buffer, under master request (only in slave mode)
10	MME	error when operating in the system with few masters (occurs when SPISEL signal appears in master mode)
9	NE	receiver buffer includes data
8	NF	transmit buffer has free space
0-7	—	*reserved*

SPIxMSK — **Mask register**

Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0	0																0	0	0	0	0	0	0	0							
Description	TIPE	—																LTE	-	OVE	UNE	MMEE	NEE	NFE	—							

For all register bits: ('1' - interruption permitted , '0' - interruption forbidden)

31	TIPE	data word is transmitted
15-30	—	*reserved*
14	LTE	last data word is transmitted: transmit buffer is empty
13	—	*reserved*
12	OVE	receiver buffer if full, new data are ignored
11	UNE	data for transition are absent in buffer, under master's request (only in slave mode)
10	MMEE	error when operation in system with few masters
9	NEE	receive buffer includes data
8	NFE	transmit buffer has free space
0-7	—	*reserved*

SPIxCMD	Command register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0									0	0																					
Description	—									LST	—																					

23-31 — *reserved*

22 LST data for transmission, bits' width and sequence order are determined in SPIxCR. Writing into register is possible only under SPIxST(NF) = '1'

0-21 — *reserved*

SPIxTX	Transmitted data register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	TDATA																															

0-31 TDATA data for transmission, bits' width and sequence order are determined in SPIxCR register. Data are effective, if SPIxST(NF) = '1'. for REV = '0' SPIxCR – LSB is placed in bit 0, for REV = '1' SPIxCR – MSB is placed in bit 31 Under 8-bit word 0xAB byte will receive the following placement for transmission: for REV = '0' - 0x000000AB, for REV = '1' - 0xAB000000

SPIxRX	Received data register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	RDATA																															

0-31 RDATA received data, bits' width and sequence order are determined in SPIxCR register. Data are effective, if NE = '1' in SPIxST register. for REV = '0' SPIxCR – MSB is placed in bit 15 (under word width 4-16 bit), for REV = '1' SPIxCR – LSB is placed in bit 16 (under word width 4-16 bit). Under 8-bit word 0xAB byte will receive the following placement for transmission: for REV = '0' - 0x0000AB00, for REV = '1' - 0x00AB0000

SPIxSS	Slave device selection register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																													0		
Description	—																													SLVSEL		

3-31 — *reserved*

0-2 SLVSEL slave device number, with which data exchange is to be implemented

7.5 Interface I^2C «master» (I2C0)

7.5.1 Brief characteristics

- works in mode «master»;
- compatible with Philips I^2Cstandard;
- supports 7-bit and 10-bit addressing;
- rate of exchange - 100Kb/s и 400Kb/s;
- external supporting resistors setting is needed on lines SCA and SDA.

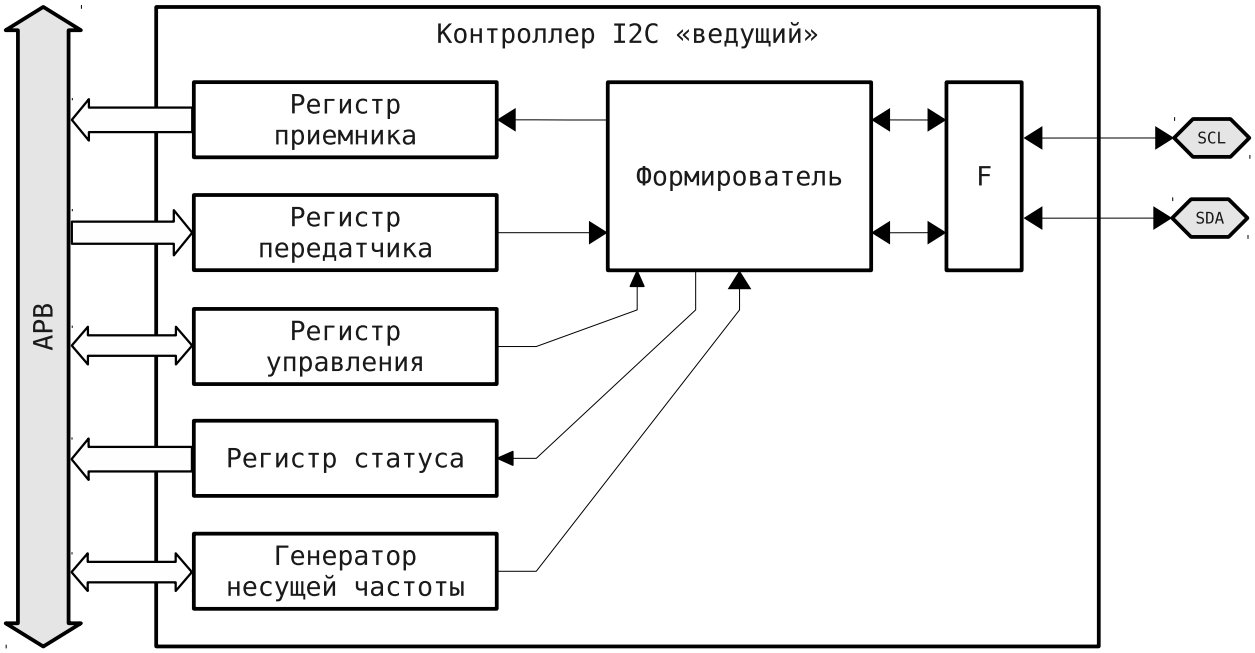

Figure 13: block diagram I2C0 («master»)

In MP I2C0 works only in «master» operation mode.

I^2Csimple two-wire serial interface wsuitable for operation of several «master» on the same physical line. Interface provides collision detection and arbitration. itc has two physical lines SDA (serial data line) and SCL (serial clock line).

Fig. 13 is a block diagram of the interface controller described. Digital low-pass filter is set at external interface lines input.

7.5.2 General description of receive-transmit protocol

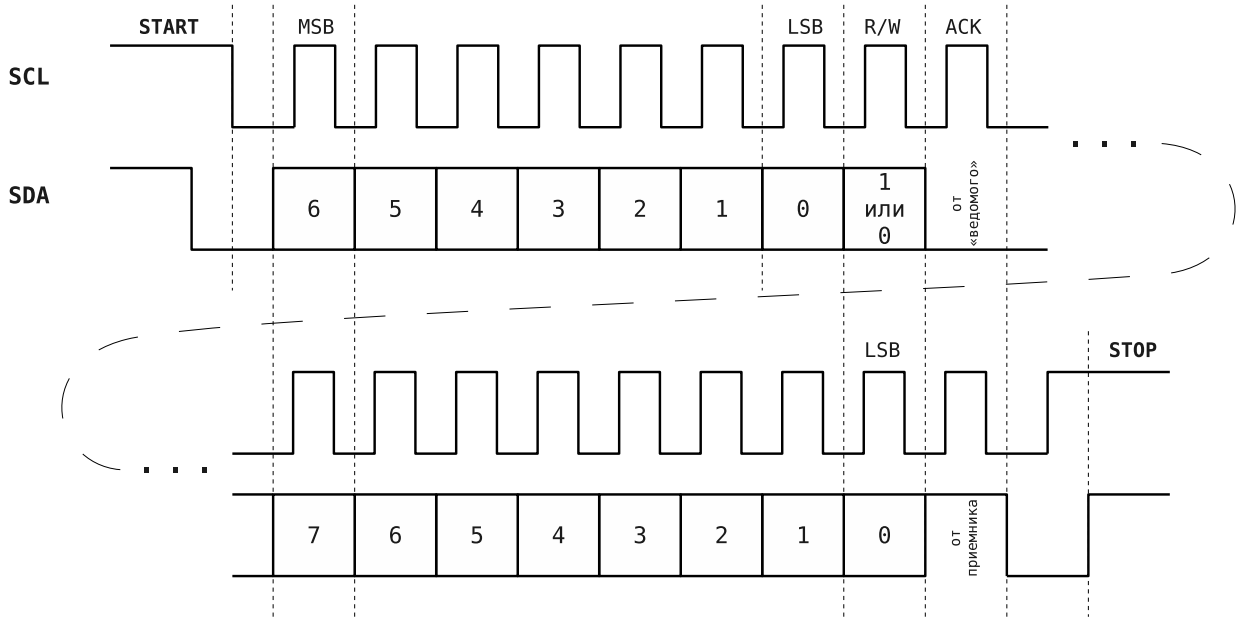

Figure 14: Transaction at the bus I^2C

Data receive-transmit is implemented bit-to-bit.

Start of transaction on the bus I^2C is determined by the state « START » on the lines SDA, SLC: transition from the state '1' in the state '0' in state SCL - '1'.

Closing the transaction is determined by the state « STOP » on the lines SDA, SLC: transition from the state of SDA line '0' in state '1' in the state of SCL - '1'.

States « START », « STOP » are formed only by «master» of the bus. After the formation of the state « START » by «master» of the bus, the bus is considered reserved and released only after the formation of «master» state «STOP». The time between « STOP » and « START » is determined by I2C standard and depends on its current operation speed.

Fig. 16 shows examples of lines SDA, SLC state during transactions. Bus «master» generates « START » state and sends 7-bit address of «slave» device. Bit $R/\overline{W}$ follows the address, it determines the direction of data transmission ('1' reading from «slave» device '0' writing into «slave» device). After address and $R/\overline{W}$ bit transmission, «masterg» bus releases the line SDA, and «slave» should lead the SDA line into state '0'. If this does not happen, it is believed that no signal is received ACK (acknowledgment) from «slave». Bus «master» can form a state « STOP » and repeat the transaction with this «slave» or take other actions incorporated in the algorithm for its work.

If signal ACK is received from «slave», then the data transfer begins, which direction was determined by $R/\overline{W}$ bit. Data can be transmitted until the receiver responds ACK for each transmitted data byte. That is, after each data byte transfer, the transmitter releases line SDA for a period SLC, so that receiver could report whether data byte is received or not, or whether it is ready or not to receive the following one. After NAK respond (during await process ACK command is being given '1') «master» forms «STOP» state. «master» can also abort a transaction to form the state «STOP» state.

7.5.3 Carrier frequency generation

Controller I^2C generates two frequencies: for external clock on SCL line and for internal block clock five times exceeding the rate on SCL line. To calculate controller clock frequency predivision division ratio value(I2CxPSC (PSC)) the following formula is used:

$$PSC = \frac{F_{sys}}{5 \cdot F_{SCL}} - 1$$

Predivision division ratio can be changed only when the controller is switched off I^2C (bit I2CxCR(EN)).

The minimum recommended value of the coefficient is equal to 3, so that synchronization protocol accesses are followed. It also imposes a limit on controller's minimum clock frequency. Under exchange rate 100kbit/s minimum recommended clock frequency will be equal to 2 MHz. But under exchange rate 400kbit/s 2 MHz frequency will be insufficient to meet the requirements for time of the data set. According to this, controller clock frequency must be at least 20 MHz.

7.5.4 Interface operation algorythm

To enable controller operation required value should be written in I2CxPSC(PSC) and bit should be set I2CxCR(EN) = '1'. Interruptions are allowed by bit I2CxCR(IEN).

Below examples of interaction with «slave» device are described. When operating real devices you should carefully read their documentation and description protocol about interaction with them, they may differ from the descriptions below.

7.5.4.1 Data record To transfer the data byte to «slave» device «master» I^2C must form state «START» on lines SDA, SCL, send «slave» device address and the direction flag $R/\overline{W}$ = '0'. «Slave» device must reply with ACK. Then «master» transmits data byte, awaits for ACK signal and generates a state «STOP».

- write data byte including «master» address and $R/\overline{W}$ = '0' into I2CxTX;
- generate «START» state on SDA, SCL lines, by writing bits I2CxCMD(WR) = '0' и I2CxCMD(STA) = '1';
- wait until bit I2CxST(TIP) takes value '0';
- raed bit I2CxST(RxACK). If bit is equal to '0', then «slave» received information, one can further wait for transaction. If bit is equal to '1', repeat sections from the beginning, «slave» havent received information under some circumstances;
- write data for transmission into I2CxST;
- generate «STOP» state I2CxCMD(WR) = '1' и I2CxCMD(STO) = '1';
- wait until bit I2CxST(TIP) takes value '0';
- read bit I2CxST(RxACK). if bit is equal to '0', then «slave» received data.

7.5.4.2 Data reading I order ro read data byte from a random address in «slave» device «master» I^2C must form « START » state on SDA and SCL lines or transmit «slave» device address and direction flag $R/\overline{W}$ = '0'. Then «master» transmits a byte(s) containing the internal address for «slave» device. It repeatedly forms state «START» on SDA, SCL lines, transmits «slave» device address and the direction flag $R/\overline{W}$ = '1'. it receives data byte(s) from «slave» while responding ACK. After receiving the last data byte, it responds NACK. It generates «STOP» state.

It is worth remembering that the register I2CxRX is being rewritten every time when new data bytes are being received.

- write data byte containing «slave» address and $R/\overline{W}$ = '0' in I2CxTX;
- in order to form «START» state on lines SDA and SCL, write bits I2CxCMD(WR) = '1' и I2CxCMD(STA) = '1';
- wait until I2CxST(TIP) bit takes value '0';
- read I2CxST(RxACK) bit. if bit is equal to '0', then «slave» received information, transaction can be continued. If bit is equal to '1', repeat sections from the beginning, «slave» haven't received information under some reasons;
- write data for transmission into I2CxST and set bit I2CxCMD(WR) = '1';
- wait until bit I2CxST(TIP) takes value '0';

- read bit I2CxST(RxACK). If bit is equal to '0', then «slave» received information, transaction can be continued. If bit is equal to '1', repeat sections from the beginning, «slave» haven't received information under some reasons;
- repeat previous 3 sections until all bytes containing internal address in «slave» are transmitted;
- write data byte containing «slave» address and $R/\overline{W}$ = '1' in I2CxTX;
- form «STOP» state I2CxCMD(WR) = '1' and I2CxCMD(STO) = '1';
- in order to form «START» state on SDA and SCL lines, write I2CxCMD(WR) = '1' and I2CxCMD(STA) = '1';
- wait until bit I2CxST(TIP) takes value '0';
- read bit I2CxST(RxACK). If bit is equal to '0', then «slave» received information, transaction can be continued. If bit is equal to '1', repeat sections from the beginning, «slave» haven't received information under some reasons;
- in order to read data byte from «slave», set I2CxCMD(RD) = '1', I2CxCMD(STO) = '1', I2CxCMD(ACK) = '1';
- wait until bit I2CxST(TIP) takes value '0';
- read data from register I2CxRX, save in DM.
- if it is required to read few data bytes from «slave» then repeat previous 3 sections but do not set I2CxCMD(STO) = '1', I2CxCMD(ACK) = '1' until required amount of data bytes is not received;

7.5.5 Description of registers

Base address I2C0 - 0xC000 1000.

In order to receive real register address add register address displacement to base (initial) address on the bus.

Register	Address displacement	Access	Description
I2CxPSC	00h	RW	Clock frequency predivision register
I2CxCR	04h	RW	Control register
I2CxTX	08h	W	Transmit data register
I2CxRX	08h	R	Receive data register
I2CxCMD	0Ch	W	Command register
I2CxST	0Ch	R	State register

I2CxPSC	Predivision register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																PSC															

16-31 — *reserved*
0-15 PSC predivision value

I2CxCR	Control register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								EN	IEN	—					

8-31 — *reserved*
7 EN controller operation enabling ('1' - permitted, '0' - forbidden)
6 IEN interruption enabling under transmission completion ('1' - permitted, '0' - forbidden)
0-5 — *reserved*

I2CxTX	Transmit data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								TDATA							RW

8-31 — *reserved*
7 TDATA 7 transmit data most significant bits
0-6 RW bit $R/\overline{W}$ during «slave» address transmission, in other cases – data least significant bit

I2CxRX	Receive data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								RDATA							

8-31 — *reserved*
0-7 RDATA last received data byte

I2CxCMD	Command register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								STA	STO	RD	WR	ACK	—		IACK

8-31	—	*reserved*
7	STA	form consequence START(RESTART) ('1' — form)
6	STO	form consequence STOP ('1' — form)
5	RD	reading from slave device ('1' — raed)
4	WR	write in slave device ('1' — write)
3	ACK	data receipt acknowledgment ('0' – ACK, '1' – NACK)
1-2	—	*reserved*
0	IACK	reset of bit I2CxST(IF) ('1' — reset)

I2CxST	State register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								RxACK	BUSY	AL	—	TIP	IF		

8-31	—	*reserved*
7	RxACK	received ACK
6	BUSY	Bus is busy (START state detected, is reset when STOP is detected)
5	AL	control loss under bus
2-4	—	*reserved*
1	TIP	data transmission sign, and formation of STOP
0	IF	byte transmitted or control under line is lost. If bit I2CxCR(IEN) = '1', then interruption requests will occur, even if to cleanse this bit

7.6 Interface I^2C «slave» (I2C1)

7.6.1 Brief characteristics

- operates in «slave» mode;
- compatible with Philips I^2C;
- supports 7-bit addressing;
- exchange speed - 100Kb/s and 400Kb/s;
- external supporting resistors setting is needed on lines SCA and SDA.

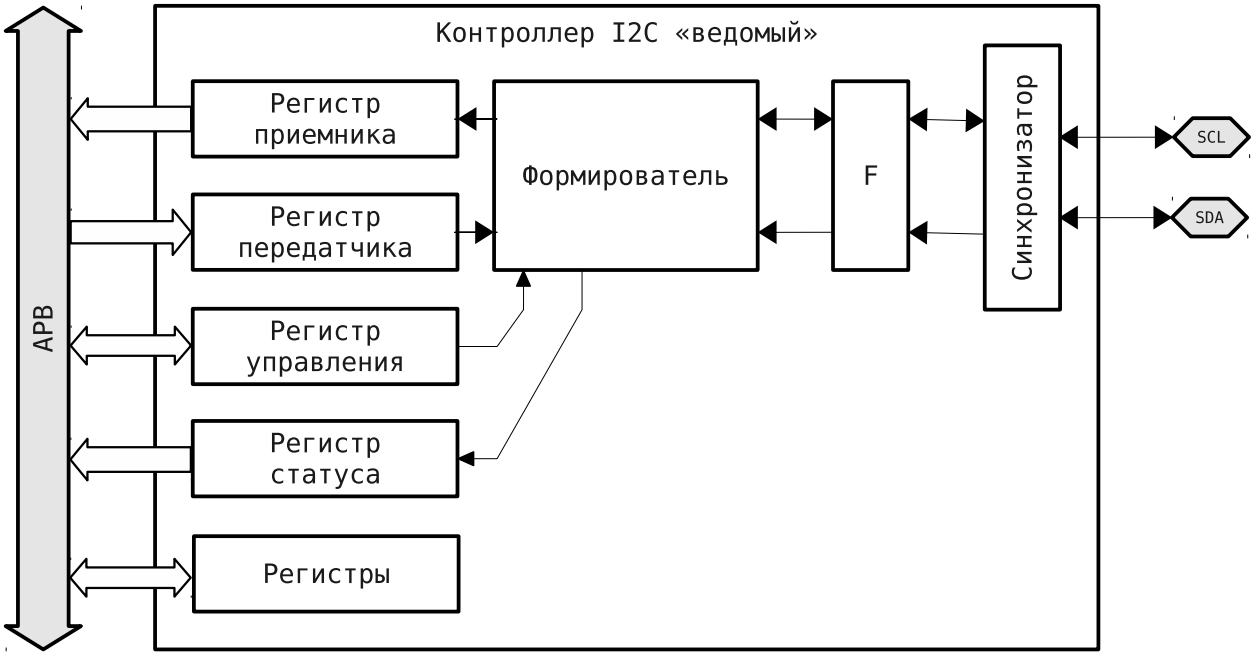

Figure 15: block diagram I2C1 («slave»)

In MP I2C1 operates in «slave» mode.

I^2Ca simple two-wire serial interface with opportunity of several «masters» on the same physical line. Interface provides collision and arbitration detection. I^2C has two physical lines SDA (serial data line) and SCL (serial clock line).

Fig. 13 is a block diagram of the interface controller described. At the external interface lines input a digital low-pass filter is set.

7.6.2 General description of receive-transmit protocol

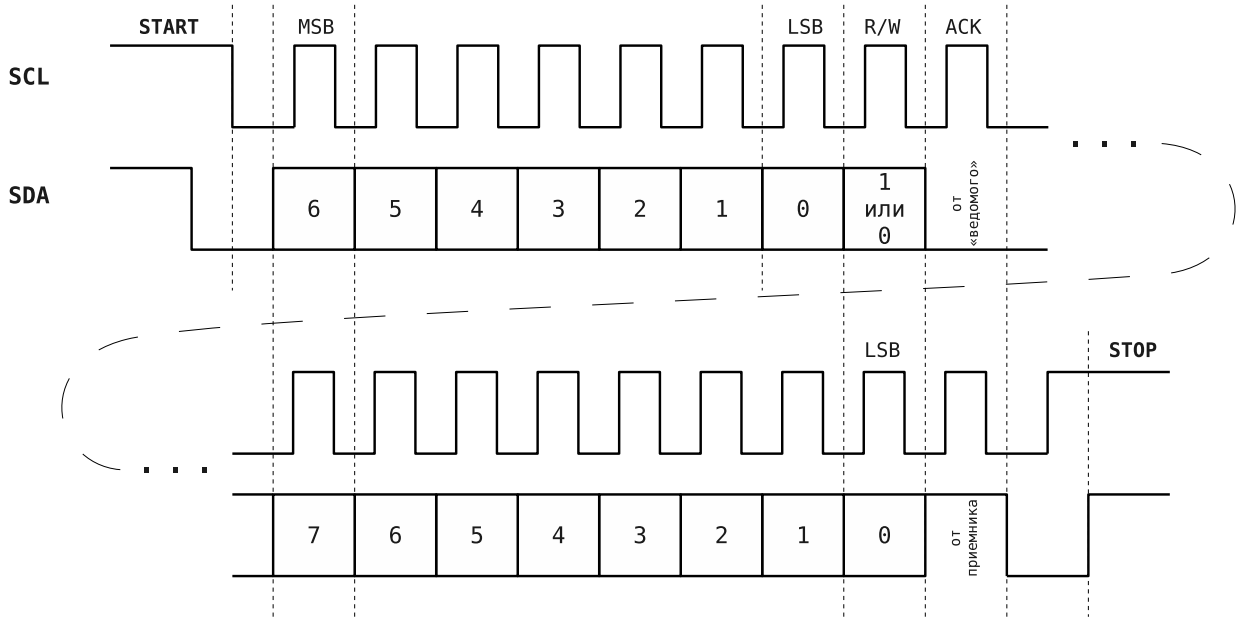

Figure 16: Transaction on the bus I^2C

Receive-transmit is byte-wise implemented.

Transaction start on the bus I^2C is determined by the state «START» on lines SDA and SLC: SDA line transition from state '1' into state '0' under state SCL - '1'.

Transaction completion is determined by state «STOP» on lines SDA and SLC: SDA line transition from state '0' into state '1' under state SCL - '1'.

States «START», «STOP» are formed by bus «master». After «START» formation by bus «master», the bus is considered busy and released only after the formation of «STOP» state «master» which took its place. The time between «STOP» and «START» is determined by I2C standard and depends on its current operation speed.

Fig. 16 shows SDA and SLC lines state examples during transactions. Bus «master» generates «START» state and transmits 7-bit «slave» device address. After the address bit $R/\overline{W}$ follows, which determines data transfer direction ('1' - reading from «slave» device '0' - writing into «slave» device). After transmission of address and bit $R/\overline{W}$, «master» releases SDA line, and «slave» should lead SDA line into state '0'. If this does not happen, it is believed that ACK (acknowledgment) signal is not received from «slave». Bus «master» can form «STOP» state and repeat the transaction with this «slave» or take other actions incorporated in its operation algorithm.

If ACK signal is received from «slave», then data transfer begins, which direction was determined by bit $R/\overline{W}$. Data can be transmitted until receiver responds ACK for each transmitted data byte. That is, after each data byte transfer, the transmitter releases SDA line for one SLC period, so that receiver could sigmnalize if information byte is received or not and if it is ready to receive or not the following one. After NAK respond (while waiting for ACK '1'is transmitted) «master» forms «STOP» state. «Master» may also interrupt transaction to form the state «STOP».

7.6.3 Carrier frequency generation

Controller I^2C I2C1 is clocked from outside. SLC and SDA signals pass through the synchronizer and LPF. Their states are synchronized with the system frequency F_{sys}. This imposes a limit on the minimum frequency value F_{sys}, it must not be less than 8 times exceeding bus exchange frequency. Recommended for speed 100kbit/s $F_{sys} \geq 2MHz$, for speed 400kbit s $F_{sys} \geq 6MHz$.

7.6.4 Interface operation algorythm

The interface has four modes, which are defined in register I2CxCR. They define controller behavior after data byte receipt of transmitdata byte. The states are recorded in the register I2CxST. They are also a source of interrupt request from the controller.

7.6.4.1 Data receipt from «master» After receiving data byte, controller will respond NAK for all subsequent, until I2CxRX register is being read. ACK is automatically formed I2CxRX is being read . Data bytes that have not been confirmed by ACK, are not stored in I2CxRX.

Controller behavior after receiving data byte is set by bit I2CxCR(RMOD).

If I2CxCR(RMOD) = '0', then controller awaits bus «master» respond. It will receive data into shift register and respond NAK to each data byte, until I2CxRX register is read I2CxRX.

If I2CxCR(RMOD) = '1', then controller blocks SCL line and keeps it in state '0' until I2CxRX register is read and bit I2CxST(REC) is cleansed (cleansed automatically when reading I2CxRX).

7.6.4.2 Data transmission to «master» Data transmission to «master» is controlled by bit I2CxCR(TV). If the bit is equal to '1', after receiving address controller confirms it

with ACK state and begins to transmit data located in register I2CxTX. After data byte has been transmitted, bit I2CxCR(TV) is assigned with value I2CxCR(TAV). This allows you to transmit the same byte for all requests without wasting CPU time.

Controller behavior after data transmission «master» and ACK receipt. If «master» responds NAK, the controller will change into START await state lines SLC and SDA. Bit I2CxST (NAK) will take value '1'.

If I2CxCR (TMOD) = '0', then after ACK respond by «master», the controller continues to wait for respond from «master». If it continues data reading operation of data from «slave», then controller will transmit data that should be stored in register I2CxTX.

If I2CxCR (TMOD) = '1', then after ACK respond by «master», the controller blocks SCL line and holds it in a state '0' until bit I2CxCR(TV) is equal to '1'. Use this mode carefully since bus «master» does not have the ability to control the signal SCL, as well as bus operation depends on the software algorithm written by the user for the system where «slave» device operates.

If I2CxCR (TAV) is '1', then controller behavior with any value of bit I2CxCR (TMOD) is the same.

7.6.5 description of registers

Base address I2C1 - 0xC000 1100.

In order to receive real register address add register address displacement to base (initial) address on the bus.

Register	Register displacement	Access	Description
I2CxSAD	00h	RW	«Slave» address register
I2CxCR	04h	RW	Control register
I2CxST	08h	RW	State register
I2CxMSK	0Ch	RW	Mask register
I2CxRX	10h	R	Receive data register
I2CxTX	14h	W	Transmit data register

I2CxSAD	«Slave» address register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																											SLVADDR				

8-31	—		*reserved*
0-7	SLVADDR		7-bit «slave» device address

I2CxCR	Control register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																											U				0
Description	—																											RMOD	TMOD	TV	TAV	EN

5-31	—		*reserved*
4	RMOD		data receipt mode ('1' - «slave» receives data and holds SCL in '0' until data from I2CxRX will be read (ACK will be formed and transmitted), '0' - «slave» receives data) and forms NAK for given information byte and subsequent, until I2CxRX register will be read
3	TMOD		data transmit mode ('1' - «slave» transmits one and the same data byte and forms NAK for all requests after, until I2CxCR(TV) = '0', '0' - «slave» transmits one byte and holds SCL in '0' until I2CxCR(TV) = '0')
2	TV		transmission acknowledgement ('1' - acknowledges data transmission (after transmission data byte automatically takes value '0'), '0' - forms NAK and holds SCL in '0' depending on I2CxCR(TMOD))
1	TAV		data transmission is always acknowledged ('1' - permitted, '0' - forbidden)
0	EN		controller operation enabling ('1' - permitted, '0' - forbidden, lines SCL and SDA are in 3d state)

I2CxST	State register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																													REC	TRA	NAK

I2CxMSK	Mask register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	U																															
Description	—																													RECE	TRAE	NAKE

3-31	—	*reserved*
2	REC	byte received ('1' - received (automatically cleansed, when I2CxRX is read), '0' - not received)
1	TRA	byte transmitted ('1' - transmitted (cleansed by record '1' в I2CxST(TRA)))
0	NAK	NAK formed for request ('1' - NAK formed)). If «slave» address does not coincide with I2CxSAD, then NAK transmit does not affect this bit

3-31	—	*reserved*
2	RECE	interruption formation enabling under I2CxST(REC) ('1' - permitted, '0' - forbidden)
1	TRAE	interruption formation enabling under I2CxST(TRA) ('1' - permitted, '0' - forbidden)
0	NAKE	interruption formation enabling under I2CxST(NAK) ('1' - permitted, '0' - forbidden)

I2CxTX	Transmit data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								TDATA							

8-31 — *reserved*

0-7 TDATA 7 transmit data most significant bits

I2CxRX	Receive data register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																								RDATA							

8-31 — *reserved*

0-7 RDATA last received data byte

7.7 Controller I^2S(I2Sx)

7.7.1 Brief characteristics

- data receiver implements, operating in «master» mode;
- compatible with Philips I^2S;
- sample bit selection: from 16 to 32 bit;
- synchronizing frequency predivision;

7.7.2 Bus general description I^2S

Bus is designed only for audio data processing, while other signals, such as the sub-coding and control signals, are transmitted separately. To minimize the number of required contacts serial bus is used comprising three lines, which includes data transmitting line of two channels with time time multiplexing, selective line and synchronization line.

Since transmitter and receiver have the same clock signals for data transmittion, transmitter as master device must generate a synchroniztion signal, data signal and selective signal. However, in complex systems, there may be several receivers and transmitters and this makes difficult to determine the master. Such systems typically have a drive system for managing digital audio data streams between various devices. Then, the transmitters need to generate data under control of an external synchro signal, and operate as slave devices. In general, I^2Sinterface consists of two distinct cores - transmitter and receiver. Both can operate in either master or slave mode. To transfer the sound through I^2Sone-way minimum 3 lines are required:

- Bit clock — SCK (clock);
- Word select — WS (channel selection line);
- Data line – SD (audio data transmossion line).

Signals WS and SCK are designed only by master device (fig. 17).

Time signal diagram I^2Srepresented on fig. 18:

WS line indicates which channel data is being transmitted, low level ('0') corresponds with the left channel, high ('1') with right, WS change occurs on negative edge SCK. The transmitter changes SD data line value SD data at the negative edge of the signal SCK, the

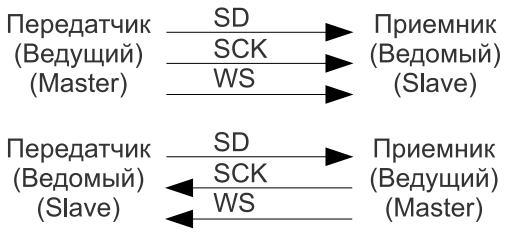

Figure 17: Signal direction I^2S

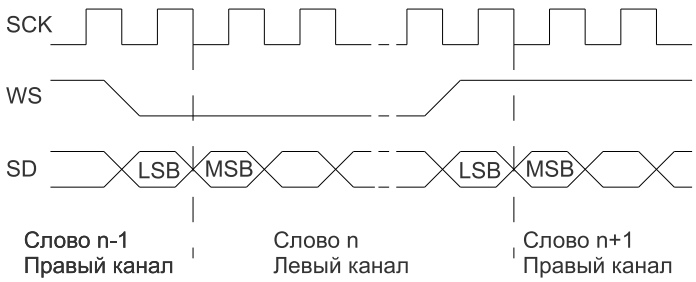

Figure 18: Time signal diagram I^2S

receiver reads at the positive. High word bit is transmitted on the second positive edge of SCK signal after WS signal change.

7.7.3 Description of registers

Base address I^2S- 0xC010 2000

In order to receive real register address add register address displacement to base (initial) address on the bus.

Регистр	Address displacement	Access	Description
I2SxCFG	00h	RW	Receiver setting register
I2SxMSK	04h	RW	Interrupt mask register
I2SxINT	08h	RW	Interrupt register
I2SxRX	0Ch	R	Receive data register

I2SxCFG	Receiver setting register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0										0						0								0					0	0	0
Description	—										RES						PSC								—					SWAP	INTEN	RXEN

22-31	—	*reserved*
16-21	RES	bit amount in written audio data (sample size)(16-32 bit)
8-15	PSC	transmitting frequency division value
3-7	—	*reserved*
2	SWAP	left channel write setting ('1' - into odd addresses, '0' - even addresses)
1	INTEN	interrupt enabling ('1' - permitted, '0' - forbidden)
0	RXEN	operation enabling ('1' - permitted, '0' - forbidden)

I2SxMSK	interrupt mask register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																													0	0	
Description	—																													HSBF	LSBF	

For all register bits: ('1' - permit, '0' - forbidden)

2-31	—	*reserved*
1	HSBF	upper audio data buffer is full
0	LSBF	lower audio data buffer is full

I2SxINT	Interrupt register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																													0	0	
descriptio	—																													HSBF_ST	LSBF_ST	

For all register bits: ('1' - permit, '0' - forbidden)

2-31	—	*reserved*
1	HSBF_ST	upper audio data buffer is full
0	LSBF_ST	lower audio data buffer is full

I2SxRX	Receive register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	**RX_OUT**																															

0-31 RX_OUT received data

7.8 general purpose timer (GPTIMx)

7.8.1 Brief characteristics

- represents decrementing 32-bit spinner;
- predivision 16 bit;
- single-shot and continuous calculating mode;

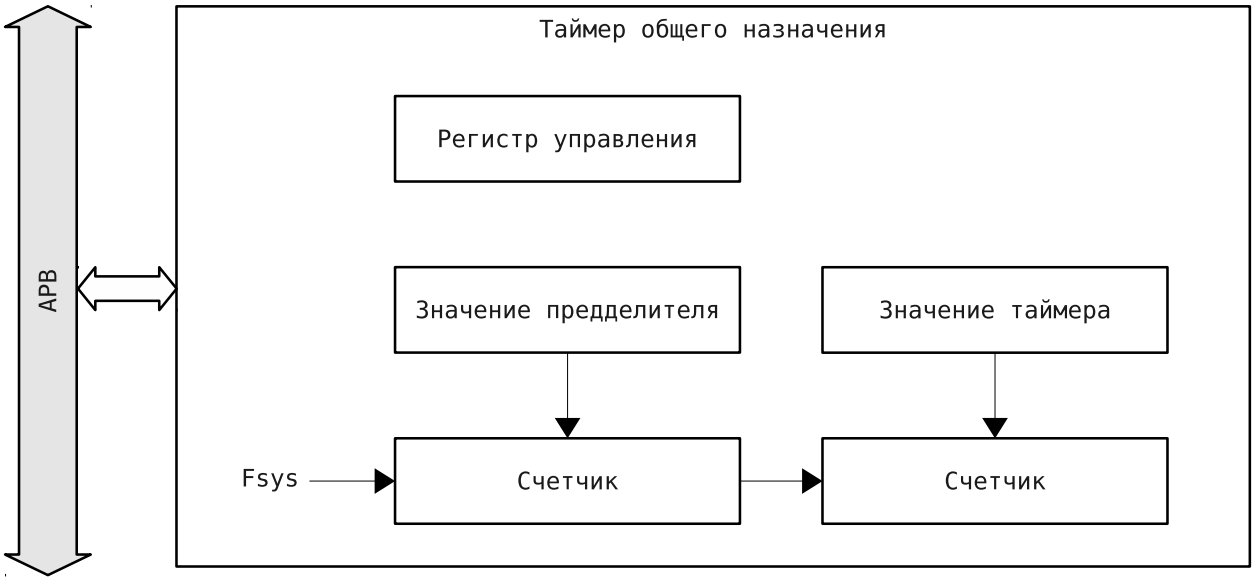

Figure 19: Block diagramm GPTIMx

Executive timer's part consists of predivision and timer. Predivision and timer represent decrementing spinners with initial value registers, from which it is loaded into spinner after reaching value −1. Fig. 19 represents block diagramm GPTIMx.

7.8.2 Work alogorythm

Timer begins counter after bit TIMxCR(EN) = '1'. is set The internal clock signal after the predivision is applied to the timer spinner. Once its value is −1, an interrupt handling request is formed bit TIMxCR (IP) takes the value = one, value TIMxCNTPER (CNTPER) is loaded into current spinner value register TIMxCNTVAL (CNTVAL). If continuous operation mode of spinner is set (bit TIMxCR (RS) = one), then these events are recurrent.

If single-shot operation mode is set (bit TIMxCR (RS) = zero), then calculation renewal is not being implemented, spinner is not being decremented.

At any time, the timer can be reset by its initial value at setting bit TIMxCR (LD) = one.

Timer period may be calculated by means of the following formula:

$$T_{GPTIM} = T_{sys}\ cdot PSCPER\ cdot CNTPER,\ quad PSCPER\ geq 2$$

It has tobe emphasized, that value TIMxPSCPER(PSCPER) cannot be less then 2, even though such value could be written there.

7.8.3 Register description

Base register addresses GPTIMx:
GPTIM0 - 0xC001 0000
GPTIM1 - 0xC001 0100
GPTIM2 - 0xC001 0200
GPTIM3 - 0xC011 0000
GPTIM4 - 0xC011 0100
GPTIM5 - 0xC011 0200
GPTIM6 - 0xC011 0300

To receive the real address of the register register address displacement is to be added to to base (initial) address on the bus.

Register	Address displacement	Access	Description
TIMxPSCVAL	00h	RW	Predivision current value register
TIMxPSCPER	04h	RW	Predivision initial value register
TIMxCNTVAL	10h	RW	Timer current value register
TIMxCNTPER	14h	RW	Timer initial value register
TIMxCR	18h	RW	Control register

TIMxPSCVAL	Predivision initial value register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																PSCVAL															

16-31 — *reserved*
0-15 PSCVAL Spinner predivision current value

TIMxPSCPER	Predivision initial value register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																PSCPER															

16-31 — *reserved*
0-15 PSCPER Predivision initial value (predivision period)

TIMxCNTVAL	Timer current value register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																CNTVAL															

16-31 — *reserved*
0-15 CNTVAL timer spinner current value

TIMxCNTPER	Timer initial value register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																CNTPER															

16-31	—		*reserved*
0-15	CNTPER		Predivision spinner initial value (predivision period)

TIMxCR	Control register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	—																											IP	IE	LD	RS	EN

5-31	—		*reserved*
4	IP		attribute of formed interruption ('1' - formed, '0' - no interrupt request), cleansed by record '1' in this bit
3	IE		permission to form interruption ('1' - permitted, '0' - forbidden)
2	LD		timer restart ('1' - load TIMxCNTPER(CNTPER) in TIMxCNTVAL(CNTVAL)
1	RS		timer operation mode ('1' - continious, '0' - single-shot)
0	EN		timer operation permit ('1' - permitted, '0' - forbidden)

7.9 Controller Ethernet(Ethernet0)

7.9.1 Brief description

- supports speed 10/100МБит/с
- full duplex, semi-duplex operation modes;
- direct RAM access channel;
- communication interfaces MII, RMII support;
- has MDIO interface;
- supports standard IEEE 802.3-2002 и IEEE 802.3Q-2003

Controller Ethernet0 consists of 3 functional models:

- direct memory acces controller (DMAC)
- MDIO
- Ethernet Debug Communication Link (EDCL)(option, see table MP integration)

DMAC is used for data tarnsmission between MP internal memory and Ethernet0 controller. All received and formed for transmission data bursts are stored in MP internal memory. The receiver and transmitter have separate DMAC.

MDIO is used to configure and control external media conversion (PHY).

EDCL (optional) provides access to the internal peripheral bus via Ethernet network. It uses protocols UDP, IP, ARP. EDCL, uses Ethernet0 receiver and transmitter.

Ethernet0 supports the following standards: IEEE 802.3-2002 and IEEE 802.3Q- 2003 (optional, see table MP integration). The controller does not support packages such as 0x8808, they will not be accepted.

Ethernet0 receiver and transmitter are connected with external media conversion via interface converter Media Independent Interface (MII). Interface Reduced Media Independent Interface (RMII) is also supported.

Size of receiver and transmitter descriptor tables - 1KB.

7.9.2 Clocking

Ethernet0 transmitter and receiver are clocked by external media conversion, clock signals are independent for receiver and transmitter, and are the part of interfaces MII or RMII. Internal control structure are clocked by system frequency Fsys. Controller supports half-duplex and full-duplex operation modes that can work on transmission rates of 10 and 100 Mbit/s. Minimum Fsys required for proper operation at speeds of 10Mbit / s - 2.5MGts, to 100Mb / s - 18MHz. Fsys below the required values may cause packet loss.

7.9.3 Access to internal FIFO receive-transmit buffers.

To enable this feature, you must set CR bit (ramdebugen) = 1. When this mode is enabled, EDCL packets are not received. Ethernet0 controller receiver and transmitter must be switched off or data in FIFO buffers can be damaged.

The controller provides access to the internal receiver and transmitter FIFO buffers of Ethernet0 controller and EDCL. Transmitter buffer is accessed via peripheral bus with displacement from base address to 0x10000 and 0x107FC. A total of 512 32-bit words. Receiver buffer is accessed via peripheral bus with displacement from base address of 0x20000 Total 512 32-bit words. Buffer EDCL is accessed via peripheral bus with displacement from base address of 0x30000. Total 256-16384 32-bit

7.9.4 Transmitter DMAC

Transmitter uses descriptors placed in internal MP memory. Descriptor cannot be changed during transmission.

7.9.4.1 Decriptor setting Descriptor has direction to address where data block and its size are placed. It also contains control information. Data block address must be aligned 32 bits. The descriptor should not be changed until Ethernet0 controller is not set. TD0(EN) = '0'.

TD0	Ethernet0 descriptor, part 0 (address displacement 0x0)																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	—																AL	UE	IE	WR	EN	LENGTH										

31..16	—	*reserved*
15	AL	packet not transmitted, since trial amount has exceeded the maximum
14	UE	packet is transmitted incorrectly, since FIFO was completely filled
13	IE	interrupt enabling under packet transmission completion, regardless of whether it is transmitted correctly or not
12	WR	permission for descriptor table pointer to receive value 0 after given packet transmission ('1' - permitted, '0' - forbidden). If WR=0, then descriptor table pointer is incremented to 8 and takes value 0 only after reaching descriptor table border.
11	EN	one descriptor operation enabling ('1' - permitted, '0' - forbidden)
10..0	LENGTH	data block size for receipt in byte

TD1	Ethernet0 descriptor, part 1 (address displacement 0x4)																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	ADDRESS																													—		

31..2	ADDRESS	direct to initial memory address, where data for transmission is placed
1..0	—	*reserved*

7.9.4.2 Data preparation for transmission Full data packet except CRC is to be placed in the memory starting address must be specified in the descriptor. Package length specified in the descriptor should not exceed 1514 bytes as possible, otherwise the packet will not be transmitted.

7.9.4.3 Data transmission To start data transmission, you need to set the pointer to descriptor table address and set the bit TD0(EN) in the corresponding descriptor. Descriptor table address must be aligned to 1K. Bits 31..10 contain base address of descriptor table, 9..3 - pointer to a specific descriptor (in bytes). The pointer is set to 0 as soon as it exceeds 1KB. In case if bit TD0(WR)='1'is set in some descriptor, then descriptor pointer takes value 0 when it comes to that descriptor.

After address setting data transmission CR(TX _EN)='1'required addresses are to be enabled. this designates that all descriptions are prepared, data transmission is allowed.

7.9.4.4 Descriptor operation after data transmission completion After transmission is complete, the appropriate status bits will be written in TD0 after packet transmission completion described by descriptor. The package is transmitted successfully if TD0(UE) and TD0(AL) have value '0'. TD0(UE)='1', if during transmission FIFO transmitter is empty. TD0(AL)='1' is set if during transmission collisions occurred more than foreseen by protocol. All other TD0 bits are set = '0' after the completion of package transmission. TD1 remains unchanged. Bit TD0(EN) may be used as an indicator that the descriptor is ready to be used as Etherneth controller automatically sets it into '0' after packet transmission.

In addition to displaying information in the descriptor, there are transmitter status bits in the controller: ... (TE) - a transmission error, ... (TI) - interrupt handling request, is set every time when transmission is completed successfully. .. (TA) - data exchange error via peripheral bus. In this case, the transmitter will be stopped.

7.9.5 Receiver DMAC

Receiver uses descriptors placed in internal MP memory. Receiver DMAC is designed to receive data through Ethernet network.

7.9.5.1 Descriptor setting Descriptor has direction to address where data block and its size are placed. Control information is also there. Data block address should be aligned to 32 bit.

TD0	Ethernet0 descriptor, part 0 (address displacement 0x0)																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	—					MC	—							LE	OE	CE	FT	AE	IE	WR	EN	LENGTH										

31..27	—	*reserved*
26	MC	packet destination address is multicast (not transmitted)
25..19	—	*reserved*
18	LE	error, value in "packet length" does not match the number of received bytes
17	OE	error, block was received incorrectly due to receiver buffer overflow
16	CE	error CRC in block
15	FT	error, received block is above maximum size, the excess part owas rejected
14	AE	error, odd number of halfbytes is received
13	IE	interrupt enabling ('1' - permitted, '0' - forbidden). Interrupts will be generated after packet receipt (bit ETHxCR(RI) should be in '1'), Interrupts are generated regardless of whether the packet receipt completed successfully or an error occurred.
12	WR	descriptor table pointer enabled to take value 0 after given packet transmission ('1' - permitted, '0' - forbidden). If WR=0, then descriptor table pointer incremented to 8 and takes value 0 only after raeching descriptor table border.
11	EN	descriptor operation enabling (field is set last) ('1' - permitted, '0' - forbidden)
10..0	LENGTH	data block size for transmission in bytes

TD1	Ethernet0 descriptor, part 1 (address displacement 0x4)																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
description	ADDRESS																													—		

31..2	ADDRESS	memory initial address pointer, where received data placed
1..0	—	*reserved*

7.9.5.2 Data receipt To begin data receipt, it is necessary to set a pointer to descriptor table address and set TD0(EN) bit in the corresponding descriptor. Descriptor table address must be aligned to 1K. Bits 31..10 contain descriptor table base address, 9..3 - pointer to a specific descriptor (in bytes). The pointer is set to 0 as soon as it exceeds 1KB. In case bit

TD0(WR)='1'is set in some descriptor, descriptor pointer takes value 0 when it comes to that descriptor.

After address setting is is necessary to enable data receipt ETHxCR(RE)='1'. This designates that all descriptions are prepared, data transmission is enabled to start.

7.9.5.3 Descriptor processimg after date transmission is finished After receipt is completed, bit TD0(EN) has description '0'. Bits TD0(WR) and TD0(IE) have description '0'as well. Amount of received bytes is represented in TD0(LENGTH). Parts of Ethernet frame contain destination address, source address, data type and field. Bits 17..14 in TD0 signalize about receiving errors. After successful receipt all 4 bits have description '0'. Lower 64 byte limit batch is not received and is rejected. Current receiving register is forbidden to be changed before receiving of the first batch with the accepted volume. Bit ETHxST(TS) signalizes about receiving error of the lowerlimit volume batch. Bit ETHxST(IA) signalizes about batch receipt with forbidden MAC address. Bit TD0(FT) signalizes about data batch receipt exceeding maximum allowed size. TDO(LENGHT) field does not garantee correct data receipt. Empty bytes amount lower maximum packet size after word, containing the last byte, is written to memory.

7.9.6 Description of registers

Base address Ethernet0 - 0xC000 5000

To get a real registers's add address displacement of a register to the base (initial) address of the bus

Register	Address displacement	Access	Description
ETHxCR	00h	RW	Configuration settings register
ETHxST	04h	RW	State register
ETHxMACMSB	08h	RW	MAC address upper part
ETHxMACLSB	0Ch	RW	MAC address lower part
ETHxTDP	14h	RW	Transmitting descriptor word index
ETHxRDP	18h	RW	Receiving descriptor word index

ETHxCR	Control register																											
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0						0	0													0	0	0	0	0	0	0	0	0	0	0	0
Description	—						MC	—													ME	PI	-	SP	RS	PM	FD	RI	TI	RE	TE	

26-31 — *reserved*

25 MC Multiaddress condition status ('1' - permitted, '0' - forbidden)

12-24 — *reserved*

11 ME permit to receive multiaddress batches ('1' - permitted, '0' - forbidden)

10 PI permit interruptions when changing the status of external PHY ('1' - permitted, '0' - forbidden)

8-9 — *reserved*

7 SP speed ('1' - 100 Мбит/с, '0' - 10 Мбит/с)

6 RS reset, bit will be cleansed after controller's reset completion, no other operations are to be conducted with the controller when bit is equal to '1' ('1' - initiate reset of Ethernet controller)

5 PM receive all batches in spite of destination unit address ('1' - permitted, '0' - forbidden)

4 FD full duplex operation ('1' - permitted, '0' - forbidden)

3 RI permit receiver interruptions ('1' - permitted, '0' - forbidden)

2 TI permit transmitter interruptions ('1' - permitted, '0' - forbidden)

1 RE receipt permit, bit is automatically reset into '0' after receiving of batch is finished. Bit is to be set only after receive descriptor is recorded ('1' - permitted, '0' - forbidden)

0 TE receipt permit, bit is autimatically reset into zero '0' after receiving of the batch is finished. Set bit only after transmitting descriptor is recorded. ('1' - permitted, '0' - forbidden)

ETHxST	State register																											
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																							0	0	0	0	0	0	0	0	0
Description	—																							PS	IA	TS	TA	RA	TI	RI	TE	RE

For all register bits: ('1' - presence of an attribute, '0' - absence of an attribute)

9-31 — *reserved*

8 PS PHY status changes

7 IA packet with the address inconsistent with MAC is received. Is under cleansing by record '1'

6 TS data batch with a lower limit size is received. Is under cleansing by record '1'

5 TA Transmission unit error when processing in DMA channel. Collisions in system bus or in the process of memory access. Is under cleansing by record '1'

4 RA Receiver error when processing in DMA channel. Collisions in system bus or in the process of memory access. Is under cleansing by record '1'

3 TI batch is received without errors. Is under cleansing by record '1'

2	RI	batch is received without errors. Is under cleansing by record '1'
1	TE	Batch transmission is discontinued by error. Is under cleansing by record '1'
0	RE	Batch transmission is discontinued by error. Is under cleansing by record '1'

ETHxMACMSB	MAC address upper part																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																0															
Description	—																47..32 bits MAC															

16-31 — *reserved*
0-15 MACMSB two most significant bytes of MAC address

ETHxMACLSB	MAC address upper part																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																															
Description	31..0 bits MAC																															

16-31 — *reserved*
0-15 MACLSB lower bytes of MAC address

ETHxTDP	Transfer descriptor word index																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																						0							0		
Descriptoon	BASEADDR																						DESCPNT							-		

10-31 BASEADDR base address for descriptor words. Address 0xE0200000 + real address in data memory are to be set, only uppermost bits 31..10 are recordered
3-9 DESCPNT descriptor indicator, is automatically incremented when new data batch is received
0-2 — *reserved*

ETHxRDP	Receiving descriptor word index																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																						0							0		
Description	BASEADDR																						DESCPNT							-		

10-31 BASEADDR base address for descriptor words. Address 0xE0200000 + real address in data memory are to be set, only uppermost bits 31..10 are recordered
3-9 DESCPNT descriptor indicator, is automatically incremented when new data batch is received
0-2 — *reserved*

7.10 PWM controller (PWMx)

7.10.1 Brief characteristics

- single-pulse operation;
- spinner period updatability in the process of operation (under certain conditions);

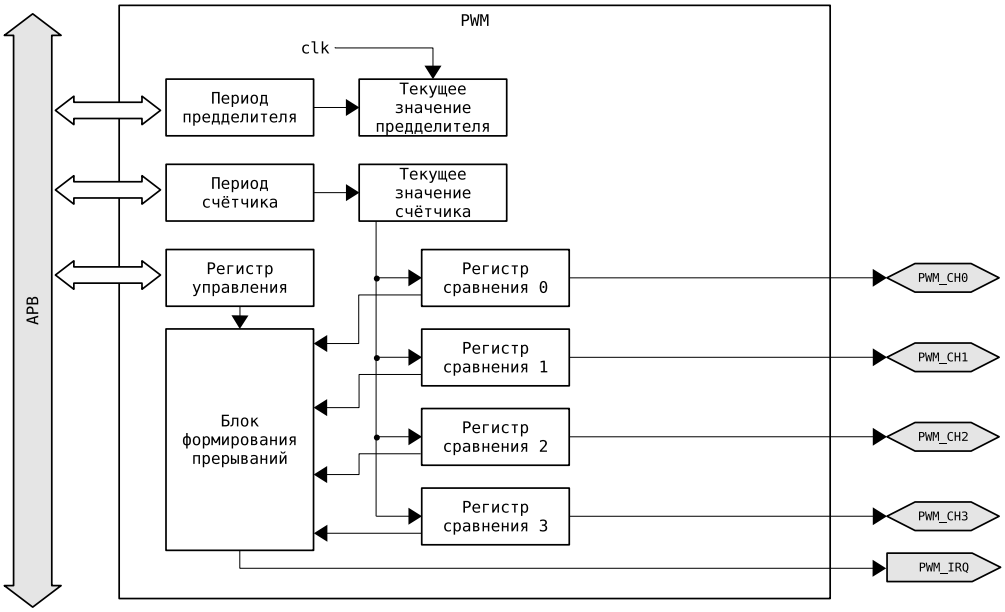

Figure 20: PWM block diagram

PWM controller (PWM) is used for generating latitudinal pulse-modulated waves. PWM inludes 4 channels and has single pulse mode generating regime, and is subject to change spinner period in the process of operation (subject to the conditions described in the relevant section).

7.10.2 PWM initialization

In order to initialize PWM interface and spinner operation mode must be set in register PWM_CR, and channels which operation needs to be enabled must be chosen. The next step is to set the active / inactive level on / off channel. Additionally, interrupts of applicable channel may be enabled, interrupt enabling is possible under overflow spinner.

7.10.3 PWM operation modes

PWM can be generated at once or intermittently, installation is performed in bit PWM_CR(PULSE_
Spinner is run in three modes: incrementing, decrementing, maximum value increase and decrease to zero mode. Spinner operation mode setting is implemented in bit AUTO_RELOAD Spinner operation mode setting is implemented in bit PWM_CR(AUTO_RELOAD) of control register. Permission to activate spinner period updatability in the process of operation in bit CNT_MODE is granted, but only in case of spinner period operating in bit PWM_CR(CNT_MODE), but only under certain conditions.

7.10.4 PWM Interrupts

Interrupts are of two types: overflow spinner and condition when spinner reaches specific channel value compare register. Interrupts are allowed in control register management, and interrupt request is recorded in PWM_INT register.

7.10.5 PWM pulse duration

PWM pulse duration will be determined as the difference between the value of spinner period PWM_CNT and compare register value PWM_CMPCHn, multiplied by intersection of single processor cycle duration and predivision value PWM_PSC. PWM impulse active level duration will be determined as the difference between spinner period value PWM_CNT and compare register value PWM_CMPCHn, multiplied by intersection of single processor cycle duration and the predivision value PWM_PSC.

$$T_{ACT} = (T_{cnt} - T_{cmpch}) \cdot T_{sys} \cdot (PSC + 1)$$

7.10.6 Description of registors

Base address PWM - 0xC001 2000. To receive the real address of the register add register address displacement to base (initial) address on the bus

Annotation: compare registers under n=[0;3] will have the following displacement addresses :

PWMxCMPCH0 – 0x20

PWMxCMPCH1 – 0x24

PWMxCMPCH2 – 0x28

PWMxCMPCH3 – 0x2C

Register	Address displacement	Access	Description
PWMxCR	00h	RW	Control register
PWMxINT	04h	RW	Interrupt register
PWMxCNTVAL	08h	RW	Current spinner value register
PWMxPSC	0Ch	RW	Predivision value register
PWMxCNT	10h	RW	Spinner period register
PWMxCMPCHn	0x20-0x2Ch	RW	Spinner period register

PWMxCR	Control register																													
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0		0	0
Description	—								CH3MODE	CH2MODE	CH1MODE	CH0MODE	EN_CH3_IRQ	EN_CH2_IRQ	EN_CH1_IRQ	EN_CH0_IRQ	OUT_LVL_3	OUT_LVL_2	OUT_LVL_1	OUT_LVL_0	CH_EN_3	CH_EN_2	CH_EN_1	CH_EN_0	—			OVF_IRQ	CNT_MODE		AUTO_RELOAD	PULSE_MODE

24-31	—	*reserved*
23	CH3_MODE	Active level setting of switched channel 3 ('0' – logic 0, '1' – logic 1)
22	CH2_MODE	Active level setting of switched channel 2 ('0' – logic 0, '1' – logic 1)
21	CH1_MODE	Active level setting of switched channel 1 ('0' – logic 0, '1' – logic 1)
20	CH0_MODE	Active level setting of switched channel 0 ('0' – logic 0, '1' – logic 1)
19	EN_CH3_IRQ	Interrupt enabling for channel 3 ('0' – forbidden, '1' – permitted)
18	EN_CH2_IRQ	Interrupt enabling for channel 2 ('0' – forbidden, '1' – permitted)
17	EN_CH1_IRQ	Interrupt enabling for channel 1 ('0' – forbidden, '1' – permitted)
16	EN_CH0_IRQ	Interrupt enabling for channel 0 ('0' – forbidden, '1' – permitted)
15	OUT_LVL_3	Inactive level setting of switched channel 3 ('0' – logic 0, '1' – logic 1)
14	OUT_LVL_2	Inactive level setting of switched channel 2 ('0' – logic 0, '1' – logic 1)
13	OUT_LVL_1	Inactive level setting of switched channel 1 ('0' – logic 0, '1' – logic 1)
12	OUT_LVL_0	Inactive level setting of switched channel 0 ('0' – logic 0, '1' – logic 1)
11	CH_EN3	Interrupt enabling for channel 3 ('0' – forbidden, '1' – permitted)
10	CH_EN2	Interrupt enabling for channel 2 ('0' – forbidden, '1' – permitted)
9	CH_EN1	Interrupt enabling for channel 1 ('0' – forbidden, '1' – permitted)
8	CH_EN0	Interrupt enabling for channela 0 ('0' – forbidden, '1' – permitted)
5-7	—	*reserved*
4	OVF_IRQ	Interrupt enabling under spinner overflow ('0' – forbidden, '1' – permitted)
2-3	CNT_MODE	Spinner operation mode setting: 10 – spinner increases to maximum value, and then decreases to zero 01 – decrementing 00 – incrementing

1	AUTO_RELOAD	Spinner period enabling in the process of operation ('0' – forbidden, '1' – permitted)
0	PULSE_MODE	Разрешение работы в однократном режиме ('0' – forbidden, '1' – permitted)

PWMxINT	Interrupt register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																												0	0	0	0
Description	—s																												CH3_IRQ	CH2_IRQ	CH1_IRQ	CH0_IRQ

4-31	—		*reserved*
3	CH3_IRQ	Channel 3 spinner reached compare register value	
2	CH2_IRQ	Channel 2 spinner reached compare register value	
1	CH1_IRQ	Channel 1 spinner reached compare register value	
0	CH0_IRQ	Channel 0 spinner reached compare register value	

PWMxCNTVAL	Current spinner value register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																0															
Description	—																CNT_VAL															

16-31	—		*reserved*
0-15	CNT_VAL	Curent spinner value	

PWMxPSC	Predivision value register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																0															
Description	—																PSC															

16-31	—		*reserved*
0-15	PSC	Predivision value	

PWMxCNT	Spinner period register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																0															
Description	—																CNT_PER															

16-31	—		*reserved*
0-15	CNT_PER	Spinner period value	

PWMxCMPCHn	Compare register																														
Bit number	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Initial state	0																0															
Description	—																CMP_VAL															

16-31	—		*reserved*
0-15	CMP_VAL	Channel compare register value	

8 Processor's pinmap assignation

8.1 Processor input/output table QFP256

Условные обозначения	
s	подсоединение к линиям электропитания
i	вход
o	выход
a	вход/выход аналоговых сигналов
nc	не подсоединен
vdd	1.8В ядро
vss	gnd питания ядра
dvss	3.3В порты ввода/вывода
dvdd	gnd питания портов
vdd_pll	1.8В ядро PLL
vss_pll	gnd питания ядра PLL
avdd_pll	1.8В аналоговые блоки PLL
avss_pll	gnd питания аналоговых блоков PLL
avss_dac	gnd питания аналоговых блоков DAC
avdd_dac	1.8В аналоговые блоки DAC
vdd_conv	1.8В ядра конверторов (АЦП/ЦАП)
vss_conv	gnd питания ядер конверторов (АЦП/ЦАП)
avss_adc	gnd питания аналоговых блоков DAC
avdd_adc	1.8В аналоговые блоки DAC

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
1	i/o	GPIOF	0	usb0_data[0]	
2	i/o		1	usb0_data[1]	
3	i/o		2	usb0_data[2]	
4	i/o		3	usb0_data[3]	
5	i/o		4	usb0_data[4]	
6	i/o		5	usb0_data[5]	
7	s	dvss			
8	i/o	GPIOF	6	usb0_data[6]	
9	i/o		7	usb0_data[7]	
10	i/o		8	usb0_nxt	
11	i/o		9	usb0_dir	
12	i/o		10	usb0_stp	
13	i/o		11	usb0_urstdrive	
14	i/o		12	usb0_reset	
15	i/o	GPIOE	27	em_bexcn	
16	i/o		26	em_brdyn	
17	i/o		25	em_sdram_dqm[3]	
18	i/o		24	em_sdram_dqm[2]	
19	i/o		23	em_sdram_dqm[1]	
20	i/o		22	em_sdram_dqm[0]	
21	i/o		21	em_writen	
22	i/o	GPIOD	31	em_data[31]	
23	i/o	GPIOE	0	em_ramsn[0]	
24	s	vdd			
25	s	vss			
26	s	dvss			
27	s	dvdd			
28	s	vdd_pll			
29	s	vss_pll			

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
30	s	avdd_pll			
31	s	avss_pll			
32	s			pad_a	
33	s	avss_tpll			
34	s	avdd_tpll			
35	s			pad_b	
36	s	avss_tpll			
37	i/o	GPIOE	20	em_read	
38	i/o		19	em_oen	
39	i/o		18	em_iosn	
40	i/o		17	em_romsn[1]	
41	i/o		16	em_romsn[0]	
42	i/o		15	em_mben[3]	
43	s	vdd			
44	s	vss			
45	i/o	GPIOE	14	em_mben[2]	
46	i/o		13	em_mben[1]	
47	i/o		12	em_mben[0]	
48	i/o		11	em_wrn[3]	
49	i/o		10	em_wrn[2]	
50	i/o		6	em_ramoen[2]	
51	i/o		5	em_ramoen[1]	
52	i/o		4	em_ramoen[0]	
53	i/o		3	em_ramsn[3]	
54	i/o		2	em_ramsn[2]	
55	i/o		1	em_ramsn[1]	
56	s	dvss			
57	s	dvdd			
58	s	vdd			
59	s	vss			
60	i/o	GPIOD	30	em_data[30]	
61	i/o		29	em_data[29]	
62	i/o		28	em_data[28]	
63	i/o		27	em_data[27]	
64	s	dvss			
65	i/o	GPIOE	9	em_wrn[1]	
66	i/o		8	em_wrn[0]	
67	i/o		7	em_ramoen[3]	
68	i/o	GPIOD	26	em_data[26]	
69	i/o		25	em_data[25]	
70	i/o		24	em_data[24]	
71	s	dvss			
72	i/o	GPIOD	23	em_data[23]	
73	i/o		22	em_data[22]	
74	i/o		21	em_data[21]	
75	i/o		20	em_data[20]	
76	i/o		19	em_data[19]	
77	i/o		18	em_data[18]	
78	i/o		17	em_data[17]	
79	i/o		16	em_data[16]	

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
80	i/o		15	em_data[15]	
81	i/o		14	em_data[14]	
82	s	vdd			
83	s	vss			
84	s	dvss			
85	s	dvdd			
86	i/o	GPIOD	13	em_data[13]	
87	i/o		12	em_data[12]	
88	i/o		11	em_data[11]	
89	i/o		10	em_data[10]	
90	i/o		9	em_data[9]	
91	i/o		8	em_data[8]	
92	i/o		7	em_data[7]	
93	i/o		6	em_data[6]	
94	s	vdd			
95	s	vss			
96	i/o	GPIOD	5	em_data[5]	
97	i/o		4	em_data[4]	
98	i/o		3	em_data[3]	
99	i/o		2	em_data[2]	
100	i/o		1	em_data[1]	
101	i/o		0	em_data[0]	
102	s	dvss			
103	s	dvdd			
104	s	vdd			
105	s	vss			
106	i/o	GPIOC	27	i2s_ws	
107	i/o		26	i2s_sd	
108	i/o		25	i2s_sck	
109	i/o		22	uart2_txd	
110	i/o		21	uart2_rxd	
111	i/o		20	em_sdram_casn	
112	i/o		19	em_sdram_rasn	
113	i/o		18	em_sdram_sdwen	
114	i/o		17	em_sdram_sdcsn[1]	
115	i/o		16	em_sdram_sdcsn[0]	
116	i/o		15	em_sdram_sdcke[1]	
117	i/o		14	em_sdram_sdcke[0]	
118	i/o		11	uart0_txd	
119	i/o		10	uart0_rxd	
120	i/o		9	spi0_ssl[2]	
121	i/o		8	spi0_ssl[1]	
122	s	dvss			
123	i/o	GPIOC	7	spi0_ssl[0]	
124	i/o		6	spi0_aready	
125	i/o		5	spi0_astart_o	
126	i/o		4	spi0_astart_i	
127	i/o		3	spi0_spisel	
128	i/o		2	spi0_miso	
129	i/o		1	spi0_mosi	

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
130	i/o		0	spi0_sck	
131	i/o	GPIOB	29	em_addr[29]	
132	i/o		28	em_addr[28]	
133	i/o		27	em_addr[27]	
134	i/o		26	em_addr[26]	
135	s	dvss			
136	i/o	GPIOB	25	em_addr[25]	
137	i/o		24	em_addr[24]	
138	i/o		25	em_addr[23]	
139	i/o		24	em_addr[22]	
140	i/o		23	em_addr[21]	
141	i/o		20	em_addr[20]	
142	i/o		19	em_addr[19]	
143	i/o		18	em_addr[18]	
144	i/o		17	em_addr[17]	
145	i/o		16	em_addr[16]	
146	s	vdd			
147	s	vss			
148	s	dvss			
149	s	dvdd			
150	i/o	GPIOB	15	em_addr[15]	
151	i/o		14	em_addr[14]	
152	i/o		13	em_addr[13]	
153	i/o		12	em_addr[12]	
154	i/o		11	em_addr[11]	
155	i/o		10	em_addr[10]	
156	i/o		9	em_addr[9]	
157	i/o		8	em_addr[8]	
158	i/o		7	em_addr[7]	
159	i/o		6	em_addr[6]	
160	s	vdd			
161	s	vss			
162	i/o	GPIOB	5	em_addr[5]	
163	i/o		4	em_addr[4]	
164	i/o		3	em_addr[3]	
165	i/o		2	em_addr[2]	
166	i/o		1	em_addr[1]	
167	i/o		0	em_addr[0]	
168	i/o	GPIOA	29	pwm_out[3]	
169	i/o		28	pwm_out[2]	
170	i/o		27	pwm_out[1]	
171	i/o		26	pwm_out[0]	
172	i/o		25	uart1_rtsn	
173	i/o		24	uart1_ctsn	
174	i/o		23	uart1_txd	
175	i/o		22	uart1_rxd	
176	i/o		21	i2c0_sda	
177	i/o		20	i2c0_scl	
178	s	dvss			
179	s	dvdd			

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
180	s	vdd			
181	s	vss			
182	i/o	GPIOA	19	eth0_reset	
183	i/o		18	eth0_mdint	
184	i/o		17	eth0_mdc	
185	i/o		16	eth0_mdio	
186	s	dvss			
187	i/o	GPIOA	15	eth0_rx_clk	
188	i/o		14	eth0_rxd[3]	
189	i/o		13	eth0_rxd[2]	
190	i/o		12	eth0_rxd[1]	
191	i/o		11	eth0_rxd[0]	
192	i/o		10	eth0_rx_er	
193	i/o		9	eth0_rx_dv	
194	i			en_rcfg	
195	s	dvss			
196	s	dvdd			
197	i			trst	JTAG (IEEE 1149.1)
198	i			tck	
199	i			tms	
200	i			st[0]	
201	o			cr[0]	
202	s	dvss			
203	i			tdi	JTAG (IEEE 1149.1)
204	o			tdo	
205	i			wakeup	
206	i			prom_width[0]	
207	i			nmi	
208	i			reset	
209	i			rref	
210	o			vref	
211	o			dac[0]	
212	o			dac[1]	
213	s	avss_dac			
214	s	avdd_dac			
215	s	vdd_conv			
216	s	vss_conv			
217	o			vref_adc	
218	i			adc[0]	
219	i			adc[1]	
220	i			adc[2]	
221	s	avss_adc			
222	s	avdd_adc			
223	i			adc[3]	
224	i			adc[4]	
225	i			adc[5]	
226	i			adc[6]	
227	i			adc[7]	
228	i			start_addr[1]	
229	i/o		8	eth0_rx_crs	

GPIOA

№	Тип	Порт	Бит порта	Альтернативная функция	Примечания
230	i/o		7	eth0_tx_clk	
231	i/o		6	eth0_txd[3]	
232	i/o		5	eth0_txd[2]	
233	i/o		4	eth0_txd[1]	
234	i/o		3	eth0_txd[0]	
235	s	dvss			
236	s	dvdd			
237	s	vdd			
238	s	vss			
239	i			osc_main_in	
240	o			osc_main_out	
241	s	dvss			
242	i/o		2	eth0_tx_er	
243	i/o	GPIOA	1	eth0_tx_en	
244	i/o		0	eth0_rx_col	
245	i			osc32k_in	
246	o			osc32k_out	
247	i			usb_clk	
248	i/o		29	rtc0_a	
249	i/o		28	i2c1_sda	
250	i/o		27	i2c1_scl	
251	i/o		20	spi1_spisel	
252	i/o	GPIOF	19	spi1_miso	
253	i/o		18	spi1_mosi	
254	i/o		17	spi1_sck	
255	i/o		14	uart3_txd	
256	i/o		13	uart3_rxd	

8.2 Processor output diagram in QFP256 package

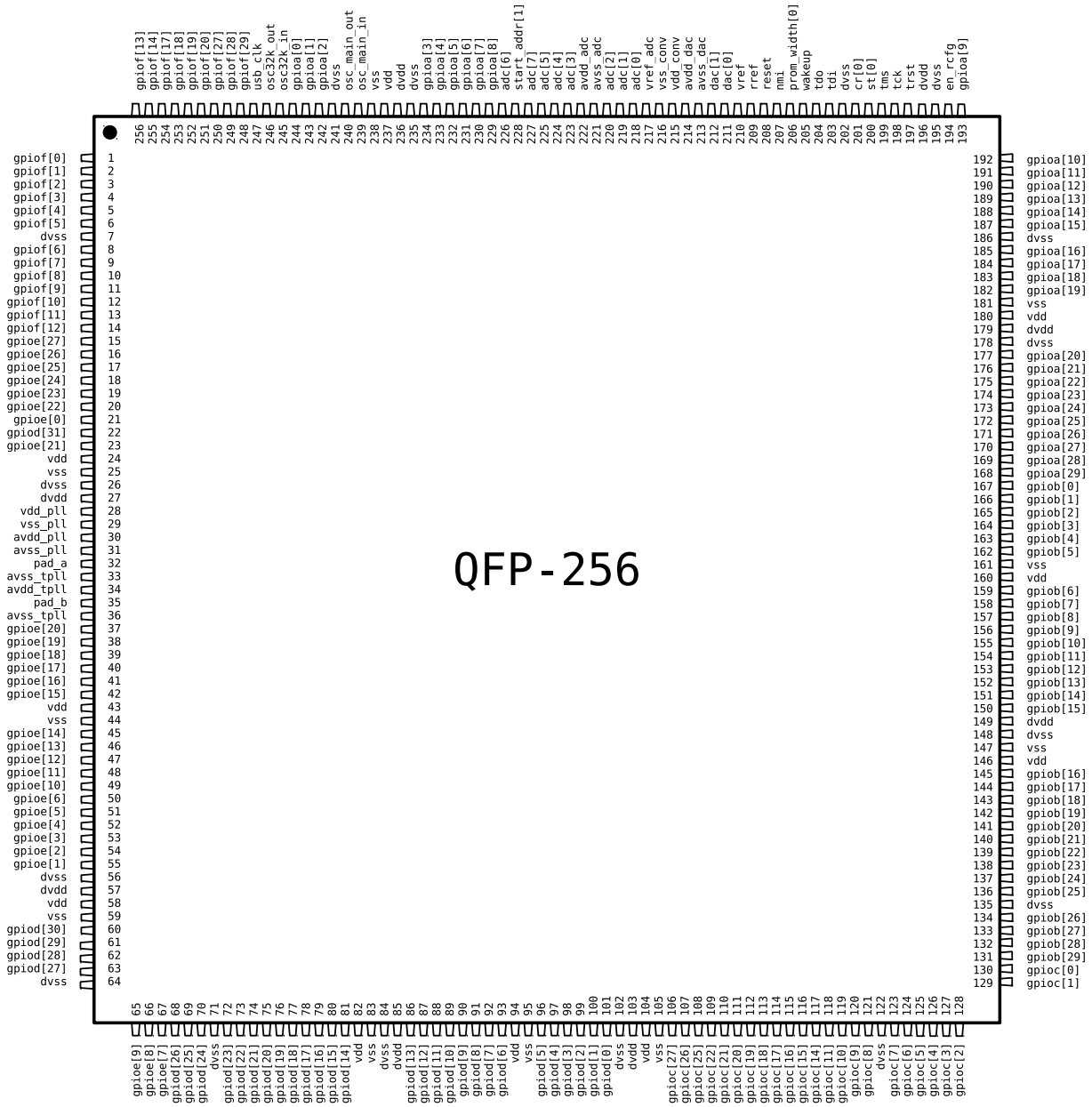

Figure 21: Processor input/output diagram

9 Electric characteristics

9.1 Electrical characteristics of input-output ports

Table 14: Electrical characteristics of input-output ports

Parameter		conditions	Min.	Typ.	Max.
V_{IL}	Low-level input voltage, V	CMOS, LVTTL	$-0,3$		$0,8$
V_{IH}	High-level input voltage, V		$2,0$		5.5
I_{IL}	Low-level input current, mcA	$V_{in} = V_{SS}$	-10		10
I_{IH}	High-level input current, mcA	$V_{in} = DVDD$	-10		10
V_{OL}	Low-level output voltage, V	$I_{OL} = -12$мА	0		$0,4$
V_{OH}	High-level output voltage, v	$I_{OH} = 12$мА	2.4		3.6
	High-level raising register, kOhm		$68,2$		$118,1$
	Low-level raising register , kOhm		$30,2$		$80,6$