MCp0411100101 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.

Peculiarities:

- Cell amount — 4
- Processor word — 32/64 бита
- DM — 128KiB (4*4K*64)
- PM — 128KiB (4*4K*64)
- ROM — 256KiB [1]
- operation block under floating point numbers (in each cell)
- Clock frequency — 100 MHz
- Processor performance — 2,4 GFLOPS;

General characteristics:

- Package — QFP-208
- Operation environment – (-60 . . . +125)
- Maximum power consumption:
- Cores — 1.08W with frequency 100MHz
- Peripherals — 80мВт
- Supply voltage (separate): cores — 1,8V, peripheral — 3,3V

Peripheral devices:

- 2 SPI interfaces with slave devices' selector (in "master" mode)
- 4 universal asynchronous receiver/-transmitters 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 1.1 FS (device) controller with continious external interface for receive-transmit connection
- RTC with calendar
- 7 GP timers
- 4 input-output ports, total amount of input-output ports – 104
- PWM controller with 4 channels
- watchdog timer

[1] information about supporting ROM в п. 2

Contents

1 Introduction

The document describes the features and capabilities of the microprocessor MSp0411100101 in plastic QFP-208 and ceramic-metal SQFP-240 (4245.240-6) packages, its internal organization, storing devices' address space, and gives an overview of the microprocessor peripherals.

Multicellular microprocessor MCp0411100101 in plastic QFP-208 package is designed to implement a wide range of control tasks and signal digital processing in applications that require minimal power consumption and high performance, such as:

- industrial automation systems from smart sensors to engine control system;
- universal navigation receivers GLONASS/ GPS/ Galileo/ COMPASS(China)/ IRNSS(India)/ QZSS(Japan);
- mobile phones;
- 3D video technology;
- in-car electronics for smart on-board systems, which control traffic conditions and warn a driver about danger and traffic jams;
- safety systems automatically detecting "insiders" and "outsiders".

Multicellular processor MCp0411100101 in ceramic-metal CQFP- 240 (4245.240-6) package is designed for special applications requiring extra resistance.

2 Type codes and abbreviations

2.1 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;

PU — peripheral unit;

DDC — direct digital control;

PBR — physical block RAM;

PM — program memory;

DM — data memory;

LPF — low-pass filter;

ACEM — access controller to external memory;

DMA — direct memory access;

GPR— general purpose register (s);

2.2 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

3 Description

3.1 General technical characteristics

Conventions	**MCp041p100101**
General purpose of function	64-bit microcomputer
Multicellular core	4 cells 32/64 bits
Floating point arithmetic block (corresponds ieee754)	single precision
Peak performance	24 MFLOPS/MHz
Internal scratch-pad store, PM/DM KiB	128/128
User inputs-outputs, units	104
USB 1.1 FS device, units	1
UART, units	4
SPI, units	3
I2C	1 leading, 1 slave
I2S	1 leading (received data)
PWM, channels	4
RTC with calendar, units	1
Ethernet 10/100, units	1

Table 2: List of recommended FLASH ROM to use with processor MCp0411100101

№ Section	**Nomenclature**	**Producer**	**Volume, Mb**	**Temperature range**	**Recommended by Russian Ministry of Defense** [1]
1	XCF04S	Xilinx	4	–40°C ... +85°C	no
2	XCF08P	Xilinx	8	–40°C ... +85°C	no
3	XCF16P	Xilinx	16	–40°C ... +85°C	yes
4	XCF32P	Xilinx	32	–40°C ... +85°C	yes

Notes:

[1] "Rationally unified and optimized foreign-made ECB nomenclature for application in REA «Nomenclature 2012» Book 2

3.2 MP structure

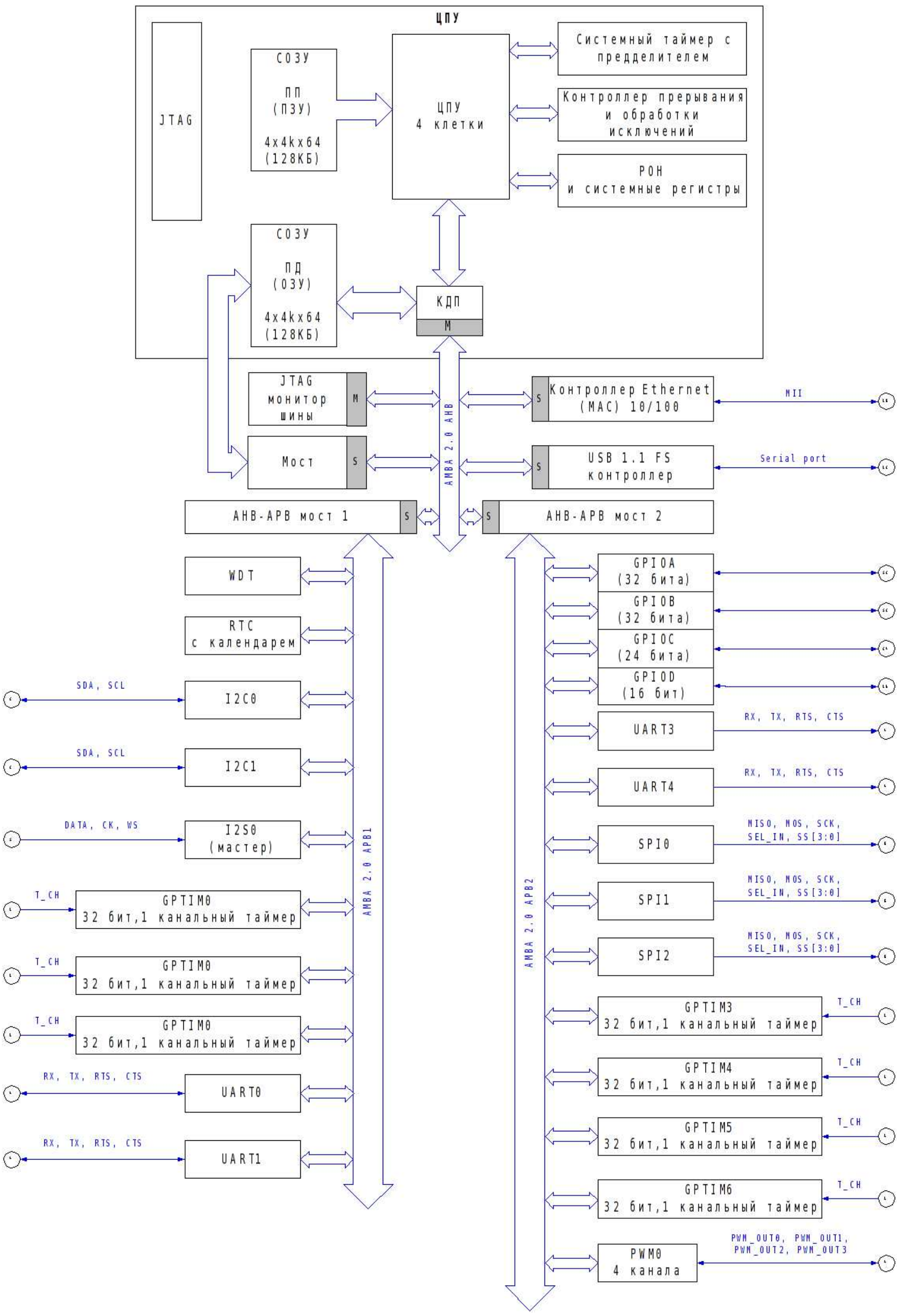

Figure 1: General structure MCp041p100101

DDC:

- interrupt controller with exception handling device;

- core, designed for calculations and control functions implementation;
- system timer;
- in-curcuit debugging modules;
- data and system bus access interfaces;
- system registers and general purpose registers.

Peripheral bus:

- For description of peripheral bus linking see Section 5

4 Central processor unit

4.1 Electronic core

The core includes:

- 4 processing blocks (PB), having numbering [0, 3];
- commutation environment, combining PC;
- field of system registers and general purpose registers (GPR);
- commutator for DM and PU access;

MP core's structural scheme is displayed on fig. 2

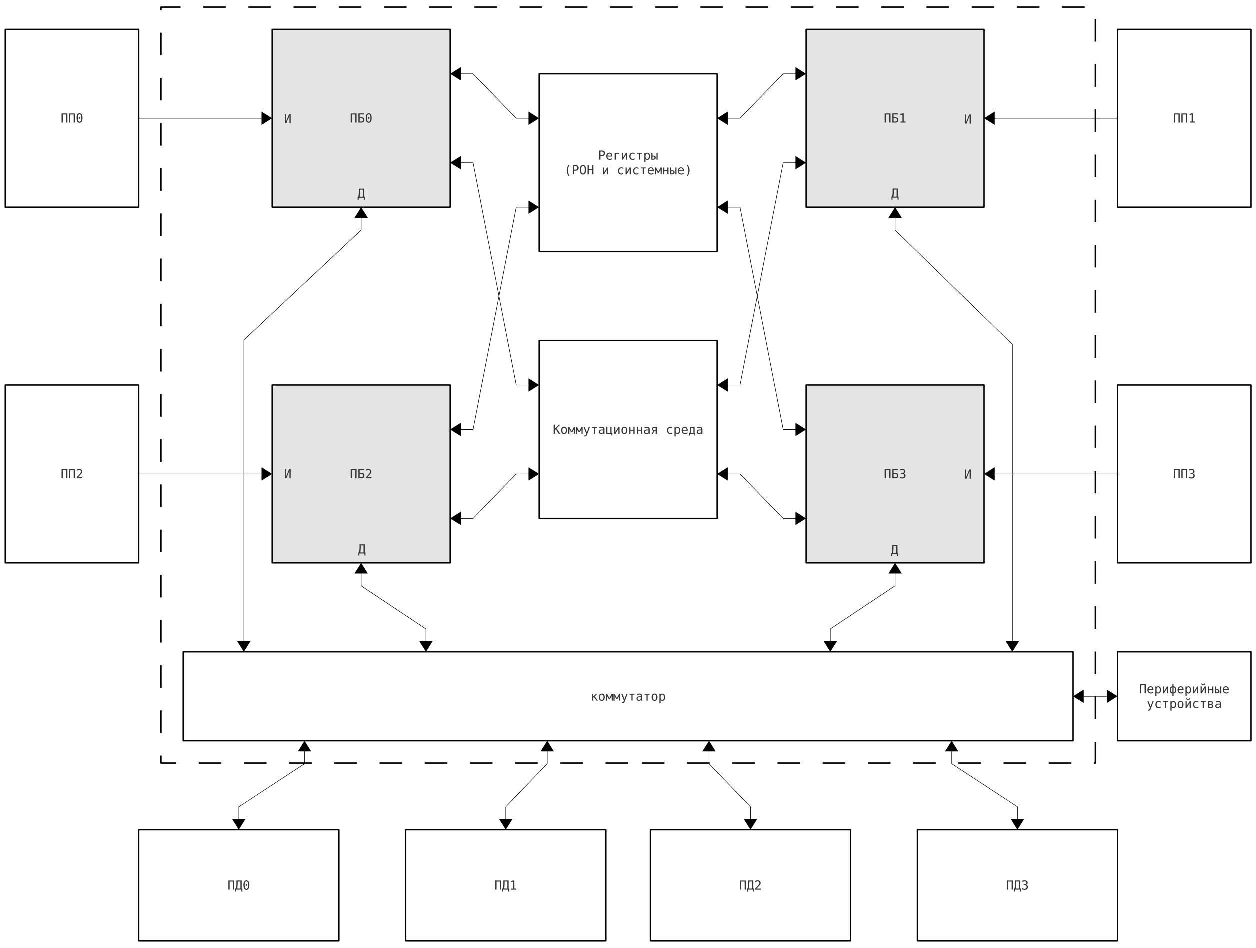

Figure 2: Core's structural scheme

PB is a set of control and execution units, has developed command system. PB includes integer ALU and processing block with single precision floating-point. Fig. 2 shows that PBs have independent data and instruction channels, GPR outputs, system registers, and inter-connected commutation environment.

Data is only received through instruction channel, there are no opportunities to write PM. Transitions to all directions are possible through data channel. DM blocks are accessable through commutator.

4.1.1 Registers

All the registers, except the system ones have a width of 64 bits. Registers' reading / writing is implemented via specialised commands.

Register types	**Number**
General purpose registers	
	0-7
reserved	8-31
Index registers	
	32-47
Control registers	
PSW	48
INTR	49
MSKR	50
ER	51
IRETADDR	52
STVALR	53
STCR	54
IHOOKADDR	55
INTNUMR	56
MODR	57

4.1.1.1 Index registers

are used for indirect addressing and have the following logical structure:

Bit numbers	**63..48**	**47..32**	**31..0**
	Index(Index)	Mask(Mask)	Base (Base)

In general, when using the register of this type as an operation argument, the value of the argument is generated according to the following algorithm:

- calculation of executive address:

$$Address = Index + Base$$

- data memory addressing at the executive address "Address" to read argument value in accordance with the type of used operation. Modification of index register's value is performed hardwarily upon completion of the paragraph in case when relevant bit PSW(MODR) is set in accordance with the following formula:

$$Index = ((Index|\overline{Mask}) + 1)\&Mask$$

where | — bit-to-bit operation «OR», & — bit-to-bit operation «AND», $\overline{X}$ — bit-to-bit operation of reversing

4.1.1.2 Control registers

Processor includes the following control registers:

Register	Register number	Access	Description
PSW	30h	RW	Control register
INTR	31h	RW	Interrupt register
MSKR	32h	RW	Mask interruption register
ER	33h	RC	Error register
IRETADDR	34h	R	Return address register
STVALR	35h	RW	Counter period
STCR	36h	RW	Counter control register
IHOOKADDR	37h	RW	Register of initial interrupt handler's address
INTNUMR	38h	R	Number of formulated interruption
MODR	39h	RW	Register of index regisers' modification

4.2 Interrupt controller

MP interrupt system enables handling of 37 interrupts. Source with the number "0" is of the highest priority during interrupt processing. Work with interruptions in MCp0411100101 has features described in section A.

0	Unmasked internal interruption (INMI)
1	Unmasked external interruption (ENMI)
2	Unmasked exclusion in hardware (PERE)
3	Unmasked program exclusion (PPGE)
4	
5	System timer interruption (STI0)
6	System timer interruption (STI1)
7	System timer interruption (STI2)
8	System timer interruption (STI3)
9	Software interruption (SWI0)
10	Software interruption (SWI1)
11	Software interruption (SWI2)
12	Software interruption (SWI3)
13	
14	
15	
16	Masked interruption from UART0
17	Masked interruption from UART1
18	Masked interruption from UART2
19	Masked interruption from UART3
20	Masked interruption from I2C0
21	Masked interruption from I2C1
22	Masked interruption from SPI0
23	Masked interruption from SPI1
24	Masked interruption from SPI2
25	Masked interruption from I2S0
26	Masked interruption from GPTIM0
27	Masked interruption from GPTIM1
28	Masked interruption from GPTIM2
29	Masked interruption from GPTIM3
30	Masked interruption from GPTIM4
31	Masked interruption from GPTIM5
32	Masked interruption from GPTIM6
33	Masked interruption from PWM0
34	Masked interruption from RTC
35	Masked interruption from GPIOA
36	Masked interruption from GPIOB
37	Masked interruption from GPIOC
38	Masked interruption from GPIOD
39	Masked interruption from ETHERNET0
40	Masked interruption from USB0
41	
...	
63	

4.2.0.3 Interrupt structure

Interrupts can be divided into two groups: system and peripheral. System interrupt occupy addresses from 0 to 15 in interrupt controller and are of a more priority in comparison with peripheral. System interrupts with numbers from 0 to 4 are unmasked. Peripheral interrupts occupy addresses from 16 to 63.

4.2.1 Interrupt controller structure

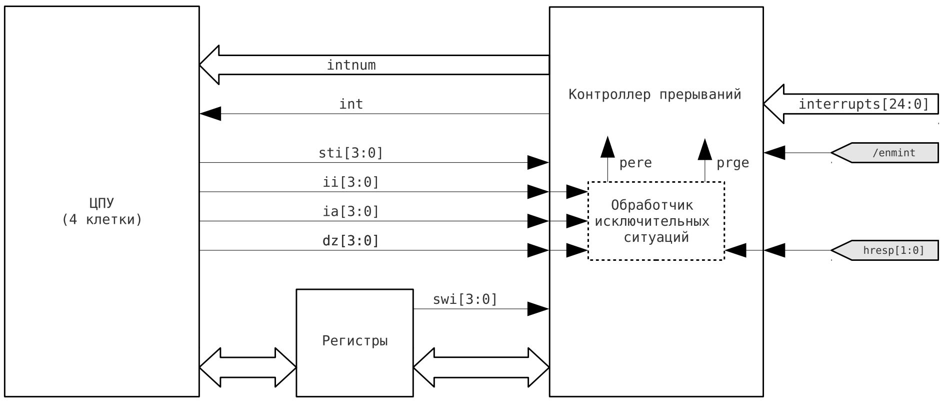

Figure 3: Interrupt controller block diagram

Interrupt controller performs the following functions:

- detects interrupt of the highest priority on each clock and forms its number on bus INTNUM;
- processes information about software and hardware faults, transmits signals of their occurance into interrupt controller, what leads to interrupt request formation.

3 error signal groups are transmitted from CPU to interrupt controller during execution program under which signal PRGE is formed in exception handler:

- DZ – divide by zero implementation trial;
- II – nonexistent instruction is selected;
- IA – nonexistent address is formed.

hresp signals are transmitted from peripheral device bus to an exception handler, which line state informs about errors or their absence when accessing to bus. PERE signal occurs in case of error.

Each CPU also generates SWT signal - software timer interrupt, and SWI-software interruption Interrupt request signals from each peripheral device (interrupts [24:0]) are also received from peripheral device bus. From MP input signal /enmi is received - external unmasked interruption.

4.2.2 Interrupt controller register

The following registers are designed for interrupt system operation and interrupt handling program functioning:

Register	Access	Description
FORCE	*W*	Interrupt setting register.
INTR	*RW*	Interrupt register.
MASK	*RW*	Interrupt mask register.
ER	*R*	Error register.
INTNUM	*R*	Received interrupt number register.

FORCE register is write-only. Writing 1 to any bits of this register forms a single pulse at interrupt register input which in turn causes relevant bit setting in the INTR register. On the next clock cycle after writing 1 to any bits of this register, all bits of this register will be reset to 0.

INTR register is available for both reading and writing. Writing 1 to any bits of this register will cause reset of relevant bits in this register to 0. This register is a latch register, and it captures come received data at its input until it will be reset by writing 1 to relevant bits.

MASK register is available for both reading and writing. To enable interrupts with certain numbers, you need to put 1 into relevant register bits.

Register ER is read-only, this register contains information about software and hardware failures.

INTNUM register is read-only, this register contains the number of the highest priority interrupt among those waiting for processing at the given moment.

4.2.3 Interrupt handling order

4.3 System timer

System timer is designed to generate set periodic or single time slots. The timer is decremented spinner with clock signal divider at the input. Spinner's initial value is recorded in STVALR register, control through STCR register. Interrupt processing request is formed after specified time slot completion.

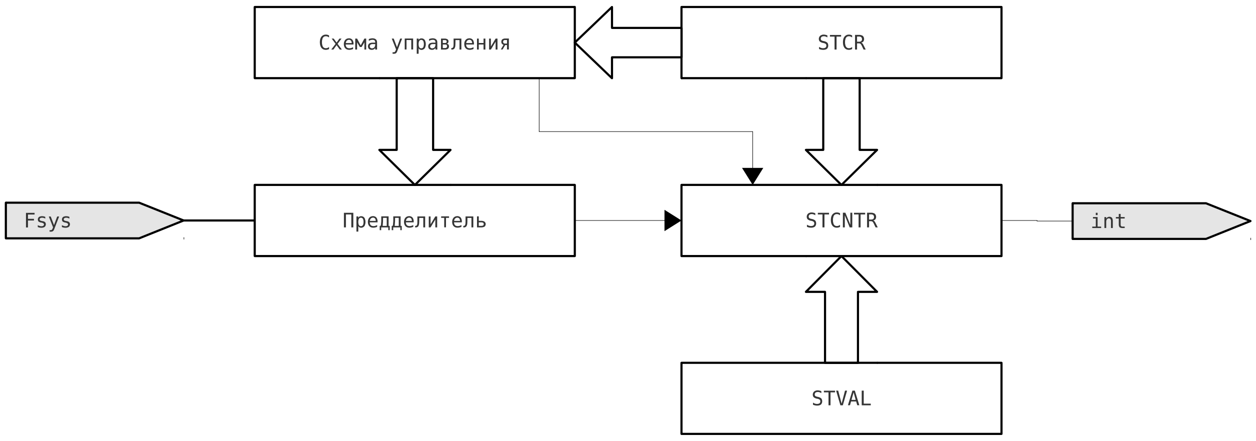

Figure 4: system timer block diagram

Fig.4 shows timer block diagram. Below are formulas for calculating intervals' frequency and period formed

- formed interval period:

$$T = T_{clk} \cdot PREDIV \cdot CNTVAL$$

- time slot passing frequency:

$$F = \frac{F_{clk}}{PREDIV \cdot CNTVAL};$$

bypassing the divider. Therefore in the formula substitute 1 instead PREDIV .

4.3.1 Operation mode

- Single time slot formation – Timer is started by the user (в бит STCR(EN)='1'), and when timer spinner reaches value «0», timer issues interrupt processing request, thereafter '0'is set in STCR(EN) bit, and timer stops until the next value write '1'in STCR(EN);

- Periodic time slot generation – The timer is started and stopped by the user (into STCR(EN) bit corresponding value is written). When timer reaches value «0», timer issues interrupt handling request, spinner is being reloaded with the value set by user in STVALR register and timer operation continues until the user writes the value in STRCR(EN) bit.

When writing in STVALR register new value will be transferred to the spinner at its next reboot, when it reaches "0". If, during operation register STCR is changed, then the timer will stop immediately and start with the new parameters. It is strongly recommended before changing timer operation mode first to stop it writing STCR(EN)='0', then to set the new values in STCR register. It is forbidden to change STVALR register's value into 0 and change STCR after that. This may cause interrupt handling request.

5 Peripheral devices

5.1 Input-output port (GPIO)

5.1.1 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;

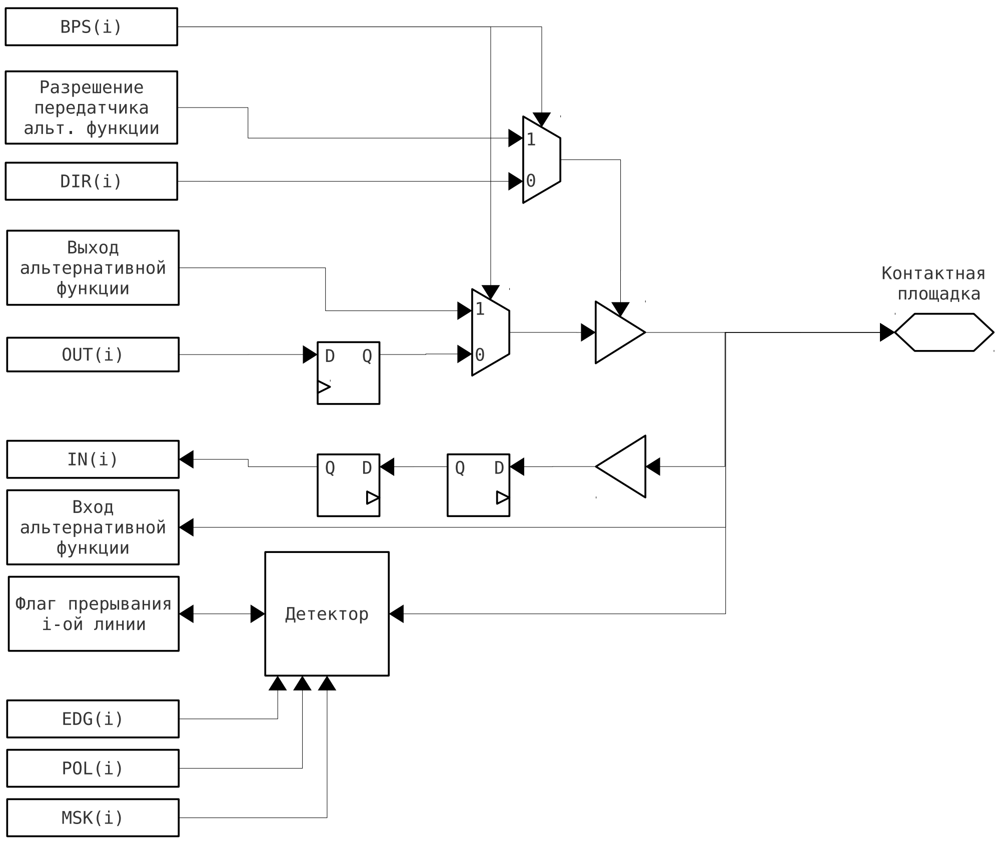
BPS(i)
Разрешение передатчика альт. функции
DIR(i)
Выход альтернативной функции
OUT(i)
IN(i)
Вход альтернативной функции
Флаг прерывания i-ой линии
EDG(i)
POL(i)
MSK(i)
Детектор
Контактная площадка
1
0
D Q
Q D

Figure 5: Block diagramm of i line GPIO

5.1.2 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 6

5.1.3 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.

5.2 UART(UARTx) interface

5.2.1 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);

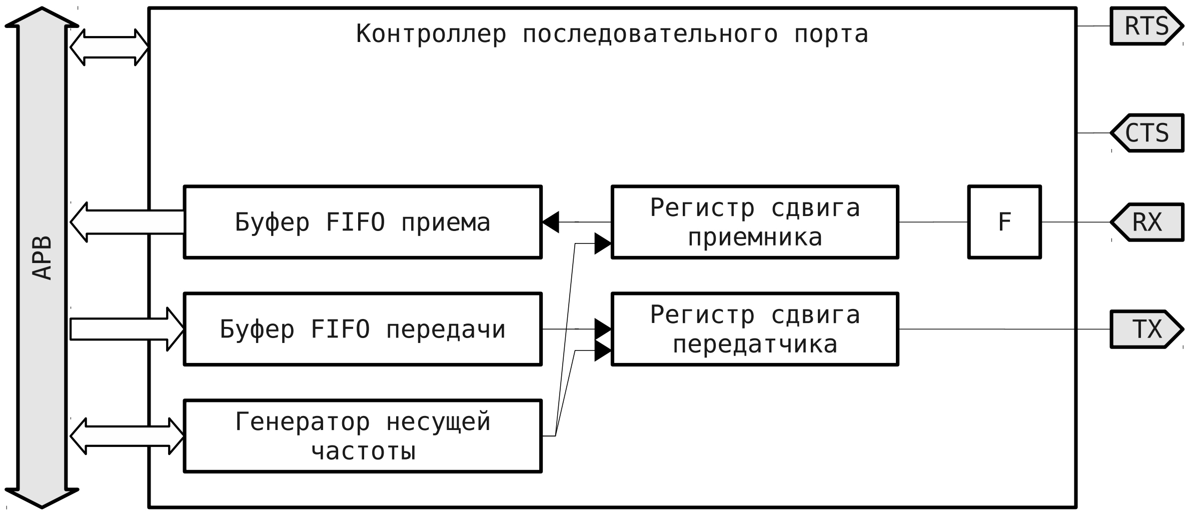
Контроллер последовательного порта
APB
Буфер FIFO приема
Буфер FIFO передачи
Генератор несущей частоты
Регистр сдвига приемника
Регистр сдвига передатчика
F
RTS
CTS
RX
TX

Figure 6: Block diagram UARTx

Fig. 6 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.

5.2.2 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. 7 shows possible transmitted data format, for given controller.

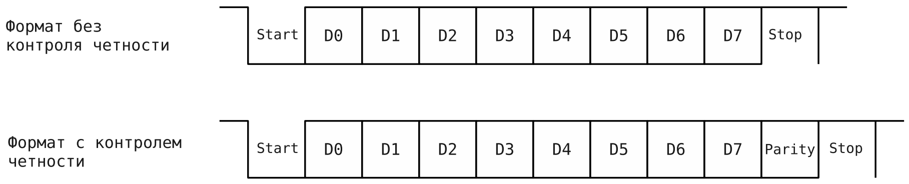

Figure 7: Data burst format

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'.

5.2.3 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'.

5.2.4 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).

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

5.2.5 Self-test modes

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

5.2.5.1 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.

5.2.5.2 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.

5.2.6 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;

5.2.6.1 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.

5.2.7 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

5.3 SPI interface (SPIx)

5.3.1 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.

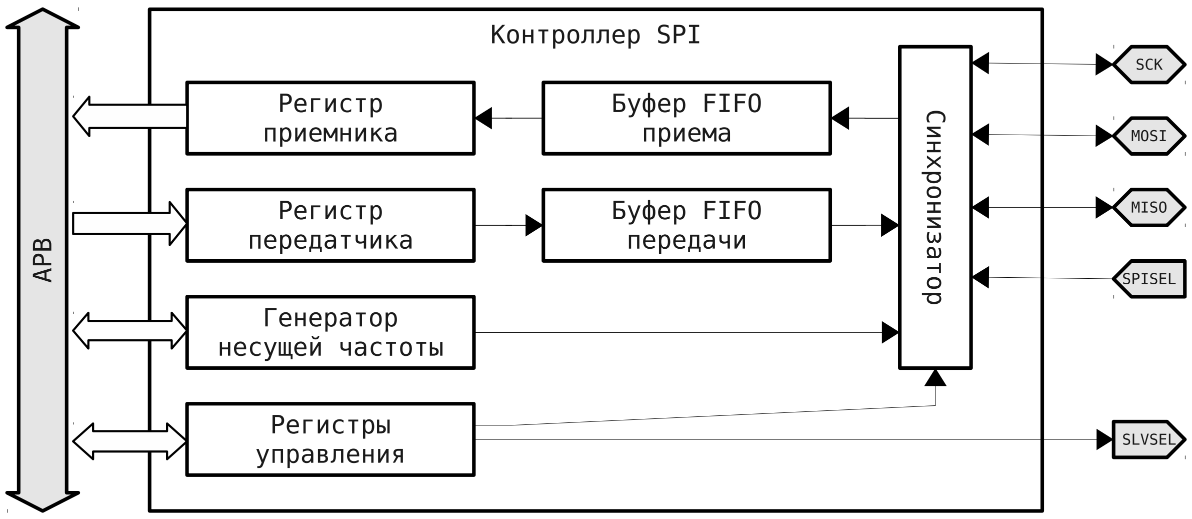

Figure 8: Block diagram SPIx

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. 9 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 5.3.6.

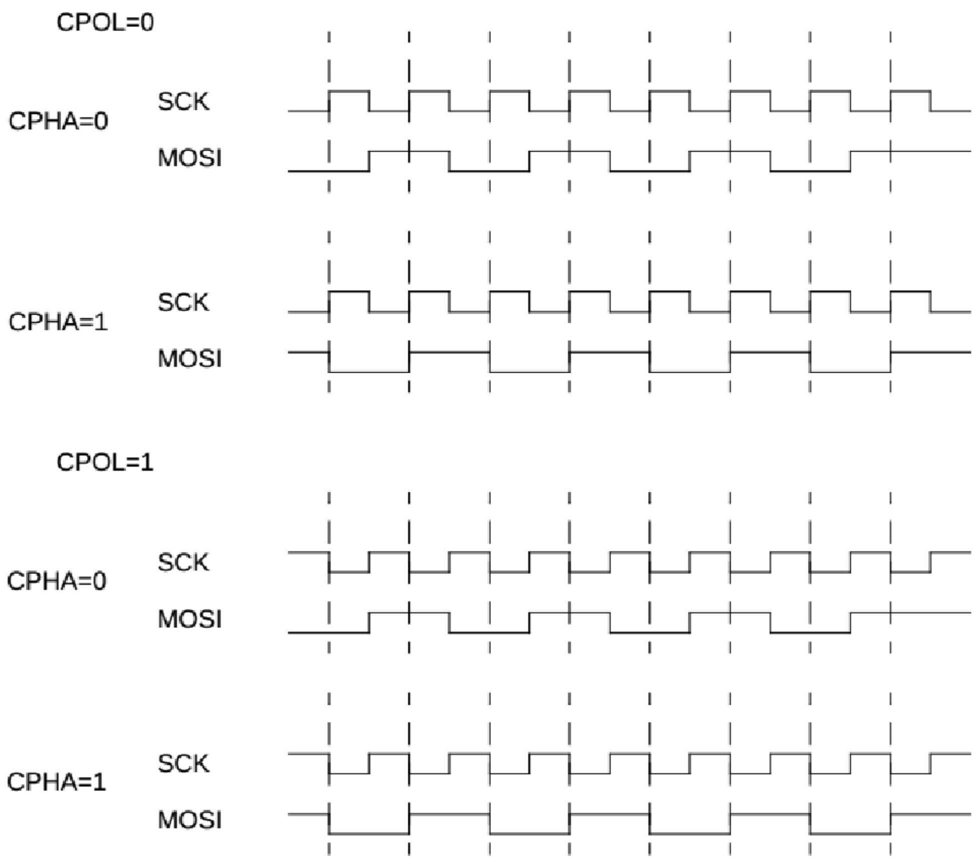

Figure 9: SPI operation mode

5.3.2 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.

5.3.3 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.

5.3.4 SCK clock signal

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

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

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

5.3.5 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.

5.3.6 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.

5.3.7 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

5.4 Interface I^2C«master» (I2C0)

5.4.1 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.

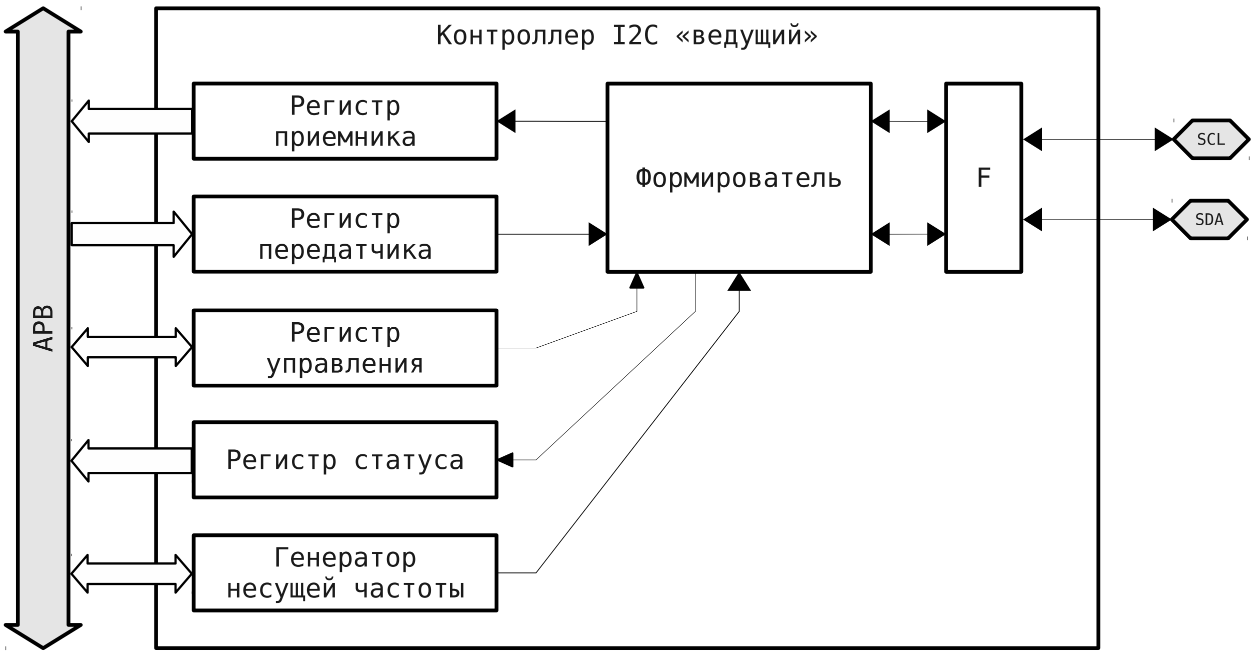

Figure 10: block diagram I2C0 («master»)

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

5.4.2 General description of receive-transmit protocol

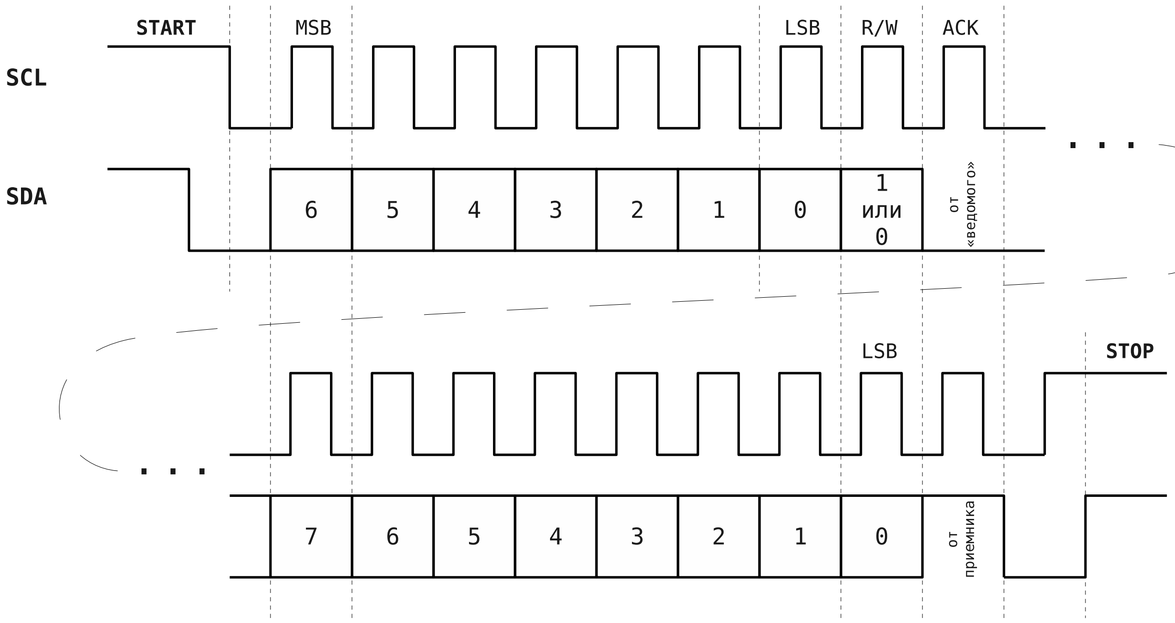

Figure 11: Transaction at the bus I^2C

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. 13 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.

5.4.3 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.

5.4.4 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.

5.4.4.1 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.

5.4.4.2 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;

5.4.5 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

5.5 Interface I^2C «slave» (I2C1)

5.5.1 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.

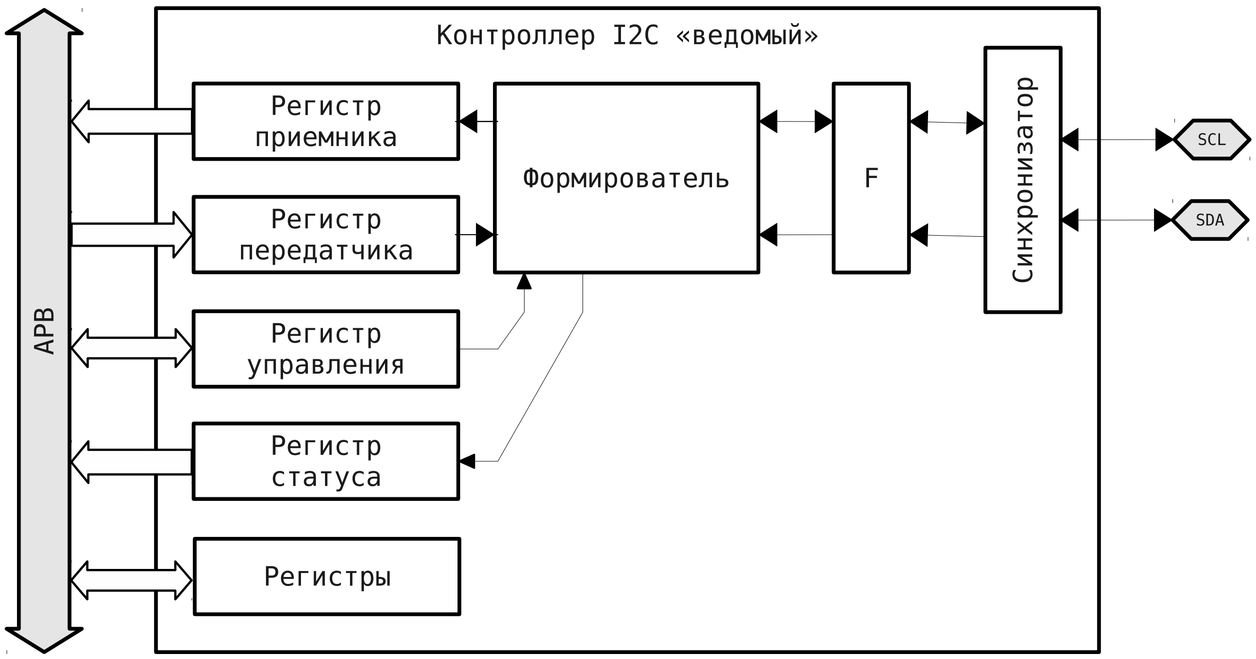

Figure 12: block diagram I2C1 («slave»)

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

5.5.2 General description of receive-transmit protocol

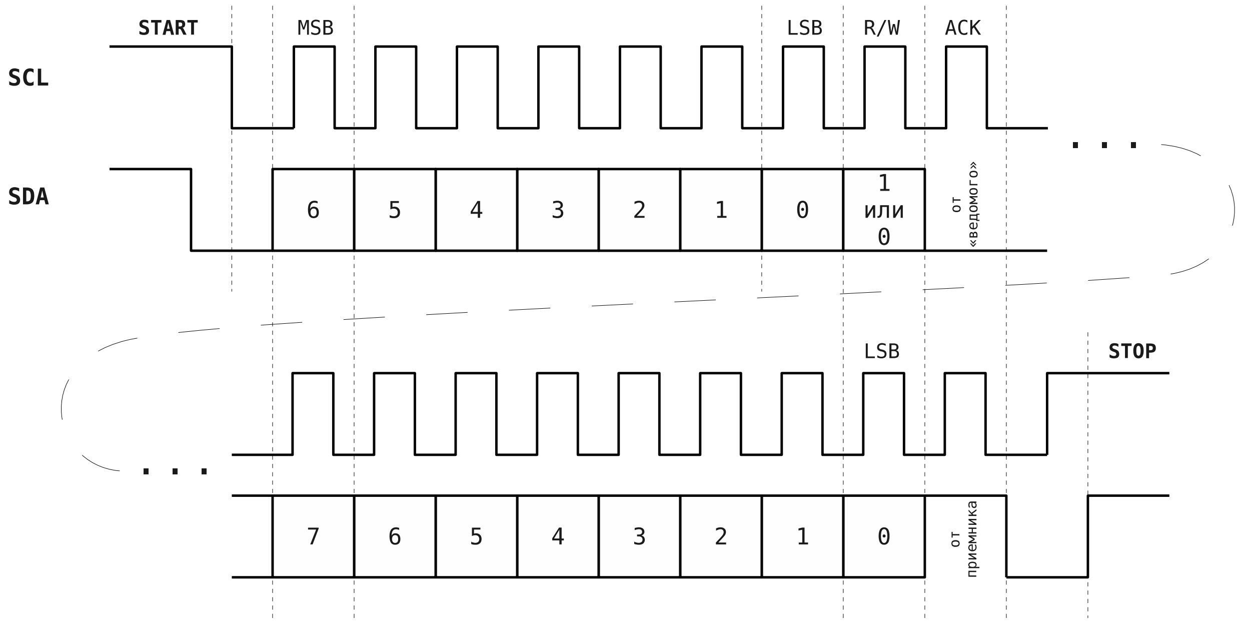

Figure 13: Transaction on the bus I^2C

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. 13 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».

5.5.3 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 $F_{sys} \geq 6MHz$.

5.5.4 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.

5.5.4.1 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).

5.5.4.2 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.

5.5.5 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

5.6 Controller I^2S(I2Sx)

5.6.1 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;

5.6.2 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. 14).

Time signal diagram I^2Srepresented on fig. 15:

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

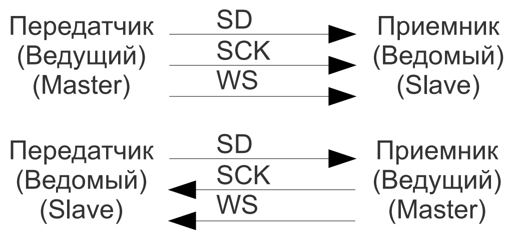

Figure 14: Signal direction I^2S

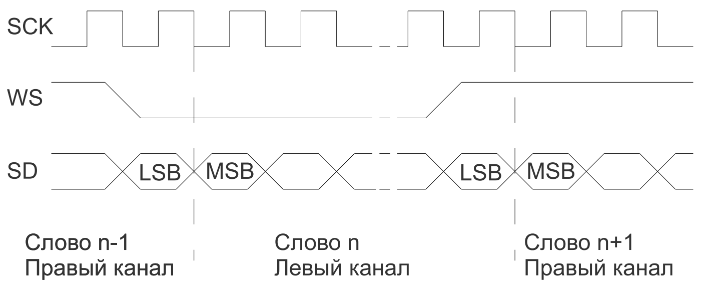

Figure 15: Time signal diagram I^2S

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

5.6.3 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

5.7 general purpose timer (GPTIMx)

5.7.1 Brief characteristics

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

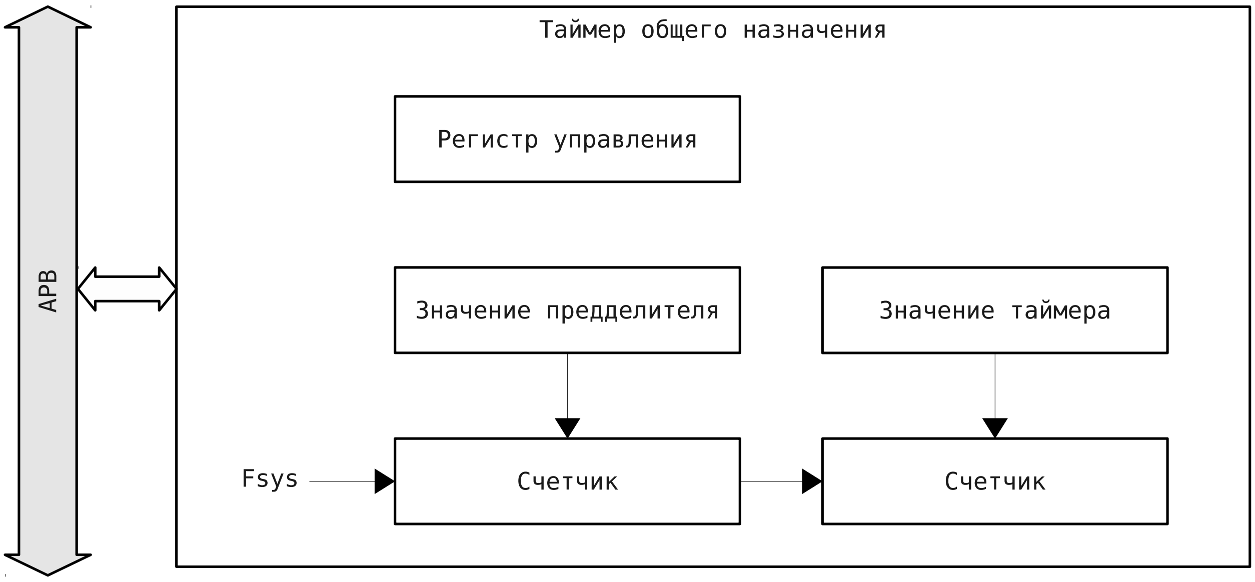

Figure 16: Block diagramm GPTIMx

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. 16 represents block diagramm GPTIMx.

5.7.2 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 (CNT-PER) 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.

5.7.3 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)

5.8 Controller Ethernet(Ethernet0)

5.8.1 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 fщr 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.

5.8.2 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.

5.8.3 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

5.8.4 Transmitter DMAC

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

5.8.4.1 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*

5.8.4.2 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.

5.8.4.3 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.

5.8.4.4 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.

5.8.5 Receiver DMAC

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

5.8.5.1 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*

5.8.5.2 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.

5.8.5.3 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.

5.8.6 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
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 recorderd
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 recorderd
3-9	DESCPNT	descriptor indicator, is automatically incremented when new data batch is received
0-2	—	*reserved*

5.9 USB(USBx) controller

5.9.1 Brief characteristics

- implements USB 1.1 FS, Fairchild USB1T11A circuit compatible;
- operates only in “device” mode;
- 4 channel classes supported: in-line, control, isochronal, interrupt;
- supports LS (1,5 Mb/s) and FS (12 Mb/s) modes;
- includes 4-channel exchange buffer (FIFO);
- external clock generator 48 MHz is required;

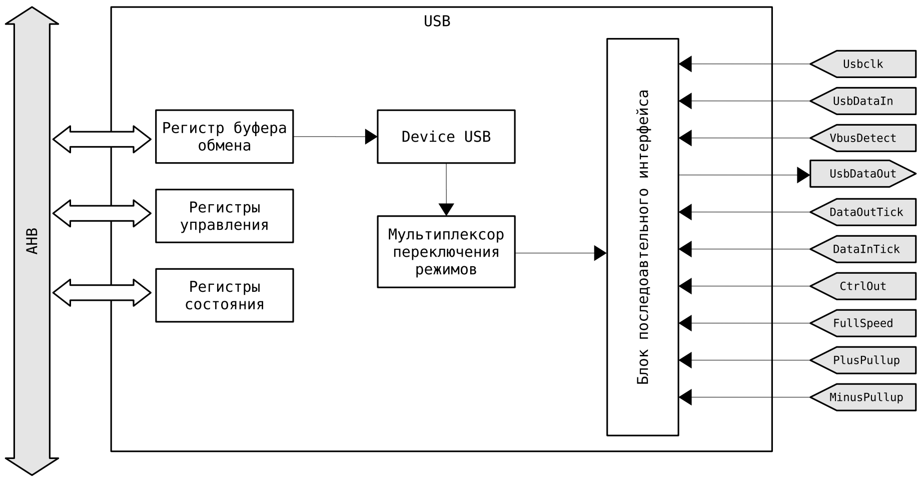

Figure 17: USBx block diagram

USB 1.1 interface operates in «device» mode only. If connection to USB host is lost, the connection status is displayed as a reset in register USBxLNST. When connected to the USB host, then connection status and connection speed will be displayed in USBxLNST register.

If transaction is detected, then USB (Device) is to be ready to accept the transaction. If the incoming data burst is detected, it is necessary to set operation enabling bit to the endpoint (EndPoint) and wait until transaction is completed, i.e. USBxINT (DONE) bit setting (or wait interrupt under bit USBxINT (DONE)). If transaction is detected, including outbound parcel, the data must be loaded into endpoint FIFO before operation enabling bit setting.

Change of the following parameters is allowed:

- USB operation speed;

- USB polarity;
- USB address (on default 0);
- global endpoint operation enabling setting;

5.9.2 Registers' description

Base address USB - 0xFFF1 4000.

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

Register	Address displacement	Access	Description
USBxHSCR	00h	RW	Control register
USBxEPCRn	08,18,28,38h	RW	Control register EPn
USBxEPSTn	0C,1C,2C,3Ch	RW	State register EPn
USBxEPTRSTn	10,20,30,40h	R	Connection condition register EPn
USBxEPNTRSTn	14,24,34,44h	RW	NACK connection condition register EPn
USBxCR	48h	RW	Controller control register
USBxLNST	4Ch	R	Controller connection condition register
USBxINT	50h	RW	Interrupt controller register
USBxMSKINT	54h	RW	Interrupt mask register
USBxADDR	58h	R	Device address register
USBxMSPFRAME	5Ch	RW	SOF packet counter most significant bits
USBxLSPFRAME	60h	RW	SOF packet counter least-significant bits
USBxEPRXDATAn	6A,7C,94,ACh	R	EPn receiver buffer
USBxEPRXMSBn	68,80,98,B0h	RW	EPn receiver buffer most significant bit
USBxEPRXLSBn	6C,84,9C,B4h	RW	EPn receiver buffer least-significant bit
USBxEPRXCLRn	70,88,A0,B8h	W	EPn receiver buffer control register
USBxEPTXDATAn	74,8C,A4,BCh	W	EPn receiver buffer
USBxEPTXCLRn	78,90,A8,C0h	W	EPn receiver buffer control register

USBxHSCR	**Control register**		
Bit number	31 … 2	1	0
Initial state	0	0	0
Description	–	RESET	-

2-31	—	*reserved*
1		USB controller reset, 10 system frequency clock required for reload('1' – reset)
0		*reserved*

USBxEPCRn	**EPn control register**					
Bit number	31 … 5	4	3	2	1	0
Initial state	0	0	0	0	0	0
Description	–	ISO	STALL	OUT	READY	EN

5-31	—	*reserved*
4	ISO	isochronal exchage enabled, data (received/transmitted) are not acknowledged ACK ('0' – forbidden, '1' – permitted)
3	STALL	STALL signal transmission, if Host initialized data exchange ('1' – STALL signal to transmit)
2	OUT	'0' - reply is implemented by DATA1 packet, '1' - reply is implemented by DATA0
1	READY	EP rediness to receive request, automatically cleansed after exchange completion ('1' – EP ready)
0	EN	EndPoint operation enabling ('0' – forbidden, '1' – permitted)

USBxEPSTn	EPn 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
Description	—																								SEQ	ACK	STALL	NAK	TIME	OVF	STUFF	CRC

		For all register bits '1' - attribute presence, '0' - attribute absence
8-31	—	*reserved*
7	SEQ	if the last transmission was OUT_TRANS type, then bit shows, where the last packet is placed(DATA0 = 0, DATA1 = 1)
6	ACK	ACK received from Host
5	STALL	STALL sent to Host
4	NAK	NACK sent to Host
3	TIME	no reply from Host
2	OVF	not enough space in receiver buffer
1	STUFF	data structure error
0	CRC	CRC error

USBxEPTRSTn	EPn condition 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	
Description	—																														TYPE	

2-31	—	*reserved*
0-1	TYPE	Last transaction type (in case when EP was ready) (00 – SETUP, 01 – IN, 10 – OUT_DATA)

USBxEPNTRSTn	EPn connection condition NACK 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	—																														TYPE	

2-31	—	*reserved*
0-1	TYPE	Last transaction type, completed with NACK transmit into Host (00 – SETUP, 01 – IN, 10 – OUT_DATA)

USBxCR	Controller 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
Description	—																									PULL	SPEED	POL	DIRCR	LINEST		EN_GLOB

7-31	—	*reserved*
6	PULL	raising resistors' control D+, D- ('0' – control switched off, '1' – D+ (SPEED = '1') or D- (SPEED = '0') raising to high level)
5	SPEED	speed ('0' – 1,5 Мбит/с, '1' – 12 Мбит/с)
4	POL	line polarity ('0' – low speed line polarity (J=1, K=0), '1' – high speed line polarity (J=0, K=1))
3	DIRCR	line control enabling D+, D- ('0' – permitted, '1' – forbidden)
1-2	LINEST	line control D+, D- (if bit DIRCR = '1') 01 – line control D-, 10 – line control D+
0	EN_GLOB	operation enabling EP global bit ('0' – forbidden, '1' – permitted)

USBxLNST **Controller connection 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
Description	—																															TYPE

3-31	—	*reserved*
2	VBUS	bus voltage ('0' – +5 B USB supplied, '1' – +5 B USB not supplied)
0-1	LINE	connection state (00 – reset, 01 – 1,5 Mb/s, 10 – 12 Mb/s)

USBxINT **Controller 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	0	0
Description	—																										VBUS	NAK	SOF	RESET	RESUME	DONE

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

6-31	—	*reserved*
5	VBUS	external supply given. Cleansed by record '1'
4	NAK	NACK transmitted. Cleansed by record '1'
3	SOF	SOF received. Cleansed by record '1'
2	RESET	D+ и D- on low level. Cleansed by record '1'
1	RESUME	transaction renewal. Cleansed by record '1'
0	DONE	transaction completion. Cleansed by record '1'

USBxMSKINT **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
Description	—																										MSKBUS	MSKNAK	MSKSOF	MSKRESET	MSKRESUME	MSKDONE

For all register bits: '1' - event permitted, '0' - event forbidden

6-31	—	*reserved*
5	VBUS	external supply given
4	NAK	NACK transmitted
3	SOF	SOF received
2	RESET	D+ и D- on low level
1	RESUME	transaction renewal
0	DONE	transaction completion

USBxADDR **Device 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																									0						
Description	—																									ADDR						

7-31	—	*reserved*
0-6	ADDR	USB device address

USBxMSPFRAME **SOF packet counter most significant bits**

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	—																													MSPNUM		

3-31	—	*reserved*
0-2	MSPNUM	bits [10:8] of last SOF transaction received packets' amount

USBxLSPFRAME	SOF packet spinner least significant bits																															
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	—																								LSPNUM							

8-31	—	*reserved*
0-7	LSPNUM	bits [0:7] of last SOF transaction received packets' amount

USBxEPRXDATAn	EPn receiver buffer																															
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	—																								RXDATA							

8-31	—	*reserved*
0-7	RXDATA	received data buffer

USBxEPRXMSBn	EPn receiver buffer spinner most significant bit																															
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	—																								MSB_NUM							

8-31	—	*reserved*
0-7	MSB_NUM	receiver buffer data spinner most significant bit

USBxEPRXLSBn	EPn receiver buffer spinner least significant bit																															
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	—																								LSB_NUM							

8-31	—	*reserved*
0-7	LSB_NUM	receiver buffer data spinner least significant bit

USBxEPRXCLRn	EPn receiver buffer 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
Description	—																															CLR

1-31	—	*reserved*
0	CLR	receiver buffer cleansing (1 – cleanse)

USBxEPTXDATAn	EPn receiver buffer																															
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	—																								TXDATA							

8-31	—	*reserved*
0-7	TXDATA	transmitted data buffer

USBxEPTXCLRn	EPn transmitter buffer 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
Description	—																															CLR

1-31	—	*reserved*
0	CLR	transmitter buffer cleansing (1 – cleanse)

5.10 PWM controller (PWMx)

5.10.1 Brief characteristics

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

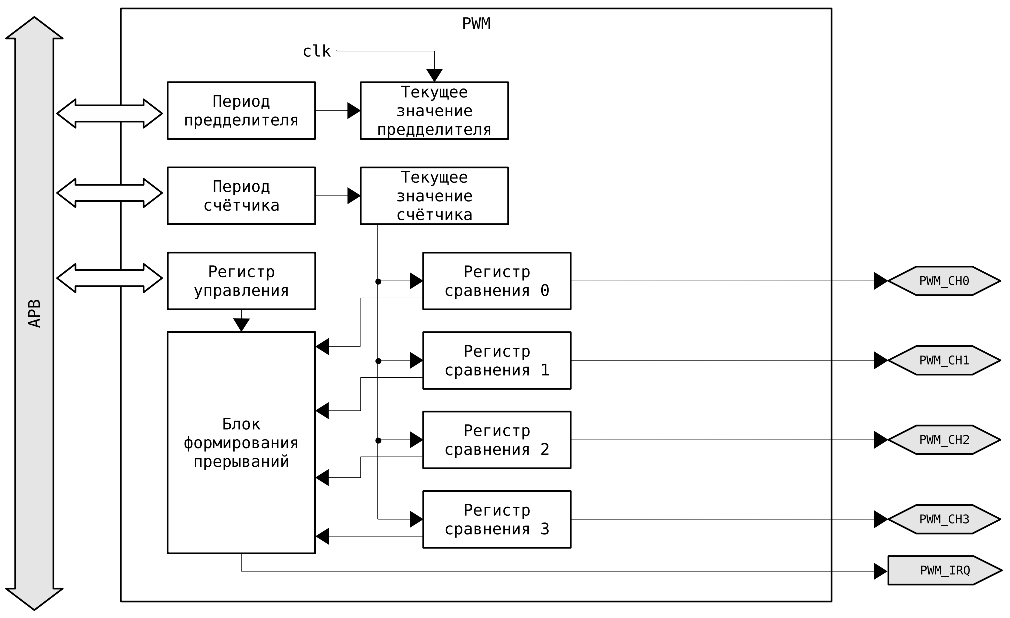

Figure 18: PWM block diagram

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).

5.10.2 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.

5.10.3 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.

5.10.4 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.

5.10.5 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.

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

5.10.6 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

6 Processor's pinmap assignation

6.1 Assignation of processor outputs in QFP208 package

Conventions	
S	supply line connection
I	input
O	output
NC	not connected
OSC	for oscillator/allocator connection
DVDD	+ output cascades (3.3B)
DVSS	output cascades' GND
VDD	+ cores (1.8B)
VSS	GND cores

№	Type	Port	Bit		Alternative function	Annotation
1	I/O	GPIOA	0	GPIOA[0]	eth0_col	
2	I/O		1	GPIOA[1]	eth0_tx_en	
3	I/O		2	GPIOA[2]	eth0_tx_er	
4	I/O		3	GPIOA[3]	eth0_txd0	
5	I/O		4	GPIOA[4]	eth0_txd1	
6	S			DVDD		
7	S			DVSS		
8	S			VSS		
9	S			VDD		
10	I/O	GPIOA	5	GPIOA[5]	eth0_txd2	
11	I/O		6	GPIOA[6]	eth0_txd3	
12	I/O		7	GPIOA[7]	eth0_tx_clk	
13	I/O		8	GPIOA[8]	eth0_crs	
14	I/O		9	GPIOA[9]	eth0_rx_dv	
15	S			DVDD		
16	S			DVSS		
17	S			VSS		
18	S			VDD		
19	I/O	GPIOA	10	GPIOA[10]	eth0_rx_er	
20	I/O		11	GPIOA[11]	eth0_rxd0	
21	I/O		12	GPIOA[12]	eth0_rxd1	
22	I/O		13	GPIOA[13]	eth0_rxd2	
23	I/O		14	GPIOA[14]	eth0_rxd3	
24	S			DVDD		
25	S			DVSS		
26	S			VSS		
27	S			VDD		
28	I/O	GPIOA	15	GPIOA[15]	eth0_rx_clk	
29	I/O		16	GPIOA[16]	eth0_mdio	
30	I/O		17	GPIOA[17]	eth0_mdc	
31	I/O		18	GPIOA[18]	usb0_v_det	
32	I/O		19	GPIOA[19]	-	
33	S			DVDD		
34	S			DVSS		
35	S			VSS		
36	S			VDD		
37	I/O		20	GPIOA[20]	-	

GPIOA

№	Type	Port	Bit		Alternative function	Annotation
38	I/O		21	GPIOA[21]	-	
39	I/O		22	GPIOA[22]	i2c0_scl	
40	I/O		23	GPIOA[23]	i2c0_sda	
41	I/O		24	GPIOA[24]	usb0_vp_in	
42	S			DVDD		
43	S			DVSS		
44	S			VSS		
45	S			VDD		
46	I/O	GPIOA	25	GPIOA[25]	usb0_vm_in	
47	I/O		26	GPIOA[26]	usb0_vp_out	
48	I/O		27	GPIOA[27]	usb0_vm_out	
49	I/O		28	GPIOA[28]	/usb0_oe	
50	I/O		29	GPIOA[29]	usb0_fs	
51	I/O		30	GPIOA[30]	usb0_dp_pullup	
52	I/O		31	GPIOA[31]	usb0_dm_pullup	
53	I/O	GPIOB	0	GPIOB[0]	spi0_sck_out	
54	I/O		1	GPIOB[1]	spi0_mosi	
55	I/O		2	GPIOB[2]	spi0_miso	
56	I/O		3	GPIOB[3]	spi0_sel_in	
57	I/O		4	GPIOB[4]	spi0_sck_in	
58	S			DVDD		
59	S			DVSS		
60	S			VSS		
61	S			VDD		
62	I/O	GPIOB	5	GPIOB[5]	spi0_ss0	
63	I/O		6	GPIOB[6]	spi0_ss1	
64	I/O		7	GPIOB[7]	spi0_ss2	
65	I/O		8	GPIOB[8]	uart0_txd	
66	I/O		9	GPIOB[9]	uart0_rxd	
67	S			DVDD		
68	S			DVSS		
69	S			VSS		
70	S			VDD		
71	I/O	GPIOB	10	GPIOB[10]	uart0_cts	
72	I/O		11	GPIOB[11]	uart0_rts	
73	I/O		12	GPIOB[12]	i2s_din	
74	I/O		13	GPIOB[13]	i2s_ws	
75	I/O		14	GPIOB[14]	i2s_sck	
76	S			DVDD		
77	S			DVSS		
78	S			VSS		
79	S			VDD		
80	I/O	GPIOB	15	GPIOB[15]	gptim3_extclk	
81	I/O		16	GPIOB[16]	spi1_sck	
82	I/O		17	GPIOB[17]	spi1_mosi	
83	I/O		18	GPIOB[18]	spi1_miso	
84	I/O		19	GPIOB[19]	spi1_sel_in	
85	S			DVDD		
86	S			DVSS		
87	S			VSS		
88	S			VDD		

№	Type	Port	Bit		Alternative function	Annotation
89	I/O	GPIOB	20	GPIOB[20]	spi1_sck_in	
90	I/O		21	GPIOB[21]	spi1_ss0	
91	I/O		22	GPIOB[22]	spi1_ss1	
92	I/O		23	GPIOB[23]	spi1_ss2	
93	I/O		24	GPIOB[24]	uart1_txd	
94	S			DVDD		
95	S			DVSS		
96	S			VSS		
97	S			VDD		
98	I/O	GPIOB	25	GPIOB[25]	uart1_rxd	
99	I/O		26	GPIOB[26]	uart1_cts	
100	I/O		27	GPIOB[27]	uart1_rts	
101	I/O		28	GPIOB[28]	-	
102	I/O		29	GPIOB[29]	-	
103	I/O		30	GPIOB[30]	-	
104	I/O		31	GPIOB[31]	gptim4_extclk	
105	O			extrom_clk		External ROM clock output
106	I			extrom_data		External ROM data input
107	O			/extrom_ce		External ROM operation enabling signal
108	O			mem_ready		MP memory signal ready
109	I			/nmi		External interrupt request signal (non-maskable)
110	S			DVDD		
111	S			DVSS		
112	S			VSS		
113	S			VDD		
114	I			/trst		JTAG (IEEE 1149.1)
115	I			tms		
116	O			tdo		
117	I			tdi		
118	I			tck		
119	S			DVDD		
120	S			DVSS		
121	S			VSS		
122	S			VDD		
123	NC	-	-	-	-	
124	I/O	GPIOC	0	GPIOC[0]	gptim0_extclk	
125	I/O		1	GPIOC[1]	gptim1_extclk	
126	I/O		2	GPIOC[2]	gptim2_extclk	
127	I/O		3	GPIOC[3]	-	
128	S			DVDD		
129	S			DVSS		
130	S			VSS		
131	S			VDD		
132	I/O	GPIOC	4	GPIOC[4]	pwm0	
133	I/O		5	GPIOC[5]	pwm1	
134	I/O		6	GPIOC[6]	pwm2	
135	I/O		7	GPIOC[7]	pwm3	
136	I/O		8	GPIOC[8]	uart2_txd	
137	S			DVDD		

№	Type	Port	Bit		Alternative function	Annotation
138	S			DVSS		
139	S			VSS		
140	S			VDD		
141	I/O	GPIOC	9	GPIOC[9]	uart2_rxd	
142	I/O		10	GPIOC[10]	uart2_cts	
143	I/O		11	GPIOC[11]	uart2_rts	
144	I/O		12	GPIOC[12]	-	
145	I/O		13	GPIOC[13]	-	
146	S			DVDD		
147	S			DVSS		
148	S			VSS		
149	S			VDD		
150	I/O	GPIOC	14	gpioc[14]	-	
151	I/O		15	GPIOC[15]	gptim5_extclk	
152	I/O		16	GPIOC[16]	-	
153	I/O		17	GPIOC[17]	-	
154	I/O		18	GPIOC[18]	-	
155	I/O		19	GPIOC[19]	-	
156	I/O		20	GPIOC[20]	-	
157	I/O		21	GPIOC[21]	-	
158	I/O		22	GPIOC[22]	-	
159	I/O		23	GPIOC[23]	-	
160	I/O	GPIOD	0	GPIOD[0]	spi2_sck	
161	I/O		1	gpiod[1]	spi2_mosi	
162	S			DVDD		
163	S			DVSS		
164	S			VSS		
165	S			VDD		
166	I/O	GPIOD	2	GPIOD[2]	spi2_miso	
167	I/O		3	GPIOD[3]	spi2_sel_in	
168	I/O		4	GPIOD[4]	spi2_sck_in	
169	I/O		5	GPIOD[5]	spi2_ss0	
170	I/O		6	GPIOD[6]	spi2_ss1	
171	S			DVDD		
172	S			DVSS		
173	S			VSS		
174	S			VDD		
175	I/O	GPIOD	7	GPIOD[7]	spi2_ss2	
176	I/O		8	GPIOD[8]	uart3_txd	
177	I/O		9	GPIOD[9]	uart3_rxd	
178	I/O		10	GPIOD[10]	uart3_cts	
179	I/O		11	GPIOD[11]	uart3_rts	
180	S			DVDD		
181	S			DVSS		
182	S			VSS		
183	S			VDD		
184	I/O	GPIOD	12	GPIOD[12]	i2c1_scl	
185	I/O		13	GPIOD[13]	i2c1_sda	
186	I/O		14	GPIOD[14]	-	
187	I/O		15	GPIOD[15]	gptim6_extclk	
188	I			clk_usb	usb0_clk	

№	Type	Port	Bit		Alternative function	Annotation
189	S			DVDD		
190	S			DVSS		
191	S			VSS		
192	S			VDD		
193	I			wake_up		Sleep mode quit injected signal
194	I			xtal_in		External allocator input
195	NC	-	-	-	-	
196	NC	-	-	-	-	
197	NC	-	-	-	-	
198	S			DVDD		
199	S			DVSS		
200	S			VSS		
201	S			VDD		
202	NC	-	-	-	-	
203	NC	-	-	-	-	
204	NC	-	-	-	-	
205	I		-	nreset		Reset signal (log. "0" – active)
206	NC	-	-	-	-	
207	NC	-	-	-	-	
208	NC	-	-	-	-	

6.2 Assignation of processor outputs in QFP240 package

Conventions	
S	supply line connection
I	input
O	output
NC	not connected
OSC	for oscillator/allocator connection
DVDD	+ output cascades (3.3B)
DVSS	output cascades' GND
VDD	+ cores (1.8B)
VSS	GND cores

№	Type	Port	bit		Alternative function	Annotation
1	NC	-	-	-	-	
2	NC	-	-	-	-	
3	NC	-	-	-	-	
4	NC	-	-	-	-	
5	I/O		0	GPIOA[0]	eth0_col	
6	I/O		1	GPIOA[1]	eth0_tx_en	
7	I/O	GPIOA	2	GPIOA[2]	eth0_tx_er	
8	I/O		3	GPIOA[3]	eth0_txd0	
9	I/O		4	GPIOA[4]	eth0_txd1	
10	S			DVDD		
11	S			DVSS		
12	S			VSS		
13	S			VDD		
14	I/O		5	GPIOA[5]	eth0_txd2	
15	I/O		6	GPIOA[6]	eth0_txd3	
16	I/O	GPIOA	7	GPIOA[7]	eth0_tx_clk	
17	I/O		8	GPIOA[8]	eth0_crs	
18	I/O		9	GPIOA[9]	eth0_rx_dv	
19	S			DVDD		
20	S			DVSS		
21	S			VSS		
22	S			VDD		
23	I/O		10	GPIOA[10]	eth0_rx_er	
24	I/O		11	GPIOA[11]	eth0_rxd0	
25	I/O	GPIOA	12	GPIOA[12]	eth0_rxd1	
26	I/O		13	GPIOA[13]	eth0_rxd2	
27	I/O		14	GPIOA[14]	eth0_rxd3	
28	S			DVDD		
29	S			DVSS		
30	S			VSS		
31	S			VDD		
32	I/O		15	GPIOA[15]	eth0_rx_clk	
33	I/O		16	GPIOA[16]	eth0_mdio	
34	I/O	GPIOA	17	GPIOA[17]	eth0_mdc	
35	I/O		18	GPIOA[18]	usb0_v_det	
36	I/O		19	GPIOA[19]	-	
37	S			DVDD		
38	S			DVSS		
39	S			VSS		
40	S			VDD		
41	I/O		20	GPIOA[20]	-	

GPIOA

№	Type	Port	bit		Alternative function	Annotation
42	I/O		21	GPIOA[21]	-	
43	I/O		22	GPIOA[22]	i2c0_scl	
44	I/O		23	GPIOA[23]	i2c0_sda	
45	I/O		24	GPIOA[24]	usb0_vp_in	
46	S			DVDD		
47	S			DVSS		
48	S			VSS		
49	S			VDD		
50	I/O		25	GPIOA[25]	usb0_vm_in	
51	I/O		26	GPIOA[26]	usb0_vp_out	
52	I/O		27	GPIOA[27]	usb0_vm_out	
53	I/O	GPIOA	28	GPIOA[28]	/usb0_oe	
54	I/O		29	GPIOA[29]	usb0_fs	
55	I/O		30	GPIOA[30]	usb0_dp_pullup	
56	I/O		31	GPIOA[31]	usb0_dm_pullup	
57	NC	-	-	-	-	
58	NC	-	-	-	-	
59	NC	-	-	-	-	
60	NC	-	-	-	-	
61	NC	-	-	-	-	
62	NC	-	-	-	-	
63	NC	-	-	-	-	
64	NC	-	-	-	-	
65	I/O		0	GPIOB[0]	spi0_sck_out	
66	I/O		1	GPIOB[1]	spi0_mosi	
67	I/O	GPIOB	2	GPIOB[2]	spi0_miso	
68	I/O		3	GPIOB[3]	spi0_sel_in	
69	I/O		4	GPIOB[4]	spi0_sck_in	
70	S			DVDD		
71	S			DVSS		
72	S			VSS		
73	S			VDD		
74	I/O		5	GPIOB[5]	spi0_ss0	
75	I/O		6	GPIOB[6]	spi0_ss1	
76	I/O	GPIOB	7	GPIOB[7]	spi0_ss2	
77	I/O		8	GPIOB[8]	uart0_txd	
78	I/O		9	GPIOB[9]	uart0_rxd	
79	S			DVDD		
80	S			DVSS		
81	S			VSS		
82	S			VDD		
83	I/O		10	GPIOB[10]	uart0_cts	
84	I/O		11	GPIOB[11]	uart0_rts	
85	I/O	GPIOB	12	GPIOB[12]	i2s_din	
86	I/O		13	GPIOB[13]	i2s_ws	
87	I/O		14	GPIOB[14]	i2s_sck	
88	S			DVDD		
89	S			DVSS		
90	S			VSS		
91	S			VDD		
92	I/O		15	GPIOB[15]	gptim3_extclk	

GPIOB

№	Type	Port	bit		Alternative function	Annotation
93	I/O		16	GPIOB[16]	spi1_sck	
94	I/O		17	GPIOB[17]	spi1_mosi	
95	I/O		18	GPIOB[18]	spi1_miso	
96	I/O		19	GPIOB[19]	spi1_sel_in	
97	S			DVDD		
98	S			DVSS		
99	S			VSS		
100	S			VDD		
101	I/O		20	GPIOB[20]	spi1_sck_in	
102	I/O		21	GPIOB[21]	spi1_ss0	
103	I/O	GPIOB	22	GPIOB[22]	spi1_ss1	
104	I/O		23	GPIOB[23]	spi1_ss2	
105	I/O		24	GPIOB[24]	uart1_txd	
106	S			DVDD		
107	S			DVSS		
108	S			VSS		
109	S			VDD		
110	I/O		25	GPIOB[25]	uart1_rxd	
111	I/O		26	GPIOB[26]	uart1_cts	
112	I/O		27	GPIOB[27]	uart1_rts	
113	I/O	GPIOB	28	GPIOB[28]	-	
114	I/O		29	GPIOB[29]	-	
115	I/O		30	GPIOB[30]	-	
116	I/O		31	GPIOB[31]	gptim4_extclk	
117	NC	-	-	-	-	
118	NC	-	-	-	-	
119	NC	-	-	-	-	
120	NC	-	-	-	-	
121	NC	-	-	-	-	
122	NC	-	-	-	-	
123	NC	-	-	-	-	
124	NC	-	-	-	-	
125	O			extrom_clk		External ROM clock output
126	I			extrom_data		External ROM data input
127	O			/extrom_ce		External ROM operation enabling signal
128	O			mem_ready		MP memory signal ready
129	I			/nmi		External interrupt request signal (non-maskable)
130	S			DVDD		
131	S			DVSS		
132	S			VSS		
133	S			VDD		
134	I			/trst		JTAG (IEEE 1149.1)
135	I			tms		
136	O			tdo		
137	I			tdi		
138	I			tck		
139	S			DVDD		
140	S			DVSS		
141	S			VSS		

№	Type	Port	bit		Alternative function	Annotation
142	S			VDD		
143	NC	-	-	-	-	
144	I/O		0	GPIOC[0]	gptim0_extclk	
145	I/O	GPIOC	1	GPIOC[1]	gptim1_extclk	
146	I/O		2	GPIOC[2]	gptim2_extclk	
147	I/O		3	GPIOC[3]	-	
148	S			DVDD		
149	S			DVSS		
150	S			VSS		
151	S			VDD		
152	I/O		4	GPIOC[4]	pwm0	
153	I/O		5	GPIOC[5]	pwm1	
154	I/O	GPIOC	6	GPIOC[6]	pwm2	
155	I/O		7	GPIOC[7]	pwm3	
156	I/O		8	GPIOC[8]	uart2_txd	
157	S			DVDD		
158	S			DVSS		
159	S			VSS		
160	S			VDD		
161	I/O		9	GPIOC[9]	uart2_rxd	
162	I/O		10	GPIOC[10]	uart2_cts	
163	I/O	GPIOC	11	GPIOC[11]	uart2_rts	
164	I/O		12	GPIOC[12]	-	
165	I/O		13	GPIOC[13]	-	
166	S			DVDD		
167	S			DVSS		
168	S			VSS		
169	S			VDD		
170	I/O		14	gpioc[14]	-	
171	I/O		15	GPIOC[15]	gptim5_extclk	
172	I/O		16	GPIOC[16]	-	
173	I/O		17	GPIOC[17]	-	
174	I/O	GPIOC	18	GPIOC[18]	-	
175	I/O		19	GPIOC[19]	-	
176	I/O		20	GPIOC[20]	-	
177	NC	-	-	-	-	
178	NC	-	-	-	-	
179	NC	-	-	-	-	
180	NC	-	-	-	-	
181	NC	-	-	-	-	
182	NC	-	-	-	-	
183	NC	-	-	-	-	
184	NC	-	-	-	-	
185	I/O		21	GPIOC[21]	-	
186	I/O		22	GPIOC[22]	-	
187	I/O		23	GPIOC[23]	-	
188	I/O	GPIOD	0	GPIOD[0]	spi2_sck	
189	I/O		1	gpiod[1]	spi2_mosi	
190	S			DVDD		
191	S			DVSS		
192	S			VSS		

№	Type	Port	bit		Alternative function	Annotation
193	S			VDD		
194	I/O		2	GPIOD[2]	spi2_miso	
195	I/O		3	GPIOD[3]	spi2_sel_in	
196	I/O	GPIOD	4	GPIOD[4]	spi2_sck_in	
197	I/O		5	GPIOD[5]	spi2_ss0	
198	I/O		6	GPIOD[6]	spi2_ss1	
199	S			DVDD		
200	S			DVSS		
201	S			VSS		
202	S			VDD		
203	I/O		7	GPIOD[7]	spi2_ss2	
204	I/O		8	GPIOD[8]	uart3_txd	
205	I/O	GPIOD	9	GPIOD[9]	uart3_rxd	
206	I/O		10	GPIOD[10]	uart3_cts	
207	I/O		11	GPIOD[11]	uart3_rts	
208	S			DVDD		
109	S			DVSS		
110	S			VSS		
111	S			VDD		
112	I/O		12	GPIOD[12]	i2c1_scl	
113	I/O	GPIOD	13	GPIOD[13]	i2c1_sda	
114	I/O		14	GPIOD[14]	-	
115	I/O		15	GPIOD[15]	gptim6_extclk	
116	I			clk_usb	usb0_clk	
117	S			DVDD		
118	S			DVSS		
119	S			VSS		
120	S			VDD		
121	I			wake_up		Sleep mode quit injected signal
122	I			xtal_in		External allocator input
123	NC	-	-	-	-	
124	NC	-	-	-	-	
125	NC	-	-	-	-	
126	S			DVDD		
127	S			DVSS		
128	S			VSS		
129	S			VDD		
130	NC	-	-	-	-	
131	NC	-	-	-	-	
132	NC	-	-	-	-	
133	I		-	nreset		Reset signal (log. "0" – active)
134	NC	-	-	-	-	
135	NC	-	-	-	-	
136	NC	-	-	-	-	
137	NC	-	-	-	-	
138	NC	-	-	-	-	
139	NC	-	-	-	-	
140	NC	-	-	-	-	

6.3 Assignation of processor outputs in LQFP128 package

Conventions	
S	supply line connection
I	input
O	output
NC	not connected
OSC	for oscillator/allocator connection
DVDD	+ output cascades (3.3B)
DVSS	output cascades' GND
VDD	+ cores (1.8B)
VSS	GND cores

6.4 Assignation of processor outputs in LQFP144 package

Conventions	
S	supply line connection
I	input
O	output
NC	not connected
OSC	for oscillator/allocator connection
DVDD	+ output cascades (3.3B)
DVSS	output cascades' GND
VDD	+ cores (1.8B)
VSS	GND cores

6.5 Processor output diagram in QFP208 package

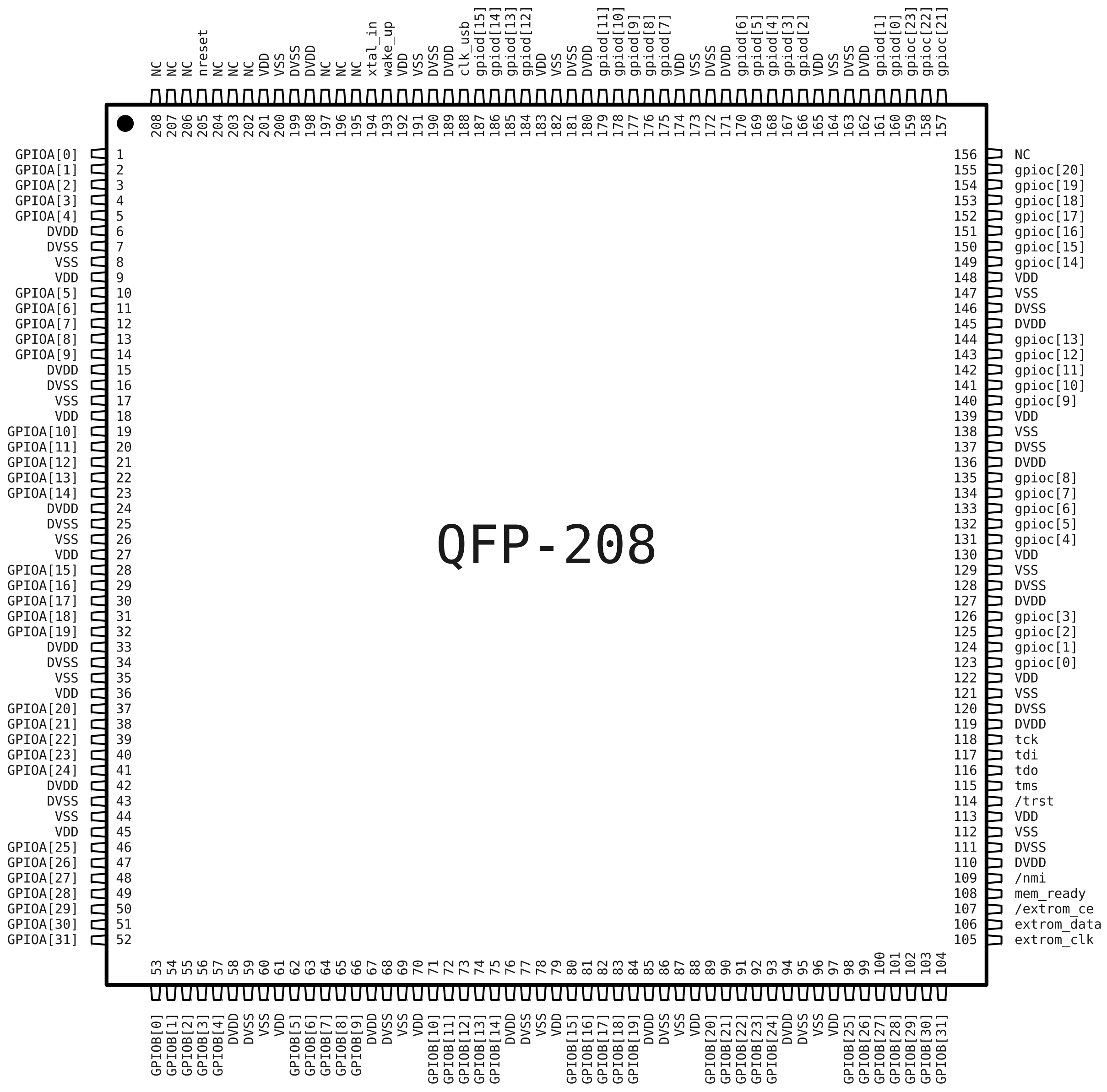

Figure 19: Processor output diagram

6.6 Processor output diagram in QFP208 package

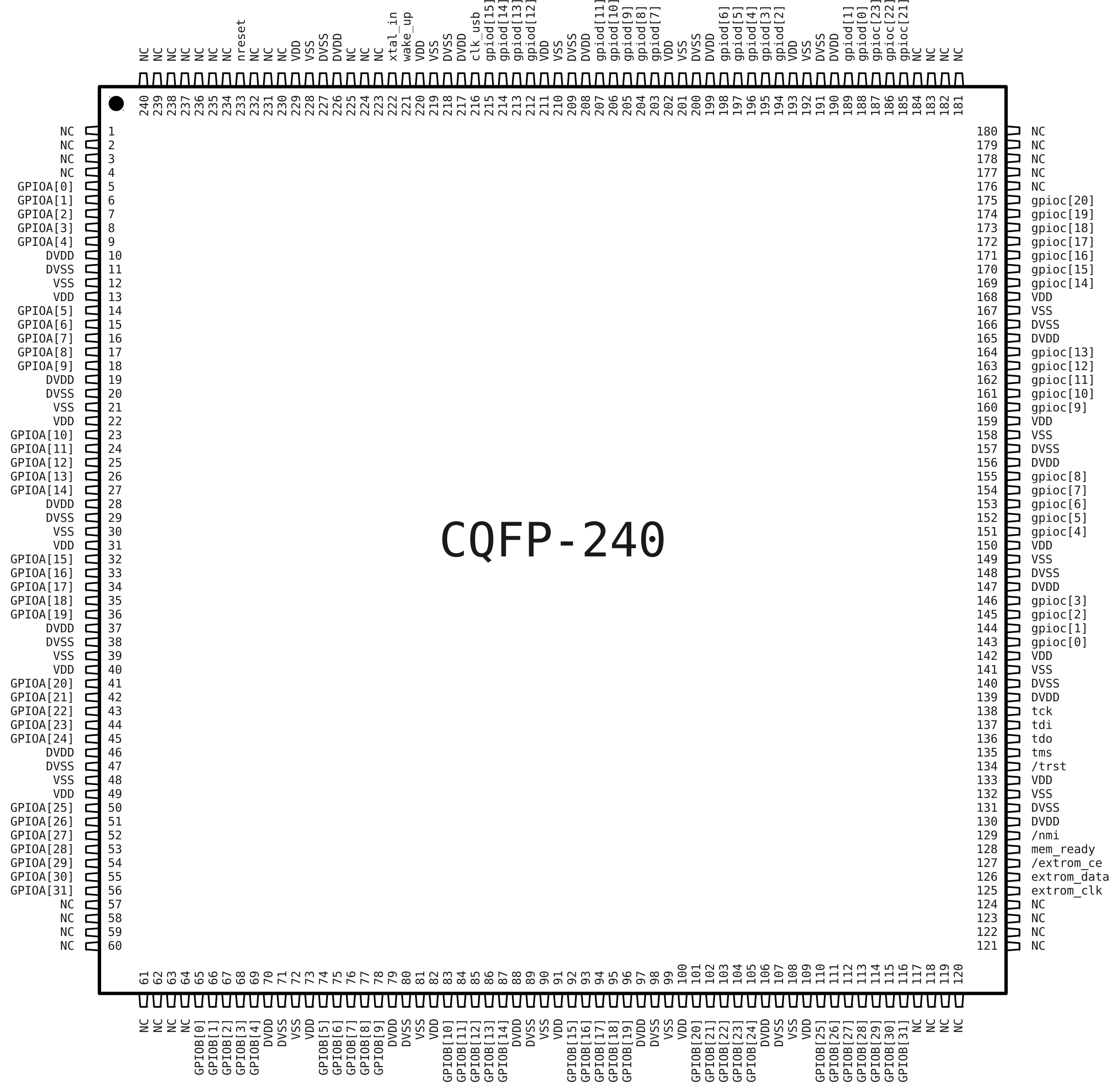

Figure 20: Processor output diagram

6.7 Processor output diagram in LQFP128 package

6.8 Processor output diagram in LQFP144 package

7 Electric characteristics

7.1 Electrical characteristics of input-output ports

Table 15: 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	$I_{OL} = -12$mA	0		$0,4$
V_{OH}	High-level output voltage, v	$I_{OH} = 12$mA	2.4		3.6
	High-level raising register, kOhm		$68,2$		$118,1$
	Low-level raising register , kOhm		$30,2$		$80,6$

A Annotation

1 — In MCp0411100101 query analyzer of interruptions (№ № 7-31) from PU can only be executed through a survey of their status registers. In other words, in the program cycle or by means of interrupt from the system timer, status of the required PU is to be read, and in the process of analyzing it, the necessity of interrupt query handler run is determined. Interrupts of the core (№ № 0-6) hardwarily causes the transition to interrupt query handler;