



Programming Reference Guide
netX 50

Hilscher Gesellschaft für Systemautomation mbH
www.hilscher.com

DOC081107PRG11EN | Revision 1.1 | English | 2015-04 | Released | Public

Table of Contents

1	NAMING CONVENTIONS	4
2	SYSTEM FUNCTIONS	5
2.1	ACCESS_KEY – Access Protection	7
2.2	NETX_REV – netX Revision	8
2.3	IO Configuration	9
2.4	RESET – Reset Controller	16
2.5	Clock Control – Clock Generation and Control	18
2.6	WDG - Watchdog	22
2.7	SYS_STAT – System Status	24
3	MEMORY CONTROLLER	26
3.1	MEM_SRAM – Memory Controller for SRAM and FLASH	27
3.2	MEM_SDRAM – Memory Controller for SDRAM	28
3.3	MEM_PRIO – Memory Priority Controller	33
4	EXTENSION BUS	35
5	DUAL-PORT MEMORY	37
5.1	DPM_HOST – Dual-Port Memory Host Side	37
5.2	DPM_ARM – Dual-Port Memory ARM Side	44
6	PERIPHERAL FUNCTIONS	63
6.1	GPIOs – General Purpose IOs and Timers	63
6.2	IO-Link	76
6.3	PIO – Programmable Input Output	80
6.4	UART – Universal Asynchronous Receiver Transmitter	81
6.5	SPI – Serial Peripheral Interface	95
6.6	I2C – Serial I2C-Interface	109
6.7	CCDC – CCD Controller	122
6.8	SYS_TIME – System time with IEEE 1588 functionality	129
6.9	MMIO – Multiplex Matrix IOs	132
6.10	USB – Serial USB-Interface	132
6.11	VIC – Vectored Interrupt Controller	158
7	COMMUNICATION FUNCTIONS	166
7.1	PHY – Controller for internal PHYs	167
7.2	PTR_FIFO – Pointer FIFO	171
7.3	Buffer Management Unit (BMU)	175
7.4	CRC – Configurable CRC-Generator	180
7.5	ARM_to_XPEC_IRQ	182
8	DMA CONTROLLER	183
8.1	Functional Overview	183
8.2	Functional Description	184
8.3	Software Interface	186
8.4	Register Definition	192
9	ARM SYSTEM CONTROL AND CONFIGURATION REGISTERS	207
9.1	CP15 Registers Summary	207

9.2	CP15 Registers Description	207
10	APPENDIX A: REGISTER TABLE	213
11	REVISION HISTORY.....	224

1 Naming Conventions

Generally for the various functions and parts of a microcontroller a number of acronyms come into play. For ease of use, in this reference guide a consistent naming scheme is introduced to all the netX register names which will be used throughout. A full list of register names can be found in appendix A.

The naming of the registers is carried out as follows: The first component determines the register group, e.g. GPIO or DPM, etc., while the subsequent parts, separated by underscores "_", specify the function of the particular register more detailed. Note that the word "register" will never occur in the name as it does not yield any additional information.

Sometimes the notation [m-n] will be used to indicate a set of registers ranging from m to n, e.g. IRQ_XP[0-3] stands for the (sequence of) registers IRQ_XP0, IRQ_XP1, IRQ_XP2 and IRQ_XP3.

2 System Functions

This chapter lists major system functions, like IO configuration, clock- and reset control, watchdog and status, access protection, etc.

The following table is a summary of the registers, related to these functions.

ARM Address	Register Name	Short Description
0x1c000070	ACCESS_KEY	Access Protection Register
0x1c000034	NETX_REV	netX Revision Register
0x1c000004	IO_CFG	IO Configuration Register
0x1c001300	MMIO0_CFG	Multiplex matrix Configuration Register for MMIO0
0x1c001304	MMIO1_CFG	Multiplex matrix Configuration Register for MMIO1
0x1c001308	MMIO2_CFG	Multiplex matrix Configuration Register for MMIO2
0x1c00130c	MMIO3_CFG	Multiplex matrix Configuration Register for MMIO3
0x1c001310	MMIO4_CFG	Multiplex matrix Configuration Register for MMIO4
0x1c001314	MMIO5_CFG	Multiplex matrix Configuration Register for MMIO5
0x1c001318	MMIO6_CFG	Multiplex matrix Configuration Register for MMIO6
0x1c00131c	MMIO7_CFG	Multiplex matrix Configuration Register for MMIO7
0x1c001320	MMIO8_CFG	Multiplex matrix Configuration Register for MMIO8
0x1c001324	MMIO9_CFG	Multiplex matrix Configuration Register for MMIO9
0x1c001328	MMIO10_CFG	Multiplex matrix Configuration Register for MMIO10
0x1c00132c	MMIO11_CFG	Multiplex matrix Configuration Register for MMIO11
0x1c001330	MMIO12_CFG	Multiplex matrix Configuration Register for MMIO12
0x1c001334	MMIO13_CFG	Multiplex matrix Configuration Register for MMIO13
0x1c001338	MMIO14_CFG	Multiplex matrix Configuration Register for MMIO14
0x1c00133c	MMIO15_CFG	Multiplex matrix Configuration Register for MMIO15
0x1c001340	MMIO16_CFG	Multiplex matrix Configuration Register for MMIO16
0x1c001344	MMIO17_CFG	Multiplex matrix Configuration Register for MMIO17
0x1c001348	MMIO18_CFG	Multiplex matrix Configuration Register for MMIO18
0x1c00134c	MMIO19_CFG	Multiplex matrix Configuration Register for MMIO19
0x1c001350	MMIO20_CFG	Multiplex matrix Configuration Register for MMIO20
0x1c001354	MMIO21_CFG	Multiplex matrix Configuration Register for MMIO21
0x1c001358	MMIO22_CFG	Multiplex matrix Configuration Register for MMIO22
0x1c00135c	MMIO23_CFG	Multiplex matrix Configuration Register for MMIO23
0x1c001360	MMIO24_CFG	Multiplex matrix Configuration Register for MMIO24
0x1c001364	MMIO25_CFG	Multiplex matrix Configuration Register for MMIO25
0x1c001368	MMIO26_CFG	Multiplex matrix Configuration Register for MMIO26
0x1c00136c	MMIO27_CFG	Multiplex matrix Configuration Register for MMIO27
0x1c001370	MMIO28_CFG	Multiplex matrix Configuration Register for MMIO28
0x1c001374	MMIO29_CFG	Multiplex matrix Configuration Register for MMIO29
0x1c001378	MMIO30_CFG	Multiplex matrix Configuration Register for MMIO30
0x1c00137c	MMIO31_CFG	Multiplex matrix Configuration Register for MMIO31
0x1c001380	MMIO32_CFG	Multiplex matrix Configuration Register for MMIO32
0x1c001384	MMIO33_CFG	Multiplex matrix Configuration Register for MMIO33
0x1c001388	MMIO34_CFG	Multiplex matrix Configuration Register for MMIO34

ARM Address	Register Name	Short Description
0x1c00138c	MMIO35_CFG	Multiplex matrix Configuration Register for MMIO35
0x1c001390	MMIO36_CFG	Multiplex matrix Configuration Register for MMIO36
0x1c001394	MMIO37_CFG	Multiplex matrix Configuration Register for MMIO37
0x1c001398	MMIO38_CFG	Multiplex matrix Configuration Register for MMIO38
0x1c00139c	MMIO39_CFG	Multiplex matrix Configuration Register for MMIO39
0x1c00000c	RESET_CTRL	Reset Control Register
0x1c000024	CLK_EN	Global Clock Enable Register
0x1c000014	ARM_CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000018	USB_CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c00001c	FB0CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000020	FB1CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000200	WDG_TRIG	Watchdog Trigger Register
0x1c000204	WDG_CNTR	Watchdog Counter
0x1c000208	WDG_IRQ_TIMEOUT	Watchdog Interrupt Timeout
0x1c00020c	WDG_RESET_TIMEOUT	Watchdog Reset Timeout
0x1c0034d8	SYS_STAT	System Status

2.1 ACCESS_KEY – Access Protection

Writing to any register in the CTRL or MMIO_CTRL address area is protected by the netX Access Key to avoid undesired changes e.g. by crashed software. The following table shows these protected registers:

ARM Address	Register Name	Short Description
0x1c000004	IO_CFG	IO Config Register
0x1c00000c	RESET_CTRL	Reset Control Register
0x1c000010	PHY_CTRL	Phy Control Register
0x1c000014	ARM_CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000018	USB_CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c00001c	FB0CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000020	FB1CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x1c000024	CLK_EN	Global Clock Enable Register
0x1c00002c	MISC_CTRL	Miscellaneous ASIC Control Register
0x1c000034	NETX_REV	netX Revision Register (written once during bootup)
0x1c001300	MMIO0_CFG	Multiplex matrix Configuration Register for MMIO0
...
0x1c00139c	MMIO39_CFG	Multiplex matrix Configuration Register for MMIO39

For writing to one of these registers, the software must perform the following sequence:

- 1.: read out Locking Access Key
- 2.: write back Locking Access Key
- 3.: write desired value to the register

The access key will become invalid after each access to any register in the CTRL or MMIO_CTRL address area and has to be read and written again for subsequent accesses.

IO_CFG_ACCESS_KEY – ASIC Control Locking Access Key Register

0x1c000070

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
Reserved																access_key															

Bits	Name	Description	R/W	Default
31:16	-	Reserved	R	0x0
15:0	access_key	Locking Access Key for next write access.	R/W	0x0

2.2 NETX_REV – netX Revision

NETX_REV_BL – netX Boot Loader Version Register

0x1c000034

This register contains information about the netX boot loader version.

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
reserved																								BL_VERSION							

Bits	Name	Description	R/W	Default
31:8	-	reserved netX Boot loader Version:	R	0x0
7:0	BL_VERSION	0x41 (A) = 'ABoot' 0x42 (B) = 'HBoot'	R	0x42

NETX_REV_HW – netX Hardware Revision Register

0x08200008

This register contains information about netX hardware and ROM code revision.

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
reserved								STEP								CHIP_TYPE								ROM_REV							

Bits	Name	Description	R/W	Default
31:28	-	reserved	R	0x0
27:20	STEP	Silicon Step 0x00 = Step A 0x01 = Step B	R	0x00
19:12	CHIP_TYPE	Chip Type 0x01 = netX500 0x02 = netX50 0x03 = netX100 0x04 = netX5	R	0x02
11:0	ROM_REV	ROM Code Revision	R	0x001

2.3 IO Configuration

To keep the chip package small, the netX50 uses shared IO pads, which are configurable for multiplexing different functions to the same pad. Two general methods of pad multiplexing are distinguished: synchronous and asynchronous multiplexing.

Synchronous multiplexing is configured by one MMIO*_CFG register for each of the 40 MMIO pads. The Multiplexing Matrix unit inside netX50 allows to select 1 of 148 internal functions for each MMIO pad. Synchronous multiplexing does only work with signals derived from the same clock.

Some signals that do not run on 100MHz system clock are multiplexed asynchronously. Asynchronous multiplexing offers less pad sharing combinations and is configured by only one register IO_CFG. The following table shows the asynchronously shared pads:

Pad	Standard function Signal	Option1		Option2	
		Select signal	Signal	Select signal	Signal
MMIO1	MMIO1	SEL_XM0_TX	XM0_TX		
MMIO2	MMIO2	SEL_XM0_ECLK	XM0_ECLK	SEL_FBCLK0_A	FB0CLK
MMIO4	MMIO4	SEL_XM1_TX	XM1_TX		
MMIO5	MMIO5	SEL_XM1_ECLK	XM1_ECLK	SEL_FBCLK1_A	FB1CLK
MMIO6	MMIO6			SEL_FBCLK0_B	FB0CLK
MMIO7	MMIO7			SEL_FBCLK1_B	FB1CLK
MMIO17	MMIO17			SEL_ETM	ETM_TRACECLK
MMIO18	MMIO18			SEL_ETM	ETM_TRACESYNC
MMIO19	MMIO19			SEL_ETM	ETM_DBGRRQ
MMIO20	MMIO20			SEL_ETM	ETM_DBGACK
MMIO21	MMIO21			SEL_ETM	ETM_PIPESTAT0
MMIO22	MMIO22			SEL_ETM	ETM_PIPESTAT1
MMIO23	MMIO23			SEL_ETM	ETM_PIPESTAT2
MMIO24	MMIO24			SEL_ETM	ETM_TRACEPKT0
MMIO25	MMIO25			SEL_ETM	ETM_TRACEPKT1
MMIO26	MMIO26			SEL_ETM	ETM_TRACEPKT2
MMIO27	MMIO27			SEL_ETM	ETM_TRACEPKT3
MMIO28	MMIO28			SEL_ETM	ETM_TRACEPKT4
MMIO29	MMIO29			SEL_ETM	ETM_TRACEPKT5
MMIO30	MMIO30			SEL_ETM	ETM_TRACEPKT6
MMIO31	MMIO31			SEL_ETM	ETM_TRACEPKT7
MMIO32	MMIO32	SEL_F00	FO0_FN_EN	SEL_ETM	ETM_TRACEPKT8
MMIO33	MMIO33	SEL_F00	FO0_RD	SEL_ETM	ETM_TRACEPKT9
MMIO34	MMIO34	SEL_F00	FO0_SD	SEL_ETM	ETM_TRACEPKT10
MMIO35	MMIO35	SEL_F00	FO0_TD	SEL_ETM	ETM_TRACEPKT11
MMIO36	MMIO36	SEL_F01	FO1_FN_EN	SEL_ETM	ETM_TRACEPKT12
MMIO37	MMIO37	SEL_F01	FO1_RD	SEL_ETM	ETM_TRACEPKT13
MMIO38	MMIO38	SEL_F01	FO1_SD	SEL_ETM	ETM_TRACEPKT14
MMIO39	MMIO39	SEL_F01	FO1_TD	SEL_ETM	ETM_TRACEPKT15

The select signals are set in register IO_CFG. Their priority in the table above rises from left to right, i.e. "Option 2" has higher priority than "Option 1" which again has higher priority than the standard function. If neither Option1 or Option 2 is activated, the standard function applies automatically.

IO_CFG – IO Config Register**0x1c000004**

The following register is used for selecting particular functions of the shared netX pins. By setting the appropriate bits of IO_CFG register, the desired functions can be activated.

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register

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				
IF_SELECT_N	SEL_EXT_PHY1	SEL_EXT_PHY0	Reserved					SEL_ETM	reserved													SEL_I2C_MMIO	SEL_XM1_ECLK	SEL_XM0_ECLK	SEL_FO1		SEL_FO0		SEL_FB1_CLK_B	SEL_FB1_CLK_A	SEL_FB0_CLK_B	SEL_FB0_CLK_A	SEL_XM1_TX		SEL_XM0_TX

Bits	Name	Description	R/W	Default
31	IF_SELECT_N	1: Host interface modes disabled 0: DPM mode / Extension Bus mode / PIO-Mode	R/W	0x0
30	SEL_EXT_PHY1	external/internal PHY1 select: 1: use external PHY instead of internal PHY1 (s. mmio table) 0: use internal PHY	R/W	0x0
29	SEL_EXT_PHY0	external/internal PHY0 select: 1: use external PHY instead of internal PHY0 (s. mmio table) 0: use internal PHY	R/W	0x0
28:24	-	reserved	R	0x0
23	SEL_ETM	select pins for ETM9 (s. pinning table)	R/W	0x0
22:11	-	reserved	R	0x0
10	SEL_I2C_MMIO	select pads for I2C 1: use I2C via MMIO pads (s. mmio table) 0: use dedicated I2C_SDA/I2C_SCL-pads	R/W	0x0
9	SEL_XM1_ECLK	select pad for xMAC1 eclk (s. pinning table)	R/W	0x0
8	SEL_XM0_ECLK	select pad for xMAC0 eclk (s. pinning table)	R/W	0x0
7	SEL_FO1	select Fiber Optics of PHY1: 1: use Fiber Optics of PHY1 0: use standard interface of PHY1	R/W	0x0
6	SEL_FO0	select Fiber Optics of PHY0: 1: use Fiber Optics of PHY0 0: use standard interface of PHY0	R/W	0x0
5	SEL_FB1_CLK_B	select pad for fieldbus-clk1 at position b (s. pinning table)	R/W	0x0
4	SEL_FB1_CLK_A	select pad for fieldbus-clk1 at position a (s. pinning table)	R/W	0x0
3	SEL_FB0_CLK_B	select pad for fieldbus-clk0 at position b (s. pinning table)	R/W	0x0
2	SEL_FB0_CLK_A	select pad for fieldbus-clk0 at position a (s. pinning table)	R/W	0x0
1	SEL_XM1_TX	select pad for xMAC1 tx-bitstream direct output (s. pinning table)	R/W	0x0
0	SEL_XM0_TX	select pad for xMAC0 tx-bitstream direct output (s. pinning table)	R/W	0x0

If no select signal for asynchronous pad multiplexing is set in register IO_CFG, the functionality of the pad will be defined by the appropriate MMIO_CFG register:

MMIO0_CFG – IO-Multiplex matrix Configuration Register For Signal MMIO0**0x1c001300**

MMIO1_CFG – IO-Multiplex matrix Configuration Register For Signal MMIO1**0x1c001304**

...

MMIO39_CFG – IO-Multiplex matrix Configuration Register For Signal MMIO39**0x1c00139c**

These registers are protected by the netX access key mechanism; changing a register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register

Core-inputs not mapped to any MMIO will be assigned to 0. If one core-connection is mapped to more than one MMIO, the core-input-state will be these ORed MMIO-states. For signal selection coding (MMIO*_SEL) look at the table below.

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		
reserved																						MMIO_IN_INV	MMIO_OUT_INV	MMIO_SEL									

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0
9	MMIO[0-39]_IN_INV	Invert input: 1: invert input signal 0: keep original signal polarity	R/W	0
8	MMIO[0-39]_OUT_INV	Invert output: 1: invert output signal 0: keep original signal polarity	R/W	0
7:0	MMIO[0-39]_SEL	MMIO[0-39] signal selection	R/W	s. table below

The coding of MMIO*_SEL is as follows:

Coding	netX internal Function (core connection)	Signal Type	Functional Group	Default of
0x00	XM0_IO0	bi-directional	Fieldbus0	
0x01	XM0_IO1	bi-directional	Fieldbus0	
0x02	XM0_IO2	bi-directional	Fieldbus0	
0x03	XM0_IO3	bi-directional	Fieldbus0	
0x04	XM0_IO4	bi-directional	Fieldbus0	
0x05	XM0_IO5	bi-directional	Fieldbus0	
0x06	XM0_RX	input	Fieldbus0	
0x07	XM0_TX_OE	push/pull output	Fieldbus0	
0x08	XM0_TX_OUT	tristateable output	Fieldbus0	
0x09	XM1_IO0	bi-directional	Fieldbus1	
0x0a	XM1_IO1	bi-directional	Fieldbus1	
0x0b	XM1_IO2	bi-directional	Fieldbus1	
0x0c	XM1_IO3	bi-directional	Fieldbus1	

Coding	netX internal Function (core connection)	Signal Type	Functional Group	Default of
0x0d	XM1_IO4	bi-directional	Fieldbus1	
0x0e	XM1_IO5	bi-directional	Fieldbus1	
0x0f	XM1_RX	input	Fieldbus1	
0x10	XM1_TX_OE	push/pull output	Fieldbus1	
0x11	XM1_TX_OUT	tristateable output	Fieldbus1	
0x12	GPIO0	bi-directional	GPIO	MMIO0
0x13	GPIO1	bi-directional	GPIO	MMIO1
0x14	GPIO2	bi-directional	GPIO	MMIO2
0x15	GPIO3	bi-directional	GPIO	MMIO3
0x16	GPIO4	bi-directional	GPIO	MMIO4
0x17	GPIO5	bi-directional	GPIO	MMIO5
0x18	GPIO6	bi-directional	GPIO	MMIO6
0x19	GPIO7	bi-directional	GPIO	MMIO7
0x1a	GPIO8	bi-directional	GPIO	MMIO8
0x1b	GPIO9	bi-directional	GPIO	MMIO9
0x1c	GPIO10	bi-directional	GPIO	MMIO10
0x1d	GPIO11	bi-directional	GPIO	MMIO11
0x1e	GPIO12	bi-directional	GPIO	MMIO12
0x1f	GPIO13	bi-directional	GPIO	MMIO13
0x20	GPIO14	bi-directional	GPIO	MMIO14
0x21	GPIO15	bi-directional	GPIO	MMIO15
0x22	GPIO16	bi-directional	GPIO	MMIO16
0x23	GPIO17	bi-directional	GPIO	MMIO17
0x24	GPIO18	bi-directional	GPIO	MMIO18
0x25	GPIO19	bi-directional	GPIO	MMIO19
0x26	GPIO20	bi-directional	GPIO	MMIO20
0x27	GPIO21	bi-directional	GPIO	MMIO21
0x28	GPIO22	bi-directional	GPIO	MMIO22
0x29	GPIO23	bi-directional	GPIO	MMIO23
0x2a	GPIO24	bi-directional	GPIO	MMIO24
0x2b	GPIO25	bi-directional	GPIO	MMIO25
0x2c	GPIO26	bi-directional	GPIO	MMIO26
0x2d	GPIO27	bi-directional	GPIO	MMIO27
0x2e	GPIO28	bi-directional	GPIO	MMIO28
0x2f	GPIO29	bi-directional	GPIO	MMIO29
0x30	GPIO30	bi-directional	GPIO	MMIO30
0x31	GPIO31	bi-directional	GPIO	MMIO31
0x32	PHY0_LED0	push/pull output	internal PHY0 Status	
0x33	PHY0_LED1	push/pull output	internal PHY0 Status	
0x34	PHY0_LED2	push/pull output	internal PHY0 Status	
0x35	PHY0_LED3	push/pull output	internal PHY0 Status	
0x36	PHY1_LED0	push/pull output	internal PHY1 Status	

Coding	netX internal Function (core connection)	Signal Type	Functional Group	Default of
0x37	PHY1_LED1	push/pull output	internal PHY1 Status	
0x38	PHY1_LED2	push/pull output	internal PHY1 Status	
0x39	PHY1_LED3	push/pull output	internal PHY1 Status	
0x3a	MII_MDC	push/pull output	MDIO	
0x3b	MII_MDIO	bi-directional	MDIO	
0x3c	MII0_COL	input	MII0	
0x3d	MII0_CRS	input	MII0	
0x3e	MII0_LED0	input	MII0	
0x3f	MII0_LED1	input	MII0	
0x40	MII0_LED2	input	MII0	
0x41	MII0_LED3	input	MII0	
0x42	MII0_RXCLK	input	MII0	
0x43	MII0_RXD0	input	MII0	
0x44	MII0_RXD1	input	MII0	
0x45	MII0_RXD2	input	MII0	
0x46	MII0_RXD3	input	MII0	
0x47	MII0_RXDV	input	MII0	
0x48	MII0_RXER	input	MII0	
0x49	MII0_TXCLK	input	MII0	
0x4a	MII0_TXD0	tristateable output	MII0	
0x4b	MII0_TXD1	tristateable output	MII0	
0x4c	MII0_TXD2	tristateable output	MII0	
0x4d	MII0_TXD3	tristateable output	MII0	
0x4e	MII0_TXEN	push/pull output	MII0	
0x4f	MII0_TXER	push/pull output	MII0	
0x50	MII1_COL	input	MII1	
0x51	MII1_CRS	input	MII1	
0x52	MII1_LED0	input	MII1	
0x53	MII1_LED1	input	MII1	
0x54	MII1_LED2	input	MII1	
0x55	MII1_LED3	input	MII1	
0x56	MII1_RXCLK	input	MII1	
0x57	MII1_RXD0	input	MII1	
0x58	MII1_RXD1	input	MII1	
0x59	MII1_RXD2	input	MII1	
0x5a	MII1_RXD3	input	MII1	
0x5b	MII1_RXDV	input	MII1	
0x5c	MII1_RXER	input	MII1	
0x5d	MII1_TXCLK	input	MII1	
0x5e	MII1_TXD0	tristateable output	MII1	
0x5f	MII1_TXD1	tristateable output	MII1	
0x60	MII1_TXD2	tristateable output	MII1	

Coding	netX internal Function (core connection)	Signal Type	Functional Group	Default of
0x61	MII1_TXD3	tristateable output	MII1	
0x62	MII1_TXEN	push/pull output	MII1	
0x63	MII1_TXER	push/pull output	MII1	
0x64	PIO0	bi-directional	PIO	
0x65	PIO1	bi-directional	PIO	
0x66	PIO2	bi-directional	PIO	
0x67	PIO3	bi-directional	PIO	
0x68	PIO4	bi-directional	PIO	
0x69	PIO5	bi-directional	PIO	
0x6a	PIO6	bi-directional	PIO	
0x6b	PIO7	bi-directional	PIO	
0x6c	SPI0_CS2N	bi-directional	SPI0 3rd chip select	
0x6d	SPI1_CLK	bi-directional	SPI1	
0x6e	SPI1_CS0N	bi-directional	SPI1	
0x6f	SPI1_CS1N	bi-directional	SPI1	
0x70	SPI1_CS2N	bi-directional	SPI1	
0x71	SPI1_MISO	bi-directional	SPI1	
0x72	SPI1_MOSI	bi-directional	SPI1	
0x73	I2C_SCL_MMIO	bi-directional	I2C	
0x74	I2C_SDA_MMIO	bi-directional	I2C	
0x75	XC_SAMPLE0	input	Trigger/Latch Unit	
0x76	XC_SAMPLE1	input	Trigger/Latch Unit	
0x77	XC_TRIGGER0	tristateable output	Trigger/Latch Unit	
0x78	XC_TRIGGER1	tristateable output	Trigger/Latch Unit	
0x79	UART0_CTS	input	UART 0	MMIO32
0x7a	UART0_RTS	tristateable output	UART 0	MMIO33
0x7b	UART0_RXD	input	UART 0	MMIO34
0x7c	UART0_TXD	tristateable output	UART 0	MMIO35
0x7d	UART1_CTS	input	UART 1	
0x7e	UART1_RTS	tristateable output	UART 1	
0x7f	UART1_RXD	input	UART 1	
0x80	UART1_TXD	tristateable output	UART 1	
0x81	UART2_CTS	input	UART 2	
0x82	UART2_RTS	tristateable output	UART 2	
0x83	UART2_RXD	input	UART 2	
0x84	UART2_TXD	tristateable output	UART 2	
0x85	USB_ID_DIG	input	USB	MMIO36
0x86	USB_ID_PULLUP_CTRL	push/pull output	USB	
0x87	USB_RPD_ENA	push/pull output	USB	
0x88	USB_RPU_ENA	push/pull output	USB	
0x89	CCD_DATA0	input	CCD-Sensor	
0x8a	CCD_DATA1	input	CCD-Sensor	

Coding	netX internal Function (core connection)	Signal Type	Functional Group	Default of
0x8b	CCD_DATA2	input	CCD-Sensor	
0x8c	CCD_DATA3	input	CCD-Sensor	
0x8d	CCD_DATA4	input	CCD-Sensor	
0x8e	CCD_DATA5	input	CCD-Sensor	
0x8f	CCD_DATA6	input	CCD-Sensor	
0x90	CCD_DATA7	input	CCD-Sensor	
0x91	CCD_PIXCLK	input	CCD-Sensor	
0x92	CCD_LINE_VALID	input	CCD-Sensor	
0x93	CCD_FRAME_VALID	input	CCD-Sensor	
0xff	NC	force input state if unused		MMIO37, MMIO38, MMIO39

2.4 RESET – Reset Controller

This register controls the reset functions of the netX chip and indicates the reset state. The reset state shows which resets have occurred, since the last Power On Reset. In order to determine the source of the last reset, the firmware should evaluate and reset these bits during its start sequence. After a power on reset, the RESET_CTRL register is cleared completely.

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register

The access key protection for this register does not work in the first netX 50 chip revision. Please consult the netX 50 Errata Sheet for details, when writing code that accesses this register!

RESET_CTRL – Reset Control Register

0x1c00000c

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
reserved				EN_RSTOUTn	RSTOUTn	FIRMW_RES	FIRMW_STATUS3	FIRMW_STATUS2	FIRMW_STATUS1	FIRMW_STATUS0	reserved		DIS_XPEC1_RES	DIS_XPEC0_RES	Reserved										XPEC1_RES	XPEC0_RES	FIRMW_RES	HOST_RES	WDG_RES	RSTInn	

Bits	Name	Description	R/W	Default
31:27	reserved	-	R	0x00
26	EN_RSTOUTn	This bit enables the output driver of the reset out pin. If the driver is enabled the physical level can be set by bit 25. If the bit is cleared the reset out pin is in high impedance state.	R/W	0
25	RSTOUTn	Programmable reset: This bit controls the signal of the RSTOUT pin of the netX (if enabled by Bit 26). 1: RSTOUT pin is low. 0: RSTOUT pin is high.	R/W	0
24	FIRMW_RES	Setting this bit activates a firmware reset.	W	0
23	FIRMW_STATUS3	Bits 23:20 can be used to save a 4-Bit value which will not be cleared by any reset, except a power on reset.	R/W	0
22	FIRMW_STATUS2	Bits 23:20 can be used to save a 4-Bit value which will not be cleared by any reset, except a power on reset.	R/W	0
21	FIRMW_STATUS1	Bits 23:20 can be used to save a 4-Bit value which will not be cleared by any reset, except a power on reset.	R/W	0
20	FIRMW_STATUS0	Bits 23:20 can be used to save a 4-Bit value which will not be cleared by any reset, except a power on reset.	R/W	0
19:18	reserved	-	R	0x00
17	DIS_XPEC1_RES	The xpec1 module can set this bit to avoid being affected by a system reset. For ARM software, this bit is read only .	R	0

16	DIS_XPEC0_RES	The xpec0 module can set this bit to avoid being affected by a system reset. For ARM software, this bit is read only.	R	0
15:6	reserved	-	R	0x00
5	XPEC1_RES	reset from XPEC1: This reset is generated by xpec1 reset.	R/W	0
4	XPEC0_RES	reset from XPEC0: This reset is generated by xpec0 reset.	R/W	0
3	FIRMW_RES	reset from FIRMWARE: This reset is activated by arm software. The firmware reset, resets the netX chip completely (system reset) and sets this bit. It can be cleared by trying to set it through a write access.	R/W	0
2	HOST_RES	reset from Host interface: If the internal Host Interface module generates a reset, the netX chip will be completely reset (system reset) and this bit is set. It can be cleared by trying to set it through a write access. There is a 1ms wait state from setting the bit till execution of the reset request.	R/W	0
1	WDG_RES	reset from System WDG: If the netX watchdog is activated and a watchdog timeout occurs the, system will be reset and this bit is set. It can be cleared by trying to set it through a write access.	R/W	0
0	RSTINn	reset from external pin: The netX chip has an external reset input If this reset is activated the netX chip will be completely reset (system reset) and this bit is set. It can be cleared by trying to set it through a write access.	R/W	0

The firmware can disable the internal system reset signals to the XPEC modules, allowing these modules to continue to run even while the chip is performing a reset, however a power on reset will always reset the complete chip and hence also the XPECs.

Code Examples:

Firmware Reset:

```
/* Defines the POKE Macro */
#define POKE(addr, val) (*(volatile unsigned int *)(addr) = (unsigned int)(val))

/* Defines the RESET_CTRL - Register */
#define RESET_CTRL 0x1c00000c

int main (void)
{
    POKE (RESET_CTRL, 0x1000008);    /* Activates a FW-Reset */
}
```

2.5 Clock Control – Clock Generation and Control

The netX50 provides five modules in the 100MHz system clock domain that can separately be switched off, which may be done for power saving reasons, if some of these modules are not used in specific applications. However, most of the power is consumed by the Ethernet PHYs (~500mW) and IO pads (depending on external load). Switching off a clock domain will save less than 50mW.

Clock module enabling / disabling is done through register CLK_EN, which, also allows to enable two fieldbus clocks. These clocks are derived from an internal 400MHz clock by rate multipliers, allowing low jitter clocks, for fieldbus systems with a frequency that can not directly be derived from the 100MHz system clock. With the netX50, an external oscillator can be saved in that case, due to the additional rate multipliers).

CLK_EN – Global Clock Enable Register

0x1c000024

Use this register to disable modules completely for power saving purposes.

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register

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
reserved																				FB1	FB0	reserved	HIF	reserved	XMAC1	XMAC0	reserved	XPEC1	XPEC0		

Bits	Name	Description	R/W	Default
31:12	-	reserved	R	0x0
11	FB1	enables clock for fieldbus1: 1: use internally generated fb1clk to resample xMAC1 outputs 0: use external xm1_eclk to resample xMAC outputs	R/W	0x1
10	FB0	enables clock for fieldbus0: 1: use internally generated fb0clk to resample xMAC0 outputs 0: use external xm0_eclk to resample xMAC outputs	R/W	0x1
9	-	reserved	R	0x0
8	HIF	enables clock for HIF	R/W	0x1
7:6	-	reserved	R	0x0
5	XMAC1	enables clock for xMAC1	R/W	0x1
4	XMAC0	enables clock for xMAC0	R/W	0x1
3:2	-	reserved	R	0x0
1	XPEC1	enables clock for xPEC1	R/W	0x1
0	XPEC0	enables clock for xPEC0	R/W	0x1

The following registers are used to control frequencies of internal clocks for armclk200, usb_clk and fieldbus clocks. The 200MHz ARM clock is always in fixed relation to 100MHz system clock. Hence, ARM_CLK_RATE_MUL_ADD also changes the system frequency.

ARM_CLK_RATE_MUL_ADD – Rate Multiplier Add Value of System Clock**0x1c000014**

This register can be used to change internal system frequency (200MHz of ARM and 100MHz of system), however as proper netX functionality is only guaranteed for the default value, this register should not be modified unless there is good reason for doing so..

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out Locking Access Key
- 2.: write back Locking Access Key
- 3.: write desired value to this register

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
reserved																ARMCLK_RATE_MUL_ADD															

Bits	Name	Description	R/W	Default
31:9	-	reserved	R	0x0
8:0	ARMCLK_RATE_MUL_ADD	This value is added each clk400 cycle to armclk_rate_mul to generate armclk. Change value according to formula: $\text{armclk_rate_mul_add} = [\text{freq in MHz}] / 400 * 2^9$	R/W	0x100

USB_CLK_RATE_MUL_ADD – Rate Multiplier Add Value of USB clock**0x1c000018**

This register can be used to change the USB core clock frequency, however, as proper netX functionality is only guaranteed for the default value, this register should not be modified unless there is good reason for doing so..

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out Locking Access Key
- 2.: write back Locking Access Key
- 3.: write desired value to this register

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
USBCLK_RATE_MUL_ADD																UNUSED															

Bits	Name	Description	R/W	Default
31:16	USBCLK_RATE_MUL_ADD	This value is added each clk400 cycle to usbclk_rate_mul to generate usbclk. Change value according to formula: $\text{usbclk_rate_mul_add} = [\text{freq in MHz}] / 400 * 2^{16}$	R/W	0x1eb8
15:0	UNUSED	unused	R/W	0x51ec

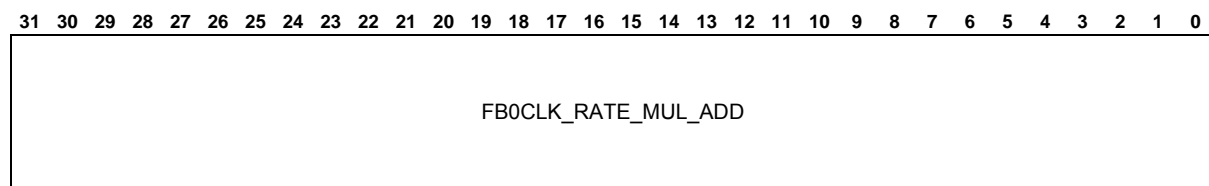
FB0CLK_RATE_MUL_ADD – Rate Multiplier Add Value**0x1c00001c**

Fieldbus0 clock is generated by internal 400MHz rate multiplier. At some fieldbus-frequencies, this clock has less jitter, than the xMAC generated output clock. xMAC fieldbus outputs (xm0_tx_out, xm0_tx_oe) can optionally (io_config-sel_xm0_eclk) be sampled by an extra register running on this clock, resulting in fieldbus outputs with less jitter.

Alternatively to this internally generated clock, an external clock (xm0_eclk) can be used to make xMAC outputs jitter free (clock_enable-fb0). Using external clocks to resample xMAC outputs requires modified xMAC software!.

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out Locking Access Key
- 2.: write back Locking Access Key
- 3.: write desired value to this register



Bits	Name	Description	R/W	Default
31:0	FB0CLK_RATE_MUL_ADD	This value is added each clk400 cycle to fb0clk_rate_mul to generate fb0clk. Change value according to formula: $\text{fb0clk_rate_mul_add} = [\text{freq in MHz}] / 400 * 2^{32}$	R/W	0x1000000

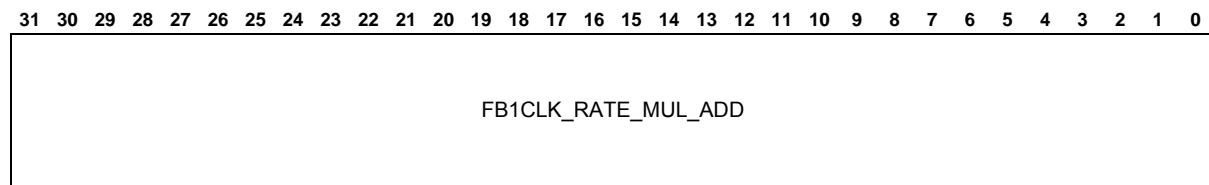
FB1CLK_RATE_MUL_ADD – Rate Multiplier Add Value**0x1c000020**

Fieldbus1 clock is generated by internal 400MHz rate multiplier. At some fieldbus-frequencies, this clock has less jitter, than the xMAC generated output clock. xMAC fieldbus outputs (xm1_tx_out, xm1_tx_oe) can optionally (io_config-sel_xm1_eclk) be sampled by an extra register running on this clock, resulting in fieldbus outputs with less jitter.

Alternatively to this internally generated clock, an external clock (xm1_eclk) can be used to make xMAC outputs jitter free (clock_enable-fb1). Using external clocks to resample xMAC outputs requires modified xMAC software!.

This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register



Bits	Name	Description	R/W	Default
31:0	FB1CLK_RATE_MUL_ADD	This value is added each clk400 cycle to fb1clk_rate_mul to generate fb1clk. Change value according to formula: $fb1clk_rate_mul_add = [freq \text{ in MHz}] / 400 * 2^{32}$	R/W	0x1000000

2.6 WDG - Watchdog

The netX system watchdog is used for supervision of the netX status. After power on reset, the watchdog timer is disabled. The firmware has to load the watchdog timer registers with two timeout values in order to arm the watchdog, which then has to be retriggered continuously.

One of the registers is used for setting a timeout which will generate an interrupt. The register for the second value comes in, when the first timeout has occurred. When the counter (which starts when the first timeout value is reached) reaches the second timeout value, a system reset is initiated.

The timer has a fixed time base of 100µs. The timeouts can be configured in a wide range. The following formula is used to calculate the desired values:

$$T_{\text{IRQ}} = \text{WDG_IRQ_TIMEOUT} \times 100 \mu\text{s}$$

$$T_{\text{RESET}} = (\text{WDG_IRQ_TIMEOUT} + \text{WDG_RESET_TIMEOUT}) \times 100 \mu\text{s}$$

The timeout register values are always loaded into the watchdog timer when the timer is being retriggered.

Note:

Writing bits [31:24] is only possible when writing the correct access code at the same time. Hence only 32 bit accesses to this register are possible. To get the next valid access code it is necessary to read the WDG_TRIG register. Bits [19:0] provide the new access code, which is generated by a pseudo random counter.

WDG_TRIG – Watchdog Trigger Register

0x1c000200

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
WR_ENABLE	reserved	WDG_OUTPUT_EN	WDG_TRIG	Reserved			IRQ_Status	reserved																							
WDG_ACCESS_CODE																															

Bits	Name	Description	R/W	Default
31:	WR_ENABLE	Write enable bit for timeout register. As long as this bit is not set all write accesses to the timeout register are ignored.	R/W	0
30	Reserved	-	R	0
29	WDG_ACT_EN	Watchdog Active Enable. If this bit is set, the WDGACT output signal (PIN G17) is enabled.	R/W	0
28	WDG_TRIG	Watchdog trigger bit. Bit must be set to trigger the watchdog counter. When read, this bit is always '0'.	W	0
27:25	Reserved	-	R	0x00
24	IRQ_Status	Interrupt request bit. Set by watchdog when an IRQ timeout has occurred. This bit can be cleared by trying to set it through a write access.	R/W	0
23:20	Reserved	-	R	0x00
19:0	WDG_ACCESS_CODE	Watchdog access code for retriggering the watchdog or modifying Bits 31:24. A read access provides the 20 bit code, to be written back with the next write access.	R/W	0x00

WDG_CNTR – Watchdog Counter**0x1c000204**

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
reserved																WDG_COUNTER															

Bits	Name	Description	R/W	Default
31:17	Reserved	-	R	0x00
16:0	WDG_COUNTER	Current watchdog counter value.	R	0x00

WDG_IRQ_TIMEOUT – Watchdog Interrupt Timeout**0x1c000208**

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
Reserved																WDG_IRQ_TIMEOUT															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	WDG_IRQ_TIMEOUT	Watchdog interrupt request timeout value.	R/W ¹	0x00

WDG_RESET_TIMEOUT – Watchdog Reset Timeout**0x1c00020c**

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
Reserved																WDG_RESET_TIMEOUT															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	WDG_RESET_TIMEOUT	Watchdog reset request timeout value.	R/W ¹	0x00

¹ Write access to the register is only possible when the WR_ENABLE bit in the Watchdog Trigger Register WDG_TRIG is set.

2.7 SYS_STAT – System Status

The general status of a netX based system is usually indicated by the System LED, which can either consist of a dual LED or two single LEDs (see netX Product Brief for further information).

SYS_STA – System Status

0x1c0034d8

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
reserved						RUN_DRV	RDY_DRV		reserved				RUN_POL	RDY_POL	RUN_IN	RDY_IN	NETX_STA_CODE								HOST_STATE[3:0]			NETX_STATE[3:2]		RUN	RDY

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x00
25	RUN_DRV	Driver enable for RUN LED. Enables output driver when set.	R/W	0
24	RDY_DRV	Driver enable for RDY LED. Enables output driver when set.	R/W	0
23:20	reserved	-	R	0x00
19	RUN_POL	Output polarity RUN LED; outsig = RUN exor RUN_POL	R/W	0
18	RDY_POL	Output polarity RUN LED; outsig = RDY exor RDY_POL	R/W	0
17	RUN_IN	Physical input signal level at RUN pin	R	0
16	RDY_IN	Physical input signal level at RDY pin	R	0
15:8	NETX_STA_CODE	netX status code. The netX status codes are software defined. The predefined code values are: F0h: Status after power on reset	R/W	0xf0
7:4	HOST_STATE[3:0]	User defined status signals	R	0x00
3:2	NETX_STATE[3:2]	User defined status signals	R/W	0
1	RUN	Signal Level of the RUN LED output	R/W	0
0	RDY	Signal level of the RDY LED output	R/W	0

Hint: The lower 16 bits can also be accessed from an external host processor through the dual-port memory interface, while Bits (7:4) can be written and all other Bits can be read only.

Note:

If the netX50 system has not only System LED(s), but also a security EEPROM connected to the RDY and RUN signals, programmers must take care to avoid presenting an I2C start condition on the RDY / RUN pins when turning on and off the System LED or changing their colour. This means, that prior to changing the status of the RUN pin (low to high or high to low), the RDY pin must always be driven to a low level!

Code Example:

```
/* Defines the POKE Macro */
#define POKE(addr, val) (*(volatile unsigned int *)(addr) = (unsigned int)(val))

/* Defines the System-Status - Register */
#define SYS_STA 0x1c0034d8

int main (void)
{
    /* Turn ON the SYS-LED to Green */
    POKE (SYS_STA, 0x030c0002);

    /* Turn ON the SYS-LED to Red */
    POKE (SYS_STA, 0x030c0001);

    /* Turn OFF the SYS-LED */
    POKE (SYS_STA, 0x030c0000);
}
```

3 Memory Controller

The Memory Controller can drive SRAM, FLASH and SDRAM without any additional glue logic. The bus width and timing parameters can be set individually for each memory chip select area. SRAM and FLASH data bus width can be set to 8, 16 or 32 Bit, SDRAM data bus width to 16 or 32 Bit.

The following table provides an address overview about each external memory chip select area:

ARM Address area	Short Description
0x80000000 – 0xbfffffff	SDRAM Chip Select Area
0xc0000000 – 0xc7ffffff	SRAM/FLASH Chip Select Area 0
0xc8000000 – 0xcfffffff	SRAM/FLASH Chip Select Area 1
0xd0000000 – 0xd7ffffff	SRAM/FLASH Chip Select Area 2

Note:

According to address area table above, SRAM or FLASH chip select areas are not fully accessible by the 24 address lines of the netX Memory Controller. Non accessible upper address areas are mirrors of the first 16 MByte (8 Bit area) 32 MByte (16 Bit area) or 64 MByte (32 Bit area).

The followings table provides a summary of memory controller registers.

ARM Address	Register Name	Short Description
0x1c000100	MEM_SRAM0_CTRL	Memory SRAM Control Register for Chip Select Area 0
0x1c000104	MEM_SRAM1_CTRL	Memory SRAM Control Register for Chip Select Area 1
0x1c000108	MEM_SRAM2_CTRL	Memory SRAM Control Register for Chip Select Area 2
0x1c000140	MEM_SDRAM_CFG_CTRL	Memory SDRAM Configuration Control Register
0x1c000144	MEM_SDRAM_TIMING_CTRL	Memory SDRAM Timing Control Register
0x1c000148	MEM_SDRAM_MODE	Memory SDRAM Mode Register
0x1c000180	MEM_PRIO_TIMESLOT_CTRL	Memory Priority Timeslot Control Register
0x1c000184	MEM_PRIO_ACCESS_CTRL	Memory Priority Access Control Register

3.1 MEM_SRAM – Memory Controller for SRAM and FLASH

The bus width and wait state parameters can be set individually for each memory area, allowing bus widths of 8, 16 or 32 Bit and wait states of up to 63 clock cycles. Additionally a pre-pause and a post-pause each 0 to 3 clock cycles can be configured. Pre-pauses are inserted between chip select assertion and output- or write-enable and data signal generation to avoid driving conflicts with prior accesses at access start. Post-pauses are inserted between output- or write-enable signal release and chip select signal release at access end to avoid driving conflicts with following accesses (e.g. by long read data hold times of devices).

Note:

Internal architecture of all netX devices is 32 Bit oriented. 32 Bit accesses to 8 or 16 Bit SRAM or FLASH devices are split into 2 or 4 appropriate external accesses. If external devices require output enable signal toggling between subsequent read accesses, at least one pre- or post-pause must be configured. If not, netX runs pseudo read bursts (chip select and output enable signals always active low, only address lines changing) in case of subsequent external addresses.

For further information on the SRAM Memory Controller, please refer to the netX50 Technical Reference Guide.

MEM_SRAM0_CTRL – Memory SRAM Control Register for Chip Select Area 0 **0x1c000100**
MEM_SRAM1_CTRL – Memory SRAM Control Register for Chip Select Area 1 **0x1c000104**
MEM_SRAM2_CTRL – Memory SRAM Control Register for Chip Select Area 2 **0x1c000108**

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
reserved						WIDTHEXTMEM		reserved						WSPPOSTPAUSEEXTMEM		reserved						WSPREPASEEXTMEM		reserved		WSEXTMEM					

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x00
25:24	WIDTHEXTMEM	Data path width of ExtMem0 area. The external address bus will be shifted according to the selected memory width. So the external maximum address range varies depending on the data bus width. 00 : 8 bit A _{ext} [23:0] reflect the byte address 01 : 16 bit A _{ext} [23:0] reflect the word address 10 : 32 bit A _{ext} [23:0] reflect the dword address 11 : reserved	R/W	00
23:18	reserved	-	R	0x00
17:16	WSPPOSTPAUSEEXTMEM	additional wait states after access (0 – 3 cycles)	R/W	11
15:10	reserved	-	R	0x00
9:8	WSPREPASEEXTMEM	additional wait states for setup time nCS, Aext to nOE, nWE (0 – 3 cycles)	R/W	11
7:6	reserved	-	R	00
5:0	WSEXTMEM	Wait states (0 – 63 cycles)	R/W	11111

3.2 MEM_SDRAM – Memory Controller for SDRAM

The following parameters can be set:

- Number of banks 2, 4, 8
- Number of rows 2k, 4k, 8k, 16k, 32k, 64k
- Number of columns 256, 512, 1k, 2k, 4k, 8k, 16k
- Data width 16 Bit, 32 Bit
- Timing Parameters
- Refresh-mode priority adjustable in 4 steps
- Power save mode SDRAM-Self-refresh-Mode with disabled clock switch on / off SDRAM Controller

Note:

For further information on the SDRAM Memory Controller, please refer to the netX50_Product_Brief and Technical Reference Guide.

MEM_SDRAM_CFG_CTRL – Memory SDRAM Configuration Control Register

0x1c000140

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
REFRESH_REQUEST	SDRAM_READY	reserved				REFRESH_MODE	reserved				CTRL_EN	EXTCLK_EN	SDRAM_PWDN	DBUS32	reserved				COLUMNS	reserved	ROWS	reserved	BANKS								

Bits	Name	Description	R/W	Default
31	REFRESH_REQUEST	Refresh request flag. Refresh generation has always lower priority than accesses on external memory interface. This bit is set by the SDRAM Controller if there was too much traffic to SDRAM to run refreshes according to programmed refresh_mode. If external traffic to SDRAM is not predictable or there might be too much traffic, this bit should be read by software periodically. If it is set, traffic to SDRAM should be reduced to avoid critical refresh situation. This can be done by timer IRQ handler. This bit can be cleared by writing '0'.	R/W	0
30	SDRAM_READY	This bit is set when SDRAM is ready for access. If Bit CTRL_EN is cleared or SDRAM_PWDN is set, SDRAM_READY will automatically be cleared. It will be set after SDRAM has been initialized or after power down wake up.	R/W	0
29:26	reserved	-	R	0x00
25:24	REFRESH_MODE	Refresh request generation mode: 00 : fixed interval (T_REFI us, refresh has highest priority) 01 : collect up to 8 refreshes (data access has higher priority than refresh, default) 10 : collect up to 16 refreshes (data access has higher priority than refresh) 11 : collect up to 2047 refreshes (for SDRAM only, data access has higher priority than refresh)	R/W	0x01
23:20	reserved	-	R	0x00

19	CTRL_EN	SDRAM controller enable The MEM_SDRAM_TIMING_CTRL and the MEM_SDRAM_MODE registers can only be written while this bit is cleared. Upon setting this bit, the SDRAM controller will be enabled, running the following SDRAM initialization procedure (100 MHz, t_clk = 10 ns). NOP (200 us = 20,000 t_clk, running sd_clk (if extclk_en), n_cs low, cke high) PRECHARGE ALL NOP (160 ns = 16 t_clk) 7x (AUTO REFRESH, NOP (310 ns = 31 t_clk)) AUTO REFRESH NOP (220 ns = 22 t_clk) LOAD MODE REGISTER (with settings done by these configuration registers) NOP (40 ns = 4 t_clk) ACTIVATE (for first access, if requested, sdram_ready will be set to 1 here) Accesses requested before sdram_ready is 1 will be blocked (no ready). The external SDRAM-clk will not run if the controller is disabled.	R/W	0
18	EXTCLK_EN	external SDRAM clock enable	R/W	0
17	SDRAM_PWDN	SDRAM Power Down If this bit is set, the Controller will move SDRAM to power down self refresh mode (no data loss) and stop the external SDRAM clock.	R/W	0
16	DBUS32	SDRAM data bus width 0 : SDRAM bus is 16 bit wide. (default) 1 : SDRAM bus is 32 bit wide.	R/W	0
15:11	reserved	-	R	0x00
10:8	COLUMNS	column address coding. 000 : 256 (A0..A7) (default) 001 : 512 (A0..A8) 010 : 1k (A0..A9) 011 : 2k (A0..A9,A11) 100 : 4k (A0..A9,A11,A12) 101 : 8k (A0..A9,A11..A13) (for future devices) 110 : 16k (A0..A9,A11..A14) (for future devices) 111 : reserved	R/W	
7	reserved	-	R	0
6:4	ROWS	row address coding. 000 : 2k (A0..A10) (default) 001 : 4k (A0..A11) 010 : 8k (A0..A12) 011 : 16k (A0..A13) 100 : 32k (A0..A14) (for future devices) 101 : 64k (A0..A15) (for future devices) 11x : reserved	R/W	0x00
3:2	reserved	-	R	0x00
1:0	BANKS	bank address coding; mapped to A18 (=BA2), A17 (=BA1) and A16(BA0). 00 : 2 01 : 4 (default) 10 : 8 11 : reserved	R/W	0x01

MEM_SDRAM_TIMING_CTRL – Memory SDRAM Timing Control Register**0x1c000144**

This register can only be modified while the SDRAM-Controller is disabled (Bit CTRL_EN of MEM_SDRAM_CFG_CTRL is cleared). It holds the timing parameters for the selected SDRAM type (please consult the data sheet of the SDRAM component for correct timing settings)

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
reserved			BYPASS_NEG_DELAY	reserved		DATA_SAMPLE_PHASE		MEM_SDCLK_SSNEG		MEM_SDCLK_PHASE		reserved		T_REFI		T_RFC		reserved		T_RAS		T_RP		T_WR		reserved		T_RCD			

Bits	Name	Description	R/W	Default
31:29	reserved	-	R	00
28	BYPASS_NEG_DELAY	0 : use phase shifted (negative delayed) SDRAM loopback clock for data sampling. 1 : bypass phase shift logic for SDRAM data sampling use SDRAM loopback clock for data sampling (for system clock frequency ≤ 80 MHz)	R/W	0
27	reserved	-	R	0
26:24	DATA_SAMPLE_PHASE	0..5: adjustable phase-shift for data sampling SDRAM loopback clock (clk_sdloopback) depending external capacitive load and SDRAM access time (t_AC). The phase can be shifted in 1.25 ns steps. clk_sdloopback will internally rise (sample SDRAM read data) at the data_sample_phase+4th clk400 edge after rise of external mem_sdclk (including external capacitive load). For correct settings, the delays depending on external capacitive have to be considered. Data sampling has to be done at least 8 ns after internal changes of SDRAM ctrl-signals (mem_sd*-signals, driven by clk_memsig) .	R/W	011
23	MEM_SDCLK_SSNEG	Bit must be set	R/W	1
22:20	MEM_SDCLK_PHASE	0..5: adjustable phase-shift for external SDRAM clock depending on external capacitive load on mem_sdclk-signal to match SDRAM ctrl-signal setup times. The phase can be shifted in 1.25 ns steps. mem_sdclk will internally rise at the mem_sdclk_phase+2nd clk400 edge after internal changes of SDRAM ctrl-signals (mem_sd*-signals, driven by clk_memsig), where the 1st egde is defined by the mem_sdclk_ssneq-bit. For correct settings, delays depending on external capacitive have to be considered.	R/W	000
19:18	reserved	-	R	00
17:16	T_REFI	Average Periodic refresh interval (time = 3.90 us * 2^T_REFI) 00 : 3.90 us 01 : 7.80 us (default) 10 : 15.60 us 11 : 31.20 us	R/W	01

15:12	T_RFC	<p>REFRESH to Command time</p> <p>Minimum time interval between AUTOREFRESH command and next command (next AUTOREFRESH or ACTIVE). In some SDRAM datasheets also called auto refresh period.</p> <p>0x00 : 4 clks 0x01 : 5 clks : 0x0f : 19 clks (default)</p>	R/W	1111
11	reserved	-	R	0
10:8	T_RAS	<p>ACTIVE to PRECHARGE command time</p> <p>Minimum time interval between ACTIVE command (row address and RAS asserted) and PRECHARGE command (deactivation of row). In some SDRAM datasheets also called row active time.</p> <p>000 : 3 clks 001 : 4 clks : 111 : 10 clks (default)</p>	R/W	111
7:6	T_RP	<p>PRECHARGE command period time (Precharge to command)</p> <p>Minimum time interval between PRECHARGE (deactivation of row) and next command.</p> <p>00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved</p>	R/W	11
5:4	T_WR	<p>WRITE recovery time</p> <p>Minimum time interval between last write data (data in) and Precharge command (deactivation of row).</p> <p>00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved</p>	R/W	11
3:2	reserved	-	R	00
1:0	T_RCD	<p>ACTIVE to READ or WRITE time</p> <p>Minimum time interval between ACTIVE command (row address and RAS asserted) and READ or WRITE command (column address and CAS asserted). In some SDRAM datasheets also called RAS-to-CAS time. This parameter will also be used to configure t_RRD (minimum time between consecutive ACTIVE commands to different banks)</p> <p>00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved</p>	R/W	11

MEM_SDRAM_MODE – Memory SDRAM Mode Register**0x1c000148**

The SDRAM Controller will initialize the Mode Register of the connected SDRAM component(s) with the value, stored in this register, after enabling the SDRAM Controller (200 µs SDRAM memory initialization procedure).

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
reserved		EMR														reserved	reserved						MR_CAS		reserved	MR_BURST					

Bits	Name	Description	R/W	Default
31:30	reserved	-	R	00
29:16	EMR	Extended SDRAM Mode Register. These bits are reserved and must not be modified!	R/W	0x00
15:14	reserved	-	R	00
13:7	reserved	SDRAM Mode Register. Reflect bits [13:7] of mode register of SDRAM component. These bits are reserved and must not be modified!	R/W	00 0000 0
6:4	MR_CAS	SDRAM Mode Register. Reflect CAS latency bits of mode register of SDRAM component (Bits [6:4]). The controller supports two settings: 010: CL2 011: CL3 (default)	R/W	011
3	reserved	SDRAM Mode Register. Reflects bit [3] of mode register of SDRAM component. This bit is reserved and must not be modified!	R/W	0
2:0	MR_BURST	SDRAM Mode Register. Reflect Burst Length Bits of mode register of SDRAM component (Bits [2:0]). The controller, which is fixed to 4 DWORD Bursts, only supports two settings, which are automatically done by the DBUS32 Bit in the MEM_SDRAM_CFG_CTRL register: 010: Burst Length 4 (data width 32 Bit) 011: Burst Length 8 (data width 16 Bit, default)	R	011

3.3 MEM_PRIO – Memory Priority Controller

MEM_PRIO_TIMESLOT_CTRL – Memory Priority Timeslot Control Register

0x1c000180

Memory interface master timeslot priority control register.

Note:

Any master can access in one timeslot $((ts_accessrate_mX * ts_length_mX) / 64) + 1$ times (i.e. at maximum $(ts_accessrate_mX) / 64$ bandwidth on external memory bus, $ts_accessrate_mX$ is programmed by MEM_PRIO_ACCESS_CTRL register).

Priority control will watch data accesses on external memory data bus (SDRAM and non SDRAM), including pauses on non SDRAM-accesses, not including control commands to SDRAM. Any master requesting more accesses will be forced to wait for the remaining timeslot.

master channel m0: Host Bus Interface (highest priority)

master channel m1: XC

master channel m2: ARM channel (instruction and data) (lowest priority)

master channel m3: DMA-Controller

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
reserved																	TS_LENGTH_DMA_MI		reserved		TS_LENGTH_ARM_MI		reserved		TS_LENGTH_XC_MI		reserved		TS_LENGTH_HIF_MI		

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x00
14:12	TS_LENGTH_DMA_MI	0..7: the timeslot of master m3 (ARM instruction fetch) is on external memory interface $64 * 2^{ts_length_DMA_mi}$ system clock cycles		111
11	reserved	-	R	0
10:8	TS_LENGTH_ARM_MI	0..7: the timeslot of master m2 is on external memory interface $64 * 2^{ts_length_ARM_mi}$ system clock cycles		111
7	reserved	-	R	0
6:4	TS_LENGTH_XC_MI	0..7: the timeslot of master m1 is on external memory interface $64 * 2^{ts_length_XC_mi}$ system clock cycles		111
3	reserved	-	R	0
2:0	TS_LENGTH_HIF_MI	0..7: the timeslot of master m0 is on external memory interface $64 * 2^{ts_length_HIF_mi}$ system clock cycles		111

MEM_PRIO_ACCESS_CTRL – Memory Priority Access Control Register**0x1c000184**

Control Register for master channel accesses per timeslot on external memory interface.

This register may be partially locked by the LOCK_MEM_PRIO_CTRL register in ASIC_CTRL address area.

For detailed priority controlling read note at MEM_PRIO_TIMESLOT_CTRL register description.

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
reserved								TS_ACCESSRATE_DMA_MI				TS_ACCESSRATE_ARM_MI				TS_ACCESSRATE_XC_MI				TS_ACCESSRATE_HIF_MI											

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	00
23:18	TS_ACCESSRATE_DMA_MI	0..63: master m3 (ARM instruction fetch) is allowed to request $((ts_accessrate_ARMI_mi * ts_length_DMA_mi) / 64) + 1$ accesses on external memory	R/W	111111
17:12	TS_ACCESSRATE_ARM_MI	0..63: master m2 is allowed to request $((ts_accessrate_LCD_mi * ts_length_ARM_mi) / 64) + 1$ accesses on external memory	R/W	111111
11:6	TS_ACCESSRATE_XC_MI	0..63: master m1 is allowed to request $((ts_accessrate_XC_mi * ts_length_XC_mi) / 64) + 1$ accesses on external memory	R/W	111111
5:0	TS_ACCESSRATE_HIF_MI	0..63: master m0 is allowed to request $((ts_accessrate_HIF_mi * ts_length_HIF_mi) / 64) + 1$ accesses on external memory	R/W	111111

4 Extension Bus

In order to use the Extension Bus, the netX host interface must be configured for 'Extension Bus Mode' in register DPM_ARM_IF_CFG0. Further, it is also important to configure each required signal line of the Extension Bus for host interface mode in registers DPM_ARM_IO_MODE0 and DPM_ARM_IO_MODE1 (please refer to section 5.2: DPM_ARM – Dual-Port Memory ARM Side). Unused signal lines of the Extension Bus may be left configured as I/O mode (e.g. unused upper address lines or data lines 15-8 when using an 8 Bit device only).

Please note, that the configuration for Extension Bus is done separately for each chip select, resulting in a set of four identical registers. For details on the Extension bus timing parameters, please refer to the appropriate chapters of the “netX50 Technical Reference Guide”.

EXT_CFG_CS0 – Extension Bus Configuration Chip Select 0 **0x1c003610**
EXT_CFG_CS1 – Extension Bus Configuration Chip Select 1 **0x1c003614**
EXT_CFG_CS2 – Extension Bus Configuration Chip Select 2 **0x1c003618**
EXT_CFG_CS3 – Extension Bus Configuration Chip Select 3 **0x1c00361c**

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				
Talewidth				Tadrhold				Tcson				Trdon				Twron				Trdwroff				Trdwrcyc				WAIT_POLARITY	WAIT_EN	nRD_MODE	DS_MODE	nWR_MODE	8/16BIT		CS_EN

Bits	Name	Description	R/W	Default
31:29	Talewidth	Delay time from start of cycle until ALE inactive	R/W	0x00
28:26	Tadrhold	Delay time from start of cycle until invalid address at the data bus.	R/W	0x00
25:23	Tcson	Delay time from start of cycle until Chip Select low	R/W	0x00
22:20	Trdon	Delay time from start of cycle until RD low in system clocks	R/W	0x00
19:17	Twron	Delay time from start of cycle until WR low in system clocks	R/W	0x00
16:12	Trdwroff	Delay time from start of cycle until RD and WR inactive for accesses in system clocks	R/W	0x00
11:7	Trdwrcyc	Set the end of an access cycle in system clocks 0x00 .. 0x01 = 320 ns 0x02 .. 0x0F = 20 ns ... 150 ns 0x10 = 320 ns 0x11 = 340 ns 0x12 = 360 ns ... 0x1F = 620 ns	R/W	0x00
6	WAIT_POLARITY	WAIT input polarity sampled before rising edge of RD/WR end to lengthen a cycle.	R/W	0
5	WAIT_EN	External Wait Enable	R/W	0
4	nRD_MODE	nRD mode. When this bit is zero the interface pin function as normal low active read signal. If the bit is set the pin signals the data direction for each transfer, RD/#WR = DIR	R/W	0
3	DS_MODE	Data Strobe Mode 0 nWR/nWRL/nWRH 1 nDS/nDSL/nDSH	R/W	0
2	nWR_MODE	nWR mode. If this bit is set the nWR signal is only active for low byte write accesses (nWRL).	R/W	

1	8/16BIT	Interface width selection: 0 = 8 Bit, 1 = 16 Bit	R/W	0
0	CS_EN	Chip Select Enable	R/W	0

5 Dual-Port Memory

In order to use the netX Dual-port memory interface, the netX host interface must be configured to 'µP Bus Mode' in the DPM_ARM_IF_CFG0 register. Further, it is also important to configure each required signal line of the DPM to host interface mode in the DPM_ARM_IO_MODE0 and DPM_ARM_IO_MODE1 registers. Unused signal lines of the DPM may be left configured as I/O mode (e.g. data lines 15-8 when operating in 8 Bit mode only).

For detailed information about DPM, please refer to the appropriate chapters of the “netX50 Technical Reference Guide” .

5.1 DPM_HOST – Dual-Port Memory Host Side

The following two tables show the Dual-port memory structure as it is seen from the host side. The first table shows the global configuration area with control and status registers, always located between 0xfe00 and 0xffff.

Host Address	Arm Address	Register Name	Short Description
0xfffc	0x1c0031fc	reserved	-
0xffff8	0x1c0031f8	reserved	-
0xffff4	0x1c0031f4	reserved	-
0xffff0	0x1c0031f0	DPM_HOST_INT_EN0	Interrupt Enable 0
0xffec	0x1c0031ec	reserved	-
0xffe8	0x1c0031e8	reserved	-
0xffe4	0x1c0031e4	reserved	-
0xffe0	0x1c0031e0	DPM_HOST_INT_STAT0	Interrupt Status 0
0xffdc	0x1c0031dc	DPM_HOST_RESET_REQ	Reset Request
0xffd8	0x1c0031d8	DPM_HOST_SYS_STAT	System Status
0xffd4	0x1c0031d4	DPM_HOST_TMR_START_VAL	Timer Start Value
0xffd0	0x1c0031d0	DPM_HOST_TMR_CTRL	Timer Control
0xffcc	0x1c0031cc	reserved	-
0xffc8	0x1c0031c8	DPM_HOST_WDG_ARM_TIMEOUT	Watchdog ARM Timeout
0xffc4	0x1c0031c4	DPM_HOST_WDG_HOST_TRIG	Watchdog Host Trigger
0xffc0	0x1c0031c0	DPM_HOST_WDG_HOST_TIMEOUT	Watchdog Host Timeout, read only
0xffbc - 0xfe00	0x1c0031bc-0x1c003100	reserved	-

Address space from 0x0000 to 0xfdfc is data memory area. *The following table is only an example, as the actual DPM memory layout is user programmable and hence completely firmware dependant.*

Host Address	Short Description		
0xfdfc - 0xa000	Unused area with nearly 24k bytes		
0x9fff - 0x9ffc	Host→netX Handshake 16 bit	netX→Host Handshake 16 bit	
0x9ffb - 0x3800	Data Memory Block 8 with nearly 10k bytes		
0x37ff - 0x3000	Data Memory Block 7 with 2k bytes		
0x6fff - 0x6000	Data Memory Block 6 with 4k bytes		
0x5fff - 0x5000	Data Memory Block 5 with 4k bytes		
0x4fff - 0x4ffc	Host→netX Handshake	netX→Host Handshake	Handshake Data Memory 2 Byte

0x4ffb - 0x2800	Data Memory Block 4 with nearly 10k bytes
0x27ff - 0x2000	Data Memory Block 3 with 2k bytes
0x1fff - 0x1000	Data Memory Block 2 with 4k bytes
0x0fff - 0x0000	Data Memory Block 1 with 4k bytes

The DPM host side registers are described following:

DPM_HOST_INT_EN0 – DPM Host Side Interrupt Enable 0

0xfff0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	13	16	15	14	13	12	11	10	9	8	3	6	5	4	3	2	1	0
INT_EN	MEM_LCK	WDG_NETX	reserved		SYS_STA	TMR	reserved									HS_EVENT[15:0]															

Bits	Name	Description	R/W	Default
31	INT_EN	Global interrupt enable	R/W	0
30	MEM_LCK	Memory lock interrupt enable	R/W	0
29	WDG_NETX	Watchdog timeout of netX supervision interrupt enable	R/W	0
28:27	reserved	-	R	0x00
26	SYS_STA	System status change interrupt enable	R/W	0
25	TMR	Timer interrupt enable	R/W	0
24:16	reserved	-	R	0x00
15	HS_EVENT15	Handshake event 15 interrupt enable	R	0
14	HS_EVENT14	Handshake event 14 interrupt enable	R	0
13	HS_EVENT13	Handshake event 13 interrupt enable	R	0
12	HS_EVENT12	Handshake event 12 interrupt enable	R	0
11	HS_EVENT11	Handshake event 11 interrupt enable	R	0
10	HS_EVENT10	Handshake event 10 interrupt enable	R	0
9	HS_EVENT9	Handshake event 9 interrupt enable	R	0
8	HS_EVENT8	Handshake event 8 interrupt enable	R	0
7	HS_EVENT7	Handshake event 7 interrupt enable	R	0
6	HS_EVENT6	Handshake event 6 interrupt enable	R	0
5	HS_EVENT5	Handshake event 5 interrupt enable	R	0
4	HS_EVENT4	Handshake event 4 interrupt enable	R	0
3	HS_EVENT3	Handshake event 3 interrupt enable	R	0
2	HS_EVENT2	Handshake event 2 interrupt enable	R	0
1	HS_EVENT1	Handshake event 1 interrupt enable	R	0
0	HS_EVENT0	Handshake event 0 interrupt enable	R	0

This register controls the host side interrupt behaviour of the chip. The interrupt status flags will be always set upon the corresponding event, but the physical interrupt line will only be active when the enable bit is set. All interrupt requests can be enabled or disabled independently and are wired together to the global interrupt request for the host interface, however no interrupt will be generated unless the global interrupt enable bit (bit 31) is also set.

DPM_HOST_INT_STAT0 – DPM Host Side Interrupt Status 0**0xffe0**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	13	16	15	14	13	12	11	10	9	8	3	6	5	4	3	2	1	0
INT_REQ	MEM_LCK	WDG_NETX	reserved		SYS_STA	TMR	reserved	IRQ_VECTOR[7:0]								HS_EVENT[15:0]															

Bits	Name	Description	R/W	Default
31	INT_REQ	Interrupt request	R	0
30	MEM_LCK	Flag of Memory lock interrupt	R/W	0
29	WDG_NETX	Flag of Watchdog timeout netX supervision interrupt	R/W	0
28:27	reserved	-	R	0x00
26	SYS_STA	Flag of System status change interrupt	R/W	0
25	TMR	Flag of Timer interrupt	R/W	0
24	reserved	-	R	0x00
23:16	IRQ_VECTOR[7:0]	Interrupt Vector according to the status flags	R	0x00
		0x00 : no interrupt		
		0x10 : handshake event interrupt 0		
		0x11 : handshake event interrupt 1		
		0x12 : handshake event interrupt 2		
		0x13 : handshake event interrupt 3		
		0x14 : handshake event interrupt 4		
		0x15 : handshake event interrupt 5		
		0x16 : handshake event interrupt 6		
		0x17 : handshake event interrupt 7		
		0x18 : handshake event interrupt 8		
		0x19 : handshake event interrupt 9		
		0x1a : handshake event interrupt 10		
		0x1b : handshake event interrupt 11		
		0x1c : handshake event interrupt 12		
		0x1d : handshake event interrupt 13		
		0x1e : handshake event interrupt 14		
		0x1f : handshake event interrupt 15		
		0x20 - 0x5f : reserved		
		0x60 : Memory lock interrupt		
		0x61 : Watchdog timeout ARM supervision		
		0x62 - 0x6f : reserved		
		0x70 : System status change interrupt		
		0x71 : Timer interrupt		
		0x72 - 0xff : Reserved		
15:0	HS_EVENT[15:0]	Flag of Handshake event [15:0] interrupt	R/W	0x00

This register does not only show the interrupt status flag of each interrupt event, but also provides the interrupt vector. The flags are always set when the corresponding interrupt events occur, regardless of the interrupt enable flag, but the physical interrupt which is reflected by the interrupt vector, will only be generated when the corresponding enable flag is set. An interrupt status flag can be cleared by trying to set the flag through a write access. When a handshake register cell is read, the corresponding interrupt status flag will be cleared automatically. The interrupt flag will be not cleared when the interrupt event reoccurs at the same time.

DPM_HOST_RESET_REQ – DPM Host Side Reset Request**0xffdc**

The host system can cause a chip reset by writing a special sequence to this register. After writing the last word of the sequence, the netX will perform a reset after a delay of 1 ms.

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
reserved																DEL_CNT[16:8]				DEL_CNT[7:0] CONTROL[7:0]											

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x00
16:8	DEL_CNT[16:8]	Delay counter value (not linear counting)	R	0x00
7:0	DEL_CNT[7:0] CONTROL[7:0]	If the written value is not equal to the delay counter then the delay counter is reset to 0000h. Writing the same value as read (only low byte) will switch one cycle of delay counter step. When the delay counter switches eight times and reaches the 80h value the delay counter will start to count up automatically and could not be stopped. After 1 ms the netX performs a hardware reset.	R/W	0x00

DPM_HOST_SYS_STAT – DPM Host Side System Status**0xffd8**

The netX provides an ARM side system status register, called 'SYS_STAT', which shows the status of the netX chip. Some bits of this register some can be controlled from netX side and some bits can be controlled from host side. The DPM_HOST_SYS_STAT register is an alias mapping to the SYS_STAT register, but only the lower 16 bits can be accessed, and only bits(7:4) can be written from the host side. It is possible to enable a system status interrupt for the host system, then any write access from netX side to the data will generate an interrupt to the host.

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
Reserved																NETX_STA_CODE[7:0]				HOST_STATE [3:0]				NETX_STATE[3:2]				RUN		RDY	

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:8	NETX_STA_CODE	netX status code. The netX status codes are software defined. The predefined code values are: F0h: Status during and after power on reset	R	0xf0
7:4	HOST_STATE[3:0]	User defined host status signals	R/W	0x00
3:2	NETX_STATE[1:0]	User defined netX status signals	R	0x00
1	RUN	Signal level of the RUN LED output	R	0
0	RDY	Signal level of the RDY LED output	R	0

See also chapter 2.7 (System Status).

DPM_HOST_TMR_START_VAL – DPM Host Side Timer Start Value**0xffd4**

The first step in programming the timer is to set the timer start count value DPM_HOST_TMR_START in the 'DPM_HOST_TMR_START_VAL' register. The timer operation is controlled by the 'DPM_HOST_TMR_CTRL' register. When setting the start flag, the counter will be loaded with the timer start count value. This can also be done during timer operation to reload the timer.

The clock divider field provides the possibility to select different clock time bases. The divider for the time base generation will be reset to its default value, when the counter start flag is being set or the divider value is being changed.

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
reserved																DPM_HOST_TMR_START															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	DPM_HOST_TMR_START	Timer start value for count down or cyclic reload. The timeout time can be calculated by the following formula: $T_{\text{TIMEOUT}} = \text{TMR_START} \times \text{Time base}$	R/W	0x00

DPM_HOST_TMR_CTRL – DPM Host Side Timer Control**0xffd0**

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
reserved																START	reserved										FNCT	CLKDIV			

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15	START	Start Timer Count. Writing a one bit will load the timer with the timer start value and then the timer counts down to zero. A zero bit stops the timer from counting.	R/W	0
14:4	reserved	-	R	0x00
3	FNCT	Timer mode function 0 : Timer stops after count down to zero 1 : Timer start again with start value (reload mode)	R/W	0
2:0	CLKDIV	Timer clock divider 000 : 100 µs 001 : 10 µs 010 : 1 µs 011 : 100 ns 1xx : Reserved (100µs)	R/W	0x00

DPM_HOST_WDG_ARM_TIMEOUT – DPM Host Side Watchdog ARM**0xffc8**

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
reserved																TIMEOUT_VAL															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	TIMEOUT_VAL	Timeout value $T_{\text{TIMEOUT}} = \text{TIMEOUT_VAL} \times 100 \mu\text{s}$	R/W	0x00

DPM_HOST_WDG_HOST_TRIG – DPM Host Side Watchdog Host Trigger**0xffc4**

This register is used for triggering the watchdog timer from the host system. Triggering the watchdog timer is only possible with a special access code, generated by a pseudo random generator. The following sequence is required to trigger the watchdog timer:

- 1.) read the DPM_HOST_WDG_HOST_TRIG register to get the next WDG_TRIGGER_CODE
- 2.) write back the new watchdog access code to the DPM_HOST_WDG_HOST_TRIG register

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
reserved																WDG_TRIGGER_CODE															

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	WDG_TRIGGER_CODE	Watchdog trigger code (read-write back-value)	R/W	0x00

DPM_HOST_WDG_HOST_TIMEOUT – DPM Host Side Watchdog Host Timeout**0xffc0**

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
reserved																TIMEOUT_VAL															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	TIMEOUT_VAL	Timeout value	R	0x00

Handshake Registers**DPM Host Side Handshake Register Pair (Host→netX and netX→Host)**

The location of each handshake register pair within the DPM area is programmed from ARM side and can be located at any dword address between area 0x0000 and 0xffdc. A total of 16 handshake register pairs are available.

16 Bit Handshake Register (Host→netX / netX→Host)**Area: 0x0000 - 0xffdc**

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
Host→NETX_DATA [15:0]																NETX→Host_DATA [15:0]															

Bits	Name	Description	R/W	Default
31:16	HOST→NETX_DATA[15:0]	Handshake Data Flags host to netX [15:0]	R/W	0x00
15:0	NETX→HOST_DATA[15:0]	Handshake Data Flags netX to host [15:0]	R	0x00

8 Bit Handshake Register (Host→netX / netX→Host)**Area: 0x0000 - 0xffdc**

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
HOST→NETX_DATA [7:0]								NETX→HOST_DATA [7:0]								DATA_MEMORY[15:8]								DATA_MEMORY[7:0]							

Bits	Name	Description	R/W	Default
31:24	HOST→NETX_DATA[7:0]	Handshake Data Flags host to netX [7:0]	R/W	0x00
23:16	NETX→HOST_DATA[7:0]	Handshake Data Flags netX to host [7:0]	R	0x00
15:8	DATA_MEMORY[15:8]	Data Memory [15:8]	R/W	0x00
7:0	DATA_MEMORY[7:0]	Data Memory [7:0]	R/W	0x00

5.2 DPM_ARM – Dual-Port Memory ARM Side

The following table is a summary of the Dual-Port Memory registers.

ARM Address	Register Name	Short Description
0x1c003ffc - 0x1c003800	reserved	-
0x1c0037fc - 0x1c003700	reserved	-
0x1c0036fc - 0x1c0036c0	Reserved	-
0x1c0036bc	DPM_ARM_HS_CTRL15	Handshake Control Register 15
0x1c0036b8	DPM_ARM_HS_CTRL14	Handshake Control Register 14
0x1c0036b4	DPM_ARM_HS_CTRL13	Handshake Control Register 13
0x1c0036b0	DPM_ARM_HS_CTRL12	Handshake Control Register 12
0x1c0036ac	DPM_ARM_HS_CTRL11	Handshake Control Register 11
0x1c0036a8	DPM_ARM_HS_CTRL10	Handshake Control Register 10
0x1c0036a4	DPM_ARM_HS_CTRL9	Handshake Control Register 9
0x1c0036a0	DPM_ARM_HS_CTRL8	Handshake Control Register 8
0x1c00369c	DPM_ARM_HS_CTRL3	Handshake Control Register 3
0x1c003698	DPM_ARM_HS_CTRL6	Handshake Control Register 6
0x1c003694	DPM_ARM_HS_CTRL5	Handshake Control Register 5
0x1c003690	DPM_ARM_HS_CTRL4	Handshake Control Register 4
0x1c00368c	DPM_ARM_HS_CTRL3	Handshake Control Register 3
0x1c003688	DPM_ARM_HS_CTRL2	Handshake Control Register 2
0x1c003684	DPM_ARM_HS_CTRL1	Handshake Control Register 1
0x1c003680	DPM_ARM_HS_CTRL0	Handshake Control Register 0
0x1c00367c	DPM_ARM_DB_MAP7	Data Block Mapping Address Register 7
0x1c003678	DPM_ARM_DB_END7	Data Block 7 End Address
0x1c003674	DPM_ARM_DB_MAP6	Data Block Mapping Address Register 6
0x1c003670	DPM_ARM_DB_END6	Data Block 6 End Address
0x1c00366c	DPM_ARM_DB_MAP5	Data Block Mapping Address Register 5
0x1c003668	DPM_ARM_DB_END5	Data Block 5 End Address
0x1c003664	DPM_ARM_DB_MAP4	Data Block Mapping Address Register 4
0x1c003660	DPM_ARM_DB_END4	Data Block 4 End Address
0x1c00365c	DPM_ARM_DB_MAP3	Data Block Mapping Address Register 3
0x1c003658	DPM_ARM_DB_END3	Data Block 3 End Address
0x1c003654	DPM_ARM_DB_MAP2	Data Block Mapping Address Register 2
0x1c003650	DPM_ARM_DB_END2	Data Block 2 End Address
0x1c00364c	DPM_ARM_DB_MAP1	Data Block Mapping Address Register 1
0x1c003648	DPM_ARM_DB_END1	Data Block 1 End Address
0x1c003644	DPM_ARM_DB_MAP0	Data Block Mapping Address Register 0
0x1c003640	DPM_ARM_DB_END0	Data Block 0 End Address
0x1c00363c	reserved	-
0x1c003638	DPM_ARM_IO_DATA1	Input / Output Data Register 1
0x1c003634	DPM_ARM_IO_DRV_EN1	Input / Output Driver Enable Register 1
0x1c003630	DPM_ARM_IO_MODE1	Input / Output Mode Register 1

0x1c00362c	Reserved	-
0x1c003628	DPM_ARM_IO_DATA0	Input / Output Data Register 0
0x1c003624	DPM_ARM_IO_DRV_EN0	Input / Output Driver Enable Register 0
0x1c003620	DPM_ARM_IO_MODE0	Input / Output Mode Register 0
0x1c00361c	EXT_CFG_CS3	Extension Bus Configuration Chip Select 3
0x1c003618	EXT_CFG_CS2	Extension Bus Configuration Chip Select 2
0x1c003614	EXT_CFG_CS1	Extension Bus Configuration Chip Select 1
0x1c003610	EXT_CFG_CS0	Extension Bus Configuration Chip Select 0
0x1c00360c	DPM_ARM_IF_CFG1	Interface Configuration Register 1
0x1c003608	DPM_ARM_IF_CFG0	Interface Configuration Register 0
0x1c003604	DPM_ARM_CLKOUT_CFG	Clockout Configuration Register
0x1c003600	Reserved	-
0x1c0035fh - 0x1c003540	Reserved	-
0x1c00353c	DPM_ARM_HS_DATA15	Handshake Data Register 15
0x1c003538	DPM_ARM_HS_DATA14	Handshake Data Register 14
0x1c003534	DPM_ARM_HS_DATA13	Handshake Data Register 13
0x1c003530	DPM_ARM_HS_DATA12	Handshake Data Register 12
0x1c00352c	DPM_ARM_HS_DATA11	Handshake Data Register 11
0x1c003528	DPM_ARM_HS_DATA10	Handshake Data Register 10
0x1c003524	DPM_ARM_HS_DATA9	Handshake Data Register 9
0x1c003520	DPM_ARM_HS_DATA8	Handshake Data Register 8
0x1c00351c	DPM_ARM_HS_DATA7	Handshake Data Register 7
0x1c003518	DPM_ARM_HS_DATA6	Handshake Data Register 6
0x1c003514	DPM_ARM_HS_DATA5	Handshake Data Register 5
0x1c003510	DPM_ARM_HS_DATA4	Handshake Data Register 4
0x1c00350c	DPM_ARM_HS_DATA3	Handshake Data Register 3
0x1c003508	DPM_ARM_HS_DATA2	Handshake Data Register 2
0x1c003504	DPM_ARM_HS_DATA1	Handshake Data Register 1
0x1c003500	DPM_ARM_HS_DATA0	Handshake Data Register 0
0x1c0034fc	Reserved	-
0x1c0034f8	Reserved	-
0x1c0034f4	Reserved	-
0x1c0034f0	DPM_ARM_INT_EN0	Interrupt Enable 0 Register
0x1c0034ec	Reserved	-
0x1c0034e8	Reserved	-
0x1c0034e4	Reserved	-
0x1c0034e0	DPM_ARM_INT_STAT0	Interrupt Status 0 Register
0x1c0034dc	Reserved	-
0x1c0034d8	DPM_ARM_SYS_STAT	System Status Register
0x1c0034d4	Reserved	-
0x1c0034d0	Reserved	-
0x1c0034cc	DPM_ARM_WDG_ARM_TRIG	Watchdog Trigger ARM
0x1c0034c8	DPM_ARM_WDG_ARM_TIMEOUT	Watchdog Timeout ARM, read only

0x1c0034c4	Reserved	-
0x1c0034c0	DPM_ARM_WDG_HOST_TIMEOUT	Watchdog Timeout Host
0x1c0034bc	DPM_ARM_CIS_MAP	Card Information Structure Mapping Address
0x1c0034b8 - 0x1c003400	Reserved	-
0x1c0033fc - 0x1c003300	Reserved	-
0x1c0032fc - 0x1c003240	Reserved	-
0x1c00323c	DPM_HOST_HS_DATA15	Handshake Data Register 15
0x1c003238	DPM_HOST_HS_DATA14	Handshake Data Register 14
0x1c003234	DPM_HOST_HS_DATA13	Handshake Data Register 13
0x1c003230	DPM_HOST_HS_DATA12	Handshake Data Register 12
0x1c00322c	DPM_HOST_HS_DATA11	Handshake Data Register 11
0x1c003228	DPM_HOST_HS_DATA10	Handshake Data Register 10
0x1c003224	DPM_HOST_HS_DATA9	Handshake Data Register 9
0x1c003220	DPM_HOST_HS_DATA8	Handshake Data Register 8
0x1c00321c	DPM_HOST_HS_DATA7	Handshake Data Register 7
0x1c003218	DPM_HOST_HS_DATA6	Handshake Data Register 6
0x1c003214	DPM_HOST_HS_DATA5	Handshake Data Register 5
0x1c003210	DPM_HOST_HS_DATA4	Handshake Data Register 4
0x1c00320c	DPM_HOST_HS_DATA3	Handshake Data Register 3
0x1c003208	DPM_HOST_HS_DATA2	Handshake Data Register 2
0x1c003204	DPM_HOST_HS_DATA1	Handshake Data Register 1
0x1c003200	DPM_HOST_HS_DATA0	Handshake Data Register 0
0x1c0031fc	Reserved	-
0x1c0031f8	Reserved	-
0x1c0031f4	Reserved	-
0x1c0031f0	DPM_HOST_INT_EN0	Interrupt Enable 0
0x1c0031ec	Reserved	-
0x1c0031e8	Reserved	-
0x1c0031e4	Reserved	-
0x1c0031e0	DPM_HOST_INT_STAT0	Interrupt Status 0
0x1c0031dc	DPM_HOST_RESET_REQ	Reset Request
0x1c0031d8	DPM_HOST_SYS_STAT	System Status
0x1c0031d4	DPM_HOST_TMR_START_VAL	Timer Start Value
0x1c0031d0	DPM_HOST_TMR_CTRL	Timer Control
0x1c0031cc	Reserved	-
0x1c0031c8	DPM_HOST_WDG_ARM_TIMEOUT	Watchdog ARM Timeout
0x1c0031c4	DPM_HOST_WDG_HOST_TRIG	Watchdog Host Trigger
0x1c0031c0	DPM_HOST_WDG_HOST_TIMEOUT	Watchdog Host Timeout, read only
0x1c0031bc - 0x1c003100	reserved	-
0x1c0030fc - 0x1c003000	reserved	-

Note:

Register with grey high lighted names will be accessed from host system. They are mapped into the netX memory range. Access from ARM side is only for debugging.

DPM_ARM_HS_CTRL0 – DPM ARM Side Handshake Control Register 0	0x1c003680
DPM_ARM_HS_CTRL1 – DPM ARM Side Handshake Control Register 1	0x1c003684
DPM_ARM_HS_CTRL2 – DPM ARM Side Handshake Control Register 2	0x1c003688
DPM_ARM_HS_CTRL3 – DPM ARM Side Handshake Control Register 3	0x1c00368c
DPM_ARM_HS_CTRL4 – DPM ARM Side Handshake Control Register 4	0x1c003690
DPM_ARM_HS_CTRL5 – DPM ARM Side Handshake Control Register 5	0x1c003694
DPM_ARM_HS_CTRL6 – DPM ARM Side Handshake Control Register 6	0x1c003698
DPM_ARM_HS_CTRL7 – DPM ARM Side Handshake Control Register 7	0x1c00369c
DPM_ARM_HS_CTRL8 – DPM ARM Side Handshake Control Register 8	0x1c0036a0
DPM_ARM_HS_CTRL9 – DPM ARM Side Handshake Control Register 9	0x1c0036a4
DPM_ARM_HS_CTRL10 – DPM ARM Side Handshake Control Register 10	0x1c0036a8
DPM_ARM_HS_CTRL11 – DPM ARM Side Handshake Control Register 11	0x1c0036ac
DPM_ARM_HS_CTRL12 – DPM ARM Side Handshake Control Register 12	0x1c0036b0
DPM_ARM_HS_CTRL13 – DPM ARM Side Handshake Control Register 13	0x1c0036b4
DPM_ARM_HS_CTRL14 – DPM ARM Side Handshake Control Register 14	0x1c0036b8
DPM_ARM_HS_CTRL15 – DPM ARM Side Handshake Control Register 15	0x1c0036bc

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
ENABLE		reserved														DPM_BASE_ADDR										reserved		8/16_BIT			

Bits	Name	Description	R/W	Default
31	ENABLE	When set the handshake register pair function is enabled.	R/W	0
30:16	reserved	-	R	0x00
15:2	DPM_BASE_ADDR	Base address of handshake register pair in Dual-Port memory.	R/W	0x00
1	reserved	-	R	0
0	8/16_BIT	Select the handshake register pair width 0 : 8 bit cell 1 : 16 bit cell	R/W	0

DPM_ARM_DB_END0	– DPM ARM Side Data Block 0 End Address	0x1c003640
DPM_ARM_DB_END1	– DPM ARM Side Data Block 1 End Address	0x1c003648
DPM_ARM_DB_END2	– DPM ARM Side Data Block 2 End Address	0x1c003650
DPM_ARM_DB_END3	– DPM ARM Side Data Block 3 End Address	0x1c003658
DPM_ARM_DB_END4	– DPM ARM Side Data Block 4 End Address	0x1c003660
DPM_ARM_DB_END5	– DPM ARM Side Data Block 5 End Address	0x1c003668
DPM_ARM_DB_END6	– DPM ARM Side Data Block 6 End Address	0x1c003670
DPM_ARM_DB_END7	– DPM ARM Side Data Block 7 End Address	0x1c003678

	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
ENABLE	reserved																DATA_BLOCK_END								reserved							

Bits	Name	Description	R/W	Default
31	ENABLE	Data Block Enable. When set the block mapping function is enabled.	R/W	0
30:16	reserved	-	R	0x00
15:8	DATA_BLOCK_END	Data Block End Address	R/W	0x00
7:0	reserved	-	R	0x00

These registers set the end address of each data block in the Dual-Port memory. It is possible to set each end address anywhere in the 64KB linear address range. The first memory block will always start at address 0x0000.

Data Block Number	Priority	Start Base Address	:	End Base Address
DATA_BLOCK0	highest	0x0000	:	DATA_BLOCK_END0 - 0x0001
DATA_BLOCK1		DATA_BLOCK_END0	:	DATA_BLOCK_END1 - 0x0001
DATA_BLOCK2		DATA_BLOCK_END1	:	DATA_BLOCK_END2 - 0x0001
DATA_BLOCK3		DATA_BLOCK_END2	:	DATA_BLOCK_END3 - 0x0001
DATA_BLOCK4	:	DATA_BLOCK_END3	:	DATA_BLOCK_END4 - 0x0001
DATA_BLOCK5	:	DATA_BLOCK_END4	:	DATA_BLOCK_END5 - 0x0001
DATA_BLOCK6	:	DATA_BLOCK_END5	:	DATA_BLOCK_END6 - 0x0001
DATA_BLOCK7	lowest	DATA_BLOCK_END6	:	DATA_BLOCK_END7 - 0x0001

DPM_ARM_DB_MAP0	– DPM ARM Side Data Block 0 Address Mapping	0x1c003644
DPM_ARM_DB_MAP1	– DPM ARM Side Data Block 1 Address Mapping	0x1c00364c
DPM_ARM_DB_MAP2	– DPM ARM Side Data Block 2 Address Mapping	0x1c003654
DPM_ARM_DB_MAP3	– DPM ARM Side Data Block 3 Address Mapping	0x1c00365c
DPM_ARM_DB_MAP4	– DPM ARM Side Data Block 4 Address Mapping	0x1c003664
DPM_ARM_DB_MAP5	– DPM ARM Side Data Block 5 Address Mapping	0x1c00366c
DPM_ARM_DB_MAP6	– DPM ARM Side Data Block 6 Address Mapping	0x1c003674
DPM_ARM_DB_MAP7	– DPM ARM Side Data Block 7 Address Mapping	0x1c00367c

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
NETX_MAP_ADDR																								reserved							

Bits	Name	Description	R/W	Default
31:8	NETX_MAP_ADDR	Data block base address in netX memory range	R/W	0x00
7:0	reserved	-	R	0x00

These registers set the physical base addresses in the netX address range of each data block enabled in the Dual-Port memory.

Data Block	netX Physical Mapping Address
DATA_BLOCK0	Physical_netX_Address0 = DATA_BLOCK_0_NETX_MAP_ADDR + 0x00000000
DATA_BLOCK1	Physical_netX_Address1 = DATA_BLOCK_1_NETX_MAP_ADDR + DATA_BLOCK_END0
DATA_BLOCK2	Physical_netX_Address2 = DATA_BLOCK_2_NETX_MAP_ADDR + DATA_BLOCK_END1
DATA_BLOCK3	Physical_netX_Address3 = DATA_BLOCK_3_NETX_MAP_ADDR + DATA_BLOCK_END2
DATA_BLOCK4	Physical_netX_Address4 = DATA_BLOCK_4_NETX_MAP_ADDR + DATA_BLOCK_END3
DATA_BLOCK5	Physical_netX_Address5 = DATA_BLOCK_5_NETX_MAP_ADDR + DATA_BLOCK_END4
DATA_BLOCK6	Physical_netX_Address6 = DATA_BLOCK_6_NETX_MAP_ADDR + DATA_BLOCK_END5
DATA_BLOCK7	Physical_netX_Address7 = DATA_BLOCK_7_NETX_MAP_ADDR + DATA_BLOCK_END6

Data Memory Area / Data Memory Blocks

The complete data memory area can be divided up to eight data memory blocks for data transfer between the netX and the host system. They are located at the address range 0x0000 until 0xFDFF. The start address in the Dual-port memory of each data block is programmable by the netX. Accesses to these data memory blocks are physically mapped to accesses in the netX memory range. This could be an internal memory area, an internal register or an external memory. The internal netX memory base mapping addresses are also programmable by the netX.

Due to the programmable memory block structure there is no special address base for accessing the data memory blocks at netX side.

The 'DPMAS_DB_END0-7' registers, set the end addresses of each data memory block in the Dual-Port memory. The area of each block is defined from the previous end pointer until the next end pointer minus one. The first memory block will always start at DPM address 0. All start addresses must be a multiple of 256 bytes. The end addresses must be ascending from memory block end register 0 to 7.

DPM handshake register always have a higher priority than the data memory areas. If a programmed data memory block address range overlaps with a handshake register pair, the DPM address where the handshake register pair has been mapped to, will always be redirected to the handshake register, which means that the corresponding memory location in the data memory area can not be accessed from the host side. This also applies to the global control block.

The 'DPMAS_DB_MAPPING0-7' registers controlled by the netX set the physical memory mapping address of each data memory block. The physical address must also be a multiple of 256 Bytes. The register value for the physical base address of data memory blocks 0 to 7 is calculated by the following formula:

$$\text{DB_MAPPING}_{(x)} = \text{NETX_ADDRESS}_{(x)} - \text{DB_END}_{(x-1)}$$

$$x = 0 \dots 7, \text{DB_END}_{(-1)} = 0000_0000\text{h.}$$

The following table shows an example for a programmed Dual-port memory structure with eight data memory blocks. The first four data areas are mapped to netX memory address 0x80000000 and following (external memory). The following four data memory blocks are mapped into internal memory address 0x18000 and following.

Block Number	DPM Address	DB_END	Block Size	netX Address	DB_MAPPING
0	0x0000 - 0x1FFF	0x2000	0x2000	0x80000000	0x80000000
1	0x2000 - 0x3FFF	0x4000	0x2000	0x80020000	0x80000000
2	0x4000 - 0x4FFF	0x5000	0x1000	0x80040000	0x80000000
3	0x5000 - 0x7FFF	0x8000	0x3000	0x80050000	0x80000000
4	0x8000 - 0x9FFF	0xA000	0x2000	0x00018000	0x00010000
5	0xA000 - 0xAFFF	0xB000	0x1000	0x0001A000	0x00010000
6	0xB000 - 0xBFFF	0xC000	0x1000	0x0001B000	0x00010000
7	0xC000 - 0xFDFF	0xFE00	0x3E00	0x0001C000	0x00010000

The register settings for this example are as follows:

Register	Value	Register	Value
DPM_ARM_DB_END0	0x80002000	DPM_ARM_DB_MAP0	0x80000000
DPM_ARM_DB_END1	0x80004000	DPM_ARM_DB_MAP1	0x80000000
DPM_ARM_DB_END2	0x80005000	DPM_ARM_DB_MAP2	0x80000000
DPM_ARM_DB_END3	0x80008000	DPM_ARM_DB_MAP3	0x80000000
DPM_ARM_DB_END4	0x8000A000	DPM_ARM_DB_MAP4	0x00010000
DPM_ARM_DB_END5	0x8000B000	DPM_ARM_DB_MAP5	0x00010000
DPM_ARM_DB_END6	0x8000C000	DPM_ARM_DB_MAP6	0x00010000
DPM_ARM_DB_END7	0x8000FE00	DPM_ARM_DB_MAP7	0x00010000

DPM_ARM_IO_DATA1 – DPM ARM Side Input / Output Data 1**0x1c003638**

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
reserved											PIO_DATA[85:64]																				

Bits	Name	Description	R/W	Default
31:22	reserved	-	R	0x00
21:0	PIO_DATA[85:64]	Input or Output Data of each I/O Pin Read: 0 : Physical Input Level is 0 1 : Physical Input Level is 1 Write: 0 : Sets the output pin level to 0 when configured as output 1 : Sets the output pin level to 1 when configured as output.	R/W	0x00

DPM_ARM_IO_DRV_EN1 – DPM ARM Side Input / Output Driver Enable 1**0x1c003634**

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
reserved											PIO_DRV[85:64]																				

Bits	Name	Description	R/W	Default
31:22	reserved	-	R	0x00
21:0	PIO_DRV[85:64]	Driver output enable of each I/O Pin 0 : Output driver disabled 1 : Output driver enabled	R/W	0x00

DPM_ARM_IO_MODE1 – DPM ARM Side Input / Output Mode 1**0x1c003630**

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
IN_CONTROL		reserved										PIO_MODE[85:64]																			

Bits	Name	Description	R/W	Default
31:30	IN_CONTROL	Input data control 00 : Input data latched by nPOR 01 : Input data always sampled with 100 MHz system clock Input 10 : data only sampled when PIO[77] pin is low. 11 : Input data only sampled when PIO[77] pin is high.	R/W	0x0
29:22	reserved	-	R	0x00
21:0	PIO_MODE[85:64]	Pin mode selection 0 : PIO Mode 1 : Host interface Mode	R/W	0x00

DPM_ARM_IO_DATA0 – DPM ARM Side Input / Output Data 0**0x1c003628**

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
PIO_DATA[63:32]																															

Bits	Name	Description	R/W	Default
31:0	PIO_DATA[63:32]	Input or Output Data of each I/O Pin Read: 0 : Physical Input Level is 0 1 : Physical Input Level is 1 Write: 0 : Sets the output pin level to 0 when configured as output 1 : Sets the output pin level to 1 when configured as output	R/W	0x00

DPM_ARM_IO_DRV_EN0 – DPM ARM Side Input / Output Driver Enable 0**0x1c003624**

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
PIO_DRV[63:32]																															

Bits	Name	Description	R/W	Default
31:0	PIO_DRV[63:32]	Driver output enable of each I/O Pin 0 : Output driver disabled 1 : Output driver enabled	R/W	0x00

DPM_ARM_IO_MODE0 – DPM ARM Side Input / Output Mode 0**0x1c003620**

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
PIO_MODE[63:32]																															

Bits	Name	Description	R/W	Default
31:0	PIO_MODE[63:32]	Pin mode selection 0 : PIO Mode 1 : Host interface mode	R/W	0x00

DPM_ARM_IF_CFG1 – DPM ARM Side Interface Configuration Register 1**0x1c00360c**

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
WRITE_PROTECT	IRQ_POL_PIO72	IRQ_POL_PIO47	IRQ_POL_PIO40	IRQ_POL_PIO36	IRQ_POL_PIO35	DATAOUT_VALID_TIME	DIS_BUSY_TIMEOUT	reserved	ADR_IN[15:12]	CS_COMP_SRC[19:12]	CS_COMP_VAL[19:12]																				

Bits	Name	Description	R/W	Default
31	WRITE_PROTECT	When this bit is set any following write accesses to the register are impossible. The bit will be only cleared by a netX system reset.	R/W	0
30	IRQ_POL_PIO72	External interrupt polarity for PIO[72]	R/W	0
29	IRQ_POL_PIO47	External interrupt polarity for PIO[47]	R/W	0
28	IRQ_POL_PIO40	External interrupt polarity for PIO[40]	R/W	0
27	IRQ_POL_PIO36	External interrupt polarity for PIO[36]	R/W	0
26	IRQ_POL_PIO35	External interrupt polarity for PIO[35]	R/W	0
25:24	DATAOUT_VALID_TIME	This value sets the delay between valid read data available at outputs and de-assertion of the BUSY- / assertion of the READY signal in system clocks (10 ns @ 100 MHz) 00 : 0 system clock 01 : 1 system clock 10 : 2 system clock 11 : 3 system clock	R/W	0x00
23	DIS_BUSY_TIMEOUT	When this flag is not set, any access to the netX host interface will be aborted after 256 system clocks (2.56µs @ 100 MHz) and the BUSY /READY signal will be released. This feature avoids system lockups when a read is not possible (e.g. access to SDRAM while SDRAM controller has not been configured and enabled).	R/W	0
22:20	reserved	-	R	0x00
19:16	ADR_IN[15:12]	Logic level for Dual-Port memory address lines [15:12] when used as input / output pin (pio mode) or when the chip select comparator is enabled.	R/W	0x00
15:8	CS_COMP_SRC[19:12]	Chip select compare source for address [19:12] 0 : internal (compare with CS_COM_VAL) 1 : external (compare with signals at pins SEL_A[19:12])	R/W	0x00
7:0	CS_COMP_VAL[19:12]	Chip Select compare value when internal compare source is used	R/W	0x00

DPM_ARM_IF_CFG0 – DPM ARM Side Host Interface Configuration Register 0**0x1c003608**

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
DISABLE_WR		HIF_MODE		RD_CTRL		WR_CTRL		BE1_MODE		BE0_MODE		CIS_MODE		WAIT_DRV		WAIT_MODE	WAIT_POLARITY			CS_MODE		IRQ_MODE		IRQ_POLARITY		ALE_MODE		ADDR_MODE		OE_MODE	

Bits	Name	Description	R/W	Default
31	DISABLE_WR	When this bit is set any consecutive write accesses to the register are blocked. This bit will only be cleared by a netX system reset.	R/W	0
30:28	HIF_MODE	Host interface mode selection. This selection is only relevant when the 'IF_SELECT_N' bit is not set in the 'IO Configuration Register' IO_CFG. 000 : Disabled outputs (all host interface pins in high impeded. mode) 001 : Extension Bus 010 : μ P Bus 8 bit (Dual-Port memory) 011 : μ P Bus 16 bit (Dual-Port memory) 100 : I/O Mode 101 : μ P Bus 32 bit (Dual-Port memory) 110 : Reserved (all host interface pins in high impedance mode) 111 : Reserved (all host interface pins in high impedance mode)	R/W	0x00
27:26	RD_CTRL	These bits configure the control lines for Dual-Port memory read accesses to netX 00 : RDn; PIO[52] A read cycle is selected when the RDn line has a low level (in 32 Bit mode also the Byte Enable Signals BE[3:0] must be low) 01 : RDn, A0, BHEn; PIO[52,73,43] A read cycle is selected when the RDn (RD/WRn = DIR) line has a high level and the A0 (BE0n) or BHEn (BE1n) line (or both of them) have a low level (e.g. Motorola 68000) 10 : Read access disabled 11 : RDn, BHEn; PIO[52,43] A read cycle is selected when the RDn (RD/WRn = DIR) line has a high level and the BHEn (EN) line has a high level (e.g. Motorola 6800).	R/W	0x00
25:24	WR_CTRL	These bits configure the control lines for Dual-Port memory write accesses to netX 00 : WRLn; PIO[45] A write cycle is selected when the WRLn (WRn) line has a low level (in 32 Bit mode also the Byte Enable Signals BE[3:0] must be low) 01 : RDn, A0, BHEn; PIO[52,73,43] A write cycle is selected when the RDn (RD/WRn = DIR) line has low level and the A0 (BE0n) or BHEn (BE1n) line or both of them have low level (e.g. Motorola 68000). 10 : WRLn, WRHn; PIO[45,44]	R/W	0x00

		<p>A write cycle is selected when the WRLn (low byte) or the WRHn (high byte) line (or both of them) have a low level.</p> <p>11 : RDn, BHEn; PIO[52,43] A write cycle is selected when the RDn (RD/WRn = DIR) line has a low level and the BHEn (EN) line has a high level (e.g. Motorola 6800).</p>		
23:21	BE1_MODE	<p>Byte Enable 1 (internal signal) generation :</p> <p>000 : BHEn; PIO[43] The high byte is selected when the BHEn (= CE2n) signal has a low level (PCMCIA Mode)</p> <p>001 : A0; PIO[73] The high byte is selected when the A0 line has a high level (e.g. 8 bit Dual-Port memory interface).</p> <p>010 : RDn, WRLn; PIO[52,45] The high byte is selected when the RDn or WRLn lines have low level (only one write signal, only 16 Bit accesses)</p> <p>011 : RDn, WRHn; PIO[52,44] High byte is selected when the RDn or WRHn lines have low level.</p> <p>100 : internal A0 line controlled via PIO Mode or ALE Mode (A0==1)</p> <p>101 : nBHE; PIO[43] The high byte is selected when the BHEn (SBHE) line has a high level (ISA Mode)</p> <p>101 : high byte access always active</p> <p>11x : high byte access always active</p>	R/W	0x00
20:18	BE0_MODE	<p>Byte Enable 0 (internal signal) generation :</p> <p>000 : CS0n, PIO[51] The low byte is selected when the CS0n (= CE1n) signal has a low level (PCMCIA Mode)</p> <p>001 : A0; PIO[73] The low byte is selected when the A0 line has a low level (e.g. 8 bit Dual-Port memory interface).</p> <p>010 : RDn, WRLn; PIO[52,45] The low byte is selected when the RDn or WRLn lines have a low level.</p> <p>011 : internal A0 line controlled via PIO Mode or ALE Mode (A0==0)</p> <p>1xx : low byte access always active</p>	R/W	0x00
17:16	CIS_MODE	<p>The CIS memory array is selected by the following condition:</p> <p>00 : Never</p> <p>01 : Always</p> <p>10 : by low WRHn / PIO[44] signal == REGn (PCMCIA Mode)</p> <p>11 : by low PIO[40] signal</p>	R/W	0x00
15:14	WAIT_DRV	<p>Wait mode output drive control; RDY / PIO[46]</p> <p>00 : high impedance output</p> <p>01 : push / pull output</p> <p>10 : open drain / open source output</p> <p>11 : sustained tri state output</p>	R/W	0x00
13	WAIT_MODE	<p>0 : WAIT / BUSY mode function. An active signal indicates that the</p>	R/W	0

		current access can not yet be finished by the host CPU. 1 : READY mode function. An active signal indicates that the host CPU may finish the current access.		
12	WAIT_POLARITY	0 : low active polarity output 1 : high active polarity output	R/W	0
11:9	CS_MODE	These bits configure the logic for internal chip select generation 000 : chip access is disabled 001 : chip select generated by internal address comparator 010 : CS0n, BHEn; PIO[51,43] The netX DPM is selected when CS0n or BHEn lines have a low level (PCMCIA Mode) 011 : CS0n, BHEn; PIO[51,43] The netX DPM is selected when CS0n or BHEn lines have a high level 100 : CS0n, PIO[51] The netX DPM is selected when the CS0n line has a low level. 101 : chip select disabled 110 : Internal Address Comparator and ALE; PIO[35] The netX chip is selected when the internal address comparator detects a hit and the ALE (= AEN) has a low level 111 : Internal Address Comparator and ALE; PIO[35] The netX chip is selected when the internal address comparator detects a hit and the ALE (= AEN) has a high level	R/W	0x00
8:7	IRQ_MODE	Select the interrupt pin output function of IRQ / INT pin (PIO[47]) 00 : high impedance output 01 : fixed output level 10 : push pull output 11 : open drain / open source output	R/W	0x00
6	IRQ_POLARITY	0 : active low polarity output 1 : active high polarity output	R/W	0
5:4	ALE_MODE	Selection of address input mode (PIO[35] == ALE/AEN). This function is only activated when no input / output mode is selected for the PIO[35] pin. 00 : address latching on low signal level of PIO[35] 01 : address latching on high signal level of PIO[35] 10 : address latching on falling PIO[35] edge 11 : address latching on rising PIO[35] edge	R/W	0x00
3	ADDR_MODE	0 : non multiplexed 8/16 bit address mode 1 : multiplexed 8/16 bit address mode	R/W	0
2:0	OE_MODE	Selection bits for output driver control of data lines for read accesses 000 : RDn, BHEn, A0; PIO[52,43,73] Output drivers are enabled by a high RDn (RD/WRn) signal and a low BHEn (high byte) or a low A0 (low byte) signal. (e.g. Motorola 68000) 001 : RDn, BHEn, A0; PIO[52,43,73]	R/W	0x00

		Output drivers are enabled by a low RDn (RDn) signal and a low BHEn (high byte enable) or a low A0 (low byte enable) signal (e.g. 80186)		
	010 :	RDn; PIO[52] Output drivers are enabled by a low RDn signal (8 and 16 Bit mode).		
	011 :	RDn, BHEn, CS0n; PIO[52,43,51] The output drivers are enabled by a low RDn signal and a low CS0n (CE0n) or a low BHEn (CE1n) signal (PCMCIA Mode).		
	100 :	RDn, BHEn; PIO[52,43] The output drivers are enabled when the RDn (RD/WRn = DIR) signal is high and the BHEn (EN) signal is high (e.g. Motorola 6800)		
	101 :	reserved (output drivers disabled)		
	110 :	RDn , BE[3:0] (Intel 32 Bit mode)		
	111 :	RD/WRn, BE[3:0] (Motorola 32 Bit mode)		

The following table shows some configuration examples.

Connection Type	DPM_ARM_IF_CFG0
Intel, 8 Bit non multiplexed, one write signal	x_010_00_00_???_???_??_??_?_???_??_?_??_?_???
Intel, 8 Bit multiplexed, one write signal	x_010_00_00_???_???_??_??_?_???_??_?_??_?_???
Intel, 16 Bit non-multiplexed, one write signal	x_011_00_00_???_???_??_??_?_???_??_?_??_?_???
Intel, 16 Bit non-multiplexed, write low/high signal	x_011_00_00_???_???_??_??_?_???_??_?_??_?_???
Motorola, 8 Bit non multiplexed	x_010_00_00_???_???_??_??_?_???_??_?_??_?_???
Motorola Coldfire, 16 Bit non-multiplexed	x_011_00_00_???_???_??_??_?_???_??_?_??_?_???
Motorola MC68000	x_011_00_00_???_???_??_??_?_???_??_?_??_?_???

DPM_ARM_CLKOUT_CFG – DPM ARM Side Clock out configuration

0x1c003604

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
CLKOUT_EN	reserved	CLK_SEL																													

Bits	Name	Description	R/W	Default
31	CLKOUT_EN	CLKOUT driver enable.	R/W	0
30	reserved	-	R/W	0
29:0	CLK_SEL	Clock output frequency selection. $F = \text{Value} \times 200 \text{ MHz} / 2^{32}$	R/W	0x00

DPM_ARM_HS_DATA15 – DPM ARM View Handshake Register 15	0x1c00353c
DPM_ARM_HS_DATA14 – DPM ARM View Handshake Register 14	0x1c003538
DPM_ARM_HS_DATA13 – DPM ARM View Handshake Register 13	0x1c003534
DPM_ARM_HS_DATA12 – DPM ARM View Handshake Register 12	0x1c003530
DPM_ARM_HS_DATA11 – DPM ARM View Handshake Register 11	0x1c00352c
DPM_ARM_HS_DATA10 – DPM ARM View Handshake Register 10	0x1c003528
DPM_ARM_HS_DATA9 – DPM ARM View Handshake Register 9	0x1c003524
DPM_ARM_HS_DATA8 – DPM ARM View Handshake Register 8	0x1c003520
DPM_ARM_HS_DATA7 – DPM ARM View Handshake Register 7	0x1c00351c
DPM_ARM_HS_DATA6 – DPM ARM View Handshake Register 6	0x1c003518
DPM_ARM_HS_DATA5 – DPM ARM View Handshake Register 5	0x1c003514
DPM_ARM_HS_DATA4 – DPM ARM View Handshake Register 4	0x1c003510
DPM_ARM_HS_DATA3 – DPM ARM View Handshake Register 3	0x1c00350c
DPM_ARM_HS_DATA2 – DPM ARM View Handshake Register 2	0x1c003508
DPM_ARM_HS_DATA1 – DPM ARM View Handshake Register 1	0x1c003504
DPM_ARM_HS_DATA0 – DPM ARM View Handshake Register 0	0x1c003500

16 Bit Handshake Data

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
HOST→NETX_DATA [15:0]																NETX→HOST_DATA [15:0]															

Bits	Name	Description	R/W	Default
31:16	HOST→NETX_DATA[15:0]	Handshake Data Flags host to netX [15:0]	R	0x00
15:0	NETX→Host_DATA[15:0]	Handshake Data Flags netX to host [15:0]	R/W	0x00

8 Bit Handshake Data

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
HOST→NETX_DATA [7:0]								NETX→HOST_DATA [7:0]								DATA_MEMORY[15:8]								DATA_MEMORY [7:0]							

Bits	Name	Description	R/W	Default
31:24	HOST→NETX_DATA[7:0]	Handshake Data Flags host to netX [7:0]	R	0x00
23:16	NETX→HOST_DATA[7:0]	Handshake Data Flags netX to host [7:0]	R/W	0x00
15:8	DATA_MEMORY [15:8]	Data Memory [15:8]	R/W	0x00
7:0	DATA_MEMORY [7:0]	Data Memory [7:0]	R/W	0x00

DPM_ARM_INT_EN0 – DPM ARM Side Interrupt Enable 0**0x1c0034f0**

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
GLB_EN	MEM_LCK	WDG	INT_PIO72	INT_PIO47	INT_PIO40	INT_PIO36	INT_PIO35	reserved								HANDSHAKE_EVENT[15:0]															

Bits	Name	Description	R/W	Default
31	GLB_EN	Global Interrupt Enable	R/W	0
30	MEM_LCK	Memory Lock Error (generated by invalid access from host side)	R/W	0
29	WDG	Watchdog timeout host supervision interrupt enable	R/W	0
28	INT_PIO72	External interrupt enable for pin PIO72	R/W	0
27	INT_PIO47	External interrupt enable for pin PIO47	R/W	0
26	INT_PIO40	External interrupt enable for pin PIO40	R/W	0
25	INT_PIO36	External interrupt enable for pin PIO36	R/W	0
24	INT_PIO35	External interrupt enable for pin PIO35	R/W	0
23:16	reserved	-	R	0x00
15:0	HS[15:0]	Handshake event 15:0 interrupt enable	R/W	0x00

This register controls the interrupt behaviour of the host interface. The interrupt status flags will be always set on the event but the physical interrupt line will only be active when the enable bit is set. All interrupt re-requests can be enabled or disabled independently and are wired together to the global interrupt request for the host interface.

DPM_ARM_INT_STAT0 – DPM ARM Side Interrupt Status 0**0x1c0034e0**

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
IRQ_REQ	MEM_LCK	WDG_IRQ	INT_PIO72	INT_PIO47	INT_PIO40	INT_PIO36	INT_PIO35	IRQ_VECTOR[7:0]								HANDSHAKE_EVENT[15:0]															

Bits	Name	Description	R/W	Default
31	IRQ_REQ	Global signaling of interrupt request	R	0
30	MEM_LCK	Flag of Memory Lock Error	R/W	0
29	WDG_IRQ	Flag of Watchdog timeout host supervision	R/W	0
28	INT_PIO72	Flag of External interrupt status for pin PIO72	R/W	0
27	INT_PIO47	Flag of External interrupt status for pin PIO47	R/W	0
26	INT_PIO40	Flag of External interrupt status for pin PIO40	R/W	0
25	INT_PIO36	Flag of External interrupt status for pin PIO36	R/W	0
24	INT_PIO35	Flag of External interrupt status for pin PIO35	R/W	0
23:16	IRQ_VECTOR[7:0]	Interrupt Vector generated by the interrupt status flags 0x00 : no interrupt 0x10 : handshake event interrupt 0 0x11 : handshake event interrupt 1 0x12 : handshake event interrupt 2 0x13 : handshake event interrupt 3 0x14 : handshake event interrupt 4 0x15 : handshake event interrupt 5 0x16 : handshake event interrupt 6 0x17 : handshake event interrupt 7 0x18 : handshake event interrupt 8 0x19 : handshake event interrupt 9 0x1a : handshake event interrupt 10 0x1b : handshake event interrupt 11 0x1c : handshake event interrupt 12 0x1d : handshake event interrupt 13 0x1e : handshake event interrupt 14 0x1f : handshake event interrupt 15 0x20 - 0x5f : reserved 0x60 : Memory Lock Error 0x61 : Watchdog timeout host supervision 0x62 : PIO72 interrupt request 0x63 : PIO47 interrupt request 0x64 : PIO40 interrupt request 0x65 : PIO36 interrupt request 0x66 : PIO35 interrupt request 0x67 - 0xff : reserved	R	0x00
15:0	HS[15:0]	Flag of Handshake Event Status [15:0]	R/W	0x00

This register does not only show the interrupt status flags of each interrupt event, but also the interrupt vector. The flags are always set when the interrupt event occurs regardless of the interrupt enable flag, but the physical interrupt which is indicated by the interrupt vector will only be generated when the corresponding enable flag is set. The interrupt status flags can be cleared by writing a one bit to the flag. When the handshake register cell is read, the corresponding interrupt status flags will be cleared too. It makes no difference if an 8 bit or 16 bit access occurs. The interrupt flag will not be cleared when the interrupt event reoccurs while the cell is being read.

DPM_ARM_SYS_STAT – System Status**0x1c0034d8**

This register is just an alias of the register 'SYS_STAT' described in chapter 2.7.

DPM_ARM_WDG_ARM_TRIG – DPM ARM Side Watchdog ARM Supervision Trigger 0x1c0034cc

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
reserved																								WDG_ACCESS_CODE							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	WDG_ACCESS_CODE	Watchdog access code for triggering (read-write back-value)	R/W	0x00

DPM_ARM_WDG_ARM_TIMEOUT – DPM ARM Side Watchdog ARM Timeout**0x1c0034c8**

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
reserved																TIMEOUT_VAL															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	TIMEOUT_VAL	Timeout value $T_{\text{TIMEOUT}} = \text{TIMEOUT_VAL} \times 100 \mu\text{s}$	R	0x00

DPM_ARM_WDG_HOST_TIMEOUT – DPM ARM Side Watchdog Host Timeout**0x1c0034c0**

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
reserved																TIMEOUT_VAL															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x00
15:0	TIMEOUT_VAL	Timeout value $T_{\text{TIMEOUT}} = \text{TIMEOUT_VAL} \times 100 \mu\text{s}$	R/W	0x00

DPM_ARM_CIS_MAP – DPM ARM Side CIS Mapping Address**0x1c0034bc**

In Dual-port memory mode it is possible to allocate a 256 byte data field which contains an image of the card information structure CIS. The netX has to enable this function and has to store the data information via software into the data field. To access the CIS from host side it is possible to program the CIS-Mode in the 'DPM_ARM_IF_CFG0 – DPM ARM Side Host Interface Configuration Register 0'. Different access types are possible. For CIS accesses only the lower address lines DPM_ADR0-7 are used.

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
CIS_MAPPING																								reserved				WR_ENABLE	CIS_ENABLE		

Bits	Name	Description	R/W	Default
31:8	CIS_MAPPING	CIS Mapping Address. This value sets the mapping address for CIS accesses in the netX memory range. The data length is fixed to 256 byte.	R/W	0x00
7:2	reserved	-	R	0x00
1	WR_ENABLE	If this bit is set write accesses to the CIS are enabled. otherwise write accesses are ignored.	R/W	0
0	CIS_ENABLE	Enable CIS MODE	R/W	0

6 Peripheral Functions

6.1 GPIOs – General Purpose IOs and Timers

The neX50 provides 32 general purpose IOs (GPIOs) and 5 timers. The following table shows a summary of all GPIO- and Timer related registers:

ARM Address	Register Name	Short Description
0x1c000800	GPIO_CFG0	GPIO 0 Configuration Register
0x1c000804	GPIO_CFG1	GPIO 1 Configuration Register
0x1c000808	GPIO_CFG2	GPIO 2 Configuration Register
0x1c00080c	GPIO_CFG3	GPIO 3 Configuration Register
0x1c000810	GPIO_CFG4	GPIO 4 Configuration Register
0x1c000814	GPIO_CFG5	GPIO 5 Configuration Register
0x1c000818	GPIO_CFG6	GPIO 6 Configuration Register
0x1c00081c	GPIO_CFG7	GPIO 7 Configuration Register
0x1c000820	GPIO_CFG8	GPIO 8 Configuration Register
0x1c000824	GPIO_CFG9	GPIO 9 Configuration Register
0x1c000828	GPIO_CFG10	GPIO 10 Configuration Register
0x1c00082c	GPIO_CFG11	GPIO 11 Configuration Register
0x1c000830	GPIO_CFG12	GPIO 12 Configuration Register
0x1c000834	GPIO_CFG13	GPIO 13 Configuration Register
0x1c000838	GPIO_CFG14	GPIO 14 Configuration Register
0x1c00083c	GPIO_CFG15	GPIO 15 Configuration Register
0x1c000840	GPIO_CFG16	GPIO 16 Configuration Register
0x1c000844	GPIO_CFG17	GPIO 17 Configuration Register
0x1c000848	GPIO_CFG18	GPIO 18 Configuration Register
0x1c00084c	GPIO_CFG19	GPIO 19 Configuration Register
0x1c000850	GPIO_CFG20	GPIO 20 Configuration Register
0x1c000854	GPIO_CFG21	GPIO 21 Configuration Register
0x1c000858	GPIO_CFG22	GPIO 22 Configuration Register
0x1c00085c	GPIO_CFG23	GPIO 23 Configuration Register
0x1c000860	GPIO_CFG24	GPIO 24 Configuration Register
0x1c000864	GPIO_CFG25	GPIO 25 Configuration Register
0x1c000868	GPIO_CFG26	GPIO 26 Configuration Register
0x1c00086c	GPIO_CFG27	GPIO 27 Configuration Register
0x1c000870	GPIO_CFG28	GPIO 28 Configuration Register
0x1c000874	GPIO_CFG29	GPIO 29 Configuration Register
0x1c000878	GPIO_CFG30	GPIO 30 Configuration Register
0x1c00087c	GPIO_CFG31	GPIO 31 Configuration Register
0x1c000880	GPIO_THRSH_CAPT0	GPIO 0 Threshold or Capture Register
0x1c000884	GPIO_THRSH_CAPT1	GPIO 1 Threshold or Capture Register
0x1c000888	GPIO_THRSH_CAPT2	GPIO 2 Threshold or Capture Register
0x1c00088c	GPIO_THRSH_CAPT3	GPIO 3 Threshold or Capture Register

0x1c000890	GPIO_THRSH_CAPT4	GPIO 4 Threshold or Capture Register
0x1c000894	GPIO_THRSH_CAPT5	GPIO 5 Threshold or Capture Register
0x1c000898	GPIO_THRSH_CAPT6	GPIO 6 Threshold or Capture Register
0x1c00089c	GPIO_THRSH_CAPT7	GPIO 7 Threshold or Capture Register
0x1c0008a0	GPIO_THRSH_CAPT8	GPIO 8 Threshold or Capture Register
0x1c0008a4	GPIO_THRSH_CAPT9	GPIO 9 Threshold or Capture Register
0x1c0008a8	GPIO_THRSH_CAPT10	GPIO 10 Threshold or Capture Register
0x1c0008ac	GPIO_THRSH_CAPT11	GPIO 11 Threshold or Capture Register
0x1c0008b0	GPIO_THRSH_CAPT12	GPIO 12 Threshold or Capture Register
0x1c0008b4	GPIO_THRSH_CAPT13	GPIO 13 Threshold or Capture Register
0x1c0008b8	GPIO_THRSH_CAPT14	GPIO 14 Threshold or Capture Register
0x1c0008bc	GPIO_THRSH_CAPT15	GPIO 15 Threshold or Capture Register
0x1c0008c0	GPIO_THRSH_CAPT16	GPIO 16 Threshold or Capture Register
0x1c0008c4	GPIO_THRSH_CAPT17	GPIO 17 Threshold or Capture Register
0x1c0008c8	GPIO_THRSH_CAPT18	GPIO 18 Threshold or Capture Register
0x1c0008cc	GPIO_THRSH_CAPT19	GPIO 19 Threshold or Capture Register
0x1c0008d0	GPIO_THRSH_CAPT20	GPIO 20 Threshold or Capture Register
0x1c0008d4	GPIO_THRSH_CAPT21	GPIO 21 Threshold or Capture Register
0x1c0008d8	GPIO_THRSH_CAPT22	GPIO 22 Threshold or Capture Register
0x1c0008dc	GPIO_THRSH_CAPT23	GPIO 23 Threshold or Capture Register
0x1c0008e0	GPIO_THRSH_CAPT24	GPIO 24 Threshold or Capture Register
0x1c0008e4	GPIO_THRSH_CAPT25	GPIO 25 Threshold or Capture Register
0x1c0008e8	GPIO_THRSH_CAPT26	GPIO 26 Threshold or Capture Register
0x1c0008ec	GPIO_THRSH_CAPT27	GPIO 27 Threshold or Capture Register
0x1c0008f0	GPIO_THRSH_CAPT28	GPIO 28 Threshold or Capture Register
0x1c0008f4	GPIO_THRSH_CAPT29	GPIO 29 Threshold or Capture Register
0x1c0008f8	GPIO_THRSH_CAPT30	GPIO 30 Threshold or Capture Register
0x1c0008fc	GPIO_THRSH_CAPT31	GPIO 31 Threshold or Capture Register
0x1c000900	GPIO_CNTR0_CTRL	GPIO counter 0 control register
0x1c000904	GPIO_CNTR1_CTRL	GPIO counter 1 control register
0x1c000908	GPIO_CNTR2_CTRL	GPIO counter 2 control register
0x1c00090c	GPIO_CNTR3_CTRL	GPIO counter 3 control register
0x1c000910	GPIO_CNTR4_CTRL	GPIO counter 4 control register
0x1c000914	GPIO_CNTR0_MAX	GPIO counter 0 max values
0x1c000918	GPIO_CNTR1_MAX	GPIO counter 1 max values
0x1c00091c	GPIO_CNTR2_MAX	GPIO counter 2 max values
0x1c000920	GPIO_CNTR3_MAX	GPIO counter 3 max values
0x1c000924	GPIO_CNTR4_MAX	GPIO counter 4 max values
0x1c000928	GPIO_CNTR0_CNT	GPIO counter 0 current value
0x1c00092c	GPIO_CNTR1_CNT	GPIO counter 1 current value
0x1c000930	GPIO_CNTR2_CNT	GPIO counter 2 current value
0x1c000934	GPIO_CNTR3_CNT	GPIO counter 3 current value
0x1c000938	GPIO_CNTR4_CNT	GPIO counter 4 current value
0x1c00093c	GPIO_SYSTIME_NS_CMP	GPIO System Time NS Compare Value
0x1c000940	GPIO_OUT	GPIO Output Register
0x1c000944	GPIO_IN	GPIO Input Register

0x1c000948	GPIO_IRQ_RAW	GPIO raw IRQ register
0x1c00094c	GPIO_IRQ_MSK	GPIO Masked IRQ register
0x1c000950	GPIO_IRQ_MSK_SET	GPIO Interrupt Request Mask Enable
0x1c000954	GPIO_IRQ_MSK_RESET	GPIO Interrupt Request Mask Disable
0x1c000958	CNTR_IRQ_RAW	Counter raw IRQ register
0x1c00095c	CNTR_IRQ_MSK	Counter Masked IRQ register
0x1c000960	CNTR_IRQ_MSK_SET	Counter Interrupt Request Mask Enable
0x1c000964	CNTR_IRQ_MSK_RESET	Counter Interrupt Request Mask Disable

GPIOs have the following features:

- Each GPIO can be configured individually as input or output, inverted or non inverted.
- Each GPIO can be assigned to one of the Timers or the System Time in order to be used as capture input or PWM output.
- Each GPIO can generate an interrupt, when it is configured in one of the capture modes.
- Four consecutive GPIOs (0-3, 4-7, etc.) can be configured as IO-Link channel

Each GPIO has its own configuration register GPIO_CFGi, which can be used to read or write the corresponding IO configuration individually. All inputs can be read in the GPIO_IN register, respectively can be written in the GPIO_OUT register.

If a GPIO is to generate an Interrupt then the corresponding interrupt request bit must be set in the GPIO_IRQ_MSK_SET register. To disable any GPIO IRQ, the corresponding bit must be set in the GPIO_IRQ_MSK_RESET register. When a GPIO generates an interrupt, then the corresponding bit in register GPIO_IRQ_RAW will be automatically set.

The five internal timers, realized by 32-Bit Counters, can be configured to:

- count from zero to a maximum value and backward (symmetric Mode)
- count from zero to a maximum value and set back to zero (asymmetric Mode)
- single shot or count continuously
- generate an interrupt if it reaches zero
- count external events
- set back to zero by an external event
- capture the timer value by an external event
- generate a PWM signal by comparing the timer value to a threshold

Any GPIO can be assigned as external event, which can be a rising or falling edge, or a high or low level at the GPIO by setting the inverting bit at the GPIO configuration register. The counter value can be read and overwritten at any time.

GPIO_CFG0 – GPIO 0 Configuration Register	0x1c000800
GPIO_CFG1 – GPIO 1 Configuration Register	0x1c000804
GPIO_CFG2 – GPIO 2 Configuration Register	0x1c000808
GPIO_CFG3 – GPIO 3 Configuration Register	0x1c00080c
GPIO_CFG4 – GPIO 4 Configuration Register	0x1c000810
GPIO_CFG5 – GPIO 5 Configuration Register	0x1c000814
GPIO_CFG6 – GPIO 6 Configuration Register	0x1c000818
GPIO_CFG7 – GPIO 7 Configuration Register	0x1c00081c
GPIO_CFG8 – GPIO 8 Configuration Register	0x1c000820
GPIO_CFG9 – GPIO 9 Configuration Register	0x1c000824
GPIO_CFG10 – GPIO 10 Configuration Register	0x1c000828
GPIO_CFG11 – GPIO 11 Configuration Register	0x1c00082c
GPIO_CFG12 – GPIO 12 Configuration Register	0x1c000830
GPIO_CFG13 – GPIO 13 Configuration Register	0x1c000834
GPIO_CFG14 – GPIO 14 Configuration Register	0x1c000838
GPIO_CFG15 – GPIO 15 Configuration Register	0x1c00083c
GPIO_CFG16 – GPIO 16 Configuration Register	0x1c000840
GPIO_CFG17 – GPIO 17 Configuration Register	0x1c000844
GPIO_CFG18 – GPIO 18 Configuration Register	0x1c000848
GPIO_CFG19 – GPIO 19 Configuration Register	0x1c00084c
GPIO_CFG20 – GPIO 20 Configuration Register	0x1c000850
GPIO_CFG21 – GPIO 21 Configuration Register	0x1c000854
GPIO_CFG22 – GPIO 22 Configuration Register	0x1c000858
GPIO_CFG23 – GPIO 23 Configuration Register	0x1c00085c
GPIO_CFG24 – GPIO 24 Configuration Register	0x1c000860
GPIO_CFG25 – GPIO 25 Configuration Register	0x1c000864
GPIO_CFG26 – GPIO 26 Configuration Register	0x1c000868
GPIO_CFG27 – GPIO 27 Configuration Register	0x1c00086c
GPIO_CFG28 – GPIO 28 Configuration Register	0x1c000870
GPIO_CFG29 – GPIO 29 Configuration Register	0x1c000874
GPIO_CFG30 – GPIO 30 Configuration Register	0x1c000878
GPIO_CFG31 – GPIO 31 Configuration Register	0x1c00087c

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
reserved																								COUNT_REF	INV	MODE					

Bits	Name	Description	R/W	Default
31:8	reserved		R	0x00
7:5	COUNT_REF	counter reference 000 : counter 0 001 : counter 1 010 : counter 2 011 : counter 3 100 : counter 4 111 : system time	R/W	0x00
4	INV	0 : don't invert, 1 : invert input/output value	R/W	0
3:0	MODE	defines the GPIO input or output mode - depends on iocfg Input mode: 0000: read mode 0001: capture continued at rising edge (allows gpio_irq on each capture) 0010: capture once at rising edge (reset gpio_irq to capture again) 0011: capture once at high level (reset gpio_irq to capture again)	R/W	0x00

		Output mode: 0100: set to 0 0101: set to 1 0110: set to gpio out[0] 0111: pwm mode, direct threshold update Multi pin modes: 1000: IO-Link mode (always uses 4 sequential GPIOs, e.g. [0123], [4567],...) 1111: pwm2-mode with threshold update at counter=0		
--	--	--	--	--

GPIO_THRSH_CAPT0 – GPIO 0 Threshold or Capture Register	0x1c000880
GPIO_THRSH_CAPT1 – GPIO 1 Threshold or Capture Register	0x1c000884
GPIO_THRSH_CAPT2 – GPIO 2 Threshold or Capture Register	0x1c000888
GPIO_THRSH_CAPT3 – GPIO 3 Threshold or Capture Register	0x1c00088c
GPIO_THRSH_CAPT4 – GPIO 4 Threshold or Capture Register	0x1c000890
GPIO_THRSH_CAPT5 – GPIO 5 Threshold or Capture Register	0x1c000894
GPIO_THRSH_CAPT6 – GPIO 6 Threshold or Capture Register	0x1c000898
GPIO_THRSH_CAPT7 – GPIO 7 Threshold or Capture Register	0x1c00089c
GPIO_THRSH_CAPT8 – GPIO 8 Threshold or Capture Register	0x1c0008a0
GPIO_THRSH_CAPT9 – GPIO 9 Threshold or Capture Register	0x1c0008a4
GPIO_THRSH_CAPT10 – GPIO 10 Threshold or Capture Register	0x1c0008a8
GPIO_THRSH_CAPT11 – GPIO 11 Threshold or Capture Register	0x1c0008ac
GPIO_THRSH_CAPT12 – GPIO 12 Threshold or Capture Register	0x1c0008b0
GPIO_THRSH_CAPT13 – GPIO 13 Threshold or Capture Register	0x1c0008b4
GPIO_THRSH_CAPT14 – GPIO 14 Threshold or Capture Register	0x1c0008b8
GPIO_THRSH_CAPT15 – GPIO 15 Threshold or Capture Register	0x1c0008bc
GPIO_THRSH_CAPT16 – GPIO 16 Threshold or Capture Register	0x1c0008c0
GPIO_THRSH_CAPT17 – GPIO 17 Threshold or Capture Register	0x1c0008c4
GPIO_THRSH_CAPT18 – GPIO 18 Threshold or Capture Register	0x1c0008c8
GPIO_THRSH_CAPT19 – GPIO 19 Threshold or Capture Register	0x1c0008cc
GPIO_THRSH_CAPT20 – GPIO 20 Threshold or Capture Register	0x1c0008d0
GPIO_THRSH_CAPT21 – GPIO 21 Threshold or Capture Register	0x1c0008d4
GPIO_THRSH_CAPT22 – GPIO 22 Threshold or Capture Register	0x1c0008d8
GPIO_THRSH_CAPT23 – GPIO 23 Threshold or Capture Register	0x1c0008dc
GPIO_THRSH_CAPT24 – GPIO 24 Threshold or Capture Register	0x1c0008e0
GPIO_THRSH_CAPT25 – GPIO 25 Threshold or Capture Register	0x1c0008e4
GPIO_THRSH_CAPT26 – GPIO 26 Threshold or Capture Register	0x1c0008e8
GPIO_THRSH_CAPT27 – GPIO 27 Threshold or Capture Register	0x1c0008ec
GPIO_THRSH_CAPT28 – GPIO 28 Threshold or Capture Register	0x1c0008f0
GPIO_THRSH_CAPT29 – GPIO 29 Threshold or Capture Register	0x1c0008f4
GPIO_THRSH_CAPT30 – GPIO 30 Threshold or Capture Register	0x1c0008f8
GPIO_THRSH_CAPT31 – GPIO 31 Threshold or Capture Register	0x1c0008fc

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
VAL																															

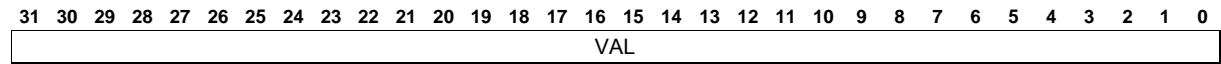
Bits	Name	Description	R/W	Default
31:0	VAL	<p>Threshold/Capture register:</p> <p>PWM mode (threshold):</p> <p>The counter threshold value equals the number of inactive clockcycles per period (cycles with pwm=0). Therefore it is interpreted different in symmetrical and asymmetrical counter mode:</p> <p>Asymmetrical mode (sawtooth): $pwm = (counter \geq gpio_thrsh_capt)$</p> <p>Symmetrical mode (triangle) : counter is compared with $gpio_thrsh_capt[31:1]$, $gpio_thrsh_capt[0]$ prolongs the inactive phase by 1 cc only while upcounting. This allows running 10ns resolution even in symmetrical mode.</p> <p>Capture mode (capture register)</p> <p>In capture mode this register holds the captured counter value.</p>	R/W	0x00

GPIO_CNTR0_CTRL – GPIO Counter 0 Control	0x1c000900
GPIO_CNTR1_CTRL – GPIO Counter 1 Control	0x1c000904
GPIO_CNTR2_CTRL – GPIO Counter 2 Control	0x1c000908
GPIO_CNTR3_CTRL – GPIO Counter 3 Control	0x1c00090c
GPIO_CNTR4_CTRL – GPIO Counter 4 Control	0x1c000910

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
reserved																				GPIO_REF				EVENT_ACT	ONCE	SEL_EVENT	IRQ_EN	SYM_NASYM	RUN		

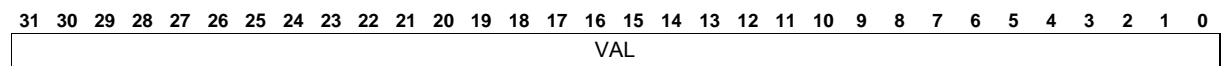
Bits	Name	Description	R/W	Default
31:12	reserved	-	R	0x00
11:7	GPIO_REF	gpio reference (0 - 31)	R/W	0x00
6:5	EVENT_ACT	Define action of selected external event (dependant on sel_event, gpio_ref) 00: count every clock cycle, ignore external events 01: count only on external event (edge or level according to sel_event bit) 10: enable watchdog mode of counter (external event resets without IRQ, overflow generates IRQ). 11: enable automatic run by external event (set run bit at external event, used for DC-DC PWM in once mode)	R/W	0
4	ONCE	1: count once (reset run bit after 1 period) 0: count continue	R/W	0
3	SEL_EVENT	select external event 0: high level, invert gpio in gpio_cfg register to select low level 1: pos. edge, invert gpio in gpio_cfg register to select neg. edge	R/W	0
2	IRQ_EN	1 : enable interrupt request 0 : disable interrupt request	R/W	0
1	SYM_NASYM	1 : symmetric mode (triangle) 0 : asymmetric mode (sawtooth)	R/W	0
0	RUN	1 : start counter 0 : stop counter	R/W	0

GPIO_CNTR0_MAX – GPIO Counter 0 Maximum Value	0x1c000914
GPIO_CNTR1_MAX – GPIO Counter 1 Maximum Value	0x1c000918
GPIO_CNTR2_MAX – GPIO Counter 2 Maximum Value	0x1c00091c
GPIO_CNTR3_MAX – GPIO Counter 3 Maximum Value	0x1c000920
GPIO_CNTR4_MAX – GPIO Counter 4 Maximum Value	0x1c000924



Bits	Name	Description	R/W	Default
31:0	VAL	Asymmetric mode: counting period in cc + 1 Symmetric mode: counting period in cc	R/W	0x00

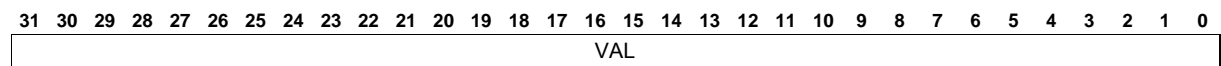
GPIO_CNTR0_CNT – GPIO Counter 0 Current Value	0x1c000928
GPIO_CNTR1_CNT – GPIO Counter 1 Current Value	0x1c00092c
GPIO_CNTR2_CNT – GPIO Counter 2 Current Value	0x1c000930
GPIO_CNTR3_CNT – GPIO Counter 3 Current Value	0x1c000934
GPIO_CNTR4_CNT – GPIO Counter 4 Current Value	0x1c000938



Bits	Name	Description	R/W	Default
31:0	VAL	current counter value	R/W	0x00

GPIO_SYSTIME_NS_CMP – GPIO System Time NS Compare Value	0x1c00093c
--	-------------------

Compares this value with systime_ns considering incontinous behaviour of systime_ns



Bits	Name	Description	R/W	Default
31:0	VAL	compare value for systime	R/W	0x00

GPIO_OUT– GPIO Output Register**0x1c000940**

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
VAL																															

Bits	Name	Description	R/W	Default
31:0	VAL	GPIO[31:0] output values	R/W	0x00

GPIO_IN– GPIO Input Register**0x1c000944**

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
VAL																															

Bits	Name	Description	R/W	Default
31:0	VAL	GPIO[31:0] input values	R	0x00

GPIO_IRQ_RAW – GPIO raw IRQ Register**0x1c000948**

The GPIO_IRQ_RAW register holds the interrupt bits for every GPIO. Write access with '1' resets the appropriate IRQ. Write access with '0' does not influence this bit.

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
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Bits	Name	Description	R/W	Default
31	GPIO31	interrupt bit for GPIO31 or IO-Link7-wakeup_interrupt	R/W	0
30	GPIO30	interrupt bit for GPIO30 or IO-Link7-rx_interrupt	R/W	0
29	GPIO29	interrupt bit for GPIO29 or IO-Link7-tx_interrupt	R/W	0
28	GPIO28	interrupt bit for GPIO28 or IO-Link7-cycle timer interrupt	R/W	0
27	GPIO27	interrupt bit for GPIO27 or IO-Link6-wakeup_interrupt	R/W	0
26	GPIO26	interrupt bit for GPIO26 or IO-Link6-rx_interrupt	R/W	0
25	GPIO25	interrupt bit for GPIO25 or IO-Link6-tx_interrupt	R/W	0
24	GPIO24	interrupt bit for GPIO24 or IO-Link6-cycle timer interrupt	R/W	0
23	GPIO23	interrupt bit for GPIO23 or IO-Link5-wakeup_interrupt	R/W	0
22	GPIO22	interrupt bit for GPIO22 or IO-Link5-rx_interrupt	R/W	0
21	GPIO21	interrupt bit for GPIO21 or IO-Link5-tx_interrupt	R/W	0
20	GPIO20	interrupt bit for GPIO20 or IO-Link5-cycle timer interrupt	R/W	0
19	GPIO19	interrupt bit for GPIO19 or IO-Link4-wakeup_interrupt	R/W	0
18	GPIO18	interrupt bit for GPIO18 or IO-Link4-rx_interrupt	R/W	0
17	GPIO17	interrupt bit for GPIO17 or IO-Link4-tx_interrupt	R/W	0
16	GPIO16	interrupt bit for GPIO16 or IO-Link4-cycle timer interrupt	R/W	0
15	GPIO15	interrupt bit for GPIO15 or IO-Link3-wakeup_interrupt	R/W	0
14	GPIO14	interrupt bit for GPIO14 or IO-Link3-rx_interrupt	R/W	0
13	GPIO13	interrupt bit for GPIO13 or IO-Link3-tx_interrupt	R/W	0
12	GPIO12	interrupt bit for GPIO12 or IO-Link3-cycle timer interrupt	R/W	0
11	GPIO11	interrupt bit for GPIO11 or IO-Link2-wakeup_interrupt	R/W	0
10	GPIO10	interrupt bit for GPIO10 or IO-Link2-rx_interrupt	R/W	0
9	GPIO9	interrupt bit for GPIO9 or IO-Link2-tx_interrupt	R/W	0
8	GPIO8	interrupt bit for GPIO8 or IO-Link2-cycle timer interrupt	R/W	0
7	GPIO7	interrupt bit for GPIO7 or IO-Link1-wakeup_interrupt	R/W	0
6	GPIO6	interrupt bit for GPIO6 or IO-Link1-rx_interrupt	R/W	0
5	GPIO5	interrupt bit for GPIO5 or IO-Link1-tx_interrupt	R/W	0
4	GPIO4	interrupt bit for GPIO4 or IO-Link1-cycle timer interrupt	R/W	0
3	GPIO3	interrupt bit for GPIO3 or IO-Link0-wakeup_interrupt	R/W	0
2	GPIO2	interrupt bit for GPIO2 or IO-Link0-rx_interrupt	R/W	0
1	GPIO1	interrupt bit for GPIO1 or IO-Link0-tx_interrupt	R/W	0
0	GPIO0	interrupt bit for GPIO0 or IO-Link0-cycle timer interrupt	R/W	0

GPIO_IRQ_MSK – GPIO Masked IRQ Register**0x1c00094c**

Read access shows status of masked IRQs (as connected to VIC/ARM).

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
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Bits	Name	Description	R/W	Default
31:0	GPIO[31:0]	One bit per GPIO	R	0

GPIO_IRQ_MSK_SET – GPIO Interrupt Enable**0x1c000950**

Write access with '1' sets interrupt mask bit (enables interrupt request for corresponding GPIO). Write access with '0' does not influence this bit. Read access shows actual interrupt mask.

Attention: Before activating interrupt mask, delete old pending interrupts by writing the same value to GPIO_IRQ_RAW.

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
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Bits	Name	Description	R/W	Default
31:0	GPIO[31:0]	One bit per GPIO	R/W	0

GPIO_IRQ_MSK_RESET – GPIO Interrupt Disable**0x1c000954**

Write access with '1' resets interrupt mask bit (disables interrupt request for corresponding GPIO). Write access with '0' does not influence this bit. Read access shows actual interrupt mask.

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
GPIO31	GPIO30	GPIO29	GPIO28	GPIO27	GPIO26	GPIO25	GPIO24	GPIO23	GPIO22	GPIO21	GPIO20	GPIO19	GPIO18	GPIO17	GPIO16	GPIO15	GPIO14	GPIO13	GPIO12	GPIO11	GPIO10	GPIO9	GPIO8	GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Bits	Name	Description	R/W	Default
31:0	GPIO[31:0]	One bit per GPIO	R/W	0

CNTR_IRQ_RAW – Counter raw IRQ register**0x1c000958**

Write access with '1' resets the appropriate IRQ. Write access with '0' does not influence this bit.
Read access shows status of unmasked IRQs

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
reserved																										SYSTIME	CNT4	CNT3	CNT2	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	SYSTIME	interrupt bit for sys_time (if sys_time = gpio_systime_cmp)	R/W	0
4	CNT4	interrupt bit for counter4	R/W	0
3	CNT3	interrupt bit for counter3	R/W	0
2	CNT2	interrupt bit for counter2	R/W	0
1	CNT1	interrupt bit for counter1	R/W	0
0	CNT0	interrupt bit for counter0	R/W	0

CNTR_IRQ_MSK – Counter masked IRQ register**0x1c00095c**

Read access shows status of masked IRQs (as connected to VIC/ARM)

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
reserved																										SYSTIME	CNT4	CNT3	CNT2	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	SYSTIME	interrupt bit for sys_time (if sys_time = gpio_systime_cmp)	R	0
4	CNT4	interrupt bit for counter4	R	0
3	CNT3	interrupt bit for counter3	R	0
2	CNT2	interrupt bit for counter2	R	0
1	CNT1	interrupt bit for counter1	R	0
0	CNT0	interrupt bit for counter0	R	0

CNTR_IRQ_MSK_SET – Counter Interrupt Request Mask Enable**0x1c000960**

Write access with '1' sets interrupt mask bit (enables interrupt request for corresponding counter). Write access with '0' does not influence this bit. Read access shows actual interrupt mask.

Attention: Before activating interrupt mask, delete old pending interrupts by writing the same value to CNTR_IRQ_RAW.

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
reserved																										SYSTIME	CNT4	CNT3	CNT2	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	SYSTIME	sys_time interrupt mask bit.	R/W	0
4	CNT4	Counter4 interrupt mask bit	R/W	0
3	CNT3	Counter3 interrupt mask bit	R/W	0
2	CNT2	Counter2 interrupt mask bit	R/W	0
1	CNT1	Counter1 interrupt mask bit	R/W	0
0	CNT0	Counter0 interrupt mask bit	R/W	0

CNTR_IRQ_MSK_RESET – Counter Interrupt Request Mask Disable**0x1c000964**

Write access with '1' resets interrupt mask bit (disables interrupt request for corresponding counter). Write access with '0' does not influence this bit. Read access shows actual interrupt mask.

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
reserved																										SYSTIME	CNT4	CNT3	CNT2	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	SYSTIME	sys_time interrupt mask bit.	R/W	0
4	CNT4	Counter4 interrupt mask bit	R/W	0
3	CNT3	Counter3 interrupt mask bit	R/W	0
2	CNT2	Counter2 interrupt mask bit	R/W	0
1	CNT1	Counter1 interrupt mask bit	R/W	0
0	CNT0	Counter0 interrupt mask bit	R/W	0

6.2 IO-Link

The netX provides 8 individual IO-Link controller modules in total, whereas. Each IO-Link interface uses 4 consecutive GPIO ports. To use a IO-Link module, the corresponding GPIO_CFG registers must be set to IO-Link mode.

The following table is a summary of registers of GPIOs and Timers.

ARM Address	Register Name	Short Description
0x1c000880	IOLINK0_CFG	IO-Link 0 Configuration Register
0x1c000884	IOLINK0_TX_FRM0	IO-Link 0 TX0 Data Register
0x1c000888	IOLINK0_TX_FRM1	IO-Link 0 TX1 Data Register
0x1c00088c	IOLINK0_RX_FRM	IO-Link 0 RX Data Register
0x1c000890	IOLINK1_CFG	IO-Link 1 Configuration Register
0x1c000894	IOLINK1_TX_FRM0	IO-Link 1 TX0 Data Register
0x1c000898	IOLINK1_TX_FRM1	IO-Link 1 TX1 Data Register
0x1c00089c	IOLINK1_RX_FRM	IO-Link 1 RX Data Register
0x1c0008a0	IOLINK2_CFG	IO-Link 2 Configuration Register
0x1c0008a4	IOLINK2_TX_FRM0	IO-Link 2 TX0 Data Register
0x1c0008a8	IOLINK2_TX_FRM1	IO-Link 2 TX1 Data Register
0x1c0008ac	IOLINK2_RX_FRM	IO-Link 2 RX Data Register
0x1c0008b0	IOLINK3_CFG	IO-Link 3 Configuration Register
0x1c0008b4	IOLINK3_TX_FRM0	IO-Link 3 TX0 Data Register
0x1c0008b8	IOLINK3_TX_FRM1	IO-Link 3 TX1 Data Register
0x1c0008bc	IOLINK3_RX_FRM	IO-Link 3 RX Data Register
0x1c0008c0	IOLINK4_CFG	IO-Link 4 Configuration Register
0x1c0008c4	IOLINK4_TX_FRM0	IO-Link 4 TX0 Data Register
0x1c0008c8	IOLINK4_TX_FRM1	IO-Link 4 TX1 Data Register
0x1c0008cc	IOLINK4_RX_FRM	IO-Link 4 RX Data Register
0x1c0008d0	IOLINK5_CFG	IO-Link 5 Configuration Register
0x1c0008d4	IOLINK5_TX_FRM0	IO-Link 5 TX0 Data Register
0x1c0008d8	IOLINK5_TX_FRM1	IO-Link 5 TX1 Data Register
0x1c0008dc	IOLINK5_RX_FRM	IO-Link 5 RX Data Register
0x1c0008e0	IOLINK6_CFG	IO-Link 6 Configuration Register
0x1c0008e4	IOLINK6_TX_FRM0	IO-Link 6 TX0 Data Register
0x1c0008e8	IOLINK6_TX_FRM1	IO-Link 6 TX1 Data Register
0x1c0008ec	IOLINK6_RX_FRM	IO-Link 6 RX Data Register
0x1c0008f0	IOLINK7_CFG	IO-Link 7 Configuration Register
0x1c0008f4	IOLINK7_TX_FRM0	IO-Link 7 TX0 Data Register
0x1c0008f8	IOLINK7_TX_FRM1	IO-Link 7 TX1 Data Register
0x1c0008fc	IOLINK7_RX_FRM	IO-Link 7 RX Data Register

15	reserved	-	R/W	0
14:13	NR_RX_OCT	Number for octets to receive: code rx octets per frame ----- 00 1 01 2 10 3 11 4	R/W	0
12:10	NR_TX_OCT	Number for octets to transmit code tx octets per frame ----- 000 1 001 2 010 3 011 4 100 5 101 reserved 110 reserved 111 reserved	R/W	0
9:8	FREQ_SEL	Set the baud rate (T_clk): 00: 250ns test only 01: 208,33 us, com1 - 4800 Baud 10: 26,04 us, com2 - 38400 Baud 11: 4,34 us, com3 - 230400 Baud	R/W	0
7:0	FRAME_CYCLE	Half of number of T_clk cycles, this compare value is doubled by the hardware!	R/W	0

IOLINK0_TX_FRM0 – IO-Link 0 TX0 Data Register	0x1c000884
IOLINK1_TX_FRM0 – IO-Link 1 TX0 Data Register	0x1c000894
IOLINK2_TX_FRM0 – IO-Link 2 TX0 Data Register	0x1c0008a4
IOLINK3_TX_FRM0 – IO-Link 3 TX0 Data Register	0x1c0008b4
IOLINK4_TX_FRM0 – IO-Link 4 TX0 Data Register	0x1c0008c4
IOLINK5_TX_FRM0 – IO-Link 5 TX0 Data Register	0x1c0008d4
IOLINK6_TX_FRM0 – IO-Link 6 TX0 Data Register	0x1c0008e4
IOLINK7_TX_FRM0 – IO-Link 7 TX0 Data Register	0x1c0008f4

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
TX_OCTET4								TX_OCTET3								TX_OCTET2								TX_OCTET1							

Bits	Name	Description	R/W	Default
31:24	TX_OCTET4	Transmit octet 4	R/W	0x00
23:16	TX_OCTET3	Transmit octet 3	R/W	0x00
15:8	TX_OCTET2	Transmit octet 2	R/W	0x00
7:0	TX_OCTET1	Transmit octet 1	R/W	0x00

IOLINK0_TX_FRM1	– IO-Link 0 TX1 Data Register	0x1c000888
IOLINK1_TX_FRM1	– IO-Link 1 TX1 Data Register	0x1c000898
IOLINK2_TX_FRM1	– IO-Link 2 TX1 Data Register	0x1c0008a8
IOLINK3_TX_FRM1	– IO-Link 3 TX1 Data Register	0x1c0008b8
IOLINK4_TX_FRM1	– IO-Link 4 TX1 Data Register	0x1c0008c8
IOLINK5_TX_FRM1	– IO-Link 5 TX1 Data Register	0x1c0008d8
IOLINK6_TX_FRM1	– IO-Link 6 TX1 Data Register	0x1c0008e8
IOLINK7_TX_FRM1	– IO-Link 7 TX1 Data Register	0x1c0008f8

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
TX_OCTET8								TX_OCTET7								TX_OCTET6								TX_OCTET5							

Bits	Name	Description	R/W	Default
31:24	TX_OCTET8	Transmit octet 8 , debug only! if en_delay_r bit set, this value is a tx delay value	R/W	0x00
23:16	TX_OCTET7	Transmit octet 7 , debug only!	R/W	0x00
15:8	TX_OCTET6	Transmit octet 6 , debug only!	R/W	0x00
7:0	TX_OCTET5	Transmit octet 5	R/W	0x00

IOLINK0_RX_FRM	– IO-Link 0 RX Data Register	0x1c00088c
IOLINK1_RX_FRM	– IO-Link 1 RX Data Register	0x1c00089c
IOLINK2_RX_FRM	– IO-Link 2 RX Data Register	0x1c0008ac
IOLINK3_RX_FRM	– IO-Link 3 RX Data Register	0x1c0008bc
IOLINK4_RX_FRM	– IO-Link 4 RX Data Register	0x1c0008cc
IOLINK5_RX_FRM	– IO-Link 5 RX Data Register	0x1c0008dc
IOLINK6_RX_FRM	– IO-Link 6 RX Data Register	0x1c0008ec
IOLINK7_RX_FRM	– IO-Link 7 RX Data Register	0x1c0008fc

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
RX_OCTET4								RX_OCTET3								RX_OCTET2								RX_OCTET1							

Bits	Name	Description	R/W	Default
31:24	RX_OCTET4	Receive octet 4, debug only!	R/W	0x00
23:16	RX_OCTET3	Receive octet 3	R/W	0x00
15:8	RX_OCTET2	Receive octet 2	R/W	0x00
7:0	RX_OCTET1	Receive octet 1	R/W	0x00

6.3 PIO – Programmable Input Output

Besides the Host interface PIOs, the netX50 provides 8 standard PIOs (PIO0 – 7), which are configured by the following three registers:

ARM Address	Register Name	Short Description
0x1c000a00	PIO_IN	PIO Input Register
0x1c000a04	PIO_OUT	PIO Output Register
0x1c000a08	PIO_OUT_EN	PIO Output Enable Register

PIO_IN – PIO Input Register

0x1c000a00

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					
reserved																								VAL												

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	VAL	PIO[7:0] input values	R	0x00

PIO_OUT – PIO Output Register

0x1c000a04

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					
reserved																								VAL												

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0
7:0	VAL	PIO[7:0] output values	R/W	0x00

PIO_OUT_EN – PIO Output Enable Register

0x1c000a08

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					
reserved																								VAL												

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0
7:0	VAL	PIO[7:0] output enables	R/W	0x00

6.4 UART – Universal Asynchronous Receiver Transmitter

The following table shows a summary of all registers of UART0, UART1 and UART2.

ARM Address	Register Name	Short Description
0x1c000b00	UART0_DATA	UART0 Data Register
0x1c000b04	UART0_STAT	UART0 Status Register
0x1c000b08	UART0_LINE_CTRL	UART0 Line Control Register
0x1c000b0c	UART0_BAUD_DIV_MSB	UART0 Baud Rate Divisor MSB
0x1c000b10	UART0_BAUD_DIV_LSB	UART0 Baud Rate Divisor LSB
0x1c000b14	UART0_CTRL	UART0 Control Register
0x1c000b18	UART0_FLAG	UART0 Flag Register
0x1c000b1c	UART0_INT_ID	UART0 Interrupt Identification Register
0x1c000b20	UART0_IRDA_LO_PWR_CNTR	UART0 IrDA Low Power Counter Register
0x1c000b24	UART0_RTS_CTRL	UART0 RTS Control Register
0x1c000b28	UART0_RTS_LEAD_CYC	UART0 RTS Leading Cycles
0x1c000b2c	UART0_RTS_TRAIL_CYC	UART0 RTS Trailing cycles
0x1c000b30	UART0_OUT_DRV_EN	UART0 UART Output Driver Enable Register
0x1c000b34	UART0_BAUD_MODE_CTRL	UART0 Baud Rate Mode Control Register
0x1c000b38	UART0_RX_FIFO_IRQ_LVL	UART0 RX FIFO Trigger Level and RX-DMA Enable
0x1c000b3c	UART0_TX_FIFO_IRQ_LVL	UART0 TX FIFO Trigger Level and TX-DMA Enable
0x1c000b40	UART1_DATA	UART1 Data Register
0x1c000b44	UART1_STAT	UART1 Status Register
0x1c000b48	UART1_LINE_CTRL	UART1 Line Control Register
0x1c000b4c	UART1_BAUD_DIV_MSB	UART1 Baud Rate Divisor MSB
0x1c000b50	UART1_BAUD_DIV_LSB	UART1 Baud Rate Divisor LSB
0x1c000b54	UART1_CTRL	UART1 Control Register
0x1c000b58	UART1_FLAG	UART1 Flag Register
0x1c000b5c	UART1_INT_ID	UART1 Interrupt Identification Register
0x1c000b60	UART1_IRDA_LO_PWR_CNTR	UART1 IrDA Low Power Counter Register
0x1c000b64	UART1_RTS_CTRL	UART1 RTS Control Register
0x1c000b68	UART1_RTS_LEAD_CYC	UART1 RTS Leading Cycles
0x1c000b6c	UART1_RTS_TRAIL_CYC	UART1 RTS Trailing cycles
0x1c000b70	UART1_OUT_DRV_EN	UART1 UART Output Driver Enable Register
0x1c000b74	UART1_BAUD_MODE_CTRL	UART1 Baud Rate Mode Control Register
0x1c000b78	UART1_RX_FIFO_IRQ_LVL	UART1 RX FIFO Trigger Level and RX-DMA Enable
0x1c000b7c	UART1_TX_FIFO_IRQ_LVL	UART1 TX FIFO Trigger Level and TX-DMA Enable
0x1c000b80	UART2_DATA	UART2 Data Register
0x1c000b84	UART2_STAT	UART2 Status Register
0x1c000b88	UART2_LINE_CTRL	UART2 Line Control Register
0x1c000b8c	UART2_BAUD_DIV_MSB	UART2 Baud Rate Divisor MSB
0x1c000b90	UART2_BAUD_DIV_LSB	UART2 Baud Rate Divisor LSB
0x1c000b94	UART2_CTRL	UART2 Control Register
0x1c000b98	UART2_FLAG	UART2 Flag Register

0x1c000b9c	UART2_INT_ID	UART2 Interrupt Identification Register
0x1c000ba0	UART2_IRDA_LO_PWR_CNTR	UART2 IrDA Low Power Counter Register
0x1c000ba4	UART2_RTS_CTRL	UART2 RTS Control Register
0x1c000ba8	UART2_RTS_LEAD_CYC	UART2 RTS Leading Cycles
0x1c000bac	UART2_RTS_TRAIL_CYC	UART2 RTS Trailing cycles
0x1c000bb0	UART2_OUT_DRV_EN	UART2 UART Output Driver Enable Register
0x1c000bb4	UART2_BAUD_MODE_CTRL	UART2 Baud Rate Mode Control Register
0x1c000bb8	UART2_RX_FIFO_IRQ_LVL	UART2 RX FIFO Trigger Level and RX-DMA Enable
0x1c000bbc	UART2_TX_FIFO_IRQ_LVL	UART2 TX FIFO Trigger Level and TX-DMA Enable

UART0_DATA – UART 0 Data Register
UART1_DATA – UART 1 Data Register
UART2_DATA – UART 2 Data Register

0x1c000b00
0x1c000b40
0x1c000b80

These registers are the send and receive registers for the UARTs.

For data to be transmitted:

- If the FIFOs are enabled, data written to this location is pushed onto the 16 byte deep transmit FIFO.
- If the FIFOs are not enabled, data is stored into the transmitter holding register.

The write operation initiates transmission from the UART. The data is prefixed with a start bit, appended with the appropriate parity bit (if parity is enabled), and a stop bit. The resultant word is then transmitted. Before you can send any data you have to enable the UARTi_TXD or UARTi_RTS driver (see UART_OUT_DRV_EN register).

For received data:

- If the FIFOs are enabled, the data byte is extracted and a 3-bit status (break, frame and parity) is pushed onto the 11-bit wide receive FIFO.
- If the FIFOs are not enabled, the data byte and status are stored in the receiving holding register (the bottom of the receive FIFO).

The received data must be read first from UART_DATA registers, followed by the status error associated with the data from UART_STAT registers. This read sequence cannot be reversed. The UART must be disabled before any of the control registers are reprogrammed.

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			
reserved																					BE	PE	FE	DATA										

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x00
10	BE	Break Error, read only, mirrored from uart_stat, to handle in DMA-read-out data	R	0
9	PE	Parity Error, read only, mirrored from uart_stat, to handle in DMA-read-out data	R	0
8	FE	Framing Error, read only, mirrored from uart_stat, to handle in DMA-read-out data	R	0
7:0	DATA	Receive data character by reading. Transmit data character by writing.	R/W	0x00

UART0_STAT – UART 0 Status Register**0x1c000b04****UART1_STAT – UART 1 Status Register****0x1c000b44****UART2_STAT – UART 2 Status Register****0x1c000b84**

Receive status is read from UART_STAT registers. The status information corresponds to the data character read from UART_DATA registers prior to reading UART_STAT registers. A write to UART_STAT registers clears the framing, parity, break and overrun errors. All the bits are cleared to 0 on reset.

The received data character must be read first from UART_DATA before reading the error status associated with that data character from UART_STAT. This read sequence cannot be reversed, since the status register UART_STAT is updated only when a read occurs from the data register UART_DATA.

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
reserved																												OE	BE	PE	FE

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x00
3	OE	Overrun Error The FIFO contents remain valid since no further data is written when the FIFO is full, only the contents of the shift register are overwritten. The ARM must now read the data in order to empty the FIFO. 0: after a write to UART_STAT register. 1: if data is received and the FIFO is already full.	R/W	0
2	BE	Break Error In FIFO mode, this error is associated with the character at the top of the FIFO. When a break occurs, only one 0 character is loaded into the FIFO. The next character is only enabled after the receive data input goes to 1 (marking state) and the next valid start bit is received. 0: after a write to UART_STAT register. 1: if a break condition was detected, indicating that the received data input was held LOW for longer than a full-word transmission time (defined as start, data, parity and stop bits).	R/W	0
1	PE	Parity Error In FIFO mode, this error is associated with the character at the top of the FIFO. 0: by a write to UART_STAT register. 1: it indicates that the parity of the received data character does not match the parity selected in UART_LINE_CTRL (bit 2).	R/W	0
0	FE	Framing Error In FIFO mode, this error is associated with the character at the top of the FIFO. 0: by a write to UART_STAT register. 1: it indicates that the received character did not have a valid stop bit (a valid stop bit is 1).	R/W	0

UART0_LINE_CTRL – UART 0 Line Control Register**0x1c000b08****UART1_LINE_CTRL – UART 1 Line Control Register****0x1c000b48****UART2_LINE_CTRL – UART 2 Line Control Register****0x1c000b88**

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
reserved																								WLEN	FEN	STP2	EPS	PEN	BRK		

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x00
6:5	WLEN	Word length. The bits indicate the number of data bits transmitted or received in a frame as follows: 00 : 5 bits 01 : 6 bits 10 : 7 bits 11 : 8 bits	R/W	00
4	FEN	Enable FIFOs. 0: the FIFOs are disabled (character mode) that is, the FIFOs become 1-byte-deep holding registers. 1: transmit and receive FIFO buffers are enabled (FIFO mode).	R/W	0
3	STP2	Two Stop Bits Select. The receive logic does not check for two stop bits being received. 1: two stop bits are transmitted at the end of the frame.	R/W	0
2	EPS	Even Parity Select. This bit has no effect when parity is disabled by Parity Enable (bit 1) being cleared to 0. 1: even parity generation and checking is performed during transmission and reception, which checks for an even number of 1 in data and parity bits. 0: odd parity is performed which checks for an odd number of 1.	R/W	0
1	PEN	Parity Enable. 0: parity is disabled and no parity bit added to the data frame. 1: parity checking and generation is enabled.	R/W	0
0	BRK	Send Break. 0: for normal use this bit must be cleared to 0. 1: a low level is continually output on the uart_txd output, after completing transmission of the current character. This bit must be asserted for at least one complete frame transmission time in order to generate a break condition. The transmit FIFO contents remain unaffected during a break condition.	R/W	0

UART Baud Rate Generation

Depending on the UART_BAUD_MODE_CTRL[0] bit the baud rate is calculated by the following formulas:

$$\text{BAUDDIV} = (100 \text{ MHz} / (16 \cdot \text{BaudRate})) - 1 \quad (\text{UART_BAUD_MODE_CTRL}[0] = 0)$$

$$\text{BAUDDIV} = ((\text{Baud Rate} \cdot 16) / 100 \text{ MHz}) \cdot 2^{16}. \quad (\text{UART_BAUD_MODE_CTRL}[0] = 1)$$

The maximum Baud rate is 3.125 MBaud.

UART0_BAUD_DIV_MSB – UART 0 Baud Rate Divisor MSB **0x1c000b0c**
UART1_BAUD_DIV_MSB – UART 1 Baud Rate Divisor MSB **0x1c000b4c**
UART2_BAUD_DIV_MSB – UART 2 Baud Rate Divisor MSB **0x1c000b8c**

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
reserved																								BAUDDIVMSB							

Bits	Name	Description	R/W	Default
31:8	Reserved	-	R	0x00
7:0	BAUDDIV_MSB	Baud Rate Divisor [15:8]. Most significant byte of baud rate divisor (BAUDDIV).	R/W	0x00

UART0_BAUD_DIV_LSB – UART 0 Baud Rate Divisor LSB **0x1c000b10**
UART1_BAUD_DIV_LSB – UART 1 Baud Rate Divisor LSB **0x1c000b50**
UART2_BAUD_DIV_LSB – UART 2 Baud Rate Divisor LSB **0x1c000b90**

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
reserved																								BAUD_DIV_LSB							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	BAUDDIV_LSB	Baud Rate Divisor [7:0]. Least significant byte of baud rate divisor (BAUDDIV).	R/W	0x00

Note 1:

The data values of the UART_BAUD_DIV_MSB and UART_BAUD_DIV_LSB registers are first stored into a shadow register. Only after a write to the UART_LINE_CTRL register the programmed baudrate is used by the UART baudrate generator.

Note 2:

A divisor value of zero is illegal, and so no transmission or reception will occur.

UART0_CTRL – UART 0 Control Register**0x1c000b14****UART1_CTRL – UART 1 Control Register****0x1c000b54****UART2_CTRL – UART 2 Control Register****0x1c000b94**

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
reserved																							TX_RX_LOOP	LBE	RTIE	TIE	RIE	MSIE	SIRLP	SIREN	UARTEN

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x00
8	TX_RX_LOOP	internal loop (TX -> RX) (test purpose only).	R/W	0
7	LBE	Loop back enable. This bit is cleared to 0 on reset, which disables the loop back mode. Loop Back is only in IrDA mode possible.	R/W	0
6	RTIE	Receive timeout interrupt enable. 1: the receive timeout interrupt is enabled.	R/W	0
5	TIE	Transmit interrupt enable. 1: the transmit interrupt is enabled.	R/W	0
4	RIE	Receive interrupt enable. 1: the receive interrupt is enabled.	R/W	0
3	MSIE	Modem status interrupt enable. 1: the modem status interrupt is enabled.	R/W	0
2	SIRLP	IrDA SIR low power mode. This bit selects the IrDA encoding mode. 0: low level bits are transmitted as an active high pulse with a width of 3 / 16 the of the bit period. 1: low level bits are transmitted with a pulse width which is 3 times the period of the IrLPBaud16 input signal, regardless of the selected bit rate. Setting this bit uses less power, but may reduce transmission distance.	R/W	0
1	SIREN	SIR enable. 1: the IrDA SIR Endec is enabled. This bit has no effect if the UART is not enabled by bit 0 being set to 1. When the IrDA SIR Endec is enabled, data is transmitted and received on nSROUT and SIRIN. Uart_txd remains in the marking state (set to 1). Signal transitions or modem status inputs will have no effect. When the IrDA SIR Endec is disabled, nSROUT remains cleared to 0 (no light pulse generated), and signal transitions on SIRIN will have no effect.	R/W	0
0	UARTEN	UART enable. If this bit is set to 1, the UART is enabled. Data transmission and reception occurs for either UART signals or SIR signals according to the setting of SIR Enable (bit 1).	R/W	0

UART0_FLAG – UART 0 Flag Register**0x1c000b18****UART1_FLAG – UART 1 Flag Register****0x1c000b58****UART2_FLAG – UART 2 Flag Register****0x1c000b98**

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
reserved																								TXFE	RXFF	TXFF	RXFE	BUSY	DCD	DSR	CTS

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7	TXFE	Transmit FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UART_LINE_CTRL register. If the FIFO is disabled, this bit is set when the transmit holding register is empty. If the FIFO is enabled, the TXFE bit is set when the transmit FIFO is empty.	R	0
6	RXFF	Receive FIFO full. The meaning of this bit depends on the state of the FEN bit in the UART_LINE_CTRL register. If the FIFO is disabled, this bit is set when the receive holding register is full. If the FIFO is enabled, the RXFF bit is set when the receive FIFO is full.	R	0
5	TXFF	Transmit FIFO full. The meaning of this bit depends on the state of the FEN bit in the UART_LINE_CTRL register. If the FIFO is disabled, this bit is set when the transmit holding register is full. If the FIFO is enabled, the TXFF bit is set when the transmit FIFO is full.	R	0
4	RXFE	Receive FIFO empty. The meaning of this bit depends on the state of the FEN bit in the UART_LINE_CTRL register. If the FIFO is disabled, this bit is set when the receive holding register is empty. If the FIFO is enabled, the RXFE bit is set when the receive FIFO is empty.	R	0
3	BUSY	UART busy. If this bit is set to 1, the UART is busy transmitting data. This bit remains set until the complete byte, including all the stop bits, has been sent from the shift register. This bit is set as soon as the transmit FIFO becomes non-empty (regardless of whether the UART is enabled or not).	R	0
2	DCD	Data carrier detect. This bit is actually not supported. Always read as 0.	R	0
1	DSR	Data set ready. This bit is actually not supported. Always read as 0.	R	0
0	CTS	Clear to send. This bit is the complement of the modem status input pin UARTi_CTS. That is (when CTS_POL of UART_RTS_CTRL register is not set) the bit is 1 when the modem status input is 0. When CTS_POL of UART_RTS_CTRL register is set to 1 this bit is inverted.	R	0

UART0_INT_ID – UART 0 Interrupt Identification Register**0x1c000b1c****UART1_INT_ID – UART 1 Interrupt Identification Register****0x1c000b5c****UART2_INT_ID – UART 2 Interrupt Identification Register****0x1c000b9c**

UART_INT_ID is the interrupt identification register/interrupt clear register. The memory location has different functions. Interrupt status is read from UART_INT_ID. A write to this register clears the modem status interrupt.

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
reserved																												RTIS	TIS	RIS	MIS

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x00
3	RTIS	Receive timeout interrupt status. 1: if the receive timeout interrupt is asserted.	R/W	0
2	TIS	Transmit interrupt status. 1: if the transmit interrupt is asserted.	R/W	0
1	RIS	Receive interrupt status. 1: if the receive interrupt is asserted.	R/W	0
0	MIS	Modem Interrupt Status. 1: if the modem status interrupt is asserted.	R/W	0

Interrupts

There are four different interrupts generated by the UART. They are combined into a single interrupt request which is an OR function of the individual masked sources. This output is connected to the system interrupt controller VIC to provide another level of masking on an individual peripheral basis. The combined UART interrupt is asserted if any of the four individual interrupts above are asserted and enabled.

The transmit and receive dataflow interrupts have been separated from the status interrupts. This allows to be used in a DMA controller, so that data can be read or written in response to just the FIFO trigger levels. The status of the individual interrupt sources can be read from UART_INT_ID.

The modem status interrupt

is asserted if the modem status line UARTi_CTS change. It is cleared by writing to the UART_INT_ID register.

The receive interrupt

changes state when one of the following events occurs:

- If the FIFOs are enabled and the receive FIFO is half or more full (it contains eight or more words), then the receive interrupt is asserted HIGH. The receive interrupt is cleared by reading data from the receive FIFO until it becomes less than half full. By changing the value of UART_RX_FIFO_IRQ_LVL register you can change the interrupt trigger level.
- If the FIFOs are disabled (have a depth of one location) and data is received thereby filling the location, the receive interrupt is asserted HIGH. The receive interrupt is cleared by performing a single read of the receive FIFO.

The transmit interrupt

changes state when one of the following events occurs:

- If the FIFOs are enabled and the transmit FIFO is at least half empty (it has space for eight or more words), then the transmit interrupt is asserted HIGH. It is cleared by filling the transmit FIFO to more

than half full. By changing the value of UART_TX_FIFO_IRQ_LVL register you can change the interrupt trigger level.

- If the FIFOs are disabled (have a depth of one location) and there is no data present in the transmitters single location, the transmit FIFO is asserted HIGH. It is cleared by performing a single write to the transmitter FIFO.

The transmit interrupt is not qualified with the UART Enable signal, which allows operation in one of two ways. Data can be written to the transmit FIFO prior to enabling the UART and the interrupts.

Alternatively, the UART and interrupts can be enabled so that data can be written to the transmit FIFO by an interrupt service routine.

The receive timeout interrupt

is asserted when the receive FIFO is not empty and no further data is received over a 32-bit period. The receive timeout interrupt is cleared when the FIFO becomes empty through reading all the data (or by reading the holding register).

UART0_IRDA_LO_PWR_CNTR – UART 0 IrDA Low Power Counter Register	0x1c000b20
UART1_IRDA_LO_PWR_CNTR – UART 1 IrDA Low Power Counter Register	0x1c000b60
UART2_IRDA_LO_PWR_CNTR – UART 2 IrDA Low Power Counter Register	0x1c000ba0

UART_IRDA_LO_PWR_CNTR is the IrDA low-power counter register. This is an 8-bit read/write register that stores the low-power counter divisor value used to generate the internal IrLPBaud16 ($16 \times \text{Baudrate} = \text{IrLPBaud16}$) signal.

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
reserved																								ILPDVSR							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	ILPDVSR	IrDA Low Power Divisor [7:0]. 8-bit low-power divisor value.	R/W	0x00

The low power divisor value is calculated as follows:

$$\text{ILPDVSR} = (100\text{MHz} / 16 \times \text{Baudrate}) - 1$$

Note:

Zero is an illegal value. Programming a zero value will result in no IrLPBaud16 pulses being generated.

UART0_RTS_CTRL – UART 0 RTS Control Register**0x1c000b24****UART1_RTS_CTRL – UART 1 RTS Control Register****0x1c000b64****UART2_RTS_CTRL – UART 2 RTS Control Register****0x1c000ba4**

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
reserved																								STICK	CTS_POL	CTS_CTR	RTS_POL	MOD2	COUNT	RTS	AUTO

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7	STICK	Parity bit works as stick bit. 0 : the parity bit not stick and normally calculated. 1 : parity bit is the inverted bit EPS of UART_LINE_CTRL register.	R/W	0
6	CTS_POL	CTS polarity. 0 : CTS input is active low. 1 : CTS input is active high.	R/W	0
5	CTS_CTR	CTS control. 0 : the UART starts transmitting regardless CTS input pin. 1 : the UART starts transmitting only if the CTS input signal is active. After each character which has been send the UART checks if the CTS input is still active. If it is active it continues transmitting otherwise it will wait for the CTS input.	R/W	0
4	RTS_POL	RTS polarity. 0 : RTS output is active low. 1 : RTS output is active high.	R/W	0
3	MOD2	There are two modes you can choose when AUTO is set to 1. 0 : After every character which has been send the internal state machine goes into the trail state which means that the bit stream is stopped for a while (see UART_RTS_TRAIL_CYC register). 1 : The internal state machine goes only into the trail state when the transmit FIFO is empty.	R/W	0
2	COUNT	RTS counter time base. 0 : the internal counter bases on baud times. 1 : the forerun and trail cycles of RTS Signal are counted is system clock cycles (100 MHz).	R/W	0
1	RTS	If AUTO=0 then the RTS output is set by this bit.	R/W	0
0	AUTO	0 : RTS output is controlled directly by bit 1 of UART_RTS_CTRL register. 1 : RTS output is automatically assigned by the internal state machine. See also bit 2-6 of UART_RTS_CTRL register.	R/W	0

UART0_RTS_LEAD_CYC – UART 0 RTS Leading Cycles**0x1c000b28****UART1_RTS_LEAD_CYC – UART 1 RTS Leading Cycles****0x1c000b68****UART2_RTS_LEAD_CYC – UART 2 RTS Leading Cycles****0x1c000ba8**

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
reserved																								LEAD_CYC							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	LEAD_CYC	Number of leading cycles in system clocks or baud rate cycles.	R/W	0x00

UART0_RTS_TRAIL_CYC – UART 0 RTS Trailing cycles**0x1c000b2c****UART1_RTS_TRAIL_CYC – UART 1 RTS Trailing cycles****0x1c000b6c****UART2_RTS_TRAIL_CYC – UART 2 RTS Trailing cycles****0x1c000bac**

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
reserved																								TRAIL_CYC							

Bits	Name	Description	R/W	Default
31:8	Reserved	-	R	0x00
7:0	TRAIL_CYC	Number of trail cycles in system clocks or baud rate cycles.	R/W	0x00

UART0_OUT_DRV_EN – UART 0 Output Driver Enable Register**0x1c000b30****UART1_OUT_DRV_EN – UART 1 Output Driver Enable Register****0x1c000b70****UART2_OUT_DRV_EN – UART 2 Output Driver Enable Register****0x1c000bb0**

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
reserved																														DRVTS	DRVTX

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x00
1	DRVTS	This bit enables the driver for UARTi_RTS output pin.	R/W	0
0	DRVTX	This bit enables the driver for UARTi_TXD output pin.	R/W	0

UART0_BAUD_MODE_CTRL – UART 0 Baud Rate Mode Control Register	0x1c000b34
UART1_BAUD_MODE_CTRL – UART 1 Baud Rate Mode Control Register	0x1c000b74
UART2_BAUD_MODE_CTRL – UART 2 Baud Rate Mode Control Register	0x1c000bb4

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
reserved																															BAUD_RATE_MODE

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x00
0	BAUD_RATE_MODE	Sets the generation method of baudrate. See the 'BAUDDIV' of UART_BAUD_DIV registers for calculating the baudrate.		0

UART0_RX_FIFO_IRQ_LVL – UART 0 RX FIFO Trigger Level and RX-DMA Enable	0x1c000b38
UART1_RX_FIFO_IRQ_LVL – UART 1 RX FIFO Trigger Level and RX-DMA Enable	0x1c000b78
UART2_RX_FIFO_IRQ_LVL – UART 2 RX FIFO Trigger Level and RX-DMA Enable	0x1c000bb8

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					
reserved																										RXDMA	RFIRQLEVEL									

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	RXDMA	<p>Enable DMA-requests for RX-fifo-data. A request will be generated if RX-FIFO is not empty and uart_ctrl. uartEN (module enable) is set. Burst request to DMA-Ctrl will be done if the RX-FIFO contains at least 4 words (set DMA-burst-size to 4) If this bit is reset or the module is disabled, DMA-request will also be reset. single transfer request: RX-FIFO contains 1 byte or more, burst request: 4 bytes or more note: set dmac_ch_ctrl.SBSize = 1 (i.e. burst size: 4) in the DMA module</p>	R/W	0
4:0	RFIRQLEVEL	<p>IRQ trigger level of the receive FIFO. Choose a number between 1 and 16. The UART receive interrupt will be set if the number of received bytes in the receive FIFO are greater than or equal RFIRQLEVEL.</p>	R/W	0x08

UART0_TX_FIFO_IRQ_LVL – UART 0 TX FIFO Trigger Level and TX-DMA Enable **0x1c000b3c**
UART1_TX_FIFO_IRQ_LVL – UART 1 TX FIFO Trigger Level and TX-DMA Enable **0x1c000b7c**
UART2_TX_FIFO_IRQ_LVL – UART 2 TX FIFO Trigger Level and TX-DMA Enable **0x1c000bbc**

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
reserved																										TXDMA	RFIRQLEVEL				

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	TXDMA	Enable DMA-requests for TX-fifo-data. A request will be generated if TX-FIFO is not full and uart_ctrl.uartEN (module enable) is set. Burst request to DMA-Ctrl will be done if at least 4 words are writable to the TX-FIFO (set DMA-burst-size to 4) If this bit is reset or the module is disabled, DMA-request will also be reset. note: set dmac_ch_ctrl.DBSize = 1 (i.e. burst size: 4) in the DMA module	R/W	0
4:0	TFIRQLEVEL	IRQ trigger level of the transmit FIFO. Choose a number between 1 and 16. The UART transmit interrupt will be set if the number of transmitted bytes in the transmit FIFO are less than TFIRQLEVEL.	R/W	0x08

6.5 SPI – Serial Peripheral Interface

The netX50 is equipped with two independent, DMA capable SPI units. To allow the generation of netX50 code that is still compatible with the SPI unit of the netX100/500, a set of four additional legacy registers per SPI channel was implemented, located at 0x00100d30 – 0x00100d3f (SPI channel 0) and 0x00100d70 – 0x00100d7c (SPI channel 1). However, in order to fully use the features of the SPI unit, the new register sets (0x1c000d00 - 0x00100d27 and 0x1c000d40 - 0x00100d67) must be used.

ARM Address	Register Name	Short Description
0x1c000d00	SPI0_CTRL0	SPI Channel 0, Control Register 0
0x1c000d04	SPI0_CTRL1	SPI Channel 0, Control Register 1
0x1c000d08	SPI0_DATA	SPI Channel 0, Data Register
0x00100d0c	SPI0_STAT	SPI Channel 0, Status Register
0x00100d10	SPI0_CLK_PRE_SCL	SPI Channel 0, Clock Prescale Register
0x00100d14	SPI0_INT_MSK_SET_CLR	SPI Channel 0, Interrupt Mask Set or Clear Register
0x00100d18	SPI0_RAW_INT_STAT	SPI Channel 0, RAW Interrupt Status Register
0x00100d1c	SPI0_MASK_INT_STAT	SPI Channel 0, Masked Interrupt Status Register
0x00100d20	SPI0_INT_CLR	SPI Channel 0, Interrupt Clear Register
0x00100d24	SPI0_DMA_CTRL	SPI Channel 0, DMA Control Register
0x00100d30	SPI0_LGY_DATA	SPI Channel 0, Legacy Data Register
0x00100d34	SPI0_LGY_STAT	SPI Channel 0, Legacy Status Register
0x00100d38	SPI0_LGY_CTRL	SPI Channel 0, Legacy Control Register
0x00100d3C	SPI0_LGY_INT_CTRL	SPI Channel 0, Legacy Interrupt Control Register
0x1c000d40	SPI1_CTRL0	SPI Channel 1, Control Register 0
0x1c000d44	SPI1_CTRL1	SPI Channel 1, Control Register 1
0x1c000d48	SPI1_DATA	SPI Channel 1, Data Register
0x00100d4c	SPI1_STAT	SPI Channel 1, Status Register
0x00100d50	SPI1_CLK_PRE_SCL	SPI Channel 1, Clock Prescale Register
0x00100d54	SPI1_INT_MSK_SET_CLR	SPI Channel 1, Interrupt Mask Set or Clear Register
0x00100d58	SPI1_RAW_INT_STAT	SPI Channel 1, RAW Interrupt Status Register
0x00100d5c	SPI1_MASK_INT_STAT	SPI Channel 1, Masked Interrupt Status Register
0x00100d60	SPI1_INT_CLR	SPI Channel 1, Interrupt Clear Register
0x00100d64	SPI1_DMA_CTRL	SPI Channel 1, DMA Control Register
0x00100d70	SPI1_LGY_DATA	SPI Channel 1, Legacy Data Register
0x00100d74	SPI1_LGY_STAT	SPI Channel 1, Legacy Status Register
0x00100d78	SPI1_LGY_CTRL	SPI Channel 1, Legacy Control Register
0x00100d7C	SPI1_LGY_INT_CTRL	SPI Channel 1, Legacy Interrupt Control Register

SPI0_CTRL0 – SPI0 Control Register 0**0x1c000d00****SPI1_CTRL0 – SPI1 Control Register 0****0x1c000d40**

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						
NETX100_COMP		reserved		SLAVE_SIG_EARLY		FILTER_IN		reserved		FORMAT		reserved				SCK_MULADD												SPH		SPO		reserved		DSS			

Bits	Name	Description	R/W	Default
31	NETX100_COMP	use netx100/500-compatible SPI mode: 0: start transfer after writing data 1: start transfer after setting CR_write or CR_read	R/W	1
30:29	reserved	-	R/W	0
28	SLAVE_SIG_EARLY	Generate MISO in slavemode 1 spi_sck clock edge earlier than Spec-defined. This is to compensate Pad/sampling-delays on fast data rates. If filter_in is enabled, it takes in worst case 3 system clocks to generate MISO after SCK. If filter_in is disabled, it takes in worst case 2 system clocks to generate MISO after SCK.	R/W	0
27	FILTER_IN	Receive-data is sampled every 10ns (100MHz sysem clock). If this bit is set, the stored receive value will be the result of a majority decision of the three sampling points around a SPI-clock edge (if two or more '1s' were sampled a '1' will be stored, else a '0' will be stored. In slave mode FSS and SPI-clock edges will also be detected by oversampling if this bit is set: An edge will be detected if the majority-result of thre subsequent sampled values toggles. Input filtering should be used for sck_muladd<=0x200 (i.e. below 12.5MHz). For higher frequencies stable signal phases are too short.	R/W	0
26	reserved	-	R/W	0
25:24	FORMAT	frame format 00: Motorola SPI frame format 01..11: reserved	R/W	0
23:20	reserved	-	R/W	0
19:8	SCK_MULADD	serial clock rate multiply add value for master spi_sck generation spi_sck-frequency: $f_{spi_sck} = (sck_muladd * 100)/4096$ [MHz] in slave-mode SPI-clock must not exceed (system-frequency/4) if correct data sampling should be always guaranteed.	R/W	0x800
7	SPH	serial clock phase (netx500: CR_ncpha) 1: sample data at second clock edge edge, data is generated half a clock phase before sampling 0: sample data at first clock edge edge, data is generated half a clock phase before sampling	R/W	0
6	SPO	serial clock polarity (netx500: CR_cpol) 0: idle: clock is low, first edge is rising 1: idle: clock is high, first edge is falling	R/W	0x00
5:4	reserved	-	R/W	0x00

3:0	DSS	DSS: data size select (transfer size = data size + 1 bits) 0000...0010: reserved 0011: 4 bit 0100: 5 bit ... 0111: 8 bit ... 1111: 16 bit	R/W	0x07
-----	-----	--	-----	------

SPI0_CTRL1 – SPI0 Control Register 1
SPI1_CTRL1 – SPI1 Control Register 1

0x1c000d04
0x1c000d44

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			
reserved				RX_FIFO_CLR	RX_FIFO_WM				reserved				TX_FIFO_CLR	TX_FIFO_WM				reserved				FSS_STATIC	FSS				reserved				SOD	MS	SSE	LBM

Bits	Name	Description	R/W	Default
31:29	reserved	-	R/W	0
28	RX_FIFO_CLR	extended: writing "1" to this bit will clear the receive-FIFOs	R/W	0
27:24	RX_FIFO_WM	receive FIFO watermark for IRQ-generation	R/W	0x8
23:21	reserved	-	R/W	0
20	TX_FIFO_CLR	extended: writing "1" to this bit will clear the transmit-FIFOs There must be at least 1 system-clock idle after clear before writing new data to the FIFO.	R/W	0
19:16	TX_FIFO_WM	transmit FIFO watermark for IRQ-generation	R/W	0x8
15:12	reserved	-	R/W	0
11	FSS_STATIC	SPI static chip select 0: SPI-chip select will be toggled automatically at data frame begin/end according to fss and FRF0 1: SPI-chip select will be set statically according to fss and FRF0 If fss is set to statically, fss must be toggled manually after each data frame in Motorola SPI mode when SPH is 0 for spec compatibility!	R/W	0
10:8	FSS	extended: Frame or slave select (up to 3 devices can be assigned directly, up to 8 devices can be assigned if an external de-multiplexer is used if device is master. For active low slave select (e.g. Motorola SPI frame format) the bits will be inverted before output. If device is slave, the programmed value is a mask to select which slave-fss-input should be considered. e.g.: "010" : fss[1] is slave frame or select input.	R/W	0
7:4	reserved	-	R/W	0x00
3	SOD	slave mode output disable (to connect multiple slaves to one master) 0: SPI-MISO can be driven in slave mode 1: SPI-MISO is not driven in slave mode	R/W	0
2	MS	mode select: 0: device is configured as master 1: device is configured as slave	R/W	0
1	SSE	SPI enable. 0: interface disabled 1: interface enabled	R/W	0
0	LBM	loop back mode 0: internal loop back disabled 1: internal loop back enabled, spi_ctrl0.filter_in must be set for loop-back function	R/W	0

SPI0_DATA – SPI0 Data Register
SPI1_DATA – SPI1 Data Register

0x1c000d08
0x1c000d48

read access: received data byte is delivered from receive FIFO

write access: send data byte is written to send FIFO

Both, receive and transmit FIFO have a depth of 16.

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
reserved																DATA															

Bits	Name	Description	R/W	Default
31:17	reserved	-	R/W	0x00
16:0	DATA	<p>Transmit data, must be right aligned on writing, only bits according to spi_ctrl0.DSS are considered</p> <p>Receive data will be delivered right aligned, unused bits (spi_ctrl0.DSS < 0xF) will be "0".</p> <p>In slavemode transmit data is requested from the FIFO when the last bit of the current transfer-word is set to spi_miso.</p> <p>If no next transmit data could be read from the FIFO until current words last bit was transfered,</p> <p>FIFO underrun will occur if FSS does not go inactive (last word was transfer end) at the next detected spi_sck-edge.</p>	R/W	0x00

SPI0_STAT – SPI0 Status Register**0x1c000d0c****SPI1_STAT – SPI1 Status Register****0x1c000d4c**

SPI master mode: MISO-input-data will be stored in the receive FIFO, transmit FIFO generates MOSI-output-data

SPI slave mode: MOSI-input-data will be stored in the receive FIFO, transmit FIFO generates MISO-output-data

Shows the current status of the SPI interface.

Both, receive and transmit FIFO have a depth of 16.

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
RX_FIFO_ERR_UNDR	RX_FIFO_ERR_OVFL	reserved	RX_FIFO_LEVEL					TX_FIFO_ERR_UNDR	TX_FIFO_ERR_OVFL	reserved	TX_FIFO_LEVEL				reserved												BSY	RFF	RNE	TNF	TFE

Bits	Name	Description	R/W	Default
31	RX_FIFO_ERR_UNDR	extended: receive FIFO under-run error occurred, data is lost	R	0
30	RX_FIFO_ERR_OVFL	extended: receive FIFO overflow error occurred, data is lost	R	0
29	reserved	-	R	0
28:24	RX_FIFO_LEVEL	extended: receive FIFO level (number of received words to read out are left in FIFO)	R	0
23	TX_FIFO_ERR_UNDR	extended: transmit FIFO under-run error occurred, data is lost	R	0
22	TX_FIFO_ERR_OVFL	extended: transmit FIFO overflow error occurred, data is lost	R	0
21	reserved	-	R	0
20:16	TX_FIFO_LEVEL	extended: transmit FIFO level (number of words to transmit are left in FIFO)	R	0
15:5	reserved	-	R	0
4	BSY	device busy (1 if data is currently transmitted/received or the transmit FIFO is not empty)	R	0
3	RFF	receive FIFO is full (1 if full)	R	0
2	RNE	receive FIFO is not empty (0 if empty)	R	0
1	TNF	transmit FIFO is not full (0 if full)	R	0
0	TFE	transmit FIFO is empty (1 if empty)	R	0

SPI0_CLK_PRE_SCL – SPI0 Clock Prescale Register**0x1c000d10****SPI1_CLK_PRE_SCL – SPI1 Clock Prescale Register****0x1c000d50**

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						
reserved																								CPSDVSR													

Bits	Name	Description	R/W	Default
31:8	reserved	-	R/W	0x00
7:0	CPSDVSR	obsolete	R/W	0

SPI0_INT_MSK_SET_CLR – SPI0 Interrupt Mask Set or Clear**0x1c000d14****SPI1_INT_MSK_SET_CLR – SPI1 Interrupt Mask Set or Clear****0x1c000d54**

Both, receive and transmit FIFO have a depth of 16.

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
reserved																								TXEIM	RXFIM	RXNEIM	TXIM	RXIM	RTIM	RORIM	

Bits	Name	Description	R/W	Default
31:7	reserved	-	R/W	0x00
6	TXEIM	transmit FIFO empty interrupt mask (for netx100/500 compliance)	R/W	0
5	RXFIM	receive FIFO full interrupt mask (for netx100/500 compliance)	R/W	0
4	RXNEIM	receive FIFO not empty interrupt mask (for netx100/500 compliance)	R/W	0
3	TXIM	transmit FIFO interrupt mask	R/W	0
2	RXIM	receive FIFO interrupt mask	R/W	0
1	RTIM	receive timeout interrupt mask	R/W	0
0	RORIM	receive FIFO overrun interrupt mask	R/W	0

SPI0_RAW_INT_STAT – SPI0 RAW Interrupt Status Register**0x1c000d18****SPI1_RAW_INT_STAT – SPI1 RAW Interrupt Status Register****0x1c000d58**

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	
reserved																										TXERIS	RXFRIS	RXNERIS	TXRIS	RXRIS	RTRIS	RORRIS

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x00
6	TXERIS	unmasked transmit FIFO empty interrupt state (for netx100/500 compliance) 1: transmit FIFO is empty 0: transmit FIFO is not empty	R	0
5	RXFRIS	unmasked receive FIFO full interrupt state (for netx100/500 compliance) 1: receive FIFO is full 0: receive FIFO is not full	R	0
4	RXNERIS	unmasked receive FIFO not empty interrupt state (for netx100/500 compliance) 1: receive FIFO is not empty 0: receive FIFO is empty	R	0
3	TXRIS	unmasked transmit FIFO interrupt state 1: transmit FIFO level is below spi_ctrl1.tx_fifo_wm 0: transmit FIFO equals or is higher than spi_ctrl1.tx_fifo_wm	R	0
2	RXRIS	unmasked receive FIFO interrupt state 1: receive FIFO is higher than spi_ctrl1.rx_fifo_wm 0: receive FIFO is equals or is below spi_ctrl1.rx_fifo_wm	R	0
1	RTRIS	unmasked receive timeout interrupt state timeout period are 32 SPI-clock periods depending on spi_ctrl0.SCR 1: receive FIFO is not empty and not read out in the passed timeout period 0: receive FIFO is empty or read during the last timeout period	R	0
0	RORRIS	unmasked receive FIFO overrun interrupt state 1: receive FIFO overrun error occurred 0: no receive FIFO overrun error occurred	R	0

SPI0_MSK_INT_STAT – SPI0 Mask Interrupt Status Register**0x1c000d1c****SPI1_MSK_INT_STAT – SPI1 Mask Interrupt Status Register****0x1c000d5c**

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	
reserved																										TXEMIS	RXFMIS	RXNEMIS	TXMIS	RXMIS	RTMIS	RORMIS

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x00
6	TXEMIS	masked transmit FIFO empty interrupt state (for netx100/500 compliance)	R	0
5	RXFMIS	masked receive FIFO full interrupt state (for netx100/500 compliance)	R	0
4	RXNEMIS	masked receive FIFO not empty interrupt state (for netx100/500 compliance)	R	0
3	TXMIS	masked transmit FIFO interrupt state	R	0
2	RXMIS	masked receive FIFO interrupt state	R	0
1	RTMIS	masked receive timeout interrupt state	R	0
0	RORMIS	masked receive FIFO overrun interrupt state	R	0

SPI0_INT_CLR – SPI0 Interrupt Clear Register
SPI1_INT_CLR – SPI1 Interrupt Clear Register

0x1c000d20
0x1c000d60

Interrupt is cleared by writing "1" to the corresponding bit.

FIFO-state interrupts are cleared automatically if interrupt criteria is no longer true.

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	
reserved																										TXEIC	RXFIC	RXNEIC	TXIC	RXIC	RTIC	RORIC

Bits	Name	Description	R/W	Default
31:7	reserved	-	R/W	0x00
6	TXEIC	clear transmit FIFO empty interrupt (for netx100/500 compatibility)	R/W	0
5	RXFIC	clear receive FIFO full interrupt (for netx100/500 compatibility)	R/W	0
4	RXNEIC	clear receive FIFO not empty interrupt (for netx100/500 compliance)	R/W	0
3	TXIC	PL022 extension: clear transmit FIFO interrupt	R/W	0
2	RXIC	PL022 extension: clear receive FIFO interrupt	R/W	0
1	RTIC	clear receive FIFO overrun interrupt	R/W	0
0	RORIC	clear receive FIFO overrun interrupt writing '1' here will clear the receive FIFO	R/W	0

SPI0_DMA_CTRL – SPI0 DMA Control Register**0x1c000d24****SPI1_DMA_CTRL – SPI1 DMA Control Register****0x1c000d64**

Only single transfer requests will be generated by this module.

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
reserved																												TXDMAE	RXDMAE		

Bits	Name	Description	R/W	Default
31:2	reserved	-	R/W	0x00
1	TXDMAE	enable DMA for SPI-transmit data A request will be generated if TX-FIFO is not full and spi_ctrl1.SSE (module enable) is set. Burst request to DMA-Ctrl will be done if at least 4 words are writable to the TX-FIFO (set DMA-burst-size to 4) If this bit is reset or the module is disabled, DMA-request will also be reset. note: set dmac_ch_ctrl.SBSize = 1 (i.e. burst size: 4) in the DMA module	R/W	0
0	RxDMAE	enable DMA for SPI-receive data A request will be generated if RX-FIFO is not empty and spi_ctrl1.SSE (module enable) is set. Burst request to DMA-Ctrl will be done if the RX-FIFO contains at least 4 words (set DMA-burst-size to 4) If this bit is reset or the module is disabled, DMA-request will also be reset. note: set dmac_ch_ctrl.SBSize = 1 (i.e. burst size: 4) in the DMA module	R/W	0

Legacy Registers:**SPI0_LGY_DATA – SPI0 Legacy Data Register****0x1c000d30****SPI1_LGY_DATA – SPI1 Legacy Data Register****0x1c000d70**

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
reserved														DR_VALID1	DR_VALID0	DATA_BYTE_1								DATA_BYTE_0							

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x00
17	DR_VALID1	Obsolete, always 0	R/W	0
16	DR_VALID0	valid bit for data_byte_0 This bit shows if DATA_BYTE_0 is valid. Must be set during FIFO write access	R/W	0
15:8	DATA_BYTE_1	Obsolete, don't use!	R/W	0x00
7:0	DATA_BYTE_0	data byte 0	R/W	0x00

SPI0_LGY_STAT – SPI0 Legacy Status Register**0x1c000d34****SPI1_LGY_STAT – SPI1 Legacy Status Register****0x1c000d74**

Shows current status of the SPI interface. Bits 23 and 21-18 show active interrupts, writing '1' to these bits clears the interrupts. Writing other bits has no effect

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
reserved						SR_SELECTED	SR_OUT_FULL	SR_OUT_EMPTY	SR_OUT_FW	SR_OUT_FUEL	SR_IN_FULL	SR_IN_RECDATA	SR_IN_FUEL	SR_OUT_FUEL_VAL								SR_IN_FUEL_VAL									

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x00
25	SR_SELECTED	external master has access to spi-interface	R	0
24	SR_OUT_FULL	output FIFO is full (no IRQ)	R/W	0
23	SR_OUT_EMPTY	output FIFO is empty (in slave mode)	R/W	0
22	SR_OUT_FW	ARM is writing data too fast into output FIFO (no IRQ)	R/W	0
21	SR_OUT_FUEL	adjustable fuel value of output FIFO reached	R/W	0
20	SR_IN_FULL	input FIFO is full	R/W	0
19	SR_IN_RECDATA	valid data bytes in input FIFO	R/W	0
18	SR_IN_FILL_LEVEL	adjustable fill level of input FIFO reached	R/W	0
17:9	SR_OUT_FILL_VAL	output FIFO fill value	R	0x00
8:0	SR_IN_FILL_VAL	input FIFO fill value	R	0x00

SPI0_LGY_CTRL – SPI0 Legacy Control Register
SPI1_LGY_CTRL – SPI1 Legacy Control Register
0x1c000d38
0x1c000d78

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
CR_EN	CR_MS	CR_CPOL	CR_NCPHA	CR_BURST		CR_BURSTDELAY		CR_CLR_OUTFIFO		CR_CLR_INFIFO	reserved										CS_MODE	CR_SS		CR_WRITE	CR_READ	reserved	CR_SPEED			CR_SOFTRESET	

Bits	Name	Description	R/W	Default
31	CR_EN	0 : disable SPI interface 1 : enable SPI interface	R/W	0x00
30	CR_MS	0 : slave mode 1 : master mode	R/W	0x00
29	CR_CPOL	0 : rising edge of spi_sck is primary 1 : falling edge of spi_sck is primary	R/W	0x00
28	CR_NCPHA	relative to CR_cpol 1 : change data to primary spi_sck edge data active at secondary spi_sck edge 0 : change data to secondary spi_sck edge data active at primary spi_sck edge	R/W	0x00
27:25	CR_BURST	Obsolete	R	0x00
24:22	CR_BURSTDELAY	Obsolete	R	0x00
21	CR_CLR_OUTFIFO	clear output FIFO	R/W	0x00
20	CR_CLR_INFIFO	clear input FIFO	R/W	0x00
19:12	reserved	-	R	0x00
11	CS_MODE	0 : chip select is directly controlled by software. (see bits CR_SS). 1 : chip select is generated automatically by the internal state machine.	R/W	0
10:8	CR_SS	external slave select	R/W	0x00
7	CR_WRITE	Obsolete	R	0x01
6	CR_READ	Obsolete	R	0x01
5	reserved	-	R	0
4:1	CR_SPEED	clock divider for SPI clock ($2 - 2^{16}$) If SPI Clock-rate is changed by adr_spi_cr0.SCR, this value will not be updated and may be incorrect. There are 16 different SPI Clocks to choose: 0001 : 0,025 MHz (Mode not compatible to netX 500/100) 0010 : 0,05 MHz 0011 : 0,2 MHz 0100 : 0,5 MHz 0101 : 1 MHz 0110 : 1,25 MHz 0111 : 2 MHz 1000 : 2,5 MHz 1001 : 3,3333 MHz 1010 : 5 MHz 1011 : 10 MHz 1100 : 12,5 MHz 1101 : 16,6666 MHz 1110 : 25 MHz 1111 : 50 MHz	R/W	0x00
0	CR_SOFTRESET	Clears IRQs and FIFOs	R/W	0

SPI0_LGY_INT_CTRL – SPI0 Legacy Interrupt Control Register**0x1c000d3c****SPI1_LGY_INT_CTRL – SPI1 Legacy Interrupt Control Register****0x1c000d7c**

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
reserved							IR_OUT_FULL_EN	IR_OUT_EMPTY_EN	IR_OUT_FW_EN	IR_OUT_FUEL_EN	IR_IN_FULL_EN	IR_IN_RECDATA_EN	IR_IN_FUEL_EN	IR_OUT_FUEL								IR_IN_FUEL									

Bits	Name	Description	R/W	Default
31:25	reserved	-	R	0x00
24	IR_OUT_FULL_EN	Obsolete	R/W	0
23	IR_OUT_EMPTY_EN	IRQ enable for output FIFO is empty and interface sending data (occurs only in slave mode)	R/W	0
22	IR_OUT_FW_EN	Obsolete	R/W	0
21	IR_OUT_FUEL_EN	IRQ enable for adjustable fuel value of output FIFO reached	R/W	0
20	IR_IN_FULL_EN	IRQ enable for input FIFO is full	R/W	0
19	IR_IN_RECDATA_EN	IRQ enable for valid data bytes in input FIFO	R/W	0
18	IR_IN_FUEL_EN	IRQ enable for adjustable fill level of input FIFO reached	R/W	0
17:9	IR_OUT_FUEL	Watermark level for output FIFO	R	0x00
8:0	IR_IN_FUEL	Watermark level for input FIFO	R	0x00

6.6 I2C – Serial I2C-Interface

The I2C module provides the following registers for configuration and data access:

ARM Address	Register Name	Short Description
0x1c000e00	I2C_MASTER_CTRL	I2C Master Control Register
0x1c000e04	I2C_SLAVE_CTRL	I2C Slave Control Register
0x1c000e08	I2C_MASTER_CMD	I2C Master Command Register
0x1c000e0c	I2C_MASTER_DATA	I2C Master Data Register
0x1c000e10	I2C_SLAVE_DATA	I2C Slave Data Register
0x1c000e14	I2C_MASTER_FIFO_CTRL	I2C Master FIFO Control Register
0x1c000e18	I2C_SLAVE_FIFO_CTRL	I2C Slave FIFO Control Register
0x1c000e1c	I2C_STAT	I2C Status Register
0x1c000e20	I2C_INT_MSK_SET_CLR	I2C Interrupt Mask set or clear Register
0x1c000e24	I2C_RAW_INT_STAT	I2C RAW Interrupt Status Register
0x1c000e28	I2C_MSK_INT_STAT	I2C Mask Interrupt Status Register
0x1c000e2c	I2C_DMA_CTRL	I2C DMA Control Register

Note:

This module is not compatible to the netX100/netX500 I2C module.

I2C_MASTER_CTRL – I2C Master Control Register**0x1c000e00**

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
reserved																					SADR							MODE		EN_I2C	

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x00
10:4	SADR	7-bit Slave address send after (r)START: For 10-bit addressing, the first byte (10bit-start "11110", MSB[9:8] must be programmed here, second start byte (slave address LSBs) must be top of the master FIFO (i2c_master_data). This register must be rewritten (even if value does not change) if another 10-bit addressed slave shall be addressed (run 2-byte start sequence). It must not be rewritten before repeated START on the same 10-bit addressed slave (run 1-byte start sequence e.g. write to read change).	R/W	0x00
3:1	MODE	I2C-speed-mode: If this device is used only as slave, mode should be set to the maximum data rate generated by the fastest master on the I2C-bus for appropriate input filtering and spike suppression 000: Fast/Standard-mode, 50kbit/s 001: Fast/Standard-mode, 100kbit/s 010: Fast/Standard-mode, 200kbit/s 011: Fast/Standard-mode, 400kbit/s 100: Fast/Standard-mode, 800kbit/s 101: Fast/Standard-mode, 1.2Mbit/s 110: High-speed-mode, 1.7Mbit/s 111: High-speed-mode, 3.4Mbit/s)	R/W	0x00
0	EN_I2C	0 : interface disable 1 : interface enable	R/W	0

I2C_SLAVE_CTRL – I2C Slave Control Register**0x1c000e04**

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
reserved													AC_GCALL	AC_START	AC_SRX	reserved						SID10	SID								

Bits	Name	Description	R/W	Default
31:19	Reserved	-	R	0
18	AC_GCALL	General Call acknowledge: 0: do not generate acknowledge after General Call 1: generate acknowledge after General Call	R/W	0
17	AC_START	Enable start sequence acknowledge: The start byte (2 bytes if sid10 is set) will be acknowledged if the received address matches the sid-bits. If master requests a read transfer, slave FIFO read access is done immediately after acknowledge, so valid data must be present in the slave FIFO before acknowledge is enabled. This bit should be reset by software after the start sequence was acknowledged to avoid acknowledge and FIFO errors after next (r)START. 0: do not generate acknowledge after start sequence. 1: generate acknowledge after start sequence.	R/W	0
16	AC_SRX	Enable slave-receive-data acknowledge: 0: do not generate acknowledge on receive bytes. 1: generate acknowledge on receive bytes. No acknowledge will be generated on receive data if the slave FIFO is full.	R/W	0
15:11	reserved	-	R	0
10	SID10	10-bit Slave device ID: 0: listen for 7bit slave address after (r)START 1: listen for 10bit slave address after (r)START	R/W	0
9:0	SID	Slave device ID: External masters can address this device by this address.	R/W	0

I2C_MASTER_CMD – I2C Master Command Register**0x1c000e08**

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
reserved				ACPOLLMAX								reserved	TSIZE								reserved				CMD			NWR			

Bits	Name	Description	R/W	Default
31:28	reserved	-	R	0
27:20	ACPOLLMAX	acpollmax+1 (1...256) tries for start sequence acknowledge polling: For 7-bit addressed acknowledge polling START and the first byte containing the slave address (i2c_master_ctrl.sadr) will be repeated up to acpollmax+1 times until a slave generates acknowledge. If no acknowledge is received within acpollmax+1 tries, the cmd_err IRQ will be generated. For 10-bit-addressed slaves, the 2-byte start sequence is done. The second address-byte (LSBs) must be top of the master FIFO (i2c_master_data). This byte must not be regarded by the value programmed in tsize for subsequent transfers. This value will count down during acknowledge polling after each start sequence.	R/W	0
19:18	reserved	-	R/W	0
17:8	TSIZE	Transfer tsize+1 bytes (1...1024): If no acknowledge was generated by slave (receiver), write transfers will be terminated and the cmd_err IRQ will be generated. For 10-bit-addressed slaves, the second start-byte (LSBs) must be top of the master FIFO. This byte must not be regarded by the value programmed here for subsequent transfers. This value will count down during transfers after each byte.	R/W	0
7:4	reserved	-	R	0
3:1	CMD	I2C sequence command: All commands will either generate the cmd_ok IRQ or the cmd_err IRQ. Successful command termination will always generate cmd_ok IRQ. If a command could not be finished successfully, cmd_err IRQ will be set. For 10-bit-addressed slaves, the second start-byte (LSBs) must be top of the master FIFO. 000: START: generate (r)START-condition. 001: S_AC: acknowledge-polling: generate up to acpollmax+1 START-sequences (until acknowledged by slave). 010: S_AC_T: run S_AC, then transfer tsize+1 bytes from/to master FIFO. Not to be continued. 011: S_AC_TC: run S_AC, then transfer tsize+1 bytes from/to master FIFO. To be continued. 100: CT: Continued transfer not to be continued. 101: CTC: Continued transfer to be continued. 110: STOP generate STOP-condition. 111: IDLE nothing to do, last command finished, break current command. Sequences including not to be continued transfers (S_AC_T, CT) will generate no acknowledge (if read) after last the received byte (read transfer ends). To be continued transfers (S_AC_TC, CTC) will generate acknowledge after the last received byte and must be followed by CT or CTC. Before continued transfers (CT, CTC) a command including START (START, S_AC, S_AC_T, S_AC_TC) must be done to	R/W	0x07

		generate a valid I2C sequence.. View i2c_master_data description for FIFO error handling. STOP must always be done by software to free the bus after transfer end. STOP is not included in any command sequence and never done automatically by this device. Some commands are handled as sequences (i.e after setting S_AC_T, first S_AC, later CT will be seen when read out).		
0	NWR	Transfer direction: 0: cmd will be done as write. 1: cmd will be done as read. Master FIFO-requests (IRQ and DMA) are generated depending this direction flag.	R/W	0

I2C_MASTER_DATA – I2C Master Data Register (Master FIFO)**0x1c000e0c**

There is only one FIFO for both, receive and transmit master data with a depth of 16 bytes. For master write access, data send by the master is delivered from the FIFO, for master read access data received by the master is stored in the FIFO.

In case of imminent data transfer failure (read transfer and FIFO is full or write transfer and FIFO is empty), the cmd_err IRQ will be set after the last byte that could be transmitted. No FIFO-underrun or overflow will occur. i2c_master_cmd.tsize+1 will show amount of not transmitted data.

In case of master write transfer direction, either the FIFO can be filled and the transfer may be completed (CTC, CT) or the transfer can be broken (rSTART, STOP).

In case of master read transfer direction, the command will terminate when the FIFO is full. The last read byte will be acknowledged and stored in the FIFO. After reading out data from the FIFO the transfer must be completed (CTC, CT) to flag read data end (no acknowledge at last byte). STOP or rSTART will fail if next read data MSB is 0 (as the next bit already driven by the slave is 0).

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
reserved																								MDATA							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0
7:0	MDATA	I2C master transmit or receive data: Write data will be removed from the FIFO after receiving slave has generated the according acknowledge. Not acknowledged write data will not be removed from the FIFO.	R/W	0

I2C_SLAVE_DATA – I2C Slave Data Register (Slave FIFO)**0x1c000e10**

There is only one FIFO for both, receive and transmit slave data with a depth of 16 bytes. For master read access, data send by the slave is delivered from the FIFO, for master write access data received by the slave is stored in the FIFO.

A transfer is initiated after detection of I2C-start-sequence to the device address (`i2c_slave_ctrl.sid`, `sreq IRQ`) which is acknowledged by this device (`i2c_slave_ctrl.ac_start`). For read transfers send data is read from the FIFO immediately after acknowledge was detected on the I2C-bus. SDA will be driven with next data MSB immediately after acknowledge SCL high phase.

In case of master read transfer and slave FIFO underrun, corrupted data will be send to the master and the `fifo_err` IRQ will be set.

In case of master write transfer and slave FIFO is full, no acknowledge will be generated for the last received byte. No FIFO overflow will occur but the last transferred byte (not acknowledged) will be lost and has to be send again by the master.

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
reserved																								SDATA							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0
7:0	SDATA	I2C slave transmit or receive data: <code>i2c_slave_ctrl.ac_start</code> must be handled correctly by software to avoid FIFO errors after (r)START.	R/W	0

I2C_MASTER_FIFO_CTRL – I2C Master FIFO Control Register**0x1c000e14**

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
reserved																							MFIFO_CLR	reserved				MFIFO_WM			

Bits	Name	Description	R/W	Default
31:9	reserved	-	R/W	0
8	MFIFO_CLR	Clear master data FIFO, write only bit.	W	0
7:4	reserved	-	R/W	0
3:0	MFIFO_WM	Master FIFO watermark for mfifo_req IRQ generation: If master is transmitter (enabled and nwr==0-command), mfifo_req IRQ is generated if mfifo_level<mfifo_wm. If master is receiver (enabled and nwr==1-command), mfifo_req IRQ is generated if mfifo_level>mfifo_wm. Setting the watermark to 0 at transfer end avoids further IRQ generation.	R/W	0

I2C_SLAVE_FIFO_CTRL – I2C Slave FIFO Control Register**0x1c000e18**

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
reserved																							SFIFO_CLR	reserved				SFIFO_WM			

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0
8	SFIFO_CLR	Clear slave data FIFO, write only bit.	W	0
7:4	reserved	-	R	0
3:0	SFIFO_WM	Slave FIFO Watermark for sfifo_req IRQ generation: If slave is transmitter (start sequence with set read bit was acknowledged by this slave), sfifo_req IRQ is generated if sfifo_level<sfifo_wm. If slave is not transmitter (is receiver or not selected), sfifo_req IRQ is generated if sfifo_level>sfifo_wm	R/W	0

I2C_STAT – I2C Status Register**0x1c000e1c**

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
SDA_STATE	SCL_STATE	reserved		SID10_ACED	GCALL_ACED	NWR_ACED	LAST_AC	SLAVE_ACCESS	STARTED	NWR	BUS_MASTER	SFIFO_ERR_UNDR	SFIFO_ERR_OVFL	SFIFO_FULL	SFIFO_EMPTY	reserved	SFIFO_LEVEL				MFIFO_ERR_UNDR		MFIFO_ERR_OVFL	MFIFO_FULL	MFIFO_EMPTY	reserved	MFIFO_LEVEL				

Bits	Name	Description	R/W	Default
31	SDA_STATE	SDA signal state sampled and filtered from bus (e.g. to detect bus blockings)	R	1
30	SCL_STATE	SCL signal state sampled and filtered from bus (e.g. to detect bus blockings)	R	1
29:28	reserved	-	R	0
27	SID10_ACED	Master detected that a 10-bit addressed slave acknowledge the 2-byte start sequence. Master will generate only first START-byte during rSTART. 0: SDA was high i.e no acknowledge. 1: SDA was low i.e acknowledge).	R	0
26	GCALL_ACED	Received General Call was acknowledged (General Call was done and i2c_slave_ctrl.ac_gcall is set). 0: SDA was high i.e no acknowledge. 1: SDA was low i.e acknowledge).	R	0
25	NWR_ACED	Last transfer direction (nwr-bit during start-byte with address matching this slave) acknowledged by this slave to handle slave FIFO (0: write; 1: read). Slave FIFO-requests (IRQ and DMA) are generated depending this direction flag	R	0
24	LAST_AC	Last acknowledge detected on bus: 0: SDA was high i.e no acknowledge. 1: SDA was low i.e acknowledge).	R	0
23	SLAVE_ACCESS	0: No slave access on this device (reset at START or STOP). 1: A master addressed this slave device (set if start-byte with address matching this slave).	R	0
22	STARTED	START condition detection: This detection is also done, if this device is not enabled to get current bus state after enable. 0: bus is idle (Stop was detected, not started). 1: (r)START was detected on bus.	R	0
21	NWR	Transfer direction detected after last (s)START. 0: write; 1: read. This bit is reset to 0 during START and does not care for slave acknowledge.	R	0
20	BUS_MASTER	1: master gains bus arbitration or bus is idle, 0: master lost bus arbitration, bus is busy by another master	R	0
19	SFIFO_ERR_UNDR	slave FIFO underrun error occurred, data is lost	R	0
18	SFIFO_ERR_OVFL	slave FIFO overflow error occurred, data is lost	R	0
17	SFIFO_FULL	slave FIFO is full (1 if full)	R	0
16	SFIFO_EMPTY	slave FIFO is empty (1 if empty)	R	1
15	reserved	-	R	0
14:10	SFIFO_LEVEL	slave FIFO level (0..16)	R	0
9	MFIFO_ERR_UNDR	master FIFO underrun error occurred, data is lost	R	0

8	MFIFO_ERR_OVFL	master FIFO overflow error occurred, data is lost	R	0
7	MFIFO_FULL	master FIFO is full (1 if full)	R	0
6	MFIFO_EMPTY	master FIFO is empty (1 if empty)	R	1
5	reserved	-	R	0
4:0	MFIFO_LEVEL	master FIFO level (0..16)	R	0

I2C_INT_MSK_SET_CLR – I2C Interrupt Mask Set or Clear Register**0x1c000e20**

These bits have AND-mask character (only if mask bit is set, the appropriate IRQ generates the module IRQ). Enabling (writing '1' and prior mask was "0") will clear according raw IRQ-state if it was set before.

For detailed IRQ-description view I2C_RAW_INT_STAT.

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
reserved																								SREQ	SFIFO_REQ	MFIFO_REQ	BUS_BUSY	FIFO_ERR	CMD_ERR	CMD_OK	

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0
6	SREQ	Slave request interrupt mask.	R/W	0
5	SFIFO_REQ	Slave FIFO action request interrupt mask.	R/W	0
4	MFIFO_REQ	Master FIFO action request interrupt mask.	R/W	0
3	BUS_BUSY	External I2C-bus is busy interrupt mask.	R/W	0
2	FIFO_ERR	FIFO error interrupt mask.	R/W	0
1	CMD_ERR	Command error interrupt mask.	R/W	0
0	CMD_OK	Command OK interrupt mask.	R/W	0

I2C_RAW_INT_STAT – I2C RAW Interrupt Status Register**0x1c000e24**

I2C interrupt state register (raw interrupt before masking). Writing '1' will clear according IRQ.

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
reserved																								SREQ	SFIFO_REQ	MFIFO_REQ	BUS_BUSY	FIFO_ERR	CMD_ERR	CMD_OK	

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0
6	SREQ	Unmasked slave request interrupt state: Purpose: set up slave FIFO. 1: external master was running START-sequence and requested this slave. 0: slave is not requested.	R/W	0
5	SFIFO_REQ	Unmasked slave FIFO action request interrupt state: Purpose: slave FIFO should be updated. 1: slave FIFO request: i2c_stat.sfifo_level is above or below i2c_slave_fifo_ctrl.sfifo_wm (view description i2c_slave_fifo_ctrl). 0: slave FIFO state not critical	R/W	0
4	MFIFO_REQ	Unmasked master FIFO action request interrupt state: Purpose: master FIFO should be updated. 1: master FIFO request: i2c_stat.mfifo_level is above or below i2c_master_fifo_ctrl.mfifo_wm (view description i2c_master_fifo_ctrl). 0: master FIFO state not critical	R/W	0
3	BUS_BUSY	Unmasked external I2C-bus is busy interrupt state: Purpose: detect I2C-bus arbitration loss. 1: master did not gain requested bus access due to another master accessing the bus. 0: bus is idle or no transfer is requested by this master.	R/W	0
2	FIFO_ERR	Unmasked FIFO error interrupt state: Purpose: detect FIFO errors/transfer failures. 1: FIFO error occurred, check i2c_stat for details. 0: FIFOs ok.	R/W	0
1	CMD_ERR	Unmasked command error interrupt state: Purpose: check last command termination. 1: last command finished erroneous. 0: command not finished, no command or command finished successfully.	R/W	0
0	CMD_OK	Unmasked command OK interrupt state: Purpose: check last command termination. 1: last command finished successfully. 0: command not finished, no command or command finished erroneous.	R/W	0

I2C_MSK_INT_STAT – I2C Mask Interrupt Status Register**0x1c000e28**

Read only I2C masked interrupt state register.

If one of these bits is set, the I2C IRQ will be asserted to the Interrupt-Controller.

For detailed IRQ-description view I2C_RAW_INT_STAT.

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		
reserved																											SREQ	SFIFO_REQ	MFIFO_REQ	BUS_BUSY	FIFO_ERR	CMD_ERR	CMD_OK

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0
6	SREQ	Masked slave request interrupt state.	R	0
5	SFIFO_REQ	Masked slave FIFO action request interrupt state.	R	0
4	MFIFO_REQ	Masked master FIFO action request interrupt state.	R	0
3	BUS_BUSY	Masked external I2C-bus is busy interrupt state.	R	0
2	FIFO_ERR	Masked FIFO error interrupt state.	R	0
1	CMD_ERR	Masked command error interrupt state.	R	0
0	CMD_OK	Masked command OK interrupt state.	R	0

I2C_DMA_CTRL – I2C DMA Control Register**0x1c000e2c**

DMA transfer size to/from I2C-module: byte.

DMA burst length to/from I2C-module: 4.

DMA burst requests are generated if the according FIFO contains more than 4 bytes (receive case), or if there are more than 4 bytes writable to the according FIFO (transmit case).

DMA single transfer requests are generated if the according FIFO contains more than 1 byte (receive case), or if there is more than 1 byte writable to the according FIFO (transmit case).

No further DMA requests will be generated if all transmit data was written to the master FIFO and flow controlling is done by this module (for master data only). Once all data is written to the master FIFO last burst/single request is generated for the DMA controller.

If the DMA-Controller flags transfer end by setting DMACTC (terminal count) the appropriate bit will be cleared.

If one of the bits of this register is set to 0 by software and a DMA-transfer was requested before, one last transfer will be done by the DMA-Controller to reset DMA-request signals.

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
reserved																												SDMAB_EN	SDMAS_EN	MDMAB_EN	MDMAS_EN

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0
3	SDMAB_EN	Enable DMA burst requests for I2C slave data. Flowcontrolling must be done my DMA-Controller.	R/W	0
2	SDMAS_EN	Enable DMA single requests for I2C slave data. Flowcontrolling must be done my DMA-Controller.	R/W	0
1	MDMAB_EN	Enable DMA burst requests for I2C master data. Flowcontrolling may be done my DMA-Controller or by I2C-controller controlled by decrementing i2c_master_cmd.tsize.	R/W	0
0	MDMAS_EN	Enable DMA single requests for I2C master data. Flowcontrolling may be done my DMA-Controller or by I2C-controller controlled by decrementing i2c_master_cmd.tsize	R/W	0

6.7 CCDC – CCD Controller

The CCD module provides the following registers for configuration and data access:

ARM Address	Register Name	Short Description
0x1c000f00	CCDC_CFG	CCDC Config Register
0x1c000f04	CCDC_HORIZ_START_STOP	CCDC Horizontal Start/Stop Values
0x1c000f08	CCDC_VERT_START_STOP	CCDC Vertical Start/Stop Values
0x1c000f0c	CCDC_HORIZ_VERT_CNTR	CCDC Horizontal/Vertical Counter
0x1c000f10	CCDC_BRIGHT	CCDC Brightness Counter
0x1c000f14	CCDC_FIFO0	CCDC FIFO 0
0x1c000f18	CCDC_FIFO1	CCDC FIFO 1
0x1c000f1c	CCDC_FIFO2	CCDC FIFO 2
0x1c000f20	CCDC_BYTE0_POS	CCDC Byte 0 Position Register
0x1c000f24	CCDC_BYTE1_POS	CCDC Byte 1 Position Register
0x1c000f28	CCDC_BYTE2_POS	CCDC Byte 2 Position Register

CCDC_CFG - CCDC Config Register**0x1c000f00**

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		
reserved			LAST_MODE	FIFO_2_NXT	FIFO_2_UNFL	FIFO_2_OVFL	FIFO_2_RES	FIFO_1_NXT	FIFO_1_UNFL	FIFO_1_OVFL	FIFO_1_RES	FIFO_0_NXT	FIFO_0_UNFL	FIFO_0_OVFL	FIFO_0_RES	SAMPLE_TIME					BRIGHT_STEP										reserved	EDGE_MODE	ENABLE

Bits	Name	Description	R/W	Default
31:29	reserved	-	R	0x0
28	LAST_MODE	reserved	R/W	0x0
27	FIFO_2_NXT	value in fifo read only	R	0x0
26	FIFO_2_UNFL	underflow read only	R	0x0
25	FIFO_2_OVFL	overflow read only	R	0x0
24	FIFO_2_RES	1'b1 reset; 1'b0 release	R/W	0x0
23	FIFO_1_NXT	value in fifo read only	R	0x0
22	FIFO_1_UNFL	underflow read only	R	0x0
21	FIFO_1_OVFL	overflow read only	R	0x0
20	FIFO_1_RES	1'b1 reset; 1'b0 release	R/W	0x0
19	FIFO_0_NXT	value in fifo read only	R	0x0
18	FIFO_0_UNFL	underflow read only	R	0x0
17	FIFO_0_OVFL	overflow read only	R	0x0
16	FIFO_0_RES	1'b1 reset; 1'b0 release	R/W	0x0
15:12	SAMPLE_TIME	simple_time 0 - 15 100Mhz clocks after pos/negedge 0 : with edge 1 - 14 : 100 Mhz clocks after edge 15 : one 100Mhz before edge	R/W	0x0
11:4	BRIGHT_STEP	add every n (bright_step) byte: 0 no add 1 - 255 every n byte add	R/W	0x0
3:2	reserved	-	R	0x0
1	EDGE_MODE	0 : posedge (pixclk), 1 : negedge (pixclk)	R/W	0x0
0	ENABLE	enable CCD - Controller	R/W	0x0

CCDC_HORIZ_START_STOP - CCDC Horizontal Start/Stop Values**0x1c000f04**

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
HORIZ_STOP																HORIZ_START															

Bits	Name	Description	R/W	Default
31:16	HORIZ_STOP	byte count stop (exclusive), to cut picture	R/W	0x0
15:0	HORIZ_START	byte count start (inclusive), to cut picture	R/W	0x0

CCDC_VERT_START_STOP - CCDC Vertical Start/Stop Values**0x1c000f08**

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
VERTI_STOP																VERTI_START															

Bits	Name	Description	R/W	Default
31:16	VERTI_STOP	line count stop (exclusive), to cut picture	R/W	0x0
15:0	VERTI_START	line count start (inclusive), to cut picture	R/W	0x0

CCDC_HORIZ_VERT_CNTR - CCDC Horizontal/Vertical Counter**0x1c000f0c**

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
VERTI_COUNTER																HORIZ_COUNTER															

Bits	Name	Description	R/W	Default
31:16	VERTI_COUNTER	line count while frame valid, reset with posedge frame_valid	R	0x0
15:0	HORIZ_COUNTER	byte count while line_valid, reset with posedge line_valid	R	0x0

CCDC_BRIGHT - CCDC Brightness Counter**0x1c000f10**

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
BRIGHTNESS																															

Bits	Name	Description	R/W	Default
31:0	BRIGHTNESS	brightness adder, every bright_step byte, reset with posedge frame_valid	R	0x0

CCDC_FIFO0 - CCDC FIFO 0**0x1c000f14**

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
FIFO_0																															

Bits	Name	Description	R/W	Default
31:0	FIFO_0	2*4 bytes depth	R	0x0

CCDC_FIFO1 - CCDC FIFO 1**0x1c000f18**

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
FIFO_1																															

Bits	Name	Description	R/W	Default
31:0	FIFO_1	2*4 bytes depth	R	0x0

CCDC_FIFO2 - CCDC FIFO 2**0x1c000f1c**

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
FIFO_2																															

Bits	Name	Description	R/W	Default
31:0	FIFO_2	2*4 bytes depth	R	0x0

CCDC_BYTE0_POS - CCDC Byte 0 Position Register**0x1c000f20**

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
FIFO_0_AND_MASK								FOURTH_0_BYTE_ENABLE	FOURTH_0_BYTE_START				THIRD_0_BYTE_ENABLE	THIRD_0_BYTE_START				SECOND_0_BYTE_ENABLE	SECOND_0_BYTE_START				FIRST_0_BYTE_ENABLE	FIRST_0_BYTE_START							

Bits	Name	Description	R/W	Default
31:24	FIFO_0_AND_MASK	and_mask	R/W	0xff
23	FOURTH_0_BYTE_ENABLE	enable byte selection	R/W	0x0
22:18	FOURTH_0_BYTE_START	fourth byte start position	R/W	0x0
17	THIRD_0_BYTE_ENABLE	enable byte selection	R/W	0x0
16:12	THIRD_0_BYTE_START	third byte start position	R/W	0x0
11	SECOND_0_BYTE_ENABLE	enable byte selection	R/W	0x0
10:6	SECOND_0_BYTE_START	second byte start position	R/W	0x0
5	FIRST_0_BYTE_ENABLE	enable byte selection	R/W	0x0
4:0	FIRST_0_BYTE_START	first byte start position	R/W	0x0

CCDC_BYTE1_POS - CCDC Byte 1 Position Register**0x1c000f24**

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
FIFO_1_AND_MASK								FOURTH_1_BYTE_ENABLE	FOURTH_1_BYTE_START				THIRD_1_BYTE_ENABLE	THIRD_1_BYTE_START				SECOND_1_BYTE_ENABLE	SECOND_1_BYTE_START				FIRST_1_BYTE_ENABLE	FIRST_1_BYTE_START							

Bits	Name	Description	R/W	Default
31:24	FIFO_1_AND_MASK	and_mask	R/W	0xff
23	FOURTH_1_BYTE_ENABLE	enable byte selection	R/W	0x0
22:18	FOURTH_1_BYTE_START	fourth byte start position	R/W	0x0
17	THIRD_1_BYTE_ENABLE	enable byte selection	R/W	0x0
16:12	THIRD_1_BYTE_START	third byte start position	R/W	0x0
11	SECOND_1_BYTE_ENABLE	enable byte selection	R/W	0x0
10:6	SECOND_1_BYTE_START	second byte start position	R/W	0x0
5	FIRST_1_BYTE_ENABLE	enable byte selection	R/W	0x0
4:0	FIRST_1_BYTE_START	first byte start position	R/W	0x0

CCDC_BYTE2_POS - CCDC Byte 2 Position Register**0x1c000f28**

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
FIFO_2_AND_MASK								FOURTH_2_BYTE_ENABLE	FOURTH_2_BYTE_START				THIRD_2_BYTE_ENABLE	THIRD_2_BYTE_START				SECOND_2_BYTE_ENABLE	SECOND_2_BYTE_START				FIRST_2_BYTE_ENABLE	FIRST_2_BYTE_START							

Bits	Name	Description	R/W	Default
31:24	FIFO_2_AND_MASK	And_mask	R/W	0xff
23	FOURTH_2_BYTE_ENABLE	enable byte selection	R/W	0x0
22:18	FOURTH_2_BYTE_START	fourth byte start position	R/W	0x0
17	THIRD_2_BYTE_ENABLE	enable byte selection	R/W	0x0
16:12	THIRD_2_BYTE_START	Third byte start position	R/W	0x0
11	SECOND_2_BYTE_ENABLE	enable byte selection	R/W	0x0
10:6	SECOND_2_BYTE_START	second byte start position	R/W	0x0
5	FIRST_2_BYTE_ENABLE	enable byte selection	R/W	0x0
4:0	FIRST_2_BYTE_START	first byte start position	R/W	0x0

6.8 SYS_TIME – System time with IEEE 1588 functionality

The following table shows a summary of all registers related to system time generation

ARM Address	Register Name	Short Description
0x1c001100	SYS_TIME_NS	System Time Nanosecond Register
0x1c001104	SYS_TIME_S	System Time Second Register
0x1c001108	SYS_TIME_NS_BORDER	System Time Nanoseconds Border Register
0x1c00110c	SYS_TIME_NS_ADD_UP	System Time Nanoseconds Add Up Register
0x1c001110	SYS_TIME_S_CMP	System Time Second Compare Register
0x1c001114	SYS_TIME_S_CMP_EN	System Time Second Compare Enable Register
0x1c001118	SYS_TIME_S_CMP_INT	System Time Second Compare Interrupt Register

SYS_TIME_NS – System Time Nanosecond Register

0x1c001100

The SYS_TIME_NS register provides the lower part of the System time in nanoseconds. Used for counting nanoseconds according IEEE 1588.

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
SYS_TIME_NS																															

Bits	Name	Description	R/W	Default
31:0	SYS_TIME_NS	Lower part of the System time in nanoseconds.	R/W	0x00000000

SYS_TIME_S – System Time Second Register**0x1c001104**

The SYS_TIME_S register provides the upper part of the System time in seconds. Used for counting seconds according IEEE 1588.

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
SYS_TIME_S																															

Bits	Name	Description	R/W	Default
31:0	SYS_TIME_S	Higher part of the system time in seconds. Value is incremented if SYS_TIME_NS reaches SYS_TIME_NS_BOR.	R/W	0x00000000

SYS_TIME_NS_BOR – System Time Nanoseconds Border Register**0x1c001108**

The SYS_TIME_NS_BOR register configures the border for the lower part of the system time.

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
SYS_TIME_NS_BOR																															

Bits	Name	Description	R/W	Default
31:0	SYS_TIME_NS_BOR	SYS_TIME_NS counts from 0 to this value (including this value). I.e. SYS_TIME_NS counts modulo (SYS_TIME_NS_BOR + 1). Default value corresponds to 1 second.	R/W	0x3b9ac9ff

SYS_TIME_NS_ADD_UP – System Time Nanoseconds Add Up Register**0x1c00110c**

The SYS_TIME_NS_ADD_UP register provides the value which is add up to SYS_TIME_NS register per clock cycle.

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
SYS_TIME_NS_ADD_UP																															

Bits	Name	Description	R/W	Default
31:0	SYS_TIME_NS_ADD_UP	Each clock cycle (SYS_TIME_NS_ADD_UP >> 28) will be added to System time. I.e. value 0x10000000 can be used for counting in 10ns steps. Default value corresponds to 1 ns per step.	R/W	0xa0000000

SYS_TIME_S_CMP – System Time Second Compare Register**0x1c001110**

The SYS_TIME_S_CMP register provides the compare value to SYS_TIME_S.

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
SYS_TIME_S_CMP_VAL																															

Bits	Name	Description	R/W	Default
31:0	SYS_TIME_S_CMP_VAL	Compare value with SYS_TIME_S (seconds). Set SYS_TIME_S_CMP_INT register if SYS_TIME_S_CMP_EN is set.	R/W	0x00

SYS_TIME_S_CMP_EN – System Time Second Compare Enable Register**0x1c001114**

The SYS_TIME_S_CMP_EN register enables or disables the comparison to SYS_TIME_S.

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
reserved																															ENABLE

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x00
0	ENABLE	Enable compare with SYS_TIME_S (seconds). Automatic set with the value in register SYS_TIME_S_CMP, automatic reset after interrupt.	R/W	0

SYS_TIME_S_CMP_INT – System Time Second Compare Interrupt Register**0x1c001118**

The SYS_TIME_S_CMP_INT register signals an interrupt because SYS_TIME_S is equal SYS_TIME_S_CMP.

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
reserved																															COMPARE_IRQ

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x00
0	COMPARE_IRQ	System time compare interrupt. Set if SYS_TIME_S is equal SYS_TIME_S_CMP. To clear the IRQ you have to write a '1' at this bit position.	R/W	0

6.9 MMIO – Multiplex Matrix IOs

See chapter 2.3 (IO Configuration) for MMIO Configuration options

6.10 USB – Serial USB-Interface

The following table shows a summary of all USB registers:

ARM Address	Register Name	Short Description
0x1c020000	USB_ID	USB ID Register
0x1c020004	USB_CTRL	USB Control Register
0x1c020008	USB_FRM_TMR	USB Frame Timer Register
0x1c02000c	USB_MAIN_EV	USB Main Event Register
0x1c020010	USB_MAIN_EV_MSK	USB Main Event Mask Register
0x1c020014	USB_PIPE_EV	USB Pipe Event Register
0x1c020018	USB_PIPE_EV_MSK	USB Pipe Event Mask Register
0x1c020024	USB_PIPE_SEL	USB Pipe Select Register
0x1c02002c	USB_PORT_STAT	USB Port Status Register
0x1c020030	USB_PORT_CTRL	USB Port Control Register
0x1c020034	USB_PORT_STAT_CHG_EV	USB Port Status Change Event Register
0x1c020038	USB_PORT_STAT_CHG_EV_MSK	USB Port Status Change Event Mask Register
0x1c020040	USB_PIPE_CTRL	USB Pipe Control Register
0x1c020044	USB_PIPE_CFG	USB Pipe Configuration Register
0x1c020048	USB_PIPE_ADDR	USB Pipe Address Register
0x1c02004c	USB_PIPE_STAT	USB Pipe Status Register
0x1c020050	USB_PIPE_DATA_PTR	USB Pipe Data Pointer Register
0x1c020054	USB_PIPE_DATA_TOT	USB Pipe Total Bytes Register
0x1c020058	USB_PIPE_ALT_DATA_PTR	USB Pipe Alternative Data Pointer Register
0x1c02005c	USB_PIPE_ALT_DATA_TOT	USB Pipe Alternative Data Total Bytes Register
0x1c020060	USB_DBG_CTRL	USB Debug Control Register
0x1c020064	USB_DBG_PID	USB Debug PID Register
0x1c020068	USB_DBG_STAT	USB Debug Status Register
0x1c02006c	USB_TEST	USB Test Register
0x1c020080	USB_MAIN_CFG	USB Main Configuration Register
0x1c020084	USB_MODE_CFG	USB Mode Configuration Register
0x1c020088	USB_CORE_CTRL	USB Core Control and Status Register
0x1c030000	USB_FIFO	FIFO for USB-Interface

USB_ID – USB ID Register**0x1c020000**

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
reserved																			CORE_ID				REV_ID								

Bits	Name	Description	R/W	Default
31:13	reserved	-	R	0x00
12:8	CORE_ID	Core-ID This bitfield shows the unique ID of the core.	R	0x07
7:0	REV_ID	Revision ID These bits show the revision number of the core.	R	0x00

USB_CTRL – USB Control Register**0x1c020004**

This register is only valid if the core works in host mode.

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
reserved																										PSE	ASE	HRS	XSUSP	CSUSP	

Bits	Name	Description	R/W	Default
31:5	reserved	-	R	0x00
4	PSE	Periodic Schedule Enable When set to 1 and the Host Run/Stop Bit is set, the core schedules periodic transactions (interrupt and isochronous). If software sets this bit to zero all pending periodic transactions of the current frame will be completed.	R/W	0
3	ASE	Async Schedule Enable When set to 1 and the Host Run/Stop Bit is set, the core schedules asynchronous transactions (bulk and control). If software sets this bit to zero, all pending asynchronous transactions of the current frame will be completed.	R/W	0
2	HRS	Host Run/Stop When set to 1, the core starts framework and transaction scheduling. If software sets this bit to zero, all pending transactions of the current frame will be completed, and the Host Controller Halted. Event Bit is set at the end of the frame, and no more SOFs will be sent. The bit will be cleared by hardware if a serious error was detected.	R/W	0
1	XSUSP	XCVR Suspend This bit controls the xcvr suspend n output signal. If this bit is set to 1, the low-active xcvr suspend n output will be set to 0 to enable the low-power mode of the connected Transceiver. Any change at the ser_rx_dm or ser_rx_dp signal will force the deactivation of the xcvr_suspend_n output signal. (Returns to 1)	R/W	0
0	CSUSP	Core Suspend This bit controls the core suspend n output signal. If this bit is set to 1, the low-active core suspend n output will be set to 0 to enable the low-power mode of this core. This signal should be connected to an external clock controller. Any change at the signals ser_rx_dm or ser_rx_dp will force the deactivation of the core suspend n output signal. (Returns to 1)	R/W	0

USB_FRM_TMR – USB Frame Timer Register**0x1c020008**

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
reserved																			AGSOF	FTLOCK	FRAME_NR										

Bits	Name	Description	R/W	Default
31:13	reserved	-	R	0x00
12	AGSOF	Artificially Generated SOF This field is only valid, when the core works in peripheral mode. This bit is set, if the Frame Event Bit of the Main Event Register was generated artificially in the case of a lost SOF token.	R	0
11	FTLOCK	Frame Timer Locked This field is only valid, when the core works in peripheral mode. It indicates that the internal frame timer is locked. It will be set if more than two consecutive SOF tokens are received, and will be cleared if at least two consecutive SOF tokens are missed.	R	0
10:0	FRAME_NR	Current Frame Number This field shows the current frame number. If the core works in host mode the frame timer is a free running counter that will be incremented every 1 ms. This register cannot be written unless the Host Controller is in the Halted state. If the core works in peripheral mode the frame timer will synchronize to incoming SOF tokens. This field shows the frame number of the current received SOF token.	R/W	0x00

USB_MAIN_EV – USB Main Event Register**0x1c02000c**

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
reserved																										BWERR_EV	HCHA_EV	GPIPE_EV	GPORT_EV	FRM32_EV	FRM_EV

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	BWERR_EV	Bandwidth Error Event This field is only valid, when the core works in host mode. This bit will be set by hardware if no sufficient bandwidth is available to complete all periodic transfers in a frame. It leads to an interrupt, if the Bandwidth Error Event Mask Bit of the Main Event Mask Register is set. Write a 1 in this position to clear this bit.	R/W	0
4	HCHA_EV	Host Controller Halted Event This field is only valid, when the core works in host mode. The bit is set, if the host controller has stopped as a result of setting the Host Run/Stop Bit to the halted state and all pending transactions of a frame are completed. It leads to an interrupt, if the corresponding HC Halted Mask Bit of the Main Event Mask Register was set. Write a 1 in this position to clear this bit.	R/W	0
3	GPIPE_EV	Global Pipe Transfer Event The bit is set if one of the bits of the Pipe Event Register is set. It leads to an interrupt, if the Global Pipe Event Mask Bit of the Main Event Mask Register is set.	R	0
2	GPORT_EV	Global Port Status Change Event The bit is set if one of the bits of the Port Status Change Event Register is set. It leads to an interrupt, if the Global Port Status Change Event Mask Bit of the Main Event Mask Register is set.	R	0
1	FRM32_EV	Frame 32 Event The bit is set after every 32nd SOF token. It leads to an interrupt, if the corresponding Frame 32 Event Mask Bit of the Main Event Mask Register was set. Can be used by software timers. Write a 1 in this position to clear this bit.	R/W	0
0	FRM_EV	Frame Event The bit is set every 1 ms. It leads to an interrupt, if the Frame Event Mask Bit of the Main Event Mask Register was set. Write a 1 in this position to clear this bit.	R/W	0

USB_MAIN_EV_MSK – USB Main Event Mask Register**0x1c020010**

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
reserved																										BWERR_EM	HCHA_EM	GPIPE_EM	GPORT_EM	FRM32_EM	FRM_EM

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	BWERR_EM	Bandwidth Error Event Mask Setting this bit will lead to an interrupt if the Bandwidth Error Event Bit is active.	R/W	0
4	HCHA_EM	HC Halted Mask Setting this bit will lead to an interrupt if the Host Controller Halted Event Bit is active.	R/W	0
3	GPIPE_EM	Global Pipe Event Mask Setting this bit will lead to an interrupt if the Global Pipe Transfer Event Bit is active.	R/W	0
2	GPORT_EM	Global Port Status Change Event Mask Setting this bit will lead to an interrupt if the Global Port Status. Change Event Bit is active.	R/W	0
1	FRM32_EM	Frame 32 Event Mask Setting this bit will lead to an interrupt if the Frame 32 Event Bit is active.	R/W	0
0	FRM_EM	Frame Event Mask Setting this bit will lead to an interrupt if the Frame Event Bit is active.	R/W	0

USB_PIPE_EV – USB Pipe Event Register**0x1c020014**

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
reserved																								PIPE_EV							

Bits	Name	Description	R/W	Default
31:8	reserved	-	-	-
7:0	PIPE_EV	Pipe Event Flag for each of the eight pipes. If the core works in host mode this event is triggered by the core if one of the following conditions apply: The pipe goes inactive and the Active Status Bit of the Pipe Status Register was cleared by hardware because of normal termination or an error event. A data buffer was completed: the Data Buffer Valid Bit or the Alternative Data Buffer Valid Bit was cleared by hardware. If the core works in peripheral mode this event is triggered by the core if a data buffer was completed: the Data Buffer Valid Bit or the Alternative Data Buffer Valid Bit was cleared by hardware. Write a 1 in this position to clear this bit.	R/W	0x00

USB_PIPE_EV_MSK – USB Pipe Event Mask Register**0x1c020018**

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
reserved																								PIPE_EM							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	PIPE_EM	Pipe Event Mask Setting one of these bits enables the pipe event interrupt generation for the appropriated pipe.	R/W	0x00

USB_PIPE_SEL – USB Pipe Select Register**0x1c020024**

The physical register space for the pipe configuration and status register is paged by the 'USB Pipe Select Register'.

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	
reserved																															PIPE_SEL	

Bits	Name	Description	R/W	Default
31:5	reserved	-	R	0x00
4:0	PIPE_SEL	Pipe Select Depending on the number of supported pipes the core is configured to, one of the pipes may be selected using this register for access of the appropriate status and configuration registers. The width of this bitfield depends on the parameter number of pipes and is equal to log2(NOP). This bitfield is coded as follows: 000 : Pipe 0 : 111 : Pipe 7	R/W	0x00

USB_PORT_STAT – USB Port Status Register**0x1c02002c**

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
reserved																						LINESTATE	OCURC	DLS	PCS	CONN_ID	VB_SESS_END	VB_SESS_VLD	VA_SESS_VLD	VBUS_VLD	

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9:8	LINESTATE	USB Line State This bit shows the current line state: LINESTATE[1:0] = [ser_rx_dp, ser_rx_dm]	R	0x00
7	OCURC	Over Current Condition This bit will be set to one if the core has detected an overcurrent condition, other-wise it is zero. An overcurrent condition is defined as vb_vld is deasserted but vb on is set.	R	0
6	DLS	Device Low Speed This field is only valid if the core works in host mode. This bit indicates, if the connected device is a full or low speed device. The bit is only valid if the Port Connect Status Bit indicates a connected device. 0 : Full speed device connected 1 : Low speed device connected	R	0
5	PCS	Port Connect Status This field is only valid if the core works in host mode. This bit indicates, if a device is connected. 0 : No device attached [label = "PCS DISCONN"] 1 : Device attached to root port [label = "PCS CONN"]	R	0
4	CONN_ID	USB Connector ID Value This bit shows the ID value of the USB connector. If set to one, it indicates that the core works as B-Device, with initial peripheral mode. If set to zero, the core will work as A-Device, with initial host mode. The core can change the host/peripheral roles using the Host Negotiation Protocol (HNP). Therefore, the real mode of the core depends on this bit in conjunction with the selected termination. (refer the Termination Select Bit). The core will update this bit, when a falling edge is detected at the ID-Pullup Enable Bit! 0 : A-Device (initial host) [label = "CONN ID A"] 1 : B-Device (initial peripheral) [label = "CONN ID B"]	R	0
3	VB_SESS_END	VB Session End Hardware will set this bit to one if the vb_sess_end input is set, indicating VBUS is above VB-SESS-END.	R	0
2	VB_SESS_VLD	VB Session Valid Hardware will set this bit to one if the vb_sess_vld input is set, indicating VBUS is above VB-SESS-VLD.	R	0
1	VA_SESS_VLD	VA Session Valid Hardware will set this bit to one if the va_sess_vld input is set, indicating VBUS is above VA-SESS-VLD.	R	0
0	VBUS_VLD	Vbus Valid Hardware will set this bit to one if the vb_vld input is set, indicating VBUS is above 4.4 V. For OTG Devices this bit is set to one, when Vbus is above VA-VBUS-VLD. When Vbus goes below this threshold an overcurrent condition did occur.	R	0

USB_PORT_CTRL – USB Port Control Register

0x1c020030

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
reserved								P_LEN								reserved			ID_PU	reserved	VBS_ON	DCHRG	TERM_ENA	TERM_SEL	VB_ON	PSUSP	PENA	FPRESU	URESET	PTESTC	

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x00
23:16	P_LEN	<p>Pulse Length It defines the pulse length in milliseconds if the core wants to request a session using the data line pulsing or VBUS pulsing method. Setting this bitfield to zero allows software controlled pulse length. If set to a non-zero value, this field will be updated every millisecond by hardware.</p> <p>0x00 : Software controlled pulse length. Software must reset the VBUS Session Request Control Bit or the Termination Enable Bit manually.</p> <p>0x01 : 1 ms pulse length. Hardware will reset the VBUS Session Request Control Bit and the Termination Enable Bit automatically after 1 ms.</p> <p>:</p> <p>0xFF : 255 ms pulse length. Hardware will reset the VBUS Session Request Control Bit and the Termination Enable Bit automatically after 255 ms. If the this bitfield counts down from one to zero, the Pulse End Event Bit of the Main Event Register will be set.</p>	R/W	0x00
15:13	reserved	-	R	0x00
12	ID_PU	<p>ID-Pullup Enable Software can control the ID-pullup output signal (id pullup) using this bit. The core will update the status of the USB Connector ID Value Bit only if this bit becomes inactive.</p>	R/W	0
11	reserved	-	R	0x00
10	VBS_ON	<p>VBUS Session Request Control Software can control the power circuitry (chrg_vbus) using this bit. If set to 1, the vbs_on output will be set.</p>	R/W	0
9	DCHRG	<p>Enable Discharge Circuitry Software can control the discharge circuitry using this bit to accelerate the discharge of the host bus power capacitors. Setting this bit to 1 will lead to setting the dischrg_vbus output.</p>	R/W	0
8	TERM_ENA	<p>Termination Enable Software can enable the data line termination at the OTG port using this bit. If set to 0, the rpu_ena and rpd_ena outputs are tied to zero. Software can set this bit after it has detected the following conditions after power on reset:</p> <ul style="list-style-type: none"> The ID value indicates the core must work as B-Device and is initially in peripheral mode and valid bus power was detected The ID value indicates the core must work as A-Device <p>Software can set this bit to switch the host/peripheral role following the Host Negotiation Protocol. The pull-up resistor can also be switched on if software wants to request a session using data line pulsing (according to Session Request Protocol). This requires that the VBUS Control Bit and the VBUS Session Request Control Bit must indicate the powered off state, and will also lead to setting the dlp_active output.</p>	R/W	0

7	TERM_SEL	<p>Termination Select</p> <p>Software can select the data line termination at the port using this bit:</p> <p>0 : Host termination. If the Termination Enable Bit is set, this will lead to setting rpd_ena = 1 and rpu_ena = 0. [label = "TERM SEL HOST"]</p> <p>1 : Device termination. If the Termination Enable Bit is set, this will lead to setting rpu_ena = 1 and rpd_ena = 0. [label = "TERM SEL DEV"]</p> <p>Software can set this bit after it has detected the following conditions after power on reset:</p> <ul style="list-style-type: none"> • The ID value indicates the core must work as B-Device and is initially in peripheral mode • Valid bus power was detected <p>Software can set this bit to switch the host/peripheral role following the Host Negotiation Protocol.</p>	R/W	0
6	VB_ON	<p>VBUS Control</p> <p>Software can control the power circuitry (vb_on output) using this bit. If set to 1, the vb_on output will be set.</p> <p>Note: Software must set this bit to be able to detect a connected device!</p>	R/W	0
5	PSUSP	<p>Port Suspend</p> <p>This field is only valid if the core works in host mode. The bit can be set by software if the device driver wants to set the connected device into suspended state. Setting the bit will block data propagation on the root port. If the bit is set, the core is sensitive to resume detection. The bit must also be set to prepare the core for changing the host/peripheral role. (Host Negotiation Protocol) Setting the Force Resume Bit will automatically clear this bit.</p>	R/W	0
4	PENA	<p>Port Enable</p> <p>This field is only valid if the core works in host mode. The bit will be set automatically by hardware after the software has finished the USB reset. The core will start sending SOF tokens if the connected device is working at full speed, or low speed keepalive strobes if the connected device is working at low speed.</p>	R/W	0
3	FPRESU	<p>Force Resume</p> <p>If the core works in host mode:</p> <p>The bit can be set by software if, the connected device has to finish its suspend mode. The bit will be set by hardware, if a Remote Wakeup condition was detected. (Resume Event Bit is set). In that case, software must finish the resume sequence by setting this bit to zero. The bit will remain set until the resume sequence was completed by hardware by a low speed EOP. (indicated by Resume Complete Event Bit)</p> <p>If the core works in peripheral mode:</p> <p>The bit can be set by software if it wants to wake up the host/hub it is connected (Remote Wakeup). Software is responsible for the timing of the remote wakeup resume, and must set the bit for at least 1 ms, but no more than 15 ms.</p>	R/W	0
2	URESET	<p>USB Reset</p> <p>This field is only valid if the core works in host mode. When software writes a one to this bit, an USB reset sequence (SE0 state) will be started by the core. Writing a zero to this bit will terminate the reset sequence. There may be a delay before the bit status changes to zero, because hardware will clear the bit if it has completed the reset sequence. Software must keep the bit at one long enough as defined in the USB Specification 2.0, chapter 7.3.2. (10..20 ms)</p> <p>0 : No USB reset driven. [label = "URESET INACTIVE"]</p> <p>1 : Core drives USB reset. [label = "URESET ACTIVE"]</p>	R/W	0

1:0	PTESTC	Port Test Control This bitfield allows to set the port into specific test modes. When the field is zero, the port is not operating in test mode. The encoding of test mode is: 00 : Test mode disabled [label = "PTESTC DIS"] 01 : Test J State [label = "PTESTC JST"] 10 : Test K State [label = "PTESTC KST"] 11 : Test SE0 [label = "PTESTC SE0"]	R/W	0x00
-----	--------	--	-----	------

USB_PORT_STAT_CHG_EV – USB Port Status Change Event Register**0x1c020034**

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
reserved																							P_END_EV	PWRSC_EV	CDC_EV	URES_EV	SUSP_EV	RSUC_EV	RSU_EV	BERR_EV	OCU_EV

Bits	Name	Description	R/W	Default
31:9	-	-	-	-
8	P_END_EV	Pulse End Event This field is only valid, when the core works in peripheral mode. The bit is set if the Pulse Length Field changes from one to zero. Write a 1 in this position to clear this bit.	R/W	0
7	PWRSC_EV	Power Status Change Event The bit is set if one of the following bits has changed: • Vbus Valid Bit • VA Session Valid Bit • VB Session Valid Bit • VB Session End Bit Write a 1 in this position to clear this bit.	R/W	0
6	CDC_EV	Connect/Disconnect Event This field is only valid if the core works in host mode. If set to one, this bit indicates that the Port Connect Status Bit value has changed. Write a 1 in this position to clear this bit.	R/W	0
5	URES_EV	USB Reset Event If the core works in host mode: After software has completed the USB reset by clearing the USB Reset Bit bit, this event bit will be set after the reset was completed by hardware. (The core continues driving the reset after URESET is deasserted until the next Start Of Frame) If the core works in peripheral mode: This bit is set if an USB reset condition was detected. (data lines more than 2,5 µs SE0). Write a 1 in this position to clear this bit.	R/W	0
4	SUSP_EV	Suspend Event This bit is only valid if the core works in peripheral mode. This bit indicates a lack of bus activity for more than 3 ms. Software can use this event to activate a low power mode. Note: If the low power mode causes that the clock of the USB Core is removed, the bounding hardware is responsible to turn on the clock controller if a state other than IDLE state at the USB data lines is asserted. This requires some additional, asynchronous circuitry. If HNP was not enabled previously, software shall set the Port Suspend Bit to be sensitive for resume signaling, and enter the low power state. If HNP was enabled, software can now switch to host mode according to the HNP rules. Write a 1 in this position to clear this bit.	R/W	0

3	RSUC_EV	<p>Resume Complete Event</p> <p>If the core works in host mode: This bit will be set after the core has completed the resume sequence with a low speed EOP.</p> <p>If the core works in peripheral mode: This bit is set if the resume sequence was completed by the host with a low speed EOP.</p> <p>Write a 1 in this position to clear this bit.</p>	R/W	0
2	RSU_EV	<p>Resume Event</p> <p>If the core works in host mode: This bit is set if the host driver has set the port in suspended state, and a resume condition driven by the connected device was detected. (remote wakeup)</p> <p>If the core works in peripheral mode: This bit is set if resume condition was detected (more than 2.5 μs K-State).</p> <p>Write a 1 in this position to clear this bit.</p>	R/W	0
1	BERR_EV	<p>Babble Error Event</p> <p>This field is only valid, when the core works in host mode. this bit gets only to a one when the root port is disabled due to the appropriate conditions existing at the EOF2 point (/USB-Spec20/ chapter 11.2.5).</p> <p>Write a 1 in this position to clear this bit.</p>	R/W	0
0	OCU_EV	<p>Over Current Event</p> <p>This bit will be set to one if the core has detected an overcurrent condition. (refer the Over Current Condition Bit)</p> <p>Write a 1 in this position to clear this bit.</p>	R/W	0

USB_PORT_STAT_CHG_EV_MSK – USB Port Status Change Event Mask Register 0x1c020038

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
reserved																							P_END_EM	PWRSC_EM	CDC_EM	URES_EM	SUSP_EM	RSUC_EM	RSU_EM	BERR_EM	OCU_EM

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x00
8	P_END_EM	Pulse End Event Mask Setting this bit will lead to an interrupt if the Pulse End Event Bit is active.	R/W	0
7	PWRSC_EM	Power Status Change Event Mask Setting this bit will lead to an interrupt if the Power Status Change Event Bit is active.	R/W	0
6	CDC_EM	Connect/Disconnect Event Mask Setting this bit will lead to an interrupt if the Connect/Disconnect Event Bit is active.	R/W	0
5	URES_EM	USB Reset Event Mask Setting this bit will lead to an interrupt if the USB Reset Event Bit is active.	R/W	0
4	SUSP_EM	Suspend Event Mask Setting this bit will lead to an interrupt if the Suspend Event Bit is active.	R/W	0
3	RSUC_EM	Resume Complete Event Mask Setting this bit will lead to an interrupt if the Resume Complete Event Bit is active.	R/W	0
2	RSU_EM	Resume Event Mask Setting this bit will lead to an interrupt if the Resume Event Bit is active.	R/W	0
1	BERR_EM	Babble Error Event Mask Setting this bit will lead to an interrupt if the Babble Error Event Bit is active.	R/W	0
0	OCU_EM	Over Current Event Mask Setting this bit will lead to an interrupt if the Over Current Event Bit is active.	R/W	0

Pipe Register Group

This section defines the interface data structures, which will be used to communicate control/status information and data between the software and the hardware.

If the core works in peripheral mode:

Pipe[0] is reserved to handle control transfers. No other pipe can be used for control transfer. Because the control pipe is defined as bidirectional pipe, the core is sensitive to tokens for both directions. A SETUP transaction will always be accepted with an ACK handshake as required by the USB Specification. If no data buffer is available, the default response to OUT or IN tokens is NAK. All pipe registers will be set to their reset values after the core has detected an USB reset.

USB_PIPE_CTRL – USB Pipe Control Register

0x1c020040

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
reserved																													ACT	TPID	

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x00
2	ACT	<p>Activate Pipe</p> <p>If the core works in host mode: Setting this bit will initiate one or more transactions with the given configuration for pipe. The bit will be cleared automatically if the transfer is finished or an error condition has occurred. Data transfer is only initiated, if valid buffer space is provided by software.</p> <p>If the core works in peripheral mode: Setting this bit will enable the response to an endpoint with the given pipe parameters. If no valid buffer space is provided, the endpoint will respond with a NAK handshake. The bit will be set automatically for pipe[0] after an USB reset was detected. It will be automatically cleared by hardware for all other pipes after the detection of an USB reset.</p>	R/W	0
1:0	TPID	<p>Token PID/Direction</p> <p>If the core works in host mode: This field indicates the PID (and direction) used for the token to be sent.</p> <p>If the core works in peripheral mode: This field indicates the direction of data flow for this endpoint. For receive direction, the bitfield must be set to TPID IN. For transmit direction, the bitfield must be set to TPID OUT. The bitfield will be automatically set to TPID SETUP for pipe[0] after a SETUP token was received.</p> <p>00 : OUT transfer [label = "TPID_OUT"] 01 : IN transfer [label = "TPID_IN"] 10 : SETUP transfer [label = "TPID_SETUP"] 11 : Reserved.</p>	R/W	0

USB_PIPE_CFG – USB Pipe Configuration Register**0x1c020044****Access Rules:**

If the core works in host mode:

The contents of this register must not be changed when the Activate Pipe Bit or the Active Status Bit is one.

If the core works in peripheral mode:

The contents of this register (except the Stall Pipe Bit) must not be changed when the Activate Pipe Bit is one.

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
reserved	IOT	HIDBE	SKIPISO	PI				POFF				STALL				ACID	EPS	STRM	ET		MPS										

Bits	Name	Description	R/W	Default
31	reserved	-	R	0
30	IOT	Interrupt on Transaction. This field is only valid if the core works in host mode. If this bit is set, the core will assert an interrupt after every performed transaction on this pipe, independently on the transaction status. That means, also for NAKed transactions or after an error condition (timeout, CRC-error) an interrupt will be generated. Setting this bit is only useful if the transaction scheduling must be performed by software.	R/W	0
29	HIDBE	Halt on ISO Data Buffer Error This field is only valid if the core works in host mode. If the Skip ISO Token Bit is not set, and the core sends an IN token for isochronous IN pipes, and there is no buffer available, the core will halt this pipe, when the HIDBE bit is activated. If HIDBE is deactivated, the core will not change the pipe state.	R/W	0
28	SKIPISO	Skip ISO Token This field is only valid if the core works in host mode. When this bit is set, the HC will send no IN/OUT token for isochronous pipes, when no data buffer space is available. In the case, the IN token is send (the Skip ISO Token Bit is 0) and there is no bufferspace the HC will not write any data to the RAM. The received data is lost. For OUT pipes the HC will send an OUT-DATA0 sequence, when no buffer space is available.	R/W	0
27:24	PI	Polling Interval This field is only valid if the core works in host mode. This value defines the polling interval for this pipe as 2 ^{PI} , if the endpoint is configured as an interrupt or isochronous endpoint. The polling intervals for full/low speed interrupt endpoints is encoded as follows: 0000 : 1 frame 0001 : 2 frames 0010 : 4 frames : 1000 : 256 frames PI must be less/equal 8. Setting PI to a value greater 8 will lead to undefined results.	R/W	0x00
23:16	POFF	Polling Offset This field is only valid if the core works in host mode and for interrupt endpoints. Defines the polling interval frame offset the pipe is scheduled for. The pipe is polled if PI OFF[PI-1:0] matches the current frame number slice FRAME NR[PI-1:0]. (8 >= PI >= 1)	R/W	0x00
15	STALL	Stall Pipe	R/W	0

		This bit is only valid if the core works in peripheral mode. Setting this bit will lead to a STALL response to an endpoint with the given pipe parameters. The bit will be automatically cleared for pipe[0] after an USB reset was detected or a SETUP token was received.		
14	ACID	Accept corrupted ISO Data When this bit is set, the core accepts also corrupted data, transferred by isochronous receive pipes. Packets with CRC error are accepted completely. Packets with bitstuff error are accepted until the bitstuff error occurs.	R/W	0
13	EPS	Endpoint Speed This field is only valid if the core works in host mode. Indicates the speed of the function with this endpoint. 0 : Full speed [label = "EPS FULL"] 1 : Low speed [label = "EPS LOW"] If the device connected to the root port is a full speed, and the bit is set to low speed, a PRE token will be generated automatically.	R/W	0
12	STRM	Streaming Mode This bit is only valid for receive pipes/endpoints. It should be used for high bandwidth periodic receive pipes/endpoints. If this bit is activated, the streaming mode is used for this pipe: • The data received by several bus transactions can be stored in one buffer. • The inter buffer byte alignment is done, so that different buffers can be concatenated without shifting the entire data by 1 or more bytes. If this bit is deactivated for this pipe: • The data of one transaction for this pipe is stored in a separate buffer • No inter buffer byte alignment is done.	R/W	0
11:10	ET	Endpoint Type Indicates the transfer type for the transactions of this pipe/endpoint: 00 : Control Transfer [label = "ET CTRL"] 01 : Isochronous Transfer [label = "ET ISO"] 10 : Bulk Transfer [label = "ET BULK"] 11 : Interrupt Transfer [label = "ET INT"]	R/W	0x00
9:0	MPS	Maximum Packet Size This field defines the maximum number of bytes per data packet that can be sent to or received from the endpoint of this pipe. If the core works in host mode: If more bytes are received, and the core is configured for host mode, a babble error will be generated. If the core works in peripheral mode: The bitfield will be ignored if the received transaction was a SETUP transaction. (SETUP transactions must always be accepted)	R/W	0

USB_PIPE_ADDR – USB Pipe Address Register**0x1c020048****Access Rules:**

If the core works in host mode:

The contents of this register must not be changed when the Activate Pipe Bit or the Active Status Bit is one.

If the core works in peripheral mode:

The contents of this register (except the Stall Pipe Bit) must not be changed when the Activate Pipe Bit is one.

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
reserved												EPADDR						ERNR													

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x00
10:4	EPADDR	Endpoint Address This field defines the USB device address of this pipe. It must be set for both host and peripheral mode. If working in peripheral mode, this field defines the device address. Software must set this field only for pipe[0] after the successful completion of a SetAddress-request. The value of this field of pipe[0] is used for all other pipes. It will be automatically reset to 0x00 if an USB reset was detected.	R/W	0x00
3:0	ERNR	Endpoint Number It specifies the endpoint number of this pipe/endpoint. This field must be set for both host and peripheral mode. If the core works in peripheral mode: The bitfield will be automatically set to 0x0 for pipe[0].	R/W	0x00

USB_PIPE_STAT – USB Pipe Status Register**0x1c02004c**

This register contains the complete status information of the selected pipe and can be changed by hardware.

Access Rules:

If the core works in host mode:

The contents of this register must not be changed when the Activate Pipe Bit or the Active Status Bit is one.

If the core works in peripheral mode:

The contents of this register (except the Stall Pipe Bit) must not be changed when the Activate Pipe Bit is one.

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					
reserved																						CERR	DBERR	ACTS	HALT	BBL	DBSEL	DT	DBOFF							

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9:8	CERR	Error Counter This field is only valid if the core works in host mode. Shows the status of a 2-bit down counter that, keeps track of the number of consecutive errors detected. The core decrements the counter if the transaction fails. If the counter counts from one to zero, the pipe will be halted and sets the Pipe Halted Bit. When the Activate Pipe Bit of the Pipe Control Register goes from 0x0 to 0x1, the core resets this bitfield.	R	0x3
7	DBERR	Data Buffer Error This field is only valid if the core works in host mode. Set to a 1 during the status update to indicate that the core is unable to keep up with the reception of incoming data (overflow). This bit is only valid for isochronous IN pipes. For all other kinds of transfers the core ensures, that there is enough buffer space for at least 1 packet with maximum payload size. When the Activate Pipe Bit of the Pipe Control Register goes from 0x0 to 0x1, the core resets this bit.	R	0
6	ACTS	Active Status This field is only valid if the core works in host mode. This bit shows the current status of the pipe. It will be set, after the Activate Pipe Bit was set to 1. It will be cleared, after the Activate Pipe Bit was set to 0, or the pipe was halted due to several transfer events (pipe halted, babble, short packet for control/bulk pipes etc.).	R	0
5	HALT	Pipe Halted This field is only valid if the core works in host mode. Set to a 1 by the core during status update if the core works in host mode to indicate that a serious error has occurred at the device/endpoint. This can be caused by babble errors, the error counter counting down to zero, or reception of the STALL handshake from the device. Any time that a transaction results in the this bit being set to a one, the Activate Pipe Bit is also set to 0. When the Activate Pipe Bit of the Pipe Control Register goes from 0x0 to 0x1, the core resets this bit.	R	0
4	BBL	Babble detected This field is only valid if the core works in host mode. Set to a 1 by the Host Controller during status update when a babble is detected during the transaction. In addition to setting this bit, the hardware also sets the Pipe Halted Bit to a 1 (only for IN endpoints). When the Activate Pipe Bit of the Pipe Control Register goes from 0x0 to 0x1, the core resets this bit.	R	0

3	DBSEL	<p>Selected Data Buffer This bit is only valid if the pipe is configured to have an alternative data buffer. Other-wise the bit will stuck at zero. The bit indicates, which buffer is currently processed by hardware: 0 : (Normal) Data Buffer processed. [label = "DBSEL NORMAL"] 1 : Alternative Data Buffer processed. [label = "DBSEL ALTERNATE"] The bit will be updated by hardware if the data controller switches between the data buffers. The bit can also be changed by software.</p>	R/W	0
2	DT	<p>Data Toggle This bit shows the current data toggle. It is valid for all bulk, control and interrupt transfers. This bit will be changed by hardware according to the current data toggle state. The bit can also be changed by software. 0 : DATA0 data toggle value [label = "DT DATA0"] 1 : DATA1 data toggle value [label = "DT DATA1"] If the core works in peripheral mode: When SETUP token was received, the bit will be automatically set to DT DATA1 for pipe[0].</p>	R/W	0
1:0	DBOFF	<p>Data Byte Offset For transmit transfers, this field indicates the current byte of data to be sent within a data word. For receive transfers, this field points to the first byte within the data word where the first received byte must be stored. If the core parameter DW (Data Width) is set to 16, only the lowest bit of this bitfield is valid. The bitfield can be changed by software. • For transmit pipes: this bitfield traces the current byte offset for this pipe, it is set by hardware to 0 after an buffer becomes invalid. • For streaming receive pipes: this bitfield traces the current byte offset for this pipe. If the pipe works in streaming mode (refer Streaming Mode Bit), this byte offset is also used, to eliminate byte shifting operations by software. Software has only to initialize the byte offset at the beginning of the transfer. • For non-streaming receive pipes: this bitfield traces the current byte offset for this pipe, it is set by hardware to 0 after an buffer becomes invalid. If the core works in peripheral mode: The bitfield will be automatically set to 0x0 for pipe[0] if a SETUP transaction is received.</p>	R/W	0

USB_PIPE_DATA_PTR – USB Pipe Data Pointer Register**0x1c020050**

The contents of this register can be changed by hardware.

Access Rules: the contents of this register must not be changed when the access to the Pipe Configuration Register is granted and the Data Buffer Valid Bit is active.

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		
reserved																						DPTR											

Bits	Name	Description	R/W	Default
31: 10	reserved	-	R	0x00
9:0	DPTR	<p>Data Pointer</p> <p>For transmit transfers, this field points to the first byte of data to be sent. For receive transfers, this field points to the address where the first received byte must be stored. The width of this field depends on the address width of the used RAM. It is important to note, that this bitfield represents a real DATAWORD POINTER. That means the complete address is calculated by appending the valid DBOFF bits to this bitfield. If the core works in peripheral mode: The bitfield will be automatically set to 0x0000 for pipe[0] if a SETUP transaction is received.</p>	R/W	0x00

USB_PIPE_DATA_TOT – USB Pipe Total Bytes Register**0x1c020054**

The contents of this register can be changed by hardware.

Access Rules: the contents of this register must not be changed when the access to the Pipe Configuration Register is granted and the Data Buffer Valid Bit is active.

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
DBV		30:TBW reserved																		TBW-1:0 TBYTES											

Bits	Name	Description	R/W	Default
31	DBV	Data Buffer Valid Software must set this bit to indicate that the data described by the Pipe Data Pointer Register and Pipe Total Bytes Register is valid, respectively the buffer is valid for incoming data. The bit will be cleared by hardware after the buffer was finished (e.g. all data was transmitted or received).	R/W	0
30:TBW	reserved	-	R	0x00
TBW-1:0	TBYTES	Total Bytes To Transfer This bitfield contains the total number of bytes to be transmitted or received. This field will be decremented according to the number of bytes received/transmitted with every transaction. Software must ensure that the field is only written if the data buffer is not valid. Writing to this register if the data is validated will lead to undefined results. For receive pipes, software must set this field always to a multiple of Maximum Packet Size Field. Transfers will only be initiated if TBYTES is greater/equal MPS. If the core works in peripheral mode: The bitfield will be ignored by hardware if the received transaction was a SETUP transaction. (SETUP transactions must always be accepted)	R/W	0x00

TBW: Total Bytes Width

USB_PIPE_ALT_DATA_PTR – USB Pipe Alternative Data Pointer Register**0x1c020058**

Access Rules: the contents of this register must not be changed when the access to the Pipe Configuration Register is granted and the Alternative Data Buffer Valid Bit is active

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
reserved																ALT_DATA_PTR															

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9:0	ALT_DATA_PTR	Alternative Data Pointer This register does only exist, if alternate buffer support is enabled for this pipe. Otherwise this bitfield will stuck at zero. For transmit transfers, this field points to the first byte of the alternative data buffer to be sent. For receive transfers, this field points to the address where the first received byte must be stored. The width of this field depends on the address width of the used RAM..	R/W	0x00

USB_PIPE_ALT_DATA_TOT – USB Pipe Alternative Data Total Bytes Register **0x1c02005c**

Access Rules: the contents of this register must not be changed when the access to the Pipe Configuration Register is granted and the Alternative Data Buffer Valid Bit is active.

	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
ADBV	30:TBW reserved															TBW-1:0 ATBYTES																

Bits	Name	Description	R/W	Default
31	ADBV	<p>Alternative Data Buffer Valid</p> <p>This bit is only valid if the pipe is configured to have an alternative data buffer . Otherwise the bit will stuck at zero. Software must set this bit to indicate that the data described by the Pipe Alternative Data Pointer Register and Pipe Alternative Data Total Bytes Register is valid, respectively the buffer is valid for incoming data.</p> <p>The bit will be cleared by hardware:</p> <ul style="list-style-type: none"> • After an receive buffer was finished, this condition is checked during the status update. The Buffer will be finished when the Total Bytes To Transfer Field is less Maximum Packet Size Field and Selected Data Buffer Bit is 1. • After an transmit buffer was finished, this condition is checked during the status update. The Buffer will be finished when the Total Bytes To Transfer Field is 0x0 and Selected Data Buffer Bit is 1. • The pipe state (Active Status Bit) has changed from active to inactive and the core works in host mode. 	R/W	0
30:TBW	reserved	-	R	0x00
TBW-1:0	ATBYTES	<p>Alternative Total Bytes To Transfer</p> <p>It contains the total number of bytes of the alternative data buffer to be transmitted or received. This field will be decremented according to the number of bytes received/transmitted with every transaction. Software must ensure that the field is only written if the alternative data buffer is not valid. Writing to this register if the data is validated will lead to undefined results!</p>	R/W	0x00

TBW: Total Bytes Width

Debug Register Group

These registers provide various debug capabilities. They will not be used for normal operation. Debug mode is enabled if one of the Debug Control Register bits are set to one.

Debug mode is only valid for pipe 0. The basic configuration of this pipe will be used for a transaction and changes according to the debug configuration.

The core can be configured to support debug or extended debug capability. Therefore, some registers or bits can be not valid (stuck at zero).

USB_DBG_CTRL – USB Debug Control Register

0x1c020060

Debug mode is enabled if one of the control bits is set to one.

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
reserved																						UDTPID	UDHSPID	UDDPID	FRXCRC16G	FRXCRC5G	FRXCRCE	FTXCRC16E	FTXCRC5E	DBSTX	DBSERRDET

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9	UDTPID	Use Debug Token PID If the bit is set, the host controller will use the Debug Token PID Field for the next transaction. The bit will only exist, if extended debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
8	UDHSPID	Use Debug Handshake PID If the bit is set, the host controller will use the Debug Handshake PID Field for the next transaction. The bit will only exist, if extended debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
7	UDDPID	Use Debug Data PID If the bit is set, the host controller will use the Debug Data PID Field for the next transaction. The bit will only exist, if extended debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
6	FRXCRC16G	Force Receive Good CRC16 If this bit is set, the core will treat every received data CRC16 as a good CRC. The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
5	FRXCRC5G	Force Receive Good CRC5 If this bit is set, the core will treat every received CRC5 of an incoming token or SOF as a good CRC. The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
4	FRXCRCE	Force Receive CRC Error If this bit is set, the core will treat every received CRC as a wrong CRC. The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
3	FTXCRC16E	Force Transmit CRC16 Error If this bit is set, the core will generate a wrong data CRC16 (Ones complement). The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
2	FTXCRC5E	Force Transmit CRC5 Error	R/W	0

		If this bit is set, the core will generate a wrong CRC5 for every token/SOF (Ones complement). The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.		
1	DBSTX	Disable Bitstuffing Transmit If this bit is set, the core will not perform bitstuffing at the output stream. The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0
0	DBSERRDET	Disable Bitstuff Error Detection If this bit is set, the core will not report bitstuff errors. Because every bitstuff error will lead to a CRC error, the completion code will show only CRC errors instead of bitstuff errors. The bit will only exist, if debug capability support was implemented. Otherwise the bit will stuck at zero.	R/W	0

USB_DBG_PID – USB Debug PID Register**0x1c020064**

This register will only exist, if extended debug capability support was implemented. Otherwise the bit will stuck at zero.

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
reserved								DHSPID								DTPID								DDPID							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x00
23:16	DHSPID	Debug Handshake PID This bitfield will be sent as handshake PID if the device has returned data.	R/W	0x00
15:8	DTPID	Debug Token PID This bitfield will be sent as token PID if debug mode is enabled instead of an IN/OUT/SETUP token PID.	R/W	0x00
7:0	DDPID	Debug Data PID This bitfield will be sent as data PID if debug mode is enabled instead of an DATA0, DATA1, DATA2 or MDATA PID.	R/W	0x00

USB_DBG_STAT – USB Debug Status Register**0x1c020068**

This register will only exist, if extended debug capability support was implemented. Otherwise the bit will stuck at zero.

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
reserved																								DRXPIP							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	DRXPIP	Debug Receive PID This bitfield contains the received PID of the last transaction, independently of the trans-action status.	R	0x00

USB_TEST – USB Test Register**0x1c02006c**

This register exists for system integration tests only.

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
RW32																															

Bits	Name	Description	R/W	Default
31:0	RW32	Read Write 32 This test field can be read and written in any order and with any contents. Every write access stores the 1's complement of the written value.	R/W	0x00

Core Configurations Register Group

The following registers show the parameters the core is configured with. All bitfields mirror the core's configuration, it is not possible to change the configuration by writing these registers!

USB_MAIN_CFG – USB Main Configuration Register

0x1c020080

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
reserved				RAW_CFG				reserved								DW_CFG				NOP_CFG											

Bits	Name	Description	R/W	Default
31:30	reserved	-	R	0x00
29:24	RAW_CFG	RAM Address Width Configuration This field shows the address width of the Dual Port RAM the core is configured to.	R	0x0a
23:12	reserved	-	R	0x00
11:6	DW_CFG	Data Width Configuration This bitfield shows the data width (DW) of the core.	R	0x20
5:0	NOP_CFG	Number of Pipes Configuration This field shows the number of pipes (NOP) the core is configured with.	R	0x08

USB_MODE_CFG – USB Mode Configuration Register

0x1c020084

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
reserved														XDBG_CFG	DBG_CFG	reserved								ABUFF_CFG							

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x00
17	XDBG_CFG	Extended Debug Configuration This bit shows whether this core is configured for extended debug support.	R	0
16	DBG_CFG	Debug Configuration This bit shows whether this core is configured for debug support.	R	0
15:8	reserved	-	R	0x00
7:0	ABUFF_CFG	Alternative Buffer Configuration This bitfield shows which pipe is configured for alternative Buffer usage.	R	0x00

USB_CORE_CTRL – USB Core Control and Status Register**0x1c020088**

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
reserved							USB_IRQ	UCIF_RDY	DISCHRG_VBUS	VB_ON	DLP_ACTIVE	CHRG_VBUS	VBUS_VLD	VB_SESS_VLD	VB_SESS_END	VA_SESS_VLD	OVER_CURRENT	XCVR_SUSPEND_	CORE_SUSPEND_	ALT_BUFFER_SUPPORT							SOFT_ID_DIG	XTD_DBG_SUPPORT	DBG_SUPPORT	RESET	

Bits	Name	Description	R/W	Default
31:25	reserved	-	R	0x00
24	USB_IRQ	reflects usb_irq	R	0
23	UCIF_RDY	reflects ucif_rdy	R	0
22	DISCHRG_VBUS	reflects dischrg_vbus	R	0
21	VB_ON	reflects vb_on	R	0
20	DLP_ACTIVE	reflects dlp_active	R	0
19	CHRG_VBUS	reflects chrg_vbus	R	0
18	VBUS_VLD	reflects vbus_vld	R	0
17	VB_SESS_VLD	reflects vb_sess_vld	R	0
16	VB_SESS_END	reflects vb_sess_end	R	0
15	VA_SESS_VLD	reflects va_sess_vld	R	0
14	OVER_CURRENT	reflects over_current	R	0
13	XCVR_SUSPEND_N	reflects xcvr_suspend_n	R	0
12	CORE_SUSPEND_N	reflects core_suspend_n	R	0
11:4	ALT_BUFF_SUPPORT	alt buffer support	R/W	0x00
3	SOFT_ID_DIG	to set id_dig via software	R/W	0
2	XTD_DBG_SUPPORT	enables extended debug support	R/W	0
1	DBG_SUPPORT	enables debug support	R/W	0
0	RESET	software reset	R/W	0

USB_FIFO – FIFO for USB-Interface**from 0x1c030000 to 0x1c03ffff**

The function of the USB_FIFO is sending and receiving frames.

6.11 VIC – Vectored Interrupt Controller

The following table shows a summary of all VIC registers:

ARM Address	Register Name	Short Description
0x1c7ff000	VIC_IRQ_STAT	IRQ Status Register
0x1c7ff004	VIC_FIQ_STAT	FIQ Status Register
0x1c7ff008	VIC_RAW_INT_STAT	Raw Interrupt Status Register
0x1c7ff00c	VIC_INT_SEL	Interrupt Select Register
0x1c7ff010	VIC_INT_EN	Interrupt Enable Register
0x1c7ff014	VIC_INT_EN_CLR	Interrupt Enable Clear Register
0x1c7ff018	VIC_SWI	Software Interrupt Register
0x1c7ff01c	VIC_SWI_CLR	Software Interrupt Clear Register
0x1c7ff020	VIC_PROT_EN	Protection Enable Register
0x1c7ff030	VIC_VECT_ADDR	Vector Address Register
0x1c7ff034	VIC_DFLT_VECT_ADDR	Default Vector Address Register
0x1c7ff100	VIC_VECT_ADDR0	Vector Address Register 0
0x1c7ff104	VIC_VECT_ADDR1	Vector Address Register 1
0x1c7ff108	VIC_VECT_ADDR2	Vector Address Register 2
0x1c7ff10c	VIC_VECT_ADDR3	Vector Address Register 3
0x1c7ff110	VIC_VECT_ADDR4	Vector Address Register 4
0x1c7ff114	VIC_VECT_ADDR5	Vector Address Register 5
0x1c7ff118	VIC_VECT_ADDR6	Vector Address Register 6
0x1c7ff11c	VIC_VECT_ADDR7	Vector Address Register 7
0x1c7ff120	VIC_VECT_ADDR8	Vector Address Register 8
0x1c7ff124	VIC_VECT_ADDR9	Vector Address Register 9
0x1c7ff128	VIC_VECT_ADDR10	Vector Address Register 10
0x1c7ff12c	VIC_VECT_ADDR11	Vector Address Register 11
0x1c7ff130	VIC_VECT_ADDR12	Vector Address Register 12
0x1c7ff134	VIC_VECT_ADDR13	Vector Address Register 13
0x1c7ff138	VIC_VECT_ADDR14	Vector Address Register 14
0x1c7ff13c	VIC_VECT_ADDR15	Vector Address Register 15
0x1c7ff200	VIC_VECT_CTRL0	Vector Control Register 0
0x1c7ff204	VIC_VECT_CTRL1	Vector Control Register 1
0x1c7ff208	VIC_VECT_CTRL2	Vector Control Register 2
0x1c7ff20c	VIC_VECT_CTRL3	Vector Control Register 3
0x1c7ff210	VIC_VECT_CTRL4	Vector Control Register 4
0x1c7ff214	VIC_VECT_CTRL5	Vector Control Register 5
0x1c7ff218	VIC_VECT_CTRL6	Vector Control Register 6
0x1c7ff21c	VIC_VECT_CTRL7	Vector Control Register 7
0x1c7ff220	VIC_VECT_CTRL8	Vector Control Register 8
0x1c7ff224	VIC_VECT_CTRL9	Vector Control Register 9
0x1c7ff228	VIC_VECT_CTRL10	Vector Control Register 10
0x1c7ff22c	VIC_VECT_CTRL11	Vector Control Register 11
0x1c7ff230	VIC_VECT_CTRL12	Vector Control Register 12

0x1c7ff234	VIC_VECT_CTRL13	Vector Control Register 13
0x1c7ff238	VIC_VECT_CTRL14	Vector Control Register 14
0x1c7ff23c	VIC_VECT_CTRL15	Vector Control Register 15

VIC_IRQ_STAT – VIC IRQ Status Register**0x1c7ff000**

The VIC_IRQ_STAT register provides the status of the interrupts after registers VIC_INT_EN and VIC_INT_SEL are configured. When an IRQ interrupt occurs, the corresponding bit of the interrupt source will be set to 1.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_IRQ_STAT	Interrupt Status.	R	0x00000000

VIC_FIQ_STAT – VIC FIQ Status Register**0x1c7ff004**

The VIC_FIQ_STAT register provides the status of a FIQ interrupt if the register VIC_INT_EN is enabled and the register VIC_INT_SEL select a FIQ interrupt. When a FIQ interrupt occurs, the corresponding bit of the interrupt source will be set to 1.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_FIQ_STAT	Fast Interrupt Status	R	0x00000000

VIC_RAW_INT_STAT – VIC Raw Interrupt Status Register**0x1c7ff008**

The VIC_RAW_INT_STAT register provides the status of the source interrupts (and software interrupts) to the interrupt controller.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_RAW_INT_STAT	Raw Interrupt Status	R	0x00000000

VIC_INT_SEL – VIC Interrupt Select Register**0x1c7ff00c**

The VIC_INT_SEL register selects whether the corresponding interrupt source generates an FIQ or an IRQ interrupt.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_INT_SEL	Selects type of interrupt for interrupt request: 1: FIQ interrupt 0: IRQ interrupt	R/W	0x00000000

VIC_INT_EN – VIC Interrupt Enable Register**0x1c7ff010**

The VIC_INT_EN register enables the interrupt request lines, by unmasking the interrupt sources for the IRQ interrupt. Clearing these bits are only possible by writing to the 'VIC_INT_EN_CLR - VIC Interrupt Enable Clear Register'

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_INT_EN	Enable the interrupt request lines. Read: 0: Interrupt disabled. 1: Interrupt enabled. Allows interrupt request to processor. Write: 0: no effect. 1: set the bit	R/W	0x00000000

VIC_INT_EN_CLR – VIC Interrupt Enable Clear Register**0x1c7ff014**

The VIC_INT_EN_CLR register is for clearing bits in the VIC_INT_EN register. Writing a one bit will result in clearing the corresponding bit in the 'VIC_INT_EN - VIC Interrupt Enable Register'

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_INT_EN_CLR	Clear bits in the VIC_INT_EN register. 0: has no effect. 1: clears the corresponding bit in the VIC_INT_EN register.	W	0x00000000

VIC_SWI – VIC Software Interrupt Register**0x1c7ff018**

The VIC_SWI register is used to generate software interrupts.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_SWI	Setting a bit generates a software interrupt for the specific source before interrupt masking. Read: 0: Software Interrupt inactive(reset). 1: Software Interrupt active. Write: 0: has no effect. 1: software interrupt enabled.	R/W	0x00000000

VIC_SWI_CLR – VIC Software Interrupt Clear Register**0x1c7ff01c**

The VIC_SWI_CLR register clears bits in the VIC_SWI register.

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
reserved	TIMER4	TIMER3	TRIGGER_LT	DMAC	reserved	INT_PHY	reserved	reserved	MSYNC1	MSYNC0	reserved	reserved	COM1	COM0	GPIO	PCI/HIF	reserved	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO31	SYSTIME_S	SYSTIME_NS	TIMER2	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_SWI_CLR	Clear corresponding bits in the VIC_SWI register. 0: has no effect. 1: software interrupt disabled in the VIC_SWI register.	W	0x00000000

VIC_PROT_EN – VIC Protection Enable Register**0x1c7ff020**

The VIC_PROT_EN register enables or disables protected register access, stopping register accesses when the processor is in User mode.

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
reserved																															VIC_PROT_EN

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x00
0	VIC_PROT_EN	0: VIC registers are accessible 1: VIC registers are not accessible	R/W	0

--	--	--	--	--

VIC_VECT_ADDR – VIC Vector Address Register**0x1c7ff030**

The VIC_VECT_ADDR register contains the Interrupt Service Routine (ISR) address of the currently active interrupt.

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
VIC_VECT_ADDR																															

Bits	Name	Description	R/W	Default
31:0	VIC_VECT_ADDR	Contains the address of the currently active ISR. Any writes to this register clear the current interrupt.	R/W	0x00000000

Note:

Reading from this register provides the address of the ISR, and indicates to the priority hardware that the interrupt is being serviced. Writing to this register indicates to the priority hardware that the interrupt has been serviced. The register should be used as follows:

- The ISR reads the VIC_VECT_ADDR register when an IRQ interrupt is generated
- At the end of the ISR, the VIC_VECT_ADDR register is written to, to update the priority hardware.

Reading or writing to the register at other times can cause incorrect operation.

VIC_DFLT_VECT_ADDR – VIC Default Vector Address Register**0x1c7ff034**

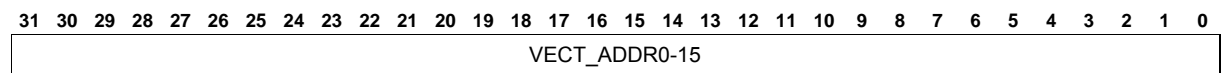
The VIC_DFLT_VECT_ADDR register contains the default ISR address.

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
DFLT_VECT_ADDR																															

Bits	Name	Description	R/W	Default
31:0	VIC_DFLT_VECT_ADDR	Contains the address of the default ISR handler.	R/W	0x00000000

VIC_VECT_ADDR0 – VIC Vector Address Registers 0	0x1c7ff100
VIC_VECT_ADDR1 – VIC Vector Address Registers 1	0x1c7ff104
VIC_VECT_ADDR2 – VIC Vector Address Registers 2	0x1c7ff108
VIC_VECT_ADDR3 – VIC Vector Address Registers 3	0x1c7ff10c
VIC_VECT_ADDR4 – VIC Vector Address Registers 4	0x1c7ff110
VIC_VECT_ADDR5 – VIC Vector Address Registers 5	0x1c7ff114
VIC_VECT_ADDR6 – VIC Vector Address Registers 6	0x1c7ff118
VIC_VECT_ADDR7 – VIC Vector Address Registers 7	0x1c7ff11c
VIC_VECT_ADDR8 – VIC Vector Address Registers 8	0x1c7ff120
VIC_VECT_ADDR9 – VIC Vector Address Registers 9	0x1c7ff124
VIC_VECT_ADDR10 – VIC Vector Address Registers 10	0x1c7ff128
VIC_VECT_ADDR11 – VIC Vector Address Registers 11	0x1c7ff12c
VIC_VECT_ADDR12 – VIC Vector Address Registers 12	0x1c7ff130
VIC_VECT_ADDR13 – VIC Vector Address Registers 13	0x1c7ff134
VIC_VECT_ADDR14 – VIC Vector Address Registers 14	0x1c7ff138
VIC_VECT_ADDR15 – VIC Vector Address Registers 15	0x1c7ff13c

The VIC_VECT_ADDR0-15 registers contain the ISR vector addresses. These registers must only be updated when the relevant interrupts are disabled. Receiving an interrupt while the vector address is being written to can result in unpredictable behavior.



Bits	Name	Description	R/W	Default
31:0	VIC_VECT_ADDR0-15	Contains ISR vector address.	R/W	0x00000000

VIC_VECT_CTRL0 – VIC Vector Control Registers 0	0x1c7ff200
VIC_VECT_CTRL1 – VIC Vector Control Registers 1	0x1c7ff204
VIC_VECT_CTRL2 – VIC Vector Control Registers 2	0x1c7ff208
VIC_VECT_CTRL3 – VIC Vector Control Registers 3	0x1c7ff20c
VIC_VECT_CTRL4 – VIC Vector Control Registers 4	0x1c7ff210
VIC_VECT_CTRL5 – VIC Vector Control Registers 5	0x1c7ff214
VIC_VECT_CTRL6 – VIC Vector Control Registers 6	0x1c7ff218
VIC_VECT_CTRL7 – VIC Vector Control Registers 7	0x1c7ff21c
VIC_VECT_CTRL8 – VIC Vector Control Registers 8	0x1c7ff220
VIC_VECT_CTRL9 – VIC Vector Control Registers 9	0x1c7ff224
VIC_VECT_CTRL10 – VIC Vector Control Registers 10	0x1c7ff228
VIC_VECT_CTRL11 – VIC Vector Control Registers 11	0x1c7ff22c
VIC_VECT_CTRL12 – VIC Vector Control Registers 12	0x1c7ff230
VIC_VECT_CTRL13 – VIC Vector Control Registers 13	0x1c7ff234
VIC_VECT_CTRL14 – VIC Vector Control Registers 14	0x1c7ff238
VIC_VECT_CTRL15 – VIC Vector Control Registers 15	0x1c7ff23c

VIC_VECT_CTRL0-15 registers select interrupt source and set if it is enabled.

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
Reserved																										ENABLE	INT_SOURCE				

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	ENABLE	Enables vector interrupt. This bit is cleared on reset.	R/W	0
4:0	INT_SOURCE	Select interrupt source. You can select any of the 32 interrupt sources.	R/W	0x00

7 Communication Functions

The following table is a summary of registers related to communication functions.

ARM Address	Register Name	Short Description
0x1c000010	PHY_CTRL	PHY Control Register
0x1c000c00	MIIMU_RXTX	MIIMU Receive/Transmit Register
0x1c000c04	MIIMU_MODE_EN	MIIMU Software Mode Enable
0x1c000c08	MIIMU_MODE_MDC	MIIMU Software Mode MDC Register
0x1c000c0c	MIIMU_MODE_MDO	MIIMU Software Mode MDO Register
0x1c000c10	MIIMU_MODE_MDOE	MIIMU Software Mode MDOE Register
0x1c000c14	MIIMU_MODE_MDI	MIIMU Software Mode MDI Register
0x1c064000	PTR_FIFO_BASE	Pointer FIFO Base Address
0x1c064080	PTR_FIFO_BOR_BASE	Pointer FIFO Upper Border Base Address
0x1c064100	PTR_FIFO_RESET	Pointer FIFO Reset Vector
0x1c064104	PTR_FIFO_FULL	Pointer FIFO Full Vector
0x1c064108	PTR_FIFO_EMPTY	Pointer FIFO Empty Vector
0x1c06410c	PTR_FIFO_OVF	Pointer FIFO Overflow Vector
0x1c064110	PTR_FIFO_UDR	Pointer FIFO Under-Run Vector
0x1c064180	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Fill Level Base Address
0x1c065600	BUF_MAN_XPEC0	BMU port of 1st master (xPEC0)
0x1c065604	BUF_MAN_XPEC1	BMU port of 2nd master (xPEC1)
0x1c065608	BUF_MAN_ARM	BMU-port of 3rd master (ARM)
0x1c06560c	BUF_MAN_HIF	BMU-port of 4th master (HIF)
0x1c001000	CRC_VAL	Calculated CRC Value
0x1c001004	CRC_IN_DATA	CRC Input Data
0x1c001008	CRC_POLYNOMIAL	Polynomial for CRC Calculation
0x1c00100c	CRC_CFG	CRC Configuration Register
0x1c064400	IRQ_XP0	IRQs between XPEC0 and ARM
0x1c064404	IRQ_XP1	IRQs between XPEC1 and ARM

7.1 PHY – Controller for internal PHYs

This chapter describes the registers used to parameterize the integrated 10/100MBit Ethernet PHY. The PHY_CTRL register is used to access the internal signals of the integrated PHY unit, while the other registers belong to the MII-Management Unit (MIIMU). The MIIMU allows to handle a standard MDIO interface (s. IEEE 802.3) used to access the internal registers of the integrated Ethernet PHY or of optional external Ethernet PHYs.

PHY_CTRL – PHY Control Register

0x1c000010

This register contains all static connectors of the NEC Ethernet PHY. Usually the PHY reads these values only during reset, which can be controlled by Bit31. This register is protected by the netX access key mechanism; changing this register is only possible by the following sequence:

- 1.: read out access key from ACCESS_KEY register
- 2.: write back access key to ACCESS_KEY register
- 3.: write desired value to this register

In total the programming sequence should be:

- a: read access key, write access key, write new value with bit phy_reset=1
- b: wait for proper reset of PHY(~100µs)
- c: read access key, write access key, write new value with bit phy_reset=0

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				
PHY_RESET		PHY_SIM_BYP		reserved									PHY1_ENABLE		PHY1_NP_MSG_CODE		PHY1_AUTOMDIX		PHY1_FXMODE		PHY1_MODE		PHY0_ENABLE		PHY0_NP_MSG_CODE		PHY0_AUTOMDIX		PHY0_FXMODE		PHY0_MODE		PHY_ADDRESS		

Bits	Name	Description	R/W	Default
31	PHY_RESET	Hardware reset for PHY 1: reset	R/W	0x0
30	PHY_SIM_BYP	PHY Power up Bypass (only used for simulation issues) 0: normal 1: bypass Bit is synchronized to phyclk and drives pwruprstbyp pin of PHY, which bypasses Power Up Reset of PHY for faster simulation.	R/W	0x1
29:22	-	reserved	R	0x0
21	PHY1_ENABLE	PHY1 enable	R/W	0x0
20:18	PHY1_NP_MSG_CODE	PHY1 Next Page Message Code (auto negotiation)	R/W	0x0
17	PHY1_AUTOMDIX	PHY1 Enables AutoMDIX state machine	R/W	0x0
16	PHY1_FXMODE	PHY1 100BASE-FX mode (phy_mode must be 01x)	R/W	0x0
15:13	PHY1_MODE	PHY1 Mode: 000: 10BASE-T Half Duplex, Auto Negotiation disabled. 001: 10BASE-T Full Duplex. Auto-Negotiation disabled. 010: 100BASE-TX/FX Half Duplex. Auto-Negotiation disabled. CRS is active during Transmit & Receive. 011: 100BASE-TX/FX Full Duplex. Auto-Negotiation disabled. CRS is active during Receive. 100: 100BASE-TX Half Duplex is advertised. Auto-Negotiation enabled. CRS is active during Transmit & Receive. 101: Repeater mode. Auto-Negotiation enabled. 100BASE-TX Half Duplex is advertised. CRS is active during Receive. 110: Power Down mode. In this mode the PHY wake-up in Power-Down mode. 111: All capable. Auto-Negotiation enabled. AutoMDIX enabled.	R/W	0x6
12	PHY0_ENABLE	PHY0 enable	R/W	0x0
11:9	PHY0_NP_MSG_CODE	PHY0 Next Page Message Code (auto negotiation)	R/W	0x0
8	PHY0_AUTOMDIX	PHY0 Enables AutoMDIX state machine	R/W	0x0
7	PHY0_FXMODE	PHY0 100BASE-FX mode (phy_mode must be 01x)	R/W	0x0
6:4	PHY0_MODE	PHY0 Mode: 000: 10BASE-T Half Duplex, Auto Negotiation disabled. 001: 10BASE-T Full Duplex. Auto-Negotiation disabled. 010: 100BASE-TX/FX Half Duplex. Auto-Negotiation disabled. CRS is active during Transmit & Receive. 011: 100BASE-TX/FX Full Duplex. Auto-Negotiation disabled. CRS is active during Receive. 100: 100BASE-TX Half Duplex is advertised. Auto-Negotiation enabled. CRS is active during Transmit & Receive. 101: Repeater mode. Auto-Negotiation enabled. 100BASE-TX Half Duplex is advertised. CRS is active during Receive. 110: Power Down mode. In this mode the PHY wake-up in Power-Down mode. 111: All capable. Auto-Negotiation enabled. AutoMDIX enabled.	R/W	0x6
3:0	PHY_ADDRESS	Bits 4:1 of PHY mdio-address. Bit0 defines 1st or 2nd internal PHY	R/W	0x0

MIIMU_RXTX – MIIMU Receive/Transmit Register**0x1c000c00**

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
MIIMU_DATA																MIIMU_PHYADDR				MIIMU_REGADDR				MIIMU_RTA	PHY_NRES	MIIMU_MDC_PERIOD	MIIMU_OPMODE	MIIMU_PREAMBLE	MIIMU_SNRDY		

Bits	Name	Description	R/W	Default
31:16	MIIMU_DATA	Data to or from PHY register	R/W	0x0
15:11	MIIMU_PHYADDR	PHY address	R/W	0x0
10:6	MIIMU_REGADDR	Register address	R/W	0x0
5	MIIMU_RTA	Read Turn Around field: 0: one bit 1: two bits	R/W	0x0
4	PHY_NRES	PHY hardware nReset (activ low!): If this bit and the miimu_snrly-bit is set, the PHYs will be hardware-reset. No data will be transmitted in this case. After reset the miimu-controller will automatically reset this bit to 1.	R/W	0x1
3	MIIMU_MDC_PERIOD	MDC period: 1: 800ns 0: 400ns	R/W	0x0
2	MIIMU_OPMODE	Operation mode: 1: write 0: read	R/W	0x0
1	MIIMU_PREAMBLE	Send preamble	R/W	0x0
0	MIIMU_SNRDY	Start not ready	R/W	0x0

MIIMU_MODE_EN – MIIMU Software Mode Enable**0x1c000c04**

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
reserved																															MIIMU_SW_EN

Bits	Name	Description	R/W	Default
31:1	-	reserved	R	0x0
0	MIIMU_SW_EN	Enables software mode: MDC, MDO and MDOE are set by software. The current md_in value can be read from miimu_sw_mdi.	R/W	0x0

0x1c000c08

Bits	Name	Description	R/W	Default
31:1	-	reserved	R	0x0
0	MIIMU_SW_MDC	MDC value for software mode	R/W	0x0

0x1c000c0c

Bits	Name	Description	R/W	Default
31:1	-	reserved	R	0x0
0	MIIMU_SW_MDO	MDO value for software mode	R/W	0x0

0x1c000c10

Bits	Name	Description	R/W	Default
31:1	-	reserved	R	0x0
0	MIIMU_SW_MDOE	MDOE value for software mode	R/W	0x0

0x1c000c14

Bits	Name	Description	R/W	Default
31:1	-	reserved	R	0x0
0	MIIMU_SW_MDI	current MDI value	R	0x0

7.2 PTR_FIFO – Pointer FIFO

The task of the Pointer FIFO module is to support handling of different data buffers in a multiprocessor system. Basically, it consists of a set of 32 FIFOs, which can be accessed by all processors. Each FIFO can replace a linked list, which is usually used to handle data channels between the processors. Accessing the Pointer FIFO is much faster and more predictable than using linked lists, as processors do not have to wait for each other before changing a linked list.

Some FIFOs are already used by some communication protocols (e.g. Ethernet uses some FIFOs for communication with ARM-CPU and some others internally when supporting a switch function), but basically the software is completely free to assign different FIFOs to different data channels between processors.

PTR_FIFO_BASE[0-31] – Pointer FIFO 0-31 Base Address

0x1c064000-0x1c06407c

The PTR_FIFO_BASE register provides the write/read interface for data into/out of the corresponding pointer FIFO.

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
FIFO_DATA																															

Bits	Name	Description	R/W	Default
31:0	FIFO_DATA	In/output data to/from FIFO: write access: write data to FIFO read access: read data from FIFO	R/W	0x0

PTR_FIFO_BOR_BASE[0-31] – Pointer FIFO0-31 Upper Border

0x1c064080-0x1c0640fc

The sizes of all FIFOs are programmable. The total size of all FIFOs must not exceed 1024 dwords. Each of the following 32 addresses accesses the upper border of the appropriate FIFO in a 1024x32 bit RAM. All upper borders should be rising with number of FIFO. Each FIFO starts at the upper border + 1 of the preceding FIFO and ends at its upper border.

If a border between two FIFOs is moved, the adjacent FIFOs should be reset first.

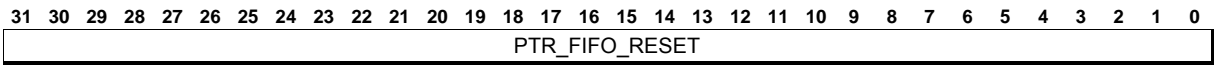
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																BORDER															

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9:0	BORDER	Last address of RAM used by appropriate FIFO = (first address – 1) of next FIFO FIFO 0 default depth : 512 FIFO 1..30 default depth : 16 FIFO 31 default depth : 32	R/W	0x0

PTR_FIFO_RESET – Pointer FIFO Reset Vector

0x1c064100

Pointer FIFO reset vector. Allows to reset each of the 32 FIFOs, i.e. set read and write pointer to lower border of FIFO, reset full, overflow, under run and set empty flag. Reset flag of adjacent FIFOs should be set before resizing the FIFO.



Bits	Name	Description	R/W	Default
31:0	PTR_FIFO_RESET	Reset Vector, 1 bit per FIFO: 1: reset FIFO 0: normal work mode	R/W	0x0

PTR_FIFO_FULL – Pointer FIFO Full Vector**0x1c064104**

This read only address shows the FIFO_FULL flag of each FIFO.

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
PTR_FIFO_FULL																															

Bits	Name	Description	R/W	Default
31:0	PTR_FIFO_FULL	FIFO full vector, 1 bit per FIFO	R	0x0

PTR_FIFO_EMPTY – Pointer FIFO Empty Vector**0x1c064108**

This read only address shows the FIFO_EMPTY flag of each FIFO.

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
PTR_FIFO_EMPTY																															

Bits	Name	Description	R/W	Default
31:0	PTR_FIFO_EMPTY	FIFO empty vector, 1 bit per FIFO	R	0xffffffff

PTR_FIFO_OVF – Pointer FIFO Overflow Vector**0x1c06410c**

This read only address shows the FIFO_OVERFLOW flag of each FIFO.

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
FIFO_OVERFLOW																															

Bits	Name	Description	R/W	Default
31:0	FIFO_OVERFLOW	FIFO overflow vector, 1 bit per FIFO	R	0x0

PTR_FIFO_UDR – Pointer FIFO Under Run Vector**0x1c064110**

This read only address shows the FIFO_UNDERRUN flag of each FIFO.

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
FIFO_UNDERRUN																															

Bits	Name	Description	R/W	Default
31:0	FIFO_UNDERRUN	FIFO underrun vector, 1 bit per FIFO	R	0x0

PTR_FIFO_FILL_LVL_BASE[0-31] – Pointer FIFO Fill Level 0-31**0x1c064180-0x1c0641fc**

Each of the following 32 addresses reads the FILL_LEVEL of the appropriate FIFO.

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
reserved																						PTR_FIFO_FILL_LVL									

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x00
9:0	PTR_FIFO_FILL_LVL	Actual number of words in appropriate FIFO (not valid, if FIFO had an overflow or underflow)	R	0x00

7.3 Buffer Management Unit (BMU)

The Buffer Management Unit (BMU) supports handling of different data buffers in a multiprocessor system. It does the distribution of the most current data to all processors (contrary to the Pointer FIFO that handles sequences of data buffers between certain processors).

The most primitive case of distribution of the most current data is between two processors by the well-known Triple-Buffer-Algorithm. As our system consists of four channels (ARM, Host-CPU/HIF, xPEC0 and xPEC1) with the appropriate processors, the BMU hardware is enhanced to a 'Penta Buffer Manager' (in general: n processors need $n+1$ buffers). Each processor can request the BMU for a read (or write) buffer and gets back the number of the most actual (or not used by others) buffer.

If realizing this data distribution in software, the handshake between processors wastes a lot of calculation performance. Especially when one CPU works much slower, the faster CPU spends many cycles waiting for an acknowledge. In contrary, the Buffer Manager directly returns the number of the optimum buffer, which can easily be translated to the physical memory address.

The semaphore mode of the Buffer Manager is a reduced n -buffer mode, where only `buf_nr=0` (you get the semaphore) and `buf_nr=7` (you do not get the semaphore) are returned. A read-request requests the semaphore; write- or release-requests release the semaphore.

The Buffer Manager Module inside the netX50 consists of 16 Buffer Controllers (channels). Each Buffer Controller can handle a Penta-Buffer between up to four processors. All Buffer Controllers work completely independent and can be used in completely independent software tasks.

Some Buffer Controllers are already used by some communication protocols (e.g. EtherCAT uses up to 8 Buffer Controllers with up to 8 Sync-Managers), but in general the software is completely free to assign different Buffer Controllers to different data channels between processors.

BUF_MAN_XPEC0 – BMU port of 1st master (xPEC0)**0x1c065600**

This register address allows to access 16 buffer controllers, where each one handles buffer numbers (0..4) between up to four processors. Due to the complex functionality in one register address, bits have different meaning depending on request type and mode.

Getting a new buffer always happens with two command accesses:

1st: Write access: Tell the buf_manager the channel(s) (0..15) and whether you request read or write buffer.

Wait for two clock cycles, until new buffer number is calculated after any write access.

2nd: Read access: Read the buffer number (0..4).

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			
reserved																				SM_UPDATE_DIS	SM_UPDATE_EN	RESET	PARALLEL_MODE	SEMAPHORE_MODE	REQ_TYPE	reserved	BUF_Nr							

Bits	Name	Description	R/W	Default
31:12	-	reserved	R	0x0
11	SM_UPDATE_DIS	De-activate SM_auto_update mode by writing 1 to this bit.	R/W	0x0
10	SM_UPDATE_EN	Activate SM_auto_update mode by writing 1 to this bit: In SM_auto_update mode the requested buffer numbers of buffer managers 0..7 will automatically be programmed to the FMMU_SM unit.	R/W	0x0
9	RESET	Reset channel.	R/W	0x0
8	PARALLEL_MODE	Activate parallel mode by writing 1 to this bit (other bits are ignored). To return to normal mode, write 0xffff0000 to this register. In parallel mode, the behaviour of all bits of this register changes completely. Parallel mode write access: 15.. 0: Request bits of all 16 channels: 1: request new buffer or semaphore. 0: don't request buffer or semaphore. 31..16: wr bits of all 16 channels: 1: request write buffer or semaphore. 0: request read buffer or semaphore. Parallel mode read access: 1,0: Actual buffer number of channel 0. ... 31,30: Actual buffer number of channel 15. In parallel mode, the number of masters is limited to 2, resulting in 3 buffers per channel. In parallel mode, buffers cannot be released without requesting new buffer numbers.	R/W	0x0
7	SEMAPHORE_MODE	Activate 'semaphore mode' for this buf_nr by writing 1 to this bit. To return from semaphore-mode reset this channel. In semaphore mode only buf_nr=0 (this master gets the semaphore) or buf_nr=7 (master does not get semaphore) are returned. Requesting or releasing a semaphore (by req_type) is allowed while switching to semaphore mode.	R/W	0x0
6:5	REQ_TYPE	Request type bits are write-only: 00: request read buffer (or semaphore) 01: request write buffer (or release semaphore) 10: release write buffer (or release semaphore) 11: do not request new buffer or semaphore (used to only change channel)	R/W	0x0
4	-	reserved	R	0x0
3:0	BUF_NR	Write access: number of buf_manager controller (0..15) Read access: number of buffer (0..m+1), where m is the number of masters using this buf_manager	R/W	0x7

BUF_MAN_XPEC1 – BMU port of 2nd master (xPEC1)**0x1c065604**

This register address allows to access 16 buffer controllers, where each one handles buffer numbers (0..4) between up to four processors. Due to the complex functionality in one register address, bits have different meaning depending on request type and mode.

Getting a new buffer always happens with two command accesses:

1st: Write access: Tell the buf_manager the channel(s) (0..15) and whether you request read or write buffer.

Wait for two clock cycles, until new buffer number is calculated after any write access.

2nd: Read access: Read the buffer number (0..4).

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			
reserved																				SM_UPDATE_DIS	SM_UPDATE_EN	RESET	PARALLEL_MODE	SEMAPHORE_MODE	REQ_TYPE	reserved	BUF_Nr							

Bits	Name	Description	R/W	Default
31:12	-	reserved	R	0x0
11	SM_UPDATE_DIS	De-activate SM_auto_update mode by writing 1 to this bit.	R/W	0x0
10	SM_UPDATE_EN	Activate SM_auto_update mode by writing 1 to this bit: In SM_auto_update mode the requested buffer numbers of buffer managers 0..7 will automatically be programmed to the FMMU_SM unit.	R/W	0x0
9	RESET	Reset channel.	R/W	0x0
8	PARALLEL_MODE	Activate parallel mode by writing 1 to this bit (other bits are ignored). To return to normal mode, write 0xffff0000 to this register. In parallel mode, the behaviour of all bits of this register changes completely. Parallel mode write access: 15.. 0: Request bits of all 16 channels: 1: request new buffer or semaphore. 0: don't request buffer or semaphore. 31..16: wr bits of all 16 channels: 1: request write buffer or semaphore. 0: request read buffer or semaphore. Parallel mode read access: 1,0: Actual buffer number of channel 0. ... 31,30: Actual buffer number of channel 15. In parallel mode, the number of masters is limited to 2, resulting in 3 buffers per channel. In parallel mode, buffers cannot be released without requesting new buffer numbers.	R/W	0x0
7	SEMAPHORE_MODE	Activate 'semaphore mode' for this buf_nr by writing 1 to this bit. To return from semaphore-mode reset this channel. In semaphore mode only buf_nr=0 (this master gets the semaphore) or buf_nr=7 (master does not get semaphore) are returned. Requesting or releasing a semaphore (by req_type) is allowed while switching to semaphore mode.	R/W	0x0
6:5	REQ_TYPE	Request type bits are write-only: 00: request read buffer (or semaphore) 01: request write buffer (or release semaphore) 10: release write buffer (or release semaphore) 11: do not request new buffer or semaphore (used to only change channel)	R/W	0x0
4	-	reserved	R	0x0
3:0	BUF_NR	Write access: number of buf_manager controller (0..15) Read access: number of buffer (0..m+1), where m is the number of masters using this buf_manager	R/W	0x7

BUF_MAN_ARM – BMU-port of 3rd master (ARM)**0x1c065608**

This register address allows to access 16 buffer controllers, where each one handles buffer numbers (0..4) between up to four processors. Due to the complex functionality in one register address, bits have different meaning depending on request type and mode.

Getting a new buffer always happens with two command accesses:

1st: Write access: Tell the buf_manager the channel(s) (0..15) and whether you request read or write buffer.

Wait for two clock cycles, until new buffer number is calculated after any write access.

2nd: Read access: Read the buffer number (0..4).

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						
reserved																						RESET	PARALLEL_MODE	SEMAPHORE_MODE	REQ_TYPE	reserved	BUF_NR										

Bits	Name	Description	R/W	Default
31:10	-	reserved	R	0x0
9	RESET	Reset channel.	R/W	0x0
8	PARALLEL_MODE	<p>Activate parallel mode by writing 1 to this bit (other bits are ignored). To return to normal mode, write 0xffff0000 to this register.</p> <p>In parallel mode, the behaviour of all bits of this register changes completely.</p> <p>Parallel mode write access:</p> <p>15.. 0: Request bits of all 16 channels:</p> <p>1: request new buffer or semaphore.</p> <p>0: don't request buffer or semaphore.</p> <p>31..16: wr bits of all 16 channels:</p> <p>1: request write buffer or semaphore.</p> <p>0: request read buffer or semaphore.</p> <p>Parallel mode read access:</p> <p>1,0: Actual buffer number of channel 0.</p> <p>...</p> <p>31,30: Actual buffer number of channel 15.</p> <p>In parallel mode, the number of masters is limited to 2, resulting in 3 buffers per channel.</p> <p>In parallel mode, buffers cannot be released without requesting new buffer numbers.</p>	R/W	0x0
7	SEMAPHORE_MODE	<p>Activate 'semaphore mode' for this buf_nr by writing 1 to this bit. To return from semaphore-mode reset this channel.</p> <p>In semaphore mode only buf_nr=0 (this master gets the semaphore) or buf_nr=7 (master does not get semaphore) are returned.</p> <p>Requesting or releasing a semaphore (by req_type) is allowed while switching to semaphore mode.</p>	R/W	0x0
6:5	REQ_TYPE	<p>Request type bits are write-only:</p> <p>00: request read buffer (or semaphore)</p> <p>01: request write buffer (or release semaphore)</p> <p>10: release write buffer (or release semaphore)</p> <p>11: do not request new buffer or semaphore (used to only change channel)</p>	R/W	0x0
4	-	reserved	R	0x0
3:0	BUF_NR	<p>Write access: number of buf_manager controller (0..15)</p> <p>Read access: number of buffer (0..m+1), where m is the number of masters using this buf_manager</p>	R/W	0x7

BUF_MAN_HIF – BMU-port of 4th master (HIF)**0x1c06560c**

This register address allows to access 16 buffer controllers, where each one handles buffer numbers (0..4) between up to four processors. Due to the complex functionality in one register address, bits have different meaning depending on request type and mode.

Getting a new buffer always happens with two command accesses:

1st: Write access: Tell the buf_manager the channel(s) (0..15) and whether you request read or write buffer.

Wait for two clock cycles, until new buffer number is calculated after any write access.

2nd: Read access: Read the buffer number (0..4).

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
reserved																						RESET	PARALLEL_MODE	SEMAPHORE_MODE	REQ_TYPE	reserved	BUF_NR				

Bits	Name	Description	R/W	Default
31:10	-	reserved	R	0x0
9	RESET	Reset channel.	R/W	0x0
8	PARALLEL_MODE	<p>Activate parallel mode by writing 1 to this bit (other bits are ignored). To return to normal mode, write 0xffff0000 to this register.</p> <p>In parallel mode, the behaviour of all bits of this register changes completely.</p> <p>Parallel mode write access:</p> <p>15.. 0: Request bits of all 16 channels:</p> <p>1: request new buffer or semaphore.</p> <p>0: don't request buffer or semaphore.</p> <p>31..16: wr bits of all 16 channels:</p> <p>1: request write buffer or semaphore.</p> <p>0: request read buffer or semaphore.</p> <p>Parallel mode read access:</p> <p>1,0: Actual buffer number of channel 0.</p> <p>...</p> <p>31,30: Actual buffer number of channel 15.</p> <p>In parallel mode, the number of masters is limited to 2, resulting in 3 buffers per channel.</p> <p>In parallel mode, buffers cannot be released without requesting new buffer numbers.</p>	R/W	0x0
7	SEMAPHORE_MODE	<p>Activate 'semaphore mode' for this buf_nr by writing 1 to this bit. To return from semaphore-mode reset this channel.</p> <p>In semaphore mode only buf_nr=0 (this master gets the semaphore) or buf_nr=7 (master does not get semaphore) are returned.</p> <p>Requesting or releasing a semaphore (by req_type) is allowed while switching to semaphore mode.</p>	R/W	0x0
6:5	REQ_TYPE	<p>Request type bits are write-only:</p> <p>00: request read buffer (or semaphore)</p> <p>01: request write buffer (or release semaphore)</p> <p>10: release write buffer (or release semaphore)</p> <p>11: do not request new buffer or semaphore (used to only change channel)</p>	R/W	0x0
4	-	reserved	R	0x0
3:0	BUF_NR	<p>Write access: number of buf_manager controller (0..15)</p> <p>Read access: number of buffer (0..m+1), where m is the number of masters using this buf_manager</p>	R/W	0x7

7.4 CRC – Configurable CRC-Generator

The CRC Generator supports generation of Cyclic Redundancy Checksums (CRC) with programmable polynomial and size (up to 32-bit). All data must be written sequentially to CRC_IN_DATA register (which can be done by DMA unit), then the CRC can simply be read from CRC_VAL register.

CRC_VAL – CRC Register

0x1c001000

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
CRC_VAL																															

Bits	Name	Description	R/W	Default
31:0	CRC_VAL	CRC value Upper bits should be masked, if crc_len smaller than 31	R/W	0x00

CRC_IN_DATA – CRC Input Data Register

0x1c001004

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
reserved																								CRC_IN_DATA							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x00
7:0	CRC_IN_DATA	CRC input data	R/W	0x00

CRC_POLYNOMIAL – CRC Polynomial Register

0x1c001008

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
CRC_POLYNOMIAL																															

Bits	Name	Description	R/W	Default
31:0	CRC_POLYNOMIAL	CRC polynomial Most significant bit of polynomial is always one, thus not considered. Default: Ethernet CRC 32.	R/W	0x4c11db7

CRC_CFG – CRC Configuration Register**0x1c00100c**

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				
reserved																					CRC_IN_MSB_LOW	CRC_NOF_BITS	CRC_DIRECT_DIV	CRC_SHIFT_DIV	CRC_LEN										

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x00
10	CRC_IN_MSB_LOW	Swap crc_data_in, only usefull when calculating multiple bits in parallel (crc_nof_bits > 0): 1: msb of incoming bits is data_in[0], 0: msb is data_in[crc_nof_bits_m1] (msb=first bit in data-stream)	R/W	0
9:8	CRC_NOF_BITS	Number of bits to be calculated in parallel 00: 1 01: 2 10: 4 11: 8	R/W	00
7	CRC_DIRECT_DIV	1: calculate direct polynomial division without n zeros after frame, useful for parity calculation 0 : disable direct division	R/W	0
6	CRC_SHIFT_DIV	1 : shift CRC right 0 : shift CRC left	R/W	0
5:0	CRC_LEN	Length of CRC Polynom – 1	R/W	0x00

7.5 ARM_to_XPEC_IRQ

The following two registers indicate the interrupt requests between xPECs and ARM.

IRQ_XP0 – IRQs between XPEC0 and ARM Registers

0x1c064400

IRQ_XP1 – IRQs between XPEC1 and ARM Registers

0x1c064404

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
ARM_IRQ																XPEC_IRQ															

Bits	Name	Description	R/W	Default
31:16	ARM_IRQ	set by arm ; reset by xpec	R/W	0x00
15:0	XPEC_IRQ	set by xpec ; reset by arm	R/W	0x00

8 DMA Controller

The DMA (Direct Memory Access) controller manages data transfers between DMA slaves and memory slaves without using the ARM CPU, allowing fast data transfers that do not affect CPU load.

8.1 Functional Overview

A slave is a device that is selected by a controlling master as either the source or the target for a transfer. A slave can also begin a service request, using an interrupt. There are three types of slaves: memory, I/O, and DMA.

The DMA controller is designed to work with 32-bit AHBL (Advanced High-performance Bus Light) bus system and is functionally compatible to the ARM Master DMA Controller (PL081). The DMA controller is designed to use only one master channel in the SoC system.

The DMA controller can support up to four DMA channels. Each DMA channel can be programmed for various features, such as transfer size, synchronized and unsynchronized transfer control, interrupt generation, memory and I/O address space, and address change direction.

Typical Applications

- Optimized memory copy function
- Optimized peripheral data block transfer function
- Periodical data transfer to slave/master (e.g. CCD sensor, TFT display, et al)

Features

- ARM DMAC software and register compatible
- 1 AHBL (32 -Bit) master port, for DMA transfer and list operations
- 1 AHBL (32 -Bit) slave port, for programming interface
- 4 DMA channels with separated linked lists
- 4 word (32 -Bit) FIFO per channel
- Linked list operation support on each channel
- Incrementing or non-incrementing addressing for source and destination (support FIFO read and write).
- Software programmable DMA channel priority strategy. Hardware priority (0 highest, 3 lowest) or priority lists (last served channel gets new lowest priority).
- Programmable burst size
- Memory-to-memory, memory-to-peripheral, peripheral-to-memory and peripheral-to-peripheral DMA transfers.
- DMAC or peripheral flow control. Support peripheral DMA flow control signals (request, last burst).
- Error and finish interrupt generation
- Interrupt masking, clear interrupt
- 32-, 16- and 8-Bit support for source and destination in all combinations.
- Hardware DMA channels priority. Each DMA channel has a specific hardware priority. DMA channel 0 has the highest priority and channel 3 has the lowest priority.
- Programmable Interrupt capabilities

Non supported Features of the DMAC

- AHB protected transfers (user mode, buffer able, cacheable)
- AHB lock transfers
- Limitation to 4 channels due to area optimization (module grows linear with the number of channels)
- No big-endian support

8.2 Functional Description

The following section provides a detailed description of DMA controller and its features.

The 4 channel DMA controller supports the following transactions in the netX50 SoC:

- Peripheral to memory transfer
- Memory to peripheral transfer
- Peripheral to peripheral transfer
- Memory to memory transfer

Each channel supports a unidirectional up to 32-Bit DMA transfer for a single source and destination address.

Therefore a bidirectional transfer requires one stream for transmit and one for receive.

The source and destination address can be either a memory region or a peripheral device of the netX50. The DMA controller is programmed by the ARM CPU via the AHBL slave interface.

Bus Transfer Widths on the AHB Master Interface

The default bus width for the AHB master is 32-bit. Source and destination transfers can be of differing widths, and can be the same width or narrower than the physical bus width. The DMA Controller packs or unpacks data observe the programming parameters.

Little-Endian Format

The DMA Controller supports little-endian addressing only. Internally, the DMAC treats all data as a stream of bytes instead of 16-bit or 32-bit quantities.

Note:

To avoid byte swapping of the data always address the peripheral interfaces in 32-bit mode.

DMA Channel Priority

The DMA channel priority is fixed. DMA channel 0 has the highest priority and DMA channel 3 has the lowest priority. If the DMA controller is transferring data for the lower priority channel and afterwards the higher priority channel goes active, it completes the number of transfers delegated to the master interface by the lower priority channel before switching over to transfer data for the higher priority channel.

AHB Masters Priority in netX50 System

The netX50 SoC has four AHB masters in total (ARM926, HIF, DMA controller and XC unit). Due to the netX50 bus matrix all master can operate in parallel, if no shared resources used. If these masters come into conflict by accessing the same resources (e.g. external memory), the bus matrix solves this conflict by a fix priority for each master. The HIF interface has the highest priority. The XC unit has the second prior. The ARM926 has the third prior and the DMA controller has the lowest priority.

NetX50 Performance Considerations

The following system considerations are recommended to reduce the latency and to improve the performance of the netX50 SoC:

- Reduce conflicts a priory by separating software of the system processors (ARM926 and XC) running in different memory areas.
- If feasible, use separated memory areas for data storage and liked list information.
- All memory transactions should be 32 bits wide to improve bus efficiency.
- All external peripherals with a word size less than 32 bits must contain byte or half word packing hardware, so all transactions can be made 32 bits wide. Internal peripherals should always be access in a 32bit word length.
- Slow peripherals should contain a FIFO, so data can be transferred using burst transfers.

Interrupt Generation Logic

The DMAC controller generates a combined interrupt output as an OR function of the individual interrupt requests.

The vector interrupt controller (VIC) has an OR function of all peripherals itself and provides masking the DMA interrupt for the fast interrupt request (FIQ) and the general interrupt request (IRQ) of the ARM CPU of each interrupt source. For further information, refer register description for the DMA controller and the vector interrupt controller.

8.3 Software Interface

This chapter describes the DMA registers and provides details required for programming the DMA in the netX50 SoC.

8.3.1 Programming the DMA Controller

The DMA controller is programmed by an external AHB master through his AHB slave interface. The access to the programming registers of the DMA controller should be 32 bits wide to avoid mismatch settings, generated by the automatic packing or unpacking of data by the AHB masters (e.g. ARM926). For further details refer the register description in section Register Definition.

Enabling the DMA Controller

The DMA controller could be enabled by setting the [DMACENABLE] bit to the value 1 in the DMA configuration register (DMAC_CFG).

Disabling the DMA Controller

The DMA controller should be disabled by the following procedure:

1. Evaluate the status in the channel enable register (DMAC_CH_EN) and ensure that all **four** DMA channels are not active. If any channels are active, deactivate each channel first (see section: Disabling a DMA channel).
2. Disable the DMA controller by writing the value 0 to the [DMACENABLE] bit in the DMA configuration register (DMAC_CFG).

Enabling a DMA Channel

A DMA channel is enabled by setting the channel Enable [E] bit to value 1 in the corresponding channel configuration register ({DMAC_CH0, DMAC_CH1, DMAC_CH2, DMAC_CH3} DMAC_CH_CFG).

The channel should be initialized properly before its activation. The DMA controller should be enabled first.

Disabling a DMA Channel

To disable a DMA channel set the Enable [E] bit to value 0 in the corresponding channel configuration register ({DMAC_CH0, DMAC_CH1, DMAC_CH2, DMAC_CH3} DMAC_CH_CFG).

To avoid data loss consider the active [A] bit and the halt [H] bit information in the relevant channel configuration register or wait until the transfer is complete, so the channel will automatically be disabled.

Note: Disabling a DMA channel without data losses

To disable a DMA channel without data loss in the FIFO, follow the procedure below:

1. Set the channel Halt [**H**] bit to value 1 in the corresponding channel configuration register. This causes any subsequent DMA requests to be ignored.
2. Read the Active [A] bit in the relevant channel configuration register, until it is reset by the controller to value 0.
3. Set the enable [E] bit to value 0 in the relevant channel configuration register (DMAC_CH_CFG).

8.3.2 Programming a DMA Channel

To program a DMA channel follow the procedure below:

1. Choose a free DMA channel with the priority required (DMA channel 0 has the highest priority, and DMA channel 3 the lowest priority).
2. Clear any pending interrupts of the used channel by writing to the Interrupt Terminal Count Clear Register (DMAC_INT_TC_CLR) and writing to the Interrupt Error Clear Register (DMAC_INT_ERR_CLR) with the value 1.
3. Set the source address for the transfer data in the Source Address Registers (DMAC_CH_SRC_ADDR) of the corresponding channel.
4. Set the destination address for the transfer data in the Destination Address Registers (dmac_chdest_ad) of the corresponding channel.
5. If the transfer data has a single packet, the value 0 should be programmed into the Next Link List Item (LLI) register (DMAC_CH_LINK) of the corresponding channel. Otherwise, program the address where the next LLI is stored in the system memory. Be sure that all Link List Item are correctly set before and the last link is terminated by the value 0 to finish the linked transfer properly (see chapter Programming the DMA channel for link list transfer).
6. Set the Channel Control Registers (DMAC_CH_CTRL) of the used channel with proper control parameters.
7. Set the Channel Configuration Registers (DMAC_CH_CFG) of the used channel with proper configuration parameters and enable the corresponding DMA channel.

DMA Channel Transfer Address Generation

Each DMA channel supports an incremental or non-incremental address generation during a DMA transfer of data packets. Address wrapping is not supported and bursts do not cross the 1KB address boundary.

Building a Link List DMA transfer

The DMA controller supports a link list transfer on each channel. The benefit of a link list transfer is that source and destination areas must not have contiguous areas in memory, so the distributed data in the system memory could be collected by the DMA controller during a transfer automatically. To use this feature a special structure must be build to link the data packets. This structure is called a list of Link List Item (LLI).

Linked List Items (LLI) Structure

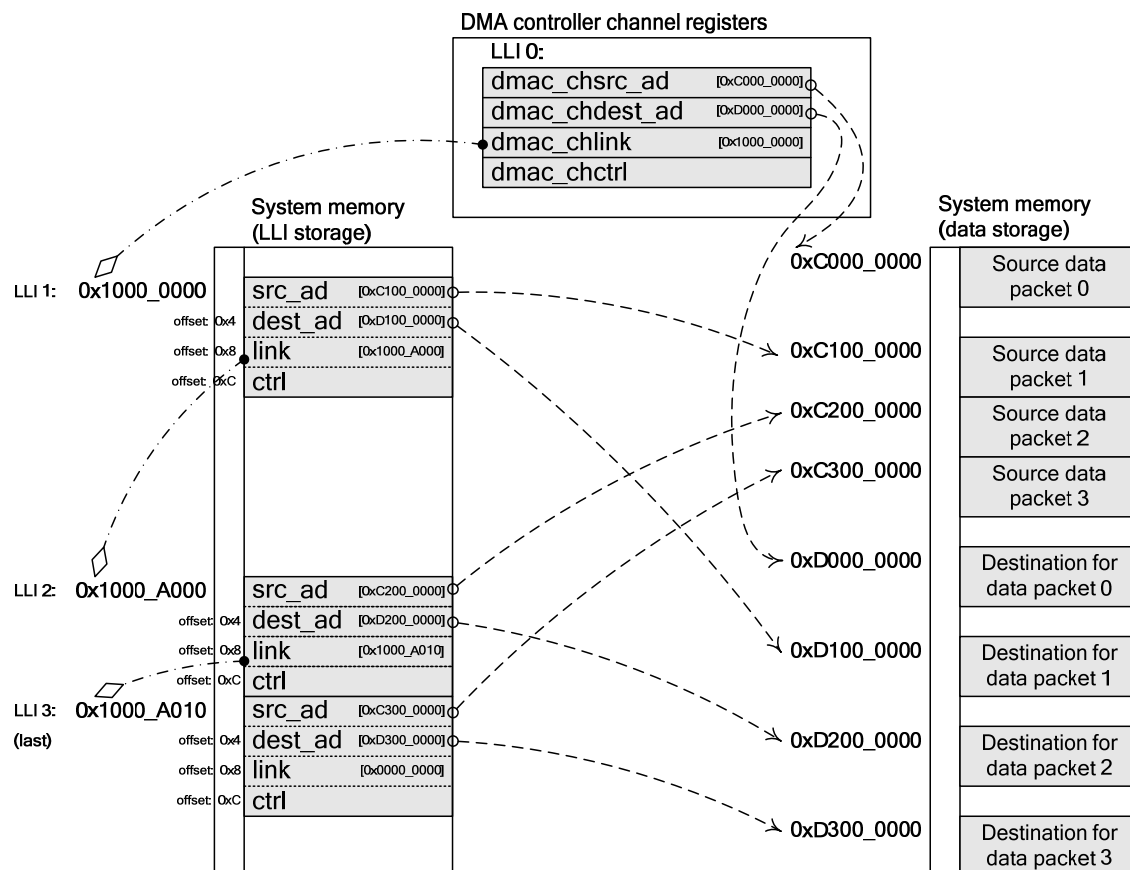
The structure of a single LLI obtains of four words (32bit x 4 data). These words are organized in the following order:

Link List Item (LLI)

dmac_chsrc_ad	1: channel source address
dmac_chdest_ad	2: channel destination address
dmac_chlink	3: channel link address to next LLI
dmac_chctrl	4: channel control information

The head (first LLI) of a link list is stored in the corresponding channel register. All other link list items are stored in the memory where the link address is pointed on.

After a complete transfer the corresponding channel of the DMA controller updates this four register with the next LLI information and starts the DMA transfer again automatically. This is resumed, until the link address to the next element has the value 0.



Note

The Channel Configuration Register (DMAC_CH_CFG) is not part of the linked list item. The settings of this register are valid for all linked list DMA transfers and should not be changed during the transfer of the active channel.

Programming a DMA Channel using a List of LLI

To program a DMA channel using a LLI structure, follow the procedure below:

1. Prepare the list of LLIs for the complete DMA transfer to the memory (as shown in the example Illustration above). Each linked list item contains four words, which must be stored contiguously in the system memory.
 - source address (offset: 0x0)
 - destination address (offset: 0x4)
 - pointer to next LLI (offset: 0x8)
 - control word (offset: 0xC)Ensure that the last LLI has the value 0 programmed in the pointer to next LLI.
2. Select an inactive DMA channel and write the first linked list item information to the relevant DMA channel registers.
3. Set the channel configuration information to the channel Configuration Register (DMAC_CH_CFG) and write the value 1 to the Channel Enable [E] bit. Ensure that the DMA controller is active.

The DMA controller starts the transfers of the first LLI, and proceeds back-to-back with each linked list item, until the last LLI element is reached. After finishing all LLI transfers, the DMA controller deactivates the channel automatically.

An interrupt can be generated after finishing the transfer of each LLI. To activate this function the Terminal count interrupt enable bit [I] should be set for the relevant LLI element of the linked list structure. If interrupts are enabled, the software should service the interrupt request by setting the relevant bit in the DMA interrupt terminal count clear register (DMAC_INT_TC_CLR). For more information refer the next section and the section "Interrupt generation logic".

8.3.3 Interrupt Requests Generation

Interrupt requests generation can be activated for an error occurred on the AHB master transfer, or for a finished transfer of the corresponding LLI settings. All interrupts of each channel could be masked separately on the corresponding bit in the channel configuration registers (DMAC_CH_CFG) and channel control registers (DMAC_CH_CTRL). The Interrupt status registers gather the requests of all channels and enable software to service the corresponding request. To find the source of an interrupt request, the software should evaluate the interrupt error status register (DMAC_INT_ERR_STAT) and/or interrupt terminal count status register (DMAC_INT_TC_STAT) depending on the active interrupts.

Evaluation procedure of DMA interrupts

1. Ensure the DMA interrupt is enabled in the interrupt enable register (VIC_INT_EN) of the vector interrupt controller and the fast interrupt (FIQ) or/and the general interrupt (IRQ) is activated in the ARM926 processor. Activate the interrupts in the channel configuration registers (DMAC_CH_CFG) and channel control registers (DMAC_CH_CTRL).
2. If an interrupt occurred, the ARM926 processor branches the programmed interrupt vector address and enters the interrupt service routine.
3. The service routine should evaluate the interrupt status register (DMAC_INT_STAT) to determine the channel that generated the interrupt. If more than one channel is active, it is recommended that to check the highest prior channel first.
4. After detecting the channel responsible for the request, the service routine should distinguish whether the interrupt request is generated due to the end of the transfer or owing to a transfer error.
5. Reset the corresponding interrupt by writing the value 1 to the relevant bit in the interrupt terminal count clear register (DMAC_INT_TC_CLR) or interrupt error clear register (DMAC_INT_ERR_CLR) and return from the interrupt service routine.

8.3.4 Data Flow Control for DMA Channel

The DMA controller supports three main data flow sequences for each channel:

- Peripheral-to-memory or memory-to-peripheral data flow control
- Peripheral-to-peripheral data flow control
- Memory-to-memory DMA data flow control

Data flow control procedure

To setup a DMA channels for a data flow control, follow the procedure below:

1. Enable the DMA controller, and program the corresponding channel configuration registers (DMAC_CH_CFG) and channel control registers (DMAC_CH_CTRL). Set the flow control bit [FlowCntrl] in the channel configuration registers (DMAC_CH_CFG) according to the data flow control requirements.
2. Wait for a DMA request by the peripheral.
3. If a DMA request is generated by the peripheral, the DMA controller starts transferring data depending on the programmed parameters (see register description for further details). If an error occurs while transferring the data, an error interrupt is generated and the DMA channel is disabled, and the data flow sequence ends. For error analyzing procedures, consider the used peripherals register descriptions (e.g. UART, IIC, SPI).
4. If the DMA controller is performing the data flow, the transfer count is automatically decremented and the transfer is finished when reaching a 0 in the transfer size bit [TransferSize] in the channel control registers (DMAC_CH_CTRL). Otherwise the transfer will be terminated by the peripheral in hardware.

8.4 Register Definition

The DMAC has four address areas for each channel (DMAC_CH0 ... DMAC_CH3) and one common address area (DMAC_REG).

8.4.1 Base Address Area: DMAC_CH

The following tables define the registers of the four DMA channels. The same registers exist for all four channels at base addresses 0x1c005100, 0x1c005120, 0x1c005140, 0x1c005160.

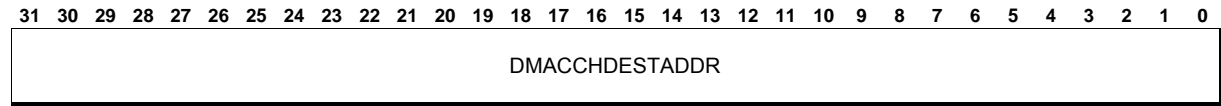
ARM Address	Register Name	Short Description
0x1c005100	DMAC_CH0_SRC_ADDR	Channel0 Source Address Registers
0x1c005104	DMAC_CH0_DEST_ADDR	Channel0 Destination Address Registers
0x1c005108	DMAC_CH0_LINK	Channel0 Linked List Item Register
0x1c00510c	DMAC_CH0_CTRL	Channel0 Control Registers
0x1c005110	DMAC_CH0_CFG	Channel0 Configuration Registers
0x1c005120	DMAC_CH1_SRC_ADDR	Channel1 Source Address Registers
0x1c005124	DMAC_CH1_DEST_ADDR	Channel1 Destination Address Registers
0x1c005128	DMAC_CH1_LINK	Channel1 Linked List Item Register
0x1c00512c	DMAC_CH1_CTRL	Channel1 Control Registers
0x1c005130	DMAC_CH1_CFG	Channel1 Configuration Registers
0x1c005140	DMAC_CH2_SRC_ADDR	Channel2 Source Address Registers
0x1c005144	DMAC_CH2_DEST_ADDR	Channel2 Destination Address Registers
0x1c005148	DMAC_CH2_LINK	Channel2 Linked List Item Register
0x1c00514c	DMAC_CH2_CTRL	Channel2 Control Registers
0x1c005150	DMAC_CH2_CFG	Channel2 Configuration Registers
0x1c005160	DMAC_CH3_SRC_ADDR	Channel3 Source Address Registers
0x1c005164	DMAC_CH3_DEST_ADDR	Channel3 Destination Address Registers
0x1c005168	DMAC_CH3_LINK	Channel3 Linked List Item Register
0x1c00516c	DMAC_CH3_CTRL	Channel3 Control Registers
0x1c005170	DMAC_CH3_CFG	Channel3 Configuration Registers

DMAC_CH0_SRC_ADDR – Channel0 Source Address Registers **0x1c005100**
DMAC_CH1_SRC_ADDR – Channel1 Source Address Registers **0x1c005120**
DMAC_CH2_SRC_ADDR – Channel2 Source Address Registers **0x1c005140**
DMAC_CH3_SRC_ADDR – Channel3 Source Address Registers **0x1c005160**

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
DMACCHSRCADDR																															

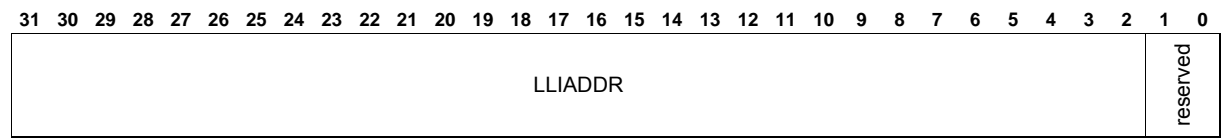
Bits	Name	Description	R/W	Default
31:0	DMACCHSRCADDR	DMA source address	R/W	0x0

DMAC_CH0_DEST_ADDR – Channel0 Destination Address Registers **0x1c005104**
DMAC_CH1_DEST_ADDR – Channel1 Destination Address Registers **0x1c005124**
DMAC_CH2_DEST_ADDR – Channel2 Destination Address Registers **0x1c005144**
DMAC_CH3_DEST_ADDR – Channel3 Destination Address Registers **0x1c005164**



Bits	Name	Description	R/W	Default
31:0	DMACCHDESTADDR	DMA destination address	R/W	0x0

DMAC_CH0_LINK – Channel0 Linked List Item Register **0x1c005108**
DMAC_CH1_LINK – Channel1 Linked List Item Register **0x1c005128**
DMAC_CH2_LINK – Channel2 Linked List Item Register **0x1c005148**
DMAC_CH3_LINK – Channel3 Linked List Item Register **0x1c005168**



Bits	Name	Description	R/W	Default
31:2	LLIADDR	Linked list item. Bits [31:2] of the address for the next LLI. Address bits [1:0] are 0.	R/W	0x0
1:0	reserved	-	R	0x0

DMAC_CH0_CTRL – Channel0 Control Registers**0x1c00510c****DMAC_CH1_CTRL – Channel1 Control Registers****0x1c00512c****DMAC_CH2_CTRL – Channel2 Control Registers****0x1c00514c****DMAC_CH3_CTRL – Channel3 Control Registers****0x1c00516c**

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
I				PROT			DI	SI	reserved	ARM_EQ		DWIDTH		SWIDTH		DBSIZE		SBSIZE													

Bits	Name	Description	R/W	Default										
31	I	Terminal count interrupt enable bit. It controls whether the current LLI is expected to trigger the terminal count interrupt.	R/W	0x0										
30:28	PROT	Protection.	R/W	0x0										
27	DI	Destination increment: When set the destination address is incremented after each transfer.	R/W	0x0										
26	SI	Source increment: When set the source address is incremented after each transfer.	R/W	0x0										
25	reserved	-	R	0x0										
24	ARM_EQ	Equal behaviour to arm implementation	R/W	0x0										
23:21	DWIDTH	<div>Destination transfer width: Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.</div> <table><thead><tr><th>bit_value</th><th>data_width</th></tr></thead><tbody><tr><td>000</td><td>8-bit</td></tr><tr><td>001</td><td>16-bit</td></tr><tr><td>010</td><td>32-bit</td></tr><tr><td>=====</td><td></td></tr></tbody></table>	bit_value	data_width	000	8-bit	001	16-bit	010	32-bit	=====		R/W	0x0
bit_value	data_width													
000	8-bit													
001	16-bit													
010	32-bit													
=====														
20:18	SWIDTH	<div>Source transfer width: Transfers wider than the AHB master bus width are illegal. The source and destination widths can be different from each other. The hardware automatically packs and unpacks the data as required.</div> <table><thead><tr><th>bit_value</th><th>data_width</th></tr></thead><tbody><tr><td>000</td><td>8-bit</td></tr><tr><td>001</td><td>16-bit</td></tr><tr><td>010</td><td>32-bit</td></tr><tr><td>=====</td><td></td></tr></tbody></table>	bit_value	data_width	000	8-bit	001	16-bit	010	32-bit	=====		R/W	0x0
bit_value	data_width													
000	8-bit													
001	16-bit													
010	32-bit													
=====														
17:15	DBSIZE	<div>Destination burst size: Indicates the number of transfers which make up a destination burst transfer request. This value must be set to the burst size of the destination peripheral, or if the destination is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACxBREQ signal goes active in the destination peripheral. The burst size is not related to the AHB HBURST signal.</div> <table><thead><tr><th>bit_value</th><th>burst_transfer_size</th></tr></thead><tbody><tr><td>000</td><td>1</td></tr><tr><td>=====</td><td></td></tr></tbody></table>	bit_value	burst_transfer_size	000	1	=====		R/W	0x0				
bit_value	burst_transfer_size													
000	1													
=====														

Bits	Name	Description	R/W	Default																		
		001 4 010 8 011 16 100 32 101 64 110 128 111 256 =====																				
14:12	SBSIZE	Source burst size: Indicates the number of transfers which make up a source burst. This value must be set to the burst size of the source peripheral, or if the source is memory, to the memory boundary size. The burst size is the amount of data that is transferred when the DMACxBREQ signal goes active in the source peripheral. The burst size is not related to the AHB HBURST signal. <table><tr><th>bit_value</th><th>burst_transfer_size</th></tr><tr><td>000</td><td>1</td></tr><tr><td>001</td><td>4</td></tr><tr><td>010</td><td>8</td></tr><tr><td>011</td><td>16</td></tr><tr><td>100</td><td>32</td></tr><tr><td>101</td><td>64</td></tr><tr><td>110</td><td>128</td></tr><tr><td>111</td><td>256</td></tr></table> =====	bit_value	burst_transfer_size	000	1	001	4	010	8	011	16	100	32	101	64	110	128	111	256	R/W	0x0
bit_value	burst_transfer_size																					
000	1																					
001	4																					
010	8																					
011	16																					
100	32																					
101	64																					
110	128																					
111	256																					
11:0	TRANSFERSIZE	Transfer size: For writes, this field indicates the number of (Source width) transfers to perform when the DMAC is the flow controller. For reads, the transfer size indicates the number of transfers completed on the destination bus. Reading the register when the channel is active does not give useful information, as by the time that the software has processed the value read, the channel might have progressed. It is intended to be used only when a channel is enabled and then disabled. If the DMAC controller is not the flow controller the transfer size value is not used.	R/W	0x0																		

DMAC_CH0_CFG – Channel0 Configuration Registers**0x1c005110****DMAC_CH1_CFG – Channel1 Configuration Registers****0x1c005130****DMAC_CH2_CFG – Channel2 Configuration Registers****0x1c005150****DMAC_CH3_CFG – Channel3 Configuration Registers****0x1c005170**

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
Reserved													H	A	L	ITC	IE	FLOWCNTRL			reserved	DestPeripheral			reserved	SrcPeripheral			U		

Bits	Name	Description	R/W	Default																											
31:19	-	Reserved	R	0x0																											
18	H	Halt: 0 = allow DMA requests 1 = ignore further source DMA requests. The contents of the channels FIFO are drained. This value can be used with the Active and Channel Enable bits to cleanly disable a DMA channel.	R/W	0x0																											
17	A	Active: 0 = there is no data in the FIFO of the channel 1 = the FIFO of the channel has data. (ro) This value can be used with the Halt and Channel Enable bits to cleanly disable a DMA channel.	R/W	0x0																											
16	L	Lock: When set this bit enables locked transfers.	R/W	0x0																											
15	ITC	Terminal count interrupt mask: When cleared this bit masks out the terminal count interrupt of the relevant channel.	R/W	0x0																											
14	IE	Interrupt error mask: When cleared this bit masks out the error interrupt of the relevant channel.	R/W	0x0																											
13:11	FLOWCNTRL	Flow control and transfer type: This value is used to indicate the flow controller and transfer type. The flow controller can be the DMAC, the source peripheral, or the destination peripheral. The transfer type can be either memory-to-memory, memory-to-peripheral, peripheral-to-memory, or peripheral-to-peripheral. <table><tr><th>bit_value</th><th>transfer_type</th><th>controller</th></tr><tr><td>000</td><td>Memory-to-memory</td><td>DMAC</td></tr><tr><td>001</td><td>Memory-to-peripheral</td><td>DMAC</td></tr><tr><td>010</td><td>Peripheral-to-memorys</td><td>DMAC</td></tr><tr><td>011</td><td>Source peripheral-to-destination peripheral</td><td>DMAC</td></tr><tr><td>100</td><td>Source peripheral-to-destination peripheral</td><td>peripheral</td></tr><tr><td>101</td><td>Memory-to-peripheral Peripheral</td><td>peripheral</td></tr><tr><td>110</td><td>Peripheral-to-memory</td><td>peripheral</td></tr><tr><td>111</td><td>Source peripheral-to-destination peripheral</td><td>peripheral</td></tr></table> =====	bit_value	transfer_type	controller	000	Memory-to-memory	DMAC	001	Memory-to-peripheral	DMAC	010	Peripheral-to-memorys	DMAC	011	Source peripheral-to-destination peripheral	DMAC	100	Source peripheral-to-destination peripheral	peripheral	101	Memory-to-peripheral Peripheral	peripheral	110	Peripheral-to-memory	peripheral	111	Source peripheral-to-destination peripheral	peripheral	R/W	0x0
bit_value	transfer_type	controller																													
000	Memory-to-memory	DMAC																													
001	Memory-to-peripheral	DMAC																													
010	Peripheral-to-memorys	DMAC																													
011	Source peripheral-to-destination peripheral	DMAC																													
100	Source peripheral-to-destination peripheral	peripheral																													
101	Memory-to-peripheral Peripheral	peripheral																													
110	Peripheral-to-memory	peripheral																													
111	Source peripheral-to-destination peripheral	peripheral																													
10	reserved	-	R	0x0																											

Bits	Name	Description	R/W	Default																																		
9:6	DestPeripheral	<p>Destination peripheral: This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is memory.</p> <table><thead><tr><th>select_value</th><th>module</th></tr></thead><tbody><tr><td>0</td><td>spi0_rx</td></tr><tr><td>1</td><td>spi0_tx</td></tr><tr><td>2</td><td>spi1_rx</td></tr><tr><td>3</td><td>spi2_tx</td></tr><tr><td>4</td><td>uart0_rx</td></tr><tr><td>5</td><td>uart0_tx</td></tr><tr><td>6</td><td>uart1_rx</td></tr><tr><td>7</td><td>uart1_tx</td></tr><tr><td>8</td><td>uart2_rx</td></tr><tr><td>9</td><td>uart3_tx</td></tr><tr><td>10</td><td>i2c_master_mode</td></tr><tr><td>11</td><td>i2c_slave_mode</td></tr><tr><td>12</td><td>ccdc_fifo0</td></tr><tr><td>13</td><td>ccdc_fifo1</td></tr><tr><td>14</td><td>ccdc_fifo2</td></tr><tr><td>12-15</td><td>not used</td></tr></tbody></table>	select_value	module	0	spi0_rx	1	spi0_tx	2	spi1_rx	3	spi2_tx	4	uart0_rx	5	uart0_tx	6	uart1_rx	7	uart1_tx	8	uart2_rx	9	uart3_tx	10	i2c_master_mode	11	i2c_slave_mode	12	ccdc_fifo0	13	ccdc_fifo1	14	ccdc_fifo2	12-15	not used	R/W	0x0
select_value	module																																					
0	spi0_rx																																					
1	spi0_tx																																					
2	spi1_rx																																					
3	spi2_tx																																					
4	uart0_rx																																					
5	uart0_tx																																					
6	uart1_rx																																					
7	uart1_tx																																					
8	uart2_rx																																					
9	uart3_tx																																					
10	i2c_master_mode																																					
11	i2c_slave_mode																																					
12	ccdc_fifo0																																					
13	ccdc_fifo1																																					
14	ccdc_fifo2																																					
12-15	not used																																					
5	reserved	-	R	0x0																																		
4:1	SrcPeripheral	<p>Source peripheral: This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is memory.</p> <table><thead><tr><th>select_value</th><th>module</th></tr></thead><tbody><tr><td>0</td><td>spi0_rx</td></tr><tr><td>1</td><td>spi0_tx</td></tr><tr><td>2</td><td>spi1_rx</td></tr><tr><td>3</td><td>spi2_tx</td></tr><tr><td>4</td><td>uart0_rx</td></tr><tr><td>5</td><td>uart0_tx</td></tr><tr><td>6</td><td>uart1_rx</td></tr><tr><td>7</td><td>uart1_tx</td></tr><tr><td>8</td><td>uart2_rx</td></tr><tr><td>9</td><td>uart3_tx</td></tr><tr><td>10</td><td>i2c_master_mode</td></tr><tr><td>11</td><td>i2c_slave_mode</td></tr><tr><td>12</td><td>ccdc_fifo0</td></tr><tr><td>13</td><td>ccdc_fifo1</td></tr><tr><td>14</td><td>ccdc_fifo2</td></tr><tr><td>12-15</td><td>not used</td></tr></tbody></table>	select_value	module	0	spi0_rx	1	spi0_tx	2	spi1_rx	3	spi2_tx	4	uart0_rx	5	uart0_tx	6	uart1_rx	7	uart1_tx	8	uart2_rx	9	uart3_tx	10	i2c_master_mode	11	i2c_slave_mode	12	ccdc_fifo0	13	ccdc_fifo1	14	ccdc_fifo2	12-15	not used	R/W	0x0
select_value	module																																					
0	spi0_rx																																					
1	spi0_tx																																					
2	spi1_rx																																					
3	spi2_tx																																					
4	uart0_rx																																					
5	uart0_tx																																					
6	uart1_rx																																					
7	uart1_tx																																					
8	uart2_rx																																					
9	uart3_tx																																					
10	i2c_master_mode																																					
11	i2c_slave_mode																																					
12	ccdc_fifo0																																					
13	ccdc_fifo1																																					
14	ccdc_fifo2																																					
12-15	not used																																					
0	E	<p>Channel enable: Reading this bit indicates whether a channel is currently enabled (1) or disabled (0). The Channel Enable bit status can also be found by reading the DMACEnbldChns register. A channel is enabled by setting this bit. A channel can be disabled by clearing the Enable bit, which causes the current AHB transfer (if one is in progress) to complete. The channel is then disabled and any data in the channels FIFO is lost. Restarting the channel by simply setting the Channel Enable bit has unpredictable effects and the channel must be fully re-initialized. The channel is also disabled, and Channel Enable bit cleared, when the last LLI is reached or if a channel error is encountered. If a channel has to be disabled without losing data in a channels FIFO, the Halt bit must be set so that further DMA requests are ignored. The Active bit must then be polled until it reaches 0, indicating there is no data left in the channels FIFO. Finally the Channel Enable bit can be cleared.</p>	R/W	0x0																																		

8.4.2 Base Address Area: DMAC_REG

ARM Address	Register Name	Short Description
0x1c005800	DMAC_INT_STAT	Interrupt Status Register
0x1c005804	DMAC_INT_TC_STAT	Interrupt Terminal Count Status Register
0x1c005808	DMAC_INT_TC_CLR	Interrupt Terminal Count Clear Register
0x1c00580c	DMAC_INT_ERR_STAT	Interrupt Error Status Register
0x1c005810	DMAC_INT_ERR_CLR	Interrupt Error Clear Register
0x1c005814	DMAC_RAW_INT_TC_STAT	Raw Interrupt Terminal Count Status Register
0x1c005818	DMAC_RAW_INT_ERR_STAT	Raw Interrupt Error Status Register
0x1c00581c	DMAC_CH_EN	Channel Enable Register
0x1c005820	DMAC_SW_BURST_REQ	Software Burst Request Register
0x1c005824	DMAC_SW_SINGLE_REQ	Software Single Request Register
0x1c005828	DMAC_SW_LAST_BURST_REQ	Software Last Burst Request Register
0x1c00582c	DMAC_SW_LAST_SINGLE_REQ	Software Last Single Request Register
0x1c005830	DMAC_CFG	Configuration Register
0x1c005834	DMAC_SYNC	Sync Register
0x1c005838	DMAC_RNG_CTRL	Random Number Generation Control Register
0x1c00583c	DMAC_RNG_SEED	Random Number Generation Seed Register
0x1c005840	DMAC_RNG_NUM	Random Number Generation Data Register

DMAC_INT_STAT - Interrupt Status Register**0x1c005800**

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
reserved																												DMACINT_ch3	DMACINT_ch2	DMACINT_ch1	DMACINT_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACINT_ch3	Status of DMA channel 3 - interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
2	DMACINT_ch2	Status of DMA channel 2 - interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
1	DMACINT_ch1	Status of DMA channel 1 - interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
0	DMACINT_ch0	Status of DMA channel 0 - interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0

DMAC_INT_TC_STAT - Interrupt Terminal Count Status Register**0x1c005804**

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
reserved																												DMACINTTC_ch3	DMACINTTC_ch2	DMACINTTC_ch1	DMACINTTC_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACINTTC_ch3	Status of DMA channel 3 - terminal count interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
2	DMACINTTC_ch2	Status of DMA channel 2 - terminal count interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
1	DMACINTTC_ch1	Status of DMA channel 1 - terminal count interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
0	DMACINTTC_ch0	Status of DMA channel 0 - terminal count interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0

DMAC_INT_TC_CLR - Interrupt Terminal Count Clear Register**0x1c005808**

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
reserved																												DMACINTTCCLR_ch3	DMACINTTCCLR_ch2	DMACINTTCCLR_ch1	DMACINTTCCLR_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACINTTCCLR_ch3	Writing a 1'b1 Bit clears the terminal count interrupt of the specific channel 3 ,1'b0 have no effect.	W	0x0
2	DMACINTTCCLR_ch2	Writing a 1'b1 Bit clears the terminal count interrupt of the specific channel 2 ,1'b0 have no effect.	W	0x0
1	DMACINTTCCLR_ch1	Writing a 1'b1 Bit clears the terminal count interrupt of the specific channel 1 ,1'b0 have no effect.	W	0x0
0	DMACINTTCCLR_ch0	Writing a 1'b1 Bit clears the terminal count interrupt of the specific channel 0 ,1'b0 have no effect.	W	0x0

DMAC_INT_ERR_STAT - Interrupt Error Status Register**0x1c00580c**

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
reserved																												DMACINTERR_ch3	DMACINTERR_ch2	DMACINTERR_ch1	DMACINTERR_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACINTERR_ch3	Status of DMA channel 3 - error interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
2	DMACINTERR_ch2	Status of DMA channel 2 - error interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
1	DMACINTERR_ch1	Status of DMA channel 1 - error interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0
0	DMACINTERR_ch0	Status of DMA channel 0 - error interrupt after masking. 1'b1 indicates an active interrupt request.	R	0x0

DMAC_INT_ERR_CLR - Interrupt Error Clear Register**0x1c005810**

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
reserved																												DMACINTERRCLR_ch3	DMACINTERRCLR_ch2	DMACINTERRCLR_ch1	DMACINTERRCLR_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACINTERRCLR_ch3	Writing a 1'b1 Bit clears the error interrupt of the specific channel 3 ,1'b0 have no effect.	W	0x0
2	DMACINTERRCLR_ch2	Writing a 1'b1 Bit clears the error interrupt of the specific channel 2 ,1'b0 have no effect.	W	0x0
1	DMACINTERRCLR_ch1	Writing a 1'b1 Bit clears the error interrupt of the specific channel 1 ,1'b0 have no effect.	W	0x0
0	DMACINTERRCLR_ch0	Writing a 1'b1 Bit clears the error interrupt of the specific channel 0 ,1'b0 have no effect.	W	0x0

DMAC_RAW_INT_TC_STAT - Raw Interrupt Terminal Count Status Register**0x1c005814**

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
reserved																												DMACRAWINTTC_ch3	DMACRAWINTTC_ch2	DMACRAWINTTC_ch1	DMACRAWINTTC_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACRAWINTTC_ch3	Status of DMA channel 3 - terminal count interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
2	DMACRAWINTTC_ch2	Status of DMA channel 2 - terminal count interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
1	DMACRAWINTTC_ch1	Status of DMA channel 1 - terminal count interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
0	DMACRAWINTTC_ch0	Status of DMA channel 0 - terminal count interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0

DMAC_RAW_INT_ERR_STAT - Raw Interrupt Error Status Register**0x1c005818**

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
reserved																												DMACRAWINTERR_ch3	DMACRAWINTERR_ch2	DMACRAWINTERR_ch1	DMACRAWINTERR_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACRAWINTERR_ch3	Status of DMA channel 3 - error interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
2	DMACRAWINTERR_ch2	Status of DMA channel 2 - error interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
1	DMACRAWINTERR_ch1	Status of DMA channel 1 - error interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0
0	DMACRAWINTERR_ch0	Status of DMA channel 0 - error interrupt prior to masking. 1'b1 indicates an active interrupt request.	R	0x0

DMAC_CH_EN - Channel Enable Register**0x1c00581c**

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
reserved																												DMACENABLEDCHNS_ch3	DMACENABLEDCHNS_ch2	DMACENABLEDCHNS_ch1	DMACENABLEDCHNS_ch0

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	DMACENABLEDCHNS_ch3	Status DMA channel 3 enable	R	0x0
2	DMACENABLEDCHNS_ch2	Status DMA channel 2 enable	R	0x0
1	DMACENABLEDCHNS_ch1	Status DMA channel 1 enable	R	0x0
0	DMACENABLEDCHNS_ch0	Status DMA channel 0 enable	R	0x0

DMAC_SW_BURST_REQ - Software Burst Request Register**0x1c005820**

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
reserved																DMACSoftBReq															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	DMACSoftBReq	Software burst request. A DMA request can be generated for each source by writing a 1'b1 to the corresponding register bit. Reading the register indicates which sources are requesting DMA burst transfers.	R/W	0x0

DMAC_SW_SINGLE_REQ - Software Single Request Register**0x1c005824**

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
reserved																DMACSoftSReq															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	DMACSoftSReq	Software single request. A DMA request can be generated for each source by writing a 1'b1 to the corresponding register bit. Reading the register indicates which sources are requesting DMA single transfers.	R/W	0x0

DMAC_SW_LAST_BURST_REQ - Software Last Burst Request Register**0x1c005828**

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
reserved																DMACSoftLBReq															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	DMACSoftLBReq	Software last burst request. A DMA request can be generated for each source by writing a 1'b1 to the corresponding register bit. Reading the register indicates which sources are requesting DMA last burst transfers.	R/W	0x0

DMAC_SW_LAST_SINGLE_REQ - Software Last Single Request Register**0x1c00582c**

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
reserved																DMACSoftLSReq															

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	DMACSoftLSReq	Software last single request. A DMA request can be generated for each source by writing a 1'b1 to the corresponding register bit. Reading the register indicates which sources are requesting DMA last single transfers.	R/W	0x0

DMAC_CFG - Configuration Register**0x1c005830**

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
reserved																															DMAC ENABLE

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	DMACENABLE	DMAC enable: 0 = disabled 1 = enabled. This bit is reset to 0. Disabling the DMAC reduces power consumption.	R/W	0x0

DMAC_SYNC - Sync Register**0x1c005834**

DMA synchronization logic for DMA request signals is enabled or disabled in this register. A 0-bit indicates that the synchronization logic for the DMACBREQ[15:0], DMACSREQ[15:0], DMACLBREQ[15:0], and DMACLSREQ[15:0] request signals is enabled. A 1-bit indicates that the synchronization logic is disabled.

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
Reserved																						DIS_SYNC_UART2_TX	DIS_SYNC_UART2_RX	DIS_SYNC_UART1_TX	DIS_SYNC_UART1_RX	DIS_SYNC_UART0_TX	DIS_SYNC_UART0_RX	DIS_SYNC_SPI1_TX	DIS_SYNC_SPI1_RX	DIS_SYNC_SPI0_TX	DIS_SYNC_SPI0_RX

Bits	Name	Description	R/W	Default
31:10	-	Reserved	R	0x0
9	DIS_SYNC_UART2_TX	disable sync register for UART2 transmit requests	R/W	0x0
8	DIS_SYNC_UART2_RX	disable sync register for UART2 receive requests	R/W	0x0
7	DIS_SYNC_UART1_TX	disable sync register for UART1 transmit requests	R/W	0x0
6	DIS_SYNC_UART1_RX	disable sync register for UART1 receive requests	R/W	0x0
5	DIS_SYNC_UART0_TX	disable sync register for UART0 transmit requests	R/W	0x0
4	DIS_SYNC_UART0_RX	disable sync register for UART0 receive requests	R/W	0x0
3	DIS_SYNC_SPI1_TX	disable sync register for SPI1 transmit requests	R/W	0x0
2	DIS_SYNC_SPI1_RX	disable sync register for SPI1 receive requests	R/W	0x0
1	DIS_SYNC_SPI0_TX	disable sync register for SPI0 transmit requests	R/W	0x0
0	DIS_SYNC_SPI0_RX	disable sync register for SPI0 receive requests	R/W	0x0

0x1c005838

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
reserved																															LOAD_SEED

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	LOAD_SEED	Use new seed value	R/W	0x0

0x1c00583c

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
SEED																															

Bits	Name	Description	R/W	Default
31:0	SEED	Seed value	R/W	0x0

0x1c005840

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
RNG_NUM																															

Bits	Name	Description	R/W	Default
31:0	RNG_NUM	Random number	R/W	0x0

9 ARM System Control and Configuration Registers

These registers which are called CP15 registers are accessible using special ARM instructions.

CP15 registers enable configuration of the Tightly-Coupled Memory (TCM) and write buffer and other ARM966E-S system options such as big-endian or little-endian operation.

For more details about the TCM and other system options see ARM966E-S Technical Reference Manual.

9.1 CP15 Registers Summary

The ARM966E-S coprocessor 15 registers are described in the following sections:

- c0 ID Code Register, TCM Size Register
- c1 Control Register
- c7 Core Control Register
- c13 Trace Process Identifier Register

Note:

1. Register c0 provides access to more than one register. The register access depends on the value of the Opcode_2 field. See the register descriptions in the following section for more information.
2. c2-c6, c8-c12 and c14 are reserved.

9.2 CP15 Registers Description

c0 – ID Code Register

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
IMPLEMENTER								Major revision				ARCHITECTURE				PART NUMBER								Minor revision							

Bits	Name	Description	R/W	Default
31:24	IMPLEMENTER	ASCII code of implementer trademark	R	0x41
23:20	Major revision	Major specification revision	R	0x2
19:16	ARCHITECTURE	Architecture (ARMv5TE)	R	0x5
15:4	PART NUMBER	Part number	R	0x966
3:0	Minor revision	Minor specification revision	R	0x1

This is a read-only register that returns the 32-bit device ID code.

You can access the ID Code Register by reading CP15 register c0 with the Opcode_2 field set to any value other than 2. For example:

```
MRC p15,0,<Rd>,c0,c0,0 ;returns ID Code Register
```

c0 – TCM Size Register

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
reserved									DTCM_size				reserved			DTCM_absent	reserved				ITCM_size				reserved			ITCM_absent	reserved		

Bits	Name	Description	R/W	Default
31:23	reserved	-	R	0x00
22:18	DTCM_size	<p>The data TCM size is configurable in the range 0 bytes to 64 MB:</p> <p>b00000 = 0 byte b00001 = 1KB b00010 = 2KB b00011 = 4KB b00100 = 8KB b00101 = 16KB b00110 = 32KB b00111 = 64KB b01000 = 128KB b01001 = 256KB b01010 = 512KB b01011 = 1 MB b01100 = 2 MB b01101 = 4 MB b01110 = 8 MB b01111 = 16 MB b10000 = 32 MB b10001 = 64 MB</p> <p>The supported sizes are 0 and 2nKB for n = 0 to 16</p>	R	0x04
17:15	reserved	-	R	0x00
14	DTCM_absent	1 : absent 0: present	R	0
13:11	reserved	-	R	0x00
10:6	ITCM_size	<p>The instruction TCM size is configurable in the range 0 bytes to 64 MB:</p> <p>b00000 = 0 byte b00001 = 1KB b00010 = 2KB b00011 = 4KB b00100 = 8KB b00101 = 16KB b00110 = 32KB b00111 = 64KB b01000 = 128KB b01001 = 256KB b01010 = 512KB b01011 = 1 MB b01100 = 2 MB b01101 = 4 MB b01110 = 8 MB b01111 = 16 MB b10000 = 32 MB b10001 = 64 MB</p> <p>The supported sizes are 0 and 2nKB for n = 0 to 16</p>	R	0x04
5:3	reserved	-	R	0x00
2	ITCM_absent	1 : absent 0: present	R	0
1:0	reserved	-	R	0x00

The TCM Size Register is a read-only register that returns the size of the Instruction and Data TCM attached to the ARM966E-S processor.

The TCM Size Register can be accessed by reading CP15 register c0 with the Opcode_2 field set to 2. For example:

```
MRC p15,0,<Rd>,c0,c0,2    ;returns Tightly-Coupled Memory Size Register
```

c1 – Control Register

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
SBZ																LT	SBZ	V	I	SBO			B	SBO		W	D	A	SBZ		

Bits	Name	Description	R/W	Default
31:16	SBZ(should be zero)	Reserved. When read, returns an unpredictable value. When written, should be zero.	R	0x00
15	LT	Load PC Thumb disable bit: 0 = loading PC sets the T bit 1 = loading PC does not set the T bit (ARMv4 behavior). Reset clears this bit	R/W	0
14	SBZ(should be zero)	Reserved. When read, returns an Unpredictable value. When written, should be zero.	R/W	0
13	V	Location of exception vectors: 0 = Normal exception vectors selected, address range = 0x0000 0000 to 0x0000 001C 1 = High exception vectors selected, address range = 0xFFFF 0000 to 0xFFFF 001C . At Reset, the VINITHI pins determine the value of this bit.	R/W	0
12	I	Instruction TCM enable: 0 = All accesses to the instruction memory space access the AMBA AHB 1 = All accesses to the fixed instruction memory space access the instruction TCM interface. At Reset, the INITRAM pins determine the value of this bit.	R/W	0
11:8	SBO(should be one)	Reserved. Should be One.	R	00
7	B	Endianness: 0 = Little-endian operation 1 = Big-endian operation.	R/W	0
6:4	SBO(should be one)	Reserved. Should be One.	R	0
3	W	BIU write buffer enable: 0 = All stores to the AMBA AHB are treated as nonbufferable 1 = All stores to the fixed bufferable space of the AMBA AHB are treated as buffered writes. Reset clears this bit.	R/W	0
2	D	Data TCM enable. At Reset, the INITRAM pins determine the value of this bit.	R/W	0
1	A	Address alignment fault checking enable: 0 = Fault checking of address alignment disabled 1 = Fault checking of address alignment enabled. Reset clears this bit.	R/W	0
0	SBZ(should be zero)	Reserved. When read, returns an Unpredictable value. When written, should be zero.	R	0

This register contains the global control bits of the ARM966E-S processor. All reserved bits must either be written with zero or one, as indicated, or written using read-modify-write. The reserved bits have an Unpredictable value when read. To read and write this register, use the instructions:

```
MRC p15,0,<Rd>,c1,c0,0    ;read control register
MCR p15,0,<Rd>,c1,c0,0    ;write control register
```

c7 – Core Control Register

You can use a write to this register, to perform wait for interrupt and drain write buffer operations.

Wait for interrupt

This operation enables the ARM966E-S processor to enter a low-power standby mode. When the operation is invoked, the clock enable to the processor core is negated until either an interrupt or a debug request occurs. This function is invoked by a write to Register 7. The following ARM instruction causes this to occur:

```
MCR p15,0,<Rd>,c7,c0,4    ;wait for interrupt
```

Note:

This is the preferred encoding that must be used by new software. For compatibility with existing software, ARM966E-S processor also supports the following ARM instruction that has the same affect:

```
MCR p15,0,<Rd>,c15,c8,2    ;wait for interrupt
```

This stalls the processor from the time that the instruction is executed until **nFIQ**, **nIRQ**, or **EDBGRQ** are asserted. Also, if the debugger sets the debug request bit in the EmbeddedICE-RT control register then this causes the wait-for-interrupt condition to terminate.

In the case of **nFIQ** and **nIRQ**, the processor core is woken up regardless of whether the interrupts are enabled or disabled (that is, independent of the I and F bits in the processor CPSR). The debug-related waking only occurs if **DBGEN** is HIGH, that is, only when debug is enabled.

If interrupts are enabled, the ARM9E-S core is guaranteed to take the interrupt before executing the instruction after the wait for interrupt. If debug request is used to wake up the system, the processor enters debug-state before executing any more instructions.

Wait for interrupt does not prevent the write buffer from emptying.

Drain write buffers

This CP15 operation causes instruction execution to be stalled until the AHB and TCM write buffers are emptied. This operation is useful in real-time applications where the processor must be sure that a write to a peripheral has completed before program execution continues. An example is where a peripheral in a bufferable region is the source of an interrupt. When the interrupt has been serviced, the request must be removed before interrupts can be re-enabled. This can be ensured if a drain write buffer operation separates the store to the peripheral and the enable interrupt functions.

The drain write buffer operation is invoked by a write to Register 7 using the following ARM instruction:

```
MCR p15,0,<Rd>,c7,c10,4    ;drain write buffer
```

Note:

This stalls the processor core until any outstanding accesses in the write buffers have been completed, that is, until all data has been written to memory.

C13 – Trace Process Identifier Register

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
Trace process ID																															

Bits	Name	Description	R/W	Default
31:0	Trace process ID	Trace process ID	R/W	0x00

This register enables the real-time trace tools to identify the currently executing process in multitasking environments.

The **ETMPROCID**[31:0] pins reflect the contents of the Trace Process Identifier Register.

Note:

Writing to the Trace Process Identifier Register sets the **ETMPROCIDWR** signal for one clock cycle.

Here are ARM instructions that you can use to access the Trace Process Identifier Register:

```

MRC p15,0,<Rd>,c13,{c0-c15}    ;Read Trace Process Identifier Register
MCR p15,0,<Rd>,c13,{c0-c15}    ;write Trace Process Identifier Register

```

10 Appendix A: Register Table

ARM Address	Register Name	Short Description
0x1c000000	BOO_STAT	Bond Out Option Status Register
0x1c000004	IO_CFG	IO Configuration Register
0x1c000008	IO_CFG_MSK	IO Configuration Mask Register
0x1c00000c	RESET_CTRL	Reset Control Register
0x1c000034	NETX_REV	netX Revision Register
0x1c000070	IO_CFG_ACCESS_KEY	IO Configuration Access Key Register
0x1c000200	WDG_TRIG	Watchdog Trigger Register
0x1c000204	WDG_CNTR	Watchdog Counter
0x1c000208	WDG_IRQ_TIMEOUT	Watchdog Interrupt Timeout
0x1c00020c	WDG_RESET_TIMEOUT	Watchdog Reset Timeout
0x1c0034d8	SYS_STAT	System Status
0x1c000100	MEM_SRAM0_CTRL	Memory SRAM Control Register for Chip Select Area 0
0x1c000104	MEM_SRAM1_CTRL	Memory SRAM Control Register for Chip Select Area 1
0x1c000108	MEM_SRAM2_CTRL	Memory SRAM Control Register for Chip Select Area 2
0x1c000140	MEM_SDRAM_CFG_CTRL	Memory SDRAM Configuration Control Register
0x1c000144	MEM_SDRAM_TIMING_CTRL	Memory SDRAM Timing Control Register
0x1c000148	MEM_SDRAM_MODE	Memory SDRAM Mode Register
0x1c00014c	MEM_SDRAM_EXT_MODE	Memory SDRAM Extended Mode Register
0x1c000180	MEM_PRIO_TIMESLOT_CTRL	Memory Priority Timeslot Control Register
0x1c000184	MEM_PRIO_ACCESS_CTRL	Memory Priority Access Control Register
0x1c003610	EXT_CFG_CS0	Extension Bus Configuration Chip Select 0
0x1c003614	EXT_CFG_CS1	Extension Bus Configuration Chip Select 1
0x1c003618	EXT_CFG_CS2	Extension Bus Configuration Chip Select 2
0x1c00361c	EXT_CFG_CS3	Extension Bus Configuration Chip Select 3
0x1c0036bc	DPM_ARM_HS_CTRL15	Handshake Control Register 15
0x1c0036b8	DPM_ARM_HS_CTRL14	Handshake Control Register 14
0x1c0036b4	DPM_ARM_HS_CTRL13	Handshake Control Register 13
0x1c0036b0	DPM_ARM_HS_CTRL12	Handshake Control Register 12
0x1c0036ac	DPM_ARM_HS_CTRL11	Handshake Control Register 11
0x1c0036a8	DPM_ARM_HS_CTRL10	Handshake Control Register 10
0x1c0036a4	DPM_ARM_HS_CTRL9	Handshake Control Register 9
0x1c0036a0	DPM_ARM_HS_CTRL8	Handshake Control Register 8
0x1c00369c	DPM_ARM_HS_CTRL7	Handshake Control Register 7
0x1c003698	DPM_ARM_HS_CTRL6	Handshake Control Register 6
0x1c003694	DPM_ARM_HS_CTRL5	Handshake Control Register 5
0x1c003690	DPM_ARM_HS_CTRL4	Handshake Control Register 4
0x1c00368c	DPM_ARM_HS_CTRL3	Handshake Control Register 3
0x1c003688	DPM_ARM_HS_CTRL2	Handshake Control Register 2
0x1c003684	DPM_ARM_HS_CTRL1	Handshake Control Register 1
0x1c003680	DPM_ARM_HS_CTRL0	Handshake Control Register 0
0x1c00367c	DPM_ARM_DB_MAP7	Data Block Mapping Address Register 7

0x1c003678	DPM_ARM_DB_END7	Data Block 7 End Address
0x1c003674	DPM_ARM_DB_MAP6	Data Block Mapping Address Register 6
0x1c003670	DPM_ARM_DB_END6	Data Block 6 End Address
0x1c00366c	DPM_ARM_DB_MAP5	Data Block Mapping Address Register 5
0x1c003668	DPM_ARM_DB_END5	Data Block 5 End Address
0x1c003664	DPM_ARM_DB_MAP4	Data Block Mapping Address Register 4
0x1c003660	DPM_ARM_DB_END4	Data Block 4 End Address
0x1c00365c	DPM_ARM_DB_MAP3	Data Block Mapping Address Register 3
0x1c003658	DPM_ARM_DB_END3	Data Block 3 End Address
0x1c003654	DPM_ARM_DB_MAP2	Data Block Mapping Address Register 2
0x1c003650	DPM_ARM_DB_END2	Data Block 2 End Address
0x1c00364c	DPM_ARM_DB_MAP1	Data Block Mapping Address Register 1
0x1c003648	DPM_ARM_DB_END1	Data Block 1 End Address
0x1c003644	DPM_ARM_DB_MAP0	Data Block Mapping Address Register 0
0x1c003640	DPM_ARM_DB_END0	Data Block 0 End Address
0x1c003638	DPM_ARM_IO_DATA1	Input / Output Data Register 1
0x1c003634	DPM_ARM_IO_DRV_EN1	Input / Output Driver Enable Register 1
0x1c003630	DPM_ARM_IO_MODE1	Input / Output Mode Register 1
0x1c003628	DPM_ARM_IO_DATA0	Input / Output Data Register 0
0x1c003624	DPM_ARM_IO_DRV_EN0	Input / Output Driver Enable Register 0
0x1c003620	DPM_ARM_IO_MODE0	Input / Output Mode Register 0
0x1c00360c	DPM_ARM_IF_CFG1	Interface Configuration Register 1
0x1c003608	DPM_ARM_IF_CFG0	Interface Configuration Register 0
0x1c003604	DPM_ARM_CLKOUT_CFG	Clockout Configuration Register
0x1c00353c	DPM_ARM_HS_DATA15	Handshake Data Register 15
0x1c003538	DPM_ARM_HS_DATA14	Handshake Data Register 14
0x1c003534	DPM_ARM_HS_DATA13	Handshake Data Register 13
0x1c003530	DPM_ARM_HS_DATA12	Handshake Data Register 12
0x1c00352c	DPM_ARM_HS_DATA11	Handshake Data Register 11
0x1c003528	DPM_ARM_HS_DATA10	Handshake Data Register 10
0x1c003524	DPM_ARM_HS_DATA9	Handshake Data Register 9
0x1c003520	DPM_ARM_HS_DATA8	Handshake Data Register 8
0x1c00351c	DPM_ARM_HS_DATA7	Handshake Data Register 7
0x1c003518	DPM_ARM_HS_DATA6	Handshake Data Register 6
0x1c003514	DPM_ARM_HS_DATA5	Handshake Data Register 5
0x1c003510	DPM_ARM_HS_DATA4	Handshake Data Register 4
0x1c00350c	DPM_ARM_HS_DATA3	Handshake Data Register 3
0x1c003508	DPM_ARM_HS_DATA2	Handshake Data Register 2
0x1c003504	DPM_ARM_HS_DATA1	Handshake Data Register 1
0x1c003500	DPM_ARM_HS_DATA0	Handshake Data Register 0
0x1c0034f0	DPM_ARM_INT_EN0	Interrupt Enable 0 Register
0x1c0034e0	DPM_ARM_INT_STAT0	Interrupt Status 0 Register
0x1c0034d8	DPM_ARM_SYS_STAT	System Status Register
0x1c0034cc	DPM_ARM_WDG_ARM_TRIG	Watchdog Trigger ARM
0x1c0034c8	DPM_ARM_WDG_ARM_TIMEOUT	Watchdog Timeout ARM, read only
0x1c0034c0	DPM_ARM_WDG_HOST_TIMEOUT	Watchdog Timeout Host

0x1c0034bc	DPM_ARM_CIS_MAP	Card Information Structure Mapping Address
0x1c00323c	DPM_HOST_HS_DATA15	Handshake Data Register 15
0x1c003238	DPM_HOST_HS_DATA14	Handshake Data Register 14
0x1c003234	DPM_HOST_HS_DATA13	Handshake Data Register 13
0x1c003230	DPM_HOST_HS_DATA12	Handshake Data Register 12
0x1c00322c	DPM_HOST_HS_DATA11	Handshake Data Register 11
0x1c003228	DPM_HOST_HS_DATA10	Handshake Data Register 10
0x1c003224	DPM_HOST_HS_DATA9	Handshake Data Register 9
0x1c003220	DPM_HOST_HS_DATA8	Handshake Data Register 8
0x1c00321c	DPM_HOST_HS_DATA7	Handshake Data Register 7
0x1c003218	DPM_HOST_HS_DATA6	Handshake Data Register 6
0x1c003214	DPM_HOST_HS_DATA5	Handshake Data Register 5
0x1c003210	DPM_HOST_HS_DATA4	Handshake Data Register 4
0x1c00320c	DPM_HOST_HS_DATA3	Handshake Data Register 3
0x1c003208	DPM_HOST_HS_DATA2	Handshake Data Register 2
0x1c003204	DPM_HOST_HS_DATA1	Handshake Data Register 1
0x1c003200	DPM_HOST_HS_DATA0	Handshake Data Register 0
0x1c0031f0	DPM_HOST_INT_EN0	Interrupt Enable 0
0x1c0031e0	DPM_HOST_INT_STAT0	Interrupt Status 0
0x1c0031dc	DPM_HOST_RESET_REQ	Reset Request
0x1c0031d8	DPM_HOST_SYS_STAT	System Status
0x1c0031d4	DPM_HOST_TMR_START_VAL	Timer Start Value
0x1c0031d0	DPM_HOST_TMR_CTRL	Timer Control
0x1c0031c8	DPM_HOST_WDG_ARM_TIMEOUT	Watchdog ARM Timeout
0x1c0031c4	DPM_HOST_WDG_HOST_TRIG	Watchdog Host Trigger
0x1c0031c0	DPM_HOST_WDG_HOST_TIMEOUT	Watchdog Host Timeout, read only
0x1c000800	GPIO_CFG0	GPIO 0 Configuration Register
0x1c000804	GPIO_CFG1	GPIO 1 Configuration Register
0x1c000808	GPIO_CFG2	GPIO 2 Configuration Register
0x1c00080c	GPIO_CFG3	GPIO 3 Configuration Register
0x1c000810	GPIO_CFG4	GPIO 4 Configuration Register
0x1c000814	GPIO_CFG5	GPIO 5 Configuration Register
0x1c000818	GPIO_CFG6	GPIO 6 Configuration Register
0x1c00081c	GPIO_CFG7	GPIO 7 Configuration Register
0x1c000820	GPIO_CFG8	GPIO 8 Configuration Register
0x1c000824	GPIO_CFG9	GPIO 9 Configuration Register
0x1c000828	GPIO_CFG10	GPIO 10 Configuration Register
0x1c00082c	GPIO_CFG11	GPIO 11 Configuration Register
0x1c000830	GPIO_CFG12	GPIO 12 Configuration Register
0x1c000834	GPIO_CFG13	GPIO 13 Configuration Register
0x1c000838	GPIO_CFG14	GPIO 14 Configuration Register
0x1c00083c	GPIO_CFG15	GPIO 15 Configuration Register
0x1c000840	GPIO_CFG16	GPIO 16 Configuration Register
0x1c000844	GPIO_CFG17	GPIO 17 Configuration Register
0x1c000848	GPIO_CFG18	GPIO 18 Configuration Register
0x1c00084c	GPIO_CFG19	GPIO 19 Configuration Register

0x1c000850	GPIO_CFG20	GPIO 20 Configuration Register
0x1c000854	GPIO_CFG21	GPIO 21 Configuration Register
0x1c000858	GPIO_CFG22	GPIO 22 Configuration Register
0x1c00085c	GPIO_CFG23	GPIO 23 Configuration Register
0x1c000860	GPIO_CFG24	GPIO 24 Configuration Register
0x1c000864	GPIO_CFG25	GPIO 25 Configuration Register
0x1c000868	GPIO_CFG26	GPIO 26 Configuration Register
0x1c00086c	GPIO_CFG27	GPIO 27 Configuration Register
0x1c000870	GPIO_CFG28	GPIO 28 Configuration Register
0x1c000874	GPIO_CFG29	GPIO 29 Configuration Register
0x1c000878	GPIO_CFG30	GPIO 30 Configuration Register
0x1c00087c	GPIO_CFG31	GPIO 31 Configuration Register
0x1c000880	GPIO_THRSH_CAPT0	GPIO 0 Threshold or Capture Register
0x1c000884	GPIO_THRSH_CAPT1	GPIO 1 Threshold or Capture Register
0x1c000888	GPIO_THRSH_CAPT2	GPIO 2 Threshold or Capture Register
0x1c00088c	GPIO_THRSH_CAPT3	GPIO 3 Threshold or Capture Register
0x1c000890	GPIO_THRSH_CAPT4	GPIO 4 Threshold or Capture Register
0x1c000894	GPIO_THRSH_CAPT5	GPIO 5 Threshold or Capture Register
0x1c000898	GPIO_THRSH_CAPT6	GPIO 6 Threshold or Capture Register
0x1c00089c	GPIO_THRSH_CAPT7	GPIO 7 Threshold or Capture Register
0x1c0008a0	GPIO_THRSH_CAPT8	GPIO 8 Threshold or Capture Register
0x1c0008a4	GPIO_THRSH_CAPT9	GPIO 9 Threshold or Capture Register
0x1c0008a8	GPIO_THRSH_CAPT10	GPIO 10 Threshold or Capture Register
0x1c0008ac	GPIO_THRSH_CAPT11	GPIO 11 Threshold or Capture Register
0x1c0008b0	GPIO_THRSH_CAPT12	GPIO 12 Threshold or Capture Register
0x1c0008b4	GPIO_THRSH_CAPT13	GPIO 13 Threshold or Capture Register
0x1c0008b8	GPIO_THRSH_CAPT14	GPIO 14 Threshold or Capture Register
0x1c0008bc	GPIO_THRSH_CAPT15	GPIO 15 Threshold or Capture Register
0x1c0008c0	GPIO_THRSH_CAPT16	GPIO 16 Threshold or Capture Register
0x1c0008c4	GPIO_THRSH_CAPT17	GPIO 17 Threshold or Capture Register
0x1c0008c8	GPIO_THRSH_CAPT18	GPIO 18 Threshold or Capture Register
0x1c0008cc	GPIO_THRSH_CAPT19	GPIO 19 Threshold or Capture Register
0x1c0008d0	GPIO_THRSH_CAPT20	GPIO 20 Threshold or Capture Register
0x1c0008d4	GPIO_THRSH_CAPT21	GPIO 21 Threshold or Capture Register
0x1c0008d8	GPIO_THRSH_CAPT22	GPIO 22 Threshold or Capture Register
0x1c0008dc	GPIO_THRSH_CAPT23	GPIO 23 Threshold or Capture Register
0x1c0008e0	GPIO_THRSH_CAPT24	GPIO 24 Threshold or Capture Register
0x1c0008e4	GPIO_THRSH_CAPT25	GPIO 25 Threshold or Capture Register
0x1c0008e8	GPIO_THRSH_CAPT26	GPIO 26 Threshold or Capture Register
0x1c0008ec	GPIO_THRSH_CAPT27	GPIO 27 Threshold or Capture Register
0x1c0008f0	GPIO_THRSH_CAPT28	GPIO 28 Threshold or Capture Register
0x1c0008f4	GPIO_THRSH_CAPT29	GPIO 29 Threshold or Capture Register
0x1c0008f8	GPIO_THRSH_CAPT30	GPIO 30 Threshold or Capture Register
0x1c0008fc	GPIO_THRSH_CAPT31	GPIO 31 Threshold or Capture Register
0x1c000900	GPIO_CNTR0_CTRL	GPIO counter 0 control register
0x1c000904	GPIO_CNTR1_CTRL	GPIO counter 1 control register

0x1c000908	GPIO_CNTR2_CTRL	GPIO counter 2 control register
0x1c00090c	GPIO_CNTR3_CTRL	GPIO counter 3 control register
0x1c000910	GPIO_CNTR4_CTRL	GPIO counter 4 control register
0x1c000914	GPIO_CNTR0_MAX	GPIO counter 0 max values
0x1c000918	GPIO_CNTR1_MAX	GPIO counter 1 max values
0x1c00091c	GPIO_CNTR2_MAX	GPIO counter 2 max values
0x1c000920	GPIO_CNTR3_MAX	GPIO counter 3 max values
0x1c000924	GPIO_CNTR4_MAX	GPIO counter 4 max values
0x1c000928	GPIO_CNTR0_CNT	GPIO counter 0 current value
0x1c00092c	GPIO_CNTR1_CNT	GPIO counter 1 current value
0x1c000930	GPIO_CNTR2_CNT	GPIO counter 2 current value
0x1c000934	GPIO_CNTR3_CNT	GPIO counter 3 current value
0x1c000938	GPIO_CNTR4_CNT	GPIO counter 4 current value
0x1c00093c	GPIO_SYSTIME_NS_CMP	GPIO System Time NS Compare Value
0x1c000940	GPIO_OUT	GPIO Output Register
0x1c000944	GPIO_IN	GPIO Input Register
0x1c000948	GPIO_IRQ_RAW	GPIO raw IRQ register
0x1c00094c	GPIO_IRQ_MSK	GPIO Masked IRQ register
0x1c000950	GPIO_IRQ_MSK_SET	GPIO Interrupt Request Mask Enable
0x1c000954	GPIO_IRQ_MSK_RESET	GPIO Interrupt Request Mask Disable
0x1c000958	CNTR_IRQ_RAW	Counter raw IRQ register
0x1c00095c	CNTR_IRQ_MSK	Counter Masked IRQ register
0x1c000960	CNTR_IRQ_MSK_SET	Counter Interrupt Request Mask Enable
0x1c000964	CNTR_IRQ_MSK_RESET	Counter Interrupt Request Mask Disable
0x1c000880	IOLINK0_CFG	IO-Link 0 Configuration Register
0x1c000884	IOLINK0_TX_FRM0	IO-Link 0 TX0 Data Register
0x1c000888	IOLINK0_TX_FRM1	IO-Link 0 TX1 Data Register
0x1c00088c	IOLINK0_RX_FRM	IO-Link 0 RX Data Register
0x1c000890	IOLINK1_CFG	IO-Link 1 Configuration Register
0x1c000894	IOLINK1_TX_FRM0	IO-Link 1 TX0 Data Register
0x1c000898	IOLINK1_TX_FRM1	IO-Link 1 TX1 Data Register
0x1c00089c	IOLINK1_RX_FRM	IO-Link 1 RX Data Register
0x1c0008a0	IOLINK2_CFG	IO-Link 2 Configuration Register
0x1c0008a4	IOLINK2_TX_FRM0	IO-Link 2 TX0 Data Register
0x1c0008a8	IOLINK2_TX_FRM1	IO-Link 2 TX1 Data Register
0x1c0008ac	IOLINK2_RX_FRM	IO-Link 2 RX Data Register
0x1c0008b0	IOLINK3_CFG	IO-Link 3 Configuration Register
0x1c0008b4	IOLINK3_TX_FRM0	IO-Link 3 TX0 Data Register
0x1c0008b8	IOLINK3_TX_FRM1	IO-Link 3 TX1 Data Register
0x1c0008bc	IOLINK3_RX_FRM	IO-Link 3 RX Data Register
0x1c0008c0	IOLINK4_CFG	IO-Link 4 Configuration Register
0x1c0008c4	IOLINK4_TX_FRM0	IO-Link 4 TX0 Data Register
0x1c0008c8	IOLINK4_TX_FRM1	IO-Link 4 TX1 Data Register
0x1c0008cc	IOLINK4_RX_FRM	IO-Link 4 RX Data Register
0x1c0008d0	IOLINK5_CFG	IO-Link 5 Configuration Register
0x1c0008d4	IOLINK5_TX_FRM0	IO-Link 5 TX0 Data Register

0x1c0008d8	IOLINK5_TX_FRM1	IO-Link 5 TX1 Data Register
0x1c0008dc	IOLINK5_RX_FRM	IO-Link 5 RX Data Register
0x1c0008e0	IOLINK6_CFG	IO-Link 6 Configuration Register
0x1c0008e4	IOLINK6_TX_FRM0	IO-Link 6 TX0 Data Register
0x1c0008e8	IOLINK6_TX_FRM1	IO-Link 6 TX1 Data Register
0x1c0008ec	IOLINK6_RX_FRM	IO-Link 6 RX Data Register
0x1c0008f0	IOLINK7_CFG	IO-Link 7 Configuration Register
0x1c0008f4	IOLINK7_TX_FRM0	IO-Link 7 TX0 Data Register
0x1c0008f8	IOLINK7_TX_FRM1	IO-Link 7 TX1 Data Register
0x1c0008fc	IOLINK7_RX_FRM	IO-Link 7 RX Data Register
0x1c000a00	PIO_IN	PIO Input Register
0x1c000a04	PIO_OUT	PIO Output Register
0x1c000a08	PIO_OUT_EN	PIO Output Enable Register
0x1c000b00	UART0_DATA	UART0 Data Register
0x1c000b04	UART0_STAT	UART0 Status Register
0x1c000b08	UART0_LINE_CTRL	UART0 Line Control Register
0x1c000b0c	UART0_BAUD_DIV_MSB	UART0 Baud Rate Divisor MSB
0x1c000b10	UART0_BAUD_DIV_LSB	UART0 Baud Rate Divisor LSB
0x1c000b14	UART0_CTRL	UART0 Control Register
0x1c000b18	UART0_FLAG	UART0 Flag Register
0x1c000b1c	UART0_INT_ID	UART0 Interrupt Identification Register
0x1c000b20	UART0_IRDA_LO_PWR_CNTR	UART0 IrDA Low Power Counter Register
0x1c000b24	UART0_RTS_CTRL	UART0 RTS Control Register
0x1c000b28	UART0_RTS_LEAD_CYC	UART0 RTS Leading Cycles
0x1c000b2c	UART0_RTS_TRAIL_CYC	UART0 RTS Trailing cycles
0x1c000b30	UART0_OUT_DRV_EN	UART0 UART Output Driver Enable Register
0x1c000b34	UART0_BAUD_MODE_CTRL	UART0 Baud Rate Mode Control Register
0x1c000b38	UART0_RX_FIFO_IRQ_LVL	UART0 RX FIFO Trigger Level and RX-DMA Enable
0x1c000b3c	UART0_TX_FIFO_IRQ_LVL	UART0 TX FIFO Trigger Level and TX-DMA Enable
0x1c000b40	UART1_DATA	UART1 Data Register
0x1c000b44	UART1_STAT	UART1 Status Register
0x1c000b48	UART1_LINE_CTRL	UART1 Line Control Register
0x1c000b4c	UART1_BAUD_DIV_MSB	UART1 Baud Rate Divisor MSB
0x1c000b50	UART1_BAUD_DIV_LSB	UART1 Baud Rate Divisor LSB
0x1c000b54	UART1_CTRL	UART1 Control Register
0x1c000b58	UART1_FLAG	UART1 Flag Register
0x1c000b5c	UART1_INT_ID	UART1 Interrupt Identification Register
0x1c000b60	UART1_IRDA_LO_PWR_CNTR	UART1 IrDA Low Power Counter Register
0x1c000b64	UART1_RTS_CTRL	UART1 RTS Control Register
0x1c000b68	UART1_RTS_LEAD_CYC	UART1 RTS Leading Cycles
0x1c000b6c	UART1_RTS_TRAIL_CYC	UART1 RTS Trailing cycles
0x1c000b70	UART1_OUT_DRV_EN	UART1 UART Output Driver Enable Register
0x1c000b74	UART1_BAUD_MODE_CTRL	UART1 Baud Rate Mode Control Register
0x1c000b78	UART1_RX_FIFO_IRQ_LVL	UART1 RX FIFO Trigger Level and RX-DMA Enable
0x1c000b7c	UART1_TX_FIFO_IRQ_LVL	UART1 TX FIFO Trigger Level and TX-DMA Enable
0x1c000b80	UART2_DATA	UART2 Data Register

0x1c000b84	UART2_STAT	UART2 Status Register
0x1c000b88	UART2_LINE_CTRL	UART2 Line Control Register
0x1c000b8c	UART2_BAUD_DIV_MSB	UART2 Baud Rate Divisor MSB
0x1c000b90	UART2_BAUD_DIV_LSB	UART2 Baud Rate Divisor LSB
0x1c000b94	UART2_CTRL	UART2 Control Register
0x1c000b98	UART2_FLAG	UART2 Flag Register
0x1c000b9c	UART2_INT_ID	UART2 Interrupt Identification Register
0x1c000ba0	UART2_IRDA_LO_PWR_CNTR	UART2 IrDA Low Power Counter Register
0x1c000ba4	UART2_RTS_CTRL	UART2 RTS Control Register
0x1c000ba8	UART2_RTS_LEAD_CYC	UART2 RTS Leading Cycles
0x1c000bac	UART2_RTS_TRAIL_CYC	UART2 RTS Trailing cycles
0x1c000bb0	UART2_OUT_DRV_EN	UART2 UART Output Driver Enable Register
0x1c000bb4	UART2_BAUD_MODE_CTRL	UART2 Baud Rate Mode Control Register
0x1c000bb8	UART2_RX_FIFO_IRQ_LVL	UART2 RX FIFO Trigger Level and RX-DMA Enable
0x1c000bbc	UART2_TX_FIFO_IRQ_LVL	UART2 TX FIFO Trigger Level and TX-DMA Enable
0x1c000d00	SPI0_CTRL0	SPI0 Control Register 0
0x1c000d04	SPI0_CTRL1	SPI0 Control Register 1
0x1c000d08	SPI0_DATA	SPI0 Data Register
0x1c000d0c	SPI0_STAT	SPI0 Status Register
0x1c000d10	SPI0_CLK_PRE_SCL	SPI0 Clock Prescale Register
0x1c000d14	SPI0_INT_MSK_SET_CLR	SPI0 Interrupt Mask Set or Clear
0x1c000d18	SPI0_RAW_INT_STAT	SPI0 RAW Interrupt Status Register
0x1c000d1c	SPI0_MSK_INT_STAT	SPI0 Mask Interrupt Status Register
0x1c000d20	SPI0_INT_CLR	SPI0 Interrupt Clear Register
0x1c000d24	SPI0_DMA_CTRL	SPI0 DMA Control Register
0x1c000d40	SPI1_CTRL0	SPI1 Control Register 0
0x1c000d44	SPI1_CTRL1	SPI1 Control Register 1
0x1c000d48	SPI1_DATA	SPI1 Data Register
0x1c000d4c	SPI1_STAT	SPI1 Status Register
0x1c000d50	SPI1_CLK_PRE_SCL	SPI1 Clock Prescale Register
0x1c000d54	SPI1_INT_MSK_SET_CLR	SPI1 Interrupt Mask Set or Clear
0x1c000d58	SPI1_RAW_INT_STAT	SPI1 RAW Interrupt Status Register
0x1c000d5c	SPI1_MSK_INT_STAT	SPI1 Mask Interrupt Status Register
0x1c000d60	SPI1_INT_CLR	SPI1 Interrupt Clear Register
0x1c000d64	SPI1_DMA_CTRL	SPI1 DMA Control Register
0x1c000e00	I2C_MASTER_CTRL	I2C Master Control Register
0x1c000e04	I2C_SLAVE_CTRL	I2C Slave Control Register
0x1c000e08	I2C_MASTER_CMD	I2C Master Command Register
0x1c000e0c	I2C_MASTER_DATA	I2C Master Data Register
0x1c000e10	I2C_SLAVE_DATA	I2C Slave Data Register
0x1c000e14	I2C_MASTER_FIFO_CTRL	I2C Master FIFO Control Register
0x1c000e18	I2C_SLAVE_FIFO_CTRL	I2C Slave FIFO Control Register
0x1c000e1c	I2C_STAT	I2C Status Register
0x1c000e20	I2C_INT_MSK_SET_CLR	I2C Interrupt Mask set or clear Register
0x1c000e24	I2C_RAW_INT_STAT	I2C RAW Interrupt Status Register
0x1c000e28	I2C_MSK_INT_STAT	I2C Mask Interrupt Status Register

0x1c000e2c	I2C_DMA_CTRL	I2C DMA Control Register
0x1c000f00	CCDC_CFG	CCDC Configuration Register
0x1c000f04	CCDC_HORIZ_START_STOP	CCDC Horizontal Start/Stop Values
0x1c000f08	CCDC_VERT_START_STOP	CCDC Vertical Start/Stop Values
0x1c000f0c	CCDC_HORIZ_VERT_CNTR	CCDC Horizontal/Vertical Counter
0x1c000f10	CCDC_BRIGHT	CCDC Brightness Counter
0x1c000f14	CCDC_FIFO0	CCDC FIFO 0
0x1c000f18	CCDC_FIFO1	CCDC FIFO 1
0x1c000f1c	CCDC_FIFO2	CCDC FIFO 2
0x1c000f20	CCDC_BYTE0_POS	CCDC Byte 0 Position Register
0x1c000f24	CCDC_BYTE1_POS	CCDC Byte 1 Position Register
0x1c000f28	CCDC_BYTE2_POS	CCDC Byte 2 Position Register
0x1c001100	SYS_TIME_NS	System Time Nanosecond Register
0x1c001104	SYS_TIME_S	System Time Second Register
0x1c001108	SYS_TIME_NS_BOR	System Time Nanoseconds Border Register
0x1c00110c	SYS_TIME_NS_ADD_UP	System Time Nanoseconds Add Up Register
0x1c001110	SYS_TIME_S_CMP	System Time Second Compare Register
0x1c001114	SYS_TIME_S_CMP_EN	System Time Second Compare Enable Register
0x1c001118	SYS_TIME_S_CMP_INT	System Time Second Compare Interrupt Register
0x1c001300	MMIO0_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO0
0x1c001304	MMIO1_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO1
0x1c001308	MMIO2_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO2
0x1c00130c	MMIO3_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO3
0x1c001310	MMIO4_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO4
0x1c001314	MMIO5_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO5
0x1c001318	MMIO6_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO6
0x1c00131c	MMIO7_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO7
0x1c001320	MMIO8_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO8
0x1c001324	MMIO9_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO9
0x1c001328	MMIO10_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO10
0x1c00132c	MMIO11_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO11
0x1c001330	MMIO12_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO12
0x1c001334	MMIO13_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO13
0x1c001338	MMIO14_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO14
0x1c00133c	MMIO15_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO15
0x1c001340	MMIO16_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO16
0x1c001344	MMIO17_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO17
0x1c001348	MMIO18_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO18
0x1c00134c	MMIO19_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO19
0x1c001350	MMIO20_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO20
0x1c001354	MMIO21_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO21
0x1c001358	MMIO22_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO22
0x1c00135c	MMIO23_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO23
0x1c001360	MMIO24_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO24
0x1c001364	MMIO25_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO25

0x1c001368	MMIO26_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO26
0x1c00136c	MMIO27_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO27
0x1c001370	MMIO28_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO28
0x1c001374	MMIO29_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO29
0x1c001378	MMIO30_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO30
0x1c00137c	MMIO31_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO31
0x1c001380	MMIO32_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO32
0x1c001384	MMIO33_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO33
0x1c001388	MMIO34_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO34
0x1c00138c	MMIO35_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO35
0x1c001390	MMIO36_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO36
0x1c001394	MMIO37_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO37
0x1c001398	MMIO38_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO38
0x1c00139c	MMIO39_CFG	IO-Multiplexmatrix Configuration Register For Signal MMIO39
0x1c020000	USB_ID	USB ID Register
0x1c020004	USB_CTRL	USB Control Register
0x1c020008	USB_FRM_TMR	USB Frame Timer Register
0x1c02000c	USB_MAIN_EV	USB Main Event Register
0x1c020010	USB_MAIN_EV_MSK	USB Main Event Mask Register
0x1c020014	USB_PIPE_EV	USB Pipe Event Register
0x1c020018	USB_PIPE_EV_MSK	USB Pipe Event Mask Register
0x1c020024	USB_PIPE_SEL	USB Pipe Select Register
0x1c02002c	USB_PORT_STAT	USB Port Status Register
0x1c020030	USB_PORT_CTRL	USB Port Control Register
0x1c020034	USB_PORT_STAT_CHG_EV	USB Port Status Change Event Register
0x1c020038	USB_PORT_STAT_CHG_EV_MSK	USB Port Status Change Event Mask Register
0x1c020040	USB_PIPE_CTRL	USB Pipe Control Register
0x1c020044	USB_PIPE_CFG	USB Pipe Configuration Register
0x1c020048	USB_PIPE_ADDR	USB Pipe Address Register
0x1c02004c	USB_PIPE_STAT	USB Pipe Status Register
0x1c020050	USB_PIPE_DATA_PTR	USB Pipe Data Pointer Register
0x1c020054	USB_PIPE_DATA_TOT	USB Pipe Total Bytes Register
0x1c020058	USB_PIPE_ALT_DATA_PTR	USB Pipe Alternative Data Pointer Register
0x1c02005c	USB_PIPE_ALT_DATA_TOT	USB Pipe Alternative Data Total Bytes Register
0x1c020060	USB_DBG_CTRL	USB Debug Control Register
0x1c020064	USB_DBG_PID	USB Debug PID Register
0x1c020068	USB_DBG_STAT	USB Debug Status Register
0x1c02006c	USB_TEST	USB Test Register
0x1c020080	USB_MAIN_CFG	USB Main Configuration Register
0x1c020084	USB_MODE_CFG	USB Mode Configuration Register
0x1c020088	USB_CORE_CTRL	USB Core Control and Status Register
0x1c030000	USB_FIFO	FIFO for USB-Interface
0x1c7ff000	VIC_IRQ_STAT	IRQ Status Register
0x1c7ff004	VIC_FIQ_STAT	FIQ Status Register
0x1c7ff008	VIC_RAW_INT_STAT	Raw Interrupt Status Register
0x1c7ff00c	VIC_INT_SEL	Interrupt Select Register

0x1c7ff010	VIC_INT_EN	Interrupt Enable Register
0x1c7ff014	VIC_INT_EN_CLR	Interrupt Enable Clear Register
0x1c7ff018	VIC_SWI	Software Interrupt Register
0x1c7ff01c	VIC_SWI_CLR	Software Interrupt Clear Register
0x1c7ff020	VIC_PROT_EN	Protection Enable Register
0x1c7ff030	VIC_VECT_ADDR	Vector Address Register
0x1c7ff034	VIC_DFLT_VECT_ADDR	Default Vector Address Register
0x1c7ff100	VIC_VECT_ADDR0	Vector Address Register 0
0x1c7ff104	VIC_VECT_ADDR1	Vector Address Register 1
0x1c7ff108	VIC_VECT_ADDR2	Vector Address Register 2
0x1c7ff10c	VIC_VECT_ADDR3	Vector Address Register 3
0x1c7ff110	VIC_VECT_ADDR4	Vector Address Register 4
0x1c7ff114	VIC_VECT_ADDR5	Vector Address Register 5
0x1c7ff118	VIC_VECT_ADDR6	Vector Address Register 6
0x1c7ff11c	VIC_VECT_ADDR7	Vector Address Register 7
0x1c7ff120	VIC_VECT_ADDR8	Vector Address Register 8
0x1c7ff124	VIC_VECT_ADDR9	Vector Address Register 9
0x1c7ff128	VIC_VECT_ADDR10	Vector Address Register 10
0x1c7ff12c	VIC_VECT_ADDR11	Vector Address Register 11
0x1c7ff130	VIC_VECT_ADDR12	Vector Address Register 12
0x1c7ff134	VIC_VECT_ADDR13	Vector Address Register 13
0x1c7ff138	VIC_VECT_ADDR14	Vector Address Register 14
0x1c7ff13c	VIC_VECT_ADDR15	Vector Address Register 15
0x1c7ff200	VIC_VECT_CTRL0	Vector Control Register 0
0x1c7ff204	VIC_VECT_CTRL1	Vector Control Register 1
0x1c7ff208	VIC_VECT_CTRL2	Vector Control Register 2
0x1c7ff20c	VIC_VECT_CTRL3	Vector Control Register 3
0x1c7ff210	VIC_VECT_CTRL4	Vector Control Register 4
0x1c7ff214	VIC_VECT_CTRL5	Vector Control Register 5
0x1c7ff218	VIC_VECT_CTRL6	Vector Control Register 6
0x1c7ff21c	VIC_VECT_CTRL7	Vector Control Register 7
0x1c7ff220	VIC_VECT_CTRL8	Vector Control Register 8
0x1c7ff224	VIC_VECT_CTRL9	Vector Control Register 9
0x1c7ff228	VIC_VECT_CTRL10	Vector Control Register 10
0x1c7ff22c	VIC_VECT_CTRL11	Vector Control Register 11
0x1c7ff230	VIC_VECT_CTRL12	Vector Control Register 12
0x1c7ff234	VIC_VECT_CTRL13	Vector Control Register 13
0x1c7ff238	VIC_VECT_CTRL14	Vector Control Register 14
0x1c7ff23c	VIC_VECT_CTRL15	Vector Control Register 15
0x1c000010	PHY_CTRL	PHY Control Register
0x1c000020	PHY_CLK_RATE_MUL_ADD	PHY Clock Rate Multiplier Add Value
0x1c064000	PTR_FIFO_BASE	Pointer FIFO Base Address
0x1c064080	PTR_FIFO_BOR_BASE	Pointer FIFO Upper Border Base Address
0x1c064100	PTR_FIFO_RESET	Pointer FIFO Reset Vector
0x1c064104	PTR_FIFO_FULL	Pointer FIFO Full Vector
0x1c064108	PTR_FIFO_EMPTY	Pointer FIFO Empty Vector

0x1c06410c	PTR_FIFO_OVF	Pointer FIFO Overflow Vector
0x1c064110	PTR_FIFO_UDR	Pointer FIFO Under-Run Vector
0x1c064180	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Fill Level Base Address
0x1c001000	CRC_VAL	Calculated CRC Value
0x1c001004	CRC_IN_DATA	CRC Input Data
0x1c001008	CRC_POLYNOMIAL	Polynomial for CRC Calculation
0x1c00100c	CRC_CFG	CRC Configuration Register
0x1c064400	IRQ_XP0	IRQs between XPEC0 and ARM
0x1c064404	IRQ_XP1	IRQs between XPEC1 and ARM
0x1c005100	DMAC_CH0_SRC_ADDR	Channel0 Source Address Registers
0x1c005104	DMAC_CH0_DEST_ADDR	Channel0 Destination Address Registers
0x1c005108	DMAC_CH0_LINK	Channel0 Linked List Item Register
0x1c00510c	DMAC_CH0_CTRL	Channel0 Control Registers
0x1c005110	DMAC_CH0_CFG	Channel0 Configuration Registers
0x1c005120	DMAC_CH1_SRC_ADDR	Channel1 Source Address Registers
0x1c005124	DMAC_CH1_DEST_ADDR	Channel1 Destination Address Registers
0x1c005128	DMAC_CH1_LINK	Channel1 Linked List Item Register
0x1c00512c	DMAC_CH1_CTRL	Channel1 Control Registers
0x1c005130	DMAC_CH1_CFG	Channel1 Configuration Registers
0x1c005140	DMAC_CH2_SRC_ADDR	Channel2 Source Address Registers
0x1c005144	DMAC_CH2_DEST_ADDR	Channel2 Destination Address Registers
0x1c005148	DMAC_CH2_LINK	Channel2 Linked List Item Register
0x1c00514c	DMAC_CH2_CTRL	Channel2 Control Registers
0x1c005150	DMAC_CH2_CFG	Channel2 Configuration Registers
0x1c005160	DMAC_CH3_SRC_ADDR	Channel3 Source Address Registers
0x1c005164	DMAC_CH3_DEST_ADDR	Channel3 Destination Address Registers
0x1c005168	DMAC_CH3_LINK	Channel3 Linked List Item Register
0x1c00516c	DMAC_CH3_CTRL	Channel3 Control Registers
0x1c005170	DMAC_CH3_CFG	Channel3 Configuration Registers
0x1c005800	DMAC_INT_STAT	Interrupt Status Register
0x1c005804	DMAC_INT_TC_STAT	Interrupt Terminal Count Status Register
0x1c005808	DMAC_INT_TC_CLR	Interrupt Terminal Count Clear Register
0x1c00580c	DMAC_INT_ERR_STAT	Interrupt Error Status Register
0x1c005810	DMAC_INT_ERR_CLR	Interrupt Error Clear Register
0x1c005814	DMAC_RAW_INT_TC_STAT	Raw Interrupt Terminal Count Status Register
0x1c005818	DMAC_RAW_INT_ERR_STAT	Raw Interrupt Error Status Register
0x1c00581c	DMAC_CH_EN	Channel Enable Register
0x1c005820	DMAC_SW_BURST_REQ	Software Burst Request Register
0x1c005824	DMAC_SW_SINGLE_REQ	Software Single Request Register
0x1c005828	DMAC_SW_LAST_BURST_REQ	Software Last Burst Request Register
0x1c00582c	DMAC_SW_LAST_SINGLE_REQ	Software Last Single Request Register
0x1c005830	DMAC_CFG	Configuration Register
0x1c005834	DMAC_SYNC	Sync Register
0x1c005838	DMAC_RNG_CTRL	Random Number Generation Control Register
0x1c00583c	DMAC_RNG_SEED	Random Number Generation Seed Register
0x1c005840	DMAC_RNG_NUM	Random Number Generation Data Register

11 Revision History

Rev.	Date	Changes	Who
1.0	2008-11-27	released	JL
1.1	2015-04-16	Chapter 2.2: Changed the Register Address of NETX_REV_HW – netX Hardware Revision Register Chapter 4: Timing Trdwrcyc corrected.	AJ HH