Program Reference Guide

netX10

netX Generation of Communication Controllers

V1.0

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC111111PRG01EN | Revision 1 | English | 2012-01 | Preliminary | Public

Table of Contents

1 Introduction

1.1 About this Document

This manual describes all available registers of the netX10. You will find the register addresses and the bit descriptions.

1.2 List of Revisions

Rev	Date	Name	Chapter	Revision
1	2010-06-21	AJ	all	Created
1.0	2012-01-26	JZ	all	DPM, VIC, SQI etc modified.

Table 1: List of Revisions

1.3 Terms, Abbreviations and Definitions

Term	Description
AP (-task)	Application (-task) on top of the stack
ARP	Address Resolution Protocol
BOOTP	Bootstrap Protocol
DHCP	Dynamic Host Configuration Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
MSS	Maximum segment size (of TCP data), normally = 1460 byte on Ethernet (Maximum); MSS = MTU - sizeof(IP header) - sizeof (TCP header) = 1500 –20 –20 = 1460
MTU	Maximum Transmission Unit, normally 1500 byte = Data part of Ethernet frame
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Table 2: Terms, Abbreviations and Definitions

All variables, parameters, and data used in this manual have the LSB/MSB ("Intel") data format. This corresponds to the convention of the Microsoft C Compiler.
All IP addresses in this document have host byte order.

1.4 References

This document based on the following specification:

[1] netX10 Product Brief
[2] netX10 Technical Reference Guide
[3] netX10_regdef.html

Table 3: References

1.5 Legal Notes

1.5.1 Copyright

1.5.2 Important Notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.
We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.
Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

1.5.3 Exclusion of Liability

The software was produced and tested with utmost care by Hilscher Gesellschaft für Systemautomation mbH and is made available as is. No warranty can be assumed for the performance and flawlessness of the software for all usage conditions and cases and for the results produced when utilized by the user. Liability for any damages that may result from the use of the hardware or software or related documents, is limited to cases of intent or grossly negligent violation of significant contractual obligations. Indemnity claims for the violation of significant contractual obligations are limited to damages that are foreseeable and typical for this type of contract.

It is strictly prohibited to use the software in the following areas:

- for military purposes or in weapon systems;
- for the design, construction, maintenance or operation of nuclear facilities;
- in air traffic control systems, air traffic or air traffic communication systems;
- in life support systems;
- in systems in which failures in the software could lead to personal injury or injuries leading to death.

We inform you that the software was not developed for use in dangerous environments requiring fail-proof control mechanisms. Use of the software in such an environment occurs at your own risk. No liability is assumed for damages or losses due to unauthorized use.

1.5.4 Export

The delivered product (including the technical data) is subject to export or import laws as well as the associated regulations of different counters, in particular those of Germany and the USA. The software may not be exported to countries where this is prohibited by the United States Export Administration Act and its additional provisions. You are obligated to comply with the regulations at your personal responsibility. We wish to inform you that you may require permission from state authorities to export, re-export or import the product.

2 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.

3 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
0x101c0004	IO_CFG	IO Configuration Register
0x101c0008	IO_CFG_MSK	IO Config Mask Register
0x101c000c	RESET_CTRL	Reset Control Register
0x101c0010	PHY_CTRL	PHY Control Register
0x101c0014	ARM_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of System Clock
0x101c0018	USB12_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of 12MHz USB clock
0x101c001c	ADC_CLK_DIV	Divisor of clock divider for 16MHz ADC clock
0x101c0020	FB0CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x101c0024	FB0CLK_DIV	Rate Multiplier Predivider
0x101c0028	CLK_EN	Global Clock Enable Register
0x101c002c	CLK_EN_MSK	Global Clock Enable Mask Register
0x101c0034	ONLY_PORN	Firmware Status register
0x101c0038	NETX_REV	netX Revision Register (written once during bootup)
0x101c0040	SAMPLE_AT_NRES	IO Sampled at Reset Status Register
0x101c0044	NETX_STATUS	netX System Status Configuration Register
0x101c0048	RDY_RUN_CFG	netX RDY/RUN IO System Status Configuration Register
0x101c004c	SYSTEM_STATUS	netX System Status Register
0x101c0050	NETX_LIC_ID	netX License ID Register
0x101c0054	NETX_LIC_FLAGS0	netX License Flags0 Register
0x101c0058	NETX_LIC_FLAGS1	netX License Flags1 Register
0x101c005c	NETX_LIC_ERRORS0	netX License Errors0 Status Register
0x101c0060	NETX_LIC_ERRORS1	netX License Errors1 Status Register
0x101c0204	WDG_CNTR	Watchdog Counter
0x101c0208	WDG_IRQ_TIMEOUT	Watchdog Interrupt Timeout
0x101c020c	WDG_RESET_TIMEOUT	Watchdog Reset Timeout
0x101c0a00	MMIO0_CFG	Multiplex matrix Configuration Register for MMIO0
0x101c0a04	MMIO1_CFG	Multiplex matrix Configuration Register for MMIO1
0x101c0a08	MMIO2_CFG	Multiplex matrix Configuration Register for MMIO2
0x101c0a0c	MMIO3_CFG	Multiplex matrix Configuration Register for MMIO3
0x101c0a10	MMIO4_CFG	Multiplex matrix Configuration Register for MMIO4
0x101c0a14	MMIO5_CFG	Multiplex matrix Configuration Register for MMIO5
0x101c0a18	MMIO6_CFG	Multiplex matrix Configuration Register for MMIO6
0x101c0a1c	MMIO7_CFG	Multiplex matrix Configuration Register for MMIO7
0x101c0a20	MMIO8_CFG	Multiplex matrix Configuration Register for MMIO8
0x101c0a24	MMIO9_CFG	Multiplex matrix Configuration Register for MMIO9
0x101c0a28	MMIO10_CFG	Multiplex matrix Configuration Register for MMIO10
0x101c0a2c	MMIO11_CFG	Multiplex matrix Configuration Register for MMIO11
0x101c0a30	MMIO12_CFG	Multiplex matrix Configuration Register for MMIO12
0x101c0a34	MMIO13_CFG	Multiplex matrix Configuration Register for MMIO13
0x101c0a38	MMIO14_CFG	Multiplex matrix Configuration Register for MMIO14
0x101c0a3c	MMIO15_CFG	Multiplex matrix Configuration Register for MMIO15
0x101c0a40	MMIO16_CFG	Multiplex matrix Configuration Register for MMIO16
0x101c0a44	MMIO17_CFG	Multiplex matrix Configuration Register for MMIO17

ARM Address	Register Name	Short Description
0x101c0a48	MMIO18_CFG	Multiplex matrix Configuration Register for MMIO18
0x101c0a4c	MMIO19_CFG	Multiplex matrix Configuration Register for MMIO19
0x101c0a50	MMIO20_CFG	Multiplex matrix Configuration Register for MMIO20
0x101c0a54	MMIO21_CFG	Multiplex matrix Configuration Register for MMIO21
0x101c0a58	MMIO22_CFG	Multiplex matrix Configuration Register for MMIO22
0x101c0a5c	MMIO23_CFG	Multiplex matrix Configuration Register for MMIO23
0x101c0a60	MMIO_PIO_OUT_LINE_CFG	MMIO PIO Line Output Level Register
0x101c0a64	MMIO_PIO_OE_LINE_CFG	MMIO PIO Line Output Enable Register
0x101c0a68	MMIO_IN_LINE_STATUS	MMIO Input Line Register
0x101c0a6c	MMIO_IS_PIO_STATUS	MMIO Mode Line Register
0x101c0c40	HIF_IO_CFG	HIF IO Config Register
0x101c0c44	HIF_PIO_OUT0	HIF PIO Output State Configuration Register 0
0x101c0c48	HIF_PIO_OUT1	HIF PIO Output State Configuration Register 1
0x101c0c4c	HIF_PIO_OE0	HIF PIO Output Enable Configuration Register 0
0x101c0c50	HIF_PIO_OE1	HIF PIO Output Enable Configuration Register 1
0x101c0c54	HIF_PIO_IN0	HIF PIO Input State Register 0
0x101c0c58	HIF_PIO_IN1	HIF PIO Input State Register 1
0x101c12d8	SYS_STAT	System Status

3.1 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:

Note:
There are three completely independend protections: one for ARM, one for xPIC and one for rest of netX system masters. That allows running proteced access from ARM while xPIC is performing a protected access in the meanwhile.

ARM Address	Register Name	Short Description
0x101c0004	IO_CFG	IO Config Register
0x101c0008	IO_CFG_MSK	IO Config Mask Register
0x101c000c	RESET_CTRL	Reset Control Register
0x101c0010	PHY_CTRL	PHY Control Register
0x101c0014	ARM_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of System Clock
0x101c0018	USB12_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of 12MHz USB clock
0x101c001c	ADC_CLK_DIV	Divisor of clock divider for 16MHz ADC clock
0x101c0020	FB0CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x101c0024	FB0CLK_DIV	Rate Multiplier Predivider
0x101c0028	CLK_EN	Global Clock Enable Register
0x101c002c	CLK_EN_MSK	Global Clock Enable Mask Register
0x101c0034	ONLY_PORN	Firmware Status Register
0x101c0038	NETX_REV	netX Revision Register (written once during bootup)
0x101c0a00	MMIO0_CFG	Multiplex matrix Configuration Register for MMIO0
…	…	…
0x101c0a5c	MMIO23_CFG	Multiplex matrix Configuration Register for MMIO23
0x101c0c40	HIF_IO_CFG	HIF IO Config Register

For writing to one of these registers, the software must perform 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 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.

ACCESS_KEY – ASIC Control Locking Access Key Register 0x101c0070

31–16	15–0
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

3.2 NETX_REV – netX Revision

NETX_REV_BL – netX Boot Loader Version Register 0x101c0038

This register contains information about the netX boot loader version.

31 … 8	7 … 0
reserved	BL_VERSION

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

NETX_REV_HW – netX Hardware Revision Register 0x00005003

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

31 … 28	27 … 20	19 … 12	11 … 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 0x05 = netX10 …..	R	0x05
11:0	ROM_REV	ROM Code Revision	R	0x003

3.3 IO Configuration

To keep the chip package small, the netX10 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 24 MMIO pads. The Multiplexing Matrix unit inside netX10 allows selecting one of 58 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	Option1		Option2		Option3		Option4	
	Signal	Select signal	Signal	Select signal	Signal	Select signal	Signal	Select signal	Signal
MMIO0	MMIO0			SEL_FO0	FO0_RD				
MMIO1	MMIO1			SEL_FO0	FO0_TD	SEL_XM0_TX	XM0_TX		
MMIO2	MMIO2			SEL_FO0	FO0_FN_EN	SEL_XM0_ECLK	XM0_ECLK	SEL_FB0CLK	FB0CLK
MMIO3	MMIO3	SEL_MII0	XM0_MII_RXD0	SEL_FO0	FO0_SD	SEL_XM0_TXOE	XM0_TXOE		
MMIO4	MMIO4	SEL_MII0	XM0_MII_RXD1	SEL_PWM0	PWM0				
MMIO5	MMIO5	SEL_MII0	XM0_MII_RXD2	SEL_PWM1	PWM1				
MMIO6	MMIO6	SEL_MII0	XM0_MII_RXD3	SEL_PWM2	PWM2				
MMIO7	MMIO7	SEL_MII0	XM0_MII_RXDV	SEL_PWM3	PWM3				
MMIO8	MMIO8	SEL_MII2	XM0_MII_RXER	SEL_PWM4	PWM4				
MMIO9	MMIO9	SEL_MII1	XM0_MII_TXCLK	SEL_PWM5	PWM5				
MMIO10	MMIO10	SEL_MII1	XM0_MII_TXD0	SEL_PWM6	PWM6				
MMIO11	MMIO11	SEL_MII1	XM0_MII_TXD1	SEL_PWM7	PWM7				
MMIO12	MMIO12	SEL_MII1	XM0_MII_TXD2						
MMIO13	MMIO13	SEL_MII1	XM0_MII_TXD3						
MMIO14	MMIO14	SEL_MII3	XM0_MII_TXEN						
MMIO15	MMIO15	SEL_MII4	XM0_MII_TXER						
MMIO16	MMIO16	SEL_MII5	XM0_MII_COL						
MMIO17	MMIO17	SEL_MII5	XM0_MII_CRS						
MMIO18	MMIO18	SEL_MII6	XM0_MII_IRQ						
MMIO19	MMIO19	SEL_MII7	MII_MDC						
MMIO20	MMIO20	SEL_MII7	MII_MDIO						

The select signals are set in register IO_CFG. Their priority in the table above rises from left to right, i.e. “Option 4” has the highest priority, and “Option 1” has the lowest priority, but has higher priority than the standard function. If none of the options is activated, the standard function applies automatically.

By default (after reset or power on), all MMIOs are configured for PIO input mode, which is a new feature (since netX10) and allows to directly control (configured as output) or read (configures as input) the signal state of an MMIO signal through appropriate registers.

IO_CFG – IO Config Register **0x101c0004**

The following register is used for selects of output pin multiplexing. By setting the appropriate bits of IO_CFG register, the desired functions can be activated.

Selects can only be activated, if appropriate bit of IO_CFG_MSK is set. Bits will be reset according to the IO_CFG_MSK register if a new mask is correctly written.

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:

IO_CFG 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

Note:
HIF IO configuration must be done in HIF_IO_CFG register (area HIF_IO_CTRL).

31:24	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	SEL_ETM	USB2JTAG_EN	SEL_PWM7	SEL_PWM6	SEL_PWM5	SEL_PWM4	SEL_PWM3	SEL_PWM2	SEL_PWM1	SEL_PWM0	SEL_MII7	SEL_MII6	SEL_MII5	SEL_MII4	SEL_MII3	SEL_MII2	SEL_MII1	SEL_MII0	SEL_FO0	SEL_RXCLK_FROM_INTPHY	SEL_FB0CLK	SEL_XM0_ECLK	SEL_XM0_TXOE	SEL_XM0_TX

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	SEL_ETM	select pins for ETM9	R/W	0x0
22	USB2JTAG_EN	Enable USB JTAG debug feature	R/W	0x0
21	SEL_PWM7	select pad for PWM output	R/W	0x0
20	SEL_PWM6	select pad for PWM output	R/W	0x0
19	SEL_PWM5	select pad for PWM output	R/W	0x0
18	SEL_PWM4	select pad for PWM output	R/W	0x0
17	SEL_PWM3	select pad for PWM output	R/W	0x0
16	SEL_PWM2	select pad for PWM output	R/W	0x0
15	SEL_PWM1	select pad for PWM output	R/W	0x0
14	SEL_PWM0	select pad for PWM output	R/W	0x0
13	SEL_MII7	select pad for MDIO interfase	R/W	0x0
12	SEL_MII6	select pad for xMAC0 external MII interrupt	R/W	0x0
11	SEL_MII5	select pads for xMAC0 external MII col and crs	R/W	0x0
10	SEL_MII4	select pad for xMAC0 external MII tx-error	R/W	0x0
9	SEL_MII3	select pad for xMAC0 external MII tx-enable	R/W	0x0
8	SEL_MII2	select pad for xMAC0 external MII rx-error	R/W	0x0
7	SEL_MII1	select pads for xMAC0 external MII transmit data	R/W	0x0
6	SEL_MII0	select pads for xMAC0 external MII receive data	R/W	0x0
5	SEL_FO0	select Fiber Optics of PHY0: 1: use Fiber Optics of PHY0 0: use standard interface of PHY0	R/W	0x0
4	SEL_RXCLK_FROM_INTPHY	select rxclk/eclk input to xMAC 0: xMAC gets rxclk/eclk from multiplexmatrix 1: xMAC gets rxclk from internal PHY This extra select for rxclk-input allows output of fbclk while using internal PHY. Note:together with SEL_FB0CLK and FB0 bit in CLK_EN register, the xMAC clock and fb0cllk is configed as followings: {SEL_RXCLK_FROM_INTPHY, SEL_FB0CLK, CLK_EN} : 000: xm0_eclk_in(MMIO2 input) is used as xMAC clock 001: fb0clk is used as xMAC clock, MMIO2 unused 011: fb0clk is used as xMAC clock and output at MMIO2 100: internal PHYclock is used as xMAC clock, fb0clk disabled 11x: internal PHYclock is used as xMAC clock, fb0clk is output	R/W	0x0

Bits	Name	Description	R/W	Default
		at MMIO2		
3	SEL_FB0CLK	select pad for fieldbus-clk0	R/W	0x0
2	SEL_XM0_ECLK	select pad for xMAC0 eclk	R/W	0x0
1	SEL_XM0_TXOE	select pad for xMAC0 tx-bitstream direct output enable	R/W	0x0
0	SEL_XM0_TX	select pad for xMAC0 tx-bitstream direct output	R/W	0x0

IO_CFG_MSK – IO Config Mask Register 0x101c0008

The IO_CFG_MSK register might be used to lock special IO configurations for restricted netX devices. Any bit of the IO_CFG register can only be set, if the corresponding mask bit in the IO_CFG_MSK register is set either.

This register is lockable by netX locking algorithm. It will be only reset on Power on, not on normal system nres. The IO_CFG register will change according to this register if a new mask is correctly written (netX locking algorithm).

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

Note:
HIF IO configuration must be done in HIF_IO_CFG register (area HIF_IO_CTRL).

31	30	29	28	27	26	25	24	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								SEL_ETM	USB2JTAG_EN	SEL_PWM7	SEL_PWM6	SEL_PWM5	SEL_PWM4	SEL_PWM3	SEL_PWM2	SEL_PWM1	SEL_PWM0	SEL_MII7	SEL_MII6	SEL_MII5	SEL_MII4	SEL_MII3	SEL_MII2	SEL_MII1	SEL_MII0	SEL_FO0	SEL_RXCLK_FROM_INTPHY	SEL_FB0CLK	SEL_XM0_ECLK	SEL_XM0_TXOE	SEL_XM0_TX

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	SEL_ETM	select pins for ETM9	R/W	0x1
22	USB2JTAG_EN	Enable USB JTAG debug feature	R/W	0x1
21	SEL_PWM7	select pad for PWM output	R/W	0x1
20	SEL_PWM6	select pad for PWM output	R/W	0x1
19	SEL_PWM5	select pad for PWM output	R/W	0x1
18	SEL_PWM4	select pad for PWM output	R/W	0x1
17	SEL_PWM3	select pad for PWM output	R/W	0x1
16	SEL_PWM2	select pad for PWM output	R/W	0x1
15	SEL_PWM1	select pad for PWM output	R/W	0x1
14	SEL_PWM0	select pad for PWM output	R/W	0x1
13	SEL_MII7	select pad for MDIO interfase	R/W	0x1
12	SEL_MII6	select pad for xMAC0 external MII interrupt	R/W	0x1
11	SEL_MII5	select pads for xMAC0 external MII col and crs	R/W	0x1

Bits	Name	Description	R/W	Default
10	SEL_MII4	select pad for xMAC0 external MII tx-error	R/W	0x1
9	SEL_MII3	select pad for xMAC0 external MII tx-enable	R/W	0x1
8	SEL_MII2	select pad for xMAC0 external MII rx-error	R/W	0x1
7	SEL_MII1	select pads for xMAC0 external MII transmit data	R/W	0x1
6	SEL_MII0	select pads for xMAC0 external MII receive data	R/W	0x1
5	SEL_FO0	select Fiber Optics of PHY0:	R/W	0x1
4	SEL_RXCLK_FROM_INTPHY	select rxclk/eclk input to xMAC	R/W	0x1
3	SEL_FB0CLK	select pad for fieldbus-clk0	R/W	0x1
2	SEL_XM0_ECLK	select pad for xMAC0 eclk	R/W	0x1
1	SEL_XM0_TXOE	select pad for xMAC0 tx-bitstream direct output enable	R/W	0x1
0	SEL_XM0_TX	select pad for xMAC0 tx-bitstream direct output	R/W	0x1

MMIO0_CFG – IO-Multiplex matrix Configuration Register for MMIO0 0x101c0a00

MMIO1_CFG – IO-Multiplex matrix Configuration Register for MMIO1 0x101c0a04

…

MMIO23_CFG – IO-Multiplex matrix Configuration Register for MMIO23 0x101c0a5c

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 … 10	9	8	7 … 6	5 … 0
reserved	MMIO_IN_INV	MMIO_OUT_INV	reserved	MMIO_SEL

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9	MMIO[0-23]_IN_INV	Invert input: 1: invert input signal 0: keep original signal polarity	R/W	0x0
8	MMIO[0-23]_OUT_INV	Invert output: 1: invert output signal 0: keep original signal polarity	R/W	0x0
7:6	reserved	-	R	0x0
5:0	MMIO[0-23]_SEL	MMIO[0-23]_ 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	GPIO0	bi-directional	GPIO	MMIO0
0x08	GPIO1	bi-directional	GPIO	MMIO1
0x09	GPIO2	bi-directional	GPIO	MMIO2
0x0a	GPIO3	bi-directional	GPIO	MMIO3
0x0b	GPIO4	bi-directional	GPIO	MMIO4
0x0c	GPIO5	bi-directional	GPIO	MMIO5
0x0d	GPIO6	bi-directional	GPIO	MMIO6
0x0e	GPIO7	bi-directional	GPIO	MMIO7
0x0f	PHY0_LED0	push/pull output	internal PHY0 Status	
0x10	PHY0_LED1	push/pull output	internal PHY0 Status	
0x11	PHY0_LED2	push/pull output	internal PHY0 Status	
0x12	PHY0_LED3	push/pull output	internal PHY0 Status	
0x13	SPI0_CS1n	bi-directional	SPI0 2nd chip select	
0x14	SPI0_CS2n	bi-directional	SPI0 3rd chip select	
0x15	SPI1_CLK	bi-directional	SPI1	
0x16	SPI1_CS0n	bi-directional	SPI1	
0x17	SPI1_CS1n	bi-directional	SPI1	
0x18	SPI1_CS2n	bi-directional	SPI1	
0x19	SPI1_MISO	bi-directional	SPI1	
0x1a	SPI1_MOSI	bi-directional	SPI1	
0x1b	I2C_SCL_MMIO	bi-directional	I2C	
0x1c	I2C_SDA_MMIO	bi-directional	I2C	
0x1d	UART0_CTS	input	UART 0	
0x1e	UART0_RTS	tristatable output	UART 0	
0x1f	UART0_RXD	input	UART 0	
0x20	UART0_TXD	tristatable output	UART 0	
0x21	UART1_CTS	input	UART 1	
0x22	UART1_RTS	tristatable output	UART 1	
0x23	UART1_RXD	input	UART 1	
0x24	UART1_TXD	tristatable output	UART 1	
0x25	PWM_FAILURE_n	input	PWM (should alternatively be controlled by SW)	
0x26	POS_ENC0_A	input	Encoder	
0x27	POS_ENC0_B	input	Encoder	
0x28	POS_ENC0_N	input	Encoder	
0x29	POS_ENC1_A	input	Encoder	
0x2a	POS_ENC1_B	input	Encoder	
0x2b	POS_ENC1_N	input	Encoder	
0x2c	POS_MP0	input	Encoder	
0x2d	POS_MP1	input	Encoder	
0x2e	IO_LINK0_IN	Input	IO-Link	
0x2f	IO_LINK0_OUT	push/pull output	IO-Link	
0x30	IO_LINK0_OE	push/pull output	IO-Link	
0x31	IO_LINK1_IN	Input	IO-Link	
0x32	IO_LINK1_OUT	push/pull output	IO-Link	
0x33	IO_LINK1_OE	push/pull output	IO-Link	
0x34	IO_LINK2_IN	Input	IO-Link	
0x35	IO_LINK2_OUT	push/pull output	IO-Link	
0x36	IO_LINK2_OE	push/pull output	IO-Link	
0x37	IO_LINK3_IN	input	IO-Link	
0x38	IO_LINK3_OUT	push/pull output	IO-Link	
0x39	IO_LINK3_OE	push/pull output	IO-Link	

0x3f	PIO mode	use MMIO PIO line registers	PIO function	MMIO[8-23]

MMIO_PIO_OUT_LINE_CFG – MMIO PIO Line Output Level Register 0x101c0a60

Note: This register is not locked by netX locking algorithm.

31 30 29 28 27 26 25 24	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	LINE

Bits	Name	Description	R/W	Default
31:24	-	reserved	R	0x0
23:0	LINE	MMIO output state if related MMIO is in PIO mode. If related MMIO is not in PIO mode, programmed setting is ignored. Bit 0 controls MMIO0, Bit 1 controls MMIO1, ...	R/W	0x0

MMIO_PIO_OE_LINE_CFG – MMIO PIO Line Output Enable Register 0x101c0a64

Note: This register is not locked by netX locking algorithm.

31 30 29 28 27 26 25 24	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	LINE

Bits	Name	Description	R/W	Default
31:24	-	reserved	R	0x0
23:0	LINE	MMIO output enable if related MMIO is in PIO mode. If related MMIO is not in PIO mode, programmed setting is ignored. Bit 0 controls MMIO0, Bit 1 controls MMIO1, ...	R/W	0x0

MMIO_IN_LINE_STATUS – MMIO Input Line Register 0x101c0a68

31 30 29 28 27 26 25 24	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	LINE

Bits	Name	Description	R/W	Default
31:24	-	reserved	R	0x0
23:0	LINE	sampled MMIO input state. Does not depend whether MMIO is in PIO mode or not. Bit 0 monitors MMIO0, Bit 1 monitors MMIO1, ...	R	0x0

MMIO_IS_PIO_STATUS – MMIO Mode Line Register 0x101c0a6c

Note: PIO Mode can be enabled or disabled in MMIO*_CFG registers.

31	30	29	28	27	26	25	24	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								LINE																							

Bits	Name	Description	R/W	Default
31:24	-	reserved	R	0x0
23:0	LINE	Bit 0 shows status of MMIO0, Bit 1 shows status of MMIO1, ... 0: related MMIO is not in PIO mode (is assigned to core functionality). 1: related MMIO is in PIO mode (is not assigned to core functionality).	R	0x0

3.4 HIF IO Configuration

HIF_IO_CFG – HIF IO Config Register **0x101c0c40**

Selects of HIF pin multiplexing. See pin table in the netX10 Technical Reference Guide for details. This configuration must be set up according to external netX connection before any access to external logic.

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

Attention: Be very careful programming this register. False settings may cause permanent damage on netX or devices connected to HIF-IOs.

31–25	24	23–21	20–8	7	6	5–4	3	2	1	0
reserved	EN_HIF_RDY_PIO_MI	reserved	EN_HIF_A23TO11_PIO_MI	reserved	EN_HIF_SDRAM_MI	HIF_MI_CFG	SEL_DPM_SERIAL_SPO	SEL_DPM_SERIAL_SPH	SEL_DPM_SERIAL	SEL_HIF_DPM

Bits	Name	Description	R/W	Default
31:25	reserved	-	R	0x0
24	EN_HIF_RDY_PIO_MI	Enable HIF_RDY for PIO usage. Note: This bit must be disabled if HIF_RDY is used as EXT_BUS RDY. Note: This bit is ignored if HIF is DPM. Use DPM RDY configuration if HIF_RDY should be used as PIO together with DPM functionality.	R/W	0x1
23:21	reserved	-	R	0x0
20:8	EN_HIF_A23TO11_PIO_MI	Enable HIF_A23..11 for PIO usage in MI function. Note: If 'hif_mi_cfg' bit is set, HIF_A18..14 are not available as PIOs even if according bits are set here. Note: For parallel DPM address line PIO usage depends on programmed DPM address range (config in area DPM). Note: For serial DPM all address lines can be used as PIOs if MI is not enabled ('hif_mi_cfg' is 0).	R/W	0x1fff
7	reserved	-	R	0x0
6	EN_HIF_SDRAM_MI	Enable HIF IOs for SDRAM Memory Interface configuration. If enabled following IOs are used as outputs for SDRAM (netx10, partial shared with SRAM/FLASH ctrl signals): netX10 IO / Function / Comment HIF_A14 / SD_BA0 / Only during SDRAM access, usable as FLASH/SRAM A14 simultaneously. HIF_A15 / SD_BA1 / Only during SDRAM access, usable as FLASH/SRAM A15 simultaneously. HIF_A16 / SD_RASN / Only during SDRAM access, usable as FLASH/SRAM A16 simultaneously. HIF_A17 / SD_CASN / Only during SDRAM access, usable as FLASH/SRAM A17 simultaneously. HIF_A18 / SD_DQM0N / Only during SDRAM access, usable as FLASH/SRAM A18	R/W	0x0

Bits	Name	Description	R/W	Default
		simultaneously. HIF_RDN SD_CKE Only during SDRAM access, usable as FLASH/SRAM nRD simultaneously. HIF_WRN SD_WEN Only during SDRAM access, usable as FLASH/SRAM nWR simultaneously. HIF_DIRQ SD_CSN For serial DPM and 8 bit MI DPM IRQ/DIRQ is available on HIF_D1. HIF_SDCLK SD_CLK For serial DPM and 8 bit MI DPM FIQ/SIRQ is available on HIF_D13. Note: For SDRAM usage, 'hif_mi_cfg' must be configured for 8 or 16 bit MI. Note: If this bit is set, HIF_A18..14 are not available as PIOs even if according bits are set in 'en_a23to11_pio'. Note: For parallel DPM fast/service IRQ functionality (SIRQ/FIQ) on HIF_SDCLK this bit must be set to '0'.		
5:4	HIF_MI_CFG	HIF IO Memory Interface usage configuration. Note: Configuration of SRAM/FLASH Chip-Select usage must be done additionally in ASYNCMEM_CTRL address area. By default, all Chip-Selects are configured for PIO usage. If any external memory is used, Chip-Select configuration must be done before the first access to external memory. Otherwise netX or memory devices may be damaged. Settings: 00: HIF IOs are used as 8 bit MI (together with serial DPM possible). Used HIF IOs: HIF_D7..0, HIF_A10..0, HIF_RDN, HIF_WRN. 01: HIF IOs are used as 16 bit MI (no DPM possible). Used HIF IOs (add. to '00' setting): HIF_D15..8, HIF_BHEN. 10: reserved. 11: No MI usage. HIF IOs can be used as PIOs or for parallel DPM. Note: If upper address lines HIF_A23...11 are not used as PIOs, this must be configured in bits 'en_a23to11'. Note: If HIF is configured as parallel DPM ('sel_hif_dpm' set and 'sel_dpm_serial' not set), HIF IOs are not available for Memory Interface usage programmed value is ignored. Note: If HIF is configured as serial DPM ('sel_hif_dpm' set and 'sel_dpm_serial' set), HIF IOs are not available for 16 bit Memory Interface. Programmed value '01' will be ignored in this case. Note: SDRAM Chip-Select is multiplext with SRAM/FLASH Chip-Select 1 on HIF_DIRQ. If 'en_hif_sdram_mi' is set and SRAM/FLASH Chip-Select 1 enabled in ASYNCMEM_CTRL address area, SDRAM Chip-Select gains priority and SRAM/FLASH Chip-Select 1 will not be mapped to HIF_DIRQ.	R/W	0x3
3	SEL_DPM_SERIAL_SPO	select serial DPM mode SPI clock polarity (sel_hif_dpm and sel_dpm_serial must be set). 0: Serial clock idle state is low. 1: Serial clock idle state is high.	R/W	0x0
2	SEL_DPM_SERIAL_SPH	select serial DPM mode SPI clock phase (sel_hif_dpm and sel_dpm_serial must be set). 0: Serial data sampling on first serial clock edge. 1: Serial data sampling on second serial clock edge.	R/W	0x0
1	SEL_DPM_SERIAL	select serial DPM mode (ignored if sel_hif_dpm not set). Note: Serial DPM is an SPI compliant interface. Note: Serial DPM can be used together with 8 bit Memory Interface on HIF IOs..	R/W	0x0
0	SEL_HIF_DPM	select HIF pins for DPM Note: For parallel DPM IO configuration use config registers in address area DPM. Note: Parallel DPM fast/service IRQ functionallity (SIRQ/FIQ) on HIF_SDCLK is controlled by en_hif_sdram_mi bit.	R/W	0x0

HIF_PIO_OUT0 – HIF PIO Output State Configuration Register 0 0x101c0c44

All unused HIF signals can be used as PIOs. IOs will be driven to the programmed state if appropriate enable bit is set in HIF_PIO_OE0 register.

PIO mode driving of HIF-IOs used in current HIF/EXT_BUS Memory Interface configuration is not possible.

31:16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	HIF_D15	HIF_D14	HIF_D13	HIF_D12	HIF_D11	HIF_D10	HIF_D9	HIF_D8	HIF_A23	HIF_A22	HIF_A21	HIF_A20	HIF_A19	HIF_A18	HIF_A17	HIF_A16

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15	HIF_D15	PIO output state of HIF_D15 signal.	R/W	0x0
14	HIF_D14	PIO output state of HIF_D14 signal.	R/W	0x0
13	HIF_D13	PIO output state of HIF_D13 signal.	R/W	0x0
12	HIF_D12	PIO output state of HIF_D12 signal.	R/W	0x0
11	HIF_D11	PIO output state of HIF_D11 signal.	R/W	0x0
10	HIF_D10	PIO output state of HIF_D10 signal.	R/W	0x0
9	HIF_D9	PIO output state of HIF_D9 signal.	R/W	0x0
8	HIF_D8	PIO output state of HIF_D8 signal.	R/W	0x0
7	HIF_A23	PIO output state of HIF_A23 signal.	R/W	0x0
6	HIF_A22	PIO output state of HIF_A22 signal.	R/W	0x0
5	HIF_A21	PIO output state of HIF_A21 signal.	R/W	0x0
4	HIF_A20	PIO output state of HIF_A20 signal.	R/W	0x0
3	HIF_A19	PIO output state of HIF_A19 signal.	R/W	0x0
2	HIF_A18	PIO output state of HIF_A18 signal.	R/W	0x0
1	HIF_A17	PIO output state of HIF_A17 signal.	R/W	0x0
0	HIF_A16	PIO output state of HIF_A16 signal.	R/W	0x0

HIF_PIO_OUT1 – HIF PIO Output State Configuration Register 1 0x101c0c48

All unused HIF signals can be used as PIOs. IOs will be driven to the programmed state if appropriate enable bit is set in HIF_PIO_OE1 register.

PIO mode driving of HIF-IOs used in current HIF/EXT_BUS Memory Interface configuration is not possible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIF_SDCLK	HIF_DIRQ	HIF_RDY	HIF_CSN	HIF_WRN	HIF_RDN	HIF_BHEN	reserved	HIF_A15	HIF_A14	HIF_A13	HIF_A12	HIF_A11	HIF_A10	HIF_A9	HIF_A8	HIF_A7	HIF_A6	HIF_A5	HIF_A4	HIF_A3	HIF_A2	HIF_A1	HIF_A0	HIF_D7	HIF_D6	HIF_D5	HIF_D4	HIF_D3	HIF_D2	HIF_D1	HIF_D0

Bits	Name	Description	R/W	Default
31	HIF_SDCLK	PIO output state of HIF_SDCLK signal.	R/W	0x0
30	HIF_DIRQ	PIO output state of HIF_DIRQ signal.	R/W	0x0
29	HIF_RDY	PIO output state of HIF_RDY signal.	R/W	0x0
28	HIF_CSN	PIO output state of HIF_CSN signal.	R/W	0x0
27	HIF_WRN	PIO output state of HIF_WRN signal.	R/W	0x0
26	HIF_RDN	PIO output state of HIF_RDN signal.	R/W	0x0
25	HIF_BHEN	PIO output state of HIF_BHEN signals.	R/W	0x0
24	reserved	-	R	0x0
23	HIF_A15	PIO output state of HIF_A15 signal.	R/W	0x0
22	HIF_A14	PIO output state of HIF_A14 signal.	R/W	0x0
21	HIF_A13	PIO output state of HIF_A13 signal.	R/W	0x0
20	HIF_A12	PIO output state of HIF_A12 signal.	R/W	0x0
19	HIF_A11	PIO output state of HIF_A11 signal.	R/W	0x0
18	HIF_A10	PIO output state of HIF_A10 signal.	R/W	0x0
17	HIF_A9	PIO output state of HIF_A9 signal.	R/W	0x0
16	HIF_A8	PIO output state of HIF_A8 signal.	R/W	0x0
15	HIF_A7	PIO output state of HIF_A7 signal.	R/W	0x0
14	HIF_A6	PIO output state of HIF_A6 signal.	R/W	0x0
13	HIF_A5	PIO output state of HIF_A5 signal.	R/W	0x0
12	HIF_A4	PIO output state of HIF_A4 signal.	R/W	0x0
11	HIF_A3	PIO output state of HIF_A3 signal.	R/W	0x0
10	HIF_A2	PIO output state of HIF_A2 signal.	R/W	0x0
9	HIF_A1	PIO output state of HIF_A1 signal.	R/W	0x0
8	HIF_A0	PIO output state of HIF_A0 signal.	R/W	0x0
7	HIF_D7	PIO output state of HIF_D7 signal.	R/W	0x0
6	HIF_D6	PIO output state of HIF_D6 signal.	R/W	0x0
5	HIF_D5	PIO output state of HIF_D5 signal.	R/W	0x0
4	HIF_D4	PIO output state of HIF_D4 signal.	R/W	0x0
3	HIF_D3	PIO output state of HIF_D3 signal.	R/W	0x0
2	HIF_D2	PIO output state of HIF_D2 signal.	R/W	0x0
1	HIF_D1	PIO output state of HIF_D1 signal.	R/W	0x0
0	HIF_D0	PIO output state of HIF_D0 signal.	R/W	0x0

HIF_PIO_OE0 – HIF PIO Output Enable Configuration Register 0 0x101c0c4c

All unused HIF signals can be used as PIOs. IOs will be driven to the output state programmed in in HIF_PIO_OUT0 register.

PIO mode driving of HIF-IOs used in current HIF/EXT_BUS Memory Interface configuration is not possible.

31	30	29	28	27	26	25	24	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																HIF_D15	HIF_D14	HIF_D13	HIF_D12	HIF_D11	HIF_D10	HIF_D9	HIF_D8	HIF_A23	HIF_A22	HIF_A21	HIF_A20	HIF_A19	HIF_A18	HIF_A17	HIF_A16

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15	HIF_D15	PIO output enable of HIF_D15 signal.	R/W	0x0
14	HIF_D14	PIO output enable of HIF_D14 signal.	R/W	0x0
13	HIF_D13	PIO output enable of HIF_D13 signal.	R/W	0x0
12	HIF_D12	PIO output enable of HIF_D12 signal.	R/W	0x0
11	HIF_D11	PIO output enable of HIF_D11 signal.	R/W	0x0
10	HIF_D10	PIO output enable of HIF_D10 signal.	R/W	0x0
9	HIF_D9	PIO output enable of HIF_D9 signal.	R/W	0x0
8	HIF_D8	PIO output enable of HIF_D8 signal.	R/W	0x0
7	HIF_A23	PIO output enable of HIF_A23 signal.	R/W	0x0
6	HIF_A22	PIO output enable of HIF_A22 signal.	R/W	0x0
5	HIF_A21	PIO output enable of HIF_A21 signal.	R/W	0x0
4	HIF_A20	PIO output enable of HIF_A20 signal.	R/W	0x0
3	HIF_A19	PIO output enable of HIF_A19 signal.	R/W	0x0
2	HIF_A18	PIO output enable of HIF_A18 signal.	R/W	0x0
1	HIF_A17	PIO output enable of HIF_A17 signal.	R/W	0x0
0	HIF_A16	PIO output enable of HIF_A16 signal.	R/W	0x0

HIF_PIO_OE1 – HIF PIO Output Enable Configuration Register 1 0x101c0c50

All unused HIF signals can be used as PIOs. IOs will be driven to the output state programmed in in HIF_PIO_OUT1 register.

PIO mode driving of HIF-IOs used in current HIF/EXT_BUS Memory Interface configuration is not possible.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIF_SDCLK	HIF_DIRQ	HIF_RDY	HIF_CSN	HIF_WRN	HIF_RDN	HIF_BHEN	reserved	HIF_A15	HIF_A14	HIF_A13	HIF_A12	HIF_A11	HIF_A10	HIF_A9	HIF_A8	HIF_A7	HIF_A6	HIF_A5	HIF_A4	HIF_A3	HIF_A2	HIF_A1	HIF_A0	HIF_D7	HIF_D6	HIF_D5	HIF_D4	HIF_D3	HIF_D2	HIF_D1	HIF_D0

Bits	Name	Description	R/W	Default
31	HIF_SDCLK	PIO output enable of HIF_SDCLK signal.	R/W	0x0
30	HIF_DIRQ	PIO output enable of HIF_DIRQ signal.	R/W	0x0
29	HIF_RDY	PIO output enable of HIF_RDY signal.	R/W	0x0
28	HIF_CSN	PIO output enable of HIF_CSN signal.	R/W	0x0
27	HIF_WRN	PIO output enable of HIF_WRN signal.	R/W	0x0
26	HIF_RDN	PIO output enable of HIF_RDN signal.	R/W	0x0
25	HIF_BHEN	PIO output enable of HIF_BHEN signals.	R/W	0x0
24	reserved	-	R	0x0
23	HIF_A15	PIO output enable of HIF_A15 signal.	R/W	0x0
22	HIF_A14	PIO output enable of HIF_A14 signal.	R/W	0x0
21	HIF_A13	PIO output enable of HIF_A13 signal.	R/W	0x0
20	HIF_A12	PIO output enable of HIF_A12 signal.	R/W	0x0
19	HIF_A11	PIO output enable of HIF_A11 signal.	R/W	0x0
18	HIF_A10	PIO output enable of HIF_A10 signal.	R/W	0x0
17	HIF_A9	PIO output enable of HIF_A9 signal.	R/W	0x0
16	HIF_A8	PIO output enable of HIF_A8 signal.	R/W	0x0
15	HIF_A7	PIO output enable of HIF_A7 signal.	R/W	0x0
14	HIF_A6	PIO output enable of HIF_A6 signal.	R/W	0x0
13	HIF_A5	PIO output enable of HIF_A5 signal.	R/W	0x0
12	HIF_A4	PIO output enable of HIF_A4 signal.	R/W	0x0
11	HIF_A3	PIO output enable of HIF_A3 signal.	R/W	0x0
10	HIF_A2	PIO output enable of HIF_A2 signal.	R/W	0x0
9	HIF_A1	PIO output enable of HIF_A1 signal.	R/W	0x0
8	HIF_A0	PIO output enable of HIF_A0 signal.	R/W	0x0
7	HIF_D7	PIO output enable of HIF_D7 signal.	R/W	0x0
6	HIF_D6	PIO output enable of HIF_D6 signal.	R/W	0x0
5	HIF_D5	PIO output enable of HIF_D5 signal.	R/W	0x0
4	HIF_D4	PIO output enable of HIF_D4 signal.	R/W	0x0
3	HIF_D3	PIO output enable of HIF_D3 signal.	R/W	0x0
2	HIF_D2	PIO output enable of HIF_D2 signal.	R/W	0x0
1	HIF_D1	PIO output enable of HIF_D1 signal.	R/W	0x0
0	HIF_D0	PIO output enable of HIF_D0 signal.	R/W	0x0

HIF_PIO_IN0 – HIF PIO Input State Register 0 0x101c0c54

IO input states can be read here regardless whether IO is used in current HIF/EXT_BUS Memory Interface configuration.

31	30	29	28	27	26	25	24	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																HIF_D15	HIF_D14	HIF_D13	HIF_D12	HIF_D11	HIF_D10	HIF_D9	HIF_D8	HIF_A23	HIF_A22	HIF_A21	HIF_A20	HIF_A19	HIF_A18	HIF_A17	HIF_A16

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15	HIF_D15	Input state of HIF_D15 signal.	R	0x0
14	HIF_D14	Input state of HIF_D14 signal.	R	0x0
13	HIF_D13	Input state of HIF_D13 signal.	R	0x0
12	HIF_D12	Input state of HIF_D12 signal.	R	0x0
11	HIF_D11	Input state of HIF_D11 signal.	R	0x0
10	HIF_D10	Input state of HIF_D10 signal.	R	0x0
9	HIF_D9	Input state of HIF_D9 signal.	R	0x0
8	HIF_D8	Input state of HIF_D8 signal.	R	0x0
7	HIF_A23	Input state of HIF_A23 signal.	R	0x0
6	HIF_A22	Input state of HIF_A22 signal.	R	0x0
5	HIF_A21	Input state of HIF_A21 signal.	R	0x0
4	HIF_A20	Input state of HIF_A20 signal.	R	0x0
3	HIF_A19	Input state of HIF_A19 signal.	R	0x0
2	HIF_A18	Input state of HIF_A18 signal.	R	0x0
1	HIF_A17	Input state of HIF_A17 signal.	R	0x0
0	HIF_A16	Input state of HIF_A16 signal.	R	0x0

HIF_PIO_IN1 – HIF PIO Input State Register 1 **0x101c0c58**

IO input states can be read here regardless whether IO is used in current HIF/EXT_BUS Memory Interface configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HIF_SDCLK	HIF_DIRQ	HIF_RDY	HIF_CSN	HIF_WRN	HIF_RDN	HIF_BHEN	reserved	HIF_A15	HIF_A14	HIF_A13	HIF_A12	HIF_A11	HIF_A10	HIF_A9	HIF_A8	HIF_A7	HIF_A6	HIF_A5	HIF_A4	HIF_A3	HIF_A2	HIF_A1	HIF_A0	HIF_D7	HIF_D6	HIF_D5	HIF_D4	HIF_D3	HIF_D2	HIF_D1	HIF_D0

Bits	Name	Description	R/W	Default
31	HIF_SDCLK	Input state of HIF_SDCLK signal.	R	0x0
30	HIF_DIRQ	Input state of HIF_DIRQ signal.	R	0x0
29	HIF_RDY	Input state of HIF_RDY signal.	R	0x0
28	HIF_CSN	Input state of HIF_CSN signal.	R	0x0
27	HIF_WRN	Input state of HIF_WRN signal.	R	0x0
26	HIF_RDN	Input state of HIF_RDN signal.	R	0x0
25	HIF_BHEN	Input state of HIF_BHEN signals.	R	0x0
24	reserved	-	R	0x0
23	HIF_A15	Input state of HIF_A15 signal.	R	0x0
22	HIF_A14	Input state of HIF_A14 signal.	R	0x0
21	HIF_A13	Input state of HIF_A13 signal.	R	0x0
20	HIF_A12	Input state of HIF_A12 signal.	R	0x0
19	HIF_A11	Input state of HIF_A11 signal.	R	0x0
18	HIF_A10	Input state of HIF_A10 signal.	R	0x0
17	HIF_A9	Input state of HIF_A9 signal.	R	0x0
16	HIF_A8	Input state of HIF_A8 signal.	R	0x0
15	HIF_A7	Input state of HIF_A7 signal.	R	0x0
14	HIF_A6	Input state of HIF_A6 signal.	R	0x0
13	HIF_A5	Input state of HIF_A5 signal.	R	0x0
12	HIF_A4	Input state of HIF_A4 signal.	R	0x0
11	HIF_A3	Input state of HIF_A3 signal.	R	0x0
10	HIF_A2	Input state of HIF_A2 signal.	R	0x0
9	HIF_A1	Input state of HIF_A1 signal.	R	0x0
8	HIF_A0	Input state of HIF_A0 signal.	R	0x0
7	HIF_D7	Input state of HIF_D7 signal.	R	0x0
6	HIF_D6	Input state of HIF_D6 signal.	R	0x0
5	HIF_D5	Input state of HIF_D5 signal.	R	0x0
4	HIF_D4	Input state of HIF_D4 signal.	R	0x0
3	HIF_D3	Input state of HIF_D3 signal.	R	0x0
2	HIF_D2	Input state of HIF_D2 signal.	R	0x0
1	HIF_D1	Input state of HIF_D1 signal.	R	0x0
0	HIF_D0	Input state of HIF_D0 signal.	R	0x0

3.5 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, allowing the firmware to detect which resets were active. 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

RESET_CTRL – Reset Control Register **0x101c000c**

31 30 29 28 27 26 25	24	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	RES_REQ_FIRMWARE	FIRMWARE_STATUS3	FIRMWARE_STATUS2	FIRMWARE_STATUS1	FIRMWARE_STATUS0	reserved	DIS_RES_XPEC0	reserved	RES_XPEC0	RES_FIRMWARE	RES_HOST	RES_WDOG	reserved

Bits	Name	Description	R/W	Default
31:25	reserved	-	R	0x0
24	RES_REQ_FIRMWARE	(software reset) writing a "1" sets the reset request to reset the whole system (write only)	W	0x0
23	FIRMWARE_STATUS3	readable and writable bit to save the firmware status; only a PowerOn Reset will clear this bit	R/W	0x0
22	FIRMWARE_STATUS2	readable and writable bit to save the firmware status; only a PowerOn Reset will clear this bit	R/W	0x0
21	FIRMWARE_STATUS1	readable and writable bit to save the firmware status; only a PowerOn Reset will clear this bit	R/W	0x0
20	FIRMWARE_STATUS0	readable and writable bit to save the firmware status; only a PowerOn Reset will clear this bit	R/W	0x0
19:17	reserved	-	R	0x0
16	DIS_RES_XPEC0	reset from RES_XPEC0 is disabled (read only)	R	0x0
15:5	reserved	-	R	0x0
4	RES_XPEC0	reset from RES_XPEC0, after reading write back a "1" to clear the status bit	R/W	0x0
3	RES_FIRMWARE	reset from FIRMWARE (software reset), after reading write back a "1" to clear the status bit	R/W	0x0
2	RES_HOST	reset from Hostinterface/DPM, after reading write back a "1" to clear the status bit	R/W	0x0
1	RES_WDOG	reset from System WDG, after reading write back a "1" to clear the status bit	R/W	0x0
0	reserved	-	R	0x0

The firmware can disable the internal system reset signals to the XPEC module, allowing this module 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 XPEC.

3.6 Clock Control – Clock Generation and Control

The netX10 provides some 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 PHY (~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 enabling the fieldbus0 clock. Fieldbus0 clock is derived from an internal 400MHz clock by a predivider combined with a rate multiplier, allowing low jitter clocks, for fieldbus systems with a frequency that can not directly be derived from the 100MHz system clock. Alternatively to this internally generated clock, an external clock (xm0_eclk) can be used to make xMAC output jitter free (CLK_EN-FB0).

CLK_EN – Global Clock Enable Register **0x101c0028**

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

Changes will only have effect if according bit in CLK_EN_MSK register is set. Bits will be reset according to the CLK_EN_MSK register, if a new mask is correctly written (netX locking algorithm).

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 … 15	14	13	12	11	10	9	8	7 … 5	4	3 … 1	0
reserved	XC_MISC	XPIC	DMA	reserved	FB0	reserved	DPM	reserved	XMAC0	reserved	XPEC0

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XC_MISC	enables clock for misc. XC logic (XC-DMAC, XC-SR, XC-BUFMAN,...)	R/W	0x0
13	XPIC	enables clock for XPIC	R/W	0x0
12	DMA	enables clock for DMA-Ctrl	R/W	0x0
11	reserved	-	R	0x0
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	0x0
9	reserved	-	R	0x0
8	DPM	enables clock for DPM	R/W	0x0
7:5	reserved	-	R	0x0
4	XMAC0	enables clock for xMAC0	R/W	0x0
3:1	reserved	-	R	0x0
0	XPEC0	enables clock for xPEC0	R/W	0x0

CLK_EN_MSK – Global Clock Enable Mask Register 0x101c002c

This register allows disabling modules for different netX-versions. It is lockable by netX locking algorithm. It will be only reset on Power on, not on normal system nres.

The CLK_EN register will change according to this register if a new mask is correctly written (netX locking algorithm).

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 … 15	14	13	12	11	10	9	8	7 … 5	4	3 … 1	0
reserved	XC_MISC	XPIC	DMA	reserved	FB0	reserved	DPM	reserved	XMAC0	reserved	XPEC0

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XC_MISC	0: misc. XC logic is disabled 1: clock can be enabled/disabled by CLK_EN register for misc. XC logic	R/W	0x1
13	XPIC	0: xPIC is disabled 1: clock can be enabled/disabled by CLK_EN register for xPIC	R/W	0x1
12	DMA	0: DMA-Ctrl. is disabled 1: clock can be enabled/disabled by CLK_EN register for DMA-Ctrl.	R/W	0x1
11	reserved	-	R	0x0
10	FB0	0: fieldbus0 clock is disabled 1: clock can be enabled/disabled by CLK_EN register for fieldbus0 clock	R/W	0x1
9	reserved	-	R	0x0
8	DPM	0: DPM is disabled 1: clock can be enabled/disabled by CLK_EN register for DPM	R/W	0x1
7:5	reserved	-	R	0x0
4	XMAC0	0: xMAC0 is disabled 1: clock can be enabled/disabled by CLK_EN register for xMAC0	R/W	0x1
3:1	reserved	-	R	0x0
0	XPEC0	0: xPEC0 is disabled 1: clock can be enabled/disabled by CLK_EN register for xPEC0	R/W	0x1

The following registers are used to control frequencies of internal clocks for armclk, usb12clk, adcclk and fieldbus clocks. The 100MHz 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 0x101c0014

This register can be used to change internal system frequency (100MHz of ARM and system). Be careful when changing this value, as proper netX functionality is only qualified for the default value.

This register is lockable by netX locking algorithm. It will be only reset on Power on, not on normal system nres.

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 ... 9	8 ... 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: armclk_rate_mul_add = [freq in MHz] / 200 * 2^9. Note: SDRAM data sampling loopback clock bypass is reconfigured automatically If clock rate is set below 80MHz (view SDRAM_TIMING_CTRL register description).	R/W	0x100

USB12_CLK_RATE_MUL_ADD – Rate Multiplier Add Value of 12MHz USB clock 0x101c0018

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 access key from ACCESS_KEY register
2.: write back access key to ACCESS_KEY register
3.: write desired value to this register

31–16	15–0
reserved	USB12CLK_RATE_MUL_ADD

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	USB12CLK_RATE_MUL_ADD	This value is added each clk400 cycle to usb12clk_rate_mul to generate usb12clk. Change value according to formula: usb12clk_rate_mul_add = [freq in MHz] / 400 * 2^16	R/W	0x7ae

ADC_CLK_DIV – Divisor of clock divider for 16MHz ADC clock 0x101c001c

This register can be used to change the ADC 16MHz 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 access key from ACCESS_KEY register
2.: write back access key to ACCESS_KEY register
3.: write desired value to this register

31–9	8–0
reserved	VAL

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8:0	VAL	Divisor for generating 16MHz adcclk out of clk400: Change value according to formula: adcclk_div = 400 / [freq in MHz]	R/W	0x19

FB0CLK_RATE_MUL_ADD – Rate Multiplier Add Value 0x101c0020

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_CFG-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 (CLK_EN-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 access key from ACCESS_KEY register
2.: write back access key to ACCESS_KEY register
3.: write desired value to this register

31 … 0
FB0CLK_RATE_MUL_ADD

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. Values bigger 0x80000000 are not allowed for proper rate_mul functionality. fb0clk_rate_mul_add[31:30] == 2'b11 define a special mode, where rate_mul is forwarding its input clock. Change value according to formula: fb0clk_rate_mul_add = [freq in MHz] / 400 * 2^32 * (fb0clk_div+1)	R/W	0x1000000

FB0CLK_DIV – Rate Multiplier Predivider **0x101c0024**

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

Alternatively to this internally generated clock, an external clock (xm0_eclk) can be used to make xMAC output jitter free (CLK_EN-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 access key from ACCESS_KEY register
2.: write back access key to ACCESS_KEY register
3.: write desired value to this register

31 ... 8	7 ... 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:8	-	reserved	R	0x0
7:0	VAL	Fieldbus 0 Predivider value: The value + 1 must be programmed, i.e. val=0 leads to no predivision Change value according to formula: fb0clk_div = (400 / [freq in MHz] * fb0clk_rate_mul_add / 2^32) - 1	R/W	0x0

3.7 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_{IRQ} = WDG_IRQ_TIMEOUT x 100 µs

T_{RESET} = (WDG_IRQ_TIMEOUT + WDG_RESET_TIMEOUT) x 100 µs

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

WDG_TRIG – netX System Watchdog Trigger Register **0x101c0200**

The watchdog access code is generated by a pseudo random generator.

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.

31	30	29	28	27 26 25	24	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_ENABLE	reserved	WDG_ACTIVE_ENABLE_W	WDG_COUNTER_TRIGGER_W	reserved	IRQ_REQ_WATCHDOG	reserved	WDG_ACCESS_CODE

Bits	Name	Description	R/W	Default
31	WRITE_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	0x0
30	reserved	-	R	0x0
29	WDG_ACTIVE_ENABLE_W	Watchdog Active Enable: If this bit is set, the WDGACT output signal (PIN G17) is enabled. When read, this bit is always '0'	R/W	0x0
28	WDG_COUNTER_TRIGGER_W	Watchdog trigger bit: Bit must be set to trigger the watchdog counter. When read, this bit is always '0'	R/W	0x0
27:25	reserved	-	R	0x0
24	IRQ_REQ_WATCHDOG	IRQ request of watchdog, writing 1 deletes IRQ	R/W	0x0
23:20	reserved	-	R	0x0
19:0	WDG_ACCESS_CODE	Watchdog access code for triggering. A read access gives the next 16 bit code for trigger. A write access with correct access code will trigger the watchdog counter.	R/W	0x0

WDG_CNTR – netX System Watchdog Register 0x101c0204

31:17	16:0
reserved	WDG_COUNTER

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16:0	WDG_COUNTER	Actual watchdog counter value	R	0x0

WDG_IRQ_TIMEOUT – netX System Wachtdog Interrupt Timout Register 0x101c0208

31:16	15:0
reserved	WDG_IRQ_TIMEOUT

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	WDG_IRQ_TIMEOUT	Watchdog interrupt timeout	R/W[1]	0x0

WDG_RESET_TIMEOUT – netX System Watchdog Reset Timeout Register 0x101c020c

31:16	15:0
reserved	WDG_RES_TIMEOUT

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	WDG_RES_TIMEOUT	Watchdog Reset Request Timeout	R/W[1]	0x0

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

3.8 SYS_STAT – System Status

The general status of a netX based system is displayed by the System LED(s). It is recommended to use a dual LED here, but two single LEDs can also be used. The general definition of this LED is:

RDY	yellow	the netX with operating system is running
RUN	green	the user application is running without errors

However, after booting a firmware, the LEDs are firmware controlled and their behaviour is hence completely application- or firmware specific.

ONLY_PORN – Firmware Status Register **0x101c0034**

This register is not reset by SW resets, only PORn will reset this register.

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 ... 0
ONLY_PORN

Bits	Name	Description	R/W	Default
31:0	ONLY_PORN	netX Firmware status	R/W	0x0

SAMPLE_AT_NRES – IO Sampled at Reset Status Register 0x101c0040

Note: Configure sample_at_nres (sar_*)-IOs with pull-ups or down resistors to configure netX environment (e.g. DPM enable, DPM serial mode selection...). Related IOs are not driven by netX by default. For correct functionality ensure that they are also not driven by external devices during netx power up and reset.

Note: View ./README_power_on_cfg.txt for HSoCT reference simulation sample at reset configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAR_SPI0_CLK	SAR_SPI0_MISO	SAR_SPI0_MOSI	SAR_SPI0_SIO2	SAR_SPI0_SIO3	SAR_HIF_A10	SAR_HIF_A9	SAR_HIF_A8	SAR_HIF_A7	SAR_HIF_A6	SAR_HIF_A5	SAR_HIF_A4	SAR_HIF_A3	SAR_HIF_A2	SAR_HIF_A1	SAR_HIF_A0	SAR_HIF_D15	SAR_HIF_D14	SAR_HIF_D13	SAR_HIF_D12	SAR_HIF_D11	SAR_HIF_D10	SAR_HIF_D9	SAR_HIF_D8	SAR_HIF_D7	SAR_HIF_D6	SAR_HIF_D5	SAR_HIF_D4	SAR_HIF_D3	SAR_HIF_D2	SAR_HIF_D1	SAR_HIF_D0

Bits	Name	Description	R/W	Default
31	SAR_SPI0_CLK	Sampled input level of IO ’spi0_clk’ at power on reset	R	0x0
30	SAR_SPI0_MISO	Sampled input level of IO ’spi0_miso’ at power on reset	R	0x0
29	SAR_SPI0_MOSI	Sampled input level of IO ’spi0_mosi’ at power on reset	R	0x0
28	SAR_SPI0_SIO2	Sampled input level of IO ’spi0_sio2’ at power on reset	R	0x0
27	SAR_SPI0_SIO3	Sampled input level of IO ’spi0_sio3’ at power on reset	R	0x0
26	SAR_HIF_A10	Sampled input level of IO ’hif_a10’ at power on reset	R	0x0
25	SAR_HIF_A9	Sampled input level of IO ’hif_a9’ at power on reset	R	0x0
24	SAR_HIF_A8	Sampled input level of IO ’hif_a8’ at power on reset	R	0x0
23	SAR_HIF_A7	Sampled input level of IO ’hif_a7’ at power on reset	R	0x0
22	SAR_HIF_A6	Sampled input level of IO ’hif_a6’ at power on reset	R	0x0
21	SAR_HIF_A5	Sampled input level of IO ’hif_a5’ at power on reset	R	0x0
20	SAR_HIF_A4	Sampled input level of IO ’hif_a4’ at power on reset	R	0x0
19	SAR_HIF_A3	Sampled input level of IO ’hif_a3’ at power on reset	R	0x0
18	SAR_HIF_A2	Sampled input level of IO ’hif_a2’ at power on reset	R	0x0
17	SAR_HIF_A1	Sampled input level of IO ’hif_a1’ at power on reset	R	0x0
16	SAR_HIF_A0	Sampled input level of IO ’hif_a0’ at power on reset	R	0x0
15	SAR_HIF_D15	Sampled input level of IO ’sar_hif_d15’ at power on reset	R	0x0
14	SAR_HIF_D14	Sampled input level of IO ’sar_hif_d14’ at power on reset	R	0x0
13	SAR_HIF_D13	Sampled input level of IO ’sar_hif_d13’ at power on reset	R	0x0
12	SAR_HIF_D12	Sampled input level of IO ’sar_hif_d12’ at power on reset	R	0x0
11	SAR_HIF_D11	Sampled input level of IO ’sar_hif_d11’ at power on reset	R	0x0
10	SAR_HIF_D10	Sampled input level of IO ’sar_hif_d10’ at power on reset	R	0x0
9	SAR_HIF_D9	Sampled input level of IO ’sar_hif_d9’ at power on reset	R	0x0
8	SAR_HIF_D8	Sampled input level of IO ’sar_hif_d8’ at power on reset	R	0x0
7	SAR_HIF_D7	Sampled input level of IO ’sar_hif_d7’ at power on reset	R	0x0
6	SAR_HIF_D6	Sampled input level of IO ’sar_hif_d6’ at power on reset	R	0x0
5	SAR_HIF_D5	Sampled input level of IO ’sar_hif_d5’ at power on reset	R	0x0
4	SAR_HIF_D4	Sampled input level of IO ’sar_hif_d4’ at power on reset	R	0x0
3	SAR_HIF_D3	Sampled input level of IO ’sar_hif_d3’ at power on reset	R	0x0
2	SAR_HIF_D2	Sampled input level of IO ’sar_hif_d2’ at power on reset	R	0x0
1	SAR_HIF_D1	Sampled input level of IO ’sar_hif_d1’ at power on reset	R	0x0
0	SAR_HIF_D0	Sampled input level of IO ’sar_hif_d0’ at power on reset	R	0x0

NETX_STATUS – netX System Status Configuration Register **0x101c0044**

This Register was implemented in Hilscher HIF module originally. From Hilscher Program Reference Guide: 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. Access to this register is not protected by any locking or access protection algorithm.

IMPORTANT: netX50/100/500 Change Note:

The netX50/100/500 SYS_STA register was byte accessible. This changed: This register is only 32bit accessible.

In netx50/100/500, write access to bits 0..15 of SYS_STA register can generate an IRQ to external host CPU. As the register now is 32bit accessible only, this is changed to whole register access. I.e. any write access to this register will generate a host IRQ if enabled.

To change the upper 16 bits of this register without host IRQ generation, use register RDY_RUN_CFG.
Note:
Changing bits here will also change RDY_RUN_CFG register bits.

Note:
Bits 0..3 and 8..15 are read-only-mirrored to DPM/Host Status register DPM_HOST_SYS_STAT(Area DPM).
Read-only bits 4..7 can be programmed by DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM).

31 30 29 28 27 26	25	24	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_RO	NETX_STATE	RUN	RDY

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	RUN_DRV	Driver enable for RUN LED. Enables output driver when set.	R/W	0x0
24	RDY_DRV	Driver enable for RDY LED. Enables output driver when set.	R/W	0x0
23:20	reserved	-	R	0x0
19	RUN_POL	Output polarity RUN LED; outsig = RUN exor RUN_POL.	R/W	0x0
18	RDY_POL	Output polarity RDY LED; outsig = RDY exor RDY_POL.	R/W	0x0
17	RUN_IN	Physical input signal level at RUN pin (read-only).	R	0x0
16	RDY_IN	Physical input signal level at RDY pin (read-only).	R	0x0
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. Note: These bits are read-only-mirrored to DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM). Changing these bits can produce an IRQ to host CPU.	R/W	0xf0
7:4	HOST_STATE_RO	Host Status Code. User defined status is read only here. These bits can be programmed by DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM).	R	0x0
3:2	NETX_STATE	User defined status bits. Note: These bits are read-only-mirrored to DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM). Changing these bits can produce an IRQ to host CPU.	R/W	0x0

Bits	Name	Description	R/W	Default
1	RUN	Signal Level of the RUN LED output. Note: This bit is read-only-mirrored to DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM). Changing this bit can produce an IRQ to host CPU.	R/W	0x0
0	RDY	Signal level of the RDY LED output. Note: This bit is read-only-mirrored to DPM/Host Status register DPM_HOST_SYS_STAT (Area DPM). Changing this bit can produce an IRQ to host CPU.	R/W	0x0

RDY_RUN_CFG – netX RDY/RUN IO System Status Configuration Register 0x101c0048

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.

Access to this register is not protected by any locking or access protection algorithm.

Note:
Use this register to change the upper 16 bits of NETX_STATUS register without host IRQ generation. For further information see NETX_STATUS register description. Changing bits here will also change NETX_STATUS register bits, however no host IRQ will be generated.

31 30 29 28 27 26	25	24	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	reserved	RUN	RDY

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	RUN_DRV	Driver enable for RUN LED. Enables output driver when set.	R/W	0x0
24	RDY_DRV	Driver enable for RDY LED. Enables output driver when set.	R/W	0x0
23:20	reserved	-	R	0x0
19	RUN_POL	Output polarity RUN LED; outsig = RUN exor RUN_POL.	R/W	0x0
18	RDY_POL	Output polarity RDY LED; outsig = RDY exor RDY_POL.	R/W	0x0
17	RUN_IN	Physical input signal level at RUN pin (read-only).	R	0x0
16	RDY_IN	Physical input signal level at RDY pin (read-only).	R	0x0
15:2	reserved	-	R	0x0
1	RUN	Signal Level of the RUN LED output.	R/W	0x0
0	RDY	Signal level of the RDY LED output.	R/W	0x0

Note:
If the netX10 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!

SYSTEM_STATUS – netX System Status Register 0x101c004c

This register provides information of special netX system events, e.g: System related interrupt activity, abort activity.

31	30	29	28	27	26	25	24	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																														EXTBUS_TO_IRQ_STATUS	reserved

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	EXTBUS_TO_IRQ_STATUS	Current status of Extension Bus Ready Timeout IRQ. Note: This IRQ is controlled / cleared by EXT_RDY_CFG register (area MEM_SRAM_CTRL).	R	0x0
0	reserved	-	R	0x0

3.9 NETX_LIC – netX License

NETX_LIC_ID – netX License ID Register 0x101c0050

This register contains license information read from security memory during boot phase.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">ID</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:0	ID	License ID from security memory	R	0x0

NETX_LIC_FLAG0 – netX License Flag0 Register 0x101c0054

This register is part of netX licence error detection mechanism. If netX software requested an unavailable licence, this will be flagged in NETX_LIC_ERRORS0 register.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">FLAGS</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:0	FLAGS	License flag bits from security memory	R	0x0

NETX_LIC_FLAG1 – netX License Flag1 Register 0x101c0058

This register is part of netX licence error detection mechanism. If netX software requested an unavailable licence, this will be flagged in NETX_LIC_ERRORS1 register.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="32">FLAGS</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:0	FLAGS	License flag bits from security memory	R	0x0

NETX_LIC_ERRORS0 – netX License Errors0 Status Register 0x101c005c

This register is part of netX licence error detection mechanism. If netX software requested a licence not provided BY NETX_LIC_FLAGS0, this will be flagged here. This register contains 0 in case of no license error.

31 … 0
ERR_RO

Bits	Name	Description	R/W	Default
31:0	ERR_RO	License error bits set in case of license mismatch according to netx_lic_flags0 (OR of all occurred errors)	R	0x0

NETX_LIC_ERRORS1 – netX License Errors1 Status Register 0x101c0060

This register is part of netX licence error detection mechanism. If netX software requested a licence not provided by NETX_LIC_FLAGS1, this will be flagged here. This register contains 0 in case of no license error.

31 … 0
ERR_RO

Bits	Name	Description	R/W	Default
31:0	ERR_RO	License error bits set in case of license mismatch according to netx_lic_flags1 (OR of all occurred errors)	R	0x0

4 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 as well as SDRAM data bus width can be set to 8 or 16 Bit.

The memory controller shares its signals with the netX10 DPM interface on HIF IOs. Hence external memory can only be connected when the DPM interface is not used (the only exception is when using the DPM interface in serial (SPI) mode, which allows to operate 8 Bit memory devices at the same time).

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

ARM Address area	**Short Description**
0x80000000 – 0xBFFFFFFFF	SDRAM Chip Select Area
0xC0000000 – 0xC7FFFFFFF	SRAM/FLASH Chip Select Area 0
0xC8000000 – 0xCFFFFFFFF	SRAM/FLASH Chip Select Area 1
0xD0000000 – 0xD7FFFFFFF	SRAM/FLASH Chip Select Area 2
0xD8000000 – 0xDFFFFFFFF	SRAM/FLASH Chip Select Area 3

The following table provides a summary of memory controller registers.

ARM Address	**Register Name**	**Short Description**
0x101c0100	MEM_SRAM0_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem0 Chip Select Area
0x101c0104	MEM_SRAM1_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem1 Chip Select Area
0x101c0108	MEM_SRAM2_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem2 Chip Select Area
0x101c010c	MEM_SRAM3_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem3 Chip Select Area
0x101c0110	EXT_CS0_APM_CTRL	Asynchronous Page Mode (APM) Control Register for ExtMem0 Chip Select Area
0x101c0120	EXT_RDY_CFG	External Memory Ready Control Register
0x101c0124	EXT_RDY_STATUS	External Memory Ready Status Register
0x101c0140	MEM_SDRAM_CFG_CTRL	Memory SDRAM Configuration Control Register
0x101c0144	MEM_SDRAM_TIMING_CTRL	Memory SDRAM Timing Control Register
0x101c0148	MEM_SDRAM_MODE	Memory SDRAM Mode Register

4.1 MEM_SRAM – Memory Controller for SRAM and FLASH

The SRAM / FLASH Interface provides a total of four memory areas, which provide their own chip select signals and independent configuration registers, allowing to set bus width and wait state parameters separately for each area. As one of these chip selects (MEMSR_CS1n) is shared with the SDRAM chip select signal, this chip select area is not available for SRAM / FLASH if SDRAM is connected.

The parameters allow bus width configurations of 8 and 16 Bit, wait states of up to 63 clock cycles and enabling or disabling the Ready/Busy signal. The Ready signal parameters (polarity, etc.) are commonly set for all chip select areas and there is also a single configuration register for the parameters related to the APM (Asynchronous Page Mode), which is only supported by chip select area 0.

Unlike with other netX controllers (netX50/100/500), address lines A23:0 always represent the byte address, regardless of the bus width of the connected component. To allow single byte access in 16Bit mode, two Byte Lane signals (MEMSR_DQM0 = MEM_A00 and MEM_DQM1) are provided.

The maximum of addressable external SRAM / FLASH memory is 32MB each, in chip select area 0 and 1, which is reduced to 16MB if Chip Select Area 2 is used (requires signal MEMSR_CS2n which is shared with address line 23) and to 8MB if Chip Select Areal 3 is used (requires signal MEMSR_CS3n which is shared with address line 22).

The following table provides a summary of addressable external SRAM/FLASH memory.

Setup using	**Maximum addressable SRAM / FLASH memory**
Chip Select 0 only (e.g. SDRAM used)	32MB
Chip Select 0 and 1	64MB (2 * 32MB)
Chip Select 0, 1 and 2	48MB (3 * 16MB)
Chip Select 0, 1, 2, and 3	32MB (4 * 8MB)

MEM_SRAM0_CTRL – Memory SRAM Control Register for Chip Select Area 0 0x101c0100
MEM_SRAM1_CTRL – Memory SRAM Control Register for Chip Select Area 1 0x101c0104
MEM_SRAM2_CTRL – Memory SRAM Control Register for Chip Select Area 2 0x101c0108
MEM_SRAM3_CTRL – Memory SRAM Control Register for Chip Select Area 3 0x101c010c

For 16 bit memory devices, A0 is used as byte low enable. Byte high enable is provided on additional signal.

For 32 bit interfaces, A0 and A1 are used as byte enables. Additionally there are 2 further byte enable signals provided.

For all wait state configuration 1 cycle is 1 netX system clock cycle, i.e. 10ns for netX running on 100MHz at normal operation.

Note:
Pause and data width configuration is compatible to netx500/100 and netx50.

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

Bits	Name	Description	R/W	Default
31	READY_EN	Ready Signal Enable. 0: Access timing is only controlled by Wait-State and Pre/Post-Pause configuration above. 1: Use external ready input to stretch Wait-State phase. Wait-States and Pre/Post-Pauses will be done according to configuration above. However Wait-State phase can be extended by an external device by holding netX ready input inactive. Data access cycle is done after external device sets netX ready input to active state. Note: An external device must assert ready to inactive state while Wait-States phase is running (defined by ws in this register). Ready input sampling and latency takes 20ns. Hence ws must be set to a value greater than 2 for proper functionality using ready. The value must be increased if there is a ready setup time of the ready generating external device. Note: For detailed ready input configuration and handling view EXT_RDY_CFG register description.	R/W	0x0
30	reserved	-	R	0x0
29	NO_P_POST_SEQ_RD	No Post-Pause insertion between sequential reads. 0: Post-Pause will be inserted after each read access. 1: Disable Post-Pause between sequential reads. Note: default setting '0' is for netx100/50 compatibility only. Typically there is no need of Post-Pause insertion between sequential reads. A Post-Pause will always be inserted if the next access addresses another chip-select area, is a write access or is not predictable by the memory controller.	R/W	0x0
28	NO_P_PRE_SEQ_RD	No Pre-Pause insertion between sequential reads. 0: Pre-Pause will be inserted after each read access. 1: Disable Pre-Pause between sequential reads. Note: default setting '0' is for netx100/50 compatibility only. Typically there is no need of Pre-Pause insertion between sequential reads.	R/W	0x0
27:26	reserved	-	R	0x0
25:24	DWIDTH	Data bus width of ExtMem area.	R/W	0x3

Bits	Name	Description	R/W	Default
		00 : 8bit memory device connected to this chip-select address area. 01 : 16bit memory device connected to this chip-select address area. 10 : reserved. 11 : memory is disabled, related chip select signal can be used for other purpose (e.g. as PIO). Note: This chip select is enabled by default but may be shared with other functions. View memory interface multiplex options for more information.		
23:18	reserved	-	R	0x0
17:16	P_POST	Post-Pause (0 - 3 cycles) of ExtMem area. Additional wait-states to match memory device Output-Disable or Address-Hold times. If programmed value is not 0, this Post-Pause will be inserted at external access end after Wait-State phase and data access cycle. Address, chip-select and byte-enable signals will remain stable in this phase. But nRD-signal and nWR-signal will become inactive high. After write access netX memory controller will always insert at least 1 Post-Pause cycle to generate positive edge on nWR-signal.	R/W	0x3
15:10	reserved	-	R	0x0
9:8	P_PRE	Pre-Pause (0 - 3 cycles) of ExtMem area. Additional wait-states to match memory device setup times. If programmed value is not 0, this Pre-Pause will be inserted at external access start before Wait-State phase is started. Address, chip-select and byte-enable signals will be stable in this phase. But nRD-signal and nWR-signal remains inactive high.	R/W	0x3
7:6	reserved	-	R	0x0
5:0	WS	Wait-States (0 - 63 cycles) of ExtMem area. During read access this is nRD-signal active low phase. During write access this is nWR-signal active low phase. Address, chip-select and byte-enable signals remain stable in this phase. After wait cycles have passed signals remain stable and, data-access cycle is done. To match memory device data access time tACC: program WS=ceil (tACC/10ns)-1.	R/W	0x3f

EXT_CS0_APM_CTRL – Asynchronous Page Mode (APM) Control Register for ExtMem0 Chip Select Area 0x101c0110

Only ExtMem0 chip-select area supports fast Asynchronous-Page-Mode (APM) Access.

31–11	10–8	7–4	3–0
reserved	APM_CFG	reserved	WS_APM

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x0
10:8	APM_CFG	APM configuration. 000 : read bursts are disabled 001 : 1 Dword address boundary for APM 010 : 2 Dword address boundary for APM 011 : 4 Dword address boundary for APM 100 : 8 Dword address boundary for APM 101 : 16 Dword address boundary for APM 110 : 32 Dword address boundary for APM all other settings are reserved.	R/W	0x0
7:4	reserved	-	R	0x0
3:0	WS_APM	APM read burst wait-states (0 - 15 cycles). If APM is enabled by apm_cfg-bits, first read access is done with number of wait-states programmed in extsram0_ctrl register. Following read accesses to ExtMem0 chip-select area are done with wait-states programmed here until APM-accesses are terminated. If netX runs internal read bursts only netX address lines will change. Chip-select and nRD signals will remain active low. APM accesses are terminated if chip-select of ExtMem0 address area becomes inactive, if write access is done between read accesses or if read access is leaving APM address boundary. Note: Chip-select remains active low after read even if no further access is currently requested by netX. Chip-select will become inactive, if access to another external chip-select area is requested or if external memory bus is shared with SDRAM and netX SDRAM-Controller performs access or refresh cycles.	R/W	0xf

EXT_RDY_CFG – External Memory Ready Control Register 0x101c0120

Note: Timeout is generated if ready usage is enabled by the MEM_SRAM*_CTRL registers and is not asserted to active state within 10us.

31 30 29 28 27 26 25 24 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	RDY_TO_DIS	reserved	RDY_TO_IRQ_EN	reserved	RDY_FILTER	reserved	RDY_ACT_LEVEL

Bits	Name	Description	R/W	Default
31:12	reserved	-	R	0x0
11	RDY_TO_DIS	Ready Timeout Disable By default ready timeout is enabled. Timeout is generated if ready usage is enabled by the MEM_SRAM*_CTRL registers and is not asserted to active state within 10us (1024 system clocks). If an external device requires even longer response time, ready timeout can be disabled by setting this bit. However be careful: If ready is not asserted anytime, netX system will stall. Escape from this can only be achieved by Hardware Reset (e.g. by system watchdog timeout). 0: Ready timeout is enabled. 1: Ready timeout is disabled.	R/W	0x0
10:9	reserved	-	R	0x0
8	RDY_TO_IRQ_EN	Ready Timeout IRQ Enable 0: No IRQ generation in case of ready timeout. 1: generate an IRQ in case of ready timeout. Note: Ready Timeout IRQ is part of netX System Status IRQ (view SYSTEM_STATUS register in area ASIC_CTRL and VIC registers)	R/W	0x0
7:6	reserved	-	R	0x0
5:4	RDY_FILTER	Ready Input Filter. Ready input filtering is implemented to avoid false ready active detection especially if ready signal is not always driven and ready active state is realized by pull-up or down resistors. 00: Ready active state is detected after ready signal is sampled once in active state (no filtering). 01: Ready active state is detected after ready signal is consecutively sampled twice in active state. 10: Ready active state is detected after ready signal is consecutively sampled 3 times in active state. 11: Ready active state is detected after ready signal is consecutively sampled 4 times in active state. Note: If ready is sampled in inactive state, active state counting will restart at zero. Note: If ready input filtering is enabled, access time will be increased at least by filter time (ready is sampled any 10ns).	R/W	0x0
3:1	reserved	-	R	0x0
0	RDY_ACT_LEVEL	Ready Active Level 0: Ready is active low / stall access while ready input is high. 1: Ready is active high / stall access while ready input is low.	R/W	0x1

EXT_RDY_STATUS – External Memory Ready Status Register 0x101c0124

Note: Timeout is generated if ready usage is enabled by the MEM_SRAM*_CTRL registers and is not asserted to active state within 10us.

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

Bits	Name	Description	R/W	Default
31	RDY_TO_ERR	Ready Timeout Error. This bit is set if a ready timeout error is detected. The external address and chip-select will be logged then in the lower bits of this register. An IRQ/Abort will be generated if enabled by the EXT_RDY_CFG register. Writing a '1' here will reset this bit and the IRQ. Note: If multiple timeouts are detected, the first timeout address and chip-select will be logged. Note: Ready Timeout IRQ is part of netX System Status IRQ (view SYSTEM_STATUS register in area ASIC_CTRL and VIC registers)	R	0x0
30:26	reserved	-	R	0x0
25:24	RDY_TO_ERR_CS	Ready timeout error chip-select logging.	R	0x0
23:0	RDY_TO_ERR_ADR	Ready timeout error address logging.	R	0x0

4.2 MEM_SDRAM – Memory Controller for SDRAM

The SDRAM controller can drive all SDRAM Single Data Rate Types from 16 MBit to 512 MBit, providing a (not completely used) 1 GByte address space from 0x80000000 to 0xBFFFFFFF. The SDRAM controller is equipped with 8 Byte read and write caches which are always active.

The following parameters can be set:

- Number of banks 2, 4
- Number of rows 2k, 4k, 8k, 16k
- Number of columns 256, 512, 1k, 2k, 4k
- Data width 16 or 8 Bit
- Refresh-mode Average refresh interval of 3.9, 7.8, 15.6 or 31.2us. Fixed or collect up to 8, 16 or 2047 refresh cycles
- Power save mode SDRAM-Self-refresh-Mode with disabled clock

Note:
For further information on the SDRAM Memory Controller, please refer to the netX10_Product_Brief and netX10_Technical_Reference_Guide.

MEM_SDRAM_CFG_CTRL – Memory SDRAM Configuration Control Register 0x101c0140

For initializing procedure of netX SDRAM Controller, view description of 'ctrl_en' bit inside 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
REFRESH_STATUS	SDRAM_READY	reserved	REFRESH_MODE	reserved	CTRL_EN	EXTCLK_EN	SDRAM_PWDN	DBUS16	reserved	COLUMNS	reserved	ROWS	reserved	BANKS

Bits	Name	Description	R/W	Default
31	REFRESH_STATUS	Refresh status flag. ------------------------------------ Due to new high priority refresh mode this bit is not required any longer to ensure correct refresh generation. ------------------------------------ Refresh behaviour changed from netx100/500/50: SDRAM Controller now has an additional high priority refresh mode (view REFRESH_MODE bit description). There is no need to guarantee sufficient SDRAM refresh generation by checking this bit by software any longer (necessary for netx100/500/50 depending on application). It is only for information purpose for netX10 or later. This bit can be reset by writing '0' to it. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
30	SDRAM_READY	This bit is set to 1 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. Note: This bit is a read only status flag.	R	0x0
29:26	reserved	-	R	0x0
25:24	REFRESH_MODE	Refresh request generation mode. Refresh behaviour changed from netx100/500/50: SDRAM Controller now has an additional high priority refresh mode. Refresh generation has lower priority than accesses on external memory interface normally. That means refreshes do not block data access. To avoid data loss under all conditions without	R/W	0x1

Bits	Name	Description	R/W	Default
		checking critical situations by software a high priority refresh mode is implemented for netX10 and later: If there was too much traffic to SDRAM to run refreshes according to programmed refresh_mode the controller changes to high priority refresh mode automatically. In this mode the controller generates immediately as many refreshes as required to avoid imminent data loss. After that the controller falls back to low priority refresh generation automatically. In normal low priority refresh mode refreshes can be collected. That means single refreshes are not necessarily done in programmed average refresh interval (t_REFI in MEM_SDRAM_TIMING_CTRL register). However the controller ensures by hardware that t_REFI is kept as mean refresh interval for a certain number of subsequent refreshes. This number of refreshes that will be collected to a long term refresh sequence can be programmed in this bit field. The following refresh request generation mode can be programmed: 00 : fix interval: expect one refresh any programmed refresh period (MEM_SDRAM_TIMING_CTRL.t_REFI) 01 : collect up to 8 refreshes (default) 10 : collect up to 16 refreshes 11 : collect up to 2047 refreshes Note: Typically SDRAM devices do not require a fix refresh interval. Collecting more refreshes will lead to improved performance (as high priority refresh mode blocking normal access is entered more often when only few refreshes can be collected). Hence, it is recommended setting this bit field to ’11’ (collecting up to 2047 refreshes). Note: Entering high priority refresh mode typically occurs when SDRAM becomes system performance bottleneck. To detect this, a status bit (refresh_status) will be set when high priority refresh mode was entered. It can be used for debugging or system status information purpose.		
23:20	reserved	-	R	0x0
19	CTRL_EN	SDRAM controller enable The MEM_SDRAM_TIMING_CTRL and the MEM_SDRAM_MODE registers can only be changed while this bit is 0. Initializing and enabling SDRAM should be done as follows: 0. If SDRAM was already enabled: Disable SDRAM controller by setting the ctrl_en-bit to 0. Ensure that no netX system master is trying to access SDRAM address area. Otherwise related master will be stalled (no ready) until re-enabling SDRAM. 1. Configure the MEM_SDRAM_TIMING_CTRL register: All timing parameters of the t_* bit fields must be taken from SDRAM device data sheet. All other timing parameters like clock and sample phases are provided by Hilscher. 2. Configure the MEM_SDRAM_MODE register: Typically only setting of correct CAS-Latency is required (CL2 or CL3 supported by netX SDRAM controller). CL2 provides better performance and should be preferred. Please read description of the MEM_SDRAM_MODE register for further details. 3. Configure the MEM_SDRAM_CFG_CTRL (this) register and enable the controller by setting the ’ctrl_en’ bit. The values for ’banks’, ’rows’ and ’columns’ depend on the used SDRAM device and must be taken from the related data sheet. 4. Wait until ’sdram_ready’ status bit is set before accessing SDRAM device. ------------------------------------ After enable, the controller will run the following SDRAM initialisation procedure (100MHz, 1 cycle = 10ns). command / cycles / time / comment NOP / 20050 / 200.5us / running sd_clk (if extclk_en), *cs low, cke high) PRECH ALL, NOP / 1+15 / 10ns + 150ns 7x(AUTO REF, NOP) / 7x(1+31) / 7x(10ns + 310ns) AUTO REF, / 1+22 / 10ns + 220ns	R/W	0x0

Bits	Name	Description	R/W	Default
		NOP LOAD MREG, NOP 1+3 10ns + 30ns with settings done by the sdram_mr registers ACTIVATE 1 10ns first access if requested, sdram_ready will be set to 1 here ------------------------------------ Attention: Accesses requested before sdram_ready is 1 will be blocked (no ready). This could cause system freezing. Note: The external SDRAM clock will not run if the controller is disabled.		
18	EXTCLK_EN	external SDRAM clock enable 0 : SDRAM clock disabled (default) 1 : SDRAM clock enabled Note: The external SDRAM clock will not run if the controller is disabled.	R/W	0x0
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. Return from power-down mode can be done by clearing this bit.	R/W	0x0
16	DBUS16	SDRAM data bus width 0 : SDRAM data bus is 8 bit wide (default) 1 : SDRAM data bus is 16 bit wide	R/W	0x0
15:11	reserved	-	R	0x0
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) all others: reserved	R/W	0x0
7	reserved	-	R	0x0
6:4	ROWS	Row address coding. 000: 2k (A0..A10) (default) 001: 4k (A0..A11) 010: 8k (A0..A12) 011: 16k (A0..A13) (BGA only, do not use in QFP as A13 is there used for CASn) All others: reserved	R/W	0x0
3:2	reserved	-	R	0x0
1:0	BANKS	Bank address coding. 00: 2 01: 4 (default) All others: reserved Bank addresses are always mapped on A15(=BA1) and A14(=BA0)	R/W	0x1

MEM_SDRAM_TIMING_CTRL – Memory SDRAM Timing Control Register 0x101c0144

Please view description of ‘CTRL_EN’ bit inside MEM_SDRAM_CFG_CTRL register for initializing-procedure of netX SDRAM Controller.

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

Important:
For correct SDRAM function it is absolutely necessary to set following SDRAM timing parameters (does not depend on used SDRAM device):

bit field: MEM_SDCLK_PHASE: 2
bit field: DATA_SAMPLE_PHASE: 1
bit: MEM_SDCLK_SSNEG: 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			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	0x0
28	BYPASS_NEG_DELAY	Bypass data sample clock phase shift. 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. Bypass must be used for system clock frequencies <= 80MHz (rate_mull_add <= 0xC0). If this bit is programmed with ’0’ by software but system clock frequency is below 80MHz, it will be changed to ’1’ to enable bypass automatically. When system frequency is changed to a rate more than 80MHz, the bit is released to ’0’ again. This allows entering netX power save mode entry and leave without reconfiguring this bit by software. However take care that no SDRAM access is running at the moment of system clock frequency change around the 80MHz border. Note: The bit will always remain ’1’ if it is programmed high. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
27	reserved	-	R	0x0
26:24	DATA_SAMPLE_PHASE	Data sample clock phase shift. 0..5: adjustable phase-shift for data sampling SDRAM loopback clock (clk_sdloopback) depending on external capacitive load and SDRAM access time (t_AC). The phase can be shifted in 1.25ns 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 respected. Data sampling has to be done at least 8ns after internal changes of SDRAM ctrl-signals (MEM_SD*-signals, driven by clk_memsig).	R/W	0x3
23	MEM_SDCLK_SSNEG	MEM_SDCLK start sample with negative clk400 edge for MEM_SDCLK phase shift 1: clk_memsig will be sampled for MEM_SDCLK-	R/W	0x1

Bits	Name	Description	R/W	Default
		generation internally first on negedge of clk400 0: clk_memsig will be sampled for MEM_SDCLK-generation internally first on posedge of clk400. Evaluation purpose only - don't use this setting!		
22:20	MEM_SDCLK_PHASE	MEM_SDCLK phase shift. 0..5: adjustable phase-shift for external SDRAM clock depending on external capacitive load on MEM_SDCLK-signal to match SDRAM signals setup times. The phase can be shifted in 1.25ns steps. MEM_SDCLK will internally rise at the mem_sdclk_phase+2nd clk400 edge after internal changes of SDRAM signals (MEM_SD*-signals, MI address and data buses driven by clk_memsig), where the 1st egde is defined by the mem_sdclk_ssneg-bit. For correct settings delays depending on external capacitive load have to be respected.	R/W	0x0
19:18	reserved	-	R	0x0
17:16	T_REFI	Average periodic refresh interval (3.90 us * 2^t_REFI 00 : 3.90 us 01 : 7.80 us (default) 10 : 15.60 us 11 : 31.20 us Note: Typically refresh of SDRAM devices is specified by a certain number of refreshes that must be performed within a certain time. E.g. 8192 refreshes for 64ms. Dividing the time by the number of refreshes leads to the average periodic refresh interval. E.g. 64ms/8192 = 7.8us. Please view also description of 'refresh_mode' of 'MEM_SDRAM_CFG_CTRL' register for details.	R/W	0x1
15:12	T_RFC	REFRESH to next command time (clk = tRFC + 4) 0000 : 4 clks 0001 : 5 clks and so on 1111 : 19 clks (default)	R/W	0xf
11	reserved	-	R	0x0
10:8	T_RAS	ACTIVE to PRECHARGE command time (clk = t_RAS + 3) 000 : 3 clks 001 : 4 clks and so on 111 : 10 clks (default)	R/W	0x7
7:6	T_RP	Precharge command period time (PRECHARGE to next command) 00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved	R/W	0x3
5:4	T_WR	Write recovery time (last write data to PRECHARGE) 00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved	R/W	0x3
3:2	reserved	-	R	0x0
1:0	T_RCD	ACTIVE to READ or WRITE time (RAS to CAS, clk = t_RCD) This value will be also taken as t_RRD (ACTIVE bank A to ACTIVE bank B time) 00 : 1 clk 01 : 2 clks 10 : 3 clks (default) 11 : reserved	R/W	0x3

MEM_SDRAM_MODE – Memory SDRAM Mode Register 0x101c0148

Changes can only be done, if the SDRAM-Controller is disabled (MEM_SDRAM_CFG_CTRL.ctrl_en == 0) to avoid configuration problems.

The SDRAM Mode Registers of the used SDRAM device will be set after enabling the SDRAM Controller in the 200us SDRAM memory initialisation procedure. It is part of the SDRAM device and programmed by the LOAD MODE REGISTER command.

For details of SDRAM Mode Register view datasheet of used SDRAM device.

Please view description of 'ctrl_en' bit inside MEM_SDRAM_CFG_CTRL register for initializing-procedure of netX SDRAM Controller.

31 … 14	13 … 0
reserved	MR

Bits	Name	Description	R/W	Default
31:14	reserved	-	R	0x0
13:0	MR	SDRAM Mode Register. CAS latency bits are typically located in MR[6:4]. Only CL2 and CL3 are supported, not CL1; default is CL3. Burst Length in MR[2:0] is read only here. Burst length depends on data bus width programmed in MEM_SDRAM_CFG_CTRL.dbus16 register bit. The netX10 controller supports only Burst Length 8 (default) for 8bit SDRAM interface and 4 for 16bit SDRAM interface. Note: SDRAM devices where burst length is not located in Mode Register bits MR[2:0] are not supported by netX SDRAM controller. However these devices are not common. Note: This bit is writable but can also be changed by hardware.	R/W	0x33

5 Dual-Port Memory

The Dual-Port Memory (DPM) interface is used for allowing data transfer between the netX and an external host system. Unlike standard DPM, the netX10 DPM is a virtual Dual-Port memory, which appears as a linear memory to the host side, while accesses to the DPM are redirected up to four different memory areas, making the whole netX functionality accessible.

The netX DPM interface can either be a parallel DPM interface based on standard SRAM access protocol extended by ready handshake mechanism or a high speed serial DPM interface based on standard Motorola SPI. Selection is done by an external mode pin. Furthermore the interface is widely configurable by configuration registers which can be set by software running on host device.

DPM interface is located on HIF-IOs for netX10. These IOs can also be used as memory interface. It is possible to use serial DPM together with 8 bit external memory on netx10. However it is neither possible using parallel DPM with external memory nor using serial DPM with 16 bit memory.

The netX10 DPM interface supports 3 different parallel modes and a high speed serial mode. Selection of either serial or parallel DPM (default is parallel) is done through register HIF_IO_CFG. For detailed information about DPM, please refer to the appropriate chapter of the “netX10 Technical Reference Guide”.

5.1 Software Interface

DPM interface parameters like parallel mode data width, data endianness, address mapping, signal timing, interrupt handling and PIO control can be accessed by the host processor.

By default, all DPM configuration registers are mapped to the first 256 bytes (address 0x00 to 0xFF) of the external DPM address range. The default bus width is 8 Bit and all basic configuration registers only use the Least Significant Byte which allows host processors with 8-Bit interface as well as pure 16-Bit (no Byte access possible) hosts to access the DPM configuration and adapt the interface to their needs (e.g bus width, RDY/BUSY signal modes, endianness, etc.) without the need for pre-configuration by firmware or second stage loader (which is still possible though).

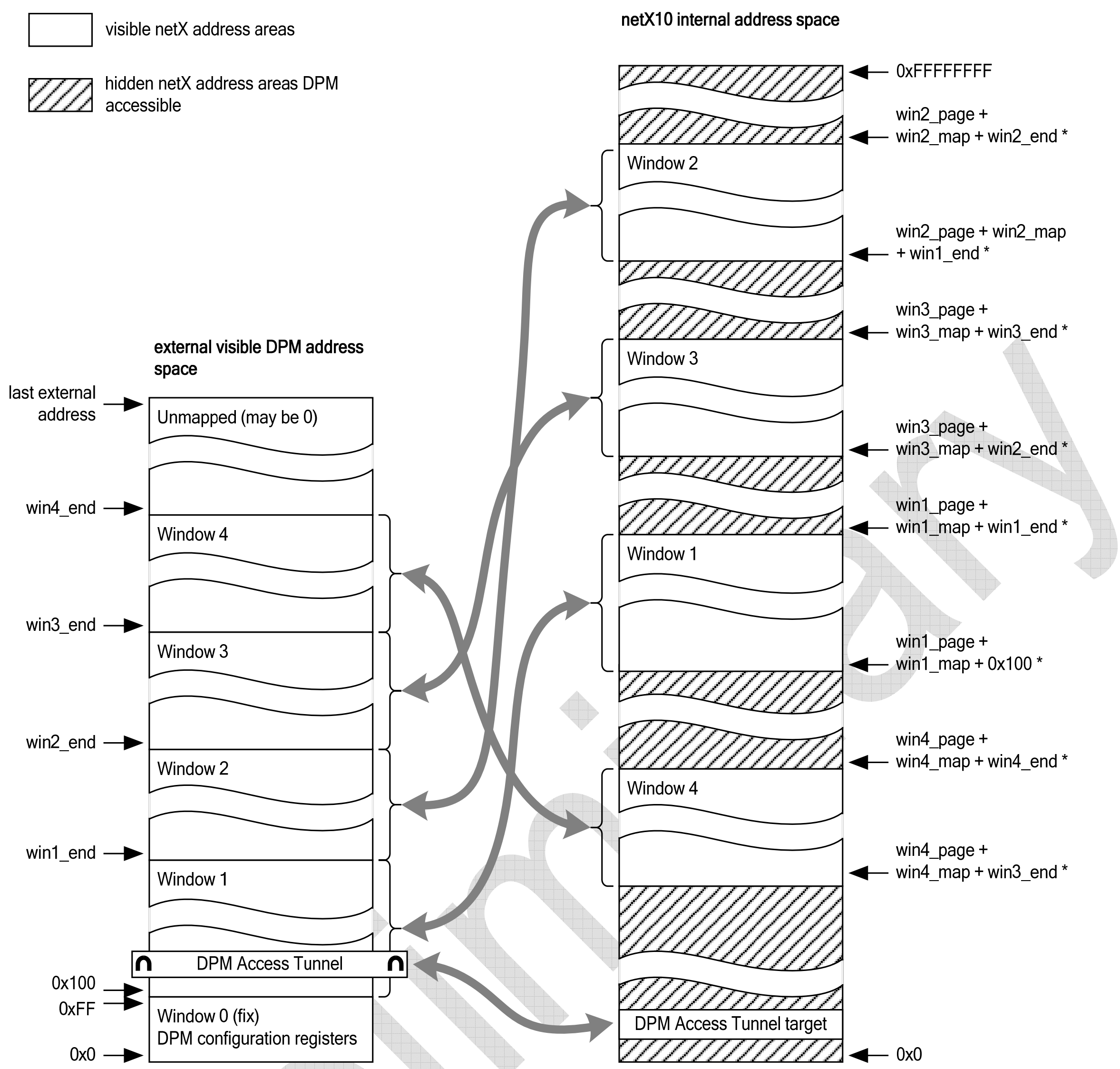
For netX50 compatibility this configuration area can be relocated to the end of the external address range (last 256 bytes) or completely switched off.

Depending on DPM mode, bus width and address range, there are almost always a number of host interface signals which are not used by the host and can be used as Programmable IOs (PIO).

5.2 DPM Window Mapping

For netX10 only DPM Configuration Window is enabled by default, all other windows are disabled. The netX10 Configuration Window is located on the lowest 256 bytes of external DPM address-space by default.

Address mapping expands external access address to an internal 32 bit wide netX address. External address range of netX10 DPM can be split off in 4 sub-regions (DPM address windows). External to internal address mapping and several features (e.g. read-ahead, byte-collecting) can be programmed for each window individually.



Following basic rules must be regarded when configuring netX DPM window mapping:

- Used (not disabled) windows must be configured address ascending:

  win_end0 < *win_end1* < *win_end2* < *win_end3* < *win_end4*

- A window can be disabled by setting the appropriate *win_end* register value to 0.
- Window mapping and size can be configured in 256 byte steps.
- First external address of window n: *win_end(n-1)*
- Last external address of window n: *win_end(n)* - 1
- Window size calculation of window n:

  win_size(n) = *win_end(n)* - *win_end(n-1)* [bytes]

- First internal address of window n:

  win_page(n) + ((*win_map(n)* + *win_end(n-1)*) & 0x000FFFFF)

- Last internal address of window n:

  win_page(n) + ((*win_map(n)* + *win_end(n)* - 1) & 0x000FFFFF)

- *win_end(0)* is fixed to 0x100

- Remapping of one window (changing appropriate *dpm_win_map* register value) without changing its size (changing appropriate *dpm_win_end* register value) does not affect the other windows.
- Changing the size of one window (changing appropriate *dpm_win_end* register value) will change location and mapping of all other windows above.

5.3 DPM Access Tunnel

DPM Access Tunnel is a small (64 bytes) special map-able DPM Window which can be laid over any normal DPM Window and also over DPM Configuration Window 0.

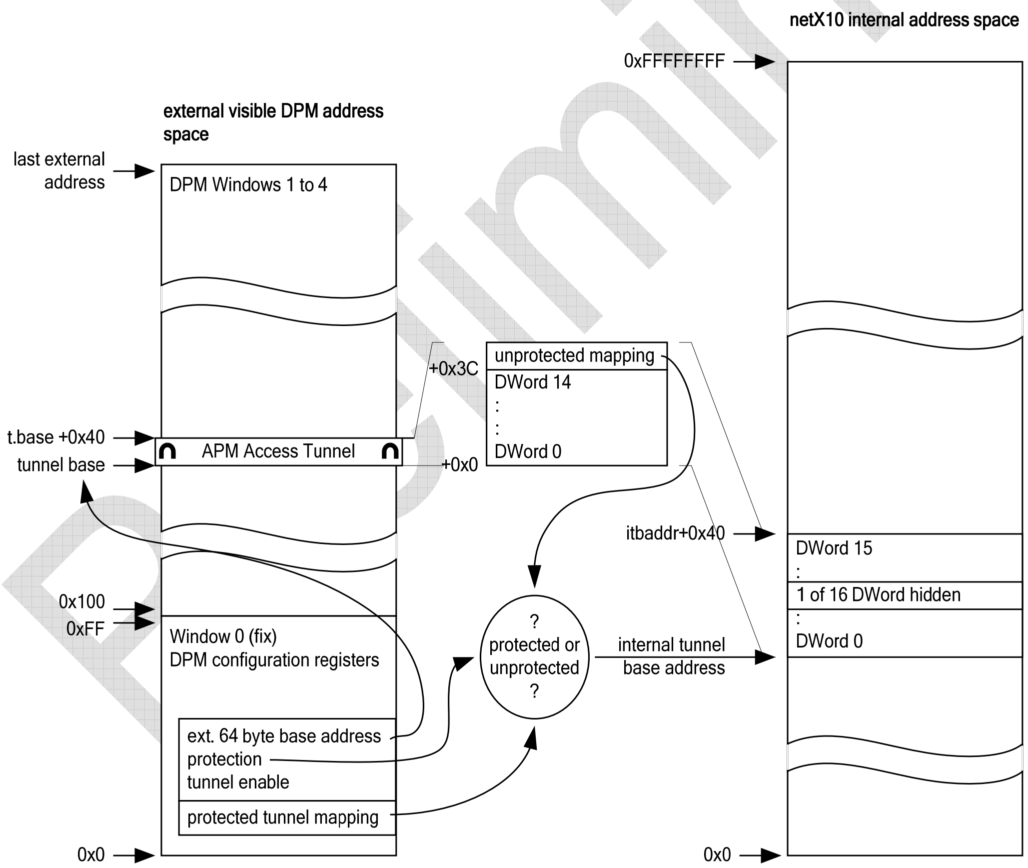
Externally the Access Tunnel base can be mapped to any available 64 byte boundary. It allows access to 15 DWords of a 64 byte target area netX internally. The last external DWord can be used to configure target area base.

There are 2 registers inside DPM Configuration Window 0 for programming DPM Access Tunnel. For further details view register description DPM_TUNNEL_CFG and DPM_ITBADDR.

Internal target area can be write-protected and protected against re-mapping by host.

Protection status can be read from Configuration DWord by host.

DPM Access Tunnel can be used for dynamic access to small netX internal address areas (e.g. for special module configuration, status or Handshake Cell access) without reconfiguring other DPM Windows.



5.4 DPM Registers Definition

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

Note: The given DPM Win0 address is the address offset inside DPM configuration window for host access. This window can be mapped to the last 256 bytes of the external DPM address space.

DPM Win0 Address	ARM Address	Register Name	Short Description
0x00000000	0x101c1200	DPM_CFG0X0	DPM IO Control Register 0
0x00000010	0x101c1210	DPM_ADDR_CFG	DPM External Address Range Configuration Register
0x00000014	0x101c1214	DPM_TIMING_CFG	DPM Timing Configuration Register
0x00000018	0x101c1218	DPM_RDY_CFG	DPM Ready (DPM_RDY) Signal Configuration Register
0x0000001c	0x101c121c	DPM_STATUS	DPM Status Register
0x00000020	0x101c1220	DPM_STATUS_ERR_RESET	DPM Error Status Reset Register
0x00000024	0x101c1224	DPM_STATUS_ERR_ADDR	DPM Error Address Status Register
0x00000028	0x101c1228	DPM_MISC_CFG	DPM Configuration Register for Some Special Functions
0x0000002c	0x101c122c	DPM_IO_CFG_MISC	DPM IO Configuration Register 0
0x00000038	0x101c1238	DPM_TUNNEL_CFG	DPM Access Tunnel Configuration Register
0x0000003c	0x101c123c	DPM_ITBADDR	DPM Access Tunnel (DATunnel) netX Internal Target Base Address (ITBAddr) Configuration Register
0x00000040	0x101c1240	DPM_WIN1_END	DPM Window 1 End Address Configuration Register
0x00000044	0x101c1244	DPM_WIN1_MAP	DPM Window 1 Address Map Configuration Register
0x00000048	0x101c1248	DPM_WIN2_END	DPM Window 2 End Address Configuration Register
0x0000004c	0x101c124c	DPM_WIN2_MAP	DPM Window 2 Address Map Configuration Register
0x00000050	0x101c1250	DPM_WIN3_END	DPM Window 3 End Address Configuration Register
0x00000054	0x101c1254	DPM_WIN3_MAP	DPM Window 3 Address Map Configuration Register
0x00000058	0x101c1258	DPM_WIN4_END	DPM Window 4 End Address Configuration Register
0x0000005c	0x101c125c	DPM_WIN4_MAP	DPM Window 4 Address Map Configuration Register
0x00000080	0x101c1280	DPM_IRQ_RAW	DPM Raw (before masking) IRQ Status Register
0x00000084	0x101c1284	DPM_IRQ_ARM_MASK_SET	DPM Interrupt Mask Register for netX Internal ARM
0x00000088	0x101c1288	DPM_IRQ_ARM_MASK_RESET	DPM Interrupt Mask Reset Register for netX Internal ARM
0x0000008c	0x101c128c	DPM_IRQ_ARM_MASKED	DPM Masked Interrupt Status Register for netX Internal ARM
0x00000090	0x101c1290	DPM_IRQ_XPIC_MASK_SET	DPM Interrupt Mask Register for netX Internal xPIC
0x00000094	0x101c1294	DPM_IRQ_XPIC_MASK_RESET	DPM Interrupt Mask Reset Register for netX Internal xPIC
0x00000098	0x101c1298	DPM_IRQ_XPIC_MASKED	DPM Masked Interrupt Status Register for netX Internal xPIC
0x0000009c	0x101c129c	DPM_IRQ_FIQ_MASK_SET	DPM Fast/SIRQ Interrupt Mask Register
0x000000a0	0x101c12a0	DPM_IRQ_FIQ_MASK_RESET	DPM Fast/SIRQ Interrupt Mask Register
0x000000a4	0x101c12a4	DPM_IRQ_FIQ_MASKED	DPM Masked Fast/SIRQ Interrupt Status Register
0x000000a8	0x101c12a8	DPM_IRQ_IRQ_MASK_SET	DPM Normal/DIRQ Interrupt Mask Register
0x000000ac	0x101c12ac	DPM_IRQ_IRQ_MASK_RESET	DPM Normal/DIRQ Interrupt Mask Register
0x000000b0	0x101c12b0	DPM_IRQ_IRQ_MASKED	DPM Masked Normal/DIRQ Interrupt Status Register
0x000000c0	0x101c12c0	DPM_HOST_WDG_HOST_TIMEOUT	Address reserved for netX50
0x000000c4	0x101c12c4	DPM_HOST_WDG_HOST_TRIG	Address reserved for netX50
0x000000c8	0x101c12c8	DPM_HOST_WDG_ARM_TIMEOUT	Address reserved for netX50
0x000000cc	0x101c12cc	DPM_SYS_STA_BIGEND16	DPM System Status Information Register in Big Endianess 16 Data Mapping
0x000000d0	0x101c12d0	DPM_HOST_TMR_CTRL	Address reserved for netX50

0x000000d4	0x101c12d4	DPM_HOST_TMR_START_VAL	Address reserved for netX50
0x000000d8	0x101c12d8	DPM_HOST_SYS_STAT	DPM System Status Information Register
0x000000dc	0x101c12dc	DPM_HOST_RESET_REQ	DPM Reset Request Register
0x000000e0	0x101c12e0	DPM_HOST_INT_STAT0	DPM Handshake Interrupt Status Register
0x000000f0	0x101c12f0	DPM_HOST_INT_EN0	DPM Handshake Interrupt Enable Register
0x000000f8	0x101c12f8	DPM_NETX_VERSION_BIGEND16	DPM netX Version Register in Big Endianess 16 Data Mapping
0x000000fc	0x101c12fc	DPM_NETX_VERSION	DPM netX Version Register

DPM_CFG0X0 – DPM IO Control Register 0 0x101c1200

This register is accessible in any DPM-mode (8, 16, 32 bit, SRAM, Intel, Motorola, little endian, big endian) by access to DPM address 0.

Basic DPM settings are configurable here to make higher addresses accessible.

31 … 6	5 4	3 2 1 0
reserved	ENDIAN	MODE

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	ENDIAN	Endianess of 32 bit (DWord) address alignment (B0: least significant byte, B3: most significant byte): coding Address A+3 A+2 A+1 A+0 00 little endian B3 B2 B1 B0 01 16 bit big endian B2 B3 B0 B1 10 32 bit big endian B0 B1 B2 B3 11 reserved Little endian is used netX inside. If big endian host device is used, set to this 01 or 10 according to host device data width.	R/W	0x0
3:0	MODE	DPM access mode: Chip-select, output-enable, write-enable and byte-enables are always low active (i.e.: nCS, nOE, nWE, nBE*, nBHE). 8 bit modes use D[7:0], 16 bit modes use D[15:0]. For DPM modes with less than 32 bit data, write data may not written immediate to netX memory or registers (view dpm_win1_map register). Internal access size decision depends on external DPM data size: Supported DPM modes are: 0000: 8 bit SRAM (write data/address valid at positive edge nWR). DPM_D0..7 are used as data lines, DPM_D8..31 can be used as PIOs (+24 PIOs). : reserved. 0100: 16 bit SRAM (address 16 bit aligned, 2 byte enables) or Intel (byte address with byte-high enable) mode. : reserved. 0110: 16 bit TI OMAP address/data multiplexed mode. : reserved. 1111: reserved.	R/W	0x0

DPM_ADDR_CFG – DPM External Address Range Configuration Register 0x101c1210

Unused address lines can be used as PIOs.

31 … 6	5 … 4	3 … 0
reserved	CFG_WIN_ADDR_CFG	ADDR_RANGE

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	CFG_WIN_ADDR_CFG	Configuration of External DPM Configuration Window 0. Supported settings are: 00: Configuration Window is located in the first 256 bytes of external DPM address range (0x0 to 0xff). It is located before the next enabled Window (1 to 4). 01: Configuration Window is located in the last 256 bytes of external DPM address range. It is located after the last enabled Window (1 to 4). Example: 'addr_range' is 8kB: Configuration Window is located in 0x1F00..0x1FFF. 10: reserved. 11: Configuration Window is disabled for external DPM access. Full DPM address range can be used for Windows 1 to 4. Note: Configuration Window 0 access detection has higher priority than normal DPM Window detection but lower priority than Access Tunnel access detection.	R/W	0x0
3:0	ADDR_RANGE	DPM external address range. coding function / address used signals / signals for PIO usage 0000 reserved 0001 reserved 0010 2KB address range / DPM_A[10:0] / +9 PIOs: DPM_A[19:11] 0011 reserved 0100 8KB address range / DPM_A[12:0] / +7 PIOs: DPM_A[19:13] 0101 16KB address range / DPM_A[13:0] / +6 PIOs: DPM_A[19:14] 0110 32KB address range / DPM_A[14:0] / +5 PIOs: DPM_A[19:15] 0111 64KB address range / DPM_A[15:0] / +4 PIOs: DPM_A[19:16] 1000 128KB address range / DPM_A[16:0] / +3 PIOs: DPM_A[19:17] : reserved 1111 reserved	R/W	0x2

DPM_TIMING_CFG – DPM Timing Configuration Register 0x101c1214

31 … 8	7	6 … 4	3	2	1 … 0
reserved	RD_BURST_EN	T_RDS	reserved	FILTER	T_OSA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7	RD_BURST_EN	Read burst enable.	R/W	0x0
6:4	T_RDS	Read data setup time (in steps of 10ns). If DPM_RDY is used (rdy_mode != 0), DPM_RDY is generated t_rds*10ns after read data is stored on data bus. Without DPM_RDY use (rdy_mode == 0) read access error is detected if access terminates before t_rds*10ns passed after read data generation. Valid settings are: 0..7. Note: Read data access time will increased by t_rds * 10ns if t_rds is not 0.	R/W	0x2
3	reserved	-	R	0x0
2	FILTER	Filter DPM Control Signals. If this bit is set, DPM signals Chip-Select, Read-Enable and Write-Enable (and Address latch enable if multiplexed Parallel DPM modes are used) are filtered for spike suppression. 0: no spike suppression. 1: Spikes < 10ns are suppressed, read data access time increased by 10ns. Note: Data, address and byte-enable inputs are not filtered and must be stable when sampled. I.e. during the last 10ns of a read access and at the first 10ns of read access start. Note: Read data access time is increased by 10ns if this bit is set.	R/W	0x1
1:0	T_OSA	Address Setup Time (t_osa * 10ns). Address sampling can be delayed for read and write accesses by this parameter. E.g. host device asserts Chip-Select, Read-Enable and address lines simultaneously but some address lines are not stable while Chip-Select and Read-Enable are both low, set t_osa to delay address sampling by t_osa * 10ns. Valid settings are: 0..3. Note: Read data access time will increased by t_osa * 10ns if t_osa is not 0.	R/W	0x3

DPM_RDY_CFG – DPM Ready (DPM_RDY) Signal Configuration Register 0x101c1218

31 … 4	3	2 1	0
reserved	RDY_SIG_MODE	RDY_DRV_MODE	RDY_POL

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	RDY_SIG_MODE	Ready signal mode. 1: DPM_RDY is generated as ready/acknowledge pulse. In this mode, DPM_RDY is only in active state at access end to sign that host device is allowed to finish the current access. If no access to DPM is done or if host device runs DPM access but is not allowed to finish it yet, DPM_RDY will remain in inactive state. 0: DPM_RDY is generated as wait/busy state signal. In this mode, DPM_RDY becomes active at access start and will remain active while host device is not allowed to finish the current access. If no access to DPM is done or if host device runs DPM access and allowed to finish it and continue access generation, DPM_RDY will be in inactive state.	R/W	0x0
2:1	RDY_DRV_MODE	Ready generation mode. 00: ready signal generation is disabled (High-Impedance mode). 01: ready is driven when active and inactive. Never highZ. (Push-Pull mode) 10: ready is driven when active and for a short time when inactive-phase starts for fast busy to ready signal state change (Sustain-Tristate mode). Inactive-phase ready driving time (tRPm02, tRPm12) depends on rdy_sig_mode: For rdy_sig_mode=0 this time (tRPm02) is 10ns. For rdy_sig_mode=1 this time (tRPm12) depends on programmed input signal filtering (register dpm_timing_cfg bit filter): If filtering is disabled tRPm12 is 20ns to 30ns, if input filtering is enabled, tRPm12 is 30ns to 40ns. 11: ready is only driven when cycle active (Open-Drain/Open-Source mode). Note: Mode 2 and 3 are reordered in comparison to netX100/500/50.	R/W	0x0
0	RDY_POL	Ready signal active state polarity. 1: DPM is ready when external RDY-signal is high. 0: DPM is busy when external RDY-signal is high.	R/W	0x1

DPM_STATUS – DPM Status Register 0x101c121c

DPM access errors can generate IRQ for host device (view DPM IRQ registers further down). For error handling, the address an error occurred with is logged in DPM_STATUS_ERR_ADDR register. Error bits can be cleared by access to DPM_STATUS_ERR_RESET register.

31 … 8	7	6	5	4	3	2	1	0
reserved	SEL_DPM_SERIAL	BUS_CONFLICT_RD_ADDR_ERR	BUS_CONFLICT_RD_ERR	BUS_CONFLICT_WR_ERR	RDY_TO_ERR	WR_ERR	RD_ERR	UNLOCKED

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7	SEL_DPM_SERIAL	DPM_MODE configuration input state. 0: DPM is in parallel mode (DPM_MODE configuration input is low). 1: DPM is in serial mode (DPM_MODE configuration input is high). If DPM is in serial mode, mode configuration pins DPM_DQM0N/1N can be read from dpm_pio_in1-register.	R	0x0
6	BUS_CONFLICT_RD_AD DR_ERR	Parallel DPM read access address change bus error detected. This bit is set if address lines change (low, after filtering if enabled) during a read access while burst support is not enabled. In this case the current read access will be terminated. Read data will not be valid then. Note: Input filtering and burst support is configured by dpm_timing_cfg register. Note: If this bit is set, there are hazards on these IOs or host device generates invalid DPM access states. This bit is used for dpm_err-IRQ generation. Ready signal will be set to ready state (if ready usage is enabled) on access errors.	R	0x0
5	BUS_CONFLICT_RD_ER R	Parallel DPM read access bus error detected. This bit is set if write-control (nWR) signal becomes active (low, after filtering if enabled) during a read access. In this case the current read access will be terminated to avoid potential IO driving conflicts. Read data will not be valid then. Note: Input filtering is configured by dpm_timing_cfg register.. Note: If this bit is set, there are hazards on these IOs or host device generates invalid DPM access states. This bit is used for dpm_err-IRQ generation. Ready signal will be set to ready state (if ready usage is enabled) on access errors.	R	0x0
4	BUS_CONFLICT_WR_E RR	Parallel DPM write access bus error detected. This bit is set if read-control (nRD) signal becomes active (low, after filtering if enabled) during a write access. In this case the current write access will be ignored and write data is junked. Note: Input filtering is configured by dpm_timing_cfg register.. Note: If this bit is set, there are hazards on these IOs or host device generates invalid DPM access states. This bit is used for dpm_err-IRQ generation. Ready signal will be set to ready state (if ready usage is	R	0x0

Bits	Name	Description	R/W	Default
		enabled) on access errors.		
3	RDY_TO_ERR	DPM_RDY Timeout Error Status Flag. This error may occur if host device tries to access permanently busy netX address area (e.g. netX xPEC program RAM while xPEC is running). To avoid host device stalling DPM_RDY signal is released to ready state after 2048 system clock cycles (i.e. 20.48us) at least. This bit must be reset by access to the dpm_status_err_reset register. 1: Last access went to netX busy address and was broken to avoid host device stalling. 0: Access was finished successfully by DPM_RDY assertion to ready state. This bit is used for dpm_err-IRQ generation.	R	0x0
2	WR_ERR	DPM Write Error Status Flag. Write errors may occur if ready signal (DPM_RDY) is not respected by an external host device and external DPM write access terminated before data could be stored. In some cases certain netX address areas may be busy for not predictable time. If DPM_RDY is not used, check for write error after write access to these areas. In case of write error this bit is set immediately after the appropriate write access. Repeat the write access until no error occurs. If the error occurs permanently stretch external DPM access by inserting wait states. This bit must be reset by access to the dpm_status_err_reset register. 1: The external DPM write access was too fast to store write data. Repeat the write access. 0: Write access terminated without error. This bit is used for dpm_err-IRQ generation.	R	0x0
1	RD_ERR	DPM Read Error Status Flag. Read errors may occur if ready signal (DPM_RDY) is not respected by an external host device and external DPM read access terminated before read data could be stored on the external DPM data bus (view also t_rds in dpm_timing_cfg register). In some cases certain netX address areas may be busy for not predictable time. If DPM_RDY is not used, check for read error after read access to these areas. In case of read error this bit is set immediately after the appropriate read access. Repeat the read access until no error occurs. If the error occurs permanently stretch external DPM access by inserting wait states. This bit must be reset by access to the dpm_status_err_reset register. 1: The external DPM read access was too fast. Repeat the read access. 0: Read data OK. This bit is used for dpm_err-IRQ generation. These bits must be reset before further access to netX internal address area. If wr_err or rd_err is set, all further access to netX internal address area will be ignored. Note: These bits may also be set if external signals are hazardous. Example: If nRD signal is not stable at access start (after t_osa passed) netX may see more than 1 read access. As result, rd_err may be flagged then. Solution: increment t_osa, avoid hazards.	R	0x0
0	UNLOCKED	DPM is locked during netX5 power up and boot phase. DPM access to other addresses than these DPM control address area cannot be done before this bit is set to 1. Poll for 1 after power up or reset.	R	0x0

DPM_STATUS_ERR_RESET – DPM Error Status Reset Register 0x101c1220

Each flags can be reset by writing a ’1’ to it. For fast error detection for DPM interfaces without ready usage, reset-on-read-function can be enabled for this register.

Note:
If reset-on-read-function is enabled, this register must be read with a single access as bits are cleared immediately after the access. You should always use a byte access in this case.

Note:
View DPM_STATUS register for detailed error description.

Note:
reset-on-read-function is controlled by enable_flag_reset_on_rd-bit in DPM_MISC_CFG-register.

31 … 7	6	5	4	3	2	1	0
reserved	BUS_CONFLICT_RD_ADDR_ERR_RST	BUS_CONFLICT_RD_ERR_RST	BUS_CONFLICT_WR_ERR_RST	RDY_TO_ERR_RST	WR_ERR_RST	RD_ERR_RST	reserved

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x0
6	BUS_CONFLICT_RD_ADDR_ERR_RST	Parallel DPM read access address change bus error detected.	R/W	0x0
5	BUS_CONFLICT_RD_ERR_RST	Parallel DPM read access bus error detected.	R/W	0x0
4	BUS_CONFLICT_WR_ERR_RST	Parallel DPM write access bus error detected.	R/W	0x0
3	RDY_TO_ERR_RST	DPM_RDY timeout error.	R/W	0x0
2	WR_ERR_RST	DPM write error detection bit with auto reset function. For fast read error detection this bit can be checked after each read access. If it was set, the read access must be repeated. Note: In cases where internal access time is not predictable and host provides no ready function, it is recommended to enable reset-on-read-function. There is only one access necessary for error detection and clearing this flag then.	R/W	0x0
1	RD_ERR_RST	DPM read error detection bit with auto reset function. For fast write error detection this bit can be checked after each write access. If it was set, the write access must be repeated. Note: In cases where internal access time is not predictable and host provides no ready function, it is recommended to enable reset-on-read-function. There is only one access necessary for error detection and clearing this flag then. View dpm_status register for detailed description.	R/W	0x0
0	reserved	-	R	0x0

DPM_STATUS_ERR_ADDR – DPM Error Address Status Register 0x101c1224

31:17	16:0
reserved	ERR_ADDR

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16:0	ERR_ADDR	Access error address. Address of first erroneous access. IRQ handler can use this value to repeat failed accesses after error bits are set in dpm_status or dpm_status_err_reset register. However, only DPM Read Error (rd_err), DPM Write Error (wr_err) and DPM_RDY Timeout Error (rdy_to_err) are cared for address logging. This register is only valid if one of the error bits is set and should be read before error bits are cleared. If no error bit is set, it is updated each access to the current address. Note: Address status during bus conflict errors will not be logged. Bus conflict error status information is for debug purpose of unstable systems. Purpose of this register is primarily access error handling for systems without ready usage.	R	0x0

DPM_MISC_CFG – DPM Configuration Register for Some Special Functions 0x101c1228

31:1	0
reserved	ENABLE_FLAG_RESET_ON_RD

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	ENABLE_FLAG_RESET_ON_RD	Enable Status Flag Reset by Reading this register. If enable_flag_reset_on_rd-bit is 0, there is only one access necessary for error detection and clearing the error status bits. In cases where internal access time is not predictable and host provides no ready function, it is recommended to enable reset-on-read-function to minimize traffic.	R/W	0x0

DPM_IO_CFG_MISC – DPM IO Configuration Register 0 0x101c122c

Unused DPM IOs can be used as PIOs.

31 … 8	7	6	5	4	3	2	1	0
reserved	FIQ_OEC	FIQ_POL	IRQ_OEC	IRQ_POL	reserved	SEL_SIRQ_PIO	SEL_DIRQ_PIO	SEL_RDY_PIO

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7	FIQ_OEC	FIQ/SIRQ output enable controlled. 0: FIQ/SIRQ signal is always driven. 1: FIQ/SIRQ signal is only driven when active. Inactive level must be realized by external pull-up or pull-down resistor.	R/W	0x1
6	FIQ_POL	FIQ/SIRQ signal polarity. 0: FIQ/SIRQ is active low. 1: FIQ/SIRQ is active high.	R/W	0x0
5	IRQ_OEC	IRQ output enable controlled. 0: IRQ/DIRQ signal is always driven. 1: IRQ/DIRQ signal is only driven when active. Inactive level must be realized by external pull-up or pull-down resistor.	R/W	0x1
4	IRQ_POL	IRQ/DIRQ signal polarity. 0: IRQ/DIRQ is active low. 1: IRQ/DIRQ is active high.	R/W	0x0
3	reserved	-	R	0x0
2	SEL_SIRQ_PIO	Use DPM_SIRQ-pin as PIO pin.	R/W	0x1
1	SEL_DIRQ_PIO	Use DPM_DIRQ-pin as PIO pin.	R/W	0x1
0	SEL_RDY_PIO	Use DPM_RDY-pin as PIO pin. RDY is by default PIO to avoid RDY-conflicts during reset.	R/W	0x1

DPM_TUNNEL_CFG – DPM Access Tunnel Configuration Register 0x101c1238

DPM Access Tunnel (DATunnel) is a 64 byte (16DWord) address window which can be mapped on any 64 byte boundary of the external visible address space.

In the last DWord (15) of DATunnel a netX Internal 32 bit Target Base Address (ITBAddr) matching a 64 byte boundary can be programmed.

By DWords 0..14 netX data starting at ITBAddr can be accessed then (read-only functionality can be configured by 'wp_data' bit).

For access to netX data with ITBAddr DWord offset 15, bits 5 to 2 of programmed ITBAddr are interpreted as mapping value. This value will be added to internal access address before tunnelling (wrapping around at the 64 byte boundary). Hence it is possible to access always 15 of the 16 netX DWord while the missing one can be selected by an appropriate mapping value.

ITBAddr can also be programmed or read from netX using DPM_ITBADDR register. Also ITBAddr can be write-protected from host by a configuration bit (wp_itbaddr) of this register.

External to internal address mapping for DATunnel area can be calculated by following formula:

INAAdr = (ITBAddr & 0xffffffc0) + ((EDAAdr + ITBAddr) & 0x3C)

With:

- INAAdr: Internal netX Access Address
- ITBAddr: Internal netX 32 bit Tunnel Target Base Address
- EDAAdr: External DPM Access Address

Condition for DATunnel access is:

EDAAdr>>6 equals value of bit field 'base' from this register.

To map netX internal DWord N to invisible last external DWord (15), use mapping value

map = (N - 15) & 0xf on bits 5 to 2.

Internal to external address offset inside DATunnel area for internal DWord N can be calculated by following formula:

External offset = (N*4 - map*4) & 0x3C = (N*4 - ITBAddr) & 0x3C

Example 1:

Access to netX sys_time module by host via DATunnel on external DPM addresses are starting at 0x240.

- Set bit field 'base' of this register to 9 (0x240>>6), set enable bit (and write protection depending on application).

DATunnel now is enabled on external DPM addresses 0x240 to 0x27f.

- ITBAddr of netX10 sys_time module is 0x101c1000.

For direct DATunnel to this address, host must write 0x101c1000 to external DPM address 0x27c. This can be done e.g. by four byte accesses to 0x27c, 0x27d, 0x27e and 0x27f or by two 16 bit accesses to 0x27c and 0x27e.

Now sys_time module registers 0 to 14 can be accessed on external DPM address 0x240 to 0x27b.

Example 2:

Register 15 of sys_time is hidden by ITBAddr configuration on 0x27c in example 1 but must also be accessed. However, sys_time Register 6 is never kind of interest.

- Configure this register like described in example 1.
- To map Register 6 (Module offset 6*4) to external offset 0x3C (hidden data on DWord 15), the following rule must be complied:

  0x3C + map*4 = 6*4.

  That leads to a mapping value of:

  map*4 = (6*4 - 0x3C) & 0x3C = 1C

  Hence, write 0x101c101C to DATunnel DWord 15 (external DPM address 0x27c) to map sys_time Register 6 to hidden DWord 15.

  INAAdr now will be derived from EDAAdr before tunneling as follows:

  INAAdr = 0x101c1000 + ((EDAAdr + 0x1C) & 0x3C)

  External offset of Module DWord N results from:

  External offset = (N*4 - 0x1C) & 0x3C

  Register 15 of sys_time unit now can be accessed by external DPM address 0x240+((0xf*4-0x1C) & 0x3C) = 0x260 (i.e. Tunnel DWord 8).

  Register 0 of sys_time unit now can be accessed by external DPM address 0x240+((0x0*4-0x1C) & 0x3C) = 0x264 (i.e. Tunnel DWord 9).

  Register 1 of sys_time unit now can be accessed by external DPM address 0x240+((0x1*4-0x1C) & 0x3C) = 0x268 (i.e. Tunnel DWord 10).
  and so on.

  Register 6 of sys_time unit can not be accessed as it is hidden by ITBAddr configuration on 0x27c (i.e. Tunnel DWord 15).

  Register 7 of sys_time unit now can be accessed by external DPM address 0x240+((0x7*4-0x1C) & 0x3C) = 0x240 (i.e. Tunnel DWord 0).

Note:

Access to netX ITBAddr data is done without read ahead and with byte collecting (view DPM_WIN1_MAP for details).

Note:

Configuration Window 0 access detection has higher priority than normal DPM Window detection but lower priority than Access Tunnel access detection.

31 … 17	16 … 6	5 … 3	2	1	0
reserved	BASE	reserved	ENABLE	WP_ITBADDR	WP_DATA

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16:6	BASE	DPM Access Tunnel (DATunnel) Base Address divided by 64 on external visible address space. Note: Default setting for tunnel base is starting on external address 0x100.	R/W	0x4
5:3	reserved	-	R	0x0
2	ENABLE	Enable/disable Access Tunnel function.	R/W	0x0
1	WP_ITBADDR	ITBAddr is write-protected from host. If this bit is set, ITBAddr (Internal netX 32 bit Tunnel Target Base Address) can only be changed from netX side using dpm_itbaddr address. Write accesses to DWords 0 to 14 of DATunnel will be ignored.	R/W	0x0
0	WP_DATA	Access Tunnel function is write-protected from data access (DWords 0 to 14 of DATunnel). Write accesses to DWords 0 to 14 of DATunnel will be ignored. Data write protection for host is enabled by default and can be disabled by clearing this bit.	R/W	0x1

DPM_ITBADDR – DPM Access Tunnel (DATunnel) netX Internal Target Base Address (ITBAddr) Configuration Register 0x101c123c

For DPM Access Tunnel (DATunnel) function view description of DPM_TUNNEL_CFG register.

This register contains ITBAddr value that can also be changed by host on last offset 0x3c (last DWord) of external DATunnel area (defined by bit field 'base' in 'DPM_TUNNEL_CFG' register). However this register can also be write-protected from host if bit 'wp_itbaddr' in 'DPM_TUNNEL_CFG' register is set.

Write protection bits of DATunnel configured in 'DPM_TUNNEL_CFG' register can also be read from this register. Host can read access rights from these bits on last DWord of external DATunnel address area.

31 … 6	5 … 2	1	0
BASE	MAP	WP_ITBADDR_RO	WP_DATA_RO

Bits	Name	Description	R/W	Default
31:6	BASE	Internal netX Tunnel Target Base Address (ITBAddr) divided by 64. View description of dpm_tunnel_cfg register.	R/W	0x0
5:2	MAP	Mapping part of ITBAddr. View description of dpm_tunnel_cfg register.	R/W	0x0
1	WP_ITBADDR_RO	ITBAddr is write-protected from host. This is a read-only bit here. Its setting can be changed in 'dpm_tunnel_cfg' register. View description of dpm_tunnel_cfg register.	R/W	0x0
0	WP_DATA_RO	Access Tunnel function is write-protected from data access (DWords 0 to 14 of DATunnel). This is a read-only bit here. Its setting can be changed in 'dpm_tunnel_cfg' register. View description of dpm_tunnel_cfg register.	R/W	0x1

DPM_WIN1_END – DPM Window 1 End Address Configuration Register 0x101c1240
DPM_WIN2_END – DPM Window 2 End Address Configuration Register 0x101c1248
DPM_WIN3_END – DPM Window 3 End Address Configuration Register 0x101c1250
DPM_WIN4_END – DPM Window 4 End Address Configuration Register 0x101c1258

Smallest DPM window configuration unit is 128 bytes for netX10 (i.e. lowest 7 bits of address configuration are always 0).

At address 0x0 DPM configuration window is mapped after reset (length: 256 bytes, containing all DPM addresses defined here). Each window starts at window end address of the preceding window. Hence external window 1 start address is 0x100, window 2 starts at value programmed in this register and so on.

Windows with programmed end addresses exceeding external address range (view DPM_ADDR_CFG) can not be accessed by host device.

There are 4 programmable DPM windows in netX10.

Note:
Configuration Window 0 access detection has higher priority than normal DPM Window detection but lower priority than Access Tunnel access detection.

31 … 18	17 … 7	6 … 0
reserved	WIN_END	reserved

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x0
17:7	WIN_END	Window n End Address divided by 128. Last external address is win_end*128-1. Setting win_end to 0 will disable this window. If programmed external address range (DPM_ADDR_CFG) is smaller than maximum external address range, access addresses will be zero-expanded for upper unused address lines before window match detection.	R/W	0x0
6:0	reserved	-	R	0x0

DPM_WIN1_MAP – DPM Window 1 Address Map Configuration Register 0x101c1244
DPM_WIN2_MAP – DPM Window 2 Address Map Configuration Register 0x101c124c
DPM_WIN3_MAP – DPM Window 3 Address Map Configuration Register 0x101c1254
DPM_WIN4_MAP – DPM Window 4 Address Map Configuration Register 0x101c125c

Smallest DPM window configuration unit is 128 bytes for netX10 (i.e. lowest 8 bits of address configuration are always 0).

There are 4 further programmable DPM windows in netX5.

31 … 20	19 … 7	6 5 4	3 2	1	0
WIN_PAGE	WIN_MAP	reserved	WIN_MAP_ALT	READ_AHEAD	BYTE_AREA

Bits	Name	Description	R/W	Default
31:20	WIN_PAGE	Window n address page. Internal address space of netX10 is divided in 1MB pages. Changing win_map allows addressing inside the whole currently set page. Example: Window n starts at 0x400 of external DPM address range (i.e. programmed win_end value of window (n-1) and targets netX address 0x01808000. The programmed value for the related page is 0x018.	R/W	0x18
19:7	WIN_MAP	Window n Address Mapping. Internal access address HADDR to netX10 logic is combined by DPM interface by: HADDR[31:20]: win_page HADDR[19:0]: mapped DPM address. This part of address is defined by programmed win_map value for each window. The value to be programmed is address bits 19 to 0 of netX internal window start address minus start address of the external window (i.e. end address of preceding window) . Example: Window n starts at 0x400 of external DPM address range (i.e. programmed win_end value of window (n-1) and targets netX address 0x01808000. For address calculation only lower 20 bits of netX address are relevant, i.e. 0x08000. The complete 20 bit address map value is then:0x08000-0x400=0x07C00. Hence the programmed 13 bit value must be 0x07C00>>7=0xf8.	R/W	0x0
6:4	reserved	-	R	0x0
3:2	WIN_MAP_ALT	Window n Alternative Address Mapping Configuration. Alternative Address Mapping can be generated by Triple Buffer Managers inside HANDSHAKE_CTRL unit. Coding: 00 : Alternative Address Mapping disabled. 01 : Alternative Address Mapping enabled: Use Triple Buffer Manager 0 from HANDSHAKE_CTRL unit. 10 : Alternative Address Mapping enabled: Use Triple Buffer Manager 1 from HANDSHAKE_CTRL unit. 11 : reserved If Alternative Address Mapping is enabled, mapping value is taken according to buffer status of related HANDSHAKE_CTRL Triple Buffer Manager as follows. buffer status — used mapping value 00 (buffer 0) — win_map entry of this register 01 (buffer 1) — Alternative win_map value 1 of related HANDSHAKE_CTRL Triple Buffer Manager. 10 (buffer 2) — Alternative win_map value 2 of related HANDSHAKE_CTRL Triple Buffer Manager. 11 (invalid buffer) — win_map entry of this register Note:	R/W	0x0

Bits	Name	Description	R/W	Default
		Alternative Triple Buffer Manager win_map values can be programmed in HANDSHAKE_CTRL address area.		
1	READ_AHEAD	Read ahead. If this bit is set, read ahead will be done. This will minimize read cycle time if ready generation is used but may cause problems with read sensitive logic (e.g. FIFOs).	R/W	0x0
0	BYTE_AREA	Window is byte area. 1: Memory area of this window is byte accessible. All sub DWord write accesses are done immediately. 0: Memory area of this window is 32 bit accessible. All sub DWord write accesses are collected to DWords (byte-collecting). Internal write access is done when all bytes of a DWord are written. Write address is defined by last access only. Setting of this bit does not affect read functionality: Read data form netX logic is always buffered inside DPM interface. Read buffer is updated (new read form netX logic) when read access targets another 32 bit address boundary than prior access or if read access targets the same data within the 32 bit address boundary of prior accesses (repeated read, polling).	R/W	0x0

DPM_IRQ_RAW – DPM Raw (before masking) IRQ Status Register 0x101c1280

If bit is set, the according interrupt is asserted.

Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Test-IRQ state is always 1 if appropriate IRQ/DIRQ or FIQ/SIRQ mask is set. If no mask is set, state is 0. Enable one of these mask bits to test IRQ/FIQ function.	R	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU. Handshake Events and netX Firmware System Status can be flagged to host by this IRQ. Firmware IRQ generation can be controlled by dpm_firmware_irq_mask register. Firmware IRQ status can be read from dpm_firmware_irq_raw register.	R	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0

DPM_IRQ_ARM_MASK_SET – DPM Interrupt Mask Register for netX Internal ARM 0x101c1284

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

If bit is set, the according interrupt will activate the IRQ for netX internal ARM.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal ARM without clearing interrupt in module, reset according mask bit to 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
TEST	reserved																		FIRMWARE	DPM_ERR	reserved						MSYNC0	reserved			COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_ARM_MASK_RESET – DPM Interrupt Mask Reset Register for netX Internal ARM 0x101c1288

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

If bit is set, the according interrupt will activate the IRQ for netX internal ARM if asserted.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal ARM without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_ARM_MASKED – DPM Masked Interrupt Status Register for netX Internal ARM0x101c128c

Bit is set, if the according mask bit is set in DPM_IRQ_ARM_MASK-register and the according interrupt is asserted.
IRQ for netX internal ARM signal is asserted if at least one bit is set here.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal ARM signal without clearing interrupt in module, reset according mask bit to 0.

31	30–13	12	11	10–5	4	3–1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Test-IRQ state is always 1 if appropriate IRQ/DIRQ or FIQ/SIRQ mask is set. If no mast is set, state is 0. Enable according mask bit to test FIQ/SIRQ function.	R	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0

DPM_IRQ_XPIC_MASK_SET – DPM Interrupt Mask Register for netX Internal xPIC 0x101c1290

Write access with ’1’ sets interrupt mask bit (enables interrupt request for corresponding interrupt source).
Write access with ’0’ does not influence this bit.
Read access shows actual interrupt mask.
If bit is set, the according interrupt will activate the IRQ for netX internal xPIC.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal xPIC without clearing interrupt in module, reset according mask bit to 0.

31	30–13	12	11	10–5	4	3–1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_XPIC_MASK_RESET – DPM Interrupt Mask Reset Register for netX Internal xPIC 0x101c1294

Write access with '1' resets interrupt mask bit (disables interrupt request for corresponding interrupt source).
Write access with '0' does not influence this bit.
Read access shows actual interrupt mask.
If bit is set, the according interrupt will activate the IRQ for netX internal xPIC if asserted.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal xPIC without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_XPIC_MASKED – DPM Masked Interrupt Status Register for netX Internal xPIC0x101c1298

Bit is set, if the according mask bit is set in DPM_IRQ_XPIC_MASK-register and the according interrupt is asserted.
IRQ for netX internal xPIC signal is asserted if at least one bit is set here.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ for netX internal xPIC signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Test-IRQ state is always 1 if appropriate IRQ/DIRQ or FIQ/SIRQ mask is set. If no mast is set, state is 0. Enable according mask bit to test FIQ/SIRQ function.	R	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0

DPM_IRQ_FIQ_MASK_SET – DPM Fast/SIRQ Interrupt Mask Register 0x101c129c

Write access with '1' sets interrupt mask bit (enables interrupt request for corresponding interrupt source).
Write access with '0' does not influence this bit.
Read access shows actual interrupt mask.
If bit is set, the according interrupt will activate the FIQ/SIRQ signal if asserted.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release FIQ/SIRQ signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_FIQ_MASK_RESET – DPM Fast/SIRQ Interrupt Mask Register 0x101c12a0

Write access with '1' resets interrupt mask bit (disables interrupt request for corresponding interrupt source).
Write access with '0' does not influence this bit.
Read access shows actual interrupt mask.
If bit is set, the according interrupt will activate the FIQ/SIRQ signal if asserted.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release FIQ/SIRQ signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test FIQ/SIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_FIQ_MASKED – DPM Masked Fast/SIRQ Interrupt Status Register 0x101c12a4

Bit is set, if the according mask bit is set in DPM_IRQ_FIQ_MASK-register and the according interrupt is asserted.
FIQ/SIRQ signal is asserted if at least one bit is set here.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release FIQ/SIRQ signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Test-IRQ state is always 1 if appropriate IRQ/DIRQ or FIQ/SIRQ mask is set. If no mast is set, state is 0. Enable according mask bit to test FIQ/SIRQ function.	R	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0

DPM_IRQ_IRQ_MASK_SET – DPM Normal/DIRQ Interrupt Mask Register 0x101c12a8

Write access with ’1’ sets interrupt mask bit (enables interrupt request for corresponding interrupt source).
Write access with ’0’ does not influence this bit.
Read access shows actual interrupt mask.
If bit is set, the according interrupt will activate the IRQ/DIRQ signal if asserted.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ/DIRQ signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test IRQ/DIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_IRQ_MASK_RESET – DPM Normal/DIRQ Interrupt Mask Register 0x101c12ac

If bit is set, the according interrupt will activate the IRQ/DIRQ signal if asserted.

Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ/DIRQ signal without clearing interrupt in module, reset according mask bit to 0.

31	30 … 13	12	11	10 … 5	4	3 … 1	0
TEST	reserved	FIRMWARE	DPM_ERR	reserved	MSYNC0	reserved	COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Enable this bit to test IRQ/DIRQ function.	R/W	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R/W	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R/W	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0

DPM_IRQ_IRQ_MASKED – DPM Masked Normal/DIRQ Interrupt Status Register 0x101c12b0

Bit is set, if the according mask bit is set in DPM_IRQ_IRQ_MASK-register and the according interrupt is asserted.

IRQ/DIRQ signal is asserted if at least one bit is set here.
Interrupts must be reset in interrupt generating module. Interrupts cannot be cleared here.
To release IRQ/DIRQ signal without clearing interrupt in module, reset according mask bit to 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
TEST	reserved																		FIRMWARE	DPM_ERR	reserved						MSYNC0	reserved			COM0

Bits	Name	Description	R/W	Default
31	TEST	Test bit for interrupt test. Test-IRQ state is always 1 if appropriate IRQ/DIRQ or FIQ/SIRQ mask is set. If no mast is set, state is 0. Enable according mask bit to test IRQ/DIRQ function.	R	0x0
30:13	reserved	-	R	0x0
12	FIRMWARE	Firmware IRQ for host CPU.	R	0x0
11	DPM_ERR	DPM access error. Check error bits in dpm_status register.	R	0x0
10:5	reserved	-	R	0x0
4	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
3:1	reserved	-	R	0x0
0	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0

DPM_HOST_WDG_HOST_TIMEOUT – Address reserved for netX50 0x101c12c0

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

Bits	Name	Description	R/W	Default
31:0	ZERO_RO	reserved for netx50 DPM_HOST_WDG_HOST_TIMEOUT.	R	0x0

DPM_HOST_WDG_HOST_TRIG – Address reserved for netX50 0x101c12c4

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

Bits	Name	Description	R/W	Default
31:0	ZERO_RO	reserved for netx50 DPM_HOST_WDG_HOST_TRIG.	R	0x0

DPM_HOST_WDG_ARM_TIMEOUT – Address reserved for netX50 0x101c12c8

31:0
ZERO_RO

Bits	Name	Description	R/W	Default
31:0	ZERO_RO	reserved for netx50 DPM_HOST_WDG_ARM_TIMEOUT.	R	0x0

DPM_SYS_STA_BIGEND16 – DPM System Status Information Register in Big Endianess 16 Data Mapping 0x101c12cc

This register is read-only, using DPM_SYS_STA for programming.

This register can be used for firmware status information.

Reading this register data can be done from uninitialized DPM interface in the same way as reading netx version (DPM_NETX_VERSION_BIGEND16, DPM_NETX_VERSION) by using DPM_SYS_STA_BIGEND16 register.

31:16	15:12	11:10	9	8	7:0
reserved	HOST_STATE_SWAP_RO	NETX_STATE_SWAP_RO	RUN_RO	RDY_RO	NETX_STA_CODE_SWAP_RO

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:12	HOST_STATE_SWAP_RO	Bit field for Hilscher firmware compatibility.	R	0x0
11:10	NETX_STATE_SWAP_RO	Bit field for Hilscher firmware compatibility.	R	0x0
9	RUN_RO	Output state of netX RUN LED IO.	R	0x0
8	RDY_RO	Output state of netX RDY LED IO.	R	0x0
7:0	NETX_STA_CODE_SWAP_RO	Bit field for Hilscher firmware compatibility.	R	0x0

DPM_HOST_TMR_CTRL – Address reserved for netX50 0x101c12d0

31–0
ZERO_RO

Bits	Name	Description	R/W	Default
31:0	ZERO_RO	reserved for netx50 DPM_HOST_TMR_CTRL	R	0x0

DPM_HOST_TMR_START_VAL – Address reserved for netX50 0x101c12d4

31–0
ZERO_RO

Bits	Name	Description	R/W	Default
31:0	ZERO_RO	reserved for netx50 DPM_HOST_TMR_START_VAL.	R	0x0

DPM_HOST_SYS_STAT – DPM System Status Information Register 0x101c12d8

This register can be used for firmware status information.

Reading this register data can be done from uninitialized DPM interface in the same way as reading netx version (DPM_NETX_VERSION_BIGEND16, DPM_NETX_VERSION) by using DPM_SYS_STA_BIGEND16 register.

Note: This register is compatible to netx50 DPM_HOST_SYS_STAT 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	NETX_STA_CODE_RO	HOST_STATE	NETX_STATE_RO	RUN_RO	RDY_RO

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:8	NETX_STA_CODE_RO	Bit field for Hilscher firmware compatibility (read only). Note: This bit field can be changed by rdy_run_cfg-register inside ASIC_CTRL address area.	R/W	0x0
7:4	HOST_STATE	Bit field for Hilscher firmware compatibility. Note: This bit field can be read also at rdy_run_cfg-register inside ASIC_CTRL address area.	R/W	0x0
3:2	NETX_STATE_RO	Bit field for Hilscher firmware compatibility. Note: This bit field can be changed by rdy_run_cfg-register inside ASIC_CTRL address area.	R/W	0x0
1	RUN_RO	Output state of netX RUN LED IO. Note: This bit field can be changed by rdy_run_cfg-register inside ASIC_CTRL address area.	R/W	0x0
0	RDY_RO	Output state of netX RDY LED IO. Note: This bit field can be changed by rdy_run_cfg-register inside ASIC_CTRL address area.	R/W	0x0

DPM_HOST_RESET_REQ – DPM Reset Request Register 0x101c12dc

Note: This register is compatible to netx50 DPM_HOST_RESET_REQ register.

31 … 8	7 … 0
reserved	RESET_KEY

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	RESET_KEY	Reset key sequence register. A netx hardware reset is generated if the following sequence is written to this register: 1st access: write 0x00 2nd access: write 0x01 3rd access: write 0x03 4th access: write 0x07 5th access: write 0x0f 6th access: write 0x1f 7th access: write 0x3f 8th access: write 0x7f 9th access: write 0xff Note: The sequence must not interrupted by any other write access to any other DPM register. Read access have no influence.	R/W	0x0

DPM_HOST_INT_STAT0 – DPM Handshake Interrupt Status Register 0x101c12e0

Writing a ’1’ to an IRQ flag will clear the Interrupt. This is always done even if dpm_firmware_irq_mask is not set (this is compatible to netx50).

Note: This register is compatible to netx50 DPM_HOST_INT_STAT0 register, however some unused IRQs have been removed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_REQ	RES_MEM_LCK_RO	RES_WDG_NETX_RO	reserved		SYS_STA	RES_TMR_RO	reserved	IRQ_VECTOR								HS_EVENT15	HS_EVENT14	HS_EVENT13	HS_EVENT12	HS_EVENT11	HS_EVENT10	HS_EVENT9	HS_EVENT8	HS_EVENT7	HS_EVENT6	HS_EVENT5	HS_EVENT4	HS_EVENT3	HS_EVENT2	HS_EVENT1	HS_EVENT0

Bits	Name	Description	R/W	Default
31	INT_REQ	Interrupt Request for IRQs handled in this register. 0: No Interrupts to host requested by IRQ sources handled in this register. 1: IRQ sources handled in this register request a host IRQ. Note: This bit is masked by INT_EN-bit in dpm_firmware_irq_mask register. For propagation of INT_REQ to host, ARM or xPIC, INT_EN-bit must be set and firmware IRQ must be activated in related dpm_irq_* register.	R	0x0
30	RES_MEM_LCK_RO	reserved for Memory Lock IRQ flag (not available in this netX version).	R	0x0
29	RES_WDG_NETX_RO	reserved for netX supervision Watchdog Timeout IRQ flag (not available in this netX version).	R	0x0
28:27	reserved	-	R	0x0
26	SYS_STA	System Status Change IRQ flag.	R	0x0
25	RES_TMR_RO	reserved for Timer IRQ flag (not available in this netX version).	R	0x0
24	reserved	-	R	0x0
23:16	IRQ_VECTOR	Interrupt Vector according to status flags generated by enabled IRQ sources. Code IRQ status 0x00 No IRQ. 0x10 Handshake Cell 0 IRQ. 0x11 Handshake Cell 1 IRQ. 0x12 Handshake Cell 2 IRQ. 0x13 Handshake Cell 3 IRQ. 0x14 Handshake Cell 4 IRQ. 0x15 Handshake Cell 5 IRQ. 0x16 Handshake Cell 6 IRQ. 0x17 Handshake Cell 7 IRQ. 0x18 Handshake Cell 8 IRQ. 0x19 Handshake Cell 9 IRQ. 0x1a Handshake Cell 10 IRQ. 0x1b Handshake Cell 11 IRQ. 0x1c Handshake Cell 12 IRQ. 0x1d Handshake Cell 13 IRQ. 0x1e Handshake Cell 14 IRQ. 0x1f Handshake Cell 15 IRQ. 0x70 SYS_STA IRQ Other values are reserved. Note: The current IRQ state in VECTOR depends only on the single IRQ enable bits. It does not depend on global IRQ enable INT_EN. VECTOR shows always the highest priority enabled flagged IRQ even is INT_EN is ’0’.	R	0x0
15	HS_EVENT15	Handshake Event 15 IRQ Enable flag.	R	0x0
14	HS_EVENT14	Handshake Event 14 IRQ Enable flag.	R	0x0

Bits	Name	Description	R/W	Default
13	HS_EVENT13	Handshake Event 13 IRQ Enable flag.	R	0x0
12	HS_EVENT12	Handshake Event 12 IRQ Enable flag.	R	0x0
11	HS_EVENT11	Handshake Event 11 IRQ Enable flag.	R	0x0
10	HS_EVENT10	Handshake Event 10 IRQ Enable flag.	R	0x0
9	HS_EVENT9	Handshake Event 9 IRQ Enable flag.	R	0x0
8	HS_EVENT8	Handshake Event 8 IRQ Enable flag.	R	0x0
7	HS_EVENT7	Handshake Event 7 IRQ Enable flag.	R	0x0
6	HS_EVENT6	Handshake Event 6 IRQ Enable flag.	R	0x0
5	HS_EVENT5	Handshake Event 5 IRQ Enable flag.	R	0x0
4	HS_EVENT4	Handshake Event 4 IRQ Enable flag.	R	0x0
3	HS_EVENT3	Handshake Event 3 IRQ Enable flag.	R	0x0
2	HS_EVENT2	Handshake Event 2 IRQ Enable flag.	R	0x0
1	HS_EVENT1	Handshake Event 1 IRQ Enable flag.	R	0x0
0	HS_EVENT0	Handshake Event 0 IRQ Enable flag.	R	0x0

DPM_HOST_INT_EN0 – DPM Handshake Interrupt Enable Register 0x101c12f0

Note:
This register is compatible to netx50 DPM_HOST_INT_STAT0 register, however some unused IRQs have been removed.

Note:
HS_EVENT-bits are not read-only. This is netX50 compliant.

Recent netX50 Documentation marks HS_EVENT-bits as read-only. This is a dokumentation error.
For netX50 compatibility, these bits can also be controlled from netX-side in HANDSHAKE_CTRL address area.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INT_EN	RES_MEM_LCK_RO	RES_WDG_NETX_RO	reserved		SYS_STA	RES_TMR_RO	reserved									HS_EVENT15	HS_EVENT14	HS_EVENT13	HS_EVENT12	HS_EVENT11	HS_EVENT10	HS_EVENT9	HS_EVENT8	HS_EVENT7	HS_EVENT6	HS_EVENT5	HS_EVENT4	HS_EVENT3	HS_EVENT2	HS_EVENT1	HS_EVENT0

Bits	Name	Description	R/W	Default
31	INT_EN	Interrupt Enable for IRQs handled in this register. Only if this bit is set, global firmware IRQ will be asserted to host CPU, ARM or xPIC by dpm_irq_* registers. 0: No Interrupts to host, ARM or xPIC are generated by IRQ sources handled in this register. 1: Enabled IRQ sources handled in this register generate a host, ARM or xPIC IRQ if asserted. Note: Enable bits for single IRQ events are not affected if this bit is set or reset.	R/W	0x0
30	RES_MEM_LCK_RO	reserved for Memory Lock IRQ (not available in this netX version).	R/W	0x0
29	RES_WDG_NETX_RO	reserved for netX supervision Watchdog Timeout IRQ (not available in this netX version).	R/W	0x0
28:27	reserved	-	R	0x0
26	SYS_STA	System Status Change IRQ Enable.	R/W	0x0
25	RES_TMR_RO	reserved for Timer IRQ (not available in this netX version).	R/W	0x0
24:16	reserved	-	R	0x0
15	HS_EVENT15	Handshake Event 15 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
14	HS_EVENT14	Handshake Event 14 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
13	HS_EVENT13	Handshake Event 13 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
12	HS_EVENT12	Handshake Event 12 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
11	HS_EVENT11	Handshake Event 11 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
10	HS_EVENT10	Handshake Event 10 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
9	HS_EVENT9	Handshake Event 9 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
8	HS_EVENT8	Handshake Event 8 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
7	HS_EVENT7	Handshake Event 7 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
6	HS_EVENT6	Handshake Event 6 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0

Bits	Name	Description	R/W	Default
5	HS_EVENT5	Handshake Event 5 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
4	HS_EVENT4	Handshake Event 4 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
3	HS_EVENT3	Handshake Event 3 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
2	HS_EVENT2	Handshake Event 2 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
1	HS_EVENT1	Handshake Event 1 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0
0	HS_EVENT0	Handshake Event 0 IRQ Enable (also netX-controllable by HANDSHAKE_CTRL, netX50 comp.).	R/W	0x0

DPM_NETX_VERSION_BIGEND16 – DPM netX Version Register in Big Endianess 16 Data Mapping 0x101c12f8

This registers content is mirrored form ASIC_CTRL register area and can be set during netX booting phase by netX firmware.

This register is not valid if unlocked bit is not set in DPM_STATUS register.

Together with DPM_NETX_VERSION register, full 32bit version can be read by any host device, even if DPM interface is not initialized yet.

Bytes byte1 and byte3 can be always read here even if DPM is uninitialized (8bit default from DPM_CFG0X0 after power on) and host device has 8, 16 or 32bit data width.

	8bit DPM	16bit DPM	32bit DPM
byte 0 (D7:0)	byte read this address +1	adr_dpm_netx_version	adr_dpm_netx_version
byte 1 (D15:8)	byte read this address +0	byte read this address	DWord read this address
byte 2 (D23:16)	byte read this address +3	adr_dpm_netx_version	adr_dpm_netx_version
byte 3 (D31:24)	byte read this address +2	byte read this address +2	byte read this address +0

31–24	23–16	15–8	7–0
NETX_VERSION_BYTE2_SWAP	NETX_VERSION_BYTE3_SWAP	NETX_VERSION_BYTE0_SWAP	NETX_VERSION_BYTE1_SWAP

Bits	Name	Description	R/W	Default
31:24	NETX_VERSION_BYTE2_SWAP	netX version bits 24 to 16.	R	0x0
23:16	NETX_VERSION_BYTE3_SWAP	netX version bits 31 to 24.	R	0x0
15:8	NETX_VERSION_BYTE0_SWAP	netX version bits 8 to 0.	R	0x0
7:0	NETX_VERSION_BYTE1_SWAP	netX version bits 16 to 8.	R	0x0

DPM_NETX_VERSION – DPM netX Version Register 0x101c12fc

This register is mirrored form ASIC_CTRL register area and can be set during netX booting phase by netX firmware.

This register is not valid if unlocked bit is not set in DPM_STATUS register.

Together with DPM_NETX_VERSION register, full 32bit version can be read by any host device, even if DPM interface is not initialized yet.

Bytes byte0 and byte2 can be always read here even if DPM is uninitialized (8bit default from DPM_CFG0X0 after power on) and host device has 8, 16 or 32bit data width.

	8bit DPM	16bit DPM	32bit DPM
byte 0 (D7:0)	byte read this address +0	byte read this address	DWord read this address
byte 1 (D15:8)	byte read this address +1	adr_dpm_netx_version_bigend16	adr_dpm_netx_version_bigend16
byte 2 (D23:16)	byte read this address +2	byte read this address +2	byte read this address +0
byte 3 (D31:24)	byte read this address +3	adr_dpm_netx_version_bigend16	adr_dpm_netx_version_bigend16

31 30 29 28 27 26 25 24	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_VERSION_BYTE3	NETX_VERSION_BYTE2	NETX_VERSION_BYTE1	NETX_VERSION_BYTE0

Bits	Name	Description	R/W	Default
31:24	NETX_VERSION_BYTE3	netX version bits 31 to 24.	R	0x0
23:16	NETX_VERSION_BYTE2	netX version bits 24 to 16.	R	0x0
15:8	NETX_VERSION_BYTE1	netX version bits 16 to 8.	R	0x0
7:0	NETX_VERSION_BYTE0	netX version bits 8 to 0.	R	0x0

5.5 Handshake Registers

The handshake cells are located within the INTRAM5 and can be mapped to any 256 byte border. The handshake data width can be set to 8bit or 16bit.

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]

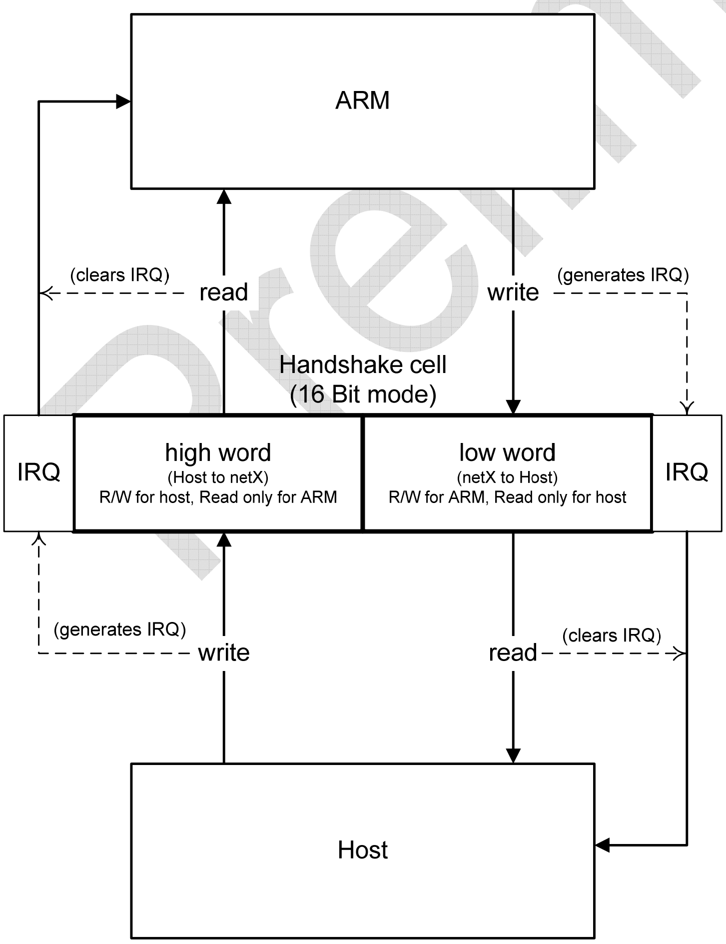
Bits	Name	Description	R/W	Default
31:16	HOST→NETX_DATA[15:0]	Handshake Data Flags host to netX [15:0]	R	0x00
15:0	NETX→Host_DATA[15:0]	Handshake Data Flags netX to host [15:0]	R/W	0x00

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

The following figure shows the16bit DPM handshake interaction.



The following table is a summary of all handshake configuration registers:

ARM Address	Register Name	Short Description
0x101c1100	HANDSHAKE_BASE_ADDR	Handshake Cell Address Base Configuration Register
0x101c1110	HANDSHAKE_DPM_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for DPM Register
0x101c1114	HANDSHAKE_DPM_IRQ_MASKED	Handshake Cell Masked Interrupt for DPM Register
0x101c1118	HANDSHAKE_DPM_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for DPM Register
0x101c111c	HANDSHAKE_DPM_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for DPM Register
0x101c1120	HANDSHAKE_ARM_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for ARM Register
0x101c1124	HANDSHAKE_ARM_IRQ_MASKED	Handshake Cell Masked Interrupt for ARM Register
0x101c1128	HANDSHAKE_ARM_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for ARM Register
0x101c112c	HANDSHAKE_ARM_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for ARM Register
0x101c1130	HANDSHAKE_XPIC_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for xPIC Register
0x101c1134	HANDSHAKE_XPIC_IRQ_MASKED	Handshake Cell Masked Interrupt for xPIC Register
0x101c1138	HANDSHAKE_XPIC_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for xPIC Register
0x101c113c	HANDSHAKE_XPIC_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for xPIC Register
0x101c1180	HANDSHAKE_HSC0_CTRL	Handshake Cell 0 Control Register
0x101c1184	HANDSHAKE_HSC1_CTRL	Handshake Cell 1 Control Register
0x101c1188	HANDSHAKE_HSC2_CTRL	Handshake Cell 2 Control Register
0x101c118c	HANDSHAKE_HSC3_CTRL	Handshake Cell 3 Control Register
0x101c1190	HANDSHAKE_HSC4_CTRL	Handshake Cell 4 Control Register
0x101c1194	HANDSHAKE_HSC5_CTRL	Handshake Cell 5 Control Register
0x101c1198	HANDSHAKE_HSC6_CTRL	Handshake Cell 6 Control Register
0x101c119c	HANDSHAKE_HSC7_CTRL	Handshake Cell 7 Control Register
0x101c11a0	HANDSHAKE_HSC8_CTRL	Handshake Cell 8 Control Register
0x101c11a4	HANDSHAKE_HSC9_CTRL	Handshake Cell 9 Control Register
0x101c11a8	HANDSHAKE_HSC10_CTRL	Handshake Cell 10 Control Register
0x101c11ac	HANDSHAKE_HSC11_CTRL	Handshake Cell 11 Control Register
0x101c11b0	HANDSHAKE_HSC12_CTRL	Handshake Cell 12 Control Register
0x101c11b4	HANDSHAKE_HSC13_CTRL	Handshake Cell 13 Control Register
0x101c11b8	HANDSHAKE_HSC14_CTRL	Handshake Cell 14 Control Register
0x101c11bc	HANDSHAKE_HSC15_CTRL	Handshake Cell 15 Control Register
0x101c11c0	HANDSHAKE_BUF_MAN0_CTRL	Handshake Triple Buffer Manager 0 Control Register
0x101c11c4	HANDSHAKE_BUF_MAN0_STATUS_CTRL_NETX	Handshake Triple Buffer Manager 0 netX Status and Control Register
0x101c11c8	HANDSHAKE_BUF_MAN0_STATUS_CTRL_HOST	Handshake Triple Buffer Manager 0 Host Status Register
0x101c11cc	HANDSHAKE_BUF_MAN0_WIN_MAP	DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 0
0x101c11d0	HANDSHAKE_BUF_MAN1_CTRL	Handshake Triple Buffer Manager 1 Control Register
0x101c11d4	HANDSHAKE_BUF_MAN1_STATUS_CTRL_NETX	Handshake Triple Buffer Manager 1 netX Status and Control Register
0x101c11d8	HANDSHAKE_BUF_MAN1_STATUS_CTRL_HOST	Handshake Triple Buffer Manager 1 Host Status Register
0x101c11dc	HANDSHAKE_BUF_MAN1_WIN_MAP	DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 1
0x00048000	HANDSHAKE_MIRROR_ITCM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0004fffc	HANDSHAKE_MIRROR_ITCM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address

0x04048000	HANDSHAKE_MIRROR_DTCM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0404fffc	HANDSHAKE_MIRROR_DTCM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x08048000	HANDSHAKE_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0804fffc	HANDSHAKE_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x10048000	HANDSHAKE_MIRROR_DPM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x1004fffc	HANDSHAKE_MIRROR_DPM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0xfff48000	HANDSHAKE_MIRROR_HI_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0xfff4fffc	HANDSHAKE_MIRROR_HI_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address

HANDSHAKE_BASE_ADDR – Handshake Cell Address Base Configuration Register0x101c1100

Handshake Cells are located in INTRAM5 and can be mapped to any 256 byte border.

Related master of an access to Handshake Cells is detected by the access to one of three INTRAM5 Mirrors:

Access via INTRAM5 dpm_mirror is interpreted by Handshake Cells as DPM access. This is regardless whether the access was really initiated by DPM master or not. E.g. if xPIC uses dpm_mirror of INTRAM5 for Handshake Cell access, this will be interpreted as DPM access and not as xPIC access.
INTRAM5 can be accessed by 4 different mirrors which are sub address areas of area HANDSHAKE. Furthermore HANDSHAKE address area is mirrored multiple inside whole netX address area. Each HANDSHAKE address area provides all 4 INTRAM5 mirrors.

There is one INTRAM5 mirror for each IRQ capable system master (DPM, xPIC, ARM) and one to access whole INTRAM5 area without any influence to HANDSHAKE_CTRL unit. However, each system master is able to address each INTRAM5 mirror. IRQs are always generated in dependency of mirror addressed by a master on access. IRQ generation does not depend on the master running an access.

Handshake Cell Setup example:
1. Configure Handshake Cell area offset (e.g. offset 0x200, set base256 to 0x2).
2. Configure used Handshake Cell width (8bit or 16 bit) in 'HANDSHAKE_HSCx_CTRL' registers.
3. Configure used Handshake Cells master association (e.g. ARM<->DPM) in 'HANDSHAKE_HSCx_CTRL' registers.

Example: typical ARM<-> DPM Handshake interaction:
1. ARM writes request to Handshake Cell N (address: intram5_arm_mirror+base256*256+N*4).
   -> DPM receives IRQ
2. DPM reads Handshake Cell N (address: intram5_dpm_mirror+base256*256+N*4).
   -> DPM IRQ clear.
3. DPM writes acknowledge to Handshake Cell N (address: intram5_dpm_mirror+base256*256+N*4).
   -> ARM receives IRQ
4. ARM reads Handshake Cell N (address: intram5_dpm_mirror+base256*256+N*4).
   -> ARM IRQ clear.

31 ... 13	12 ... 8	7 ... 0
reserved	BASE256	ZERO_RO

Bits	Name	Description	R/W	Default
31:13	reserved	-	R	0x0
12:8	BASE256	address base configuration in 256 byte steps inside INTRAM5	R/W	0x0
7:0	ZERO_RO	low address bits not configurable	R/W	0x0

HANDSHAKE_DPM_IRQ_RAW_CLEAR – Handshake Cell Raw Interrupt for DPM Register0x101c1110

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with '1' resets the appropriate IRQ.
Write access with '0' does not influence this bit.

Note:
DPM related IRQ status can also be read from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQs can also be cleared from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQ masks can also be read from DPM_HOST_INT_EN0 register (area DPM).

31 30 29 28 27 26 25 24	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	HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	VECTOR	Interrupt Vector generated by masked DPM IRQ flags. These bits are mirrored from handshake_dpm_irq_masked register. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R/W	0x0

HANDSHAKE_DPM_IRQ_MASKED – Handshake Cell Masked Interrupt for DPM Register0x101c1114

Show status of masked IRQs (as connected to DPM/host).

Note:
DPM related IRQ status can also be read from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQs can also be cleared from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQ masks can also be read from DPM_HOST_INT_EN0 register (area DPM).

31	30	29	28	27	26	25	24	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								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R	0x0
22	HSC14	Handshake Cell 14 IRQ.	R	0x0
21	HSC13	Handshake Cell 13 IRQ.	R	0x0
20	HSC12	Handshake Cell 12 IRQ.	R	0x0
19	HSC11	Handshake Cell 11 IRQ.	R	0x0
18	HSC10	Handshake Cell 10 IRQ.	R	0x0
17	HSC9	Handshake Cell 9 IRQ.	R	0x0
16	HSC8	Handshake Cell 8 IRQ.	R	0x0
15	HSC7	Handshake Cell 7 IRQ.	R	0x0
14	HSC6	Handshake Cell 6 IRQ.	R	0x0
13	HSC5	Handshake Cell 5 IRQ.	R	0x0
12	HSC4	Handshake Cell 4 IRQ.	R	0x0
11	HSC3	Handshake Cell 3 IRQ.	R	0x0
10	HSC2	Handshake Cell 2 IRQ.	R	0x0
9	HSC1	Handshake Cell 1 IRQ.	R	0x0
8	HSC0	Handshake Cell 0 IRQ.	R	0x0
7:0	VECTOR	Interrupt Vector generated by masked DPM IRQ flags. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R	0x0

HANDSHAKE_DPM_IRQ_MSK_SET – Handshake Cell Interrupt Mask Enable for DPM Register 0x101c1118

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit.
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 HANDSHAKE_DPM_IRQ_RAW_CLEAR.

Note:
DPM related IRQ status can also be read from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQs can also be cleared from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQ masks can also be read from DPM_HOST_INT_EN0 register (area DPM).

31 … 24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7 … 0
reserved	HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_DPM_IRQ_MSK_RESET – Handshake Cell Interrupt Mask Disable for DPM Register 0x101c111c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with '1' resets interrupt mask bit.
Write access with '0' does not influence this bit.
Read access shows actual interrupt mask.

Note:
DPM related IRQ status can also be read from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQs can also be cleared from DPM_HOST_INT_STAT0 register (area DPM).
DPM related IRQ masks can also be read from DPM_HOST_INT_EN0 register (area DPM).

31	30	29	28	27	26	25	24	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								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_ARM_IRQ_RAW_CLEAR – Handshake Cell Raw Interrupt for ARM Register0x101c1120

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
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
reserved								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	VECTOR	Interrupt Vector generated by masked ARM IRQ flags. These bits are mirrored from handshake_arm_irq_masked register. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R/W	0x0

HANDSHAKE_ARM_IRQ_MASKED – Handshake Cell Masked Interrupt for ARM Register0x101c1124

Show status of masked IRQs (as connected to 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	HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R	0x0
22	HSC14	Handshake Cell 14 IRQ.	R	0x0
21	HSC13	Handshake Cell 13 IRQ.	R	0x0
20	HSC12	Handshake Cell 12 IRQ.	R	0x0
19	HSC11	Handshake Cell 11 IRQ.	R	0x0
18	HSC10	Handshake Cell 10 IRQ.	R	0x0
17	HSC9	Handshake Cell 9 IRQ.	R	0x0
16	HSC8	Handshake Cell 8 IRQ.	R	0x0
15	HSC7	Handshake Cell 7 IRQ.	R	0x0
14	HSC6	Handshake Cell 6 IRQ.	R	0x0
13	HSC5	Handshake Cell 5 IRQ.	R	0x0
12	HSC4	Handshake Cell 4 IRQ.	R	0x0
11	HSC3	Handshake Cell 3 IRQ.	R	0x0
10	HSC2	Handshake Cell 2 IRQ.	R	0x0
9	HSC1	Handshake Cell 1 IRQ.	R	0x0
8	HSC0	Handshake Cell 0 IRQ.	R	0x0
7:0	VECTOR	Interrupt Vector generated by masked ARM IRQ flags. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R	0x0

HANDSHAKE_ARM_IRQ_MSK_SET – Handshake Cell Interrupt Mask Enable for ARM Register 0x101c1128

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 HANDSHAKE_ARM_IRQ_RAW_CLEAR.

31	30	29	28	27	26	25	24	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								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_ARM_IRQ_MSK_RESET – Handshake Cell Interrupt Mask Disable for ARM Register 0x101c112c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with '1' resets interrupt mask bit.
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								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_XPIC_IRQ_RAW_CLEAR – Handshake Cell Raw Interrupt for xPIC Register0x101c1130

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
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
reserved								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	VECTOR	Interrupt Vector generated by masked xPIC IRQ flags. These bits are mirrored from handshake_xpic_irq_masked register. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R/W	0x0

HANDSHAKE_XPIC_IRQ_MASKED – Handshake Cell Masked Interrupt for xPIC Register0x101c1134

Show status of masked IRQs (as connected to ARM/xPIC).

31..24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7..0
reserved	HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	VECTOR

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R	0x0
22	HSC14	Handshake Cell 14 IRQ.	R	0x0
21	HSC13	Handshake Cell 13 IRQ.	R	0x0
20	HSC12	Handshake Cell 12 IRQ.	R	0x0
19	HSC11	Handshake Cell 11 IRQ.	R	0x0
18	HSC10	Handshake Cell 10 IRQ.	R	0x0
17	HSC9	Handshake Cell 9 IRQ.	R	0x0
16	HSC8	Handshake Cell 8 IRQ.	R	0x0
15	HSC7	Handshake Cell 7 IRQ.	R	0x0
14	HSC6	Handshake Cell 6 IRQ.	R	0x0
13	HSC5	Handshake Cell 5 IRQ.	R	0x0
12	HSC4	Handshake Cell 4 IRQ.	R	0x0
11	HSC3	Handshake Cell 3 IRQ.	R	0x0
10	HSC2	Handshake Cell 2 IRQ.	R	0x0
9	HSC1	Handshake Cell 1 IRQ.	R	0x0
8	HSC0	Handshake Cell 0 IRQ.	R	0x0
7:0	VECTOR	Interrupt Vector generated by masked xPIC IRQ flags. Priority and Coding is compliant to netx50 HIF Handshake IRQ Vector: 0x00 : No IRQ. 0x10 : Handshake Cell 0 IRQ. 0x11 : Handshake Cell 1 IRQ. 0x12 : Handshake Cell 2 IRQ. 0x13 : Handshake Cell 3 IRQ. 0x14 : Handshake Cell 4 IRQ. 0x15 : Handshake Cell 5 IRQ. 0x16 : Handshake Cell 6 IRQ. 0x17 : Handshake Cell 7 IRQ. 0x18 : Handshake Cell 8 IRQ. 0x19 : Handshake Cell 9 IRQ. 0x1a : Handshake Cell 10 IRQ. 0x1b : Handshake Cell 11 IRQ. 0x1c : Handshake Cell 12 IRQ. 0x1d : Handshake Cell 13 IRQ. 0x1e : Handshake Cell 14 IRQ. 0x1f : Handshake Cell 15 IRQ. 0x20..0xff : reserved.	R	0x0

HANDSHAKE_XPIC_IRQ_MSK_SET – Handshake Cell Interrupt Mask Enable for xPIC Register 0x101c1138

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 HANDSHAKE_XPIC_IRQ_RAW_CLEAR.

31..24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7..0
reserved	HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_XPIC_IRQ_MSK_RESET – Handshake Cell Interrupt Mask Disable for xPIC Register 0x101c113c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with '1' resets interrupt mask bit.
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								HSC15	HSC14	HSC13	HSC12	HSC11	HSC10	HSC9	HSC8	HSC7	HSC6	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0	reserved							

Bits	Name	Description	R/W	Default
31:24	reserved	-	R	0x0
23	HSC15	Handshake Cell 15 IRQ.	R/W	0x0
22	HSC14	Handshake Cell 14 IRQ.	R/W	0x0
21	HSC13	Handshake Cell 13 IRQ.	R/W	0x0
20	HSC12	Handshake Cell 12 IRQ.	R/W	0x0
19	HSC11	Handshake Cell 11 IRQ.	R/W	0x0
18	HSC10	Handshake Cell 10 IRQ.	R/W	0x0
17	HSC9	Handshake Cell 9 IRQ.	R/W	0x0
16	HSC8	Handshake Cell 8 IRQ.	R/W	0x0
15	HSC7	Handshake Cell 7 IRQ.	R/W	0x0
14	HSC6	Handshake Cell 6 IRQ.	R/W	0x0
13	HSC5	Handshake Cell 5 IRQ.	R/W	0x0
12	HSC4	Handshake Cell 4 IRQ.	R/W	0x0
11	HSC3	Handshake Cell 3 IRQ.	R/W	0x0
10	HSC2	Handshake Cell 2 IRQ.	R/W	0x0
9	HSC1	Handshake Cell 1 IRQ.	R/W	0x0
8	HSC0	Handshake Cell 0 IRQ.	R/W	0x0
7:0	reserved	-	R	0x0

HANDSHAKE_HSC0_CTRL – Handshake Cell 0 Control Register 0x101c1180
HANDSHAKE_HSC1_CTRL – Handshake Cell 1 Control Register 0x101c1184
HANDSHAKE_HSC2_CTRL – Handshake Cell 2 Control Register 0x101c1188
HANDSHAKE_HSC3_CTRL – Handshake Cell 3 Control Register 0x101c118c
HANDSHAKE_HSC4_CTRL – Handshake Cell 4 Control Register 0x101c1190
HANDSHAKE_HSC5_CTRL – Handshake Cell 5 Control Register 0x101c1194
HANDSHAKE_HSC6_CTRL – Handshake Cell 6 Control Register 0x101c1198
HANDSHAKE_HSC7_CTRL – Handshake Cell 7 Control Register 0x101c119c
HANDSHAKE_HSC8_CTRL – Handshake Cell 8 Control Register 0x101c11a0
HANDSHAKE_HSC9_CTRL – Handshake Cell 9 Control Register 0x101c11a4
HANDSHAKE_HSC10_CTRL – Handshake Cell 10 Control Register 0x101c11a8
HANDSHAKE_HSC11_CTRL – Handshake Cell 11 Control Register 0x101c11ac
HANDSHAKE_HSC12_CTRL – Handshake Cell 12 Control Register 0x101c11b0
HANDSHAKE_HSC13_CTRL – Handshake Cell 13 Control Register 0x101c11b4
HANDSHAKE_HSC14_CTRL – Handshake Cell 14 Control Register 0x101c11b8
HANDSHAKE_HSC15_CTRL – Handshake Cell 15 Control Register 0x101c11bc

Handshake data width can be configured individually for each Handshake Cell.
In the ’mode’ bit field each Handshake Cell can be enabled or disabled and a handshake path (i.e. participating masters) can be configured individually.
When a Handshake Cell is enabled there are certain bytes writable only by certain related masters (view ’mode’ description).

31 … 5	4	3 … 2	1 … 0
reserved	WIDTH	reserved	MODE

Bits	Name	Description	R/W	Default
31:5	reserved	-	R	0x0
4	WIDTH	Handshake Cell Width. Coding: 0: 8 bit handshake width (netX50 compliant). 1: 16 bit handshake width (netX50 compliant).	R/W	0x0
3:2	-	reserved	R	0x0
1:0	MODE	Handshake Cell Mode. Coding: 00: Handshake Cell 0 is disabled. Related memory data is accessible without any restriction; no IRQs are generated at any access. 01: Use Handshake Cell 0 for handshaking between DPM and ARM 8bit handshaking (’width’ configuration is 0): DPM write data in data bits 31..24, bits 23..16 are read-only. ARM write data in data bits 23..16, bits 31..24 are read-only. 16bit handshaking (’width’ configuration is 1): DPM write data in data bits 31..16, bits 15..0 are read-only. ARM write data in data bits 15..0, bits 31..16 are read-only. 10: Use Handshake Cell 0 for handshaking between DPM and xPIC 8bit handshaking (’width’ configuration is 0): DPM write data in data bits 31..24, bits 23..16 are read-only, 15..0 is standard data memory xPIC write data in data bits 23..16, bits 31..24 are read-only, 15..0 is standard data memory Data bits 15..0 are standard data memory. 16bit handshaking (’width’ configuration is 1): DPM write data in data bits 31..16, bits 15..0 are read-only. xPIC write data in data bits 15..0, bits 31..16 are read-only.	R/W	0x0

Bits	Name	Description	R/W	Default
		11: Use Handshake Cell 0 for handshaking between ARM and xPIC 8bit handshaking ('width' configuration is 0): ARM write data in data bits 31..24, bits 23..16 are read-only, 15..0 is standard data memory xPIC write data in data bits 23..16, bits 31..24 are read-only, 15..0 is standard data memory Data bits 15..0 are standard data memory. 16bit handshaking ('width' configuration is 1): ARM write data in data bits 31..16, bits 15..0 are read-only. xPIC write data in data bits 15..0, bits 31..16 are read-only.		

HANDSHAKE_BUF_MAN0_CTRL – Handshake Triple Buffer Manager 0 Control Register0x101c11c0

Handshake Triple Buffer Manager 0 can be associated to Handshake Cell 2 HCF_PD_OUT_CMD/NCF_PD_OUT_ACK-bits for Host controlled DPM output data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31..4	3..2	1	0
reserved	BUF_DAM_CFG	RESET	HSC2_AUTO_PD_OUT

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3:2	BUF_DAM_CFG	Handshake Triple Buffer Manager 0 DPM Address Mapping Configuration. This bit field can be used to select DPM address mapping value manually or controlled by current buffer state of Handshake Triple Buffer Manager 0. Current buffer state of Handshake Triple Buffer Manager 0 can be determined and controlled in 'handshake_buf_man0_status_ctrl_netx' and 'handshake_buf_man0_status_ctrl_host' register. Coding: 00 : Use mapping value programmed in DPM configuration registers (i.e. buffer 0) 01 : Use alternative mapping 1 value programmed in 'handshake_buf_man0_win_map.win_map_buf1'. 10 : Use alternative mapping 2 value programmed in 'handshake_buf_man0_win_map.win_map_buf2'. 11 : Generate window mapping by current buffer state of Handshake Triple Buffer Manager 0. Note: Settings 00..10 can be used to control Window mapping manually.	R/W	0x0
1	RESET	Handshake Triple Buffer Manager 0 FSM Reset. Note: This bit is cleared automatically (it is writable but can also be changed by hardware).	R/W	0x0
0	HSC2_AUTO_PD_OUT	Handshake Cell 2 Handshake Triple Buffer Manager 0 action enable for HCF_PD_OUT_CMD/NCF_PD_OUT_ACK. If this bit is set, Triple Buffer Manager 0 is used for Host controlled DPM output data handling and DPM auto buffer window change. NCF_PD_OUT_ACK-bit of Handshake Cell 2 is then controlled by Hardware and read-only for software.	R/W	0x0

Bits	Name	Description	R/W	Default
		The following steps will be performed automatically by hardware after buffer change was requested by host: 1. Current host write buffer is released and a new host write buffer is requested. DPM window mapping is changed according to new host write buffer (must be enabled in DPM address area). 2. Handshake Cell 2 NCF_PD_OUT_ACK-bit is changed to state of Handshake Cell 2 HCF_PD_OUT_CMD-bit. to confirm new valid write buffer for host. Buffer change request event triggering this process: Host writes Handshake Cell 2 and host written data HCF_PD_OUT_CMD-bit 6 is not equal to current Handshake Cell 2 NCF_PD_OUT_ACK-bit (netX writable bit). Note: Location of HCF_PD_OUT_CMD/NCF_PD_OUT_ACK bits inside the 32 bit Handshake DWord depends on programmed width of Handshake Cell 2 ('adr_handshake_hsc2_ctrl.width'): - 16 bit handshake width of Handshake Cell 2: HCF_PD_OUT_CMD: located in bit 22 (6+16) of HS DWord. HCF_PD_OUT_ACK: located in bit 6 (6+0) of HS DWord. - 8 bit handshake width of Handshake Cell 2: HCF_PD_OUT_CMD: located in bit 30 (6+24) of HS DWord. HCF_PD_OUT_ACK: located in bit 22 (6+16) of HS DWord. Note: IRQ generation of Handshake Cell 2 for ARM and xPIC is not affected by this bit. ARM or xPIC receive IRQ when DPM/host requests output buffer change and Handshake Cell 2 IRQ is enabled for DPM and ARM or xPIC (handshake_hsc2_ctrl.mode is '01' or '10'). DPM/host receives IRQ after buffer change is performed by Handshake Triple Buffer Manager 1 if Handshake Cell 2 IRQ is enabled for DPM (handshake_hsc2_ctrl.mode is '01' or '10').		

HANDSHAKE_BUF_MAN0_STATUS_CTRL_NETX – Handshake Triple Buffer Manager 0 netX Status and Control Register 0x101c11c4

On read this register provides current status of netX side of Handshake Triple Buffer Manager 0. Buffer requests can be done by writing this register.

Handshake Triple Buffer Manager 0 can be associated to Handshake Cell 2 Bits 6 and 22 (16+6)
for Host controlled DPM output data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31–6	5–4	3–2	1–0
reserved	CMD	reserved	BUF_RO

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	CMD	Handshake Triple Buffer Manager 0 Command for netX buffer. Command coding: 00 : nop/idle 01 : request new read buffer 10 : request new write buffer 11 : release current buffer Note: This bit field will be reset to nop/idle automatically after command was performed (it is writable but can also be changed by hardware).	R/W	0x0
3:2	reserved	-	R	0x0
1:0	BUF_RO	Handshake Triple Buffer Manager 0 valid netX Buffer. Coding: 00 : Buffer 0 valid. 01 : Buffer 1 valid. 10 : Buffer 2 valid. 11 : No buffer is valid. Note: This bit field is read only accessible.	R/W	0x3

HANDSHAKE_BUF_MAN0_STATUS_CTRL_HOST – Handshake Triple Buffer Manager 0 Host Status Register 0x101c11c8

On read this register provides current status of host side of Handshake Triple Buffer Manager 0. Buffer requests can be done by writing this register.

Handshake Triple Buffer Manager 0 can be associated to Handshake Cell 2 Bits 6 and 22 (16+6)
for Host controlled DPM output data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31	30	29	28	27	26	25	24	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																										CMD		reserved		BUF_RO	

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	CMD	Handshake Triple Buffer Manager 1 Command for host buffer. Command coding: 00 : nop/idle 01 : request new read buffer 10 : request new write buffer 11 : release current buffer Note: This bit field will be reset to nop/idle automatically after command was performed (it is writable but can also be changed by hardware).	R	0x0
3:2	reserved	-	R	0x0
1:0	BUF_RO	Handshake Triple Buffer Manager 0 valid Host Buffer. If DPM auto buffer window change for host controlled DPM input data handling is enabled, this bit field defines which mapping value is used. Coding: 00 : Buffer 0 valid (mapping value programmed inside DPM module is used if auto window mapping is enabled). 01 : Buffer 1 valid (mapping value programmed from 'handshake_buf_man1_win_map.win_map_buf1' is used if auto window mapping is enabled). 10 : Buffer 2 valid (mapping value programmed from 'handshake_buf_man1_win_map.win_map_buf2' is used if auto window mapping is enabled). 11 : No buffer is valid (mapping value programmed inside DPM module is used if auto window mapping is enabled). Note: This bit field is read only accessible.	R	0x0

HANDSHAKE_BUF_MAN0_WIN_MAP – DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 0 0x101c11cc

Handshake Triple Buffer Manager 0 can be associated to Handshake Cell 2 Bits 6 and 22 (16+6) for Host controlled DPM output data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

If DPM auto buffer window change is enabled, buffer 0 related DPM window mapping is window mapping programmed for related window in DPM address are.

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="3">reserved</td><td colspan="13">WIN_MAP_BUF2</td><td colspan="3">reserved</td><td colspan="13">WIN_MAP_BUF1</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:29	reserved	-	R	0x0
28:16	WIN_MAP_BUF2	Buffer 2 of Handshake Triple Buffer Manager 0 Alternative DPM Window Address Map. This win_map entry is used if DPM auto buffer window change is enabled and if Buffer 2 of Handshake Triple Buffer Manager 0 is valid.	R/W	0x0
15:13	reserved	-	R	0x0
12:0	WIN_MAP_BUF1	Buffer 1 of Handshake Triple Buffer Manager 0 Alternative DPM Window Address Map. This win_map entry is used if DPM auto buffer window change is enabled and if Buffer 1 of Handshake Triple Buffer Manager 0 is valid.	R/W	0x0

HANDSHAKE_BUF_MAN1_CTRL – Handshake Triple Buffer Manager 1 Control Register0x101c11d0

Handshake Triple Buffer Manager 1 can be associated to Handshake Cell 2 HCF_PD_IN_CMD/NCF_PD_IN_ACK-bits for Host controlled DPM input data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31 … 4	3 … 2	1	0
reserved	BUF_DAM_CFG	RESET	HSC2_AUTO_PD_IN

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3:2	BUF_DAM_CFG	Handshake Triple Buffer Manager 1 DPM Address Mapping Configuration. This bit field can be used to select DPM address mapping value manually or controlled by current buffer state of Handshake Triple Buffer Manager 1. Current buffer state of Handshake Triple Buffer Manager 1 can be determined and controlled in 'handshake_buf_man1_status_ctrl_netx' and 'handshake_buf_man1_status_ctrl_host' register. Coding: 00 : Use mapping value programmed in DPM configuration registers (i.e. buffer 0) 01 : Use alternative mapping 1 value programmed in 'handshake_buf_man1_win_map.win_map_buf1'. 10 : Use alternative mapping 2 value programmed in 'handshake_buf_man1_win_map.win_map_buf2'. 11 : Generate window mapping by current buffer state of Handshake Triple Buffer Manager 1. Note: Settings 00..10 can be used to control Window mapping manually.	R/W	0x0
1	RESET	Handshake Triple Buffer Manager 1 FSM Reset. Note: This bit is cleared automatically (it is writable but can also be changed by hardware).	R/W	0x0
0	HSC2_AUTO_PD_IN	Handshake Cell 2 Handshake Triple Buffer Manager 1 action enable for HCF_PD_IN_CMD/NCF_PD_IN_ACK. If this bit is set, Triple Buffer Manager 1 is used for Host controlled DPM input data handling and DPM auto buffer window change. NCF_PD_IN_ACK-bit of Handshake Cell 2 is then controlled by Hardware and read-only for software. The following steps will be performed automatically by hardware after buffer change was requested by host: 1. New read buffer (last ARM written buffer) is requested for host. DPM window mapping is changed according to new host read buffer (must be enabled in DPM address area). 2. Handshake Cell 2 NCF_PD_IN_ACK-bit is changed to state of Handshake Cell 2 HCF_PD_IN_CMD-bit to confirm new valid read data for host. Buffer change request event triggering this process: Host writes Handshake Cell 2 and host written data HCF_PD_IN_CMD-bit is not equal to current Handshake Cell 2 HCF_PD_IN_ACK-bit. Note: Location of HCF_PD_IN_CMD/NCF_PD_IN_ACK bits inside the 32 bit Handshake DWord depends on programmed width of Handshake Cell 2 ('adr_handshake_hsc2_ctrl.width'):	R/W	0x0

Bits	Name	Description	R/W	Default
		- 16 bit handshake width of Handshake Cell 2: HCF_PD_IN_CMD: located in bit 23 (7+16) of HS DWord. HCF_PD_IN_ACK: located in bit 7 (7+0) of HS DWord. - 8 bit handshake width of Handshake Cell 2: HCF_PD_IN_CMD: located in bit 31 (7+24) of HS DWord. HCF_PD_IN_ACK: located in bit 23 (7+16) of HS DWord. Note: IRQ generation of Handshake Cell 2 for ARM and xPIC is not affected by this bit. ARM or xPIC receive IRQ when DPM/host requests input buffer change and Handshake Cell 2 IRQ is enabled for DPM and ARM or xPIC (handshake_hsc2_ctrl.mode is ’01’ or ’10’). DPM/host receives IRQ after buffer change is performed by Handshake Triple Buffer Manager 1 if Handshake Cell 2 IRQ is enabled for DPM (handshake_hsc2_ctrl.mode is ’01’ or ’10’).		

HANDSHAKE_BUF_MAN1_STATUS_CTRL_NETX – Handshake Triple Buffer Manager 1 netX Status and Control Register 0x101c11d4

On read this register provides current status of netX side of Handshake Triple Buffer Manager 1. Buffer requests can be done by writing this register.

Handshake Triple Buffer Manager 1 can be associated to Handshake Cell 2 Bits 6 and 22 (16+6)
for Host controlled DPM input data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31 … 6	5 4	3 2	1 0
reserved	CMD	reserved	BUF_RO

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	CMD	Handshake Triple Buffer Manager 1 Command for netX buffer. Command coding: 00 : nop/idle 01 : request new read buffer 10 : request new write buffer 11 : release current buffer Note: This bit field will be reset to nop/idle automatically after command was performed (it is writable but can also be changed by hardware).	R/W	0x0
3:2	reserved	-	R	0x0
1:0	BUF_RO	Handshake Triple Buffer Manager 1 valid netX Buffer. Coding: 00 : Buffer 0 valid. 01 : Buffer 1 valid. 10 : Buffer 2 valid. 11 : No buffer is valid. Note: This bit field is read only accessible.	R/W	0x3

HANDSHAKE_BUF_MAN1_STATUS_CTRL_HOST – Handshake Triple Buffer Manager 1 Host Status Register 0x101c11d8

On read this register provides current status of host side of Handshake Triple Buffer Manager 1. Buffer requests can be done by writing this register.

Handshake Triple Buffer Manager 1 can be associated to Handshake Cell 2 Bits 6 and 22 (16+6)
for host controlled DPM input data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

31 … 6	5 4	3 2	1 0
reserved	CMD	reserved	BUF_RO

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5:4	CMD	Handshake Triple Buffer Manager 1 Command for host buffer. Command coding: 00 : nop/idle 01 : request new read buffer 10 : request new write buffer 11 : release current buffer Note: This bit field will be reset to nop/idle automatically after command was performed (it is writable but can also be changed by hardware).	R	0x0
3:2	reserved	-	R	0x0
1:0	BUF_RO	Handshake Triple Buffer Manager 1 valid Host Buffer. If DPM auto buffer window change for host controlled DPM input data handling is enabled, this bit field defines which mapping value is used. Coding: 00 : Buffer 0 valid (mapping value programmed inside DPM module is used if auto window mapping is enabled). 01 : Buffer 1 valid (mapping value programmed from 'handshake_buf_man1_win_map.win_map_buf1' is used if auto window mapping is enabled). 10 : Buffer 2 valid (mapping value programmed from 'handshake_buf_man1_win_map.win_map_buf2' is used if auto window mapping is enabled). 11 : No buffer is valid (mapping value programmed inside DPM module is used if auto window mapping is enabled). Note: This bit field is read only accessible.	R	0x0

HANDSHAKE_BUF_MAN1_WIN_MAP – DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 1 0x101c11dc

Handshake Triple Buffer Manager 1 can be associated to Handshake Cell 2 Bits 7 and 23 (16+7)
for Host controlled DPM input data handling and DPM auto buffer window change.

Note:
DPM auto buffer window change configuration is controlled inside DPM address area at window map registers.

If DPM auto buffer window change is enabled, buffer 1 related DPM window mapping is window mapping programmed for related window in DPM address are.

31 … 29	28 … 16	15 … 13	12 … 0
reserved	WIN_MAP_BUF2	reserved	WIN_MAP_BUF1

Bits	Name	Description	R/W	Default
31:29	reserved	-	R	0x0
28:16	WIN_MAP_BUF2	Buffer 2 of Handshake Triple Buffer Manager 1 Alternative DPM Window Address Map. This win_map entry is used if DPM auto buffer window change is enabled and if Buffer 2 of Handshake Triple Buffer Manager 1 is valid.	R/W	0x0
15:13	reserved	-	R	0x0
12:0	WIN_MAP_BUF1	Buffer 1 of Handshake Triple Buffer Manager 1 Alternative DPM Window Address Map. This win_map entry is used if DPM auto buffer window change is enabled and if Buffer 1 of Handshake Triple Buffer Manager 1 is valid.	R/W	0x0

HANDSHAKE_MIRROR_ITCM_HANDSHAKE_BASE – HANDSHAKE_MIRROR_ITCM Internal Handshake AHBL Slave 5 Start Address 0x00048000
HANDSHAKE_MIRROR_DTCM_HANDSHAKE_BASE – HANDSHAKE_MIRROR_DTCM Internal Handshake AHBL Slave 5 Start Address 0x04048000
HANDSHAKE_HANDSHAKE_BASE – HANDSHAKE Internal Handshake AHBL Slave 5 Start Address 0x08048000
HANDSHAKE_MIRROR_DPM_HANDSHAKE_BASE – HANDSHAKE_MIRROR_DPM Internal Handshake AHBL Slave 5 Start Address 0x10048000
HANDSHAKE_MIRROR_HI_HANDSHAKE_BASE – HANDSHAKE_MIRROR_HI Internal Handshake AHBL Slave 5 Start Address 0xfff48000

Area size: 32kB INTRAM5 is mirrored inside this area 4 times for handshake Interrupt purpose.
For details view ’adr_intram5_base’ and HANDSHAKE_CTRL area.
Read accesses in this memory area: 0WS, byte accessible
Write accesses in this memory area: 0WS, byte accessable

31 … 0
HANDSHAKE_BASE

Bits	Name	Description	R/W	Default
31:0	HANDSHAKE_BASE	Handshake start address	R/W	0x0

HANDSHAKE_MIRROR_ITCM_HANDSHAKEEND – HANDSHAKE_MIRROR_ITCM Internal SRAM AHBL Slave 5 End Address 0x0004fffc
HANDSHAKE_MIRROR_DTCM_HANDSHAKEEND – HANDSHAKE_MIRROR_DTCM Internal SRAM AHBL Slave 5 End Address 0x0404fffc
HANDSHAKE_HANDSHAKEEND – HANDSHAKE Internal SRAM AHBL Slave 5 End Address 0x0804fffc
HANDSHAKE_MIRROR_DPM_HANDSHAKEEND – HANDSHAKE_MIRROR_DPM Internal SRAM AHBL Slave 5 End Address 0x1004fffc
HANDSHAKE_MIRROR_HI_HANDSHAKEEND – HANDSHAKE_MIRROR_HI Internal SRAM AHBL Slave 5 End Address 0xfff4fffc

31 … 0
HANDSHAKEEND

Bits	Name	Description	R/W	Default
31:0	HANDSHAKEEND	Handshake end address	R/W	0x0

6 xPIC – Peripheral Interface Controller

One of the new features invented with the netX10 is the xPIC, a special processor, optimized to control different kinds of sensors and actuators allowing fast response and data processing.

6.1 General Structure

The xPIC core is a generic 32-Bit RISC processor with Harvard architecture. To ensure maximal calculation power and fast deterministic response, the xPIC core has its own local SRAM, consisting of 8 KByte program memory and 8 KByte data memory.

The local PRAM is mapped to address area 0x00000000 – 0x00001fff for the xPIC only, the same address area as the DRAM (for data access). The ARM / System access this ram from 0x1018c000 – 0x1018dfff for preloading (preload window). This (ARM-) preloading is only possible if the xPIC is in HOLD. The xPIC can not access the PRAM within this preload window.

The local DRAM is mapped to address area 0x00000000 – 0x00001fff for the xPIC only the same address area as the PRAM (for instruction access). The ARM / System access this ram from 0x1018e000 – 0x1018ffff for preloading and data exchange. This (ARM-) access is possible if the xPIC is in HOLD or is not in HOLD. The xPIC can access the DRAM within this preload window too but this is very slow.

The DRAM should NOT be used for data exchange between xPIC and any other AHB master during runtime since this may cause undeterministic waitstates blocking the AHB channel.

The Arithmetic Logical Unit (ALU) and Address Unit together with 13 Working Registers are the “heart” of the xPIC processor, which has a 32-bit instruction and a 32-bit data bus resulting in a 32-bit address space, allowing the xPIC to access the complete netX10 internal memory range.

6.2 xPIC Debug Unit

The debug unit starts and stops the xPIC Core. A hardware reset is possible with the debug unit. There are two hardware breakpoints for debug. The breakpoints can create an interrupt to the host CPU. Software breakpoints and singlestep mode are set and reset in the debug unit. The status of xPIC break and break reasons can be polled also.

The functions of the debugging unit are accessible through registers in the netX10 address space.

6.3 xPIC_VIC – xPIC Vectored Interrupt Controller

Two interrupt types are available in xPIC: (Normal) Interrupt Request (IRQ) and Fast Interrupt Request (FIQ).

The Interrupts are controlled by the xPIC Vectored Interrupt Controller (xPIC_VIC). There are 16 interrupt sources out of 64 selectable. All sources can be triggered manually for software interrupt and debug. The vector table must be stored in memory.

The following table shows a summary of XPIC_VIC related registers.

ARM Address	Register Name	Short Description
0x10140800	XPIC_VIC_CONFIG	XPIC VIC Config Register
0x10140804	XPIC_VIC_RAW_INTR0	XPIC VIC Raw0 Interrupt Status Register
0x10140808	XPIC_VIC_RAW_INTR1	XPIC VIC Raw1 Interrupt Status Register
0x1014080c	XPIC_VIC_SOFTINT0_SET	XPIC VIC Software0 Interrupt Set Register
0x10140810	XPIC_VIC_SOFTINT1_SET	XPIC VIC Software1 Interrupt Set Register
0x10140814	XPIC_VIC_SOFTINT0_RESET	XPIC VIC Software0 Interrupt Reset Register
0x10140818	XPIC_VIC_SOFTINT1_RESET	XPIC VIC Software1 Interrupt Reset Register
0x1014081c	XPIC_VIC_FIQ_ADDR	XPIC VIC FIQ Vector Address 0 Register
0x10140820	XPIC_VIC_IRQ_ADDR	XPIC VIC Normal IRQ Address Register
0x10140824	XPIC_VIC_VECTOR_ADDR	XPIC VIC IRQ Vector Address
0x10140828	XPIC_VIC_TABLE_BASE_ADDR	XPIC VIC IRQ Table Base Address
0x1014082c	XPIC_VIC_FIQ_VECT_CONFIG	XPIC VIC FIQ Vector Config Register
0x10140830	XPIC_VIC_VECT_CONFIG0	XPIC VIC IRQ Vector0 Config Register
0x10140834	XPIC_VIC_VECT_CONFIG1	XPIC VIC IRQ Vector1 Config Register
0x10140838	XPIC_VIC_VECT_CONFIG2	XPIC VIC IRQ Vector2 Config Register
0x1014083c	XPIC_VIC_VECT_CONFIG3	XPIC VIC IRQ Vector3 Config Register
0x10140840	XPIC_VIC_VECT_CONFIG4	XPIC VIC IRQ Vector4 Config Register
0x10140844	XPIC_VIC_VECT_CONFIG5	XPIC VIC IRQ Vector5 Config Register
0x10140848	XPIC_VIC_VECT_CONFIG6	XPIC VIC IRQ Vector6 Config Register
0x1014084c	XPIC_VIC_VECT_CONFIG7	XPIC VIC IRQ Vector7 Config Register
0x10140850	XPIC_VIC_VECT_CONFIG8	XPIC VIC IRQ Vector8 Config Register
0x10140854	XPIC_VIC_VECT_CONFIG9	XPIC VIC IRQ Vector9 Config Register
0x10140858	XPIC_VIC_VECT_CONFIG10	XPIC VIC IRQ Vector10 Config Register
0x1014085c	XPIC_VIC_VECT_CONFIG11	XPIC VIC IRQ Vector11 Config Register
0x10140860	XPIC_VIC_VECT_CONFIG12	XPIC VIC IRQ Vector12 Config Register
0x10140864	XPIC_VIC_VECT_CONFIG13	XPIC VIC IRQ Vector13 Config Register
0x10140868	XPIC_VIC_VECT_CONFIG14	XPIC VIC IRQ Vector14 Config Register
0x1014086c	XPIC_VIC_VECT_CONFIG15	XPIC VIC IRQ Vector15 Config Register
0x10140870	XPIC_VIC_DEFAULT0	XPIC Default Interrupt Vector Select0
0x10140874	XPIC_VIC_DEFAULT1	XPIC Default Interrupt Vector Select1

XPIC_VIC_CONFIG – XPIC VIC Config Register 0x10140800

Use XPIC_VIC_CONFIG register to enable or disable xPIC VIC.

31	30	29	28	27	26	25	24	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																														TABLE	ENABLE

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	TABLE	use far or near Table 0 = Base Pointer Addr for IRQ Jmp Table + (n*4) DWORD Table 1 = Base Pointer Addr for IRQ Jmp Table + (n*16) 4 DWORD Table n = IRQ vector number	R/W	0x0
0	ENABLE	global enable of xPIC VIC (0: disable/ 1: enable)	R/W	0x0

XPIC_VIC_RAW_INTR0 – XPIC VIC Raw0 Interrupt Status Register 0x10140804

The XPIC_VIC_RAW_INTR0 register provides the status of the source raw 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
ADC	ENCODER	PWM	RESERVED28	DMAC	SYSSTATE	INT_PHY	MSYNC3	RESERVED23	RESERVED22	MSYNC0	RESERVED20	RESERVED19	RESERVED18	COM0	GPIO	HIF	RESERVED14	I2C	SPI	USB	RESERVED10	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	TIMER2	GPIO_TIMER	TIMER1	TIMER0	SW0

Bits	Name	Description	R/W	Default
31	ADC	ADC0 or ADC1	R	0x0
30	ENCODER	Any encoder IRQ	R	0x0
29	PWM	Any PWM IRQ	R	0x0
28	RESERVED28	reserved for netX compatibility (trigger_lt)	R	0x0
27	DMAC	DMA controller	R	0x0
26	SYSSTATE	License error or extmem_timeout	R	0x0
25	INT_PHY	Interrupt from internal Phy	R	0x0
24	MSYNC3	reserved for SW IRQ from ARM to xPIC	R	0x0
23	RESERVED23	reserved for netX compatibility (msync2)	R	0x0
22	RESERVED22	reserved for netX compatibility (msync1)	R	0x0
21	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
20	RESERVED20	reserved (com3)	R	0x0
19	RESERVED19	reserved for netX compatibility (com2)	R	0x0
18	RESERVED18	reserved for netX compatibility (com1)	R	0x0
17	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0
16	GPIO	other external Interrupts from GPIO 0-6 / IOLINK	R	0x0
15	HIF	HIF/DPM interrupt	R	0x0
14	RESERVED14	reserved for netX compatibility (lcd)	R	0x0
13	I2C	I2C	R	0x0
12	SPI	combimned SPI0, SPI1 interrupt	R	0x0
11	USB	USB interrupt	R	0x0
10	RESERVED10	reserved for netX compatibility (uart2)	R	0x0
9	UART1	UART 1	R	0x0
8	UART0	UART 0 -> Diagnostic channel, Windows CE required	R	0x0
7	WATCHDOG	Watchdog IRQ from XPIC_WDG module	R	0x0
6	GPIO7	external interrupt 7, Windows CE required (NMI)	R	0x0
5	SYSTIME_S	Systime 1day IRQ from XPIC_TIMER module	R	0x0
4	TIMER2	xPIC Timer2 from XPIC_TIMER Module	R	0x0
3	GPIO_TIMER	GPIO Timer0 or Timer1 (sep. gpio_irq registers for ARM(intlogic) and xPIC(intlogic_motion))	R	0x0
2	TIMER1	xPIC Timer1 from XPIC_TIMER Module	R	0x0
1	TIMER0	xPIC Timer0 from XPIC_TIMER Module Real time operating system timer, Windows CE required	R	0x0
0	SW0	Reserved for Software Interrupt	R	0x0

XPIC_VIC_RAW_INTR1 – XPIC VIC Raw1 Interrupt Status Register 0x10140808

The XPIC_VIC_RAW_INTR1 register provides the status of the source raw 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
MISALIGN	RESERVED30	RESERVED29	RESERVED28	RESERVED27	RESERVED26	RESERVED25	RESERVED24	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO_TIMER1	GPIO_TIMER0	SPI1	SPI0	MPWM_FAILURE	MPWM1	MPWM0	MP1	MP0	CAP3	CAP2	CAP1	CAP0	ENC1	ENC0	ADC1	ADC0

Bits	Name	Description	R/W	Default
31	MISALIGN	xPIC data misalignment	R	0x0
30	RESERVED30	reserved	R	0x0
29	RESERVED29	reserved	R	0x0
28	RESERVED28	reserved	R	0x0
27	RESERVED27	reserved	R	0x0
26	RESERVED26	reserved	R	0x0
25	RESERVED25	reserved	R	0x0
24	RESERVED24	reserved	R	0x0
23	GPIO6	gpio6	R	0x0
22	GPIO5	gpio5	R	0x0
21	GPIO4	gpio4	R	0x0
20	GPIO3	gpio3	R	0x0
19	GPIO2	gpio2	R	0x0
18	GPIO1	gpio1	R	0x0
17	GPIO0	gpio0	R	0x0
16	GPIO_TIMER1	gpio_timer1	R	0x0
15	GPIO_TIMER0	gpio_timer0	R	0x0
14	SPI1	spi1	R	0x0
13	SPI0	spi0	R	0x0
12	MPWM_FAILURE	mpwm_failure	R	0x0
11	MPWM1	mpwm1	R	0x0
10	MPWM0	mpwm0	R	0x0
9	MP1	Encoder mp1	R	0x0
8	MP0	Encoder mp0	R	0x0
7	CAP3	Encoder Capture Unit 3	R	0x0
6	CAP2	Encoder Capture Unit 2	R	0x0
5	CAP1	Encoder Capture Unit 1	R	0x0
4	CAP0	Encoder Capture Unit 0	R	0x0
3	ENC1	Encoder1 (ovfl, edge)	R	0x0
2	ENC0	Encoder0 (ovfl, edge)	R	0x0
1	ADC1	ADC1	R	0x0
0	ADC0	ADC0	R	0x0

XPIC_VIC_SOFTINT0_SET – XPIC VIC Software0 Interrupt Set Register 0x1014080c

Write access with '1' sets corresponding software interrupt bit.
Write access with '0' does not influence this bit.
Read access shows actual software interrupt status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC	ENCODER	PWM	RESERVED28	DMAC	SYSSTATE	INT_PHY	MSYNC3	RESERVED23	RESERVED22	MSYNC0	RESERVED20	RESERVED19	RESERVED18	COM0	GPIO	HIF	RESERVED14	I2C	SPI	USB	RESERVED10	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	TIMER2	GPIO_TIMER	TIMER1	TIMER0	SW0

Bits	Name	Description	R/W	Default
31	ADC	ADC0 or ADC1	R/W	0x0
30	ENCODER	Any encoder IRQ	R/W	0x0
29	PWM	Any PWM IRQ	R/W	0x0
28	RESERVED28	reserved for netX compatibility (trigger_lt)	R/W	0x0
27	DMAC	DMA controller	R/W	0x0
26	SYSSTATE	License error or extmem_timeout	R/W	0x0
25	INT_PHY	Interrupt from internal Phy	R/W	0x0
24	MSYNC3	reserved for SW IRQ from ARM to xPIC	R/W	0x0
23	RESERVED23	reserved for netX compatibility (msync2)	R/W	0x0
22	RESERVED22	reserved for netX compatibility (msync1)	R/W	0x0
21	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
20	RESERVED20	reserved (com3)	R/W	0x0
19	RESERVED19	reserved for netX compatibility (com2)	R/W	0x0
18	RESERVED18	reserved for netX compatibility (com1)	R/W	0x0
17	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0
16	GPIO	other external Interrupts from GPIO 0-6 / IOLINK	R/W	0x0
15	HIF	HIF/DPM interrupt	R/W	0x0
14	RESERVED14	reserved for netX compatibility (lcd)	R/W	0x0
13	I2C	I2C	R/W	0x0
12	SPI	combimned SPI0, SPI1 interrupt	R/W	0x0
11	USB	USB interrupt	R/W	0x0
10	RESERVED10	reserved for netX compatibility (uart2)	R/W	0x0
9	UART1	UART 1	R/W	0x0
8	UART0	UART 0 -> Diagnostic channel, Windows CE required	R/W	0x0
7	WATCHDOG	Watchdog IRQ from XPIC_WDG module	R/W	0x0
6	GPIO7	external interrupt 7, Windows CE required (NMI)	R/W	0x0
5	SYSTIME_S	Systime 1day IRQ from XPIC_TIMER module	R/W	0x0
4	TIMER2	xPIC Timer2 from XPIC_TIMER Module	R/W	0x0
3	GPIO_TIMER	GPIO Timer0 or Timer1 (sep. gpio_irq registers for ARM(intlogic) and xPIC(intlogic_motion))	R/W	0x0
2	TIMER1	xPIC Timer1 from XPIC_TIMER Module	R/W	0x0
1	TIMER0	xPIC Timer0 from XPIC_TIMER Module Real time operating system timer, Windows CE required	R/W	0x0
0	SW0	Reserved for Software Interrupt	R/W	0x0

XPIC_VIC_SOFTINT1_SET – XPIC VIC Software1 Interrupt Set Register 0x10140810

Write access with '1' sets corresponding software interrupt bit.
Write access with '0' does not influence this bit.
Read access shows actual software interrupt status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISALIGN	RESERVED30	RESERVED29	RESERVED28	RESERVED27	RESERVED26	RESERVED25	RESERVED24	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO_TIMER1	GPIO_TIMER0	SPI1	SPI0	MPWM_FAILURE	MPWM1	MPWM0	MP1	MP0	CAP3	CAP2	CAP1	CAP0	ENC1	ENC0	ADC1	ADC0

Bits	Name	Description	R/W	Default
31	MISALIGN	xPIC data misalignment	R/W	0x0
30	RESERVED30	reserved	R/W	0x0
29	RESERVED29	reserved	R/W	0x0
28	RESERVED28	reserved	R/W	0x0
27	RESERVED27	reserved	R/W	0x0
26	RESERVED26	reserved	R/W	0x0
25	RESERVED25	reserved	R/W	0x0
24	RESERVED24	reserved	R/W	0x0
23	GPIO6	gpio6	R/W	0x0
22	GPIO5	gpio5	R/W	0x0
21	GPIO4	gpio4	R/W	0x0
20	GPIO3	gpio3	R/W	0x0
19	GPIO2	gpio2	R/W	0x0
18	GPIO1	gpio1	R/W	0x0
17	GPIO0	gpio0	R/W	0x0
16	GPIO_TIMER1	gpio_timer1	R/W	0x0
15	GPIO_TIMER0	gpio_timer0	R/W	0x0
14	SPI1	spi1	R/W	0x0
13	SPI0	spi0	R/W	0x0
12	MPWM_FAILURE	mpwm_failure	R/W	0x0
11	MPWM1	mpwm1	R/W	0x0
10	MPWM0	mpwm0	R/W	0x0
9	MP1	Encoder mp1	R/W	0x0
8	MP0	Encoder mp0	R/W	0x0
7	CAP3	Encoder Capture Unit 3	R/W	0x0
6	CAP2	Encoder Capture Unit 2	R/W	0x0
5	CAP1	Encoder Capture Unit 1	R/W	0x0
4	CAP0	Encoder Capture Unit 0	R/W	0x0
3	ENC1	Encoder1 (ovfl, edge)	R/W	0x0
2	ENC0	Encoder0 (ovfl, edge)	R/W	0x0
1	ADC1	ADC1	R/W	0x0
0	ADC0	ADC0	R/W	0x0

XPIC_VIC_SOFTINT0_RESET – XPIC VIC Software0 Interrupt Reset Register 0x10140814

Write access with '1' reset the corresponding software interrupt bit.
Write access with '0' does not influence this bit.
Read access shows actual software interrupt status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC	ENCODER	PWM	RESERVED28	DMAC	SYSSTATE	INT_PHY	MSYNC3	RESERVED23	RESERVED22	MSYNC0	RESERVED20	RESERVED19	RESERVED18	COM0	GPIO	HIF	RESERVED14	I2C	SPI	USB	RESERVED10	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	TIMER2	GPIO_TIMER	TIMER1	TIMER0	SW0

Bits	Name	Description	R/W	Default
31	ADC	ADC0 or ADC1	R/W	0x0
30	ENCODER	Any encoder IRQ	R/W	0x0
29	PWM	Any PWM IRQ	R/W	0x0
28	RESERVED28	reserved for netX compatibility (trigger_lt)	R/W	0x0
27	DMAC	DMA controller	R/W	0x0
26	SYSSTATE	License error or extmem_timeout	R/W	0x0
25	INT_PHY	Interrupt from internal Phy	R/W	0x0
24	MSYNC3	reserved for SW IRQ from ARM to xPIC	R/W	0x0
23	RESERVED23	reserved for netX compatibility (msync2)	R/W	0x0
22	RESERVED22	reserved for netX compatibility (msync1)	R/W	0x0
21	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R/W	0x0
20	RESERVED20	reserved (com3)	R/W	0x0
19	RESERVED19	reserved for netX compatibility (com2)	R/W	0x0
18	RESERVED18	reserved for netX compatibility (com1)	R/W	0x0
17	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R/W	0x0
16	GPIO	other external Interrupts from GPIO 0-6 / IOLINK	R/W	0x0
15	HIF	HIF/DPM interrupt	R/W	0x0
14	RESERVED14	reserved for netX compatibility (lcd)	R/W	0x0
13	I2C	I2C	R/W	0x0
12	SPI	combimned SPI0, SPI1 interrupt	R/W	0x0
11	USB	USB interrupt	R/W	0x0
10	RESERVED10	reserved for netX compatibility (uart2)	R/W	0x0
9	UART1	UART 1	R/W	0x0
8	UART0	UART 0 -> Diagnostic channel, Windows CE required	R/W	0x0
7	WATCHDOG	Watchdog IRQ from XPIC_WDG module	R/W	0x0
6	GPIO7	external interrupt 7, Windows CE required (NMI)	R/W	0x0
5	SYSTIME_S	Systime 1day IRQ from XPIC_TIMER module	R/W	0x0
4	TIMER2	xPIC Timer2 from XPIC_TIMER Module	R/W	0x0
3	GPIO_TIMER	GPIO Timer0 or Timer1 (sep. gpio_irq registers for ARM(intlogic) and xPIC(intlogic_motion))	R/W	0x0
2	TIMER1	xPIC Timer1 from XPIC_TIMER Module	R/W	0x0
1	TIMER0	xPIC Timer0 from XPIC_TIMER Module Real time operating system timer, Windows CE required	R/W	0x0
0	SW0	Reserved for Software Interrupt	R/W	0x0

XPIC_VIC_SOFTINT1_RESET – XPIC VIC Software1 Interrupt Reset Register 0x10140818

Write access with '1' reset the corresponding software interrupt bit.
Write access with '0' does not influence this bit.
Read access shows actual software interrupt status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MISALIGN	RESERVED30	RESERVED29	RESERVED28	RESERVED27	RESERVED26	RESERVED25	RESERVED24	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0	GPIO_TIMER1	GPIO_TIMER0	SPI1	SPI0	MPWM_FAILURE	MPWM1	MPWM0	MP1	MP0	CAP3	CAP2	CAP1	CAP0	ENC1	ENC0	ADC1	ADC0

Bits	Name	Description	R/W	Default
31	MISALIGN	xPIC data misalignment	R/W	0x0
30	RESERVED30	reserved	R/W	0x0
29	RESERVED29	reserved	R/W	0x0
28	RESERVED28	reserved	R/W	0x0
27	RESERVED27	reserved	R/W	0x0
26	RESERVED26	reserved	R/W	0x0
25	RESERVED25	reserved	R/W	0x0
24	RESERVED24	reserved	R/W	0x0
23	GPIO6	gpio6	R/W	0x0
22	GPIO5	gpio5	R/W	0x0
21	GPIO4	gpio4	R/W	0x0
20	GPIO3	gpio3	R/W	0x0
19	GPIO2	gpio2	R/W	0x0
18	GPIO1	gpio1	R/W	0x0
17	GPIO0	gpio0	R/W	0x0
16	GPIO_TIMER1	gpio_timer1	R/W	0x0
15	GPIO_TIMER0	gpio_timer0	R/W	0x0
14	SPI1	spi1	R/W	0x0
13	SPI0	spi0	R/W	0x0
12	MPWM_FAILURE	mpwm_failure	R/W	0x0
11	MPWM1	mpwm1	R/W	0x0
10	MPWM0	mpwm0	R/W	0x0
9	MP1	Encoder mp1	R/W	0x0
8	MP0	Encoder mp0	R/W	0x0
7	CAP3	Encoder Capture Unit 3	R/W	0x0
6	CAP2	Encoder Capture Unit 2	R/W	0x0
5	CAP1	Encoder Capture Unit 1	R/W	0x0
4	CAP0	Encoder Capture Unit 0	R/W	0x0
3	ENC1	Encoder1 (ovfl, edge)	R/W	0x0
2	ENC0	Encoder0 (ovfl, edge)	R/W	0x0
1	ADC1	ADC1	R/W	0x0
0	ADC0	ADC0	R/W	0x0

XPIC_VIC_FIQ_ADDR – XPIC VIC FIQ Vector Address 0 Register 0x1014081c

The XPIC_VIC_FIQ_ADDR register contains the Interrupt Service Routine (ISR) address of the FIQ 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

VAL

Bits	Name	Description	R/W	Default
31:0	VAL	FIQ handler address	R/W	0x0

XPIC_VIC_IRQ_ADDR – XPIC VIC Normal IRQ Address Register 0x10140820

The XPIC_VIC_IRQ_ADDR register contains the Interrupt Service Routine (ISR) address of the normal 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

VAL

Bits	Name	Description	R/W	Default
31:0	VAL	IRQ handler address	R/W	0x0

XPIC_VIC_VECTOR_ADDR – XPIC VIC IRQ Vector Address 0x10140824

Read access get actuel highest prior IRQ.
Read access get XPIC_VIC_TABLE_BASE_ADDR + IRQ Number * (4/16) (near or far Table).

31 30 29 28 27 26 25 24 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	IRQ vector address	R	0x0

XPIC_VIC_TABLE_BASE_ADDR – XPIC VIC IRQ Table Base Address 0x10140828

This register contains the Base Pointer Address for IRQ Jump Table.

31 30 29 28 27 26 25 24 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	IRQ Table base address	R/W	0x0

XPIC_VIC_FIQ_VECT_CONFIG – XPIC VIC FIQ Vector Config Register 0x1014082c

XPIC_VIC_FIQ_VECT_CONFIG registers select FIQ 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
31:7	reserved	-	R	0x0
6	ENABLE	vector interrupt enable	R/W	0x0
5:0	INT_SOURCE	INT_SOURCE 0-64	R/W	0x0

XPIC_VIC_VECT_CONFIG0 – XPIC VIC IRQ Vector0 Config Register 0x10140830
XPIC_VIC_VECT_CONFIG1 – XPIC VIC IRQ Vector1 Config Register 0x10140834
XPIC_VIC_VECT_CONFIG2 – XPIC VIC IRQ Vector2 Config Register 0x10140838
XPIC_VIC_VECT_CONFIG3 – XPIC VIC IRQ Vector3 Config Register 0x1014083c
XPIC_VIC_VECT_CONFIG4 – XPIC VIC IRQ Vector4 Config Register 0x10140840
XPIC_VIC_VECT_CONFIG5 – XPIC VIC IRQ Vector5 Config Register 0x10140844
XPIC_VIC_VECT_CONFIG6 – XPIC VIC IRQ Vector6 Config Register 0x10140848
XPIC_VIC_VECT_CONFIG7 – XPIC VIC IRQ Vector7 Config Register 0x1014084c
XPIC_VIC_VECT_CONFIG8 – XPIC VIC IRQ Vector8 Config Register 0x10140850
XPIC_VIC_VECT_CONFIG9 – XPIC VIC IRQ Vector9 Config Register 0x10140854
XPIC_VIC_VECT_CONFIG10 – XPIC VIC IRQ Vector10 Config Register 0x10140858
XPIC_VIC_VECT_CONFIG11 – XPIC VIC IRQ Vector11 Config Register 0x1014085c
XPIC_VIC_VECT_CONFIG12 – XPIC VIC IRQ Vector12 Config Register 0x10140860
XPIC_VIC_VECT_CONFIG13 – XPIC VIC IRQ Vector13 Config Register 0x10140864
XPIC_VIC_VECT_CONFIG14 – XPIC VIC IRQ Vector14 Config Register 0x10140868
XPIC_VIC_VECT_CONFIG15 – XPIC VIC IRQ Vector15 Config Register 0x1014086c

XPIC_VIC_VECT_CONFIG 0-15 registers select interrupt source and set if it is enabled.

For all interrupt sources (wired-OR), XPIC_VIC_VECT_CONFIG0 has the highest priority, while XPIC_VIC_VECT_CONFIG15 has the lowest priority.

31 30 29 28 27 26 25 24 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
31:7	-	reserved	R	0x0
6	ENABLE	vector interrupt enable	R/W	0x0
5:0	INT_SOURCE	INT_SOURCE 0-64	R/W	0x0

XPIC_VIC_DEFAULT0 – XPIC Default Interrupt Vector Select0 0x10140870

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	select int0 - int31 (wired-OR) 1-selected 0-not selected	R/W	0x0

XPIC_VIC_DEFAULT1 – XPIC Default Interrupt Vector Select1 0x10140874

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	select int32 - int63 (wired-OR) 1-selected 0-not selected	R/W	0x0

6.4 xPIC_TIMER

The xPIC Timer module contains 3 timers with 3 independent interrupt sources. Each timer has one preload register and one timer register and can be configured in 3 different modes.

It is also possible to trigger the motion encoder unit with each timer in each mode. Additional there is a systime seconds compare function in the xPIC timer unit which also can generate an interrupt.

The following table shows a summary of all registers related to xPIC timers.

ARM Address	Register Name	Short Description
0x10140700	XPIC_TIMER_CONFIG_TIMER0	xPIC TIMER Config Register0
0x10140704	XPIC_TIMER_CONFIG_TIMER1	xPIC TIMER Config Register1
0x10140708	XPIC_TIMER_CONFIG_TIMER2	xPIC TIMER Config Register2
0x1014070c	XPIC_TIMER_PRELOAD_TIMER0	xPIC TIMER Timer 0 Preload
0x10140710	XPIC_TIMER_PRELOAD_TIMER1	xPIC TIMER Timer 1 Preload
0x10140714	XPIC_TIMER_PRELOAD_TIMER2	xPIC TIMER Timer 2 Preload
0x10140718	XPIC_TIMER_TIMER0	xPIC TIMER Timer 0
0x1014071c	XPIC_TIMER_TIMER1	xPIC TIMER Timer 1
0x10140720	XPIC_TIMER_TIMER2	xPIC TIMER Timer 2
0x10140724	XPIC_TIMER_IRQ_RAW	xPIC_TIMER Raw IRQ Register
0x10140728	XPIC_TIMER_IRQ_MASKED	xPIC_TIMER Masked IRQ Register
0x1014072c	XPIC_TIMER_IRQ_MSK_SET	xPIC_TIMER Interrupt Mask Enable
0x10140730	XPIC_TIMER_IRQ_MSK_RESET	xPIC_TIMER Interrupt Mask Disable
0x10140734	XPIC_TIMER_SYSTIME_S	xPIC_TIMER Upper SYSTIME Register
0x10140738	XPIC_TIMER_SYSTIME_NS	xPIC_TIMER Lower SYSTIME Register
0x1014073c	XPIC_TIMER_COMPARE_SYSTIME_S_VALUE	xPIC_TIMER SYSTIME Sec Compare Register

XPIC_TIMER_CONFIG_TIMER0 – xPIC TIMER Config Register0 0x10140700
XPIC_TIMER_CONFIG_TIMER1 – xPIC TIMER Config Register1 0x10140704
XPIC_TIMER_CONFIG_TIMER2 – xPIC TIMER Config Register2 0x10140708

31 … 2	1 … 0
reserved	MODE

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1:0	MODE	2'b00 : Timer stops at 0 2'b01 : Timer is preload with value from preload register at 0 2'b10 : Timer (value) compare with systime (once) 2'b11 : reserved	R/W	0x0

XPIC_TIMER_PRELOAD_TIMER0 – xPIC TIMER Timer 0 Preload 0x1014070c
XPIC_TIMER_PRELOAD_TIMER1 – xPIC TIMER Timer 1 Preload 0x10140710
XPIC_TIMER_PRELOAD_TIMER2 – xPIC TIMER Timer 2 Preload 0x10140714

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	preload value	R/W	0x0

XPIC_TIMER_TIMER0 – xPIC TIMER Timer 0 0x10140718
XPIC_TIMER_TIMER1 – xPIC TIMER Timer 1 0x1014071c
XPIC_TIMER_TIMER2 – xPIC TIMER Timer 2 0x10140720

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	actual value of timer / systime compare value	R/W	0x0

XPIC_TIMER_IRQ_RAW – xPIC_TIMER Raw IRQ Register 0x10140724

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with '1' resets the appropriate IRQ.
Write access with '0' does not influence this bit.

31 … 4	3	2	1	0
reserved	SYSTIME_S_IRQ	TIMER2_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime_s Interrupt	R/W	0x0
2	TIMER2_IRQ	Timer 2 Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

XPIC_TIMER_IRQ_MASKED – xPIC_TIMER Masked IRQ Register 0x10140728

Show status of masked IRQs (as connected to ARM/xPIC).

31 … 4	3	2	1	0
reserved	SYSTIME_S_IRQ	TIMER2_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime_s Interrupt	R	0x0
2	TIMER2_IRQ	Timer 2 Interrupt	R	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R	0x0

XPIC_TIMER_IRQ_MSK_SET – xPIC_TIMER Interrupt Mask Enable 0x1014072c

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 XPIC_TIMER _IRQ_RAW.

31 … 4	3	2	1	0
reserved	SYSTIME_S_IRQ	TIMER2_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime_s Interrupt	R/W	0x0
2	TIMER2_IRQ	Timer 2 Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

XPIC_TIMER_IRQ_MSK_RESET – xPIC_TIMER Interrupt Mask Disable 0x10140730

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with '1' resets interrupt mask bit.
Write access with '0' does not influence this bit.
Read access shows actual interrupt mask.

31:4	3	2	1	0
reserved	SYSTIME_S_IRQ	TIMER2_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime_s Interrupt	R/W	0x0
2	TIMER2_IRQ	Timer 2 Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

XPIC_TIMER_SYSTIME_S – xPIC_TIMER Upper SYSTIME Register 0x10140734

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
This register should be dedicated to accesses via xPIC.
ARM software should access systime via ARM_TIMER at systime_s.
Host software should access systime via DPM at systime_s.

31:0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Systime high: Sample systime_ns at read access to systime_s. Value is incremented, if systime_ns reaches systime_border.	R	0x0

XPIC_TIMER_SYSTIME_NS – xPIC_TIMER Lower SYSTIME Register 0x10140738

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
If no systime_s is read before (e.g. at 2nd read access of systime_ns), the actual value of systime_ns is read.
This register should be dedicated to accesses via xPIC.
ARM software should access systime via ARM_TIMER at systime_ns.
Host software should access systime via DPM at systime_ns.

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Systime low: Sample systime_ns at read access to systime_s. Without sample read systime_s, read the actual value of systime_ns.	R	0x0

XPIC_TIMER_COMPARE_SYSTIME_S_VALUE – xPIC_TIMER SYSTIME Sec Compare Register 0x1014073c

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Compare value with systime_s (seconds): xpic_timer_irq_raw-Systime_s is set, if systime_s matches.	R/W	0x0

6.5 xPIC Watchdog

The XPIC_WDG (xPIC watchdog) unit serves as an external system observer that signals a hardware or software failure. Once activated the watchdog must be “fed” by the xPIC program in order to keep it “quiet”. If the watchdog is not fed within a defined time, the system is assumed to be failed. The failure can be reported in two different ways:

- Generate an interrupt for the xPIC
- Generate an interrupt for the ARM

The following table shows a summary of xPIC_WDG registers.

ARM Address	Register Name	Short Description
0x10140900	XPIC_WDG_TRIG	xPIC Watchdog Trigger Register
0x10140904	XPIC_WDG_COUNTER	xPIC Watchdog Counter Register
0x10140908	XPIC_WDG_XPIC_IRQ_TIMEOUT	xPIC Watchdog xPIC Interrupt Timout Register
0x1014090c	XPIC_WDG_ARM_IRQ_TIMEOUT	xPIC Watchdog ARM Interrupt Timout Register
0x10140910	XPIC_WDG_IRQ_RAW	xPIC Watchdog Raw Interrupt Register
0x10140914	XPIC_WDG_IRQ_MASKED	xPIC Watchdog Masked IRQ Register
0x10140918	XPIC_WDG_IRQ_MSK_SET	xPIC Watchdog Interrupt Mask Enable
0x1014091c	XPIC_WDG_IRQ_MSK_RESET	xPIC Watchdog Interrupt Mask Disable

XPIC_WDG_TRIG – netX xPIC Watchdog Trigger Register 0x10140900

The watchdog access code is generated by a pseudo random generator.

31	30 29	28	27 26 25	24	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_ENABLE	reserved	WDG_COUNTER_TRIGGER_W	reserved	IRQ_REQ_WATCHDOG	reserved	WDG_ACCESS_CODE

Bits	Name	Description	R/W	Default
31	WRITE_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	0x0
30:29	reserved	-	R	0x0
28	WDG_COUNTER_TRIGGER_W	Watchdog trigger bit: Bit must be set to trigger the watchdog counter. When read, this bit is always ’0’	R/W	0x0
27:25	reserved	-	R	0x0
24	IRQ_REQ_WATCHDOG	xPIC IRQ request of watchdog, writing 1 deletes IRQ to xPIC	R/W	0x0
23:20	reserved	-	R	0x0
19:0	WDG_ACCESS_CODE	Watchdog access code for triggering. A read access gives the next 16 bit code for trigger. A write access with correct access code will trigger the watchdog counter.	R/W	0x0

XPIC_WDG_COUNTER – netX xPIC Watchdog Counter Register 0x10140904

31 … 17	16 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16:0	VAL	Actual watchdog counter value: Bit 16 shows: 1: Watchdog is counting down from xpic_irq_timeout to 0 for xPIC-IRQ 0: Watchdog is counting down from arm_irq_timeout to 0 for ARM-IRQ	R	0x0

XPIC_WDG_XPIC_IRQ_TIMEOUT – xPIC Watchdog xPIC Interrupt Timout Register 0x10140908

XPIC_WDG_XPIC_IRQ_TIMEOUT or XPIC_WDG_ARM_IRQ_TIMEOUT must be nonzero to enable watchdog.

31 … 16	15 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	VAL	Watchdog interrupt timeout	R/W	0x0

XPIC_WDG_ARM_IRQ_TIMEOUT – xPIC Watchdog ARM Interrupt Timout Register 0x1014090c

XPIC_WDG_XPIC_IRQ_TIMEOUT or XPIC_WDG_ARM_IRQ_TIMEOUT must be nonzero to enable watchdog.

31 … 16	15 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	VAL	Watchdog ARM interrupt timeout	R/W	0x0

XPIC_WDG_IRQ_RAW – xPIC Watchdog Raw Interrupt Register 0x10140910

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with '1' resets the appropriate IRQ.
Write access with '0' does not influence this bit.

31 … 1	0
reserved	WDG_ARM_IRQ

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	WDG_ARM_IRQ	Interrupt from xPIC Watchdog to ARM	R/W	0x0

XPIC_WDG_IRQ_MASKED – xPIC Watchdog Masked IRQ Register 0x10140914

Show status of masked IRQ (as connected to ARM/xPIC).

31 … 1	0
reserved	WDG_ARM_IRQ

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	WDG_ARM_IRQ	Interrupt from xPIC Watchdog to ARM	R	0x0

XPIC_WDG_IRQ_MSK_SET – xPIC Watchdog Interrupt Mask Enable 0x10140918

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit.
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 XPIC_WDG_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

31–1	0
reserved	WDG_ARM_IRQ

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	WDG_ARM_IRQ	Interrupt from xPIC Watchdog to ARM	R/W	0x0

XPIC_WDG_IRQ_MSK_RESET – xPIC Watchdog Interrupt Mask Disable 0x1014091c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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

31–1	0
reserved	WDG_ARM_IRQ

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	WDG_ARM_IRQ	Interrupt from xPIC Watchdog to ARM	R/W	0x0

7 Fast IO and Motion Control Functions

The Fast IO and Motion units will typically be controlled by the xPIC, however they are also accessible by the ARM CPU if necessary. To avoid arbitration conflicts when ARM and xPIC are accessing internal modules, the Fast IO and Motion units are connected via a separate bus (intlogic_motion bus), which can be accessed in parallel to the intlogic bus, where standard modules like UART, USB, SPI are connected.

7.1 MPWM Unit

The netX10 is equipped with an optimized Motion-PWM unit with a high resolution of 2.5 ns. This MPWM unit is directly coupled with ADC and Encoder units for exact sampling and synchronization of any measurement device (e.g. ADCs for motor current measurement or encoder capture).

The PWM unit has two time base counters, which both can be used as a source for any of the eight compare units. The compare level can be set for every compare unit directly or at the next timer zero-crossing via shadow register.

The PWM signals are shared with Multiplex Matrix signals MMIO[11:4], while each of the eight PWM outputs can be enabled and configured individually. A configurable failure unit allows to use any of the 24 MMIO signals as failure input which can individually set the PWM outputs to their predefined state (low, high, Hi-Z or “don’t-touch”).

Two different Interrupts can be generated by the PWM unit on counter_zero, counter_max, and posi-tive or negative signal edge on any PWM signal in input mode, while each IRQ can be delayed up to 20.48 us (programmable in 20 ns steps).

The following table is a summary of MPWM related registers.

ARM Address	Register Name	Short Description
0x10140500	MPWM_CONFIG_COUNTER	Counter Config Register
0x10140504	MPWM_CONFIG_PINS	Pins Config Register
0x10140508	MPWM_CONFIG_FAILURE	Failure Config Register
0x1014050c	MPWM_IRQ_CONFIG	IRQ Config Register
0x10140510	MPWM_IRQ_RAW	Raw IRQ
0x10140514	MPWM_IRQ_MASKED	Masked IRQ
0x10140518	MPWM_IRQ_MSK_SET	IRQ Enable Mask
0x1014051c	MPWM_IRQ_MSK_RESET	IRQ Disable Mask
0x10140520	MPWM_CNT0_PERIOD	Counter 0 Period
0x10140524	MPWM_CNT0	Counter 0 Value
0x10140528	MPWM_CNT0_SYSTIME	Counter 0 Start Systime
0x1014052c	MPWM_CNT0_WATCHDOG	Counter 0 Watchdog
0x10140530	MPWM_CNT1_PERIOD	Counter 1 Period
0x10140534	MPWM_CNT1	Counter 1 Value
0x10140538	MPWM_CNT1_SYSTIME	Counter 1 Start Systime
0x1014053c	MPWM_CNT1_WATCHDOG	Counter 1 Watchdog
0x10140540	MPWM_T0	PWM Channel 0 Threshold
0x10140544	MPWM_T1	PWM Channel 1 Threshold
0x10140548	MPWM_T2	PWM Channel 2 Threshold
0x1014054c	MPWM_T3	PWM Channel 3 Threshold
0x10140550	MPWM_T4	PWM Channel 4 Threshold
0x10140554	MPWM_T5	PWM Channel 5 Threshold
0x10140558	MPWM_T6	PWM Channel 6 Threshold
0x1014055c	MPWM_T7	PWM Channel 7 Threshold
0x10140560	MPWM_T0_SHADOW	PWM Channel 0 Threshold Shadow

0x10140564	MPWM_T1_SHADOW	PWM Channel 1 Threshold Shadow
0x10140568	MPWM_T2_SHADOW	PWM Channel 2 Threshold Shadow
0x1014056c	MPWM_T3_SHADOW	PWM Channel 3 Threshold Shadow
0x10140570	MPWM_T4_SHADOW	PWM Channel 4 Threshold Shadow
0x10140574	MPWM_T5_SHADOW	PWM Channel 5 Threshold Shadow
0x10140578	MPWM_T6_SHADOW	PWM Channel 6 Threshold Shadow
0x1014057c	MPWM_T7_SHADOW	PWM Channel 7 Threshold Shadow

MPWM_CONFIG_COUNTER – Counter Config Register 0x10140500

31 … 6	5	4	3	2	1	0
reserved	DUAL_EDGE1	WAVEFORM_CNT1	RUN_CNT1	DUAL_EDGE0	WAVEFORM_CNT0	RUN_CNT0

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5	DUAL_EDGE1	Dual Edge Mode for shadow registers references on counter1: 1: update Shadow registers at mpwm_cnt==0 and mpwm_cnt==max 0: update Shadow registers only at mpwm_cnt==0	R/W	0x0
4	WAVEFORM_CNT1	Waveform of counter1: 0: Triangle 1: Sawtooth	R/W	0x0
3	RUN_CNT1	Run counter1: 0: stop counter1 after the actual cycle 1: run counter1	R/W	0x0
2	DUAL_EDGE0	Dual Edge Mode for shadow registers references on counter0: 1: update Shadow registers at mpwm_cnt==0 and mpwm_cnt==max 0: update Shadow registers only at mpwm_cnt==0	R/W	0x0
1	WAVEFORM_CNT0	Waveform of counter0: 0: Triangle 1: Sawtooth	R/W	0x0
0	RUN_CNT0	Run counter0: 0: stop counter0 after the actual cycle 1: run counter0	R/W	0x0

MPWM_CONFIG_PINS – Pins Config Register 0x10140504

31	30 29 28	27	26 25 24	23	22 21 20	19	18 17 16	15	14 13 12	11	10 9 8	7	6 5 4	3	2 1 0
SHADOW7	CFG_7	SHADOW6	CFG_6	SHADOW5	CFG_5	SHADOW4	CFG_4	SHADOW3	CFG_3	SHADOW2	CFG_2	SHADOW1	CFG_1	SHADOW0	CFG_0

Bits	Name	Description	R/W	Default
31	SHADOW7	Shadow Mode: 0: set mpwm_t7_shadow directly when writing to mpwm_t7 1: update mpwm_t7_shadow from mpwm_t7 at end of period	R/W	0x1
30:28	CFG_7	Configuration for PWM Pin7: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
27	SHADOW6	Shadow Mode: 0: set mpwm_t6_shadow directly when writing to mpwm_t6 1: update mpwm_t6_shadow from mpwm_t6 at end of period	R/W	0x1
26:24	CFG_6	Configuration for PWM Pin6: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
23	SHADOW5	Shadow Mode: 0: set mpwm_t5_shadow directly when writing to mpwm_t5 1: update mpwm_t5_shadow from mpwm_t5 at end of period	R/W	0x1
22:20	CFG_5	Configuration for PWM Pin5: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
19	SHADOW4	Shadow Mode: 0: set mpwm_t4_shadow directly when writing to mpwm_t4 1: update mpwm_t4_shadow from mpwm_t4 at end of period	R/W	0x1
18:16	CFG_4	Configuration for PWM Pin4: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4

Bits	Name	Description	R/W	Default
15	SHADOW3	Shadow Mode: 0: set mpwm_t3_shadow directly when writing to mpwm_t3 1: update mpwm_t3_shadow from mpwm_t3 at end of period	R/W	0x1
14:12	CFG_3	Configuration for PWM Pin3: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
11	SHADOW2	Shadow Mode: 0: set mpwm_t2_shadow directly when writing to mpwm_t2 1: update mpwm_t2_shadow from mpwm_t2 at end of period	R/W	0x1
10:8	CFG_2	Configuration for PWM Pin2: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
7	SHADOW1	Shadow Mode: 0: set mpwm_t1_shadow directly when writing to mpwm_t1 1: update mpwm_t1_shadow from mpwm_t1 at end of period	R/W	0x1
6:4	CFG_1	Configuration for PWM Pin1: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4
3	SHADOW0	Shadow Mode: 0: set mpwm_t0_shadow directly when writing to mpwm_t0 1: update mpwm_t0_shadow from mpwm_t0 at end of period	R/W	0x1
2:0	CFG_0	Configuration for PWM Pin0: 000: set output to 1 001: set output to 0 010: set output to PWM signal referenced on counter0 011: set output to inverted PWM signal referenced on counter0 100: set output to High-Z 110: set output to PWM signal referenced on counter1 111: set output to inverted PWM signal referenced on counter1	R/W	0x4

MPWM_CONFIG_FAILURE – Failure Config Register 0x10140508

MPWM unit has a failure input pin, which is configured here.

31–18	17	16	15 14	13 12	11 10	9 8	7 6	5 4	3 2	1 0
reserved	FAILURE	ENABLE	CFG_7	CFG_6	CFG_5	CFG_4	CFG_3	CFG_2	CFG_1	CFG_0

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x0
17	FAILURE	To set failure mode by software. Setting this bit generates the irq_raw-failure. To reset, first reset this bit, then clear failure interrupt.	R/W	0x0
16	ENABLE	Failure input enable: 0: failure input is disabled (ignored) 1: failure input sets failure bit, which sets all PWM pins to values defined in this register	R/W	0x0
15:14	CFG_7	Configuration for PWM Pin7 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in mpwm_config_pins	R/W	0x0
13:12	CFG_6	Configuration for PWM Pin6 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in mpwm_config_pins	R/W	0x0
11:10	CFG_5	Configuration for PWM Pin5 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in pwm_config_pins	R/W	0x0
9:8	CFG_4	Configuration for PWM Pin4 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in pwm_config_pins	R/W	0x0
7:6	CFG_3	Configuration for PWM Pin3 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in pwm_config_pins	R/W	0x0
5:4	CFG_2	Configuration for PWM Pin2 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in pwm_config_pins	R/W	0x0
3:2	CFG_1	Configuration for PWM Pin1 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in pwm_config_pins	R/W	0x0
1:0	CFG_0	Configuration for PWM Pin0 in case of failure: 00: set output to 0 01: set output to 1 10: set output to High-Z 11: leave output as defined in mpwm_config_pins	R/W	0x0

MPWM_IRQ_CONFIG – IRQ Config Register 0x1014050c

This register configures sources and delay values of 2 interrupts.

31:27	26:16	15:11	10:0
NR1	DELAY1	NR0	DELAY0

Bits	Name	Description	R/W	Default
31:27	NR1	select PWM pin for interrupt 1 00000: disable interrupt 00001: generate interrupt at pwm_cnt0 == 0 00010: generate interrupt at pwm_cnt1 == 0 00011: generate interrupt at pwm_cnt0 == max 00100: generate interrupt at pwm_cnt1 == max 10000: generate interrupt at posedge of PWM pin 0 ... ... 10111: generate interrupt at posedge of PWM pin 7 11000: generate interrupt at negedge of PWM pin 0 ... ... 11111: generate interrupt at negedge of PWM pin 7	R/W	0x0
26:16	DELAY1	Delay from event to generation of interrupt 1 in steps of 20ns	R/W	0x0
15:11	NR0	select PWM pin for interrupt 0 00000: disable interrupt 00001: generate interrupt at pwm_cnt0 == 0 00010: generate interrupt at pwm_cnt1 == 0 00011: generate interrupt at pwm_cnt0 == max 00100: generate interrupt at pwm_cnt1 == max 10000: generate interrupt at posedge of PWM pin 0 ... ... 10111: generate interrupt at posedge of PWM pin 7 11000: generate interrupt at negedge of PWM pin 0 ... ... 11111: generate interrupt at negedge of PWM pin 7	R/W	0x0
10:0	DELAY0	Delay from event to generation of interrupt 0 in steps of 20ns	R/W	0x0

MPWM_IRQ_RAW – Raw IRQ Register 0x10140510

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with ’1’ resets the appropriate IRQ.
Write access with ’0’ does not influence this bit.

31:3	2	1	0
reserved	FAILURE	INTERRUPT1	INTERRUPT0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x0
2	FAILURE	interrupt if failure input is active	R/W	0x0
1	INTERRUPT1	interrupt 1 as defined in mpwm_irq_config	R/W	0x0
0	INTERRUPT0	interrupt 0 as defined in mpwm_irq_config	R/W	0x0

MPWM_IRQ_MASKED – Masked IRQ Reigster 0x10140514

Show status of masked IRQs (as connected to ARM/xPIC).

31 … 3	2	1	0
reserved	FAILURE	INTERRUPT1	INTERRUPT0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x0
2	FAILURE	interrupt if failure input is active	R	0x0
1	INTERRUPT1	interrupt 0 as defined in mpwm_irq_config	R	0x0
0	INTERRUPT0	interrupt 0 as defined in mpwm_irq_config	R	0x0

MPWM_IRQ_MSK_SET – IRQ Enable Mask Register 0x10140518

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 MPWM_IRQ_RAW.

31 … 3	2	1	0
reserved	FAILURE	INTERRUPT1	INTERRUPT0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x0
2	FAILURE	interrupt if failure input is active	R/W	0x0
1	INTERRUPT1	interrupt 1 as defined in mpwm_irq_config	R/W	0x0
0	INTERRUPT0	interrupt 0 as defined in mpwm_irq_config	R/W	0x0

MPWM_IRQ_MSK_RESET – IRQ Disable Mask Register 0x1014051c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
Write access with ’0’ does not influence this bit.
Read access shows actual interrupt mask.

31 … 3	2	1	0
reserved	FAILURE	INTERRUPT1	INTERRUPT0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x0
2	FAILURE	interrupt if failure input is active	R/W	0x0
1	INTERRUPT1	interrupt 1 as defined in mpwm_irq_config	R/W	0x0
0	INTERRUPT0	interrupt 0 as defined in mpwm_irq_config	R/W	0x0

MPWM_CNT0_PERIOD – Counter 0 Period Register 0x10140520
MPWM_CNT1_PERIOD – Counter 1 Period Register 0x10140530

This register holds the counter period in steps of 2.5ns (400MHz basis). Depending on mode, 2 or 3 lower bits are ignored.

31 … 20	19 … 2	1 … 0
reserved	VAL	reserved

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:2	VAL	PWM period of counter: Sawtooth mode: period length in steps of 10ns = cnt(n)_period[19:2] + 1 Triangle mode: period length in steps of 20ns = cnt(n)_period[19:3]	R/W	0x0
1:0	reserved	-	R	0x0

MPWM_CNT0 – Counter 0 Value 0x10140524
MPWM_CNT1 – Counter 1 Value 0x10140534

The counter value is shifted to be displayed in 2.5ns resolution (400MHz basis), as period and threshold are.

31 … 20	19 … 2	1 0
reserved	VAL	reserved

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:2	VAL	actual value of PWM counter	R	0x0
1:0	reserved	-	R	0x0

MPWM_CNT0_SYSTIME – Counter 0 Start Systime 0x10140528
MPWM_CNT1_SYSTIME – Counter 1 Start Systime 0x10140538

MPWM_CNT(n)_SYSTIME register contains captured systime at start point of counter period.
Systime will always be captured to this register, if MPWM_CONFIG_COUNTER-run_cnt(n)=1 and MPWM_CNT(n)=0.

31 … 20	19 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:0	VAL	Captured lower bits of systime at cnt(n)==0	R	0x0

MPWM_CNT0_WATCHDOG – Counter 0 Watchdog 0x1014052c
MPWM_CNT1_WATCHDOG – Counter 1 Watchdog 0x1014053c

The watchdog counter will decrease with every zero-crossing of PWM counter.
If the watchdog counter reaches 0, the MPWM module will go to failure state.

31 … 4	3 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3:0	VAL	Watchdog counter value: Set any value != 0 to retrigger. Set to 0 to disable counter watchdog.	R/W	0x0

MPWM_T0 – PWM Channel 0 Threshold 0x10140540
MPWM_T1 – PWM channel 1 Threshold 0x10140544
MPWM_T2 – PWM channel 2 Threshold 0x10140548
MPWM_T3 – PWM channel 3 Threshold 0x1014054c
MPWM_T4 – PWM channel 4 Threshold 0x10140550
MPWM_T5 – PWM channel 5 Threshold 0x10140554
MPWM_T6 – PWM channel 6 Threshold 0x10140558
MPWM_T7 – PWM channel 7 Threshold 0x1014055c

A threshold value does not exactly describe the PWM output behaviour, as it depends on mode (triangle or sawtooth) and other symmetry and accuracy factors.

To better describe the exact behaviour of PWM outputs, we use the low phase width, which can easily be set in relation with counter period.

The hardware will automatically choose the exact threshold compare values from the programmed low phase width.

31:20	19:0
reserved	VAL

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:0	VAL	Width of channel(0-7) low phase in steps of 2,5ns	R/W	0xfffff

MPWM_T0_SHADOW – PWM channel 0 Threshold Shadow 0x10140560
MPWM_T1_SHADOW – PWM channel 1 Threshold Shadow 0x10140564
MPWM_T2_SHADOW – PWM channel 2 Threshold Shadow 0x10140568
MPWM_T3_SHADOW – PWM channel 3 Threshold Shadow 0x1014056c
MPWM_T4_SHADOW – PWM channel 4 Threshold Shadow 0x10140570
MPWM_T5_SHADOW – PWM channel 5 Threshold Shadow 0x10140574
MPWM_T6_SHADOW – PWM channel 6 Threshold Shadow 0x10140578
MPWM_T7_SHADOW – PWM channel 7 Threshold Shadow 0x1014057c

In shadow mode (MPWM_CONFIG_PINS-shadow(n)) theshold value will be updated from MPWM_T(n) at end of period.

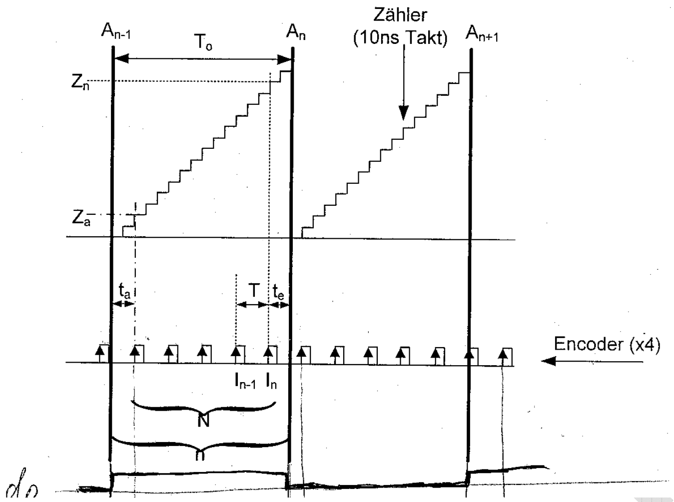
These registers show the actually used threshold (length of low phase) value.

31:20	19:0
reserved	VAL

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:0	VAL	Width of channel(0-7) low phase in steps of 2,5ns	R	0x0

7.2 Motion Encoder Unit

The netX10 includes two identical Motion Encoder (MENC) units. Beside the general functionality to count the pulses of incremental encoders, the unit supports advanced time capture functions (4 different capture registers) and interrupt generation on 16 different encoder events. The sampling of the encoder values can simultaneously capture the current System Time for synchronization purposes. A special Ta/Te mode allows very precise determination of rotary speeds especially at low speeds.



The following table is a summary of MENC related registers.

ARM Address	Register Name	Short Description
0x10140580	MENC_CONFIG	Encoder Configuration Register
0x10140584	MENC_ENC0_POSITION	Position of Encoder 0
0x10140588	MENC_ENC1_POSITION	Position of Encoder 1
0x1014058c	MENC_CAPTURE_NOW	Capture Now Register
0x10140590	MENC_CAPTURE0_CONFIG	Capture Unit 0 Configuration Register
0x10140594	MENC_CAPTURE0_VAL	Capture Unit 0 Captured Value
0x10140598	MENC_CAPTURE0_TA	Capture Unit 0 Ta
0x1014059c	MENC_CAPTURE0_TE	Capture Unit 0 Te
0x101405a0	MENC_CAPTURE1_CONFIG	Capture Unit 1 Configuration Register
0x101405a4	MENC_CAPTURE1_VAL	Capture Unit 1 Captured Value
0x101405a8	MENC_CAPTURE1_TA	Capture Unit 1 Ta
0x101405ac	MENC_CAPTURE1_TE	Capture Unit 1 Te
0x101405b0	MENC_CAPTURE2_CONFIG	Capture Unit 2 Configuration Register
0x101405b4	MENC_CAPTURE2_VAL	Capture Unit 2 Captured Value
0x101405b8	MENC_CAPTURE2_TA	Capture Unit 2 Ta
0x101405bc	MENC_CAPTURE2_TE	Capture Unit 2 Te
0x101405c0	MENC_CAPTURE3_CONFIG	Capture Unit 3 Configuration Register
0x101405c4	MENC_CAPTURE3_VAL	Capture Unit 3 Captured Value
0x101405c8	MENC_CAPTURE3_TA	Capture Unit 3 Ta
0x101405cc	MENC_CAPTURE3_TE	Capture Unit 3 Te
0x101405d0	MENC_STATUS	Position and Capture Status
0x101405d4	MENC_IRQ_MASKED	Masked IRQ Register
0x101405d8	MENC_IRQ_MSK_SET	IRQ Mask Enable
0x101405dc	MENC_IRQ_MSK_RESET	IRQ Mask Disable

MENC_CONFIG – Encoder Configuration Register 0x10140580

31	30	29	28	27	26	25	24	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				MP1_FILTER_SAMPLE_RATE			MP1_EN	reserved				MP0_FILTER_SAMPLE_RATE			MP0_EN	reserved			ENC1_COUNT_DIR	ENC1_FILTER_SAMPLE_RATE			ENC1_EN	reserved			ENC0_COUNT_DIR	ENC0_FILTER_SAMPLE_RATE			ENC0_EN

Bits	Name	Description	R/W	Default
31:28	reserved	-	R	0x0
27:25	MP1_FILTER_SAMPLE_RATE	Filter sample rate for mp1 signal: 0: none - Filter is disabled. 1: 10 ns - pulses < 10ns will be blocked, pulses > 20ns will pass. 2: 20 ns - pulses < 20ns will be blocked, pulses > 40ns will pass. 3: 50 ns - pulses < 50ns will be blocked, pulses > 100ns will pass. 4: 100 ns - pulses < 100ns will be blocked, pulses > 200ns will pass. 5: 200 ns - pulses < 200ns will be blocked, pulses > 400ns will pass. 6: 500 ns - pulses < 500ns will be blocked, pulses > 1us will pass. 7: 1 us - pulses < 1us will be blocked, pulses > 2us will pass.	R/W	0x0
24	MP1_EN	mp1 enable: 0: Disable interrupts based on mp1 signal.	R/W	0x0
23:20	reserved	-	R	0x0
19:17	MP0_FILTER_SAMPLE_RATE	Filter sample rate for mp0 signal: 0: none - Filter is disabled. 1: 10 ns - pulses < 10ns will be blocked, pulses > 20ns will pass. 2: 20 ns - pulses < 20ns will be blocked, pulses > 40ns will pass. 3: 50 ns - pulses < 50ns will be blocked, pulses > 100ns will pass. 4: 100 ns - pulses < 100ns will be blocked, pulses > 200ns will pass. 5: 200 ns - pulses < 200ns will be blocked, pulses > 400ns will pass. 6: 500 ns - pulses < 500ns will be blocked, pulses > 1us will pass. 7: 1 us - pulses < 1us will be blocked, pulses > 2us will pass.	R/W	0x0
16	MP0_EN	mp0 enable: 0: Disable interrupts based on mp0 signal.	R/W	0x0
15:13	reserved	-	R	0x0
12	ENC1_COUNT_DIR	Encoder1 count direction: 0: standard 1: inverted	R/W	0x0

Bits	Name	Description	R/W	Default
11:9	ENC1_FILTER_SAMPLE_RATE	Encoder1 filter sample rate: 0: none - Filter is disabled. 1: 10 ns - pulses < 10ns will be blocked, pulses > 20ns will pass. 2: 20 ns - pulses < 20ns will be blocked, pulses > 40ns will pass. 3: 50 ns - pulses < 50ns will be blocked, pulses > 100ns will pass. 4: 100 ns - pulses < 100ns will be blocked, pulses > 200ns will pass. 5: 200 ns - pulses < 200ns will be blocked, pulses > 400ns will pass. 6: 500 ns - pulses < 500ns will be blocked, pulses > 1us will pass. 7: 1 us - pulses < 1us will be blocked, pulses > 2us will pass.	R/W	0x0
8	ENC1_EN	Encoder1 enable: 0: Disable interrupts based on encoder1 signals.	R/W	0x0
7:5	reserved	-	R	0x0
4	ENC0_COUNT_DIR	Encoder0 count direction: 0: standard 1: inverted	R/W	0x0
3:1	ENC0_FILTER_SAMPLE_RATE	Encoder0 filter sample rate: 0: none - Filter is disabled. 1: 10 ns - pulses < 10ns will be blocked, pulses > 20ns will pass. 2: 20 ns - pulses < 20ns will be blocked, pulses > 40ns will pass. 3: 50 ns - pulses < 50ns will be blocked, pulses > 100ns will pass. 4: 100 ns - pulses < 100ns will be blocked, pulses > 200ns will pass. 5: 200 ns - pulses < 200ns will be blocked, pulses > 400ns will pass. 6: 500 ns - pulses < 500ns will be blocked, pulses > 1us will pass. 7: 1 us - pulses < 1us will be blocked, pulses > 2us will pass.	R/W	0x0
0	ENC0_EN	Encoder0 enable: 0: Disable interrupts based on encoder0 signals.	R/W	0x0

MENC_ENC0_POSITION – Position of Encoder 0 0x10140584
MENC_ENC1_POSITION – Position of Encoder 1 0x10140588

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Actual position of encoder 0/1. This register is writable but can also be changed by hardware.	R/W	0x0

MENC_CAPTURE_NOW – Capture Now Register 0x1014058c

This register allows activating the capture event by software for all 4 capture units.

31 … 4	3	2	1	0
reserved	CAP3_NOW	CAP2_NOW	CAP1_NOW	CAP0_NOW

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	CAP3_NOW	Capture menc_capture3 now (by SW). Capture by writing 1 to this register, reset automatically.	R/W	0x0
2	CAP2_NOW	Capture menc_capture2 now (by SW). Capture by writing 1 to this register, reset automatically.	R/W	0x0
1	CAP1_NOW	Capture menc_capture1 now (by SW). Capture by writing 1 to this register, reset automatically.	R/W	0x0
0	CAP0_NOW	Capture menc_capture0 now (by SW). Capture by writing 1 to this register, reset automatically.	R/W	0x0

MENC_CAPTURE0_CONFIG – Capture Unit 0 Configuration Register 0x10140590
MENC_CAPTURE1_CONFIG – Capture Unit 1 Configuration Register 0x101405a0
MENC_CAPTURE2_CONFIG – Capture Unit 2 Configuration Register 0x101405b0
MENC_CAPTURE3_CONFIG – Capture Unit 3 Configuration Register 0x101405c0

31:11	10	9	8:6	5:1	0
reserved	CONCE	SRC_NR	SRC	SEL	INV

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x0
10	CONCE	Capture once: 0: continuous capture: each event overwrites old capture register 1: capture once: capture only, if menc_status.cap(0-3) = 0	R/W	0x0
9	SRC_NR	Capture source channel: 0: encoder/channel 0 1: encoder/channel 1	R/W	0x0
8:6	SRC	Capture source (what to capture): 0: system time ns (independent of src_nr) 1: position channel 0/1 2: Ta of channel 0/1 3: Te of channel 0/1 4: Ta+Te of channel 0/1 5: period in clock cycles (independent of src_nr)	R/W	0x0
5:1	SEL	Capture start signal: 0 : off (no capture) 1 : enc0_n 2 : enc1_n 3 : enc0_edge (independent of inv) 4 : enc1_edge (independent of inv) 5 : mp0 6 : mp1 7 : pwm_cnt0_min (independent of inv) 8 : pwm_cnt0_max (independent of inv) 9 : pwm_cnt1_min (independent of inv) 10: pwm_cnt1_max (independent of inv) 11: pwm_t0 12: pwm_t1 13: pwm_t2 14: pwm_t3 15: pwm_t4 16: pwm_t5 17: pwm_t6 18: pwm_t7 19: xpic_timer0 (independent of inv) 20: xpic_timer1 (independent of inv) 21: xpic_timer2 (independent of inv) 22: arm_timer0 (independent of inv) 23: arm_timer1 (independent of inv)	R/W	0x0
0	INV	Invert capture start signal: 0: positive edge 1: negative edge	R/W	0x0

MENC_CAPTURE0_VAL – Capture unit 0 Captured Value 0x10140594
MENC_CAPTURE1_VAL – Capture Unit 1 Captured Value 0x101405a4
MENC_CAPTURE2_VAL – Capture Unit 2 Captured Value 0x101405b4
MENC_CAPTURE3_VAL – Capture Unit 3 Captured Value 0x101405c4

31:0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Captured value	R	0x0

MENC_CAPTURE0_TA – Capture Unit 0 Ta 0x10140598
MENC_CAPTURE1_TA – Capture Unit 1 Ta 0x101405a8
MENC_CAPTURE2_TA – Capture Unit 2 Ta 0x101405b8
MENC_CAPTURE3_TA – Capture Unit 3 Ta 0x101405c8

This register is only used for debug purposes.

31:20	19:0
reserved	VAL

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:0	VAL	Actual Ta: Time before first encoder pulse in period.	R	0x0

MENC_CAPTURE0_TE – Capture Unit 0 Te 0x1014059c
MENC_CAPTURE1_TE – Capture Unit 1 Te 0x101405ac
MENC_CAPTURE2_TE – Capture Unit 2 Te 0x101405bc
MENC_CAPTURE3_TE – Capture Unit 3 Te 0x101405cc

This register is only used for debug purposes.

31:20	19:0
reserved	VAL

Bits	Name	Description	R/W	Default
31:20	reserved	-	R	0x0
19:0	VAL	Actual Te: Time after last encoder pulse in period.	R	0x0

MENC_STATUS – Position and Capture Status **0x101405d0**

This register includes all raw IRQs and encoder direction.
Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with '1' resets the appropriate IRQ (except enc*_dir_ro).
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
reserved						MP1	MP0	reserved				CAP3	CAP2	CAP1	CAP0	ENC1_DIR_RO	reserved		ENC1_N	ENC1_PHASE_ERROR	ENC1_OVFL_NEG	ENC1_OVFL_POS	ENC1_EDGE	ENC0_DIR_RO	reserved		ENC0_N	ENC0_PHASE_ERROR	ENC0_OVFL_NEG	ENC0_OVFL_POS	ENC0_EDGE

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	MP1	Rising edge at Measurement Point 1	R/W	0x0
24	MP0	Rising edge at Measurement Point 0	R/W	0x0
23:20	reserved	-	R	0x0
19	CAP3	Captured register 3	R/W	0x0
18	CAP2	Captured register 2	R/W	0x0
17	CAP1	Captured register 1	R/W	0x0
16	CAP0	Captured register 0	R/W	0x0
15	ENC1_DIR_RO	Encoder1 direction (read only)	R	0x0
14:13	reserved	-	R	0x0
12	ENC1_N	Rising edge at input enc1_n.	R/W	0x0
11	ENC1_PHASE_ERROR	Phase error at encoder 1: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
10	ENC1_OVFL_NEG	Encoder1 overflow negative	R/W	0x0
9	ENC1_OVFL_POS	Encoder1 overflow positive	R/W	0x0
8	ENC1_EDGE	Edge at Encoder 1 occurred (rising or falling of enc1_a or enc1_b)	R/W	0x0
7	ENC0_DIR_RO	Encoder0 direction (read only)	R	0x0
6:5	reserved	-	R	0x0
4	ENC0_N	Rising edge at input enc0_n.	R/W	0x0
3	ENC0_PHASE_ERROR	Phase error at encoder 0: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
2	ENC0_OVFL_NEG	Encoder0 overflow negative	R/W	0x0
1	ENC0_OVFL_POS	Encoder0 overflow positive	R/W	0x0
0	ENC0_EDGE	Edge at Encoder 0 occurred (rising or falling of enc0_a or enc0_b)	R/W	0x0

MENC_IRQ_MASKED – Masked IRQ Register 0x101405d4

Show status of masked IRQs (as connected to ARM/xPIC).

31:26	25	24	23:20	19	18	17	16	15:13	12	11	10	9	8	7:5	4	3	2	1	0
reserved	MP1	MP0	reserved	CAP3	CAP2	CAP1	CAP0	reserved	ENC1_N	ENC1_PHASE_ERROR	ENC1_OVFL_NEG	ENC1_OVFL_POS	ENC1_EDGE	reserved	ENC0_N	ENC0_PHASE_ERROR	ENC0_OVFL_NEG	ENC0_OVFL_POS	ENC0_EDGE

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	MP1	Rising edge at Measurement Point 1	R	0x0
24	MP0	Rising edge at Measurement Point 0	R	0x0
23:20	reserved	-	R	0x0
19	CAP3	Captured register 3	R	0x0
18	CAP2	Captured register 2	R	0x0
17	CAP1	Captured register 1	R	0x0
16	CAP0	Captured register 0	R	0x0
15:13	reserved	-	R	0x0
12	ENC1_N	Rising edge at input enc1_n.	R	0x0
11	ENC1_PHASE_ERROR	Phase error at encoder 1: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R	0x0
10	ENC1_OVFL_NEG	Encoder1 overflow negative	R	0x0
9	ENC1_OVFL_POS	Encoder1 overflow positive	R	0x0
8	ENC1_EDGE	Edge at Encoder 1 occurred (rising or falling of enc1_a or enc1_b)	R	0x0
7:5	reserved	-	R	0x0
4	ENC0_N	Rising edge at input enc0_n.	R	0x0
3	ENC0_PHASE_ERROR	Phase error at encoder 0: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R	0x0
2	ENC0_OVFL_NEG	Encoder0 overflow negative	R	0x0
1	ENC0_OVFL_POS	Encoder0 overflow positive	R	0x0
0	ENC0_EDGE	Edge at Encoder 0 occurred (rising or falling of enc0_a or enc0_b)	R	0x0

MENC_IRQ_MSK_SET – IRQ Mask Enable 0x101405d8

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 MENC_STATUS.

31:26	25	24	23:20	19	18	17	16	15:13	12	11	10	9	8	7:5	4	3	2	1	0
reserved	MP1	MP0	reserved	CAP3	CAP2	CAP1	CAP0	reserved	ENC1_N	ENC1_PHASE_ERROR	ENC1_OVFL_NEG	ENC1_OVFL_POS	ENC1_EDGE	reserved	ENC0_N	ENC0_PHASE_ERROR	ENC0_OVFL_NEG	ENC0_OVFL_POS	ENC0_EDGE

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	MP1	Rising edge at Measurement Point 1	R/W	0x0
24	MP0	Rising edge at Measurement Point 0	R/W	0x0
23:20	reserved	-	R	0x0
19	CAP3	Captured register 3	R/W	0x0
18	CAP2	Captured register 2	R/W	0x0
17	CAP1	Captured register 1	R/W	0x0
16	CAP0	Captured register 0	R/W	0x0
15:13	reserved	-	R	0x0
12	ENC1_N	Rising edge at input enc1_n.	R/W	0x0
11	ENC1_PHASE_ERROR	Phase error at encoder 1: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
10	ENC1_OVFL_NEG	Encoder1 overflow negative	R/W	0x0
9	ENC1_OVFL_POS	Encoder1 overflow positive	R/W	0x0
8	ENC1_EDGE	Edge at Encoder 1 occurred (rising or falling of enc1_a or enc1_b)	R/W	0x0
7:5	reserved	-	R	0x0
4	ENC0_N	Rising edge at input enc0_n.	R/W	0x0
3	ENC0_PHASE_ERROR	Phase error at encoder 0: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
2	ENC0_OVFL_NEG	Encoder0 overflow negative	R/W	0x0
1	ENC0_OVFL_POS	Encoder0 overflow positive	R/W	0x0
0	ENC0_EDGE	Edge at Encoder 0 occurred (rising or falling of enc0_a or enc0_b)	R/W	0x0

MENC_IRQ_MSK_RESET – IRQ Mask Disable 0x101405dc

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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						MP1	MP0	reserved				CAP3	CAP2	CAP1	CAP0	reserved			ENC1_N	ENC1_PHASE_ERROR	ENC1_OVFL_NEG	ENC1_OVFL_POS	ENC1_EDGE	reserved			ENC0_N	ENC0_PHASE_ERROR	ENC0_OVFL_NEG	ENC0_OVFL_POS	ENC0_EDGE

Bits	Name	Description	R/W	Default
31:26	reserved	-	R	0x0
25	MP1	Rising edge at Measurement Point 1	R/W	0x0
24	MP0	Rising edge at Measurement Point 0	R/W	0x0
23:20	reserved	-	R	0x0
19	CAP3	Captured register 3	R/W	0x0
18	CAP2	Captured register 2	R/W	0x0
17	CAP1	Captured register 1	R/W	0x0
16	CAP0	Captured register 0	R/W	0x0
15:13	reserved	-	R	0x0
12	ENC1_N	Rising edge at input enc1_n.	R/W	0x0
11	ENC1_PHASE_ERROR	Phase error at encoder 1: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
10	ENC1_OVFL_NEG	Encoder1 overflow negative	R/W	0x0
9	ENC1_OVFL_POS	Encoder1 overflow positive	R/W	0x0
8	ENC1_EDGE	Edge at Encoder 1 occurred (rising or falling of enc1_a or enc1_b)	R/W	0x0
7:5	reserved	-	R	0x0
4	ENC0_N	Rising edge at input enc0_n.	R/W	0x0
3	ENC0_PHASE_ERROR	Phase error at encoder 0: Encoder inputs changed 2 phases in 1 cycle, which leads to unknown position.	R/W	0x0
2	ENC0_OVFL_NEG	Encoder0 overflow negative	R/W	0x0
1	ENC0_OVFL_POS	Encoder0 overflow positive	R/W	0x0
0	ENC0_EDGE	Edge at Encoder 0 occurred (rising or falling of enc0_a or enc0_b)	R/W	0x0

7.3 ADC Unit

The netX10 includes two 10-bit AD Converters, each with an analog 8 input multiplexer. Two input channels can be sampled simultaneously at a rate of up to 1 MS/second. The channel selection for each following conversion can be programmed by a sequencer unit. The start of conversion can be precisely synchronized with other netX units (e.g. to MPWM unit for motion control applications). Completed conversion or ADC ready for start of next conversion events can generate an interrupt to ARM or xPIC. A special pipeline mode allows continuous sampling of analog values from different channels, as the next channel can be programmed for sampling during the current conversion.

The following table is a summary of ADC_CTRL registers.

ARM Address	Register Name	Short Description
0x101406c0	ADC_CTRL_START	ADC Start Register
0x101406c4	ADC_CTRL_AUTOSAMPLE_CONFIG0	ADC0 Config Register for Autosample Mode
0x101406c8	ADC_CTRL_AUTOSAMPLE_CONFIG1	ADC1 Config Register for Autosample Mode
0x101406cc	ADC_CTRL_MANSAMPLE_CONFIG0	ADC0 Config Register for Direct Control
0x101406d0	ADC_CTRL_MANSAMPLE_CONFIG1	ADC1 Config Register for Direct Control
0x101406d4	ADC_CTRL_STATUS	ADC Status Register
0x101406d8	ADC_CTRL_ADC0_VAL	ADC0 Value
0x101406dc	ADC_CTRL_ADC1_VAL	ADC1 Value
0x101406e0	ADC_CTRL_IRQ_RAW	Raw IRQ
0x101406e4	ADC_CTRL_IRQ_MASKED	Masked IRQ
0x101406e8	ADC_CTRL_IRQ_MSK_SET	IRQ Mask Enable
0x101406ec	ADC_CTRL_IRQ_MSK_RESET	IRQ Mask Disable

ADC_CTRL_START – ADC Start Register 0x101406c0

This register is writable but can also be changed by hardware (reset).

31 … 2	1	0
reserved	START_ADC1	START_ADC0

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	START_ADC1	Start ADC1: Setting this bit to 1 starts ADC control state machine for ADC1 directly after current AD-conversion is finished. When new AD-conversion was started, this bit is automatically reset and register adc_ctrl_autosample_config1 can be written for next AD-conversion.	R/W	0x0
0	START_ADC0	Start ADC0: Setting this bit to 1 starts ADC control state machine for ADC0 directly after current AD-conversion is finished. When new AD-conversion was started, this bit is automatically reset and register adc_ctrl_autosample_config0 can be written for next AD-conversion.	R/W	0x0

ADC_CTRL_AUTOSAMPLE_CONFIG0 – ADC0 Config Register for Autosample Mode0x101406c4
ADC_CTRL_AUTOSAMPLE_CONFIG1 – ADC1 Config Register for Autosample Mode0x101406c8

31:10	9:5	4	3	2:0
reserved	EVENT_SEL	EVENT_INV	POWER_DOWN	SEL

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9:5	EVENT_SEL	Select for AD-conversion start signal: 0 : start immediately 1 : enc0_n 2 : enc1_n 3 : enc0_edge (independant of event_inv) 4 : enc1_edge (independant of event_inv) 5 : mp0 6 : mp1 7 : pwm_cnt0_min (independant of event_inv) 8 : pwm_cnt0_max (independant of event_inv) 9 : pwm_cnt1_min (independant of event_inv) 10: pwm_cnt1_max (independant of event_inv) 11: pwm_t0 12: pwm_t1 13: pwm_t2 14: pwm_t3 15: pwm_t4 16: pwm_t5 17: pwm_t6 18: pwm_t7 19: xpic_timer0 (independant of event_inv) 20: xpic_timer1 (independant of event_inv) 21: xpic_timer2 (independant of event_inv) 22: arm_timer0 (independant of event_inv) 23: arm_timer1 (independant of event_inv)	R/W	0x0
4	EVENT_INV	Invert AD-conversion start signal 0: positive edge 1: negative edge	R/W	0x0
3	POWER_DOWN	Power-down mode: 0: leave power of ADC active after sampling 1: change to Power-down after sampling	R/W	0x0
2:0	SEL	Select for analog multiplexer of ADC0/1: 000: Sample from analog pin AD0_IN0/ AD1_IN0 001: Sample from analog pin AD0_IN1/ AD1_IN1 010: Sample from analog pin AD0_IN2/ AD1_IN2 011: Sample from analog pin AD0_IN3/ AD1_IN3 100: Sample from analog pin AD0_IN4/ AD1_IN4 101: Sample from analog pin AD0_IN5/ AD1_IN5 110: Sample from analog pin AD0_IN6/ AD1_IN6 111: Sample from analog pin AD0_IN7/ AD1_IN7	R/W	0x0

ADC_CTRL_MANSAMPLE_CONFIG0 – ADC0 Config Register for Direct Control 0x101406cc
ADC_CTRL_MANSAMPLE_CONFIG1 – ADC1 Config Register for Direct Control 0x101406d0

This register is for debug purposes only!
It must not be written, when ADC autosample state machine is active (ADC_CTRL_IRQ_RAW-adc(0/1)_finish=0).
This register is writable but can also be changed by hardware.

31 … 5	4	3	2 … 0
reserved	CONV	PDB	SEL

Bits	Name	Description	R/W	Default
31:5	reserved	-	R	0x0
4	CONV	AD-conversion start pin: AD-conversion is started by setting this bit to 1 (with next edge of adcclk).	R/W	0x0
3	PDB	Power-down pin: 1: Operation mode 0: Power-down mode	R/W	0x0
2:0	SEL	Select for analog multiplexer of ADC0/1: 000: Sample from analog pin AD0_IN0/ AD1_IN0 001: Sample from analog pin AD0_IN1/ AD1_IN1 010: Sample from analog pin AD0_IN2/ AD1_IN2 011: Sample from analog pin AD0_IN3/ AD1_IN3 100: Sample from analog pin AD0_IN4/ AD1_IN4 101: Sample from analog pin AD0_IN5/ AD1_IN5 110: Sample from analog pin AD0_IN6/ AD1_IN6 111: Sample from analog pin AD0_IN7/ AD1_IN7	R/W	0x0

ADC_CTRL_STATUS – ADC Status Register 0x101406d4

31 … 5	4	3	2	1	0
reserved	ADCCLK	ADC1_FINISH	ADC0_FINISH	ADC1_EOCB	ADC0_EOCB

Bits	Name	Description	R/W	Default
31:5	reserved	-	R	0x0
4	ADCCLK	sampled adcclk (16MHz), used for both ADCs	R	0x0
3	ADC1_FINISH	ADC1 is finished (not sampling any data)	R	0x0
2	ADC0_FINISH	ADC0 is finished (not sampling any data)	R	0x0
1	ADC1_EOCB	EOCB signal of ADC1	R	0x0
0	ADC0_EOCB	EOCB signal of ADC0	R	0x0

ADC_CTRL_ADC0_VAL – ADC0 Value 0x101406d8
ADC_CTRL_ADC1_VAL – ADC1 Value 0x101406dc

31 … 10	9 … 0
reserved	VAL

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9:0	VAL	Sampled value, changed with posedge of irq_raw-adc(0/1)_finish	R	0x0

ADC_CTRL_IRQ_RAW – Raw IRQ 0x101406e0

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with ’1’ resets the appropriate IRQ.
Write access with ’0’ does not influence this bit.

31 … 4	3	2	1	0
reserved	START_ADC1_N	START_ADC0_N	ADC1_FINISH	ADC0_FINISH

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	START_ADC1_N	ADC1 start bit has returned to 0, i.e. ADC1 can be programmed for next conversion	R/W	0x0
2	START_ADC0_N	ADC0 start bit has returned to 0, i.e. ADC0 can be programmed for next conversion	R/W	0x0
1	ADC1_FINISH	ADC1 finished sampling	R/W	0x0
0	ADC0_FINISH	ADC0 finished sampling	R/W	0x0

ADC_CTRL_IRQ_MASKED – Masked IRQ 0x101406e4

Show status of masked IRQs (as connected to ARM/xPIC).

31 ... 4	3	2	1	0
reserved	START_ADC1_N	START_ADC0_N	ADC1_FINISH	ADC0_FINISH

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	START_ADC1_N	ADC1 start bit has returned to 0, i.e. ADC1 can be programmed for next conversion	R	0x0
2	START_ADC0_N	ADC0 start bit has returned to 0, i.e. ADC0 can be programmed for next conversion	R	0x0
1	ADC1_FINISH	ADC1 finished sampling	R	0x0
0	ADC0_FINISH	ADC0 finished sampling	R	0x0

ADC_CTRL_IRQ_MSK_SET – IRQ Mask Enable 0x101406e8

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with '1' sets interrupt mask bit.
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 ADC_CTRL_IRQ_RAW.

31 ... 4	3	2	1	0
reserved	START_ADC1_N	START_ADC0_N	ADC1_FINISH	ADC0_FINISH

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	START_ADC1_N	ADC1 start bit has returned to 0, i.e. ADC1 can be programmed for next conversion	R/W	0x0
2	START_ADC0_N	ADC0 start bit has returned to 0, i.e. ADC0 can be programmed for next conversion	R/W	0x0
1	ADC1_FINISH	ADC1 finished sampling	R/W	0x0
0	ADC0_FINISH	ADC0 finished sampling	R/W	0x0

ADC_CTRL_IRQ_MSK_RESET – IRQ Mask Disable 0x101406ec

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
Write access with ’0’ does not influence this bit.
Read access shows actual interrupt mask.

31 … 4	3	2	1	0
reserved	START_ADC1_N	START_ADC0_N	ADC1_FINISH	ADC0_FINISH

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	START_ADC1_N	ADC1 start bit has returned to 0, i.e. ADC1 can be programmed for next conversion	R/W	0x0
2	START_ADC0_N	ADC0 start bit has returned to 0, i.e. ADC0 can be programmed for next conversion	R/W	0x0
1	ADC1_FINISH	ADC1 finished sampling	R/W	0x0
0	ADC0_FINISH	ADC0 finished sampling	R/W	0x0

8 CORDIC Unit

To improve the performance of the data processing and control algorithms of ARM or xPIC CPUs, the netX10 is equipped with a special CORDIC (CoOrdinate Rotation Digital Computer) based hardware unit, which dramatically speeds up the calculation of transcendental functions such as sin, cos, sqrt, arctan. The precision of the unit is configurable and due to optimized hardware logic the 32bit precision values can be calculated within just 20 cycles. Depending on the function, a total calculation time of 400ns can be reached. The CORDIC unit supports rational, hyperbolic and linear mode while a dual table of coefficients, allows fast switching between the modes.

The following table is a summary of CORDIC related registers.

ARM Address	Register Name	Short Description
0x10140000	CORDIC_CTRL	CORDIC Control Register
0x10140004	CORDIC_X_REG	CORDIC Argument and Result Register X
0x10140008	CORDIC_Y_REG	CORDIC Argument and Result Register Y
0x1014000c	CORDIC_Z_REG	CORDIC Argument and Result Register Z
0x10140010	CORDIC_C_REG	CORDIC Argument Register C
0x10140014	CORDIC_FSM_STATE	CORDIC FSM State Register
0x10140018	CORDIC_LIN_39_TO_8	CORDIC Linear Coefficient Register
0x1014001c	CORDIC_LIN_7_TO_0	CORDIC Linear Coefficient Register
0x10140100	CORDIC_COEFF_RAM_START_CIRC_39_TO_8	Start of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^{-i}))
0x1014019c	CORDIC_COEFF_RAM_END_CIRC_39_TO_8	End of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^{-i}))
0x10140200	CORDIC_COEFF_RAM_START_HYP_39_TO_8	Start of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^{-1}))
0x1014029c	CORDIC_COEFF_RAM_END_HYP_39_TO_8	End of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^{-1}))
0x10140300	CORDIC_COEFF_RAM_START_CIRC_7_TO_0	Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^{-i}))
0x1014034c	CORDIC_COEFF_RAM_END_CIRC_7_TO_0	End of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^{-i}))
0x10140350	CORDIC_COEFF_RAM_START_HYP_7_TO_0	Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^{-1}))
0x1014039c	CORDIC_COEFF_RAM_END_HYP_7_TO_0	End of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^{-1}))

CORDIC_CTRL – CORDIC Control Register 0x10140000

This register controls the precision and the mode of operation. It is also used to start the CORDIC.

31 … 12	11 … 7	6 … 4	3 … 1	0
reserved	ITERATIONS	TARGET_AXIS	MODE	START

Bits	Name	Description	R/W	Default
31:12	reserved	-	R	0x0
11:7	ITERATIONS	Abort calculation after iteration k+8. When cordic_c_reg == 0 (no inverse functions being calculated) the CORDIC performs two iterations each clock cycle. Thus one can only adjust the number of iterations in steps of two (i.e. setting iterations to 0x1e will still result in 39 CORDIC iterations, like 0x1f would do). In 'inverse mode' (cordic_c_reg != 0) every iteration is executed twice. This of course results in twice as much iterations. Also now there is a difference between setting iterations to 0x1e or 0x1f. In the first case the CORDIC executes 76 (=2*38) iterations in the second one 78 (=2*39). 0x0: abort after iteration 8 0x1: abort after iteration 9 ... 0x1f: abort after iteration 39	R/W	0x1f
6:4	TARGET_AXIS	specifies which register (component of the vector) is driven towards the target value in cordic_c_reg. 001: drive X to C (if C!=0 inverse mode. Rotate until X=C.) 010: drive Y to C (if C=0: vectoring mode. Rotate until Y=0. if C!=0 inverse mode. Rotate until Y=C.) 100: drive Z to C (if C=0: rotation mode. Rotate until Z=0.)	R/W	0x4
3:1	MODE	Mode: 001: circular mode (cos, sin, ...) 010: linear mode (div, mul, ...) 100: hyperbolic mode (sinh, cosh, ...)	R/W	0x1
0	START	1: start calculation. This bit is reset to zero by the CORDIC once it finished calculation. This bit is writable but can also be changed by hardware. Reading the resolution registers while the CORDIC is running yields undefined values	R/W	0x0

CORDIC_X_REG – CORDIC Argument and Result Register X 0x10140004

31 … 0
ARGUMENT

Bits	Name	Description	R/W	Default
31:0	ARGUMENT	x-component of input vector	R/W	0x0

CORDIC_Y_REG – CORDIC Argument and Result Register Y 0x10140008

31 … 0
ARGUMENT

Bits	Name	Description	R/W	Default
31:0	ARGUMENT	y-component of input vector	R/W	0x0

CORDIC_Z_REG – CORDIC Argument and Result Register Z 0x1014000c

31 … 0
ARGUMENT

Bits	Name	Description	R/W	Default
31:0	ARGUMENT	input angle	R/W	0x0

CORDIC_C_REG – CORDIC Argument Register C 0x10140010

31 … 0
ARGUMENT

Bits	Name	Description	R/W	Default
31:0	ARGUMENT	target value to compare x/y/z value against	R/W	0x0

CORDIC_FSM_STATE – CORDIC FSM State Register 0x10140014

This register is readable for debugging purposes.

31 … 10	9	8	7	6	5 … 0
reserved	C_OVF	Z_OVF	Y_OVF	X_OVF	STATE

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9	X_OVF	C overflow	R	0x0
8	X_OVF	Z overflow	R	0x0
7	X_OVF	Y overflow	R	0x0
6	X_OVF	X overflow	R	0x0
5:0	STATE	state of FSM 0x0 - 0x1d: number of current iteration 0x1e: FINISHED (resetting ctrl_start and entereing state READY) 0x1f: READY (waiting for next calculation)	R	0x0

CORDIC_LIN_39_TO_8 – CORDIC Linear Coefficient Register 0x10140018

This is the number system scale register. This is used as a starting point for calculating the angles for the linear mode. It represents the number 1 in the utilized number system.
The default value is designed for use with the daggett angle representation in 40 bit fixed point ([-pi .. +pi] == [-2^{39} .. 2^{39}-1]).

31 … 0
L0_MSDW

Bits	Name	Description	R/W	Default
31:0	L0_MSDW	Most Significant DWord of first linear coefficient: coeff_linear_0[39:8] == 1[39:8].	R/W	0x28be60db

CORDIC_LIN_7_TO_0 – CORDIC Linear Coefficient Register 0x1014001c

This is the number system scale register. This is used as a starting point for calculating the angles for the linear mode. It represents the number 1 in the utilized number system.
The default value is designed for use with the daggett angle representation in 40 bit fixed point ([-pi .. +pi] == [-2^{39} .. 2^{39}-1]).

31 … 24	23 … 0
L0_LSB	reserved

Bits	Name	Description	R/W	Default
31:24	L0_LSB	Least Significant Byte of first linear coefficient: coeff_linear_0[7:0] == 1[7:0].	R/W	0x94
23:0	reserved	-	R	0x0

CORDIC_COEFF_RAM_START_CIRC_39_TO_8 – Start of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^(-i))) 0x10140100
…
…
CORDIC_COEFF_RAM_END_CIRC_39_TO_8 – End of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^(-i))) 0x1014019c

31 … 0
C0_MSDW … … C39_MSDW

Bits	Name	Description	R/W	Default
31:0	C(0-39)_MSDW	Address of Most Significant Double Word of circular coefficient: C0_MSDW: first circular coefficient coeff_circular_0[39:8] == arctan(2^{-0})[39:8]. … … C39_MSDW: last circular coefficient coeff_circular_39[39:8] == arctan(2^{-39})[39:8].	R/W	0x0

CORDIC_COEFF_RAM_START_HYP_39_TO_8 – Start of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^(-i))) 0x10140200
…
…
CORDIC_COEFF_RAM_END_HYP_39_TO_8 – End of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^(-i))) 0x1014029c

31 … 0
H0_MSDW … … H39_MSDW

Bits	Name	Description	R/W	Default
31:0	H(0-39)_MSDW	Address of Most Significant Double Word of hyperbolic coefficient: H0_MSDW: first hyperbolic coefficient coeff_hyperbolic_0[39:8] == arctanh(2^{-0})[39:8]. … … H39_MSDW: last hyperbolic coefficient coeff_hyperbolic_39[39:8] == arctanh(2^{-39})[39:8].	R/W	0x0

CORDIC_COEFF_RAM_START_CIRC_7_TO_0 – Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^(-i))) 0x10140300
…
…
CORDIC_COEFF_RAM_END_CIRC_7_TO_0 – End of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^(-i))) 0x1014034c

31 … 24	23 … 16	15 … 0
C0_LSB C2_LSB … … C38_LSB	C1_LSB C3_LSB … … C39_LSB	reserved

Bits	Name	Description	R/W	Default
31:24	C(0,2,…38)_LSB	Least Significant Byte of even circular coefficient: C0_LSB: first circular coefficient coeff_circular_0[7:0] == arctan(2^{-0})[7:0]. C2_LSB: coeff_circular_2[7:0] == arctan(2^{-2})[7:0]. … … C38_LSB: penultimate circular coefficient coeff_circular_38[7:0] == arctan(2^{-38})[7:0].	R/W	0x0
23:16	C(1,3,…39)_LSB	Least Significant Byte of odd circular coefficient: C1_LSB: second circular coefficient coeff_circular_1[7:0] == arctan(2^{-1})[7:0]. C3_LSB: coeff_circular_3[7:0] == arctan(2^{-3})[7:0]. … … C39_LSB: Last circular coefficient coeff_circular_39[7:0] == arctan(2^{-39})[7:0].	R/W	0x0
15:0	reserved	-	R	0x0

CORDIC_COEFF_RAM_START_HYP_7_TO_0 – Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^(-i))). 0x10140350
…
…
CORDIC_COEFF_RAM_END_HYP_7_TO_0 – End of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^(-i))). 0x1014039c

31–24	23–16	15–0
H0_LSB H2_LSB … … H38_LSB	H1_LSB H3_LSB … … H39_LSB	reserved

Bits	Name	Description	R/W	Default
31:24	H(0,2,…38)_LSB	Least Significant Byte of even hyperbolic coefficient: H0_LSB: first hyperbolic coefficient coeff_hyperbolic_0[7:0] == arctanh(2^{-0})[7:0]. H2_LSB: coeff_hyperbolic_2[7:0] == arctanh(2^{-2})[7:0]. … … H38_LSB: penultimate hyperbolic coefficient coeff_hyperbolic_38[7:0] == arctanh(2^{-38})[7:0].	R/W	0x0
23:16	H(1,3,…39)_LSB	Least Significant Byte of odd hyperbolic coefficient: H1_LSB: second hyperbolic coefficient coeff_hyperbolic_1[7:0] == arctanh(2^{-1})[7:0]. H3_LSB: coeff_hyperbolic_3[7:0] == arctanh(2^{-3})[7:0]. … … H39_LSB: last hyperbolic coefficient coeff_hyperbolic_39[7:0] == arctanh(2^{-39})[7:0].	R/W	0x0
15:0	reserved	-	R	0x0

9 Peripheral Functions

9.1 GPIOs – General Purpose IOs and Timers

The netX10 provides 8 general purpose IOs (GPIOs) and 2 timers. Each GPIO is associated with a 32 Bit register (GPIO_THRSH_CAPT) and can be configured to much functionality that are achievable with one register (e.g.: different capture and PWM modi). The following table shows a summary of all GPIO- and Timer related registers:

ARM Address	Register Name	Short Description
0x101c0800	GPIO_CFG0	GPIO 0 Configuration Register
0x101c0804	GPIO_CFG1	GPIO 1 Configuration Register
0x101c0808	GPIO_CFG2	GPIO 2 Configuration Register
0x101c080c	GPIO_CFG3	GPIO 3 Configuration Register
0x101c0810	GPIO_CFG4	GPIO 4 Configuration Register
0x101c0814	GPIO_CFG5	GPIO 5 Configuration Register
0x101c0818	GPIO_CFG6	GPIO 6 Configuration Register
0x101c081c	GPIO_CFG7	GPIO 7 Configuration Register
0x101c0820	GPIO_THRSH_CAPT0	GPIO 0 Threshold or Capture Register
0x101c0824	GPIO_THRSH_CAPT1	GPIO 1 Threshold or Capture Register
0x101c0828	GPIO_THRSH_CAPT2	GPIO 2 Threshold or Capture Register
0x101c082c	GPIO_THRSH_CAPT3	GPIO 3 Threshold or Capture Register
0x101c0830	GPIO_THRSH_CAPT4	GPIO 4 Threshold or Capture Register
0x101c0834	GPIO_THRSH_CAPT5	GPIO 5 Threshold or Capture Register
0x101c0838	GPIO_THRSH_CAPT6	GPIO 6 Threshold or Capture Register
0x101c083c	GPIO_THRSH_CAPT7	GPIO 7 Threshold or Capture Register
0x101c0840	GPIO_CNTR0_CTRL	GPIO Counter0 Control Register
0x101c0844	GPIO_CNTR1_CTRL	GPIO Counter1 Control Register
0x101c0848	GPIO_CNTR0_MAX	GPIO Counter0 Max Value
0x101c084c	GPIO_CNTR1_MAX	GPIO Counter1 Max Value
0x101c0850	GPIO_CNTR0_CNT	GPIO Counter0 Current Value
0x101c0854	GPIO_CNTR1_CNT	GPIO Counter1 Current Value
0x101c0858	GPIO_OUT	GPIO Output Register
0x101c085c	GPIO_IN	GPIO Input Register
0x101c0860	GPIO_IRQ_RAW	GPIO Raw IRQ Register
0x101c0864	GPIO_IRQ_MSK	GPIO Masked IRQ Register
0x101c0868	GPIO_IRQ_MSK_SET	GPIO Interrupt Mask Enable
0x101c086c	GPIO_IRQ_MSK_RESET	GPIO Interrupt Mask Disable
0x101c0870	CNTR_IRQ_RAW	GPIO Counter Raw IRQ Register
0x101c0874	GPIO_CNTR_IRQ_MSK	GPIO Counter Masked IRQ Register
0x101c0878	GPIO_CNTR_IRQ_MSK_SET	GPIO Counter Interrupt Mask Enable
0x101c087c	GPIO_CNTR_IRQ_MSK_RESET	GPIO Counter Interrupt Mask Disable
0x10140400	GPIO_CFG0	GPIO_MOTION 0 Configuration Register
0x10140404	GPIO_CFG1	GPIO_MOTION 1 Configuration Register
0x10140408	GPIO_CFG2	GPIO_MOTION 2 Configuration Register
0x1014040c	GPIO_CFG3	GPIO_MOTION 3 Configuration Register
0x10140410	GPIO_CFG4	GPIO_MOTION 4 Configuration Register
0x10140414	GPIO_CFG5	GPIO_MOTION 5 Configuration Register
0x10140418	GPIO_CFG6	GPIO_MOTION 6 Configuration Register

0x1014041c	GPIO_CFG7	GPIO_MOTION 7 Configuration Register
0x10140420	GPIO_THRSH_CAPT0	GPIO_MOTION 0 Threshold or Capture Register
0x10140424	GPIO_THRSH_CAPT1	GPIO_MOTION 1 Threshold or Capture Register
0x10140428	GPIO_THRSH_CAPT2	GPIO_MOTION 2 Threshold or Capture Register
0x1014042c	GPIO_THRSH_CAPT3	GPIO_MOTION 3 Threshold or Capture Register
0x10140430	GPIO_THRSH_CAPT4	GPIO_MOTION 4 Threshold or Capture Register
0x10140434	GPIO_THRSH_CAPT5	GPIO_MOTION 5 Threshold or Capture Register
0x10140438	GPIO_THRSH_CAPT6	GPIO_MOTION 6 Threshold or Capture Register
0x1014043c	GPIO_THRSH_CAPT7	GPIO_MOTION 7 Threshold or Capture Register
0x10140440	GPIO_CNTR0_CTRL	GPIO_MOTION Counter0 Control Register
0x10140444	GPIO_CNTR1_CTRL	GPIO_MOTION Counter1 Control Register
0x10140448	GPIO_CNTR0_MAX	GPIO_MOTION Counter0 Max Value
0x1014044c	GPIO_CNTR1_MAX	GPIO_MOTION Counter1 Max Value
0x10140450	GPIO_CNTR0_CNT	GPIO_MOTION Counter0 Current Value
0x10140454	GPIO_CNTR1_CNT	GPIO_MOTION Counter1 Current Value
0x10140458	GPIO_OUT	GPIO_MOTION Output Register
0x1014045c	GPIO_IN	GPIO_MOTION Input Register
0x10140460	GPIO_IRQ_RAW	GPIO_MOTION Raw IRQ Register
0x10140464	GPIO_MOTION _IRQ_MSK	GPIO_MOTION Masked IRQ Register
0x10140468	GPIO_MOTION _IRQ_MSK_SET	GPIO_MOTION Interrupt Mask Enable
0x1014046c	GPIO_MOTION _IRQ_MSK_RESET	GPIO_MOTION Interrupt Mask Disable
0x10140470	CNT_IRQ_RAW	GPIO_MOTION Counter Raw IRQ Register
0x10140474	GPIO_MOTION_CNTR_IRQ_MSK	GPIO_MOTION Counter Masked IRQ Register
0x10140478	GPIO_MOTION_CNTR_IRQ_MSK_SET	GPIO_MOTION Counter Interrupt Mask Enable
0x1014047c	GPIO_MOTION_CNTR_IRQ_MSK_RESET	GPIO_MOTION Counter Interrupt Mask Disable

GPIOs have the following features:

- Each GPIO can be configured individually as input or output, inverted or none 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.

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 two 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 0x101c0800
GPIO_CFG0 – GPIO_MOTION 0 Configuration Register 0x10140400
GPIO_CFG1 – GPIO 1 Configuration Register 0x101c0804
GPIO_CFG1 – GPIO_MOTION 1 Configuration Register 0x10140404
GPIO_CFG2 – GPIO 2 Configuration Register 0x101c0808
GPIO_CFG2 – GPIO_MOTION 2 Configuration Register 0x10140408
GPIO_CFG3 – GPIO 3 Configuration Register 0x101c080c
GPIO_CFG3 – GPIO_MOTION 3 Configuration Register 0x1014040c
GPIO_CFG4 – GPIO 4 Configuration Register 0x101c0810
GPIO_CFG4 – GPIO_MOTION 4 Configuration Register 0x10140410
GPIO_CFG5 – GPIO 5 Configuration Register 0x101c0814
GPIO_CFG5 – GPIO_MOTION 5 Configuration Register 0x10140414
GPIO_CFG6 – GPIO 6 Configuration Register 0x101c0818
GPIO_CFG6 – GPIO_MOTION 6 Configuration Register 0x10140418
GPIO_CFG7 – GPIO 7 Configuration Register 0x101c081c
GPIO_CFG7 – GPIO_MOTION 7 Configuration Register 0x1014041c

These registers are accessible via intlogic and intlogic_motion address area.

31 ... 7	6 5	4	3 2 1 0
reserved	COUNT_REF	INV	MODE

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x0
6:5	COUNT_REF	counter reference 00: counter 0 01: counter 1 10: reserved 11: system time	R/W	0x0
4	INV	1: invert input/output value 0: don't invert input/output	R/W	0x0
3:0	MODE	defines the GPIO input or output mode - depends on IO_CFG 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 (in intlogic or intlogic_motion) to capture again) 0011: capture once at high level (reset gpio_irq (in intlogic or intlogic_motion) to capture again) Output mode: 0100: set to 0 0101: set to 1 0110: set to gpio_out[0] 0111: pwm mode, direct threshold update (might cause hazards on output) Multi pin mode: 1110: dc-dc-mode: with 2 threshold values to update at counter=0 from gpio_thrsh_capt[n+1] register (hazard-free) 1111: pwm2-mode with threshold update at counter=0 from gpio_thrsh_capt [n+1] register (hazard-free)	R/W	0x0

GPIO_THRSH_CAPT0 – GPIO 0 Threshold or Capture Register 0x101c0820
GPIO_THRSH_CAPT0 – GPIO_MOTION 0 Threshold or Capture Register 0x10140420
GPIO_THRSH_CAPT1 – GPIO 1 Threshold or Capture Register 0x101c0824
GPIO_THRSH_CAPT1 – GPIO_MOTION 1 Threshold or Capture Register 0x10140424
GPIO_THRSH_CAPT2 – GPIO 2 Threshold or Capture Register 0x101c0828
GPIO_THRSH_CAPT2 – GPIO_MOTION 2 Threshold or Capture Register 0x10140428
GPIO_THRSH_CAPT3 – GPIO 3 Threshold or Capture Register 0x101c082c
GPIO_THRSH_CAPT3 – GPIO_MOTION 3 Threshold or Capture Register 0x1014042c
GPIO_THRSH_CAPT4 – GPIO 4 Threshold or Capture Register 0x101c0830
GPIO_THRSH_CAPT4 – GPIO_MOTION 4 Threshold or Capture Register 0x10140430
GPIO_THRSH_CAPT5 – GPIO 5 Threshold or Capture Register 0x101c0834
GPIO_THRSH_CAPT5 – GPIO_MOTION 5 Threshold or Capture Register 0x10140434
GPIO_THRSH_CAPT6 – GPIO 6 Threshold or Capture Register 0x101c0838
GPIO_THRSH_CAPT6 – GPIO_MOTION 6 Threshold or Capture Register 0x10140438
GPIO_THRSH_CAPT7 – GPIO 7 Threshold or Capture Register 0x101c083c
GPIO_THRSH_CAPT7 – GPIO_MOTION 7 Threshold or Capture Register 0x1014043c

These registers are accessible via intlogic and intlogic_motion address area.

31–0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Threshold/Capture register: PWM mode (threshold): The counter threshold value equals the number of inactive clockcylces per period (cycles with pwm=0). Therefore it is interpreted different in symmetrical and asymmetrical counter mode: Asymmetrical mode (sawtooth): pwm = (counter >= gpio_thrsh_capt) Symmetrical mode (triangle) : counter is compared with gpio_thrsh_capt[7:1], gpio_thrsh_capt [0] prolongs the inactive phase by 1 cc only while up counting. This allows running 10ns resolution even in symmetrical mode. Capture mode (capture register) In capture mode this register holds the captured counter value.	R/W	0x0

GPIO_CNTR0_CTRL – GPIO Counter 0 Control 0x101c0840
GPIO_CNTR0_CTRL – GPIO_MOTION Counter 0 Control 0x10140440
GPIO_CNTR1_CTRL – GPIO Counter 1 Control 0x101c0844
GPIO_CNTR1_CTRL – GPIO_MOTION Counter 1 Control 0x10140444

This register is accessible via intlogic and intlogic_motion address area.

31 … 10	9 … 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
31:10	reserved	-	R	0x0
9:7	GPIO_REF	gpio reference (0 - 7)	R/W	0x0
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	0x0
4	ONCE	1: count once (reset run bit after 1 period) 0: count continue	R/W	0x0
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	0x0
2	IRQ_EN	1: enable interrupt request on sel_event 0: disable interrupt request	R/W	0x0
1	SYM_NASYM	1: symmetric mode (triangle) 0: asymmetric mode (sawtooth)	R/W	0x0
0	RUN	1: start counter, counter is running 0: stop counter	R/W	0x0

GPIO_CNTR0_MAX – GPIO Counter 0 Maximum Value 0x101c0848
GPIO_CNTR0_MAX – GPIO_MOTION Counter 0 Maximum Value 0x10140448
GPIO_CNTR1_MAX – GPIO Counter 1 Maximum Value 0x101c084c
GPIO_CNTR1_MAX – GPIO_MOTION Counter 1 Maximum Value 0x1014044c

These registers are accessible via intlogic and intlogic_motion address area.

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Asymmetric mode: counting period in cc + 1 Symmetric mode: counting period in cc	R/W	0x0

GPIO_CNTR0_CNT – GPIO Counter 0 Current Value **0x101c0850**
GPIO_CNTR0_CNT – GPIO_MOTION Counter 0 Current Value **0x10140450**
GPIO_CNTR1_CNT – GPIO Counter 1 Current Value **0x101c0854**
GPIO_CNTR1_CNT – GPIO_MOTION Counter 1 Current Value **0x10140454**

These registers are accessible via intlogic and intlogic_motion address area.

31 30 29 28 27 26 25 24 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	current counter value	R/W	0x0

GPIO_OUT – GPIO Output Register **0x101c0858**
GPIO_OUT – GPIO_MOTION Output Register **0x10140458**

This register is accessible via intlogic and intlogic_motion address area.

31 30 29 28 27 26 25 24 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	0x0
7:0	VAL	gpio output values	R/W	0x0

GPIO_IN – GPIO Input Register **0x101c085c**
GPIO_IN – GPIO_MOTION Input Register **0x1014045c**

This register is accessible via intlogic and intlogic_motion address area.

31 30 29 28 27 26 25 24 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	0x0
7:0	VAL	gpio input values	R	0x0

GPIO_IRQ_RAW – GPIO Raw IRQ Register 0x101c0860
GPIO_IRQ_RAW – GPIO_MOTION Raw IRQ Register 0x10140460

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
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
reserved																								GPIO7	GPIO6	GPIO5	GPIO4	GPIO3	GPIO2	GPIO1	GPIO0

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7	GPIO7	interrupt bit for GPIO7	R/W	0x0
6	GPIO6	interrupt bit for GPIO6	R/W	0x0
5	GPIO5	interrupt bit for GPIO5	R/W	0x0
4	GPIO4	interrupt bit for GPIO4	R/W	0x0
3	GPIO3	interrupt bit for GPIO3	R/W	0x0
2	GPIO2	interrupt bit for GPIO2	R/W	0x0
1	GPIO1	interrupt bit for GPIO1	R/W	0x0
0	GPIO0	interrupt bit for GPIO0	R/W	0x0

GPIO_IRQ_MSK – GPIO Masked IRQ Register 0x101c0864
GPIO_MOTION_IRQ_MSK – GPIO_MOTION Masked IRQ Register 0x10140464

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.
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																								GPIO							

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	GPIO	One bit per GPIO	R	0x0

GPIO_IRQ_MSK_SET – GPIO Interrupt Enable 0x101c0868
GPIO_MOTION_IRQ_MSK_SET – GPIO_MOTION Interrupt Enable 0x10140468

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit.
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
reserved	GPIO

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	GPIO	One bit per GPIO	R/W	0x0

GPIO_IRQ_MSK_RESET – GPIO Interrupt Disable 0x101c086c
GPIO_MOTION_IRQ_MSK_RESET – GPIO_MOTION Interrupt Disable 0x1014046c

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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	GPIO

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	GPIO	One bit per GPIO	R/W	0x0

CNTR_IRQ_RAW – GPIO Counter Raw IRQ Register 0x101c0870
CNTR_IRQ_RAW – GPIO_MOTION Counter Raw IRQ Register 0x10140470

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
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
reserved																														CNT1	CNT0

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	CNT1	interrupt bit for counter1	R/W	0x0
0	CNT0	interrupt bit for counter0	R/W	0x0

GPIO_CNTR_IRQ_MSK – GPIO Counter Masked IRQ Register 0x101c0874
GPIO_MOTION_CNTR_IRQ_MSK – GPIO_MOTION Counter Masked IRQ Register 0x10140474

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.
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																														CNT1	CNT0

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	CNT1	interrupt bit for counter1	R	0x0
0	CNT0	interrupt bit for counter0	R	0x0

GPIO_CNTR_IRQ_MSK_SET – GPIO Counter Interrupt Request Mask Enable 0x101c0878
GPIO_MOTION_CNTR_IRQ_MSK_SET – GPIO_MOTION Counter Interrupt Request Mask Enable 0x10140478

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit.
Write access with ’0’ does not influence this bit.
Read access shows actual interrupt mask.

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	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	CNT1	counter1 interrupt mask bit	R/W	0x0
0	CNT0	counter0 interrupt mask bit	R/W	0x0

GPIO_CNTR_IRQ_MSK_RESET – GPIO Counter Interrupt Request Mask Disable 0x101c087c
GPIO_MOTION_CNTR_IRQ_MSK_RESET – GPIO_MOTION Counter Interrupt Request Mask Disable 0x1014047c

This register exists twice for intlogic (ARM) and intlogic_motion (xPIC) address area.

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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	CNT1	CNT0

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	CNT1	counter1 interrupt mask bit	R/W	0x0
0	CNT0	counter0 interrupt mask bit	R/W	0x0

9.2 ARM_TIMER

The following table shows a summary of all registers related to ARM timers.

ARM Address	Register Name	Short Description
0x101c0900	ARM_TIMER_CONFIG_TIMER0	ARM TIMER Config Register0
0x101c0904	ARM_TIMER_CONFIG_TIMER1	ARM TIMER Config Register1
0x101c0908	ARM_TIMER_PRELOAD_TIMER0	ARM TIMER Timer 0
0x101c090c	ARM_TIMER_PRELOAD_TIMER1	ARM TIMER Timer 1
0x101c0910	ARM_TIMER_TIMER0	ARM TIMER Timer 0
0x101c0914	ARM_TIMER_TIMER1	ARM TIMER Timer 1
0x101c0918	SYS_TIME_S	ARM_TIMER Upper SYSTIME Register
0x101c091c	SYS_TIME_NS	ARM_TIMER Lower SYSTIME Register
0x101c0920	ARM_TIMER_SYSTIME_NS_COMPARE	SYSTIME Nano Sec Compare Value
0x101c0924	ARM_TIMER_SYSTIME_S_COMPARE	SYSTIME Sec Compare Value
0x101c0928	ARM_TIMER_IRQ_RAW	ARM_TIMER Raw IRQ Register
0x101c092c	ARM_TIMER_IRQ_MASKED	ARM_TIMER Masked IRQ Register
0x101c0930	ARM_TIMER_IRQ_MSK_SET	ARM_TIMER Interrupt Mask Enable
0x101c0934	ARM_TIMER_IRQ_MSK_RESET	ARM_TIMER Interrupt Mask Disable

ARM_TIMER_CONFIG_TIMER0 – ARM TIMER Config Register0 0x101c0900
ARM_TIMER_CONFIG_TIMER1 – ARM TIMER Config Register1 0x101c0904

31 … 2	1 … 0
reserved	MODE

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1:0	MODE	Timer0/1 2'b00 : Timer stops at 0 2'b01 : Timer is preload with value from preload register at 0 2'b10 : Timer (value) compare with systime (once) 2'b11 : reserved	R/W	0x0

ARM_TIMER_PRELOAD_TIMER0 – ARM TIMER Timer 0 0x101c0908
ARM_TIMER_PRELOAD_TIMER1 – ARM TIMER Timer 1 0x101c090c

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	preload value	R/W	0x0

ARM_TIMER_TIMER0 – ARM TIMER Timer 0 0x101c0910
ARM_TIMER_TIMER1 – ARM TIMER Timer 1 0x101c0914

31	30	29	28	27	26	25	24	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	actual value of timer / systime compare value	R/W	0x0

SYS_TIME_S – ARM_TIMER Upper SYSTIME Register 0x101c0918

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
This register should be dedicated to accesses via ARM.
xPIC software should access systime via xPIC_TIMER at systime_s.
Host software should access systime via DPM at systime_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
VAL																															

Bits	Name	Description	R/W	Default
31:0	VAL	Systime high: Sample systime_ns at read access to systime_s. Value is incremented, if systime_ns reaches systime_border.	R	0x0

SYS_TIME_NS – ARM_TIMER Lower SYSTIME Register 0x101c091c

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
If no systime_s is read before (e.g. at 2nd read access of systime_ns), the actual value of systime_ns is read.
This register should be dedicated to accesses via ARM.
xPIC software should access systime via xPIC_TIMER at systime_ns.
Host software should access systime via DPM at systime_ns.

31	30	29	28	27	26	25	24	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	Systime low: Sample systime_ns at read access to systime_s. Without sample read systime_s, read the actual value of systime_ns.	R	0x0

ARM_TIMER_SYSTIME_NS_COMPARE – SYSTIME Nano Sec Compare Value 0x101c0920

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	compare value with systime_ns (nano seconds) set arm_timer_irq_raw-systime_ns_irq if systime_ns is reached	R/W	0x0

ARM_TIMER_SYSTIME_S_COMPARE – SYSTIME Sec Compare Value 0x101c0924

31 … 0
VAL

Bits	Name	Description	R/W	Default
31:0	VAL	Compare value with systime_s (seconds): Write value, then reset arm_timer_irq_raw-systime_s_irq to activate compare machine. arm_timer_irq_raw-systime_s_irq is set, if systime_s matches.	R/W	0x0

ARM_TIMER_IRQ_RAW – ARM_TIMER Raw IRQ Register 0x101c0928

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
Write access with '1' resets the appropriate IRQ.
Write access with '0' does not influence this bit.

31 … 4	3	2	1	0
reserved	SYSTIME_S_IRQ	SYSTIME_NS_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime sec Interrupt	R/W	0x0
2	SYSTIME_NS_IRQ	Systime ns Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

ARM_TIMER_IRQ_MASKED – ARM_TIMER Masked IRQ Register 0x101c092c

Show status of masked IRQs (as connected to ARM/xPIC).

31–4	3	2	1	0
reserved	SYSTIME_S_IRQ	SYSTIME_NS_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime sec Interrupt	R	0x0
2	SYSTIME_NS_IRQ	Systime ns Interrupt	R	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R	0x0

ARM_TIMER_IRQ_MSK_SET – ARM_TIMER Interrupt Mask Enable 0x101c0930

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit.
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 ARM_TIMER_IRQ_RAW.

31–4	3	2	1	0
reserved	SYSTIME_S_IRQ	SYSTIME_NS_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime sec Interrupt	R/W	0x0
2	SYSTIME_NS_IRQ	Systime ns Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

ARM_TIMER_IRQ_MSK_RESET – ARM_TIMER Interrupt Mask Disable 0x101c0934

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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_S_IRQ	SYSTIME_NS_IRQ	TIMER1_IRQ	TIMER0_IRQ

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SYSTIME_S_IRQ	Systime sec Interrupt	R/W	0x0
2	SYSTIME_NS_IRQ	Systime ns Interrupt	R/W	0x0
1	TIMER1_IRQ	Timer 1 Interrupt	R/W	0x0
0	TIMER0_IRQ	Timer 0 Interrupt	R/W	0x0

9.3 IO-Link Interface

The netX10 is equipped with four programmable serial XLINK controllers, connected to the Multiplex Matrix, which can, together with the xPIC, operate as I/O-Link Master Controllers.

The following table shows a summary of all registers of IO_Link.

ARM Address	Register Name	Short Description
0x10140600	XLINK0_XLINK_CFG	XLINK0 Configuration Register
0x10140604	XLINK0_XLINK_TX	XLINK0 Transmit Register
0x10140608	XLINK0_XLINK_RX	XLINK0 RX Register
0x1014060c	XLINK0_XLINK_STAT	XLINK0 Status Register
0x10140610	XLINK1_XLINK_CFG	XLINK1 Configuration Register
0x10140614	XLINK1_XLINK_TX	XLINK1 Transmit Register
0x10140618	XLINK1_XLINK_RX	XLINK1 RX Register
0x1014061c	XLINK1_XLINK_STAT	XLINK1 Status Register
0x10140620	XLINK2_XLINK_CFG	XLINK2 Configuration Register
0x10140624	XLINK2_XLINK_TX	XLINK2 Transmit Register
0x10140628	XLINK2_XLINK_RX	XLINK2 RX Register
0x1014062c	XLINK2_XLINK_STAT	XLINK2 Status Register
0x10140630	XLINK3_XLINK_CFG	XLINK3 Configuration Register
0x10140634	XLINK3_XLINK_TX	XLINK3 Transmit Register
0x10140638	XLINK3_XLINK_RX	XLINK3 RX Register
0x1014063c	XLINK3_XLINK_STAT	XLINK3 Status Register
0x10140640	IO_LINK_IRQ_RAW	IO-Link Raw interrupts
0x10140644	IO_LINK_IRQ_MASKED	IO-Link Masked IRQ Register
0x10140648	IO_LINK_IRQ_MSK_SET	IO-Link Interrupt Mask Enable
0x1014064c	IO_LINK_IRQ_MSK_RESET	IO-Link Interrupt Mask Disable
0x10140650	IO_LINK_IRQ_ENABLE	IO-Link Processor Enable

XLINK0_XLINK_CFG – XLINK0 Configuration Register 0x10140600
XLINK1_XLINK_CFG – XLINK1 Configuration Register 0x10140610
XLINK2_XLINK_CFG – XLINK2 Configuration Register 0x10140620
XLINK3_XLINK_CFG – XLINK3 Configuration Register 0x10140630

31:28	27:24	23:20	19	18	17	16	15:0
END_SPL	START_SPL	BITS2REC	CNT_DA	BCLK2OE_EN	FB_EN	XLINK_EN	RATE_INC

Bits	Name	Description	R/W	Default
31:28	END_SPL	end sample point for receive data	R/W	0xb
27:24	START_SPL	start sample point for receive data a sample period is defined as 1/16 of the bitrate period range: 0x0 - 0xf note: settings for start_spl and end_spl should always fulfill the condition: (start_spl < end_spl)	R/W	0x4
23:20	BITS2REC	count of bits to receive note: the reset value expect: 1stopbit, 8databits, 1paritybit and 1stopbit	R/W	0xa
19	CNT_DA	test feature, do not set this bit!	R/W	0x0
18	BCLK2OE_EN	test feature, do not set this bit!	R/W	0x0
17	FB_EN	test feature, enable internal feedback	R/W	0x0
16	XLINK_EN	disable the output enable, and activity	R/W	0x0
15:0	RATE_INC	bitrate compare value for bit clock counter (bit_cnt) BITRATE = 100e6/(rate_inc) typical settings for IOLINK: BIT_RATE rate_inc clock period - calc: 1/BIT_RATE 4800 0x5160 208,33 us - 208,3333us 38400 0xa2b 26,04 us - 26,04167us 230400 0x1b1 4,34 us - 4,340278us ... invalid: 0 0 0 - 0	R/W	0x1b

XLINK0_XLINK_TX – XLINK0 Transmit Register 0x10140604
XLINK1_XLINK_TX – XLINK1 Transmit Register 0x10140614
XLINK2_XLINK_TX – XLINK2 Transmit Register 0x10140624
XLINK3_XLINK_TX – XLINK3 Transmit Register 0x10140634

31:18	17	16	15:0
reserved	IDLE_RO	RDY_RO	HOLD

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x0
17	IDLE_RO	indicates no activity on tx	R/W	0x1
16	RDY_RO	TX buffer ready (valid on ready) 0 TX buffer not ready 1 TX buffer ready	R/W	0x1
15:0	HOLD	hold register format for a valid serial DATA sequence: <-ctrl.DATA-><------------------- serial DATA --------------------> { END_BIT:1 }[{STOPBIT:1}{DATABITS max. 12:0101..0010}{STARTBIT:0}] notes: ENDBIT is a hardware marker to stop the shifting, and will not be transmitted. this condition implied, than all other not used bits should be zero	R/W	0x0

XLINK0_XLINK_RX – XLINK0 RX Register **0x10140608**
XLINK1_XLINK_RX – XLINK1 RX Register **0x10140618**
XLINK2_XLINK_RX – XLINK2 RX Register **0x10140628**
XLINK3_XLINK_RX – XLINK3 RX Register **0x10140638**

Writing to the register, reset the ready bit, the overflow bit and the sampling error bit.

31 … 22	21	20	19	18 … 17	16	15 … 0
reserved	SPL_ERR_RO	OVF_ERR_RO	RXD_RO	reserved	RDY_RO	HOLD_RO

Bits	Name	Description	R/W	Default
31:22	reserved	-	R	0x0
21	SPL_ERR_RO	sampling error detected if the amount of sampled bits (HI or LOW) do not fulfill the condition: (end_spl - start_spl) < (count of HI/LOW bits)	R/W	0x0
20	OVF_ERR_RO	overflow error on received data	R/W	0x0
19	RXD_RO	current status of rx data	R/W	0x0
18:17	reserved	-	R	0x0
16	RDY_RO	RX buffer ready (valid on ready) 0 RX buffer not ready 1 RX buffer ready	R/W	0x0
15:0	HOLD_RO	RX byte (when valid) hold[15:0] is used to shift in RX(LSB first!) the amount of shifted bits is defined by bits2rec shift order is bit15 downto bit0	R/W	0xffff

XLINK0_XLINK_STAT – XLINK0 Status Register 0x1014060c
XLINK1_XLINK_STAT – XLINK1 Status Register 0x1014061c
XLINK2_XLINK_STAT – XLINK2 Status Register 0x1014062c
XLINK3_XLINK_STAT – XLINK3 Status Register 0x1014063c

Writing to this register set the bit clock counter to 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	HILO_CNT_RO					BITS_SH_RO				FSM_RO		TXOE_RO	RXO_RO	TXO_RO	BIT_CLK_RO	BIT_CNT_RO															

Bits	Name	Description	R/W	Default
31	reserved	-	R	0x0
30:26	HILO_CNT_RO	debug status	R/W	0x0
25:22	BITS_SH_RO	debug status	R/W	0x0
21:20	FSM_RO	debug status	R/W	0x0
19	TXOE_RO	status of tx output enable	R/W	0x0
18	RXO_RO	status of rx input	R/W	0x0
17	TXO_RO	status of tx output	R/W	0x0
16	BIT_CLK_RO	status of bit clock signal	R/W	0x0
15:0	BIT_CNT_RO	status of bit clock counter	R/W	0x0

IO_LINK_IRQ_RAW – IO-Link Raw Interrupts 0x10140640

Read access shows status of unmasked IRQs. IRQs are set automatically and reset by writing to this register:
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
reserved																	XLINK3_SHIFT_EN	XLINK3_RX_NEXT	XLINK3_TX_NEXT	reserved	XLINK2_SHIFT_EN	XLINK2_RX_NEXT	XLINK2_TX_NEXT	reserved	XLINK1_SHIFT_EN	XLINK1_RX_NEXT	XLINK1_TX_NEXT	reserved	XLINK0_SHIFT_EN	XLINK0_RX_NEXT	XLINK0_TX_NEXT

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XLINK3_SHIFT_EN	shift_en interrupt	R/W	0x0
13	XLINK3_RX_NEXT	rx_next interrupt	R/W	0x0
12	XLINK3_TX_NEXT	tx_next interrupt	R/W	0x0
11	reserved	-	R	0x0
10	XLINK2_SHIFT_EN	shift_en interrupt	R/W	0x0
9	XLINK2_RX_NEXT	rx_next interrupt	R/W	0x0
8	XLINK2_TX_NEXT	tx_next interrupt	R/W	0x0
7	reserved	-	R	0x0
6	XLINK1_SHIFT_EN	shift_en interrupt	R/W	0x0
5	XLINK1_RX_NEXT	rx_next interrupt	R/W	0x0
4	XLINK1_TX_NEXT	tx_next interrupt	R/W	0x0
3	reserved	-	R	0x0
2	XLINK0_SHIFT_EN	shift_en interrupt	R/W	0x0
1	XLINK0_RX_NEXT	rx_next interrupt	R/W	0x0
0	XLINK0_TX_NEXT	tx_next interrupt	R/W	0x0

IO_LINK_IRQ_MASKED – IO-Link Masked IRQ Register 0x10140644

Show status of masked IRQs (as connected to ARM/xPIC).

31 … 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	XLINK3_SHIFT_EN	XLINK3_RX_NEXT	XLINK3_TX_NEXT	reserved	XLINK2_SHIFT_EN	XLINK2_RX_NEXT	XLINK2_TX_NEXT	reserved	XLINK1_SHIFT_EN	XLINK1_RX_NEXT	XLINK1_TX_NEXT	reserved	XLINK0_SHIFT_EN	XLINK0_RX_NEXT	XLINK0_TX_NEXT

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XLINK3_SHIFT_EN	shift_en interrupt	R	0x0
13	XLINK3_RX_NEXT	rx_next interrupt	R	0x0
12	XLINK3_TX_NEXT	tx_next interrupt	R	0x0
11	reserved	-	R	0x0
10	XLINK2_SHIFT_EN	shift_en interrupt	R	0x0
9	XLINK2_RX_NEXT	rx_next interrupt	R	0x0
8	XLINK2_TX_NEXT	tx_next interrupt	R	0x0
7	reserved	-	R	0x0
6	XLINK1_SHIFT_EN	shift_en interrupt	R	0x0
5	XLINK1_RX_NEXT	rx_next interrupt	R	0x0
4	XLINK1_TX_NEXT	tx_next interrupt	R	0x0
3	reserved	-	R	0x0
2	XLINK0_SHIFT_EN	shift_en interrupt	R	0x0
1	XLINK0_RX_NEXT	rx_next interrupt	R	0x0
0	XLINK0_TX_NEXT	tx_next interrupt	R	0x0

IO_LINK_IRQ_MSK_SET – IO-Link Interrupt Mask Enable 0x10140648

The IRQ mask enables interrupt requests for corresponding interrupt sources. As its bits might be changed by different software tasks, the IRQ mask register is not writable directly, but by set and reset masks:
Write access with ’1’ sets interrupt mask bit (enables interrupt request for corresponding interrupt source).
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 IO_LINK_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																	XLINK3_SHIFT_EN	XLINK3_RX_NEXT	XLINK3_TX_NEXT	reserved	XLINK2_SHIFT_EN	XLINK2_RX_NEXT	XLINK2_TX_NEXT	reserved	XLINK1_SHIFT_EN	XLINK1_RX_NEXT	XLINK1_TX_NEXT	reserved	XLINK0_SHIFT_EN	XLINK0_RX_NEXT	XLINK0_TX_NEXT

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XLINK3_SHIFT_EN	shift_en interrupt	R/W	0x0
13	XLINK3_RX_NEXT	rx_next interrupt	R/W	0x0
12	XLINK3_TX_NEXT	tx_next interrupt	R/W	0x0
11	reserved	-	R	0x0
10	XLINK2_SHIFT_EN	shift_en interrupt	R/W	0x0
9	XLINK2_RX_NEXT	rx_next interrupt	R/W	0x0
8	XLINK2_TX_NEXT	tx_next interrupt	R/W	0x0
7	reserved	-	R	0x0
6	XLINK1_SHIFT_EN	shift_en interrupt	R/W	0x0
5	XLINK1_RX_NEXT	rx_next interrupt	R/W	0x0
4	XLINK1_TX_NEXT	tx_next interrupt	R/W	0x0
3	reserved	-	R	0x0
2	XLINK0_SHIFT_EN	shift_en interrupt	R/W	0x0
1	XLINK0_RX_NEXT	rx_next interrupt	R/W	0x0
0	XLINK0_TX_NEXT	tx_next interrupt	R/W	0x0

IO_LINK_IRQ_MSK_RESET – IO-Link Interrupt Mask Disable 0x1014064c

This is the corresponding reset mask to disable interrupt requests for corresponding interrupt sources:
Write access with ’1’ resets interrupt mask bit.
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																	XLINK3_SHIFT_EN	XLINK3_RX_NEXT	XLINK3_TX_NEXT	reserved	XLINK2_SHIFT_EN	XLINK2_RX_NEXT	XLINK2_TX_NEXT	reserved	XLINK1_SHIFT_EN	XLINK1_RX_NEXT	XLINK1_TX_NEXT	reserved	XLINK0_SHIFT_EN	XLINK0_RX_NEXT	XLINK0_TX_NEXT

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14	XLINK3_SHIFT_EN	shift_en interrupt	R/W	0x0
13	XLINK3_RX_NEXT	rx_next interrupt	R/W	0x0
12	XLINK3_TX_NEXT	tx_next interrupt	R/W	0x0
11	reserved	-	R	0x0
10	XLINK2_SHIFT_EN	shift_en interrupt	R/W	0x0
9	XLINK2_RX_NEXT	rx_next interrupt	R/W	0x0
8	XLINK2_TX_NEXT	tx_next interrupt	R/W	0x0
7	reserved	-	R	0x0
6	XLINK1_SHIFT_EN	shift_en interrupt	R/W	0x0
5	XLINK1_RX_NEXT	rx_next interrupt	R/W	0x0
4	XLINK1_TX_NEXT	tx_next interrupt	R/W	0x0
3	reserved	-	R	0x0
2	XLINK0_SHIFT_EN	shift_en interrupt	R/W	0x0
1	XLINK0_RX_NEXT	rx_next interrupt	R/W	0x0
0	XLINK0_TX_NEXT	tx_next interrupt	R/W	0x0

IO_LINK_IRQ_ENABLE – IO-Link Processor Enable 0x10140650

Enable all IRQs for xPIC and/or 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																														XPIC_EN	ARM_EN

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	XPIC_EN	enable interrupts to xPIC	R/W	0x1
0	ARM_EN	enable interrupts to ARM	R/W	0x0

9.4 UART – Universal Asynchronous Receiver Transmitter

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

ARM Address	Register Name	Short Description
0x101c0b00	UART0_DATA	UART0 Data Register
0x101c0b04	UART0_STAT	UART0 Status Register
0x101c0b08	UART0_LINE_CTRL	UART0 Line Control Register
0x101c0b0c	UART0_BAUD_DIV_MSB	UART0 Baud Rate Divisor MSB
0x101c0b10	UART0_BAUD_DIV_LSB	UART0 Baud Rate Divisor LSB
0x101c0b14	UART0_CTRL	UART0 Control Register
0x101c0b18	UART0_FLAG	UART0 Flag Register
0x101c0b1c	UART0_INT_ID	UART0 Interrupt Identification Register
0x101c0b20	UART0_IRDA_LO_PWR_CNTR	UART0 IrDA Low Power Counter Register
0x101c0b24	UART0_RTS_CTRL	UART0 RTS Control Register
0x101c0b28	UART0_RTS_LEAD_CYC	UART0 RTS Leading Cycles
0x101c0b2c	UART0_RTS_TRAIL_CYC	UART0 RTS Trailing Cycles
0x101c0b30	UART0_OUT_DRV_EN	UART0 UART Output Driver Enable Register
0x101c0b34	UART0_BAUD_MODE_CTRL	UART0 Baud Rate Mode Control Register
0x101c0b38	UART0_RX_FIFO_IRQ_LVL	UART0 RX FIFO Trigger Level and RX-DMA Enable
0x101c0b3c	UART0_TX_FIFO_IRQ_LVL	UART0 TX FIFO Trigger Level and TX-DMA Enable
0x101c0b40	UART1_DATA	UART1 Data Register
0x101c0b44	UART1_STAT	UART1 Status Register
0x101c0b48	UART1_LINE_CTRL	UART1 Line Control Register
0x101c0b4c	UART1_BAUD_DIV_MSB	UART1 Baud Rate Divisor MSB
0x101c0b50	UART1_BAUD_DIV_LSB	UART1 Baud Rate Divisor LSB
0x101c0b54	UART1_CTRL	UART1 Control Register
0x101c0b58	UART1_FLAG	UART1 Flag Register
0x101c0b5c	UART1_INT_ID	UART1 Interrupt Identification Register
0x101c0b60	UART1_IRDA_LO_PWR_CNTR	UART1 IrDA Low Power Counter Register
0x101c0b64	UART1_RTS_CTRL	UART1 RTS Control Register
0x101c0b68	UART1_RTS_LEAD_CYC	UART1 RTS Leading Cycles
0x101c0b6c	UART1_RTS_TRAIL_CYC	UART1 RTS Trailing Cycles
0x101c0b70	UART1_OUT_DRV_EN	UART1 UART Output Driver Enable Register
0x101c0b74	UART1_BAUD_MODE_CTRL	UART1 Baud Rate Mode Control Register
0x101c0b78	UART1_RX_FIFO_IRQ_LVL	UART1 RX FIFO Trigger Level and RX-DMA Enable
0x101c0b7c	UART1_TX_FIFO_IRQ_LVL	UART1 TX FIFO Trigger Level and TX-DMA Enable

UART0_DATA – UART0 Data Register **0x101c0b00**
UART1_DATA – UART1 Data Register **0x101c0b40**

These registers are the send and receive data 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 ... 11	10	9	8	7 ... 0
reserved	BE	PE	FE	DATA

Bits	Name	Description	R/W	Default
31:11	reserved	-	R	0x0
10	BE	Break Error, read only, mirrored from UART_STAT, to handle in DMA-read-out data	R	0x0
9	PE	Parity Error, read only, mirrored from UART_STAT, to handle in DMA-read-out data	R	0x0
8	FE	Framing Error, read only, mirrored from UART_STAT, to handle in DMA-read-out data	R	0x0
7:0	DATA	Receive data character by reading. Transmit data character by writing.	R/W	0x0

UART0_STAT – UART0 Receive Status Register (read) / Error Clear Register (write) 0x101c0b04
UART1_STAT – UART1 Receive Status Register (read) / Error Clear Register (write) 0x101c0b44

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	0x0
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	0x0
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	0x0
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	0x0
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	0x0

UART0_LINE_CTRL – UART0 Line Control Register 0x101c0b08
UART1_LINE_CTRL – UART1 Line Control Register 0x101c0b48

31 30 29 28 27 26 25 24 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	0x0
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	0x0
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	0x0
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	0x0
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	0x0
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	0x0
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	0x0

UART Baud Rate Generation

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

BAUDDIV = (100 MHz / (16 • BaudRate)) – 1 (UART_BAUD_MODE_CTRL[0] = 0)

BAUDDIV = ((Baud Rate • 16) / 100 MHz) • 2^{16} . (UART_BAUD_MODE_CTRL[0] = 1)

The maximum Baud rate is 3.125 MBaud.

UART0_BAUD_DIV_MSB – UART0 Baud Rate Divisor MSB 0x101c0b0c
UART1_BAUD_DIV_MSB – UART1 Baud Rate Divisor MSB 0x101c0b4c

31 … 8	7 … 0
reserved	BAUDDIVMS

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	BAUDDIVMS	Baud Rate Divisor [15:8]. Most significant byte of baud rate divisor (BAUDDIV).	R/W	0x0

UART0_BAUD_DIV_LSB – UART0 Baud Rate Divisor LSB 0x101c0b10
UART1_BAUD_DIV_LSB – UART1 Baud Rate Divisor LSB 0x101c0b50

31 … 8	7 … 0
reserved	BAUDDIVLS

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	BAUDDIV_LSB	Baud Rate Divisor [7:0]. Least significant byte of baud rate divisor (BAUDDIV).	R/W	0x0

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 – UART0 Control Register 0x101c0b14
UART1_CTRL – UART1 Control Register 0x101c0b54

31:9	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	0x0
8	TX_RX_LOOP	internal loop (TX -> RX) (test purpose only).	R/W	0x0
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	0x0
6	RTIE	Receive timeout interrupt enable. 1: the receive timeout interrupt is enabled.	R/W	0x0
5	TIE	Transmit interrupt enable. 1: the transmit interrupt is enabled.	R/W	0x0
4	RIE	Receive interrupt enable. 1: the receive interrupt is enabled.	R/W	0x0
3	MSIE	Modem status interrupt enable. 1: the modem status interrupt is enabled.	R/W	0x0
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	0x0
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 nSIROUT 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, nSIROUT remains cleared to 0 (no light pulse generated), and signal transitions on SIRIN will have no effect.	R/W	0x0
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	0x0

UART0_FLAG – UART0 Flag Register 0x101c0b18
UART1_FLAG – UART1 Flag Register 0x101c0b58

31:8	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	0x0
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	0x0
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	0x0
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	0x0
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	0x0
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	0x0
2	DCD	Data carrier detect. This bit is actually not supported. Always read as 0.	R	0x0
1	DSR	Data set ready. This bit is actually not supported. Always read as 0.	R	0x0
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	0x0

UART0_INT_ID – UART0 Interrupt Identification Register 0x101c0b1c
UART1_INT_ID – UART1 Interrupt Identification Register 0x101c0b5c

UART_INT_ID is the interrupt identification (read)/interrupt clear (write) 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 0x101c0b20
UART1_IRDA_LO_PWR_CNTR – UART 0 IrDA Low Power Counter Register 0x101c0b60

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*Baudrate=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:

ILPDVSR = (100MHz / 16*Baudrate) – 1

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

UART0_RTS_CTRL – UART0 RTS Control Register 0x101c0b24
UART1_RTS_CTRL – UART1 RTS Control Register 0x101c0b64

31 … 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 AUT0 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 – UART0 RTS Leading Cycles 0x101c0b28
UART1_RTS_LEAD_CYC – UART1 RTS Leading Cycles 0x101c0b68

31 … 8	7 … 0
reserved	FORERUN

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 – UART0 RTS Trailing Cycles 0x101c0b2c
UART1_RTS_TRAIL_CYC – UART1 RTS Trailing Cycles 0x101c0b6c

31 ... 8	7 ... 0
reserved	TRAIL

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 – UART0 Output Driver Enable Register 0x101c0b30
UART1_OUT_DRV_EN – UART1 Output Driver Enable Register 0x101c0b70

31 ... 2	1	0
reserved	DRVRTS	DRVTX

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x00
1	DRVRTS	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 – UART0 Baud Rate Mode Control Register 0x101c0b34
UART1_BAUD_MODE_CTRL – UART1 Baud Rate Mode Control Register 0x101c0b74

31 ... 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.	R/W	0

UART0_RX_FIFO_IRQ_LVL – UART0 RX FIFO Trigger Level and RX-DMA Enable 0x101c0b38
UART1_RX_FIFO_IRQ_LVL – UART1 RX FIFO Trigger Level and RX-DMA Enable 0x101c0b78

31 … 6	5	4 … 0
reserved	RXDMA	RXIFLSEL

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x00
5	RXDMA	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	R/W	0
4:0	RFIRQLEVEL	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.	R/W	0x08

UART0_TX_FIFO_IRQ_LVL – UART0 TX FIFO Trigger Level and TX-DMA Enable 0x101c0b3c
UART1_TX_FIFO_IRQ_LVL – UART1 TX FIFO Trigger Level and TX-DMA Enable 0x101c0b7c

31 … 6	5	4 … 0
reserved	TXDMA	TXIFLSEL

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

9.5 SPI – Serial Peripheral Interface

The netX10 is equipped with one independent, DMA capable SPI unit. To allow the generation of netX10 code that is still compatible with the SPI unit of the netX100/500, a set of four additional legacy registers was implemented, located at 0x101406b0 – 0x101406bc. However, in order to fully use the features of the SPI unit, the new register sets (0x10140680 - 0x101406a8) must be used.

The following table shows a summary of SPI registers.

ARM Address	Register Name	Short Description
0x10140680	SPI_CTRL0	SPI Control Register 0
0x10140684	SPI_CTRL1	SPI Control Register 1
0x10140688	SPI_DATA	SPI Data Register
0x1014068c	SPI_STAT	SPI Status Register
0x10140690	SPI_CLK_PRE_SCL	SPI Clock Prescale Register
0x10140694	SPI_INT_MSK_SET_CLR	SPI Interrupt Mask Set or Clear Register
0x10140698	SPI_RAW_INT_STAT	SPI RAW Interrupt Status Register
0x1014069c	SPI_MASK_INT_STAT	SPI Masked Interrupt Status Register
0x101406a0	SPI_INT_CLR	SPI Interrupt Clear Register
0x101406a4	SPI_IRQ_CPU_SEL	Interrupt CPU Select Register
0x101406a8	SPI_DMA_CTRL	SPI DMA Control Register
0x101406b0	SPI_LGY_DATA	SPI Legacy Data Register
0x101406b4	SPI_LGY_STAT	SPI Legacy Status Register
0x101406b8	SPI_LGY_CTRL	SPI Legacy Control Register
0x101406bc	SPI_LGY_INT_CTRL	SPI Legacy Interrupt Control Register

SPI_ CTRL0 – SPI Control Register 0 **0x10140680**

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

31	30 29	28	27	26	25 24	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	DATASIZE

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	0x1
30:29	reserved	-	R	0x0
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	0x0
27	FILTER_IN	Receive-data is sampled every 10ns (100MHz system 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 '1's 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 the 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	0x0
26	reserved	-	R	0x0
25:24	FORMAT	frame format 00: Motorola SPI frame format 01..11: reserved	R/W	0x0
23:20	reserved	-	R	0x0
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]. Default value 0x800 equals 50MHz SPI clock rate. Note. If sck_muladd is set to zero, SPI transfer will freeze. 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	0x0
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	0x0
5:4	reserved	-	R	0x0
3:0	DATASIZE	DSS: data size select (transfer size = datasize + 1 bits) 0000...0010: reserved 0011: 4 bit 0100: 5 bit ... 0111: 8 bit ... 1111: 16 bit	R/W	0x7

SPI_ CTRL1 – SPI Control Register 1 **0x10140684**

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

31	30	29	28	27	26	25	24	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 demultiplexer 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

SPI_DATA – SPI Data Register 0x10140688

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

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 … 17	16 … 0
reserved	DATA

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16:0	DATA*	Transmit data, must be right aligned on writing, only bits according to spi_cr0.DSS are considered Receive data will be delivered right aligned, unused bits (spi_cr0.DSS < 0xF) will be "0". In slavemode transmit data is requested from the FIFO when the last bit of the current transfer-word ist set to spi_miso. If no next transimt data could be read from the FIFO until current words last bit was transfered, FIFO underrun will occure if FSS does not go inactive (last word was transfer end) at the next detected spi_sck-edge.	R/W	0x0

SPI_STAT – SPI Status Register 0x1014068c

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

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.
Show the current status of the spi interface.
Both receive and transmit FIFO have a depth of 16.

31	30	29	28–24	23	22	21	20–16	15–5	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 underrun error occured, data is lost	R	0x0
30	RX_FIFO_ERR_OVFL	extended: receive FIFO overflow error occured, data is lost	R	0x0
29	reserved	-	R	0x0
28:24	RX_FIFO_LEVEL	extended: receive FIFO level (number of received words to read out are left in FIFO)	R	0x0
23	TX_FIFO_ERR_UNDR	extended: transmit FIFO underrun error occured, data is lost	R	0x0
22	TX_FIFO_ERR_OVFL	extended: transmit FIFO overflow error occured, data is lost	R	0x0
21	-	reserved	R	0x0
20:16	TX_FIFO_LEVEL	extended: transmit FIFO level (number of words to transmit are left in FIFO)	R	0x0
15:5	reserved	-	R	0x0
4	BSY	device busy (1 if data is currently transmitted/received or the transmit FIFO is not empty)	R	0x0
3	RFF	receive FIFO is full (1 if full)	R	0x0
2	RNE	receive FIFO is not empty (0 if empty)	R	0x0
1	TNF	transmit FIFO is not full (0 if full)	R	0x0
0	TFE	transmit FIFO is empty (1 if empty)	R	0x0

SPI_CLK_PRE_SCL – SPI Clock Prescale Register 0x10140690

No clock prediviting is done.

31:8	7:0
reserved	CPSDVSR

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	CPSDVSR	obsolete	R/W	0x0

SPI_INT_MSK_SET_CLR – SPI Interrupt Mask Set or Clear 0x10140694

The value of this register is used for AND-masking the raw interrupt register. When a bit is set, the corresponding interrupt is routed to the interrupt controller. Different to the other netX modules, the SPI interrupt mask is written directly and not by set- and reset-masks.

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

Both receive and transmit FIFO have a depth of 16.

31:7	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	0x0
6	TXEIM	transmit FIFO empty interrupt mask (for netx100/500 compliance)	R/W	0x0
5	RXFIM	receive FIFO full interrupt mask (for netx100/500 compliance)	R/W	0x0
4	RXNEIM	receive FIFO not empty interrupt mask (for netx100/500 compliance)	R/W	0x0
3	TXIM	transmit FIFO interrupt mask	R/W	0x0
2	RXIM	receive FIFO interrupt mask	R/W	0x0
1	RTIM	receive timeout interrupt mask	R/W	0x0
0	RORIM	receive FIFO overrun interrupt mask	R/W	0x0

SPI_RAW_INT_STAT – SPI RAW Interrupt Status Register 0x10140698

This register holds the raw interrupt status before masking has been applied.
Interrupt is cleared by writing"1" to the according bit of SPI_INT_CLR register.
FIFO-state interrupts are cleared automatically if interrupt criteria is no longer true.

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

Both receive and transmit FIFO have a depth of 16.

31 … 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	0x0
6	TXERIS	unmasked transmit FIFO empty interrupt state (for netx100/500 compliance) 1: transmit FIFO is empty 0: transmit FIFO is not empty	R	0x0
5	RXFRIS	unmasked receive FIFO full interrupt state (for netx100/500 compliance) 1: receive FIFO is full 0: receive FIFO is not full	R	0x0
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	0x0
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	0x0
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	0x0
1	RTRIS	unmasked receive timeout interrupt state timeout period are 32 SPI-clock periods depending on adr_spi_cr0.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	0x0
0	RORRIS	unmasked receive FIFO overrun interrupt state 1: receive FIFO overrun error occurred 0: no receive FIFO overrun error occurred	R	0x0

SPI_MSK_INT_STAT – SPI Mask Interrupt Status Register 0x1014069c

If one of these bits is set, the USB device interrupt will be asserted to the interrupt controller.

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

Both receive and transmit FIFO have a depth of 16.

31 … 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	0x0
6	TXEMIS	masked transmit FIFO empty interrupt state (for netx100/500 compliance)	R	0x0
5	RXFMIS	masked receive FIFO full interrupt state (for netx100/500 compliance)	R	0x0
4	RXNEMIS	masked receive FIFO not empty interrupt state (for netx100/500 compliance)	R	0x0
3	TXMIS	masked transmit FIFO interrupt state	R	0x0
2	RXMIS	masked receive FIFO interrupt state	R	0x0
1	RTMIS	masked receive timeout interrupt state	R	0x0
0	RORMIS	masked receive FIFO overrun interrupt state	R	0x0

SPI_INT_CLR – SPI Interrupt Clear Register 0x101406a0

Interrupt is cleared by writing"1" to the according bit.
FIFO-state interrupts are cleared automatically if interrupt criteria is no longer true.

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500.

31 … 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	0x0
6	TXEIC	clear transmit FIFO empty interrupt (for netx100/500 compliance)	R/W	0x0
5	RXFIC	clear receive FIFO full interrupt (for netx100/500 compliance)	R/W	0x0
4	RXNEIC	clear receive FIFO not empty interrupt (for netx100/500 compliance)	R/W	0x0
3	TXIC	PL022 extention: clear transmit FIFO interrupt	R/W	0x0
2	RXIC	PL022 extention: clear receive FIFO interrupt	R/W	0x0
1	RTIC	clear receive FIFO overrun interrupt	R/W	0x0
0	RORIC	clear receive FIFO overrun interrupt writing '1' here will clear the receive FIFO	R/W	0x0

SPI_IRQ_CPU_SEL – Interrupt CPU Select Register 0x101406a4

Select CPU (xPIC or ARM), which gets Interrupts from this SPI instance.

r 30 … 2	1	0
reserved	XPIC	ARM

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	XPIC	Enable for IRQ signal to xPIC	R/W	0x0
0	ARM	Enable for IRQ signal to ARM	R/W	0x1

SPI_DMA_CTRL – SPI DMA Control Register 0x101406a8

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	0x0
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	0x0
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	0x0

Legacy Registers:

SPI_LGY_DATA – SPI Legacy Data Register 0x101406b0

Registers 0x101406b0 – 0x101406bc can be used instead of registers 0x10140680 - 0x101406a8 to keep netx10 software compliant to netx100/500. In netx50 and later versions both, receive and transmit FIFO have a depth of 16; fill-values are fixed to 4. To keep software compatible, not more than 8 bytes should be in netx100/500-FIFOs.

During write-access, DATA_BYTE_1 and DR_VALID1 must not be used. DR_VALID0 must be set.

31 … 18	17	16	15 … 8	7 … 0
reserved	DR_VALID1	DR_VALID0	DATA_BYTE_1	DATA_BYTE_0

Bits	Name	Description	R/W	Default
31:18	reserved	-	R	0x0
17	DR_VALID1	obsolete, always 0	R/W	0x0
16	DR_VALID0	valid bit for data_byte_0 This bit shows if data_byte_0 Is valid and must be set during FIFO write access	R/W	0x0
15:8	DATA_BYTE_1	obsolete, don't use	R/W	0x0
7:0	DATA_BYTE_0	data byte 0	R/W	0x0

SPI_LGY_STAT – SPI Legacy Status Register 0x101406b4

Show the actual status of the spi interface. Bits 23 and 21-18 show active interrupts, writing ones into these bits deletes the interrupts. Writing into other bits has no effect.

In netx50 and later versions both, receive and transmit FIFO have a depth of 16; fill-values are fixed to 4. To keep software compatible, not more than 8 bytes should be in netx100/500-FIFOs.

31 … 26	25	24	23	22	21	20	19	18	17 … 9	8 … 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	0x0
25	SR_SELECTED	external master has access to spi-interface	R	0x0
24	SR_OUT_FULL	output FIFO is full (no IRQ)	R/W	0x0
23	SR_OUT_EMPTY	output FIFO is empty (in slave mode)	R/W	0x0
22	SR_OUT_FW	ARM is writing data too fast into output FIFO (no IRQ)	R/W	0x0
21	SR_OUT_FUEL	adjustable fuel value of output FIFO reached	R/W	0x0
20	SR_IN_FULL	input FIFO is full	R/W	0x0
19	SR_IN_RECDATA	valid data bytes in input FIFO	R/W	0x0
18	SR_IN_FUEL	adjustable fill level of input FIFO reached	R/W	0x0
17:9	SR_OUT_FUEL_VAL	output FIFO fill vlaue (number of bytes)	R	0x0

Bits	Name	Description	R/W	Default
8:0	SR_IN_FUEL_VAL	input FIFO fill value (number of bytes)	R	0x0

SPI_LGY_CTRL – SPI Legacy Control Register 0x101406b8

31	30	29	28	27	26	25	24	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	1:enable 0:disable spi interface	R/W	0x0
30	CR_MS	1:master mode 0:slave mode	R/W	0x0
29	CR_CPOL	1:falling edge of spi_sck is primary 0:rising edge of spi_sck is primary	R/W	0x0
28	CR_NCPHA	SPI clock phase mode (Note: meaning of this bit is inverted to functionality of bit SPH in spi_cr0 register): 0:change data to secondary spi_sck edge data are active to primary spi_sck edge 1:change data to primary spi_sck edge data are active to secondary spi_sck edge	R/W	0x0
27:25	CR_BURST	netx100/netx500 only, obsolete in later versions: burst lenght = 2^CR_burst	R/W	0x0
24:22	CR_BURSTDELAY	netx100/netx500 only, obsolete in later versions: delay between transmission of 2 data bytes (0 to 7 SCK cycles)	R/W	0x0
21	CR_CLR_OUTFIFO	clear output FIFO	R/W	0x0
20	CR_CLR_INFIFO	clear input FIFO	R/W	0x0
19:12	reserved	-	R	0x0
11	CS_MODE	1: chip select is generated automatically by the internal state machine 0: chip select is directly controlled by software (see bits CR_ss).	R/W	0x0
10:8	CR_SS	external slave select	R/W	0x0
7	CR_WRITE	netx100/netx500 only, in later versions always "1": 1: enable spi interface write data	R/W	0x0
6	CR_READ	netx100/netx500 only, in later versions always "1": 1: enable spi interface read data	R/W	0x0
5	reserved	-	R	0x0
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 an may be incorrect. There are 16 different SPI Clocks to choose: 0000: 0,025 MHz Note: Not compatible to netx100/500. "0000" freezes spi_clk in netx100/500. 0001: 0,05 MHz 0010: 0,1 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	R/W	0x0

Bits	Name	Description	R/W	Default
		1110:25 MHz 1111:50 MHz		
0	CR_SOFTRESET	write only: no function in netx100/netx500; later Versions: clears IRQs and FIFOs	R/W	0x0

SPI_LGY_INT_CTRL – SPI Legacy Interrupt Control Register 0x101406bc

In netx50 and later versions both, receive and transmit FIFO have a depth of 16; fill-values are fixed to 4. To keep software compatible, not more than 8 bytes should be in netx100/500-FIFOs.

31 30 29 28 27 26 25	24	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	0x0
24	IR_OUT_FULL_EN	Obsolete	R/W	0x0
23	IR_OUT_EMPTY_EN	IRQ enable for output FIFO is empty and interface sending data (occurs only in slave mode)	R/W	0x0
22	IR_OUT_FW_EN	Obsolete	R/W	0x0
21	IR_OUT_FUEL_EN	IRQ enable for adjustable fuel value of output FIFO reached	R/W	0x0
20	IR_IN_FULL_EN	IRQ enable for input FIFO is full	R/W	0x0
19	IR_IN_RECDATA_EN	IRQ enable for valid data bytes in input FIFO	R/W	0x0
18	IR_IN_FUEL_EN	IRQ enable for adjustable fill level of input FIFO reached	R/W	0x0
17:9	IR_OUT_FUEL	Watermark level for output FIFO	R/W	0x0
8:0	IR_IN_FUEL	Watermark level for input FIFO	R/W	0x0

9.6 SQI – Serial Quad I/O

The netX10 SQI (Serial Quad I/O) module provides standard SPI with one receive and one transmit data line as well as 2 bit Dual SPI and Serial Quad IO (SQI, i.e. Quad SPI). In all modes master functionality is implemented. Slave functionality is not available.

Standard SPI functionality and programming is compatible to netX50 SPI master however netX100 software compatibility is not longer supported by this module. Several new features like dummy cycles or half duplex modes necessary for SQI are also available for standard SPI.

For SQI devices a high speed eXecute-in-Place (XiP, SQIROM) mode is implemented. In this mode data from a SQI device is readable similar to a linear external memory (e.g a parallel FLASH device) for all netX system masters. A dedicated 16MByte address area (0x09000000~0x09FFFFFF) is provided therefore. In this mode CPU code can be run directly from an SQI device without being copied into RAM before.

Before SQIROM function can be used, SQIROM mode must be enabled and configured inside the SQI_SQIROM_CFG register. Before doing this, the SQI device must be initialized to 4-bit Read mode. Only SPI modes 0 and 3 are supported for SQIROM usage.

Standard SPI and SQI transfer generation is not available when SQIROM mode is enabled. SQIROM function is only available for devices connected to chip-select signal 0.

The following table shows a summary of SQI module registers.

ARM Address	Register Name	Short Description
0x101c0d00	SQI_CTRL0	SQI Control Register 0
0x101c0d04	SQI_CTRL1	SQI Control Register 1
0x101c0d08	SQI_DATA	SQI Data Register
0x101c0d0c	SQI_STAT	Read Only SQI Status Register
0x101c0d10	SQI_TCR	SQI Transfer Control
0x101c0d14	SQI_IRQ_MASK	SQI Interrupt Mask Set or Clear Register
0x101c0d18	SQI_IRQ_RAW	SQI Interrupt State Before Masking Register (Raw Interrupt)
0x101c0d1c	SQI_IRQ_MASKED	SQI Masked Interrupt Status Register
0x101c0d20	SQI_IRQ_CLEAR	SQI Interrupt Clear Register (For Compatibility To NetX10/50 SPI Module)
0x101c0d24	SQI_IRQ_CPU_SEL	SQI Interrupt CPU Select Register
0x101c0d28	SQI_DMACR	SQI DMA Control Register
0x101c0d30	SQI_PIO_OUT	SQI PIO Output Level Control Register
0x101c0d34	SQI_PIO_OE	SQI PIO Output Enable Control Register
0x101c0d38	SQI_PIO_IN	SQI PIO Input Status Register
0x101c0d3c	SQI_SQIROM_CFG	SQIROM Mode Configuration

SQI_CTRL0 – SQI Control Register 0 0x101c0d00

This register is compatible to netX50 and netX10 SPI module. However, there are some additional settings possible. SQI module is provides only master functionality, hence slave settings are omitted. Compatible mode for netx100 is not supported by SQI module.

31:28	27	26:24	23:22	21:20	19:8	7	6	5:4	3:0
reserved	FILTER_IN	reserved	SIO_CFG	reserved	SCK_MULADD	SCK_PHASE	SCK_POL	reserved	DATASIZE

Bits	Name	Description	R/W	Default
31:28	reserved	-	R	0x0
27	FILTER_IN	Receive-data is sampled every 10ns (100MHz system 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 sck clock edge (if two or more ’1s! were sampled a ’1’ will be stored, else a ’0’ will be stored. 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	0x0
26:24	reserved	-	R	0x0
23:22	SIO_CFG	SQI IO configuration. Default is all IOs are in PIO input mode. Coding 00: only SIO2,3 are controllable as PIOs (2-bit SPI or Standard Motorola SPI), 01: all SQP IOs are used for transfers (4-bit SPI/SQI). 10: reserved 11: all SQI IOs are controllable as PIOs	R/W	0x0
21:20	reserved	-	R	0x0
19:8	SCK_MULADD	serial clock rate multiply add value for sck generation. sck-frequency: f_sck = (sck_muladd * 100)/4096 [MHz]. Default value 0x800 equals 50MHz clock rate. Note: If sck_muladd is set to zero, transfer will freeze. Note: SQIROM (XiP) serial clock rate ust be programmed in ’sqi_sqirom_cfg’ register.	R/W	0x800
7	SCK_PHASE	serial clock phase 1: sample data at second clock edge, data is generated half a clock phase before sampling 0: sample data at first clock edge, data is generated half a clock phase before sampling	R/W	0x0
6	SCK_POL	serial clock polarity 0: idle: clock is low, first edge is rising 1: idle: clock is high, first edge is falling	R/W	0x0
5:4	reserved	-	R	0x0
3:0	DATASIZE	data size select for standard Motorola SPI mode. This bit field is unused in 2- and 4-bit SPI modes (running always byte transfers). (transfer size = datasize + 1 bits) 0000...0010: reserved 0011: 4 bit 0100: 5 bit ... 0111: 8 bit ... 1111: 16 bit	R/W	0x7

SQI_CTRL1 – SQI Control Register 1 0x101c0d04

This register is compatible to netX50 and netX10 SPI module. However, there are some additional settings possible. SQI module is provides only master functionality, hence slave settings are omitted.

31:29	28	27:24	23:21	20	19:16	15:13	12	11	10:8	7:2	1	0
reserved	RX_FIFO_CLR	RX_FIFO_WM	reserved	TX_FIFO_CLR	TX_FIFO_WM	reserved	SPI_TRANS_CTRL	FSS_STATIC	FSS	reserved	SQI_EN	reserved

Bits	Name	Description	R/W	Default
31:29	reserved	-	R	0x0
28	RX_FIFO_CLR	Writing ”1” to this bit will clear the receive-FIFO. This bit will be reset automatically by hardware. It is always ’0’ on read.	R/W	0x0
27:24	RX_FIFO_WM	receive FIFO watermark for IRQ-generation. If receive FIFO watermark IRQ is enabled (’RXIM’ bit is set in ’sqi_irq_mask’ register), transfers will be stopped when receive FIFO runs full. Transfers will be continued after receive data is read from receive FIFO. This is done to avoid receive FIFO overflows. If receive FIFO watermark IRQ is disabled (’RXIM’ bit is not set in ’sqi_irq_mask’ register), transfers will not be stopped when receive FIFO runs full. In this case receive FIFO overrun could occur. This is compatible to netX50 behavior and allows writing data in full duplex mode without reading receive FIFO.	R/W	0x8
23:21	reserved	-	R	0x0
20	TX_FIFO_CLR	Writing ”1” to this bit will clear the transmit-FIFO. This bit will be reset automatically by hardware. It is always ’0’ on read.	R/W	0x0
19:16	TX_FIFO_WM	transmit FIFO watermark for IRQ-generation	R/W	0x8
15:13	reserved	-	R	0x0
12	SPI_TRANS_CTRL	Transfer Control for standard Motorola SPI (default: disabled) This bit is only used for for standard Motorola SPI (register ’sqi_tcr’ ’mode’-bits) in full duplex and half duplex transmit mode. If this bit is set, SPI transfer then is controlled by ’start_transfer’ and ’transfer_size’ of register ’sqi_tcr’. If this bit is not set (default), SPI transfers start immediately after transfer data was written to TX FIFO (this is SPI module compatible). Settings of ’start_transfer’ and ’transfer_size’ of register ’sqi_tcr’ then remain unaffected and are ignored. If this bit is set and SPI is used in receive mode (full duplex or half duplex receive mode set by bit field ’duplex’ in register ’sqi_tcr’), transfers will be stopped when receive FIFO runs full. Transfers will be continued after receive data is read from receive FIFO. This is done to avoid receive FIFO overflows.	R/W	0x0
11	FSS_STATIC	SQI static chipselect 0: chipselect will be generated automatically at data frame begin/end according to fss and datasize 1: chipselect will be set statically according to fss If fss is set to statically, fss must be toggled manually after each data frame in Motorola SPI mode when sck_phase is 0 for spec compatibility!	R/W	0x0
10:8	FSS	Frame slave select (up to 3 devices can be assigned directly, up to 8 devices can be assigned if an external demultiplexer is used). This signal is active low, so the bits will be inverted before output to the SQI pins.	R/W	0x0
7:2	reserved	-	R	0x0

Bits	Name	Description	R/W	Default
1	SQI_EN	SQI enable. 0: interface disabled 1: interface enabled Note: Standard SQI/SPI function is not available if SQIROM/XiP function is selected by 'enable' bit of 'sqi_sqirom_cfg' register (see description of 'sqi_sqirom_cfg' register).	R/W	0x0
0	reserved	-	R	0x0

SQI_DATA – SQI Data Register (DR) 0x101c0d08

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.

31–0
DATA

Bits	Name	Description	R/W	Default
31:0	DATA	Transmit data, must be right aligned on writing. In Standard SPI mode only bits according to SQI_CTRL0.datasize are being transferred. In SQI mode data must be written in full DWords (i.e. software needs to collect four bytes prior to writing). Unused bytes won't be transferred and may be padded at will (number of transfered bytes depends on sqi_tcr.transfer_size). Receive data will be delivered right aligned in both modes, unused bits will be "0".	R/W	0x0

SQI_STAT – Read Only SQI Status Register 0x101c0d0c

SQI master mode: MISO-input-data will be stored in the receive FIFO, transmit FIFO generates MOSI-output-data. SQI_STAT register shows the current status of the SQI interface.

31	30	29	28	27	26	25	24	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					SQIROM_DISABLED_ERR	SQIROM_WRITE_ERR	SQIROM_TIMEOUT_ERR	reserved								BUSY	RX_FIFO_FULL	RX_FIFO_NOT_EMPTY	TX_FIFO_NOT_FULL	TX_FIFO_EMPTY

Bits	Name	Description	R/W	Default
31	RX_FIFO_ERR_UNDR	Receive FIFO underrun error occurred, data is lost. This status flag is cleared by clearing RX FIFO ('SQI_CTRL1' register).	R	0x0
30	RX_FIFO_ERR_OVFL	Receive FIFO overflow error occurred, data is lost. This status flag is cleared by clearing RX FIFO ('SQI_CTRL1' register).	R	0x0
29	reserved	-	R	0x0
28:24	RX_FIFO_LEVEL	Receive FIFO level (number of received words to read out are left in FIFO).	R	0x0
23	TX_FIFO_ERR_UNDR	Transmit FIFO underrun error occurred, data is lost. This status flag is cleared by clearing TX FIFO ('SQI_CTRL1' register).	R	0x0
22	TX_FIFO_ERR_OVFL	Transmit FIFO overflow error occurred, data is lost. This status flag is cleared by clearing TX FIFO ('SQI_CTRL1' register).	R	0x0
21	-	reserved	R	0x0
20:16	TX_FIFO_LEVEL	Transmit FIFO level (number of words to transmit are left in FIFO).	R	0x0
15	SQIROM_DISABLED_ERR	Access to SQIROM area detected while SQIROM was disabled. To enable SQIROM functionality set 'enable' bit in 'sqi_sqirom_cfg' register. This bit can be used to determine the reason for 'sqirom_error' IRQ assertion. This status flag is only cleared by writing a '1' here.	R	0x0
14	SQIROM_WRITE_ERR	Write access to SQIROM area detected. SQIROM area is read only. This bit can be used to determine the reason for 'sqirom_error' IRQ assertion. This status flag is only cleared by writing a '1' here.	R	0x0
13	SQIROM_TIMEOUT_ERR	Timeout during SQIROM area read detected. A timeout results from a fix level on netX serial clock IO. Check IO multiplexing configuration and ensure that serial clock output is not clamped. This bit can be used to determine the reason for 'sqirom_error' IRQ assertion. This status flag is only cleared by writing a '1' here. SQIROM function must be disabled and enabled again to reset module internal state machines after this bit has ben set (register 'sqirom_cfg', reset and set again 'enable' bit).	R	0x0
12:5	reserved	-	R	0x0

Bits	Name	Description	R/W	Default
4	BUSY	Device is busy (1 if data is currently transmitted/received or the transmit FIFO is not empty).	R	0x0
3	RX_FIFO_FULL	Receive FIFO is full (1 if full).	R	0x0
2	RX_FIFO_NOT_EMPTY	Receive FIFO is not empty (0 if empty).	R	0x0
1	TX_FIFO_NOT_FULL	Transmit FIFO is not full (0 if full).	R	0x0
0	TX_FIFO_EMPTY	Transmit FIFO is empty (1 if empty).	R	0x0

SQI_TCR – SQI Transfer Control 0x101c0d10

Module address offset 0x10 is reserved in netX10/50 SPI module. No compatibility problems by using this address for new register.

TBD: This register is not writable while a transfer is running ('busy' bit in register 'SQI_STAT' is '1') to avoid corrupted transfers causing hardware damage.

31 30	29	28	27 26	25 24	23	22	21	20 19	18 … 0
reserved	MS_BYTE_FIRST	MS_BIT_FIRST	DUPLEX	MODE	START_TRANSFER	TX_OE	TX_OUT	reserved	TRANSFER_SIZE

Bits	Name	Description	R/W	Default
31:30	reserved	-	R	0x0
29	MS_BYTE_FIRST	Most significant byte first 2- and 4-bit mode: Writing "1" to this bit will use most significant byte first in DWords In Standard Motorola SPI mode this bit is ignored. Endianess of a transferred 32-bit word can be controlled by this bit. Default 0 is little endianess.	R/W	0x0
28	MS_BIT_FIRST	Most significant bit first 2- and 4-bit mode: Writing "1" to this bit will transfer most significant bit first In Standard Motorola SPI mode this bit is ignored.	R/W	0x1
27:26	DUPLEX	Transfer type selection (default is '11' for SPI compatibility). 00: dummy. Generates 'transfer_size' + 1 serial clock periods. No change of RX and TX FIFOs. Data lines (standard Motorola SPI mode: SPI_MOSI) are controlled by 'tx_oe' and 'tx_out'. 01: half duplex receive Receives 'transfer_size' + 1 words. In 2-bit and 4-bit mode TX-FIFO will be cleared and are not available during reveive. In standard SPI mode SPI_MOSI is controlled by 'tx_oe' and 'tx_out'. There is no need to fill the TX-FIFO with dummy TX-data to receive RX-data. TX FIFOs are not changed and are always available. 10: half duplex transmit Transmits 'transfer_size' + 1 words. In 2-bit and 4-bit mode RX-FIFO will be cleared and are not available during receive. In standard SPI mode SPI_MISO input is ignored. RX-FIFO is available and remains unchanged. 11: full duplex (Standard Motorola SPI mode only, reserved in 2-bit and 4-bit modes) This is full duplex standard Motorola SPI mode always transmitting and receiving data. Transmit data is taken from TX-FIFO, receive data is stored in RX-FIFO. Note: If '11' is set in 2-bit or 4-bit mode, this is treated as 'receive' (like '01' setting). Note: If there was a FIFO error (overrun, underrun) before changing to '01' or '10' the FIFO error status bits in register SQI_STAT_ are not cleared by half duplex modes FIFO clearing.	R/W	0x3
25:24	MODE	SPI/SQI Mode selection 00: Standard Motorola SPI mode. 01: 2-bit SPI mode 10: 4-bit SPI mode 11: reserved	R/W	0x0
23	START_TRANSFER	Transfer start signal Writing a "1" starts the transfer of transfer_size bytes.	R/W	0x0

Bits	Name	Description	R/W	Default
		Also starts transfer of dummy cycles. This bit will be reset automatically by hardware and is always ’0’ on read. It is only writable after a transfer sequence is finished or if the transfer sequence is terminated by a FIFO clear. Note: A transfer sequence is finished completely when ’busy’ bit in SQI_STAT register is not set. Note: For standard Motorola SPI mode, this function can be controlled by ’spi_trans_ctrl’ bit in SQI_CTRL1 register (for SPI module compatibility).		
22	TX_OE	Output driver enable in dummy or standard SPI receive-only mode Writing a ”1” enables the output drivers of the data pins in dummy mode.	R/W	0x0
21	TX_OUT	Output level in dummy or standard SPI receive-only mode. This bit selects the output level when the output driver is enabled in dummy mode.	R/W	0x0
20:19	reserved	-	R	0x0
18:0	TRANSFER_SIZE	Number of bytes within the current SQI transaction (transfer_size+1). Program (actual number of bytes - 1) in SQI modes or (number of dummy clock cycles - 1) Example: 0x00000: one byte / dummy cycle ... 0x7ffff: 512k bytes / dummy cycles This bit field counts down during transfer with each transferred word/byte or dummy cycle. It is only writable after a transfer sequence is finished or if the transfer sequence is terminated by a FIFO clear. Hence, it is writable but can also be changed by hardware. A running transfer sequence can be terminated by FIFO clearing (register SQI_CTRL1). This may be necessary if a read sequence has to be terminated. Example: A half duplex write transfer of 128k bytes was programmed but there is not enough write data. To terminate this write sequence, clear TX FIFO. If there is an external transfer running at the moment of clearing the FIFO, this transfer will not be broken and finished with the last bit to be transferred. Note: A transfer sequence is finished completely when ’busy’ bit in SQI_STAT register is not set. Note: In 4-bit SQI mode it is only allowed to programm 1 to 4 bytes or sizes in multiples of full DWords. So, valid values in 4-bit mode are: 0, 1, 2, 3, 7, 11, ..., (4n - 1)	R/W	0x0

SQI_IRQ_MASK – SQI Interrupt Mask Set or Clear Register 0x101c0d14

The value of this register is used for AND-masking the raw interrupt register. When a bit is set, the corresponding interrupt is routed to the interrupt controller. Different to the other netX modules, the SQI-IRQ mask is written directly and not by set- and reset-masks. For a detailed IRQ description, see SQI_IRQ_RAW register.

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="23">reserved</td><td>SQIROM_ERROR</td><td>TRANS_END</td><td>TXEIM</td><td>RXFIM</td><td>RXNEIM</td><td>TXIM</td><td>RXIM</td><td>RTIM</td><td>RORIM</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	SQIROM_ERROR	SQIROM error interrupt mask	R/W	0x0
7	TRANS_END	transfer end interrupt mask	R/W	0x0
6	TXEIM	transmit FIFO empty interrupt mask (for netx100/500 compliance)	R/W	0x0
5	RXFIM	receive FIFO full interrupt mask (for netx100/500 compliance)	R/W	0x0
4	RXNEIM	receive FIFO not empty interrupt mask (for netx100/500 compliance)	R/W	0x0
3	TXIM	transmit FIFO interrupt mask	R/W	0x0
2	RXIM	receive FIFO interrupt mask	R/W	0x0
1	RTIM	receive timeout interrupt mask	R/W	0x0
0	RORIM	receive FIFO overrun interrupt mask	R/W	0x0

SQI_IRQ_RAW – SQI Interrupt State Before Masking Register (Raw Interrupt) 0x101c0d18

This register holds the raw interrupt status before masking has been applied. Writing '1' will clear the corresponding interrupt.

Note:
Both, receive and transmit FIFO have a depth of 16.
Note:
IRQ flags can also be cleared by using SQI_IRQ_CLEAR register for SPI module compatibility.

31:9	8	7	6	5	4	3	2	1	0
reserved	SQIROM_ERROR	TRANS_END	TXERIS	RXFRIS	RXNERIS	TXRIS	RXRIS	RTRIS	RORRIS

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	SQIROM_ERROR	SQIROM error interrupt state 1: SQIROM access error detected. This IRQ is asserted when an error occurs on a SQIROM access. Detailed error information is provided by SQIROM error bit in register SQI_STAT. For error handling both, this IRQ bit and bits in register SQI_STAT must be cleared. 0: no SQIROM error detected.	R/W	0x0
7	TRANS_END	unmasked transfer end interrupt state (related to Bit 'busy' of SQI_STAT register) 1: transfer finished. Bit 'busy' of SQI_STAT register has become inactive. 0: transfer finished not finished. Bit 'busy' of SQI_STAT register is active.	R/W	0x0
6	TXERIS	unmasked transmit FIFO empty interrupt state (for netx100/500 compliance) 1: transmit FIFO is empty 0: transmit FIFO is not empty	R/W	0x0
5	RXFRIS	unmasked receive FIFO full interrupt state (for netx100/500 compliance) 1: receive FIFO is full 0: receive FIFO is not full	R/W	0x0
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/W	0x0
3	TXRIS	unmasked transmit FIFO interrupt state 1: transmit FIFO level is below SQI_CTRL1.tx_fifo_wm 0: transmit FIFO equals or is higher than SQI_CTRL1.tx_fifo_wm	R/W	0x0
2	RXRIS	unmasked receive FIFO interrupt state 1: receive FIFO is higher than SQI_CTRL1.rx_fifo_wm 0: receive FIFO is equals or is below SQI_CTRL1.rx_fifo_wm Note: View description of register SQI_CTRL1 for bits 'spi_trans_ctrl' and 'rx_fifo_wm' for receive FIFO behavior before programming this IRQ.	R/W	0x0
1	RTRIS	unmasked receive timeout interrupt state timeout period are 32 SQI-clock periods depending on SQI_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/W	0x0
0	RORRIS	unmasked receive FIFO overrun interrupt state 1: receive FIFO overrun error occurred 0: no receive FIFO overrun error occurred	R/W	0x0

SQI_IRQ_MASKED – SQI Masked Interrupt Status Register 0x101c0d1c

If one of these bits is set, the SQI device interrupt will be asserted to the interrupt controller. For a detailed IRQ description, view SQI_IRQ_RAW register.

31:9	8	7	6	5	4	3	2	1	0
reserved	SQIROM_ERROR	TRANS_END	TXEMIS	RXFMIS	RXNEMIS	TXMIS	RXMIS	RTMIS	RORMIS

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	SQIROM_ERROR	masked SQIROM error interrupt state	R	0x0
7	TRANS_END	masked transfer end interrupt state	R	0x0
6	TXEMIS	masked transmit FIFO empty interrupt state (for netx100/500 compliance)	R	0x0
5	RXFMIS	masked receive FIFO full interrupt state (for netx100/500 compliance)	R	0x0
4	RXNEMIS	masked receive FIFO not empty interrupt state (for netx100/500 compliance)	R	0x0
3	TXMIS	masked transmit FIFO interrupt state	R	0x0
2	RXMIS	masked receive FIFO interrupt state	R	0x0
1	RTMIS	masked receive timeout interrupt state	R	0x0
0	RORMIS	masked receive FIFO overrun interrupt state	R	0x0

SQI_IRQ_CLEAR – SQI Interrupt Clear Register (for Compatibility to netX10/50 SPI Module) 0x101c0d20

This register is always '0' on read.
Note:
IRQ flags can also be cleared by writing SQI_IRQ_RAW register.

31 … 9	8	7	6	5	4	3	2	1	0
reserved	SQIROM_ERROR	TRANS_END	TXEIC	RXFIC	RXNEIC	TXIC	RXIC	RTIC	RORIC

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	SQIROM_ERROR	clear SQIROM error interrupt	R/W	0x0
7	TRANS_END	clear transfer end interrupt	R/W	0x0
6	TXEIC	clear transmit FIFO empty interrupt (for netx100/500 compliance)	R/W	0x0
5	RXFIC	clear receive FIFO full interrupt (for netx100/500 compliance)	R/W	0x0
4	RXNEIC	clear receive FIFO not empty interrupt (for netx100/500 compliance)	R/W	0x0
3	TXIC	PL022 extension: clear transmit FIFO interrupt	R/W	0x0
2	RXIC	PL022 extension: clear receive FIFO interrupt	R/W	0x0
1	RTIC	clear receive FIFO overrun interrupt	R/W	0x0
0	RORIC	clear receive FIFO overrun interrupt writing '1' here will clear the receive FIFO	R/W	0x0

SQI_IRQ_CPU_SEL – SQI Interrupt CPU Select Register 0x101c0d24

Select CPU (xPIC or ARM), which gets Interrupts from this SQI instance.

31 … 2	1	0
reserved	XPIC	ARM

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	XPIC	Enable for IRQ signal to xPIC	R/W	0x0
0	ARM	Enable for IRQ signal to ARM	R/W	0x1

SQI_DMACR – SQI DMA Control Register 0x101c0d28

Only single transfer requests will be generated by this module.

31:2	1	0
reserved	TX_DMA_EN	RX_DMA_EN

Bits	Name	Description	R/W	Default
31:2	reserved	-	R	0x0
1	TX_DMA_EN	enable DMA for SQI-transmit data A request will be generated if TX-FIFO is not full and SQI_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	0x0
0	RX_DMA_EN	Enable DMA for SQI-receive data A request will be generated if RX-FIFO is not empty and SQI_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	0x0

SQI_PIO_OUT – SQI PIO Output Level Control Register 0x101c0d30

IO PIO mode is controlable by SQI_CTRL0 register bits 'sio_cfg'.
PIO input signal states are never filtered (SQI_CTRL0 bit 'filter_in').

Note:
SQI module must be enabled by register SQI_CTRL0 bit 'sqi_en' for SQI IOs driving in PIO mode.

31:6	5	4	3	2	1	0
reserved	SIO3	SIO2	MISO	MOSI	CSN	SCLK

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5	SIO3	SIO3 output state	R/W	0x0
4	SIO2	SIO2 output state	R/W	0x0
3	MISO	MISO/SIO1 output state	R/W	0x0
2	MOSI	MOSI/SIO0 output state	R/W	0x0
1	CSN	Chip-select/FSS 0 output state Note: Chip-select/FSS 1 and 2 are only optional netX MMIO signals and can not be controlled here. Use MMIO_CTRL registers.	R/W	0x0
0	SCLK	Serial SPI Clock output state.	R/W	0x0

SQI_PIO_OE – SQI PIO Output Enable Control Register 0x101c0d34

IO PIO mode is controlable by SQI_CTRL0 register bits 'sio_cfg'.

Note:
SQI module must be enabled by register SQI_CTRL0 bit 'sqi_en' for SQI IOs driving in PIO mode.

31 … 6	5	4	3	2	1	0
Reserved	SIO3	SIO2	MISO	MOSI	CSN	SCLK

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5	SIO3	SIO3 output enable	R/W	0x0
4	SIO2	SIO2 output enable	R/W	0x0
3	MISO	MISO/SIO1 output enable	R/W	0x0
2	MOSI	MOSI/SIO0 output enable	R/W	0x0
1	CSN	Chip-select/FSS 0 output enable Note: Chip-select/FSS 1 and 2 are only optional netX MMIO signals and can not be controlled here. Use MMIO_CTRL registers.	R/W	0x0
0	SCLK	Serial SPI Clock output enable	R/W	0x0

SQI_PIO_IN – SQI PIO Input Status Register 0x101c0d38

IO PIO mode is controllable by SQI_CTRL0 register bits 'sio_cfg'.

31 … 6	5	4	3	2	1	0
reserved	SIO3	SIO2	MISO	MOSI	CSN	SCLK

Bits	Name	Description	R/W	Default
31:6	reserved	-	R	0x0
5	SIO3	SIO3 input state	R	0x0
4	SIO2	SIO2 input state	R	0x0
3	MISO	MISO/SIO1 input state	R	0x0
2	MOSI	MOSI/SIO0 input state	R	0x0
1	CSN	Chip-select/FSS 0 input state Note: Chip-select/FSS 1 and 2 are only optional netX MMIO signals and can not be controlled here. Use MMIO_CTRL registers.	R	0x0
0	SCLK	Serial SPI Clock input state	R	0x0

SQI_SQIROM_CFG – SQIROM Mode Configuration Register 0x101c0d3c

This mode supports the 'eXecute in Place' (XiP) feature of SQI flash chips. The position of the command byte and the address nibbles as well as the number of address nibbles and dummy cycles can be configured with this register. It is also possible to change the clock divider to support a wide range of frequencies for the serial clock output.

Note: Before enabling this mode, the SQI flash chip needs to be in 4 bit command mode, otherwise the module is not able to fetch data from the flash.
Note: When enabled, the SQI module is completely blocked, e.g. other SQI or SPI transactions are not possible.
Note: The chip select signal of the flash must be connected to sqi_cs0.

31–24	23–22	21–20	19	18–16	15–8	7	6–4	3–2	1	0
CLK_DIV_VAL	reserved	T_CSH	reserved	DUMMY_CYCLES	CMD_BYTE	reserved	ADDR_BITS	ADDR_NIBBLES	ADDR_BEFORE_CMD	ENABLE

Bits	Name	Description	R/W	Default
31:24	CLK_DIV_VAL	clk400 will be divided by (clk_div_val+3) for sqirom_clk generation Default setting '2' is 80MHz. Maximum serial clock rate (programming '0') is 133MHz. Serial clock period (t_sck) will be (clk_div_val+3)*2.5ns. Clock high and low Phase will be generated symmetrical.	R/W	0x2
23:22	reserved	-	R	0x0
21:20	T_CSH	Minimum SQI Chips-select-high (idle) time: (t_csh+1) * t_sck (according to clk_div_val). Programmable values are 0 to 3. Change this parameter if used SQI device requires minimum Chips-select-high times exceeding 1 serial clock period. Required timing must be taken from used SQI device datasheet. Note: Serial clock will not toggle when device is not selected. Hence only Chip-select-active timing has to be considered.	R/W	0x0
19	reserved	-	R	0x0
18:16	DUMMY_CYCLES	Selects the number of dummy cycles before data will be sampled from the SQI chip. 000 : 0 cycles 001 : 1 cycle 010 : 2 cycles (default) ... 111 : 7 cycles	R/W	0x2
15:8	CMD_BYTE	This byte is transferred to the SQI chip as the command sequence. The address-command order can be controlled by the 'addr_before_cmd' bit.	R/W	0x0
7	reserved	-	R	0x0
6:4	ADDR_BITS	Number of address bits used to generate the address for the SQI chip. This depends on the size of the SQI chip. 000 : 20 bits (1MByte/8MBit device) (default) 001 : 21 bits (2MByte/16MBit device) 010 : 22 bits (4MByte/32MBit device) 011 : 23 bits (8MByte/64MBit device) 100 : 24 bits (16MByte/128MBit device) 101 - 111 : reserved	R/W	0x0
3:2	ADDR_NIBBLES	Number of nibbles to transfer as the address to the SQI chip. This depends on the command format of the SQI chip. The address-command order can be controlled by the 'addr_before_cmd' bit. Most significant address bits will be transmitted in the first	R/W	0x1

Bits	Name	Description	R/W	Default
		address nibble. Least significant address bits will be transmitted in the last address nibble. 00 : 5 nibbles 01 : 6 nibbles (default) 10 : 7 nibbles 11 : 8 nibbles		
1	ADDR_BEFORE_CMD	When set to '1' the address nibbles will be transferred before the command byte. Otherwise the address is transferred first.	R/W	0x0
0	ENABLE	Enables the SQIROM mode of the SQI module. The SQI chip needs to be initialized to accept 4 bit commands before activating the SQIROM mode. Note: This bit is also used to switch between SQIROM/XiP and standard SQI/SPI function. If this bit is set, standard SQI/SPI function is not available. SQIROM/XiP function does not depend on programmed value of 'sqi_en' bit in SQI_CTRL1 register.	R/W	0x0

9.7 I2C – Serial I2C-Interface

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

ARM Address	Register Name	Short Description
0x101c0d40	I2C_MASTER_CTRL	I2C Master Control Register
0x101c0d44	I2C_SLAVE_CTRL	I2C Slave Control Register
0x101c0d48	I2C_MASTER_CMD	I2C Master Command Register
0x101c0d4c	I2C_MASTER_DATA	I2C Master Data Register
0x101c0d50	I2C_SLAVE_DATA	I2C Slave Data Register
0x101c0d54	I2C_MASTER_FIFO_CTRL	I2C Master FIFO Control Register
0x101c0d58	I2C_SLAVE_FIFO_CTRL	I2C Slave FIFO Control Register
0x101c0d5c	I2C_STAT	I2C Status Register
0x101c0d60	I2C_INT_MSK_SET_CLR	I2C Interrupt Mask Set or Clear Register
0x101c0d64	I2C_RAW_INT_STAT	I2C RAW Interrupt Status Register
0x101c0d68	I2C_MSK_INT_STAT	I2C Mask Interrupt Status Register
0x101c0d6c	I2C_DMA_CTRL	I2C DMA Control Register
0x101c0d70	I2C_PIO	Direct I2C IO Access Controlling

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

I2C_MASTER_CTRL – I2C Master Control Register 0x101c0d40

31:17	16	15:11	10:4	3:1	0
reserved	PIO_MODE	reserved	SADR	MODE	EN_I2C

Bits	Name	Description	R/W	Default
31:17	reserved	-	R	0x0
16	PIO_MODE	If this bit is set, SCL and SDA will be directly controllable by i2c_pio register to access non I2C compatible devices. In pio-mode the I2C-controller statemachine is disabled, so no FIFO-action is done no IRQs occur and no DMA-controlling is possible.	R/W	0x0
15:11	reserved	-	R	0x0
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	0x0
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	0x0
0	EN_I2C	0 : interface disable 1 : interface enable	R/W	0x0

I2C_SLAVE_CTRL – I2C Slave Control Register 0x101c0d44

31:21	20	19	18	17	16	15:11	10	9:0
reserved	AUTORESET_AC_START	reserved	AC_GCALL	AC_START	AC_SRX	reserved	SID10	SID

Bits	Name	Description	R/W	Default
31:21	reserved	-	R	0x0
20	AUTORESET_AC_START	auto reset ac_start (ac_start must be set again after any (r)START) 0: ac_start will not be automatically reset (netx50 compatible) 1: reset ac_start after this slave acknowledged a starts sequence	R/W	0x0
19	reserved	-	R	0x0
18	AC_GCALL	General Call acknowledge: 0: do not generate acknowledge after General Call 1: generate acknowledge after General Call	R/W	0x0
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. If autoreset_ac_start is enabled, this bit will reset automatically by the controller. If autoreset_ac_start is not 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. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
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	0x0
15:11	reserved	-	R	0x0
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	0x0
9:0	SID	Slave device ID: External masters can address this device by this address.	R/W	0x0

I2C_MASTER_CMD – I2C Master Command Register 0x101c0d48

31–28	27–20	19–18	17–8	7–4	3–1	0
reserved	ACPOLLMAX	reserved	TSIZE	reserved	CMD	NWR

Bits	Name	Description	R/W	Default
31:28	reserved	-	R	0x0
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. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
19:18	reserved	-	R	0x0
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. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
7:4	reserved	-	R	0x0
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 generates 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 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). Note: This bit is writable but can also be changed by hardware.	R/W	0x7
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	0x0

I2C_MASTER_DATA – I2C Master Data Register (Master FIFO) 0x101c0d4c

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).

31:8	7:0
reserved	MDATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
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	0x0

I2C_SLAVE_DATA – I2C Slave Data Register (Slave FIFO) 0x101c0d50

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 sent again by the master.

31:8	7:0
reserved	SDATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	SDATA	I2C slave transmit or receive data: i2c_slave_ctrl.ac_start must be handled correctly by software to avoid FIFO errors after (r)START.	R/W	0x0

I2C_MASTER_FIFO_CTRL – I2C Master FIFO Control Register 0x101c0d54

31 … 9	8	7 … 4	3 … 0
reserved	MFIFO_CLR	reserved	MFIFO_WM

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	MFIFO_CLR	Clear master data FIFO, write only bit. Note: This bit is writable but can also be changed by hardware.	W	0x0
7:4	reserved	-	R	0x0
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	0x0

I2C_SLAVE_FIFO_CTRL – I2C Slave FIFO Control Register 0x101c0d58

31 … 9	8	7 … 4	3 … 0
reserved	SFIFO_CLR	reserved	SFIFO_WM

Bits	Name	Description	R/W	Default
31:9	reserved	-	R	0x0
8	SFIFO_CLR	Clear slave data FIFO, write only bit. Note: This bit is writable but can also be changed by hardware.	W	0x0
7:4	reserved	-	R	0x0
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	0x0

I2C_STAT – I2C Status Register 0x101c0d5c

31	30	29 28	27	26	25	24	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	0x0
30	SCL_STATE	SCL signal state sampled and filtered from bus (e.g. to detect bus blockings)	R	0x0
29:28	reserved	-	R	0x0
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	0x0
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	0x0
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	0x0
24	LAST_AC	Last acknowledge detected on bus: 0: SDA was high i.e no acknowledge. 1: SDA was low i.e acknowledge).	R	0x0
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	0x0
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	0x0
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	0x0
20	BUS_MASTER	1: master gains bus arbitration or bus is idle, 0: master lost bus arbitration, bus is busy by another master	R	0x0
19	SFIFO_ERR_UNDR	slave FIFO underrun error occurred, data is lost	R	0x0
18	SFIFO_ERR_OVFL	slave FIFO overflow error occurred, data is lost	R	0x0
17	SFIFO_FULL	slave FIFO is full (1 if full)	R	0x0
16	SFIFO_EMPTY	slave FIFO is empty (1 if empty)	R	0x0
15	reserved	-	R	0x0
14:10	SFIFO_LEVEL	slave FIFO level (0..16)	R	0x0
9	MFIFO_ERR_UNDR	master FIFO underrun error occurred, data is lost	R	0x0
8	MFIFO_ERR_OVFL	master FIFO overflow error occurred, data is lost	R	0x0
7	MFIFO_FULL	master FIFO is full (1 if full)	R	0x0
6	MFIFO_EMPTY	master FIFO is empty (1 if empty)	R	0x0
5	reserved	-	R	0x0
4:0	MFIFO_LEVEL	master FIFO level (0..16)	R	0x0

I2C_INT_MSK_SET_CLR – I2C Interrupt Mask Set or Clear Register 0x101c0d60

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 … 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	0x0
6	SREQ	Slave request interrupt mask.	R/W	0x0
5	SFIFO_REQ	Slave FIFO action request interrupt mask.	R/W	0x0
4	MFIFO_REQ	Master FIFO action request interrupt mask.	R/W	0x0
3	BUS_BUSY	External I2C-bus is busy interrupt mask.	R/W	0x0
2	FIFO_ERR	FIFO error interrupt mask.	R/W	0x0
1	CMD_ERR	Command error interrupt mask.	R/W	0x0
0	CMD_OK	Command OK interrupt mask.	R/W	0x0

I2C_RAW_INT_STAT – I2C RAW Interrupt Status Register 0x101c0d64

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

31 … 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	0x0
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	0x0
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	0x0
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	0x0
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	0x0

Bits	Name	Description	R/W	Default
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	0x0
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	0x0
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	0x0

I2C_MSK_INT_STAT – I2C Mask Interrupt Status Register 0x101c0d68

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.

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="25">reserved</td><td>SREQ</td><td>SFIFO_REQ</td><td>MFIFO_REQ</td><td>BUS_BUSY</td><td>FIFO_ERR</td><td>CMD_ERR</td><td>CMD_OK</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x0
6	SREQ	Masked slave request interrupt state.	R	0x0
5	SFIFO_REQ	Masked slave FIFO action request interrupt state.	R	0x0
4	MFIFO_REQ	Masked master FIFO action request interrupt state.	R	0x0
3	BUS_BUSY	Masked external I2C-bus is busy interrupt state.	R	0x0
2	FIFO_ERR	Masked FIFO error interrupt state.	R	0x0
1	CMD_ERR	Masked command error interrupt state.	R	0x0
0	CMD_OK	Masked command OK interrupt state.	R	0x0

I2C_DMA_CTRL – I2C DMA Control Register 0x101c0d6c

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 ... 4	3	2	1	0
reserved	SDMAB_EN	SDMAS_EN	MDMAB_EN	MDMAS_EN

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	SDMAB_EN	Enable DMA burst requests for I2C slave data. Flowcontrolling must be done my DMA-Controller. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
2	SDMAS_EN	Enable DMA single requests for I2C slave data. Flowcontrolling must be done my DMA-Controller. Note: This bit is writable but can also be changed by hardware.	R/W	0x0
1	MDMAB_EN	Enable DMA burst requests for I2C master data. Note: This bit is writable but can also be changed by hardware. Flowcontrolling may be done my DMA-Controller or by I2C-controller controlled by decrementing i2c_cmd.tsize.	R/W	0x0
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_cmd.tsize Note: This bit is writable but can also be changed by hardware.	R/W	0x0

I2C_PIO – Direct I2C IO Access Controlling 0x101c0d70

The i2c signals SCL and SDA can be directly controlled by this register if in I2C_MASTER_CTRL register pio_mode is enabled.
In pio-mode the I2C-controller statemachine is disabled, so no FIFO-action is done no IRQs occurs and no DMA-controlling is possible.

Warning: i2c-signals SCL and SDA are never driven active high by i2c specification. High-level should be done by pad pull-up and setting the appropriate output enable to 0 (scl_oe, sda_oe) instead of active high level driving to avoid external driving conflicts.

31 … 7	6	5	4	3	2	1	0
reserved	SDA_IN_RO	SDA_OE	SDA_OUT	reserved	SCL_IN_RO	SCL_OE	SCL_OUT

Bits	Name	Description	R/W	Default
31:7	reserved	-	R	0x0
6	SDA_IN_RO	SDA input state (read only)	R/W	0x1
5	SDA_OE	SDA output enable 0: don't drive SDA, switch pad to highZ. 1: drive SDA, switch pad to programmed sda_out-state	R/W	0x0
4	SDA_OUT	driving level (1: high, 0: low) of SDA if output is enabled (sda_oe is set)	R/W	0x0
3	reserved	-	R	0x0
2	SCL_IN_RO	SCL input state (read only)	R/W	0x1
1	SCL_OE	SCL output enable 0: don't drive SCL, switch pad to highZ. 1: drive SCL, switch pad to programmed scl_out-state	R/W	0x0
0	SCL_OUT	driving level (1: high, 0: low) of SCL if output is enabled (scl_oe is set)	R/W	0x0

9.8 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
0x101c1000	SYSTIME_S	Upper SYSTIME Register
0x101c1004	SYSTIME_NS	Lower SYSTIME Register
0x101c1008	SYS_TIME_NS_BOR	SYSTIME Border Register
0x101c100c	SYS_TIME_NS_ADD_UP	SYSTIME Count Register

SYSTIME_S – Upper SYSTIME Register 0x101c1000

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
This register should be dedicated to accesses via DPM.
ARM software should access systime via ARM_TIMER at systime_s.
xPIC software should access systime via xPIC_TIMER at systime_s.

31 … 0
SYSTIME_S

Bits	Name	Description	R/W	Default
31:0	SYSTIME_S	systime high value is incremented, if systime_ns reaches systime_border Sample systime_ns at read access to systime_s.	R/W	0x0

SYSTIME_NS – Lower SYSTIME Register 0x101c1004

To allow consistent values of systime_s and systime_ns, lower bits of systime is latched to systime_ns, when systime_s is read.
If no systime_s is read before (or at 2nd read access of systime_ns), the actual value of systime_ns is read.
This register should be dedicated to accesses via DPM.
ARM software should access systime via ARM_TIMER at systime_ns.
xPIC software should access systime via xPIC_TIMER at systime_ns.

31 … 0
SYSTIME_NS

Bits	Name	Description	R/W	Default
31:0	SYSTIME_NS	Systime low: Sample systime_ns at read access to systime_s. Without sample read systime_s, read the actual value of systime_ns.	R/W	0x0

SYS_TIME_NS_BOR – SYSTIME Border Register 0x101c1008

systime_ns counts from 0 to the border value (inlcuded), i.e. systime_ns counts modulo (systime_border + 1).

Attention: the border value Bit 3 to 1 must be b’1111 (hex f) for all netX systime - match functions.

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

Bits	Name	Description	R/W	Default
31:0	SYSTIME_BORDER	Systime border for lower systime	R/W	0x3b9ac9ff

SYS_TIME_NS_ADD_UP – SYSTIME Count Register 0x101c100c

Each clock cycle (systime_count_value >> 28) will be added to systime (rate multiplier for IEEE1588). Value 0x10000000 can be used for counting in 10ns (ethernet clock) steps.

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

Bits	Name	Description	R/W	Default
31:0	SYSTIME_COUNT_VALUE	Systime count value	R/W	0xa0000000

9.9 MMIO – Multiplex Matrix IOs

See chapter 3.3 (IO Configuration) for MMIO Configuration options.

9.10 USB – Serial USB-Interface

The following table shows a summary of all USB registers:

ARM Address	Register Name	Short Description
0x101c0e00	USB_DEV_CFG	USB Device Configuration Register
0x101c0e04	USB_DEV_STATUS	USB Device Status Register
0x101c0e08	USB_DEV_VENDOR_FEATURES	USB Vendor Feature Status Register
0x101c0e0c	USB_DEV_IRQ_MASK	USB Device Interrupt Mask Register
0x101c0e10	USB_DEV_IRQ_RAW	USB Device Raw Interrupt Status Register
0x101c0e14	USB_DEV_IRQ_MASKED	USB Device Masked Interrupt Status Register
0x101c0e40	USB_DEV_ENUM_RAM_DESCRIPTORS_BASE	USB Device Descriptor Start
0x101c0e44	USB_DEV_ENUM_RAM_DESCRIPTORS_END	USB Device Descriptor End
0x101c0e48	USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_BASE	USB String Descriptor Start
0x101c0e7c	USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_END	USB String Descriptor End
0x101c0e80	USB_DEV_FIFO_CTRL_CONF	USB Device FIFO Configuration Register
0x101c0e84	USB_DEV_FIFO_CTRL_OUT_HANDSHAKE	USB Device FIFO Out Handshake
0x101c0e88	USB_DEV_FIFO_CTRL_IN_HANDSHAKE	USB Device FIFO In Handshake
0x101c0e8c	USB_DEV_FIFO_CTRL_STATUS0	USB Device FIFO 0 Status Register
0x101c0e90	USB_DEV_FIFO_CTRL_STATUS1	USB Device FIFO 1 Status Register
0x101c0e94	USB_DEV_FIFO_CTRL_STATUS2	USB Device FIFO 2 Status Register
0x101c0e98	USB_DEV_FIFO_CTRL_STATUS3	USB Device FIFO 3 Status Register
0x101c0e9c	USB_DEV_FIFO_CTRL_STATUS4	USB Device FIFO 4 Status Register
0x101c0ea0	USB_DEV_FIFO_CTRL_STATUS5	USB Device FIFO 5 Status Register
0x101c0ea4	USB_DEV_FIFO_CTRL_STATUS6	USB Device FIFO 6 Status Register
0x101c0ec0	USB_DEV_FIFO0	USB Device FIFO: Control Endpoint OUT
0x101c0ec4	USB_DEV_FIFO1	USB Device FIFO: Control Endpoint IN
0x101c0ec8	USB_DEV_FIFO2	USB Device FIFO: Endpoint 1 - JTAG TX
0x101c0ecc	USB_DEV_FIFO3	USB Device FIFO: Endpoint 2 - JTAG RX
0x101c0ed0	USB_DEV_FIFO4	USB Device FIFO: Endpoint 3 - UART TX
0x101c0ed4	USB_DEV_FIFO5	USB Device FIFO: Endpoint 4 - UART RX
0x101c0ed8	USB_DEV_FIFO6	USB Device FIFO: Endpoint 5 - Interrupt IN

USB_DEV_CFG – USB Device Configuration Register 0x101c0e00

This register configures the USB device functions. It allows to entirely disable the USB core and the USB to JTAG bridge.

31	30	29	28	27	26	25	24	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_DEV_RESET	USB_TO_JTAG_ENABLE	USB_CORE_ENABLE

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	0x0
2	USB_DEV_RESET	Writing a '1' to this bit will reset the USB core and JTAG module. The user must deassert the bit manually.	R/W	0x0
1	USB_TO_JTAG_ENABLE	When set, the USB to JTAG module is active.	R/W	0x1
0	USB_CORE_ENABLE	Writing a '0' to this bit will disable the USB core. By default the core is enabled by the bootloader.	R/W	0x0

USB_DEV_STATUS – USB Device Status Register 0x101c0e04

This register represents various status information of the USB core and its FIFOs.

31	30	29	28	27	26	25	24	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_BUS_RESET	USB_CONFIGURED	USB_ADDRESSED	USB_BUSY

Bits	Name	Description	R/W	Default
31:4	reserved	-	R	0x0
3	USB_BUS_RESET	This bit is set when the bus is held in reset state (i.e. the FIFOs are held in reset).	R	0x0
2	USB_CONFIGURED	This bit is set as soon as the USB host has configured the core.	R	0x0
1	USB_ADDRESSED	This bit is set as soon as the USB host set a valid address.	R	0x0
0	USB_BUSY	This bit is set while an USB transfer is active.	R	0x0

USB_DEV_VENDOR_FEATURES – USB Vendor Feature Status Register 0x101c0e08

This register represents the last valid vendor features that the USB Host has set.

31 … 16	15 … 0
reserved	VENDOR_FEATURES

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	VENDOR_FEATURES	The last valid vendor features set by the host.	R	0x0

USB_DEV_IRQ_MASK – USB Device Interrupt Mask Register 0x101c0e0c

The value of this register is used for AND-masking the raw interrupt register. When a bit is set, the corresponding interrupt is routed to the interrupt controller. Different to the other netX modules, the USB-IRQ mask is written directly and not by set- and reset-masks. For a detailed IRQ description see USB_DEV_IRQ_RAW.

31 … 14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	UART_RX_PACKET_RECEIVED	UART_TX_PACKET_SENT	JTAG_RX_PACKET_RECEIVED	JTAG_TX_PACKET_SENT	JTAG_SRST_REQUESTED	RESET_DETECTED	DEVICE_HALTED	DROPPED_FRAME	CRC16_ERROR	FIFO_OVERFLOW	UART_TX_FIFO_EMPTY	UART_TX_FIFO_FULL	UART_RX_FIFO_EMPTY	UART_RX_FIFO_FULL

Bits	Name	Description	R/W	Default
31:14	reserved	-	R	0x0
13	UART_RX_PACKET_RECEIVED	UART rx packet received interrupt mask.	R/W	0x0
12	UART_TX_PACKET_SENT	UART tx packet sent interrupt mask.	R/W	0x0
11	JTAG_RX_PACKET_RECEIVED	JTAG rx packet received interrupt mask.	R/W	0x0
10	JTAG_TX_PACKET_SENT	JTAG tx packet sent interrupt mask.	R/W	0x0
9	JTAG_SRST_REQUESTED	JTAG system reset request detected interrupt mask	R/W	0x0
8	RESET_DETECTED	Reset detected interrupt mask.	R/W	0x0
7	DEVICE_HALTED	Device halted interrupt mask.	R/W	0x0
6	DROPPED_FRAME	Dropped frame occurred interrupt mask.	R/W	0x0
5	CRC16_ERROR	CRC16 error in USB packet occurred interrupt mask.	R/W	0x0
4	FIFO_OVERFLOW	FIFO overflow interrupt mask.	R/W	0x0
3	UART_TX_FIFO_EMPTY	UART transmit FIFO empty interrupt mask.	R/W	0x0
2	UART_TX_FIFO_FULL	UART transmit FIFO full interrupt mask.	R/W	0x0
1	UART_RX_FIFO_EMPTY	UART receive FIFO empty interrupt mask.	R/W	0x0
0	UART_RX_FIFO_FULL	UART receive FIFO full interrupt mask.	R/W	0x0

USB_DEV_IRQ_RAW – USB Device Raw Interrupt Status Register 0x101c0e10

This register holds the raw interrupt status before masking has been applied. Writing ’1’ will clear the corresponding interrupt.

31 … 14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved	UART_RX_PACKET_RECEIVED	UART_TX_PACKET_SENT	JTAG_RX_PACKET_RECEIVED	JTAG_TX_PACKET_SENT	JTAG_SRST_REQUESTED	RESET_DETECTED	DEVICE_HALTED	DROPPED_FRAME	CRC16_ERROR	FIFO_OVERFLOW	UART_TX_FIFO_EMPTY	UART_TX_FIFO_FULL	UART_RX_FIFO_EMPTY	UART_RX_FIFO_FULL

Bits	Name	Description	R/W	Default
31:14	reserved	-	R	0x0
13	UART_RX_PACKET_RE CEIVED	Unmasked UART rx packet received interrupt state: A packet in the UART receive FIFO has arrived. This IRQ is useful, when the FIFO is in packet control mode.	R/W	0x0
12	UART_TX_PACKET_SE NT	Unmasked UART tx packet sent interrupt state: A packet in the UART transmit FIFO has been sent to the USB host. This IRQ is useful, when the FIFO is in packet control mode.	R/W	0x0
11	JTAG_RX_PACKET_RE CEIVED	Unmasked JTAG rx packet received interrupt state: A packet in the JTAG receive FIFO has arrived. This IRQ is useful, when the FIFO is in packet control mode. The FIFO is normally not under user control.	R/W	0x0
10	JTAG_TX_PACKET_SEN T	Unmasked JTAG tx packet sent interrupt state: A packet in the JTAG transmit FIFO has been sent to the USB host. The FIFO is normally not under user control. This IRQ is useful, when the FIFO is in packet control mode.	R/W	0x0
9	JTAG_SRST_REQUEST ED	Unmakesd JTAG system reset request detected interrupt state: This IRQ is generated, when the USB Host software requested a system reset. \	R/W	0x0
8	RESET_DETECTED	Unmasked reset detected interrupt state: This bit is set, when the USB core detected a reset condition on the bus. This means that all FIFOs have been reset and the user should check FIFO states before reading again.	R/W	0x0
7	DEVICE_HALTED	Unmasked device halted interrupt state: This bit is set, when the USB core detected a halted condition. This may happen on bus errors or when the host explicitly requests it.	R/W	0x0
6	DROPPED_FRAME	Unmasked dropped frame interrupt state: A dropped or misaligned frame has been detected. Operation continues normally. This interrupt is meant as an information to the user’s software, which may need to handle these errors.	R/W	0x0
5	CRC16_ERROR	Unmasked CRC16 error in USB packet interrupt state: A CRC16 mismatch in a USB packet has been detected. The corresponding packet has been dropped.	R/W	0x0
4	FIFO_OVERFLOW	Unmasked FIFO overflow interrupt state: An overflow of one of the endpoint FIFOs has occurred. The user needs to check the usb_dev_fifo_ctrl_status_* registers which one is affected.	R/W	0x0
3	UART_TX_FIFO_EMPTY	Unmasked UART transmit FIFO empty interrupt state: When set the transmit FIFO of the UART channel is empty.	R/W	0x0
2	UART_TX_FIFO_FULL	Unmasked UART transmit FIFO full interrupt state: When set the transmit FIFO of the UART channel is full.	R/W	0x0
1	UART_RX_FIFO_EMPTY	Unmasked UART receive FIFO empty interrupt state: When set the receive FIFO of the UART channel is empty.	R/W	0x0
0	UART_RX_FIFO_FULL	Unmasked UART receive FIFO full interrupt state:	R/W	0x0

Bits	Name	Description	R/W	Default
		When set the receive FIFO of the UART channel is full.		

USB_DEV_IRQ_MASKED – USB Device Masked Interrupt Status Register 0x101c0e14

If one of these bits is set, the USB device interrupt will be asserted to the interrupt controller. For a detailed IRQ description view USB_DEV_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																		UART_RX_PACKET_RECEIVED	UART_TX_PACKET_SENT	JTAG_RX_PACKET_RECEIVED	JTAG_TX_PACKET_SENT	JTAG_SRST_REQUESTED	RESET_DETECTED	DEVICE_HALTED	DROPPED_FRAME	CRC16_ERROR	FIFO_OVERFLOW	UART_TX_FIFO_EMPTY	UART_TX_FIFO_FULL	UART_RX_FIFO_EMPTY	UART_RX_FIFO_FULL

Bits	Name	Description	R/W	Default
31:14	reserved	-	R	0x0
13	UART_RX_PACKET_RECEIVED	Masked UART rx packet received interrupt state.	R	0x0
12	UART_TX_PACKET_SENT	Masked UART tx packet sent interrupt state.	R	0x0
11	JTAG_RX_PACKET_RECEIVED	Masked JTAG rx packet received interrupt state.	R	0x0
10	JTAG_TX_PACKET_SENT	Masked JTAG tx packet sent interrupt state.	R	0x0
9	JTAG_SRST_REQUESTED	Masked JTAG system reset request detected interrupt state.	R	0x0
8	RESET_DETECTED	Masked reset detected interrupt state.	R	0x0
7	DEVICE_HALTED	Masked device halted interrupt state.	R	0x0
6	DROPPED_FRAME	Masked dropped frame interrupt state	R	0x0
5	CRC16_ERROR	Masked CRC16 error in USB packet interrupt state.	R	0x0
4	FIFO_OVERFLOW	Masked FIFO overflow interrupt state.	R	0x0
3	UART_TX_FIFO_EMPTY	Masked UART transmit FIFO empty interrupt state.	R	0x0
2	UART_TX_FIFO_FULL	Masked UART transmit FIFO full interrupt state.	R	0x0
1	UART_RX_FIFO_EMPTY	Masked UART receive FIFO empty interrupt state.	R	0x0
0	UART_RX_FIFO_FULL	Masked UART receive FIFO full interrupt state.	R	0x0

USB_DEV_ENUM_RAM_DESCRIPTORS_BASE – USB Device Descriptor Start 0x101c0e40
USB_DEV_ENUM_RAM_DESCRIPTORS_END – USB Device Descriptor End 0x101c0e44

Device descriptor configuration start/end addresses in the enumeration RAM.

The layout of the RAM area is as following:

Enumeration RAM Addr	Byte(s)	Function	
0x101c0e40	0	Vendor ID (low)	Device descriptor
0x101c0e41	1	Vendor ID (high)	
0x101c0e42	2	Product ID (low)	
0x101c0e43	3	Product ID (high)	
0x101c0e44	4	Device release number (low)	
0x101c0e45	5	Device release number (high)	
0x101c0e46	6	Configuration characteristics	Configuration descriptor
0x101c0e47	7	Maximum power consumption	

USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_BASE – USB String Descriptor Start0x101c0e48
USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_END – USB String Descriptor End 0x101c0e7c

String descriptor start/end addresses in the enumeration RAM.

The layout of the RAM area is as following:

Enumeration RAM Addr	Byte(s)	Function	
0x101c0e48	0	Vendor string descriptor length	Vendor string descriptor
0x101c0e49	1	Vendor string descriptor type	
0x101c0e4a ~ 0x101c0e59	2 ~ 17	Vendor string	
0x101c0e5a	18	Product string descriptor length	Product string descriptor
0x101c0e5b	19	Product string descriptor type	
0x101c0e5c ~ 0x101c0e6b	20 ~ 35	Product string	
0x101c0e6c	36	S/N string descriptor length	S/N string descriptor
0x101c0e6d	37	S/N string descriptor type	
0x101c0e6e ~ 0x101c0e7d	38 ~ 53	S/N string	

USB_DEV_FIFO_CTRL_CONF – USB Device FIFO Configuration Register 0x101c0e80

This register configures the FIFOs of the USB core. The user can select the mode of the FIFO (streaming or packet oriented transmission) and the auto acknowledge feature. It is also possible to reset each FIFO individually.

Note: The user should not touch the configuration of endpoint 0 (IN and OUT). Otherwise the USB core won't work properly. Normally the user also shouldn't touch endpoint 1 und 2 (JTAG). The only reason changing the config would be when using special drivers, which use this channel not as JTAG.

The default configuration of the UART channel is stream mode in transmit direction, so data sent to the USB host will be sent as soon as data arrives at the FIFO and the USB host requests an IN transaction. No other user interaction as putting the data to be transmitted in the FIFO is necessary. The receive direction is configured in packet control mode, to make use of all USB retransmission features. This means the user sees the data sent from the USB host only when the transmission was successful. This endpoint is configured in auto acknowledge mode, so the user only needs to read out the data completely and doesn't need to do any handshaking with the FIFO.

OUT endpoints should always be configured in packet control mode (and usually with auto out ack, because the user shouldn't need to refetch already fetched data from the FIFO), so the USB core is able to discard data from invalid transactions. If the user decides to not use the auto out ack feature, the user needs to write an out acknowledgement (ctrl_out_handshake) for the corresponding endpoint when he grabbed all data from the FIFO.

IN endpoints may be configured in either mode, but auto out acknowledgement should not be activated (otherwise the USB core can't do proper retransmits as specified in the USB standard). When in packet mode, the user needs to confirm the data put into the FIFO before it will be sent to the USB host. This is done by writing a "1" to the corresponding bit in the ctrl_in_handshake register. The user may also discard the data that has been put into the FIFO. In stream mode the data put into the FIFO will be sent as soon as the USB host requests a IN transaction, meaning no handshaking done by the user is needed (but data may be transferred in different USB packets to the USB host).

Note: When an endpoint is in packet control mode and the user acked the data, the user may already put data for the next packet into the FIFO, but must wait until the previous packet has been sent before acking the new packet.

31–23	22–16	15	14–8	7	6–0
reserved	AUTO_OUT_ACK	reserved	MODE	reserved	RESET

Bits	Name	Description	R/W	Default
31:23	reserved	-	R	0x0
22:16	AUTO_OUT_ACK	Selects whether the FIFO should automatically generate an out acknowledge when the FIFO has been read until it is empty.	R/W	0x29
15	reserved	-	R	0x0
14:8	MODE	Selects the mode of the FIFO channel. A "1" means streaming mode , a "0" enables the packet control mode. Control IN and OUT must be kept in the default configuration for the core to function properly. JTAG RX and TX may be switched to packet mode, when the USB host has switched to JTAG bypass mode (see text above). UART RX and TX are fully user definable (see text above).	R/W	0x56
7	reserved	-	R	0x0
6:0	RESET	Writing a '1' to a bit resets the corresponding endpoint FIFO. Resets must be manually cleared by the user. Bit FIFO 0 Endpoint 0 - Control OUT 1 Endpoint 0 - Control IN 2 Endpoint 1 - JTAG TX 3 Endpoint 2 - JTAG RX 4 Endpoint 3 - UART TX 5 Endpoint 4 - UART RX	R/W	0x0

Bits	Name	Description	R/W	Default
		6 Endpoint 5 - Interrupt IN		

USB_DEV_FIFO_CTRL_OUT_HANDSHAKE – USB Device FIFO OUT Handshake 0x101c0e84

This register is used to control the handshake signals for the output of the FIFOs. The bits must be manually cleared by the user.

31:15	14:8	7	6:0
reserved	RETRANSMIT	reserved	ACK

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14:8	RETRANSMIT	Writing a ’1’ to a bit requests a FIFO’s retransmission of the buffered content	R/W	0x0
7	reserved	-	R	0x0
6:0	ACK	Writing a ’1’ to a bit acknowledges the current buffered content and frees the memory	R/W	0x0

USB_DEV_FIFO_CTRL_IN_HANDSHAKE – USB Device FIFO IN Handshake 0x101c0e88

This register is used to control the handshake signals for the input of the FIFOs. The bits must be manually cleared by the user.

31:15	14:8	7	6:0
reserved	DISCARD	reserved	ACK

Bits	Name	Description	R/W	Default
31:15	reserved	-	R	0x0
14:8	DISCARD	Writing a ’1’ to a bit discards the current contents of the corresponding FIFO	R/W	0x0
7	reserved	-	R	0x0
6:0	ACK	Writing a ’1’ to a bit acknowledges the current packet data (packet mode only). The packet must be fully transmitted before issuing another acknowledge.	R/W	0x0

USB_DEV_FIFO_CTRL_STATUS0 – USB Device FIFO 0 Status Register 0x101c0e8c
USB_DEV_FIFO_CTRL_STATUS1 – USB Device FIFO 1 Status Register 0x101c0e90
USB_DEV_FIFO_CTRL_STATUS2 – USB Device FIFO 2 Status Register 0x101c0e94
USB_DEV_FIFO_CTRL_STATUS3 – USB Device FIFO 3 Status Register 0x101c0e98
USB_DEV_FIFO_CTRL_STATUS4 – USB Device FIFO 4 Status Register 0x101c0e9c
USB_DEV_FIFO_CTRL_STATUS5 – USB Device FIFO 5 Status Register 0x101c0ea0
USB_DEV_FIFO_CTRL_STATUS6 – USB Device FIFO 6 Status Register 0x101c0ea4

This register holds the fill levels and other status information of FIFOs.

31–19	18	17	16	15	14–8	7	6–0
reserved	PACKET_IN_OUT_BUFFER	UNDERRUN	OVERFLOW	reserved	IN_FILL_LEVEL	reserved	OUT_FILL_LEVEL

Bits	Name	Description	R/W	Default
31:19	reserved	-	R	0x0
18	PACKET_IN_OUT_BUFFER	Set when a packet or data is in the output buffer available for reading	R	0x0
17	UNDERRUN	Set when more read request have been made than bytes available in the buffer	R	0x0
16	OVERFLOW	Set when more write request have been made than there was room in the buffer	R	0x0
15	reserved	-	R	0x0
14:8	IN_FILL_LEVEL	The fill level of the input side (write side)	R	0x0
7	reserved	-	R	0x0
6:0	OUT_FILL_LEVEL	The fill level of the output side (read side). In stream mode this is the same as the input fill level	R	0x0

USB_DEV_FIFO0 – USB device FIFO: Control Endpoint OUT 0x101c0ec0

This FIFO holds the data of the control endpoint. Direction is OUT, meaning data sent from USB host to device arrives here. The FIFO is handled by the USB core itself and should not be read or written from the ARM.

Note: Reading and writing to this register while the USB module sees an reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

31–8	7–0
reserved	FIFO_DATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO1 – USB device FIFO: Control Endpoint IN 0x101c0ec4

This FIFO holds the data of the control endpoint. Direction is IN, meaning data that should be sent from USB device to host must be placed here. The FIFO is handled by the USB core itself and should not be read or written from the ARM.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">reserved</td><td colspan="8">FIFO_DATA</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO2 – USB device FIFO: Endpoint 1 - JTAG TX 0x101c0ec8

This FIFO holds the data of the bulk endpoint used for JTAG communication. Direction is IN, meaning data placed here is sent to the USB host. The FIFO is handled by the USB JTAG core itself and should not be read or written from the ARM.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">reserved</td><td colspan="8">FIFO_DATA</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO3 – USB device FIFO: Endpoint 2 - JTAG RX 0x101c0ecc

This FIFO holds the data of the bulk endpoint used for JTAG communication. Direction is OUT, meaning data from the USB host arrives here. The FIFO is handled by the USB JTAG core itself and should not be read or written from the ARM.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="24">reserved</td><td colspan="8">FIFO_DATA</td></tr>
</table>

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO4 – USB device FIFO: Endpoint 3 - UART TX 0x101c0ed0

This FIFO holds the data of the bulk endpoint used for UART communication. Direction is IN, meaning data placed here is sent to the USB host. This FIFO may be used by the user application.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

31 … 8	7 … 0
reserved	FIFO_DATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO5 – USB device FIFO: Endpoint 4 - UART RX 0x101c0ed4

This FIFO holds the data of the bulk endpoint used for UART communication. Direction is OUT, meaning data from the USB host arrives here. This FIFO may be used by the user application.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

31 … 8	7 … 0
reserved	FIFO_DATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

USB_DEV_FIFO6 – USB device FIFO: Endpoint 5 - Interrupt IN 0x101c0ed8

This FIFO holds the data of the interrupt endpoint. Direction is IN, meaning data placed here is sent to the USB host.

Note: Reading and writing to this register while the USB module sees a reset condition on the bus results in unexpected data, because the FIFOs are held in reset state.

31 … 8	7 … 0
reserved	FIFO_DATA

Bits	Name	Description	R/W	Default
31:8	reserved	-	R	0x0
7:0	FIFO_DATA		R/W	0x0

9.11 VIC – Vectored Interrupt Controller

The following table shows a summary of all VIC registers:

ARM Address	Register Name	Short Description
0x101ff000	VIC_IRQ_STAT	IRQ Status Register
0x101ff004	VIC_FIQ_STAT	FIQ Status Register
0x101ff008	VIC_RAW_INT_STAT	Raw Interrupt Status Register
0x101ff00c	VIC_INT_SEL	Interrupt Select Register
0x101ff010	VIC_INT_EN	Interrupt Enable Register
0x101ff014	VIC_INT_EN_CLR	Interrupt Enable Clear Register
0x101ff018	VIC_SWI	Software Interrupt Register
0x101ff01c	VIC_SWI_CLR	Software Interrupt Clear Register
0x101ff020	VIC_PROT_EN	Protection Enable Register
0x101ff030	VIC_VECT_ADDR	Vector Address Register
0x101ff034	VIC_DFLT_VECT_ADDR	Default Vector Address Register
0x101ff100	VIC_VECT_ADDR0	Vector Address Register 0
0x101ff104	VIC_VECT_ADDR1	Vector Address Register 1
0x101ff108	VIC_VECT_ADDR2	Vector Address Register 2
0x101ff10c	VIC_VECT_ADDR3	Vector Address Register 3
0x101ff110	VIC_VECT_ADDR4	Vector Address Register 4
0x101ff114	VIC_VECT_ADDR5	Vector Address Register 5
0x101ff118	VIC_VECT_ADDR6	Vector Address Register 6
0x101ff11c	VIC_VECT_ADDR7	Vector Address Register 7
0x101ff120	VIC_VECT_ADDR8	Vector Address Register 8
0x101ff124	VIC_VECT_ADDR9	Vector Address Register 9
0x101ff128	VIC_VECT_ADDR10	Vector Address Register 10
0x101ff12c	VIC_VECT_ADDR11	Vector Address Register 11
0x101ff130	VIC_VECT_ADDR12	Vector Address Register 12
0x101ff134	VIC_VECT_ADDR13	Vector Address Register 13
0x101ff138	VIC_VECT_ADDR14	Vector Address Register 14
0x101ff13c	VIC_VECT_ADDR15	Vector Address Register 15
0x101ff200	VIC_VECT_CTRL0	Vector Control Register 0
0x101ff204	VIC_VECT_CTRL1	Vector Control Register 1
0x101ff208	VIC_VECT_CTRL2	Vector Control Register 2
0x101ff20c	VIC_VECT_CTRL3	Vector Control Register 3
0x101ff210	VIC_VECT_CTRL4	Vector Control Register 4
0x101ff214	VIC_VECT_CTRL5	Vector Control Register 5
0x101ff218	VIC_VECT_CTRL6	Vector Control Register 6
0x101ff21c	VIC_VECT_CTRL7	Vector Control Register 7
0x101ff220	VIC_VECT_CTRL8	Vector Control Register 8
0x101ff224	VIC_VECT_CTRL9	Vector Control Register 9
0x101ff228	VIC_VECT_CTRL10	Vector Control Register 10
0x101ff22c	VIC_VECT_CTRL11	Vector Control Register 11
0x101ff230	VIC_VECT_CTRL12	Vector Control Register 12
0x101ff234	VIC_VECT_CTRL13	Vector Control Register 13
0x101ff238	VIC_VECT_CTRL14	Vector Control Register 14
0x101ff23c	VIC_VECT_CTRL15	Vector Control Register 15

VIC_IRQ_STAT – VIC IRQ Status Register 0x101ff000

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31	ADC	ADC0 or ADC1	R	0x0
30	ENCODER	Any encoder IRQ	R	0x0
29	PWM	Any PWM IRQ	R	0x0
28	TRIGGER_LT	reserved for netX compatibility	R	0x0
27	DMAC	DMA controller	R	0x0
26	SYSSTATE	License error or extmem_timeout	R	0x0
25	INT_PHY	Interrupt from internal Phys	R	0x0
24	MSYNC3	reserved for SW IRQ from xPIC to ARM	R	0x0
23	MSYNC2	reserved for netX compatibility	R	0x0
22	MSYNC1	reserved for netX compatibility	R	0x0
21	MSYNC0	Motion synchronization channel 0 (= |xpec0_irq[15:12])	R	0x0
20	COM3	xPIC Debug	R	0x0
19	COM2	reserved for netX compatibility	R	0x0
18	COM1	reserved for netX compatibility	R	0x0
17	COM0	Communication channel 0 (= |xpec0_irq[11:0])	R	0x0
16	GPIO	other external Interrupts from GPIO 0-6 / IO-Link	R	0x0
15	HIF	HIF/DPM interrupt	R	0x0
14	LCD	reserved	R	0x0
13	I2C	I2C	R	0x0
12	SPI	combined SPI0, SPI1 interrupt	R	0x0
11	USB	USB interrupt	R	0x0
10	UART2	reserved for netX compatibility	R	0x0
9	UART1	UART 1	R	0x0
8	UART0	UART 0 -> Diagnostic channel, Windows CE required	R	0x0
7	WATCHDOG	System Watchdog from WGD_SYS or xPIC-ARM-Watchdog_IRQ	R	0x0
6	GPIO7	external interrupt 7, Windows CE required (NMI)	R	0x0
5	SYSTIME_S	systime 1day IRQ from ARM_TIMER module, Windows CE required	R	0x0
4	SYSTIME_NS	systime ns compare irq from ARM_TIMER module	R	0x0
3	GPIO_TIMER	Timer0 or Timer1 from GPIO Module	R	0x0
2	TIMER1	Timer1 / Counter1 from ARM_TIMER Module	R	0x0
1	TIMER0	Timer0 / Counter0 from ARM_TIMER Module, Windows CE required	R	0x0
0	SW	Reserved for Software Interrupt, ARM standard configuration	R	0x0

VIC_FIQ_STAT – VIC FIQ Status Register 0x101ff004

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_FIQ_STAT	Fast Interrupt Status	R	0x0

VIC_RAW_INT_STAT – VIC Raw Interrupt Status Register 0x101ff008

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	TIMER1	TIMER0	SW

Bits	Name	Description	R/W	Default
31:0	VIC_RAW_INT_STAT	Raw Interrupt Status	R	0x0

VIC_INT_SEL – VIC Interrupt Select Register 0x101ff00c

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	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	0x0

VIC_INT_EN – VIC Interrupt Enable Register 0x101ff010

The VIC_INT_EN register enables the interrupt request lines, by unmasking the interrupt sources for the IRQ interrupt. Clearing these bits is 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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	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	0x0

VIC_INT_EN_CLR – VIC Interrupt Enable Clear Register 0x101ff014

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	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	0x0

VIC_SWI – VIC Software Interrupt Register 0x101ff018

The VIC_SWI register is used to generate software interrupts. Setting a bit generates a software interrupt for the specific source before interrupt masking.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	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	0x0

VIC_SWI_CLR – VIC Software Interrupt Clear Register 0x101ff01c

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
ADC	ENCODER	PWM	TRIGGER_LT	DMAC	SYSSTATE	INT_PHY	MSYNC3	MSYNC2	MSYNC1	MSYNC0	COM3	COM2	COM1	COM0	GPIO	HIF	LCD	I2C	SPI	USB	UART2	UART1	UART0	WATCHDOG	GPIO7	SYSTIME_S	SYSTIME_NS	GPIO_TIMER	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	0x0

VIC_PROT_EN – VIC Protection Enable Register 0x101ff020

netX10 does not support protected mode, so this register is unused.

31 … 1	0
reserved	PROTECTION

Bits	Name	Description	R/W	Default
31:1	reserved	-	R	0x0
0	PROTECTION	0: VIC registers are accessable 1: VIC registers are not accessable	R/W	0x0

VIC_VECT_ADDR – VIC Vector Address Register 0x101ff030

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
VECT_ADDR																															

Bits	Name	Description	R/W	Default
31:0	VECT_ADDR	Address of the currently active ISR handler. Any writes to this register clear the current interrupt.	R/W	0x0

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 0x101ff034

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
DEF_VECT_ADDR																															

Bits	Name	Description	R/W	Default
31:0	DEF_VECT_ADDR	address of the default ISR handler	R/W	0x0

VIC_VECT_ADDR0 – VIC Vector Address Register 0 0x101ff100
VIC_VECT_ADDR1 – VIC Vector Address Register 1 0x101ff104
VIC_VECT_ADDR2 – VIC Vector Address Register 2 0x101ff108
VIC_VECT_ADDR3 – VIC Vector Address Register 3 0x101ff10c
VIC_VECT_ADDR4 – VIC Vector Address Register 4 0x101ff110
VIC_VECT_ADDR5 – VIC Vector Address Register 5 0x101ff114
VIC_VECT_ADDR6 – VIC Vector Address Register 6 0x101ff118
VIC_VECT_ADDR7 – VIC Vector Address Register 7 0x101ff11c
VIC_VECT_ADDR8 – VIC Vector Address Register 8 0x101ff120
VIC_VECT_ADDR9 – VIC Vector Address Register 9 0x101ff124
VIC_VECT_ADDR10 – VIC Vector Address Register 10 0x101ff128
VIC_VECT_ADDR11 – VIC Vector Address Register 11 0x101ff12c
VIC_VECT_ADDR12 – VIC Vector Address Register 12 0x101ff130
VIC_VECT_ADDR13 – VIC Vector Address Register 13 0x101ff134
VIC_VECT_ADDR14 – VIC Vector Address Register 14 0x101ff138
VIC_VECT_ADDR15 – VIC Vector Address Register 15 0x101ff13c

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.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VECT_ADDR0-15																															

Bits	Name	Description	R/W	Default
31:0	VECT_ADDR0-15	Contains ISR vector addresses	R/W	0x0

VIC_VECT_CTRL0 – VIC Vector Control 0 Register 0x101ff200
VIC_VECT_CTRL1 – VIC Vector Control 1 Register 0x101ff204
VIC_VECT_CTRL2 – VIC Vector Control 2 Register 0x101ff208
VIC_VECT_CTRL3 – VIC Vector Control 3 Register 0x101ff20c
VIC_VECT_CTRL4 – VIC Vector Control 4 Register 0x101ff210
VIC_VECT_CTRL5 – VIC Vector Control 5 Register 0x101ff214
VIC_VECT_CTRL6 – VIC Vector Control 6 Register 0x101ff218
VIC_VECT_CTRL7 – VIC Vector Control 7 Register 0x101ff21c
VIC_VECT_CTRL8 – VIC Vector Control 8 Register 0x101ff220
VIC_VECT_CTRL9 – VIC Vector Control 9 Register 0x101ff224
VIC_VECT_CTRL10 – VIC Vector Control 10 Register 0x101ff228
VIC_VECT_CTRL11 – VIC Vector Control 11 Register 0x101ff22c
VIC_VECT_CTRL12 – VIC Vector Control 12 Register 0x101ff230
VIC_VECT_CTRL13 – VIC Vector Control 13 Register 0x101ff234
VIC_VECT_CTRL14 – VIC Vector Control 14 Register 0x101ff238
VIC_VECT_CTRL15 – VIC Vector Control 15 Register 0x101ff23c

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	0x0
5	ENABLE	Enables vector interrupt. This bit is cleared on reset.	R/W	0x0
4:0	INT_SOURCE	Select interrupt source. You can select any of the 32 interrupt sources.	R/W	0x0

10 Communication Functions

The following table shows a summary of these registers:

ARM Address	Register Name	Short Description
0x101c0010	PHY_CTRL	PHY Control Register
0x101c0c00	MIIMU_RXTX	MIIMU Receive/Transmit Register
0x101c0c04	MIIMU_MODE_EN	MIIMU Software Mode Enable
0x101c0c08	MIIMU_MODE_MDC	MIIMU Software Mode MDC Register
0x101c0c0c	MIIMU_MODE_MDO	MIIMU Software Mode MDO Register
0x101c0c10	MIIMU_MODE_MDOE	MIIMU Software Mode MDOE Register
0x101c0c14	MIIMU_MODE_MDI	MIIMU Software Mode MDI Register
0x101a4000	PTR_FIFO_BASE	Pointer FIFO Table
0x101a4040	PTR_FIFO_BOR_BASE	Pointer FIFO Upper Borders table
0x101a4080	PTR_FIFO_RESET	Pointer FIFO Reset Vector
0x101a4084	PTR_FIFO_FULL	Pointer FIFO Full Vector
0x101a4088	PTR_FIFO_EMPTY	Pointer FIFO Empty Vector
0x101a408c	PTR_FIFO_OVF	Pointer FIFO Overflow Vector
0x101a4090	PTR_FIFO_UDR	Pointer FIFO Underrun Vector
0x101a40c0	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Fill-Level table
0x10124000	PTR_FIFO_BASE	Pointer FIFO Motion table
0x10124040	PTR_FIFO_BOR_BASE	Pointer FIFO Motion Upper Borders table
0x10124080	PTR_FIFO_RESET	Pointer FIFO Motion Reset Vector
0x10124084	PTR_FIFO_FULL	Pointer FIFO Motion Full Vector
0x10124088	PTR_FIFO_EMPTY	Pointer FIFO Motion Empty Vector
0x1012408c	PTR_FIFO_OVF	Pointer FIFO Motion Overflow Vector
0x10124090	PTR_FIFO_UDR	Pointer FIFO Motion Underrun Vector
0x101240c0	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Motion Fill-Level table
0x101a5600	BUF_MAN_BMU	BUF_MAN BMU
0x10125600	BUF_MAN_MOTION_BMU	BUF_MAN_MOTION BMU
0x101a4400	IRQ_XP0	IRQs between XPEC0 and ARM

10.1 PHY – Controller for internal PHY

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 handling 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 0x101c0010

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

Note:
Bit field ‘PHY0_NP_MSG_CODE’ was omitted. Related PHY inputs are only for test purpose.
PHY Next-Page features must be programmed in PHY registers if 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
PHY_RESET	PHY_SIM_BYP	reserved	PHY0_ENABLE	reserved	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	0x0
29:13	reserved	-	R	0x0
12	PHY0_ENABLE	PHY0 enable	R/W	0x0
11:9	reserved	-	R	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. 100BASETX 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 0x101c0c00

31–16	15–11	10–6	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_snrdy-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 0x101c0c04

31–1	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 MDI value can be read from miimu_mode_mdi.	R/W	0x0

MIIMU_MODE_MDC – MIIMU Software Mode MDC Register 0x101c0c08

31:1	0
reserved	MIIMU_SW_MDC

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

MIIMU_MODE_MDO – MIIMU Software Mode MDO Register 0x101c0c0c

31:1	0
reserved	MIIMU_SW_MDO

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

MIIMU_MODE_MDOE – MIIMU Software Mode MDOE Register 0x101c0c10

31:1	0
reserved	MIIMU_SW_MDOE

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

MIIMU_MODE_MDI – MIIMU Software Mode MDI Register 0x101c0c14

31 ... 1	0
reserved	MIIMU_SW_MDI

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

10.2 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 16 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 – POINTER_FIFO Pointer FIFO Table 0x101a4000
PTR_FIFO_BASE – POINTER_FIFO_MOTION Pointer FIFO Table 0x10124000

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

31 ... 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 – POINTER_FIFO Pointer FIFO Upper Borders Table 0x101a4040
PTR_FIFO_BOR_BASE – POINTER_FIFO_MOTION Pointer FIFO Upper Borders Table0x10124040

The sizes of all FIFOs are programmable. The total size of all FIFOs must not exceed 1024 dwords. Each of the following 16 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 ... 10	9 ... 0
reserved	BORDER

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9:0	BORDER	last address of RAM used by appropriate FIFO, = (first address-1) of next FIFO. FIFO 0 default depth: 512 FIFO 1..14 default depth: 32 FIFO 15 default depth: 64	R/W	0x0

PTR_FIFO_RESET – POINTER_FIFO Pointer FIFO Reset Vector 0x101a4080
PTR_FIFO_RESET – POINTER_FIFO_MOTION Pointer FIFO Reset Vector 0x10124080

This register allows resetting each of the 16 FIFOs, i.e. set read and write pointer to lower border of FIFO, reset full, overflow, under run and set empty flag. The reset flag of adjacent FIFOs should be set before resizing the FIFO.

31 ... 16	15 ... 0
reserved	RESET_FIFO

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	RESET_FIFO	Reset Vector, 1 bit per FIFO: 1: reset FIFO 0: normal work mode	R/W	0x0

PTR_FIFO_FULL – POINTER_FIFO Pointer FIFO Full Vector 0x101a4084
PTR_FIFO_FULL – POINTER_FIFO_MOTION Pointer FIFO Full Vector 0x10124084

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

31 ... 16	15 ... 0
reserved	FIFO_FULL

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	FIFO_FULL	FIFO full vector, 1 bit per FIFO	R	0x0

PTR_FIFO_EMPTY – POINTER_FIFO Pointer FIFO Empty Vector 0x101a4088
PTR_FIFO_EMPTY – POINTER_FIFO_MOTION Pointer FIFO Empty Vector 0x10124088

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

31 ... 16	15 ... 0
reserved	FIFO_EMPTY

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	FIFO_EMPTY	FIFO empty vector, 1 bit per FIFO	R	0x0

PTR_FIFO_OVF – POINTER_FIFO Pointer FIFO Overflow Vector 0x101a408c
PTR_FIFO_OVF – POINTER_FIFO_MOTION Pointer FIFO Overflow Vector 0x1012408c

This read only address shows the FIFO_OVERFLOW flag of each FIFO.
If the FIFO had an overflow, it should be reset.

31 ... 16	15 ... 0
reserved	FIFO_OVERFLOW

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	FIFO_OVERFLOW	FIFO overflow vector, 1 bit per FIFO	R	0x0

PTR_FIFO_UDR – POINTER_FIFO Pointer FIFO Underrun Vector 0x101a4090
PTR_FIFO_UDR – POINTER_FIFO_MOTION Pointer FIFO Underrun Vector 0x10124090

This read only address shows the FIFO_UNDERRUN flag of each FIFO.
If the FIFO had an underrun, it should be reset.

31 ... 16	15 ... 0
reserved	FIFO_UNDERRUN

Bits	Name	Description	R/W	Default
31:16	reserved	-	R	0x0
15:0	FIFO_UNDERRUN	FIFO underrun vector, 1 bit per FIFO	R	0x0

PTR_FIFO_FILL_LVL_BASE – POINTER_FIFO Pointer FIFO Fill-Level Table 0x101a40c0
PTR_FIFO_FILL_LVL_BASE – POINTER_FIFO_MOTION Pointer FIFO Fill-Level Table0x101240c0

Each of the following 16 addresses reads the fill-level of the appropriate FIFO.

31 ... 10	9 ... 0
reserved	FILL_LEVEL

Bits	Name	Description	R/W	Default
31:10	reserved	-	R	0x0
9:0	FILL_LEVEL	actual number of words in appropriate FIFO (not valid, if FIFO had an overflow or underrun)	R	0x0

10.3 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 three channels (ARM, Host-CPU/HIF and xPEC) with the appropriate processors, the BMU hardware is enhanced to a 'Quad 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 netX10 consists of 8 Buffer Controllers (channels). Each Buffer Controller can handle a Quad-Buffer between up to three 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_BMU – BUF_MAN BMU 0x101a5600
BUF_MAN_MOTION_BMU – BUF_MAN_MOTION BMU 0x10125600

BMU can be accessed via 3 ports (xPEC, Adr_buf_man, Adr_buf_man_motion).

This register address allows to access 8 buffer controllers, where each one handles buffer numbers (0..3) between up to three 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..7) 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..3).

31 … 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 buf_manager controller of selected channel (buf_nr). This bit will automatically be reset.	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 0x0000ff00 to this register. In parallel mode, the behaviour of all bits of this register changes completely. Parallel mode write access: 7.. 0: Request bits of all 8 channels: 1: request new buffer or semaphore. 0: don't request buffer or semaphore. 15..8: wr bits of all 8 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. ... 15,14: Actual buffer number of channel 7. 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
4:3	reserved	-	R	0x0
2:0	BUF_NR	Write access: number of buf_manager controller (0..7) Read access: number of buffer (0..m+1), where m is the number of masters using this buf_manager	R/W	0x7

10.4 ARM_to_XPEC_IRQ

The following register indicates the interrupt requests between xPEC and ARM.

IRQ_XP0 – IRQs between XPEC0 and ARM Registers **0x101a4400**

31–16	15–0
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

11 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.

11.1 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 three 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
- 3 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 2 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 3 channels due to area optimization
- No big-endian support

11.2 Functional Description

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

The 3 channel DMA controller supports the following transactions in the netX10 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 netX10. 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 obverse 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 2 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 netX10 System

The netX10 SoC has eight AHB masters in total (3x ARM966, 2x xPIC, HIF, DMA controller and XC unit). Due to the netX10 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 priority sequence (highest to lowest is as follows:

1. HIF/DPM
2. XC
3. XPIC data
4. XPIC instruction
5. ARM instruction
6. ARM data
7. ARM system
8. DMA

NetX10 Performance Considerations

The following system considerations are recommended to reduce the latency and to improve the performance of the netX10 SoC:

- Reduce conflicts a priory by separating software of the system processors (ARM966 and XC) running in different memory areas.
- Internal memory is faster than SDRAM, which is faster than flash or SRAM. Keep often used functions/data in internal memory.
- Different masters (especially ARM-instruction and ARM-data) should be assigned to different SDRAM-Banks (e.g. 8M SDRAM with 4 banks: bank0 from address 0 to 2M-1, bank1 from address 2M to 4M-1, …).
  In detail: SDRAM/DDRAM is devided in banks and each bank consists of rows (e.g. 4 banks of 512 rows of 1k Dwords). Each bank has 1 active row. If the address of an SDRAM access points inside one of these active rows, the accesses will be very fast. If it points to a non-active row, the access will be about 10 times slower (precharging losses). The software mapping will dramatically influence these precharging losses.
  E.g.: In a worst case scenario code and data are mapped to the same SDRAM bank and the CPU accesses code and data alternating. In this case each access requires activating a new row, i.e. the maximum precharging losses. By simply mapping code and data in different banks, the same software can be speed up by a factor of upto 10.
- 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.

11.3 Software Interface

This chapter describes the DMA registers and provides details required for programming the DMA in the netX10 SoC.

11.3.1 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. ARM966). 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 three 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_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_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).

11.3.2 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 2 the lowest priority). Usually cannels 0 and 1 are used to handle a peripheral module (rx and tx of one of USB, SPI, SQI, UART, I2C). Channel 2 should be used to handle a list of memory to memory transfers.
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_CH_DEST_ADDR) 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

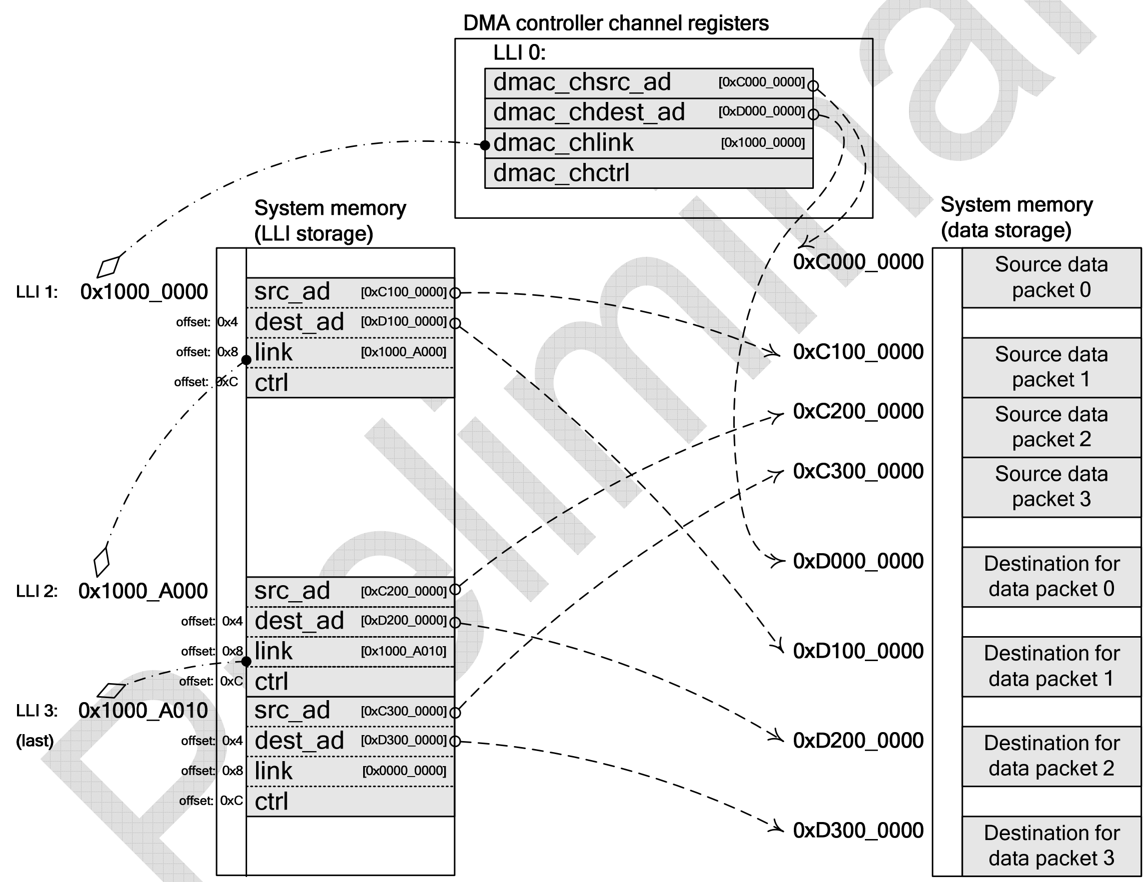
The structure of a single LLI obtains of four words (32bit x 4 data). These words are organized in the following order:

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 netx 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 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”.

11.3.3 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 ARM966 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 ARM966 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.

11.3.4 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, I2C, 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.

11.4 Register Definition

The DMAC has three address areas for each channel (DMAC_CH0 … DMAC_CH2) and one common address area (DMAC_REG).

11.4.1 Base Address Area: DMAC_CH

The following tables define the registers of the three DMA channels. The same registers exist for all three channels at base addresses 0x101c5100, 0x101c5120, and 0x101c5140.

ARM Address	Register Name	Short Description
0x101c5100	DMAC_CH0_SRC_ADDR	Channel0 Source Address Registers
0x101c5104	DMAC_CH0_DEST_ADDR	Channel0 Destination Address Registers
0x101c5108	DMAC_CH0_LINK	Channel0 Linked List Item Register
0x101c510c	DMAC_CH0_CTRL	Channel0 Control Registers
0x101c5110	DMAC_CH0_CFG	Channel0 Configuration Registers
0x101c5120	DMAC_CH1_SRC_ADDR	Channel1 Source Address Registers
0x101c5124	DMAC_CH1_DEST_ADDR	Channel1 Destination Address Registers
0x101c5128	DMAC_CH1_LINK	Channel1 Linked List Item Register
0x101c512c	DMAC_CH1_CTRL	Channel1 Control Registers
0x101c5130	DMAC_CH1_CFG	Channel1 Configuration Registers
0x101c5140	DMAC_CH2_SRC_ADDR	Channel2 Source Address Registers
0x101c5144	DMAC_CH2_DEST_ADDR	Channel2 Destination Address Registers
0x101c5148	DMAC_CH2_LINK	Channel2 Linked List Item Register
0x101c514c	DMAC_CH2_CTRL	Channel2 Control Registers
0x101c5150	DMAC_CH2_CFG	Channel2 Configuration Registers

DMAC_CH0_SRC_ADDR – DMAC_CH0 Channel Source Address Registers 0x101c5100
DMAC_CH1_SRC_ADDR – DMAC_CH1 Channel Source Address Registers 0x101c5120
DMAC_CH2_SRC_ADDR – DMAC_CH2 Channel Source Address Registers 0x101c5140

31	30	29	28	27	26	25	24	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 – DMAC_CH0 Channel Destination Address Registers 0x101c5104
DMAC_CH1_DEST_ADDR – DMAC_CH1 Channel Destination Address Registers 0x101c5124
DMAC_CH2_DEST_ADDR – DMAC_CH2 Channel Destination Address Registers 0x101c5144

31–0
DMACCHDESTADDR

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

DMAC_CH0_LINK – DMAC_CH0 Channel Linked List Item Register 0x101c5108
DMAC_CH1_LINK – DMAC_CH1 Channel Linked List Item Register 0x101c5128
DMAC_CH2_LINK – DMAC_CH2 Channel Linked List Item Register 0x101c5148

31–2	1–0
LLIADDR	reserved

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 – DMAC_CH0 Channel Control Registers 0x101c510c
DMAC_CH1_CTRL – DMAC_CH1 Channel Control Registers 0x101c512c
DMAC_CH2_CTRL – DMAC_CH2 Channel Control Registers 0x101c514c

31	30	29	28	27	26	25	24	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			TRANSFERSIZE											

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	set equal behavior to arm implementation	R/W	0x0
23:21	DWIDTH	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. bit_value data_width ------------------------ 000 8-bit 001 16-bit 010 32-bit ========================	R/W	0x0
20:18	SWIDTH	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. bit_value data_width ------------------------ 000 8-bit 001 16-bit 010 32-bit ========================	R/W	0x0
17:15	DBSIZE	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. bit_value burst_transfer_size ------------------------------- 000 1 001 4 010 8 011 16 100 32 101 64 110 128 111 256 ===============================	R/W	0x0
14:12	SBSIZE	Source burst size. Indicates the number of transfers which make	R/W	0x0

Bits	Name	Description	R/W	Default
		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. ________________________________ 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 – DMAC_CH0 Channel Configuration Registers 0x101c5110
DMAC_CH1_CFG – DMAC_CH1 Channel Configuration Registers 0x101c5130
DMAC_CH2_CFG – DMAC_CH2 Channel Configuration Registers 0x101c5150

31–19	18	17	16	15	14	13–11	10	9–6	5	4–1	0
reserved	H	A	L	ITC	IE	FLOWCNTRL	reserved	DESTPERIPHERAL	reserved	SRCPERIPHERAL	E

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. ____________________ 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
10	reserved	-	R	0x0
9:6	DESTPERIPHERAL	Destination peripheral. This value selects the DMA destination request peripheral. This field is ignored if the destination of the transfer is to memory. ____________________ select_value module ---------------------------- 0 spi0_rx (unused) 1 spi0_tx 2 spi1_rx (unused) 3 spi1_tx 4 uart0_rx (unused) 5 uart0_tx 6 uart1_rx (unused) 7 uart1_tx 8 reserved 9 reserved	R/W	0x0

Bits	Name	Description	R/W	Default
		10 i2c_master_mode 11 i2c_slave_mode 12 adc0 13 adc1 14 usb_dev_uart_rx (unused) 15 usb_dev_uart_tx ============================		
5	reserved	-	R	0x0
4:1	SRCPERIPHERAL	Source peripheral. This value selects the DMA source request peripheral. This field is ignored if the source of the transfer is from memory. ____________________________ select_value module ----------------------------- 0 spi0_rx 1 spi0_tx (unused) 2 spi1_rx 3 spi1_tx (unused) 4 uart0_rx 5 uart0_tx (unused) 6 uart1_rx 7 uart1_tx (unused) 8 reserved 9 reserved 10 i2c_master_mode 11 i2c_slave_mode 12 adc0 13 adc1 14 usb_dev_uart_rx 15 usb_dev_uart_tx (unused) ============================	R/W	0x0
0	E	Channel enable. Reading this bit indicates whether a channel is currently enabled or disabled: 0 = channel disabled 1 = channel enabled. 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. This causes the current AHB transfer (if one is in progress) to complete and the channel is then disabled. 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 that there is no data left in the channels FIFO. Finally the Channel Enable bit can be cleared.	R/W	0x0

11.4.2 Base Address Area: DMAC_REG

ARM Address	Register Name	Short Description
0x101c5800	DMAC_INT_STAT	Interrupt Status Register
0x101c5804	DMAC_INT_TC_STAT	Interrupt Terminal Count Status Register
0x101c5808	DMAC_INT_TC_CLR	Interrupt Terminal Count Clear Register
0x101c580c	DMAC_INT_ERR_STAT	Interrupt Error Status Register
0x101c5810	DMAC_INT_ERR_CLR	Interrupt Error Clear Register
0x101c5814	DMAC_RAW_INT_TC_STAT	Raw Interrupt Terminal Count Status Register
0x101c5818	DMAC_RAW_INT_ERR_STAT	Raw Interrupt Error Status Register
0x101c581c	DMAC_CH_EN	Channel Enable Register
0x101c5820	DMAC_SW_BURST_REQ	Software Burst Request Register
0x101c5824	DMAC_SW_SINGLE_REQ	Software Single Request Register
0x101c5828	DMAC_SW_LAST_BURST_REQ	Software Last Burst Request Register
0x101c582c	DMAC_SW_LAST_SINGLE_REQ	Software Last Single Request Register
0x101c5830	DMAC_CFG	Configuration Register
0x101c5834	DMAC_SYNC	Sync Register

DMAC_INT_STAT – Interrupt Status Register 0x101c5800

31	30	29	28	27	26	25	24	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_CH2	DMACINT_CH1	DMACINT_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c5804

31 … 3	2	1	0
reserved	DMACINTTC_CH2	DMACINTTC_CH1	DMACINTTC_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c5808

31 … 3	2	1	0
reserved	DMACINTTCCLR_CH2	DMACINTTCCLR_CH1	DMACINTTCCLR_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	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 0x101c580c

31:3	2	1	0
reserved	DMACINTERR_CH2	DMACINTERR_CH1	DMACINTERR_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c5810

31:3	2	1	0
reserved	DMACINTERRCLR_CH2	DMACINTERRCLR_CH1	DMACINTERRCLR_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	R	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 0x101c5814

31–3	2	1	0
reserved	DMACRAWINTTC_CH2	DMACRAWINTTC_CH1	DMACRAWINTTC_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c5818

31–3	2	1	0
reserved	DMACRAWINTERR_CH2	DMACRAWINTERR_CH1	DMACRAWINTERR_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c581c

31 … 3	2	1	0
reserved	DMACENABLEDCHNS_CH2	DMACENABLEDCHNS_CH1	DMACENABLEDCHNS_CH0

Bits	Name	Description	R/W	Default
31:3	reserved	-	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 0x101c5820

31 … 16	15 … 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 0x101c5824

31 … 16	15 … 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 0x101c5828

31	30	29	28	27	26	25	24	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 0x101c582c

reset value 0x0

31	30	29	28	27	26	25	24	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 0x101c5830

31	30	29	28	27	26	25	24	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																															DMACENABLE

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 **0x101c5834**

DMA synchronization logic for DMA request signals enabled or disabled A 1’b0 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 HIGH bit indicates that the synchronization logic is disabled.
reset value 0x0

31–14	13	12	11	10	9–8	7	6	5	4	3	2	1	0
reserved	DIS_SYNC_ADC1	DIS_SYNC_ADC0	DIS_SYNC_I2C_TX	DIS_SYNC_I2C_RX	reserved	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
31:14	reserved	-	R	0x0
13	DIS_SYNC_ADC1	disable sync register for ADC1 requests	R/W	0x0
12	DIS_SYNC_ADC0	disable sync register for ADC0 requests	R/W	0x0
11	DIS_SYNC_I2C_TX	disable sync register for I2C transmit requests	R/W	0x0
10	DIS_SYNC_I2C_RX	disable sync register for I2C receive requests	R/W	0x0
9:8	reserved	-	R	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

12 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.

12.1 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.

12.2 CP15 Registers Description

c0 – ID Code Register

31 … 24	23 … 20	19 … 16	15 … 4	3 … 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	The data TCM size is configurable in the range 0 bytes to 64 MB: 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 The supported sizes are 0 and 2nKB for n = 0 to 16	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	The instruction TCM size is configurable in the range 0 bytes to 64 MB: 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 The supported sizes are 0 and 2nKB for n = 0 to 16	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

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="16">SBZ</td><td>LT</td><td>SBZ</td><td>V</td><td>I</td><td colspan="4">SBO</td><td>B</td><td colspan="3">SBO</td><td>W</td><td>D</td><td>A</td><td>SBZ</td></tr>
</table>

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   Registe
```

13 Appendix

13.1 Register Table

ARM Address	Register Name	Short Description
0x101c0004	IO_CFG	IO Configuration Register
0x101c0008	IO_CFG_MSK	IO Config Mask Register
0x101c000c	RESET_CTRL	Reset Control Register
0x101c0010	PHY_CTRL	PHY Control Register
0x101c0014	ARM_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of System Clock
0x101c0018	USB12_CLK_RATE_MUL_ADD	Rate Multiplier Add Value of 12MHz USB clock
0x101c001c	ADC_CLK_DIV	Divisor of clock divider for 16MHz ADC clock
0x101c0020	FB0CLK_RATE_MUL_ADD	Rate Multiplier Add Value
0x101c0024	FB0CLK_DIV	Rate Multiplier Predivider
0x101c0028	CLK_EN	Global Clock Enable Register
0x101c002c	CLK_EN_MSK	Global Clock Enable Mask Register
0x101c0034	ONLY_PORN	Firmware Status register
0x101c0038	NETX_REV	netX Revision Register (written once during bootup)
0x101c0040	SAMPLE_AT_NRES	IO Sampled at Reset Status Register
0x101c0044	NETX_STATUS	netX System Status Configuration Register
0x101c0048	RDY_RUN_CFG	netX RDY/RUN IO System Status Configuration Register
0x101c004c	SYSTEM_STATUS	netX System Status Register
0x101c0050	NETX_LIC_ID	netX License ID Register
0x101c0054	NETX_LIC_FLAGS0	netX License Flags0 Register
0x101c0058	NETX_LIC_FLAGS1	netX License Flags1 Register
0x101c005c	NETX_LIC_ERRORS0	netX License Errors0 Status Register
0x101c0060	NETX_LIC_ERRORS1	netX License Errors1 Status Register
0x101c0204	WDG_CNTR	Watchdog Counter
0x101c0208	WDG_IRQ_TIMEOUT	Watchdog Interrupt Timeout
0x101c020c	WDG_RESET_TIMEOUT	Watchdog Reset Timeout
0x101c0a00	MMIO0_CFG	Multiplex matrix Configuration Register for MMIO0
0x101c0a04	MMIO1_CFG	Multiplex matrix Configuration Register for MMIO1
0x101c0a08	MMIO2_CFG	Multiplex matrix Configuration Register for MMIO2
0x101c0a0c	MMIO3_CFG	Multiplex matrix Configuration Register for MMIO3
0x101c0a10	MMIO4_CFG	Multiplex matrix Configuration Register for MMIO4
0x101c0a14	MMIO5_CFG	Multiplex matrix Configuration Register for MMIO5
0x101c0a18	MMIO6_CFG	Multiplex matrix Configuration Register for MMIO6
0x101c0a1c	MMIO7_CFG	Multiplex matrix Configuration Register for MMIO7
0x101c0a20	MMIO8_CFG	Multiplex matrix Configuration Register for MMIO8
0x101c0a24	MMIO9_CFG	Multiplex matrix Configuration Register for MMIO9
0x101c0a28	MMIO10_CFG	Multiplex matrix Configuration Register for MMIO10
0x101c0a2c	MMIO11_CFG	Multiplex matrix Configuration Register for MMIO11
0x101c0a30	MMIO12_CFG	Multiplex matrix Configuration Register for MMIO12
0x101c0a34	MMIO13_CFG	Multiplex matrix Configuration Register for MMIO13
0x101c0a38	MMIO14_CFG	Multiplex matrix Configuration Register for MMIO14
0x101c0a3c	MMIO15_CFG	Multiplex matrix Configuration Register for MMIO15
0x101c0a40	MMIO16_CFG	Multiplex matrix Configuration Register for MMIO16
0x101c0a44	MMIO17_CFG	Multiplex matrix Configuration Register for MMIO17
0x101c0a48	MMIO18_CFG	Multiplex matrix Configuration Register for MMIO18

ARM Address	Register Name	Short Description
0x101c0a4c	MMIO19_CFG	Multiplex matrix Configuration Register for MMIO19
0x101c0a50	MMIO20_CFG	Multiplex matrix Configuration Register for MMIO20
0x101c0a54	MMIO21_CFG	Multiplex matrix Configuration Register for MMIO21
0x101c0a58	MMIO22_CFG	Multiplex matrix Configuration Register for MMIO22
0x101c0a5c	MMIO23_CFG	Multiplex matrix Configuration Register for MMIO23
0x101c0a60	MMIO_PIO_OUT_LINE_CFG	MMIO PIO Line Output Level Register
0x101c0a64	MMIO_PIO_OE_LINE_CFG	MMIO PIO Line Output Enable Register
0x101c0a68	MMIO_IN_LINE_STATUS	MMIO Input Line Register
0x101c0a6c	MMIO_IS_PIO_STATUS	MMIO Mode Line Register
0x101c0c40	HIF_IO_CFG	HIF IO Config Register
0x101c0c44	HIF_PIO_OUT0	HIF PIO Output State Configuration Register 0
0x101c0c48	HIF_PIO_OUT1	HIF PIO Output State Configuration Register 1
0x101c0c4c	HIF_PIO_OE0	HIF PIO Output Enable Configuration Register 0
0x101c0c50	HIF_PIO_OE1	HIF PIO Output Enable Configuration Register 1
0x101c0c54	HIF_PIO_IN0	HIF PIO Input State Register 0
0x101c0c58	HIF_PIO_IN1	HIF PIO Input State Register 1
0x101c12d8	SYS_STAT	System Status
0x101c0100	MEM_SRAM0_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem0 Chip Select Area
0x101c0104	MEM_SRAM1_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem1 Chip Select Area
0x101c0108	MEM_SRAM2_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem2 Chip Select Area
0x101c010c	MEM_SRAM3_CTRL	Control Register for External Bus Interface and Wait-States for ExtMem3 Chip Select Area
0x101c0110	EXT_CS0_APM_CTRL	Asynchronous Page Mode (APM) Control Register for ExtMem0 Chip Select Area
0x101c0120	EXT_RDY_CFG	External Memory Ready Control Register
0x101c0124	EXT_RDY_STATUS	External Memory Ready Status Register
0x101c0140	MEM_SDRAM_CFG_CTRL	Memory SDRAM Configuration Control Register
0x101c0144	MEM_SDRAM_TIMING_CTRL	Memory SDRAM Timing Control Register
0x101c0148	MEM_SDRAM_MODE	Memory SDRAM Mode Register
0x101c1200	DPM_CFG0X0	DPM IO Control Register 0
0x101c1210	DPM_ADDR_CFG	DPM External Address Range Configuration Register
0x101c1214	DPM_TIMING_CFG	DPM Timing Configuration Register
0x101c1218	DPM_RDY_CFG	DPM Ready (DPM_RDY) Signal Configuration Register
0x101c121c	DPM_STATUS	DPM Status Register
0x101c1220	DPM_STATUS_ERR_RESET	DPM Error Status Reset Register
0x101c1224	DPM_STATUS_ERR_ADDR	DPM Error Address Status Register
0x101c1228	DPM_MISC_CFG	DPM Configuration Register for some Special Functions
0x101c122c	DPM_IO_CFG_MISC	DPM IO Configuration Register 0
0x101c1238	DPM_TUNNEL_CFG	DPM Access Tunnel Configuration Register
0x101c123c	DPM_ITBADDR	DPM Access Tunnel (DATunnel) netX Internal Target Base Address (ITBAddr) Configuration Register
0x101c1240	DPM_WIN1_END	DPM Window 1 End Address Configuration Register
0x101c1244	DPM_WIN1_MAP	DPM Window 1 Address Map Configuration Register
0x101c1248	DPM_WIN2_END	DPM Window 2 End Address Configuration Register
0x101c124c	DPM_WIN2_MAP	DPM Window 2 Address Map Configuration Register
0x101c1250	DPM_WIN3_END	DPM Window 3 End Address Configuration Register
0x101c1254	DPM_WIN3_MAP	DPM Window 3 Address Map Configuration Register
0x101c1258	DPM_WIN4_END	DPM Window 4 End Address Configuration Register
0x101c125c	DPM_WIN4_MAP	DPM Window 4 Address Map Configuration Register
0x101c1280	DPM_IRQ_RAW	DPM Raw (before masking) IRQ Status Register

ARM Address	Register Name	Short Description
0x101c1284	DPM_IRQ_ARM_MASK_SET	DPM Interrupt Mask Register for netX Internal ARM
0x101c1288	DPM_IRQ_ARM_MASK_RESET	DPM Interrupt Mask Reset Register for netX Internal ARM
0x101c128c	DPM_IRQ_ARM_MASKED	DPM Masked Interrupt Status Register for netX Internal ARM
0x101c1290	DPM_IRQ_XPIC_MASK_SET	DPM Interrupt Mask Register for netX Internal xPIC
0x101c1294	DPM_IRQ_XPIC_MASK_RESET	DPM Interrupt Mask Reset Register for netX Internal xPIC
0x101c1298	DPM_IRQ_XPIC_MASKED	DPM Masked Interrupt Status Register for netX Internal xPIC
0x101c129c	DPM_IRQ_FIQ_MASK_SET	DPM Fast/SIRQ Interrupt Mask Register
0x101c12a0	DPM_IRQ_FIQ_MASK_RESET	DPM Fast/SIRQ Interrupt Mask Register
0x101c12a4	DPM_IRQ_FIQ_MASKED	DPM Masked Fast/SIRQ Interrupt Status Register
0x101c12a8	DPM_IRQ_IRQ_MASK_SET	DPM Normal/DIRQ Interrupt Mask Register
0x101c12ac	DPM_IRQ_IRQ_MASK_RESET	DPM Normal/DIRQ Interrupt Mask Register
0x101c12b0	DPM_IRQ_IRQ_MASKED	DPM Masked Normal/DIRQ Interrupt Status Register
0x101c12c0	DPM_HOST_WDG_HOST_TIMEOUT	Address reserved for netX50
0x101c12c4	DPM_HOST_WDG_HOST_TRIG	Address reserved for netX50
0x101c12c8	DPM_HOST_WDG_ARM_TIMEOUT	Address reserved for netX50
0x101c12cc	DPM_SYS_STA_BIGEND16	DPM System Status Information Register in Big Endianess 16 Data Mapping
0x101c12d0	DPM_HOST_TMR_CTRL	Address reserved for netX50
0x101c12d4	DPM_HOST_TMR_START_VAL	Address reserved for netX50
0x101c12d8	DPM_HOST_SYS_STAT	DPM System Status Information Register
0x101c12dc	DPM_HOST_RESET_REQ	DPM Reset Request Register
0x101c12e0	DPM_HOST_INT_STAT0	DPM Handshake Interrupt Status Register
0x101c12f0	DPM_HOST_INT_EN0	DPM Handshake Interrupt Enable Register
0x101c12f8	DPM_NETX_VERSION_BIGEND16	DPM netX Version Register in Big Endianess 16 Data Mapping
0x101c12fc	DPM_NETX_VERSION	DPM netX Version Register
0x101c1100	HANDSHAKE_BASE_ADDR	Handshake Cell Address Base Configuration Register
0x101c1110	HANDSHAKE_DPM_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for DPM Register
0x101c1114	HANDSHAKE_DPM_IRQ_MASKED	Handshake Cell Masked Interrupt for DPM Register
0x101c1118	HANDSHAKE_DPM_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for DPM Register
0x101c111c	HANDSHAKE_DPM_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for DPM Register
0x101c1120	HANDSHAKE_ARM_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for ARM Register
0x101c1124	HANDSHAKE_ARM_IRQ_MASKED	Handshake Cell Masked Interrupt for ARM Register
0x101c1128	HANDSHAKE_ARM_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for ARM Register
0x101c112c	HANDSHAKE_ARM_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for ARM Register
0x101c1130	HANDSHAKE_XPIC_IRQ_RAW_CLEAR	Handshake Cell Raw Interrupt for xPIC Register
0x101c1134	HANDSHAKE_XPIC_IRQ_MASKED	Handshake Cell Masked Interrupt for xPIC Register
0x101c1138	HANDSHAKE_XPIC_IRQ_MSK_SET	Handshake Cell Interrupt Mask Enable for xPIC Register
0x101c113c	HANDSHAKE_XPIC_IRQ_MSK_RESET	Handshake Cell Interrupt Mask Disable for xPIC Register
0x101c1180	HANDSHAKE_HSC0_CTRL	Handshake Cell 0 Control Register
0x101c1184	HANDSHAKE_HSC1_CTRL	Handshake Cell 1 Control Register
0x101c1188	HANDSHAKE_HSC2_CTRL	Handshake Cell 2 Control Register
0x101c118c	HANDSHAKE_HSC3_CTRL	Handshake Cell 3 Control Register
0x101c1190	HANDSHAKE_HSC4_CTRL	Handshake Cell 4 Control Register
0x101c1194	HANDSHAKE_HSC5_CTRL	Handshake Cell 5 Control Register
0x101c1198	HANDSHAKE_HSC6_CTRL	Handshake Cell 6 Control Register
0x101c119c	HANDSHAKE_HSC7_CTRL	Handshake Cell 7 Control Register
0x101c11a0	HANDSHAKE_HSC8_CTRL	Handshake Cell 8 Control Register
0x101c11a4	HANDSHAKE_HSC9_CTRL	Handshake Cell 9 Control Register
0x101c11a8	HANDSHAKE_HSC10_CTRL	Handshake Cell 10 Control Register

ARM Address	Register Name	Short Description
0x101c11ac	HANDSHAKE_HSC11_CTRL	Handshake Cell 11 Control Register
0x101c11b0	HANDSHAKE_HSC12_CTRL	Handshake Cell 12 Control Register
0x101c11b4	HANDSHAKE_HSC13_CTRL	Handshake Cell 13 Control Register
0x101c11b8	HANDSHAKE_HSC14_CTRL	Handshake Cell 14 Control Register
0x101c11bc	HANDSHAKE_HSC15_CTRL	Handshake Cell 15 Control Register
0x101c11c0	HANDSHAKE_BUF_MAN0_CTRL	Handshake Triple Buffer Manager 0 Control Register
0x101c11c4	HANDSHAKE_BUF_MAN0_STATUS_CTRL_NETX	Handshake Triple Buffer Manager 0 netX Status and Control Register
0x101c11c8	HANDSHAKE_BUF_MAN0_STATUS_CTRL_HOST	Handshake Triple Buffer Manager 0 Host Status Register
0x101c11cc	HANDSHAKE_BUF_MAN0_WIN_MAP	DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 0
0x101c11d0	HANDSHAKE_BUF_MAN1_CTRL	Handshake Triple Buffer Manager 1 Control Register
0x101c11d4	HANDSHAKE_BUF_MAN1_STATUS_CTRL_NETX	Handshake Triple Buffer Manager 1 netX Status and Control Register
0x101c11d8	HANDSHAKE_BUF_MAN1_STATUS_CTRL_HOST	Handshake Triple Buffer Manager 1 Host Status Register
0x101c11dc	HANDSHAKE_BUF_MAN1_WIN_MAP	DPM Window Address Map Alternative Configuration Register for Handshake Triple Buffer Manager 1
0x00048000	HANDSHAKE_MIRROR_ITCM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0004fffc	HANDSHAKE_MIRROR_ITCM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x04048000	HANDSHAKE_MIRROR_DTCM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0404fffc	HANDSHAKE_MIRROR_DTCM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x08048000	HANDSHAKE_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x0804fffc	HANDSHAKE_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x10048000	HANDSHAKE_MIRROR_DPM_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0x1004fffc	HANDSHAKE_MIRROR_DPM_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0xfff48000	HANDSHAKE_MIRROR_HI_HANDSHAKE_BASE	Internal Handshake AHBL Slave 5 Start Address
0xfff4fffc	HANDSHAKE_MIRROR_HI_HANDSHAKEEND	Internal SRAM AHBL Slave 5 End Address
0x10140800	XPIC_VIC_CONFIG	XPIC VIC Config Register
0x10140804	XPIC_VIC_RAW_INTR0	XPIC VIC Raw0 Interrupt Status Register
0x10140808	XPIC_VIC_RAW_INTR1	XPIC VIC Raw1 Interrupt Status Register
0x1014080c	XPIC_VIC_SOFTINT0_SET	XPIC VIC Software0 Interrupt Set Register
0x10140810	XPIC_VIC_SOFTINT1_SET	XPIC VIC Software1 Interrupt Set Register
0x10140814	XPIC_VIC_SOFTINT0_RESET	XPIC VIC Software0 Interrupt Reset Register
0x10140818	XPIC_VIC_SOFTINT1_RESET	XPIC VIC Software1 Interrupt Reset Register
0x1014081c	XPIC_VIC_FIQ_ADDR	XPIC VIC FIQ Vector Address 0 Register
0x10140820	XPIC_VIC_IRQ_ADDR	XPIC VIC Normal IRQ Address Register
0x10140824	XPIC_VIC_VECTOR_ADDR	XPIC VIC IRQ Vector Address
0x10140828	XPIC_VIC_TABLE_BASE_ADDR	XPIC VIC IRQ Table Base Address
0x1014082c	XPIC_VIC_FIQ_VECT_CONFIG	XPIC VIC FIQ Vector Config Register
0x10140830	XPIC_VIC_VECT_CONFIG0	XPIC VIC IRQ Vector0 Config Register
0x10140834	XPIC_VIC_VECT_CONFIG1	XPIC VIC IRQ Vector1 Config Register
0x10140838	XPIC_VIC_VECT_CONFIG2	XPIC VIC IRQ Vector2 Config Register
0x1014083c	XPIC_VIC_VECT_CONFIG3	XPIC VIC IRQ Vector3 Config Register
0x10140840	XPIC_VIC_VECT_CONFIG4	XPIC VIC IRQ Vector4 Config Register
0x10140844	XPIC_VIC_VECT_CONFIG5	XPIC VIC IRQ Vector5 Config Register
0x10140848	XPIC_VIC_VECT_CONFIG6	XPIC VIC IRQ Vector6 Config Register
0x1014084c	XPIC_VIC_VECT_CONFIG7	XPIC VIC IRQ Vector7 Config Register
0x10140850	XPIC_VIC_VECT_CONFIG8	XPIC VIC IRQ Vector8 Config Register

ARM Address	Register Name	Short Description
0x10140854	XPIC_VIC_VECT_CONFIG9	XPIC VIC IRQ Vector9 Config Register
0x10140858	XPIC_VIC_VECT_CONFIG10	XPIC VIC IRQ Vector10 Config Register
0x1014085c	XPIC_VIC_VECT_CONFIG11	XPIC VIC IRQ Vector11 Config Register
0x10140860	XPIC_VIC_VECT_CONFIG12	XPIC VIC IRQ Vector12 Config Register
0x10140864	XPIC_VIC_VECT_CONFIG13	XPIC VIC IRQ Vector13 Config Register
0x10140868	XPIC_VIC_VECT_CONFIG14	XPIC VIC IRQ Vector14 Config Register
0x1014086c	XPIC_VIC_VECT_CONFIG15	XPIC VIC IRQ Vector15 Config Register
0x10140870	XPIC_VIC_DEFAULT0	XPIC Default Interrupt Vector Select0
0x10140874	XPIC_VIC_DEFAULT1	XPIC Default Interrupt Vector Select1
0x10140700	XPIC_TIMER_CONFIG_TIMER0	xPIC TIMER Config Register0
0x10140704	XPIC_TIMER_CONFIG_TIMER1	xPIC TIMER Config Register1
0x10140708	XPIC_TIMER_CONFIG_TIMER2	xPIC TIMER Config Register2
0x1014070c	XPIC_TIMER_PRELOAD_TIMER0	xPIC TIMER Timer 0 Preload
0x10140710	XPIC_TIMER_PRELOAD_TIMER1	xPIC TIMER Timer 1 Preload
0x10140714	XPIC_TIMER_PRELOAD_TIMER2	xPIC TIMER Timer 2 Preload
0x10140718	XPIC_TIMER_TIMER0	xPIC TIMER Timer 0
0x1014071c	XPIC_TIMER_TIMER1	xPIC TIMER Timer 1
0x10140720	XPIC_TIMER_TIMER2	xPIC TIMER Timer 2
0x10140724	XPIC_TIMER_IRQ_RAW	xPIC_TIMER Raw IRQ Register
0x10140728	XPIC_TIMER_IRQ_MASKED	xPIC_TIMER Masked IRQ Register
0x1014072c	XPIC_TIMER_IRQ_MSK_SET	xPIC_TIMER Interrupt Mask Enable
0x10140730	XPIC_TIMER_IRQ_MSK_RESET	xPIC_TIMER Interrupt Mask Disable
0x10140734	XPIC_TIMER_SYSTIME_S	xPIC_TIMER Upper SYSTIME Register
0x10140738	XPIC_TIMER_SYSTIME_NS	xPIC_TIMER Lower SYSTIME Register
0x1014073c	XPIC_TIMER_COMPARE_SYSTIME_S_VALUE	xPIC_TIMER SYSTIME Sec Compare Register
0x10140900	XPIC_WDG_TRIG	xPIC Watchdog Trigger Register
0x10140904	XPIC_WDG_COUNTER	xPIC Watchdog Counter Register
0x10140908	XPIC_WDG_XPIC_IRQ_TIMEOUT	xPIC Watchdog xPIC Interrupt Timout Register
0x1014090c	XPIC_WDG_ARM_IRQ_TIMEOUT	xPIC Watchdog ARM Interrupt Timout Register
0x10140910	XPIC_WDG_IRQ_RAW	xPIC Watchdog Raw Interrupt Register
0x10140914	XPIC_WDG_IRQ_MASKED	xPIC Watchdog Masked IRQ Register
0x10140918	XPIC_WDG_IRQ_MSK_SET	xPIC Watchdog Interrupt Mask Enable
0x1014091c	XPIC_WDG_IRQ_MSK_RESET	xPIC Watchdog Interrupt Mask Disable
0x10140500	MPWM_CONFIG_COUNTER	Counter Config Register
0x10140504	MPWM_CONFIG_PINS	Pins Config Register
0x10140508	MPWM_CONFIG_FAILURE	Failure Config Register
0x1014050c	MPWM_IRQ_CONFIG	IRQ Config Register
0x10140510	MPWM_IRQ_RAW	Raw IRQ
0x10140514	MPWM_IRQ_MASKED	Masked IRQ
0x10140518	MPWM_IRQ_MSK_SET	IRQ Enable Mask
0x1014051c	MPWM_IRQ_MSK_RESET	IRQ Disable Mask
0x10140520	MPWM_CNT0_PERIOD	Counter 0 Period
0x10140524	MPWM_CNT0	Counter 0 Value
0x10140528	MPWM_CNT0_SYSTIME	Counter 0 Start Systime
0x1014052c	MPWM_CNT0_WATCHDOG	Counter 0 Watchdog
0x10140530	MPWM_CNT1_PERIOD	Counter 1 Period
0x10140534	MPWM_CNT1	Counter 1 Value
0x10140538	MPWM_CNT1_SYSTIME	Counter 1 Start Systime

ARM Address	Register Name	Short Description
0x1014053c	MPWM_CNT1_WATCHDOG	Counter 1 Watchdog
0x10140540	MPWM_T0	PWM Channel 0 Threshold
0x10140544	MPWM_T1	PWM Channel 1 Threshold
0x10140548	MPWM_T2	PWM Channel 2 Threshold
0x1014054c	MPWM_T3	PWM Channel 3 Threshold
0x10140550	MPWM_T4	PWM Channel 4 Threshold
0x10140554	MPWM_T5	PWM Channel 5 Threshold
0x10140558	MPWM_T6	PWM Channel 6 Threshold
0x1014055c	MPWM_T7	PWM Channel 7 Threshold
0x10140560	MPWM_T0_SHADOW	PWM Channel 0 Threshold Shadow
0x10140564	MPWM_T1_SHADOW	PWM Channel 1 Threshold Shadow
0x10140568	MPWM_T2_SHADOW	PWM Channel 2 Threshold Shadow
0x1014056c	MPWM_T3_SHADOW	PWM Channel 3 Threshold Shadow
0x10140570	MPWM_T4_SHADOW	PWM Channel 4 Threshold Shadow
0x10140574	MPWM_T5_SHADOW	PWM Channel 5 Threshold Shadow
0x10140578	MPWM_T6_SHADOW	PWM Channel 6 Threshold Shadow
0x1014057c	MPWM_T7_SHADOW	PWM Channel 7 Threshold Shadow
0x10140580	MENC_CONFIG	Encoder Configuration Register
0x10140584	MENC_ENC0_POSITION	Position of Encoder 0
0x10140588	MENC_ENC1_POSITION	Position of Encoder 1
0x1014058c	MENC_CAPTURE_NOW	Capture Now Register
0x10140590	MENC_CAPTURE0_CONFIG	Capture Unit 0 Configuration Register
0x10140594	MENC_CAPTURE0_VAL	Capture Unit 0 Captured Value
0x10140598	MENC_CAPTURE0_TA	Capture Unit 0 Ta
0x1014059c	MENC_CAPTURE0_TE	Capture Unit 0 Te
0x101405a0	MENC_CAPTURE1_CONFIG	Capture Unit 1 Configuration Register
0x101405a4	MENC_CAPTURE1_VAL	Capture Unit 1 Captured Value
0x101405a8	MENC_CAPTURE1_TA	Capture Unit 1 Ta
0x101405ac	MENC_CAPTURE1_TE	Capture Unit 1 Te
0x101405b0	MENC_CAPTURE2_CONFIG	Capture Unit 2 Configuration Register
0x101405b4	MENC_CAPTURE2_VAL	Capture Unit 2 Captured Value
0x101405b8	MENC_CAPTURE2_TA	Capture Unit 2 Ta
0x101405bc	MENC_CAPTURE2_TE	Capture Unit 2 Te
0x101405c0	MENC_CAPTURE3_CONFIG	Capture Unit 3 Configuration Register
0x101405c4	MENC_CAPTURE3_VAL	Capture Unit 3 Captured Value
0x101405c8	MENC_CAPTURE3_TA	Capture Unit 3 Ta
0x101405cc	MENC_CAPTURE3_TE	Capture Unit 3 Te
0x101405d0	MENC_STATUS	Position and Capture Status
0x101405d4	MENC_IRQ_MASKED	Masked IRQ Register
0x101405d8	MENC_IRQ_MSK_SET	IRQ Mask Enable
0x101405dc	MENC_IRQ_MSK_RESET	IRQ Mask Disable
0x101406c0	ADC_CTRL_START	ADC Start Register
0x101406c4	ADC_CTRL_AUTOSAMPLE_CONFIG0	ADC0 Config Register for Autosample Mode
0x101406c8	ADC_CTRL_AUTOSAMPLE_CONFIG1	ADC1 Config Register for Autosample Mode
0x101406cc	ADC_CTRL_MANSAMPLE_CONFIG0	ADC0 Config Register for Direct Control
0x101406d0	ADC_CTRL_MANSAMPLE_CONFIG1	ADC1 Config Register for Direct Control
0x101406d4	ADC_CTRL_STATUS	ADC Status Register
0x101406d8	ADC_CTRL_ADC0_VAL	ADC0 Value

ARM Address	Register Name	Short Description
0x101406dc	ADC_CTRL_ADC1_VAL	ADC1 Value
0x101406e0	ADC_CTRL_IRQ_RAW	Raw IRQ
0x101406e4	ADC_CTRL_IRQ_MASKED	Masked IRQ
0x101406e8	ADC_CTRL_IRQ_MSK_SET	IRQ Mask Enable
0x101406ec	ADC_CTRL_IRQ_MSK_RESET	IRQ Mask Disable
0x10140000	CORDIC_CTRL	CORDIC Control Register
0x10140004	CORDIC_X_REG	CORDIC Argument and Result Register X
0x10140008	CORDIC_Y_REG	CORDIC Argument and Result Register Y
0x1014000c	CORDIC_Z_REG	CORDIC Argument and Result Register Z
0x10140010	CORDIC_C_REG	CORDIC Argument Register C
0x10140014	CORDIC_FSM_STATE	CORDIC FSM State Register
0x10140018	CORDIC_LIN_39_TO_8	CORDIC Linear Coefficient Register
0x1014001c	CORDIC_LIN_7_TO_0	CORDIC Linear Coefficient Register
0x10140100	CORDIC_COEFF_RAM_START_CIRC_39_TO_8	Start of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^{-i}))
0x1014019c	CORDIC_COEFF_RAM_END_CIRC_39_TO_8	End of CORDIC Coefficient RAM Containing Most Significant DWords of Circular Coefficients (arctan(2^{-i}))
0x10140200	CORDIC_COEFF_RAM_START_HYP_39_TO_8	Start of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^{-1}))
0x1014029c	CORDIC_COEFF_RAM_END_HYP_39_TO_8	End of CORDIC Coefficient RAM Containing Most Significant DWords of Hyperbolic Coefficients (arctanh(2^{-1}))
0x10140300	CORDIC_COEFF_RAM_START_CIRC_7_TO_0	Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^{-i}))
0x1014034c	CORDIC_COEFF_RAM_END_CIRC_7_TO_0	End of CORDIC Coefficient RAM Containing Least Significant Bytes of Circular Coefficients (arctan(2^{-i}))
0x10140350	CORDIC_COEFF_RAM_START_HYP_7_TO_0	Start of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^{-1}))
0x1014039c	CORDIC_COEFF_RAM_END_HYP_7_TO_0	End of CORDIC Coefficient RAM Containing Least Significant Bytes of Hyperbolic Coefficients (arctanh(2^{-1}))
0x101c0800	GPIO_CFG0	GPIO 0 Configuration Register
0x101c0804	GPIO_CFG1	GPIO 1 Configuration Register
0x101c0808	GPIO_CFG2	GPIO 2 Configuration Register
0x101c080c	GPIO_CFG3	GPIO 3 Configuration Register
0x101c0810	GPIO_CFG4	GPIO 4 Configuration Register
0x101c0814	GPIO_CFG5	GPIO 5 Configuration Register
0x101c0818	GPIO_CFG6	GPIO 6 Configuration Register
0x101c081c	GPIO_CFG7	GPIO 7 Configuration Register
0x101c0820	GPIO_THRSH_CAPT0	GPIO 0 Threshold or Capture Register
0x101c0824	GPIO_THRSH_CAPT1	GPIO 1 Threshold or Capture Register
0x101c0828	GPIO_THRSH_CAPT2	GPIO 2 Threshold or Capture Register
0x101c082c	GPIO_THRSH_CAPT3	GPIO 3 Threshold or Capture Register
0x101c0830	GPIO_THRSH_CAPT4	GPIO 4 Threshold or Capture Register
0x101c0834	GPIO_THRSH_CAPT5	GPIO 5 Threshold or Capture Register
0x101c0838	GPIO_THRSH_CAPT6	GPIO 6 Threshold or Capture Register
0x101c083c	GPIO_THRSH_CAPT7	GPIO 7 Threshold or Capture Register
0x101c0840	GPIO_CNTR0_CTRL	GPIO Counter0 Control Register
0x101c0844	GPIO_CNTR1_CTRL	GPIO Counter1 Control Register
0x101c0848	GPIO_CNTR0_MAX	GPIO Counter0 Max Value
0x101c084c	GPIO_CNTR1_MAX	GPIO Counter1 Max Value
0x101c0850	GPIO_CNTR0_CNT	GPIO Counter0 Current Value
0x101c0854	GPIO_CNTR1_CNT	GPIO Counter1 Current Value
0x101c0858	GPIO_OUT	GPIO Output Register
0x101c085c	GPIO_IN	GPIO Input Register
0x101c0860	GPIO_IRQ_RAW	GPIO Raw IRQ Register

ARM Address	Register Name	Short Description
0x101c0864	GPIO_IRQ_MSK	GPIO Masked IRQ Register
0x101c0868	GPIO_IRQ_MSK_SET	GPIO Interrupt Mask Enable
0x101c086c	GPIO_IRQ_MSK_RESET	GPIO Interrupt Mask Disable
0x101c0870	CNTR_IRQ_RAW	GPIO Counter Raw IRQ Register
0x101c0874	GPIO_CNTR_IRQ_MSK	GPIO Counter Masked IRQ Register
0x101c0878	GPIO_CNTR_IRQ_MSK_SET	GPIO Counter Interrupt Mask Enable
0x101c087c	GPIO_CNTR_IRQ_MSK_RESET	GPIO Counter Interrupt Mask Disable
0x10140400	GPIO_CFG0	GPIO_MOTION 0 Configuration Register
0x10140404	GPIO_CFG1	GPIO_MOTION 1 Configuration Register
0x10140408	GPIO_CFG2	GPIO_MOTION 2 Configuration Register
0x1014040c	GPIO_CFG3	GPIO_MOTION 3 Configuration Register
0x10140410	GPIO_CFG4	GPIO_MOTION 4 Configuration Register
0x10140414	GPIO_CFG5	GPIO_MOTION 5 Configuration Register
0x10140418	GPIO_CFG6	GPIO_MOTION 6 Configuration Register
0x1014041c	GPIO_CFG7	GPIO_MOTION 7 Configuration Register
0x10140420	GPIO_THRSH_CAPT0	GPIO_MOTION 0 Threshold or Capture Register
0x10140424	GPIO_THRSH_CAPT1	GPIO_MOTION 1 Threshold or Capture Register
0x10140428	GPIO_THRSH_CAPT2	GPIO_MOTION 2 Threshold or Capture Register
0x1014042c	GPIO_THRSH_CAPT3	GPIO_MOTION 3 Threshold or Capture Register
0x10140430	GPIO_THRSH_CAPT4	GPIO_MOTION 4 Threshold or Capture Register
0x10140434	GPIO_THRSH_CAPT5	GPIO_MOTION 5 Threshold or Capture Register
0x10140438	GPIO_THRSH_CAPT6	GPIO_MOTION 6 Threshold or Capture Register
0x1014043c	GPIO_THRSH_CAPT7	GPIO_MOTION 7 Threshold or Capture Register
0x10140440	GPIO_CNTR0_CTRL	GPIO_MOTION Counter0 Control Register
0x10140444	GPIO_CNTR1_CTRL	GPIO_MOTION Counter1 Control Register
0x10140448	GPIO_CNTR0_MAX	GPIO_MOTION Counter0 Max Value
0x1014044c	GPIO_CNTR1_MAX	GPIO_MOTION Counter1 Max Value
0x10140450	GPIO_CNTR0_CNT	GPIO_MOTION Counter0 Current Value
0x10140454	GPIO_CNTR1_CNT	GPIO_MOTION Counter1 Current Value
0x10140458	GPIO_OUT	GPIO_MOTION Output Register
0x1014045c	GPIO_IN	GPIO_MOTION Input Register
0x10140460	GPIO_IRQ_RAW	GPIO_MOTION Raw IRQ Register
0x10140464	GPIO_MOTION _IRQ_MSK	GPIO_MOTION Masked IRQ Register
0x10140468	GPIO_MOTION _IRQ_MSK_SET	GPIO_MOTION Interrupt Mask Enable
0x1014046c	GPIO_MOTION _IRQ_MSK_RESET	GPIO_MOTION Interrupt Mask Disable
0x10140470	CNT_IRQ_RAW	GPIO_MOTION Counter Raw IRQ Register
0x10140474	GPIO_MOTION_CNTR_IRQ_MSK	GPIO_MOTION Counter Masked IRQ Register
0x10140478	GPIO_MOTION_CNTR_IRQ_MSK_SET	GPIO_MOTION Counter Interrupt Mask Enable
0x1014047c	GPIO_MOTION_CNTR_IRQ_MSK_RESET	GPIO_MOTION Counter Interrupt Mask Disable
0x101c0900	ARM_TIMER_CONFIG_TIMER0	ARM TIMER Config Register0
0x101c0904	ARM_TIMER_CONFIG_TIMER1	ARM TIMER Config Register1
0x101c0908	ARM_TIMER_PRELOAD_TIMER0	ARM TIMER Timer 0
0x101c090c	ARM_TIMER_PRELOAD_TIMER1	ARM TIMER Timer 1
0x101c0910	ARM_TIMER_TIMER0	ARM TIMER Timer 0
0x101c0914	ARM_TIMER_TIMER1	ARM TIMER Timer 1
0x101c0918	SYS_TIME_S	ARM_TIMER Upper SYSTIME Register
0x101c091c	SYS_TIME_NS	ARM_TIMER Lower SYSTIME Register
0x101c0920	ARM_TIMER_SYSTIME_NS_COMPARE	SYSTIME Nano Sec Compare Value

ARM Address	Register Name	Short Description
0x101c0924	ARM_TIMER_SYSTIME_S_COMPARE	SYSTIME Sec Compare Value
0x101c0928	ARM_TIMER_IRQ_RAW	ARM_TIMER Raw IRQ Register
0x101c092c	ARM_TIMER_IRQ_MASKED	ARM_TIMER Masked IRQ Register
0x101c0930	ARM_TIMER_IRQ_MSK_SET	ARM_TIMER Interrupt Mask Enable
0x101c0934	ARM_TIMER_IRQ_MSK_RESET	ARM_TIMER Interrupt Mask Disable
0x10140600	XLINK0_XLINK_CFG	XLINK0 Configuration Register
0x10140604	XLINK0_XLINK_TX	XLINK0 Transmit Register
0x10140608	XLINK0_XLINK_RX	XLINK0 RX Register
0x1014060c	XLINK0_XLINK_STAT	XLINK0 Status Register
0x10140610	XLINK1_XLINK_CFG	XLINK1 Configuration Register
0x10140614	XLINK1_XLINK_TX	XLINK1 Transmit Register
0x10140618	XLINK1_XLINK_RX	XLINK1 RX Register
0x1014061c	XLINK1_XLINK_STAT	XLINK1 Status Register
0x10140620	XLINK2_XLINK_CFG	XLINK2 Configuration Register
0x10140624	XLINK2_XLINK_TX	XLINK2 Transmit Register
0x10140628	XLINK2_XLINK_RX	XLINK2 RX Register
0x1014062c	XLINK2_XLINK_STAT	XLINK2 Status Register
0x10140630	XLINK3_XLINK_CFG	XLINK3 Configuration Register
0x10140634	XLINK3_XLINK_TX	XLINK3 Transmit Register
0x10140638	XLINK3_XLINK_RX	XLINK3 RX Register
0x1014063c	XLINK3_XLINK_STAT	XLINK3 Status Register
0x10140640	IO_LINK_IRQ_RAW	IO-Link Raw interrupts
0x10140644	IO_LINK_IRQ_MASKED	IO-Link Masked IRQ Register
0x10140648	IO_LINK_IRQ_MSK_SET	IO-Link Interrupt Mask Enable
0x1014064c	IO_LINK_IRQ_MSK_RESET	IO-Link Interrupt Mask Disable
0x10140650	IO_LINK_IRQ_ENABLE	IO-Link Processor Enable
0x101c0b00	UART0_DATA	UART0 Data Register
0x101c0b04	UART0_STAT	UART0 Status Register
0x101c0b08	UART0_LINE_CTRL	UART0 Line Control Register
0x101c0b0c	UART0_BAUD_DIV_MSB	UART0 Baud Rate Divisor MSB
0x101c0b10	UART0_BAUD_DIV_LSB	UART0 Baud Rate Divisor LSB
0x101c0b14	UART0_CTRL	UART0 Control Register
0x101c0b18	UART0_FLAG	UART0 Flag Register
0x101c0b1c	UART0_INT_ID	UART0 Interrupt Identification Register
0x101c0b20	UART0_IRDA_LO_PWR_CNTR	UART0 IrDA Low Power Counter Register
0x101c0b24	UART0_RTS_CTRL	UART0 RTS Control Register
0x101c0b28	UART0_RTS_LEAD_CYC	UART0 RTS Leading Cycles
0x101c0b2c	UART0_RTS_TRAIL_CYC	UART0 RTS Trailing Cycles
0x101c0b30	UART0_OUT_DRV_EN	UART0 UART Output Driver Enable Register
0x101c0b34	UART0_BAUD_MODE_CTRL	UART0 Baud Rate Mode Control Register
0x101c0b38	UART0_RX_FIFO_IRQ_LVL	UART0 RX FIFO Trigger Level and RX-DMA Enable
0x101c0b3c	UART0_TX_FIFO_IRQ_LVL	UART0 TX FIFO Trigger Level and TX-DMA Enable
0x101c0b40	UART1_DATA	UART1 Data Register
0x101c0b44	UART1_STAT	UART1 Status Register
0x101c0b48	UART1_LINE_CTRL	UART1 Line Control Register
0x101c0b4c	UART1_BAUD_DIV_MSB	UART1 Baud Rate Divisor MSB
0x101c0b50	UART1_BAUD_DIV_LSB	UART1 Baud Rate Divisor LSB
0x101c0b54	UART1_CTRL	UART1 Control Register

ARM Address	Register Name	Short Description
0x101c0b58	UART1_FLAG	UART1 Flag Register
0x101c0b5c	UART1_INT_ID	UART1 Interrupt Identification Register
0x101c0b60	UART1_IRDA_LO_PWR_CNTR	UART1 IrDA Low Power Counter Register
0x101c0b64	UART1_RTS_CTRL	UART1 RTS Control Register
0x101c0b68	UART1_RTS_LEAD_CYC	UART1 RTS Leading Cycles
0x101c0b6c	UART1_RTS_TRAIL_CYC	UART1 RTS Trailing Cycles
0x101c0b70	UART1_OUT_DRV_EN	UART1 UART Output Driver Enable Register
0x101c0b74	UART1_BAUD_MODE_CTRL	UART1 Baud Rate Mode Control Register
0x101c0b78	UART1_RX_FIFO_IRQ_LVL	UART1 RX FIFO Trigger Level and RX-DMA Enable
0x101c0b7c	UART1_TX_FIFO_IRQ_LVL	UART1 TX FIFO Trigger Level and TX-DMA Enable
0x10140680	SPI_CTRL0	SPI Control Register 0
0x10140684	SPI_CTRL1	SPI Control Register 1
0x10140688	SPI_DATA	SPI Data Register
0x1014068c	SPI_STAT	SPI Status Register
0x10140690	SPI_CLK_PRE_SCL	SPI Clock Prescale Register
0x10140694	SPI_INT_MSK_SET_CLR	SPI Interrupt Mask Set or Clear Register
0x10140698	SPI_RAW_INT_STAT	SPI RAW Interrupt Status Register
0x1014069c	SPI_MASK_INT_STAT	SPI Masked Interrupt Status Register
0x101406a0	SPI_INT_CLR	SPI Interrupt Clear Register
0x101406a4	SPI_IRQ_CPU_SEL	Interrupt CPU Select Register
0x101406a8	SPI_DMA_CTRL	SPI DMA Control Register
0x101406b0	SPI_LGY_DATA	SPI Legacy Data Register
0x101406b4	SPI_LGY_STAT	SPI Legacy Status Register
0x101406b8	SPI_LGY_CTRL	SPI Legacy Control Register
0x101406bc	SPI_LGY_INT_CTRL	SPI Legacy Interrupt Control Register
0x101c0d00	SQI_CTRL0	SQI Control Register 0
0x101c0d04	SQI_CTRL1	SQI Control Register 1
0x101c0d08	SQI_DATA	SQI Data Register
0x101c0d0c	SQI_STAT	Read Only SQI Status Register
0x101c0d10	SQI_TCR	SQI Transfer Control
0x101c0d14	SQI_IRQ_MASK	SQI Interrupt Mask Set or Clear Register
0x101c0d18	SQI_IRQ_RAW	SQI Interrupt State Before Masking Register (Raw Interrupt)
0x101c0d1c	SQI_IRQ_MASKED	SQI Masked Interrupt Status Register
0x101c0d20	SQI_IRQ_CLEAR	SQI Interrupt Clear Register (For Compatibility To NetX10/50 SPI Module)
0x101c0d24	SQI_IRQ_CPU_SEL	SQI Interrupt CPU Select Register
0x101c0d28	SQI_DMACR	SQI DMA Control Register
0x101c0d30	SQI_PIO_OUT	SQI PIO Output Level Control Register
0x101c0d34	SQI_PIO_OE	SQI PIO Output Enable Control Register
0x101c0d38	SQI_PIO_IN	SQI PIO Input Status Register
0x101c0d3c	SQI_SQIROM_CFG	SQIROM Mode Configuration
0x101c0d40	I2C_MASTER_CTRL	I2C Master Control Register
0x101c0d44	I2C_SLAVE_CTRL	I2C Slave Control Register
0x101c0d48	I2C_MASTER_CMD	I2C Master Command Register
0x101c0d4c	I2C_MASTER_DATA	I2C Master Data Register
0x101c0d50	I2C_SLAVE_DATA	I2C Slave Data Register
0x101c0d54	I2C_MASTER_FIFO_CTRL	I2C Master FIFO Control Register
0x101c0d58	I2C_SLAVE_FIFO_CTRL	I2C Slave FIFO Control Register
0x101c0d5c	I2C_STAT	I2C Status Register

ARM Address	Register Name	Short Description
0x101c0d60	I2C_INT_MSK_SET_CLR	I2C Interrupt Mask Set or Clear Register
0x101c0d64	I2C_RAW_INT_STAT	I2C RAW Interrupt Status Register
0x101c0d68	I2C_MSK_INT_STAT	I2C Mask Interrupt Status Register
0x101c0d6c	I2C_DMA_CTRL	I2C DMA Control Register
0x101c0d70	I2C_PIO	Direct I2C IO Access Controlling
0x101c1000	SYSTIME_S	Upper SYSTIME Register
0x101c1004	SYSTIME_NS	Lower SYSTIME Register
0x101c1008	SYS_TIME_NS_BOR	SYSTIME Border Register
0x101c100c	SYS_TIME_NS_ADD_UP	SYSTIME Count Register
0x101c0e00	USB_DEV_CFG	USB Device Configuration Register
0x101c0e04	USB_DEV_STATUS	USB Device Status Register
0x101c0e08	USB_DEV_VENDOR_FEATURES	USB Vendor Feature Status Register
0x101c0e0c	USB_DEV_IRQ_MASK	USB Device Interrupt Mask Register
0x101c0e10	USB_DEV_IRQ_RAW	USB Device Raw Interrupt Status Register
0x101c0e14	USB_DEV_IRQ_MASKED	USB Device Masked Interrupt Status Register
0x101c0e40	USB_DEV_ENUM_RAM_DESCRIPTORS_BASE	USB Device Descriptor Start
0x101c0e44	USB_DEV_ENUM_RAM_DESCRIPTORS_END	USB Device Descriptor End
0x101c0e48	USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_BASE	USB String Descriptor Start
0x101c0e7c	USB_DEV_ENUM_RAM_STRING_DESCRIPTORS_END	USB String Descriptor End
0x101c0e80	USB_DEV_FIFO_CTRL_CONF	USB Device FIFO Configuration Register
0x101c0e84	USB_DEV_FIFO_CTRL_OUT_HANDSHAKE	USB Device FIFO Out Handshake
0x101c0e88	USB_DEV_FIFO_CTRL_IN_HANDSHAKE	USB Device FIFO In Handshake
0x101c0e8c	USB_DEV_FIFO_CTRL_STATUS0	USB Device FIFO 0 Status Register
0x101c0e90	USB_DEV_FIFO_CTRL_STATUS1	USB Device FIFO 1 Status Register
0x101c0e94	USB_DEV_FIFO_CTRL_STATUS2	USB Device FIFO 2 Status Register
0x101c0e98	USB_DEV_FIFO_CTRL_STATUS3	USB Device FIFO 3 Status Register
0x101c0e9c	USB_DEV_FIFO_CTRL_STATUS4	USB Device FIFO 4 Status Register
0x101c0ea0	USB_DEV_FIFO_CTRL_STATUS5	USB Device FIFO 5 Status Register
0x101c0ea4	USB_DEV_FIFO_CTRL_STATUS6	USB Device FIFO 6 Status Register
0x101c0ec0	USB_DEV_FIFO0	USB Device FIFO: Control Endpoint OUT
0x101c0ec4	USB_DEV_FIFO1	USB Device FIFO: Control Endpoint IN
0x101c0ec8	USB_DEV_FIFO2	USB Device FIFO: Endpoint 1 - JTAG TX
0x101c0ecc	USB_DEV_FIFO3	USB Device FIFO: Endpoint 2 - JTAG RX
0x101c0ed0	USB_DEV_FIFO4	USB Device FIFO: Endpoint 3 - UART TX
0x101c0ed4	USB_DEV_FIFO5	USB Device FIFO: Endpoint 4 - UART RX
0x101c0ed8	USB_DEV_FIFO6	USB Device FIFO: Endpoint 5 - Interrupt IN
0x101ff000	VIC_IRQ_STAT	IRQ Status Register
0x101ff004	VIC_FIQ_STAT	FIQ Status Register
0x101ff008	VIC_RAW_INT_STAT	Raw Interrupt Status Register
0x101ff00c	VIC_INT_SEL	Interrupt Select Register
0x101ff010	VIC_INT_EN	Interrupt Enable Register
0x101ff014	VIC_INT_EN_CLR	Interrupt Enable Clear Register
0x101ff018	VIC_SWI	Software Interrupt Register
0x101ff01c	VIC_SWI_CLR	Software Interrupt Clear Register
0x101ff020	VIC_PROT_EN	Protection Enable Register
0x101ff030	VIC_VECT_ADDR	Vector Address Register
0x101ff034	VIC_DFLT_VECT_ADDR	Default Vector Address Register

ARM Address	Register Name	Short Description
0x101ff100	VIC_VECT_ADDR0	Vector Address Register 0
0x101ff104	VIC_VECT_ADDR1	Vector Address Register 1
0x101ff108	VIC_VECT_ADDR2	Vector Address Register 2
0x101ff10c	VIC_VECT_ADDR3	Vector Address Register 3
0x101ff110	VIC_VECT_ADDR4	Vector Address Register 4
0x101ff114	VIC_VECT_ADDR5	Vector Address Register 5
0x101ff118	VIC_VECT_ADDR6	Vector Address Register 6
0x101ff11c	VIC_VECT_ADDR7	Vector Address Register 7
0x101ff120	VIC_VECT_ADDR8	Vector Address Register 8
0x101ff124	VIC_VECT_ADDR9	Vector Address Register 9
0x101ff128	VIC_VECT_ADDR10	Vector Address Register 10
0x101ff12c	VIC_VECT_ADDR11	Vector Address Register 11
0x101ff130	VIC_VECT_ADDR12	Vector Address Register 12
0x101ff134	VIC_VECT_ADDR13	Vector Address Register 13
0x101ff138	VIC_VECT_ADDR14	Vector Address Register 14
0x101ff13c	VIC_VECT_ADDR15	Vector Address Register 15
0x101ff200	VIC_VECT_CTRL0	Vector Control Register 0
0x101ff204	VIC_VECT_CTRL1	Vector Control Register 1
0x101ff208	VIC_VECT_CTRL2	Vector Control Register 2
0x101ff20c	VIC_VECT_CTRL3	Vector Control Register 3
0x101ff210	VIC_VECT_CTRL4	Vector Control Register 4
0x101ff214	VIC_VECT_CTRL5	Vector Control Register 5
0x101ff218	VIC_VECT_CTRL6	Vector Control Register 6
0x101ff21c	VIC_VECT_CTRL7	Vector Control Register 7
0x101ff220	VIC_VECT_CTRL8	Vector Control Register 8
0x101ff224	VIC_VECT_CTRL9	Vector Control Register 9
0x101ff228	VIC_VECT_CTRL10	Vector Control Register 10
0x101ff22c	VIC_VECT_CTRL11	Vector Control Register 11
0x101ff230	VIC_VECT_CTRL12	Vector Control Register 12
0x101ff234	VIC_VECT_CTRL13	Vector Control Register 13
0x101ff238	VIC_VECT_CTRL14	Vector Control Register 14
0x101ff23c	VIC_VECT_CTRL15	Vector Control Register 15
0x101c0010	PHY_CTRL	PHY Control Register
0x101c0c00	MIIMU_RXTX	MIIMU Receive/Transmit Register
0x101c0c04	MIIMU_MODE_EN	MIIMU Software Mode Enable
0x101c0c08	MIIMU_MODE_MDC	MIIMU Software Mode MDC Register
0x101c0c0c	MIIMU_MODE_MDO	MIIMU Software Mode MDO Register
0x101c0c10	MIIMU_MODE_MDOE	MIIMU Software Mode MDOE Register
0x101c0c14	MIIMU_MODE_MDI	MIIMU Software Mode MDI Register
0x101a4000	PTR_FIFO_BASE	Pointer FIFO Table
0x101a4040	PTR_FIFO_BOR_BASE	Pointer FIFO Upper Borders table
0x101a4080	PTR_FIFO_RESET	Pointer FIFO Reset Vector
0x101a4084	PTR_FIFO_FULL	Pointer FIFO Full Vector
0x101a4088	PTR_FIFO_EMPTY	Pointer FIFO Empty Vector
0x101a408c	PTR_FIFO_OVF	Pointer FIFO Overflow Vector
0x101a4090	PTR_FIFO_UDR	Pointer FIFO Underrun Vector
0x101a40c0	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Fill-Level table
0x10124000	PTR_FIFO_BASE	Pointer FIFO Motion table

ARM Address	Register Name	Short Description
0x10124040	PTR_FIFO_BOR_BASE	Pointer FIFO Motion Upper Borders table
0x10124080	PTR_FIFO_RESET	Pointer FIFO Motion Reset Vector
0x10124084	PTR_FIFO_FULL	Pointer FIFO Motion Full Vector
0x10124088	PTR_FIFO_EMPTY	Pointer FIFO Motion Empty Vector
0x1012408c	PTR_FIFO_OVF	Pointer FIFO Motion Overflow Vector
0x10124090	PTR_FIFO_UDR	Pointer FIFO Motion Underrun Vector
0x101240c0	PTR_FIFO_FILL_LVL_BASE	Pointer FIFO Motion Fill-Level table
0x101a5600	BUF_MAN_BMU	BUF_MAN BMU
0x10125600	BUF_MAN_MOTION_BMU	BUF_MAN_MOTION BMU
0x101a4400	IRQ_XP0	IRQs between XPEC0 and ARM
0x101c5100	DMAC_CH0_SRC_ADDR	Channel0 Source Address Registers
0x101c5104	DMAC_CH0_DEST_ADDR	Channel0 Destination Address Registers
0x101c5108	DMAC_CH0_LINK	Channel0 Linked List Item Register
0x101c510c	DMAC_CH0_CTRL	Channel0 Control Registers
0x101c5110	DMAC_CH0_CFG	Channel0 Configuration Registers
0x101c5120	DMAC_CH1_SRC_ADDR	Channel1 Source Address Registers
0x101c5124	DMAC_CH1_DEST_ADDR	Channel1 Destination Address Registers
0x101c5128	DMAC_CH1_LINK	Channel1 Linked List Item Register
0x101c512c	DMAC_CH1_CTRL	Channel1 Control Registers
0x101c5130	DMAC_CH1_CFG	Channel1 Configuration Registers
0x101c5140	DMAC_CH2_SRC_ADDR	Channel2 Source Address Registers
0x101c5144	DMAC_CH2_DEST_ADDR	Channel2 Destination Address Registers
0x101c5148	DMAC_CH2_LINK	Channel2 Linked List Item Register
0x101c514c	DMAC_CH2_CTRL	Channel2 Control Registers
0x101c5150	DMAC_CH2_CFG	Channel2 Configuration Registers
0x101c5800	DMAC_INT_STAT	Interrupt Status Register
0x101c5804	DMAC_INT_TC_STAT	Interrupt Terminal Count Status Register
0x101c5808	DMAC_INT_TC_CLR	Interrupt Terminal Count Clear Register
0x101c580c	DMAC_INT_ERR_STAT	Interrupt Error Status Register
0x101c5810	DMAC_INT_ERR_CLR	Interrupt Error Clear Register
0x101c5814	DMAC_RAW_INT_TC_STAT	Raw Interrupt Terminal Count Status Register
0x101c5818	DMAC_RAW_INT_ERR_STAT	Raw Interrupt Error Status Register
0x101c581c	DMAC_CH_EN	Channel Enable Register
0x101c5820	DMAC_SW_BURST_REQ	Software Burst Request Register
0x101c5824	DMAC_SW_SINGLE_REQ	Software Single Request Register
0x101c5828	DMAC_SW_LAST_BURST_REQ	Software Last Burst Request Register
0x101c582c	DMAC_SW_LAST_SINGLE_REQ	Software Last Single Request Register
0x101c5830	DMAC_CFG	Configuration Register
0x101c5834	DMAC_SYNC	Sync Register

13.2 Contacts

Headquarters

Germany
Hilscher Gesellschaft für
Systemautomation mbH
Rheinstrasse 15
65795 Hattersheim
Phone: +49 (0) 6190 9907-0
Fax: +49 (0) 6190 9907-50
E-Mail: info@hilscher.com

Support
Phone: +49 (0) 6190 9907-99
E-Mail: de.support@hilscher.com

Subsidiaries

China
Hilscher Systemautomation (Shanghai) Co. Ltd.
200010 Shanghai
Phone: +86 (0) 21-6355-5161
E-Mail: info@hilscher.cn

Support
Phone: +86 (0) 21-6355-5161
E-Mail: cn.support@hilscher.com

France
Hilscher France S.a.r.l.
69500 Bron
Phone: +33 (0) 4 72 37 98 40
E-Mail: info@hilscher.fr

Support
Phone: +33 (0) 4 72 37 98 40
E-Mail: fr.support@hilscher.com

India
Hilscher India Pvt. Ltd.
New Delhi - 110 025
Phone: +91 11 40515640
E-Mail: info@hilscher.in

Italy
Hilscher Italia srl
20090 Vimodrone (MI)
Phone: +39 02 25007068
E-Mail: info@hilscher.it

Support
Phone: +39 02 25007068
E-Mail: it.support@hilscher.com

Japan
Hilscher Japan KK
Tokyo, 160-0022
Phone: +81 (0) 3-5362-0521
E-Mail: info@hilscher.jp

Support
Phone: +81 (0) 3-5362-0521
E-Mail: jp.support@hilscher.com

Korea
Hilscher Korea Inc.
Suwon, 443-734
Phone: +82 (0) 31-695-5515
E-Mail: info@hilscher.kr

Switzerland
Hilscher Swiss GmbH
4500 Solothurn
Phone: +41 (0) 32 623 6633
E-Mail: info@hilscher.ch

Support
Phone: +49 (0) 6190 9907-99
E-Mail: ch.support@hilscher.com

USA
Hilscher North America, Inc.
Lisle, IL 60532
Phone: +1 630-505-5301
E-Mail: info@hilscher.us

Support
Phone: +1 630-505-5301
E-Mail: us.support@hilscher.com

