



Technical Data Reference Guide
netX 500/100

next Generation of Communication Controllers

Hilscher Gesellschaft für Systemautomation mbH
www.hilscher.com

DOC061102TRG18EN | Revision 1.8 | English | 2017-08 | Released | Public

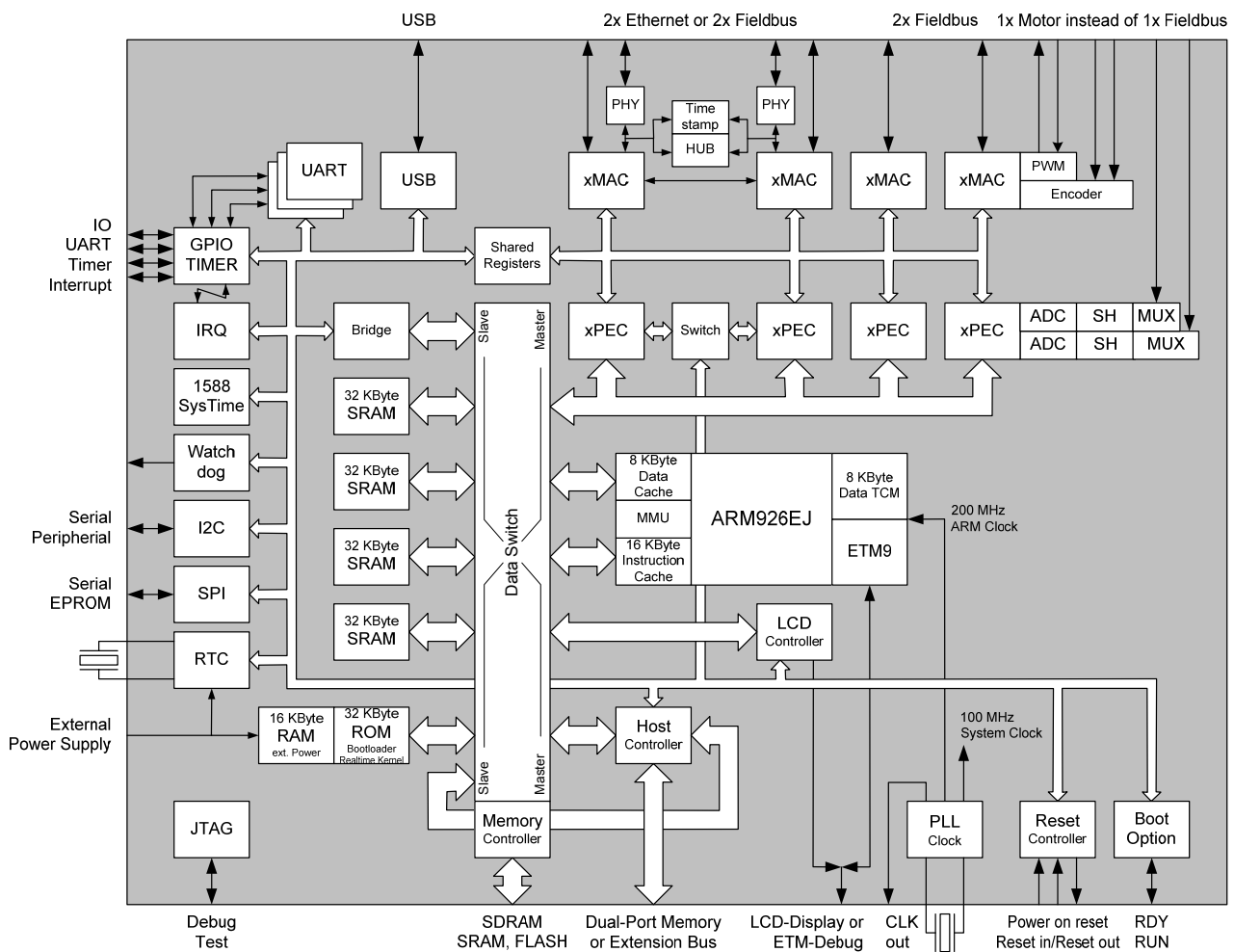
Table of Contents

1	INTRODUCTION	5
1.1	Product Features	7
1.2	Differences netX 500 – netX 100	8
1.3	Typical Applications	8
1.4	References to Documents	9
2	FUNCTIONAL OVERVIEW	10
2.1	CPU	10
2.2	Oscillator	10
2.3	System LED and Boot Modes	11
2.4	Extended System Information	13
2.5	Reset	14
2.6	Reset Configuration	15
2.7	Watchdog	16
2.7.1	WDGACT Signal	16
2.8	Internal Memory	17
2.9	External Memory	18
2.9.1	SRAM / FLASH Interface	18
2.9.2	SDRAM	29
2.10	Extension Bus	34
2.10.1	Extension Bus Configuration	34
2.10.2	Extension Bus Address Space and netX Memory Allocation	34
2.10.3	Address and Data Byte Steering	35
2.10.4	Intel / Motorola Data Format	35
2.10.5	Multiplexed / Non-Multiplexed Data Bus	35
2.10.6	Data Ready or Data Acknowledge	35
2.10.7	End-Of-Cycle	36
2.10.8	Pin Description of Extension Bus	36
2.10.9	Extension Bus Component Connection	38
2.10.10	Extension Bus Timing without Wait-states	40
2.10.11	Extension Bus Timing with Waitstates	41
2.11	Dual-Port memory	42
2.11.1	Dual-Port Memory Interface Mode	42
2.11.2	Dual-Port Memory Structure and Allocation	43
2.11.3	Global Control Block	44
2.11.4	DPM interface signals	50
2.11.5	Interrupts and Interrupt Signal	51
2.11.6	Dual-Port Memory Timing	56
2.12	Timer	58
2.13	Real-time clock and Backup RAM	59
2.14	IEEE 1588 System Time	61
2.15	JTAG Debug Interface	62
2.15.1	Standard JTAG connector	62
2.15.2	Hilscher “mini-JTAG” Connector	63
2.15.3	Boundary Scan mode	64
2.16	Embedded Trace Macrocell ETM	65
2.17	Vectored Interrupt Controller	66
2.17.1	Interrupt generation	68
2.17.2	Interrupt priority logic	69
2.17.3	Interrupt flow sequence	69
2.18	UART	70
2.19	USB	73
2.20	I2C	74

2.21	SPI	74
2.22	GPIO	75
2.23	PIO	76
2.24	LCD	79
2.25	Motion Control Functions	80
2.25.1	AD-Converter	80
2.25.2	Motor PWM	81
2.25.3	Resolver PWM	83
2.25.4	Encoder interface	83
2.26	Ethernet Interface	84
2.26.1	Real Time Ethernet	86
2.27	Fieldbus Interface	88
2.27.1	AS interface Master	89
2.27.2	CANopen Interface	90
2.27.3	CC-Link Interface	91
2.27.4	DeviceNet Interface	92
2.27.5	PROFIBUS Interface	93
2.28	XMAC Resource Sharing	94
3	ETHERNET COMMUNICATION STRUCTURE	95
3.1	Ethernet data transfer	96
3.2	Error Counters	97
4	INTERNAL ROM	98
4.1	Boot Function	98
4.1.1	The 1 st Stage Bootstrap	98
4.1.2	The 2 nd Stage Bootstrap	98
4.1.3	The Application Code	98
4.1.4	The general Bootstrap Sequence.	99
4.1.5	The 1 st Stage Bootstrap Options	100
4.1.6	The 1 st Stage Boot Option Selection	101
4.1.7	The 1 st Stage Boot Option detection	101
4.2	High Performance Real-Time Operating System rcX	103
4.2.1	Pre-emptive Kernel	104
4.2.2	Small Footprint	104
4.2.3	Modularity	104
4.2.4	Dynamic Objects	104
4.2.5	Rich API Functions	104
4.2.6	Tool Support	104
5	ELECTRICAL SPECIFICATIONS	105
5.1	Absolute Maximum Ratings	105
5.2	Power Up Sequencing	106
5.3	Power Consumption / Power Dissipation	107
5.4	AC / DC Specifications	109
5.4.1	DC Parameters	109
5.4.2	System Oscillator	113
5.4.3	RTC Oscillator	114
5.4.4	Power On Reset / Reset Input	115
5.4.5	USB	116
5.4.6	ADC	117
5.4.7	PHY	118
5.4.8	SDRAM	119
5.4.9	SRAM / FLASH	125
5.4.10	SPI	127
5.4.11	I ² C	128
5.4.12	UART	129
5.4.13	Dual-port memory	130
5.4.14	Extension bus	132
5.4.15	LCD	134

5.4.16	JTAG	135
5.4.17	Fieldbus / PWM / Encoder	136
5.5	Failure Rate (FIT)	137
6	PACKAGE INFORMATION	138
6.1	Package Thermal Specification	138
6.2	Soldering Conditions	139
6.2.1	Infrared Reflow Soldering Characterization	139
6.2.2	Vapour Phase Reflow Soldering (VPS) Characterization	140
6.3	General storage conditions	140
6.4	Signal Definitions	141
6.4.1	Schematic View of netX Pad Types	147
6.5	Pin Table Sorted By Pin Numbers	148
6.6	Pin Table Sorted By Signals	150
6.7	Pin Overview	152
6.8	Device Marking	153
6.8.1	netX 100	153
6.8.2	netX 500	153
6.9	Mechanical Dimensions / Physical Dimensions	154
6.10	Material composition	155
6.10.1	Solder balls	155
6.11	Ordering Information	155
6.12	Packing	156
6.13	Moisture Sensitivity Level	157
7	PRINTED CIRCUIT BOARD DESIGN	158
7.1	netX Trace Wiring	159
7.2	V _{cc} Pin Requirements / Decoupling Capacitors	160
7.3	Reference PCB Layout Design	161
8	REFERENCE SCHEMATICS	162
8.1	Bill of Materials of Reference Design	179
9	REVISION HISTORY	180
10	CONTACTS	181

1 Introduction



netX 500 Block diagram

The netX 500 is a highly integrated network controller with a new system architecture optimized for communication and maximum data throughput.

Based on the 32-Bit CPU ARM 926EJ-S clocked with 200 MHz, it provides a memory management unit, 16K instruction- and 8K data caches, 8K tightly coupled data memory and DSP- and Java extensions. Together with 32 Kbytes ROM holding the boot loader and a Real-Time kernel, the internal memory of 144 Kbytes RAM is sufficient for smaller applications, while 16K of the internal RAM can be buffered by a separate (battery-) power supply for storing non volatile data.

The connection to a primary host is accomplished by the dual-port memory mode of the host interface, which is also configurable as a 16 Bit extension bus for stand-alone applications. Comprehensive peripheral functions and serial interfaces such as UARTs, USB, SPI and I²C provide a wide spectrum of applications. Yet, it is the central data switch and the four freely configurable communication channels with their own intelligence that are the main characteristics of the netX as a "high end" network controller.

The data switch interconnects the ARM CPU, communication and Host controllers, memory blocks and peripheral units via five data paths. This allows the controllers to transmit their data in parallel, contrary to the traditional sequential architecture with only one common data bus and additional bus arbitration cycles. To allow efficient use of the data switch, the internal SRAM is divided in five separate blocks (4 * 32K and 1* 16K) that can be accessed individually (for instance, the ARM CPU can access one block of the internal SRAM, while, at the same time, the Host Controller and an xPEC can read from or write to the other three memory blocks).

All peripheral functions such as USB, UART, Timer and Real-Time Clock are connected via an internal peripheral bus to a slave port of the data switch. The LCD-Controller and Host Controller are linked to two additional master ports. The Memory Controller with its SDRAM, SRAM and FLASH interface is a separate part of the data switch providing very close coupling.

The identical controllers of the four communication channels are structured on two levels. They consist of dedicated ALUs and special logic units that receive their protocol functions via Microcode, combining the performance of dedicated (single-) protocol controllers with the flexibility of a CPU, allowing to perform all kind of hard Real-Time operations already on the communication channel level, without interaction of the main CPU (ARM). Two of these channels can further be linked to integrated PHYs for Ethernet communication.

The Medium-Access-Controllers (xMAC) send or receive the serial data streams according to the respective bus access process and encrypt or convert them from Bit- to Byte streams.

The Protocol Execution Controllers (xPEC) compile the bytes delivered by the xMACs into data packets and control the telegram traffic. The data packets are transferred into the internal memory in blocks, using DMA and a Round Robin process. In addition, every channel has a Dual-port-memory available for status information or as local data image.

By its intelligent communication ALUs, the netX can implement the most varied protocols and protocol combinations and can synchronize them independently of the CPU response time – an absolutely new feature in industrial communication technology.

For software development, the netX provides a JTAG interface for standard debuggers, as well as an ETM (Embedded Trace Macrocell) for sophisticated high-end development tools.

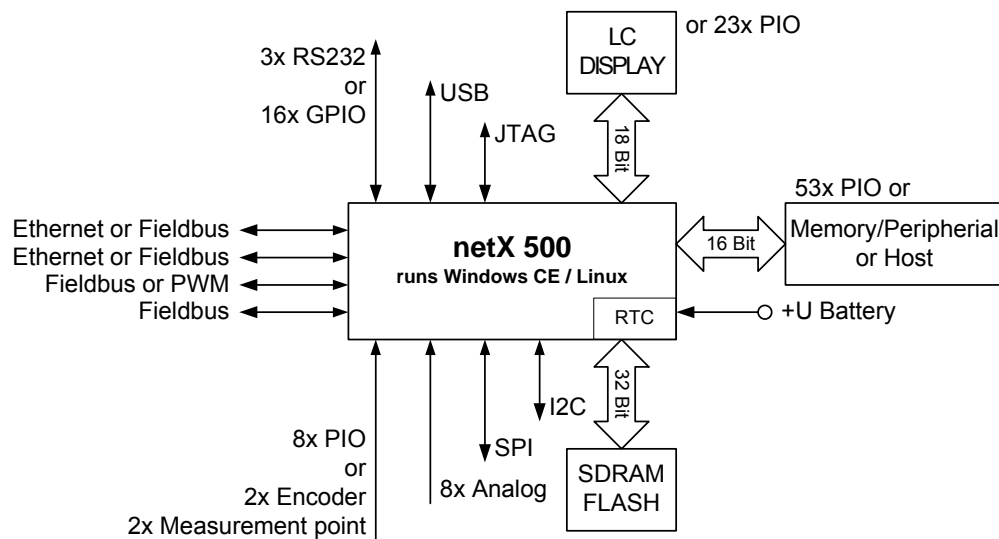
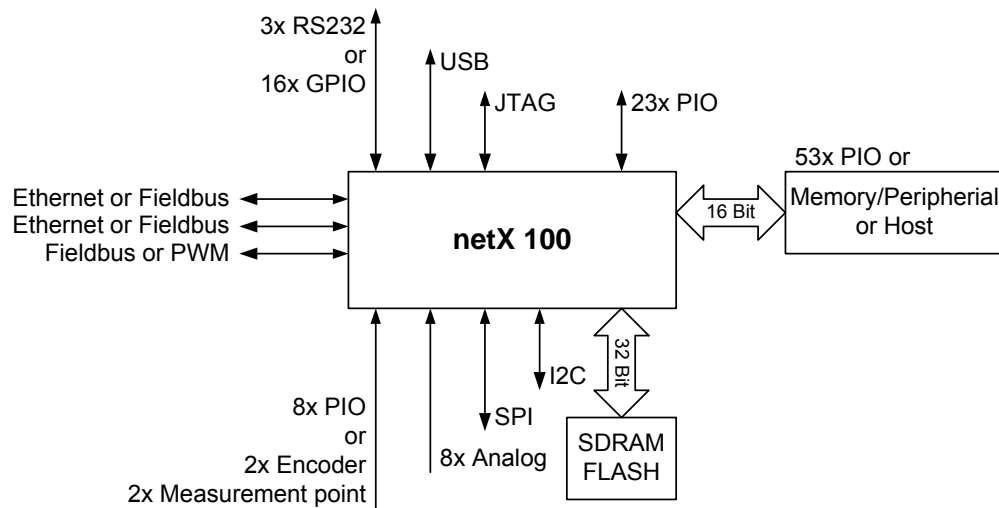
1.1 Product Features

- Powerful 200 MIPS ARM 926EJ-S CPU
- ARMv5TEJ technology with Jazelle® hardware and enhanced DSP capability
- Memory Management Unit supporting Windows CE and Linux
- Internal data switch between master and slave data resources to avoid bottle necks
- 2 10/100 MBit/s Ethernet Channels with integrated PHYs
- Time stamping and synchronization according to IEEE 1588
- Special Hardware Support of Real-time Ethernet Features
- Support for Real-time Ethernet Protocols EtherCAT, Ethernet/IP, Powerlink, PROFINET, SERCOS-III
- Fieldbus Controller for AS interface, CAN, CC-Link, PROFIBUS master and slave
- 3 phase Motor PWM, 1 phase Resolver PWM
- 2 Encoder interfaces
- 2 Measurement point inputs
- 2 10Bit A/D-converters, each with sample/hold and 4 channel input multiplexer
- 128 KByte internal RAM for small applications without external memory components
- 16 KByte RAM with separate power supply to hold non volatile data
- 32 KByte Boot ROM
- SDRAM controller for large memory
- SRAM and FLASH interface without glue logic
- Dual port memory interface for easy interface to host controller
- Extension bus to add peripherals for stand alone applications
- LCD Controller for DSTN or TFT Displays up to 640 * 480 (netX 500 only)
- USB interface, runs as host or device, eight pipes
- 3 UARTs, 16550 compatible
- SPI with separate input and output-fifos, fully interrupt driven
- I2C Interface
- Real-time clock (netX 500 only)
- Watchdog with active signal for safety purposes
- JTAG Debug Interface
- Boundary Scan
- ARM Embedded Trace Macrocell for real time tracing
- Small BGA package
- Extended temperature range
- Guaranteed 10 years life time
- Boot option via FLASH, serial EPROM, MMC, Dual-Port Memory or UART
- Real-time Kernel in Boot ROM
- Software support for protocol stacks
- Windows CE and Linux implementations available
- Debug and Development Environment support by HITEX

1.2 Differences netX 500 – netX 100

The netX 100 is a netX 500 derivate, with the following restrictions:

- LCD Controller not available
- RTC not available
- Communication channel 3 not available as fieldbus or PWM channel (can only be used for synch signals of RTE applications)



1.3 Typical Applications

- Communication Interfaces for PLC, Drives, and HMIs
- Standalone Application for low cost Drives, Machine Terminals, and I/O
- Target System for Windows CE or Linux based Application with Ethernet and Fieldbus Communication

1.4 References to Documents

- [1] Hilscher Gesellschaft für Systemautomation mbH: Programming Reference Guide, next Generation of Communication Controller (netX 100/500), Edition 4, English, 2007.
- [2] Hilscher Gesellschaft für Systemautomation mbH: Application Note, netX Design-In Guide netX 100/500, Revision 4, English, 2017.
- [3] Hilscher Gesellschaft für Systemautomation mbH: Errata, netX 100, English, 2011.
- [4] Hilscher Gesellschaft für Systemautomation mbH: Errata, netX 500, English, 2010.

2 Functional Overview

The following chapters provide an overview of the Key Function Units within the netX.

2.1 CPU

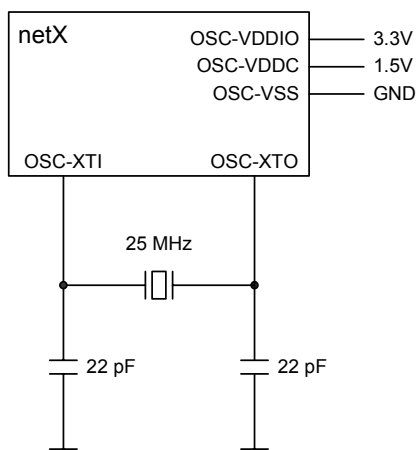
All controllers in the netX family are equipped with an ARM926EJ-S processor core based on the new ARMv5TEJ technology with Memory Management Unit, extended with enhanced Java and DSP performance.

The CPU runs on a 200 MHz system clock and has 16 KByte instruction and 8 KByte data caches. Additionally it provides 8 Kbytes of tightly coupled data memory with zero wait states.

2.2 Oscillator

All internal clock signals of the netX are generated by a PLL which is driven from an internal oscillator, requiring an external 25 MHz crystal.

Alternatively an external oscillator can be used. In this case the clock signal has to be connected to the pin OSC_XTI and OSC_XTO has to be disconnected.



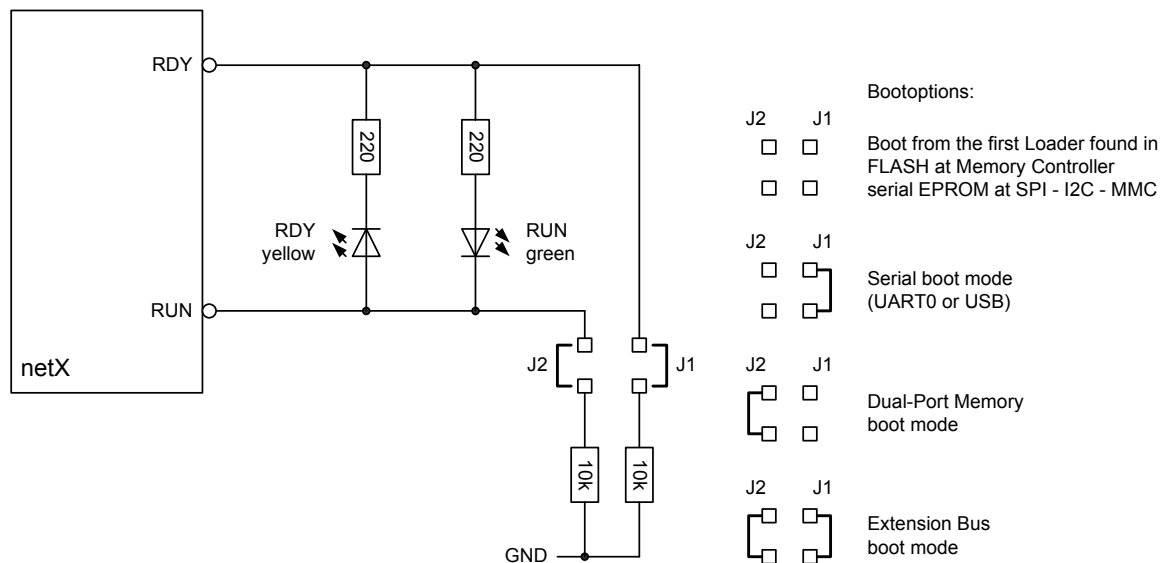
Oscillator schematic for the 25 MHz clock

2.3 System LED and Boot Modes

The general status of a netX based system is displayed by the System LED(s). It is recommended to use a dual LED as System LED, 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 behavior is hence completely application- or firmware specific.



System LED interface with settings for different Boot modes

The RDY and RUN signals are also used as inputs after a reset to select the boot mode. Applying certain logic levels to these pins after a reset, result in a pre-selection of the several available boot options and hence determine, where the ROM boot loader looks for executable program code. External 10 k pull-down resistors (in combination with the internal 50 k pull-up resistors) in can be used to define the input values.

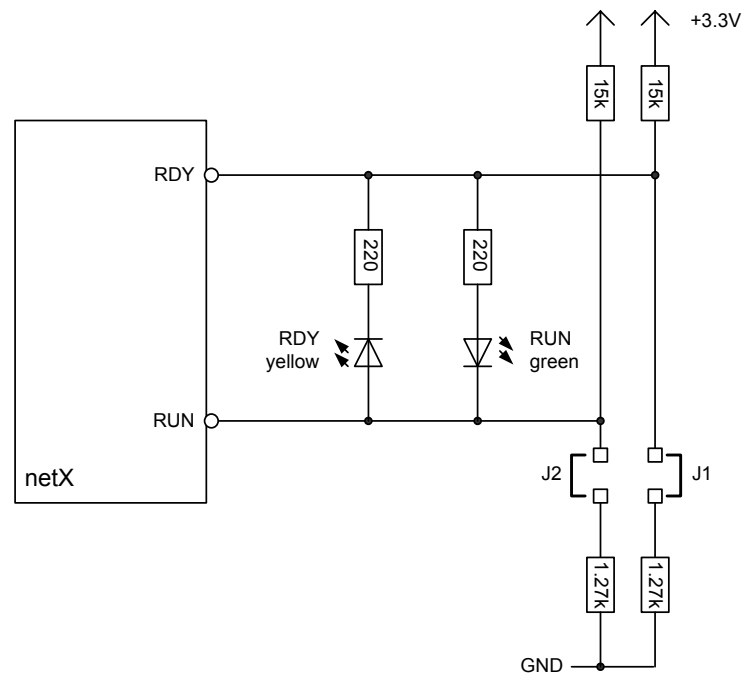
RDY / J1	RUN / J2	Boot Mode	Description
High / Open	High / Open	Memory	netX is looking for a special boot block within the connected memory in the following order: FLASH – SPI – I2C – MMC
Low / Closed	High / Open	Serial	netX waits for special command on UART0 or the USB port. This mode allows to bypass any bootable firmware stored in a boot memory, to perform firmware upgrades or re-flash erroneous firmware.
High / Open	Low / Closed	Dual-Port Memory	netX will initialize the Host Interface as standard 8-Bit Dual-Port Memory and wait for the host processor to download firmware.
Low / Closed	Low / Closed	Extension Bus	netX will initialize the Host Interface as Extension Bus and check memory connected to Chip Select 0 for bootable firmware.

(For further information on the different boot modes and the boot loader, see also chapter 4.11)

Notes:

The simple circuit shown on the previous page, already works fine on numerous netX boards. However, this circuit may result in voltage levels at the RDY and RUN pins, that are out of the specified logic levels (level on pins with jumper set may still be higher than 0.8V, while the level on the other pin (jumper not set) may not reach 2.0V). This is a result of the wide tolerance of the internal, nominal 50k pull-up resistors (min. 14.2k, typ. 31.9k, max. 80.7k). Due to the two LEDs (or one Dual LED) connected between the RDY and RUN pins, the issue is more complex than a simple voltage divider (a pull-down resistor on one pin also affects the (high) level on the other pin), hence tests with an equivalent circuit were made and revealed, that **connecting a 15k pull-up resistor to each pin, and using a value of 1.27k for pull-down resistor(s) will provide safe logic levels under all conditions**. These values assume, that as Dual LED, an HSMF-C156 (Agilent) is used. Since the LED has a considerable influence on the voltage levels, the resistor values should be re-approved when using a different LED.

The following schematic shows the improved circuit:



When using the serial boot mode over the USB port, while UART0 is unconnected, two **pull-up** resistors (10k or less) are required on the **UART0_RXD (AA19)** and the **UART0_CTS (AA16)** pins. When UART0 is connected to a RS-232 or RS-485 transceiver, these pull-ups are not required (any unconnected UART pin will be pulled low by its internal pull-down resistor, which lets the netX boot loader wrongly detect activity on the serial port, which again lets the netX remain in UART serial mode and never enter the USB serial mode).

2.4 Extended System Information

The two LEDs 'RDY' and 'RUN' described in the preceding chapter are controlled by the netX via a special system status register, containing information about the system status of the netX. The NETX_STA_CODE is set by the netX boot software or firmware. The definition of each status bit and status code is software specific. Also, there are some flags which can only be controlled by an external host system. When a write access to the status flags is performed by the netX, an interrupt request can be generated, to notify the host about the changes.

For a detailed description of the register bits see the 'netX 500/100 Program Reference Guide'.

Note:

When the netX chip is in reset state or has not yet been initialized completely, the host interface pins will be configured to their default functionality (input signal pins). The host interface pins of the netX 500/100 do not provide internal pull-up or pull-down resistors to ensure a valid signal level. External pull-up or pull-down resistors should be used to ensure a correct logic level, otherwise reading the extended System Information Register, while the netX host interface is not yet initialized, may result in invalid data values. Using external resistors enables the host system to read stable levels at all lower data lines (D7-0) which result in data value 0x00 (pull-down) or 0xff (pull-up), when the netX host interface does not yet respond. This value is defined as the netX reset value. When the netX has configured the host interface the read back value will no longer be 0x00 or 0xff but 0xf0. This indicates that the netX host interface is configured and all registers can be read.

2.5 Reset

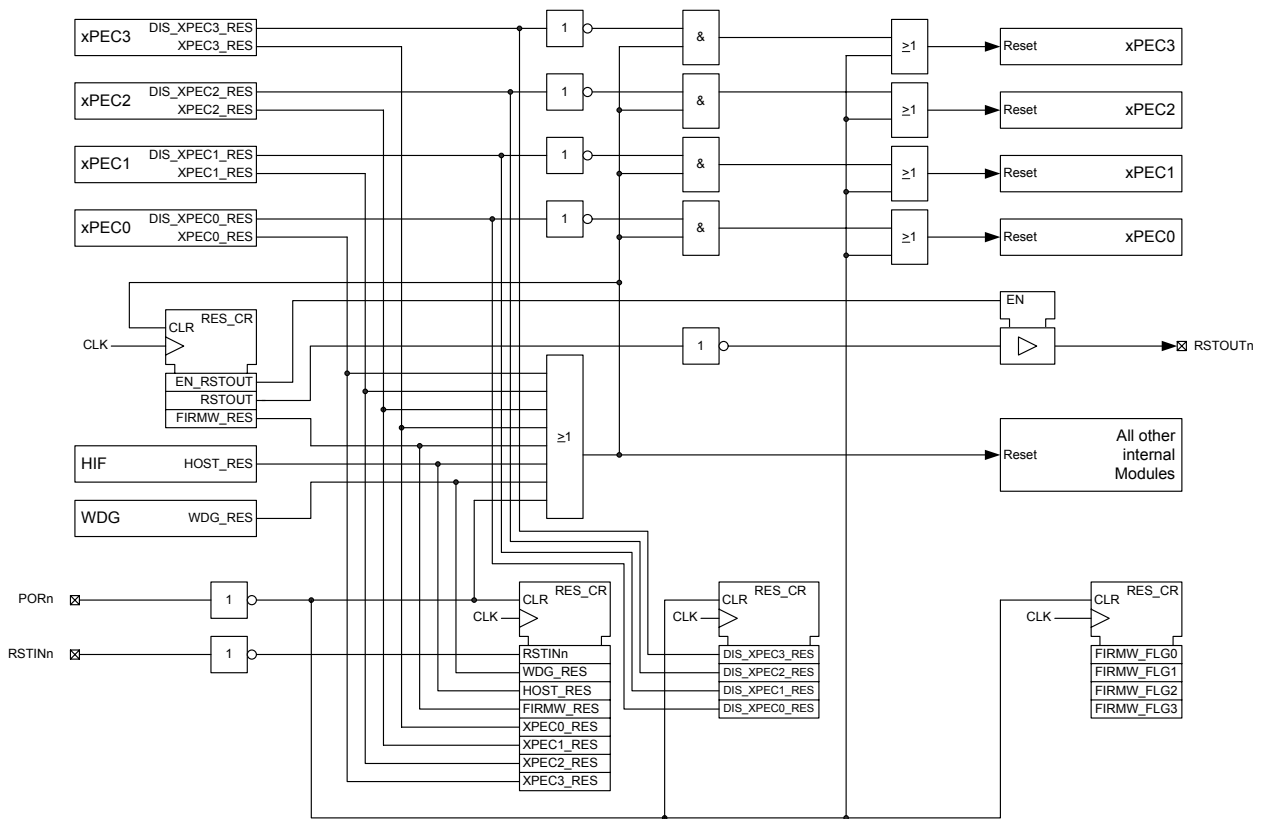
The netX offers nine reset sources which can generate a system reset of the chip. Two of them are external inputs, while the others are supplied by different internal function blocks.

PORn	<p>Power on reset, input pin</p> <p>This (active low) signal shall be connected to the output of a voltage supervisor chip, which checks the power supply voltages and pulls the power on reset pin low, whenever the voltages are below the minimum specified netX system operating voltages. The power on reset signal causes an asynchronous reset of the netX chip and initializes all internal registers and signals to their power on reset state. Reset timing is specified in chapter 5.4.4.</p>
RSTINn	<p>External reset input pin</p> <p>This is the second reset input signal (active low) from an external reset source e.g. from the host system. Unlike the PORn, this is a synchronous reset and is an optional signal. When not used, this signal has to be pulled or tied high (+3.3V) as this pin has no internal pull-up resistor.</p>
WDG_RES	<p>If the internal watchdog counter expires, this reset is generated. It is also possible first to generate an interrupt before resetting the chip. For more details see chapter 2.7 (netX system watchdog).</p>
HOST_RES	<p>This reset is initiated by the host system interface by writing a special sequence into a host interface register. The reset will occur 1 ms after starting the write cycle allowing the host to finish the access and prepare for the netX chip reset.</p>
FIRMW_RES	<p>This reset can be activated by a software command.</p>
XPEC0_RES	<p>This reset is generated by xPEC of communication channel 0.</p>
XPEC1_RES	<p>This reset is generated by xPEC of communication channel 1.</p>
XPEC2_RES	<p>This reset is generated by xPEC of communication channel 2.</p>
XPEC3_RES	<p>This reset is generated by xPEC of communication channel 3.</p>

Note:

When a netX system reset is performed, all host interface pins will enter their reset state (inputs). The host system has to provide pull-up or pull-down resistors on all host interface signal lines, where it requires a defined logic level during reset.

The following figure shows an overview of the netX reset circuit block.



Block diagram of the Reset Controller

There is also one output signal to reset external connected peripherals:

RSTOUTn This is an output signal to reset connected peripheral devices. It is a tristateable signal that can be enabled and set to high or low level by a software command. Any Reset, regardless if Power On, Reset In or internal reset, will disable the output driver of the signal, allowing to set the desired reset default level to low or high by either using a pull-down or pull-up resistor.

A power on reset is the only reset condition that will clear the value of the RESET_CTRL register. All other reset requests are stored in the RESET_CTRL register. This information can be used by the Firmware to determine which reset source has activated the last reset signal and act accordingly (e.g. not restart the system after a Watchdog initiated reset).

Additionally the RESET_CTRL register provides four bits that can be used to save information, unaffected by any reset, except the Power on reset.

The firmware can disable the internal system reset signals to the XPEC modules, allowing these modules to continue to run even while the chip is performing a reset.

2.6 Reset Configuration

With the rising edge of the last active power on reset signal, the levels of I/O pins PIO[84:32] are stored in the appropriate Hostinterface register. It is possible to read these values for hardware controlled power on configuration. For details see the register description of the host input / output port control and chapter 2.22.4 (Power-On-Reset Sampling of PIO 32-84 Pins) of this document

2.8 Internal Memory

The netX provides 144 KByte static RAM and 32 KByte of ROM, containing the Boot Loader and some RCX functions (see chapter 5, Internal ROM). The 144 KByte RAM comprises of four blocks of 32 KBytes each and one block with 16 KBytes. Each block has its own interface to the data switch. Therefore data transfer to and from the different blocks can be performed simultaneously.

The 16 KByte block can be powered separately to allow storage of non-volatile variables.

For small memory size applications, the firmware can be loaded from external serial non-volatile memory to the internal RAM. No external parallel memory devices are necessary in this case.

Address	Area	Size
0 x D7FF FFFF 0 x D000 0000	SRAM 2 (external)	
0 x CFFF FFFF 0 x C800 0000	SRAM 1 (external)	
0 x C7FF FFFF 0 x C000 0000	SRAM 0 (external)	
0 x BFFF FFFF 0 x 8000 0000	SDRAM (external)	
0 x 2FFF FFFF 0 x 2000 0000	Extension Bus (external)	
0 x 1000 1FFF 0 x 1000 0000	Tightly coupled data memory (internal)	8K
0 x 0030 3FFF 0 x 0030 0000	Backup RAM (internal)	16K
0 x 0020 7FFF 0 x 0020 0000	Boot ROM (internal)	32 K
0 x 001F FFFF 0 x 0010 0000	Internal Peripheral Register Block	
0 x 0001 FFFF 0 x 0001 8000	4. SRAM Block (internal)	32 K
0 x 0001 7FFF 0 x 0001 0000	3. SRAM Block (internal)	32 K
0 x 0000 FFFF 0 x 0000 8000	2. SRAM Block (internal)	32K
0 x 0000 7FFF 0 x 0000 0000	1. SRAM Block (internal)	32 K

netX Memory Map

2.9 External Memory

The netX Memory Controller can drive static RAM or FLASH and SDRAM without any additional glue logic. SRAM / FLASH and SDRAM Interface share the address and data lines and the DQM Byte lane signals, while control signals are separate for each interface. The controller has not been designed to support peripherals, which can be connected via the Extension Bus and does hence not support WAIT or READY signals (cycle times are fixed and can not be extended by external components).

2.9.1 SRAM / FLASH Interface

The SRAM / FLASH Interface provides a total of three memory areas, which do not only have their own chip select signals, but also provide three independent configuration registers, allowing to set Memory Bus width and wait state parameters separately for each area. The parameters allow bus width configurations of 8, 16 or 32 Bit and wait states of up to 63 clock cycles.

Depending on the bus width of the appropriate memory component, the 24 Bit address bus allows access to 16 MByte, 32MByte or 64 MByte of static memory per memory area. For 8 Bit areas, address lines A23:0 represent the byte address, for 16 Bit areas the word address and for 32 Bit memory areas the dword address. Hence no address lines and thus address space are wasted when using 32 Bit or 16 Bit components. To allow byte or word access in 32Bit and 16Bit mode, four Byte Lane signals (MEM_DQM3-0) are provided.

Note:

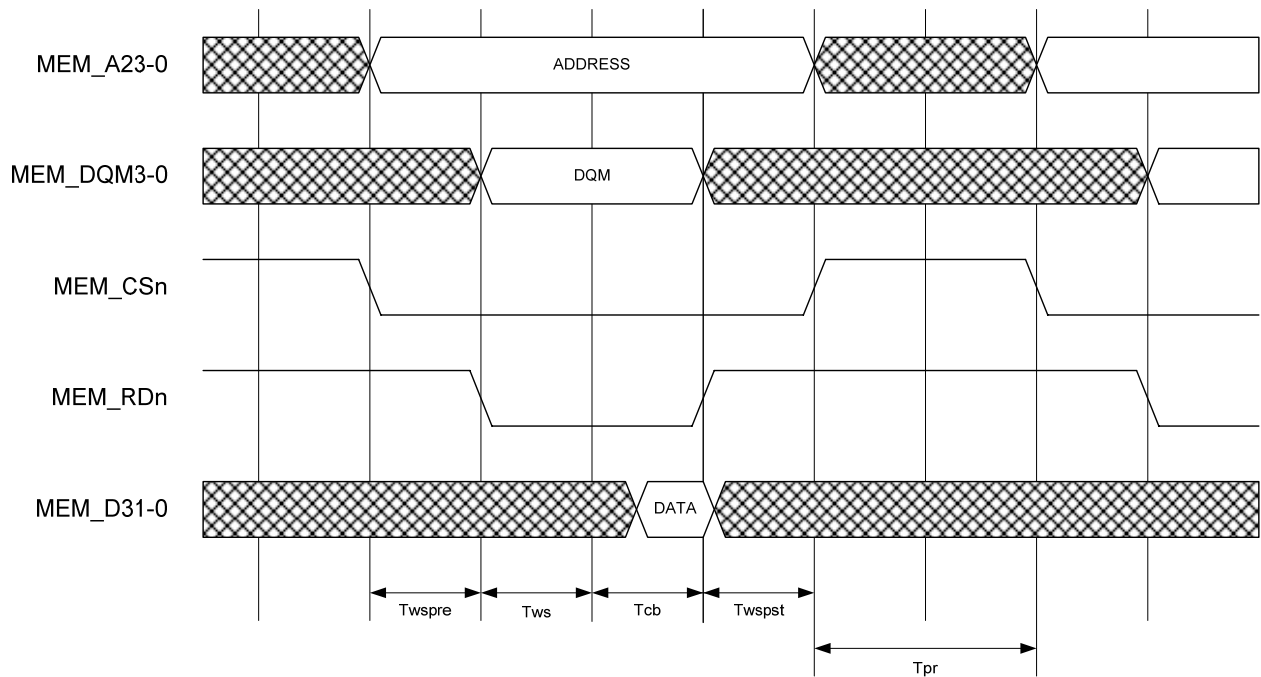
The interface does **not** support Burst or Page mode, as provided by many FLASH memory devices.

2.9.1.1 SRAM / FLASH Parameters

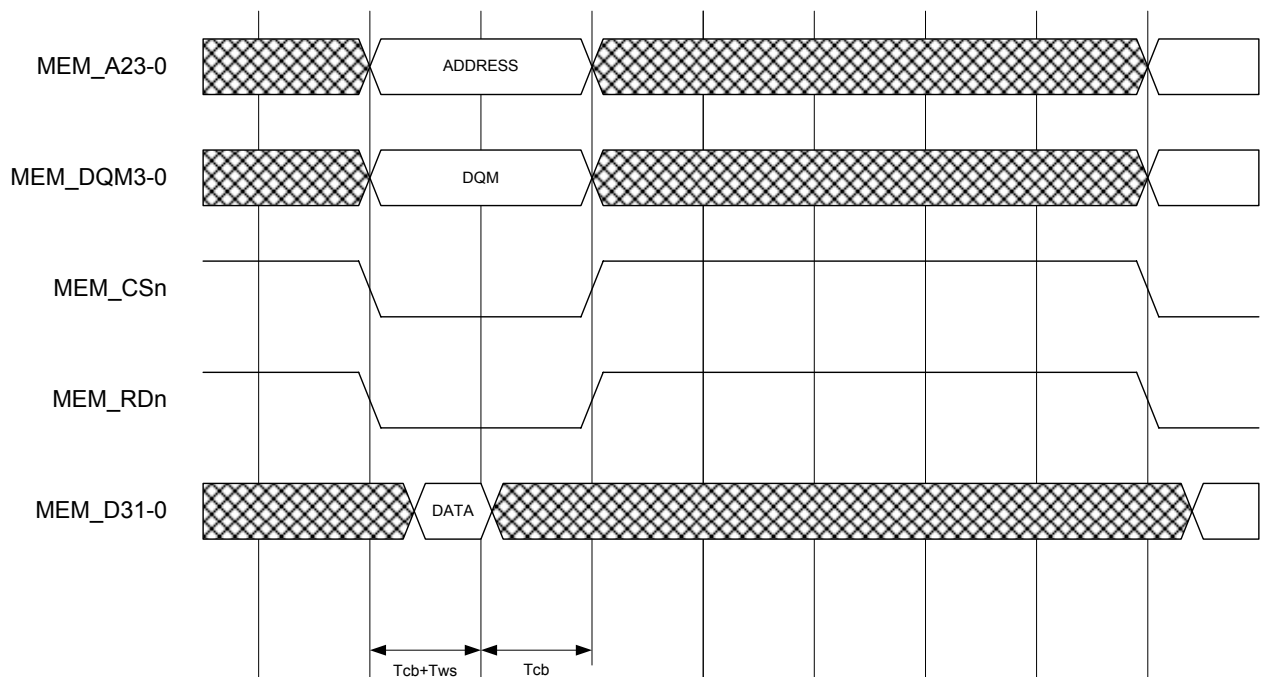
The following parameters can be set separately for every memory block:

Parameter	Description	Value	Dimension
Tws	Waitstates	0-63	CYC
Twspre	Additional Waitstates for setup time MEMSR_CS0-2 and MEM_A0-23 to MEMSR_OEn and MEMSR_WEn	0-3	CYC
Twspst	Additional Waitstates after data access Some components need this time to avoid data hold violations.	0-3	CYC
Bus width	Data bus width of the connected component	8 16 32	Bit

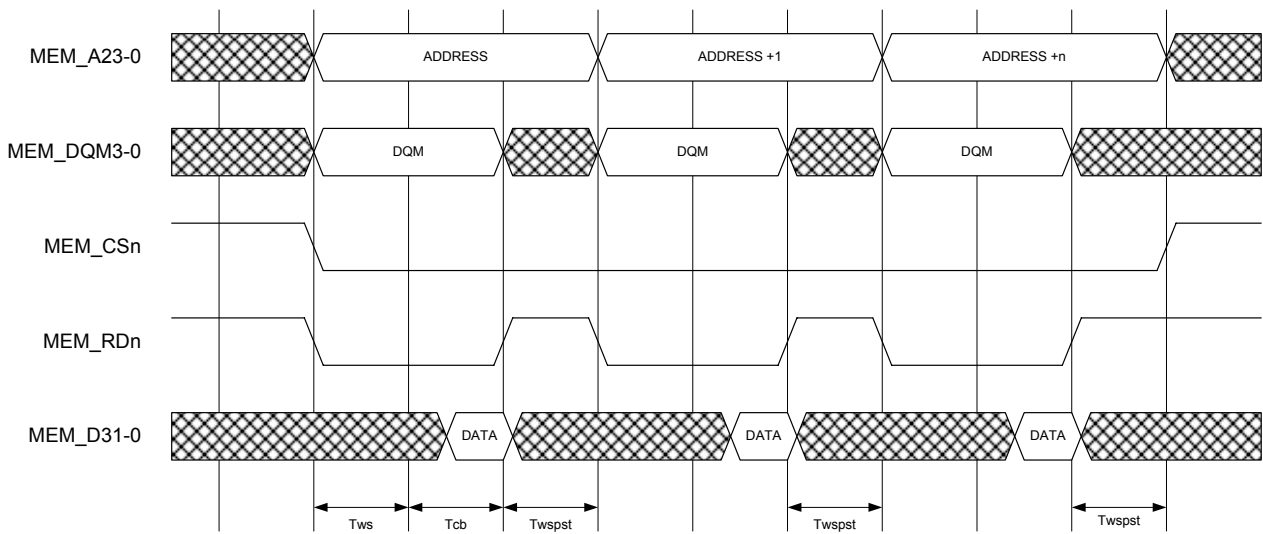
2.9.1.2 SRAM / FLASH Timing



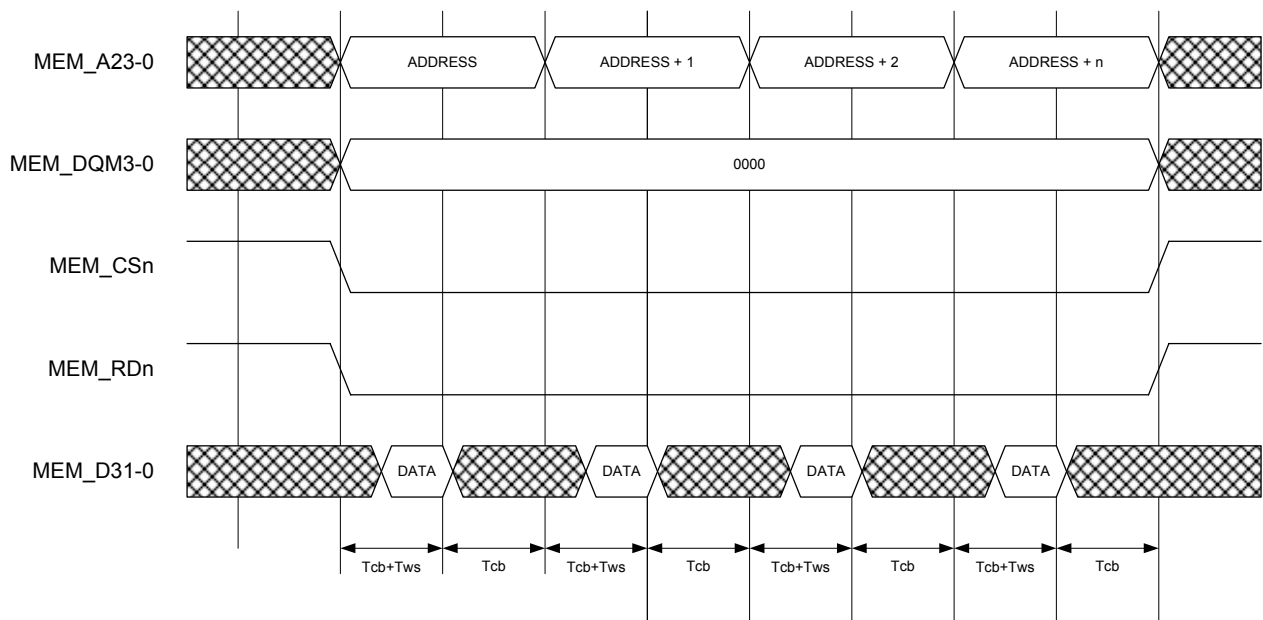
SRAM / FLASH Read Cycle from 32 Bit memory



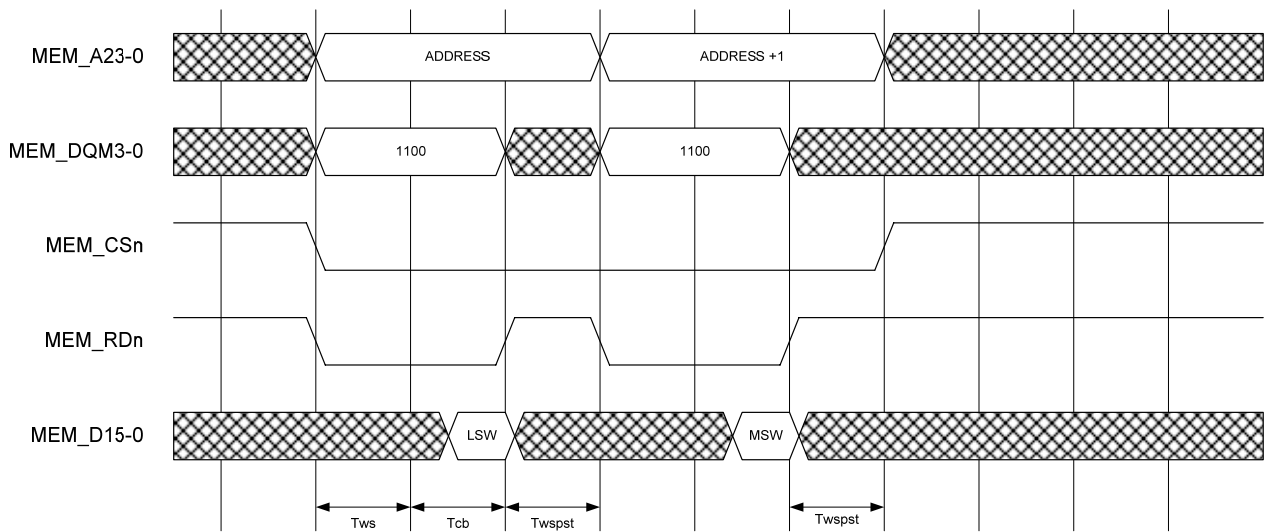
SRAM / FLASH Read Cycle from 32 Bit memory (no Pre and Post Wait-states)



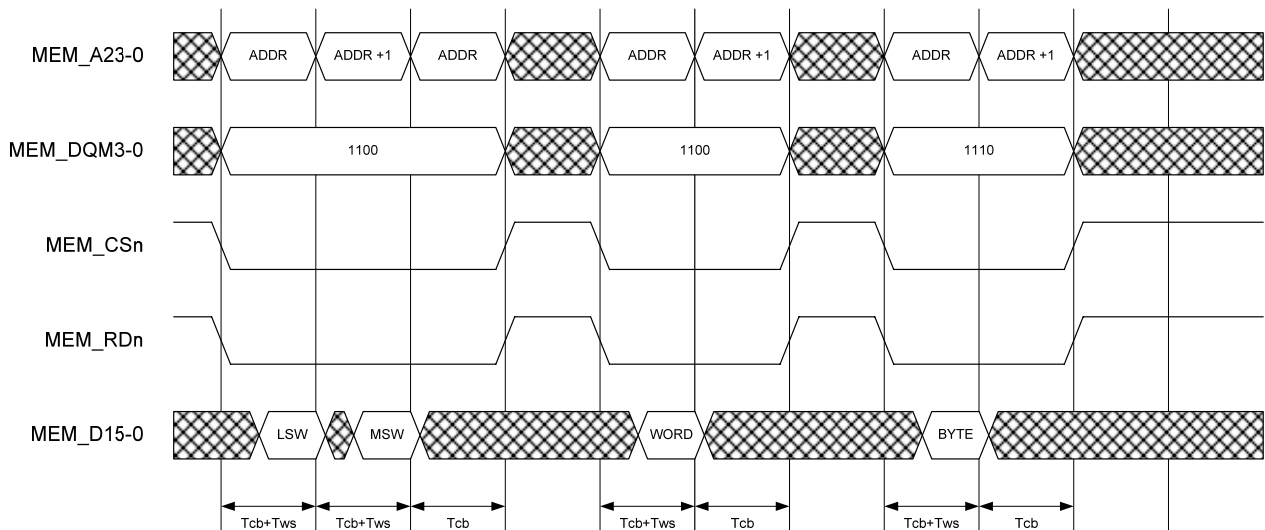
SRAM / FLASH Pseudo Burst Dword Read Cycle



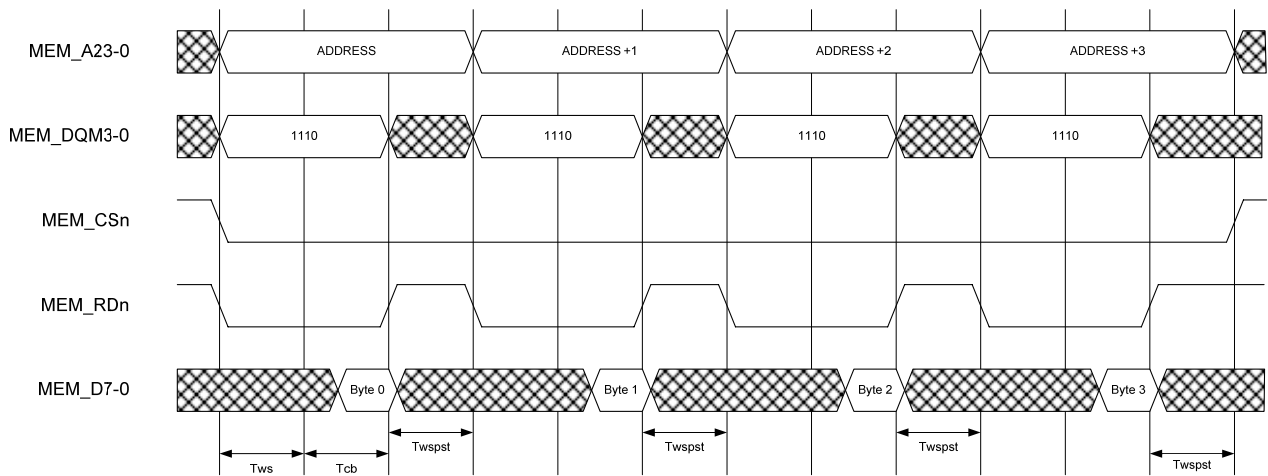
SRAM / FLASH Pseudo Burst Dword Read Cycle (no Pre and Post Wait-states)



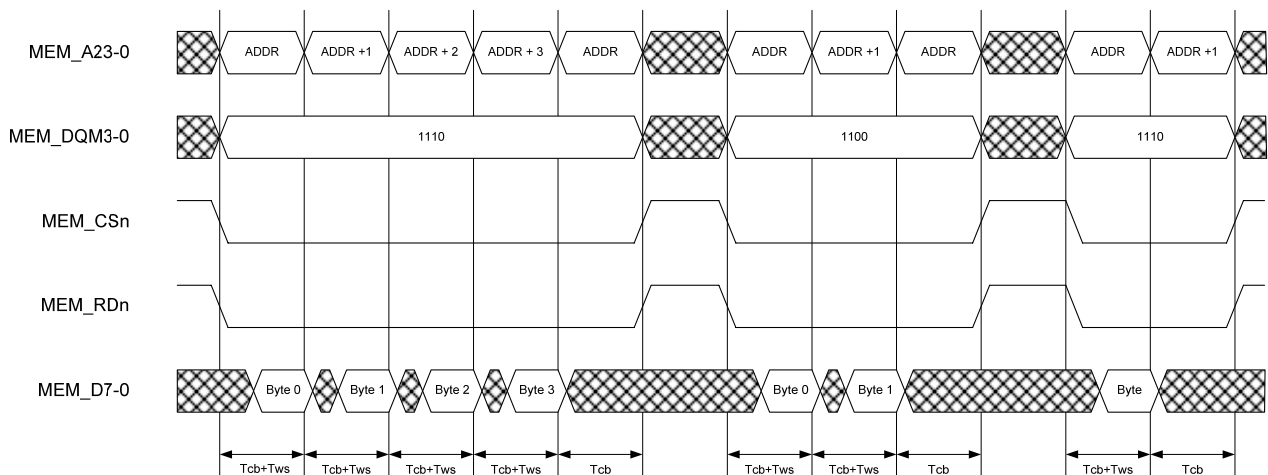
SRAM / FLASH Dword read access to 16 Bit memory



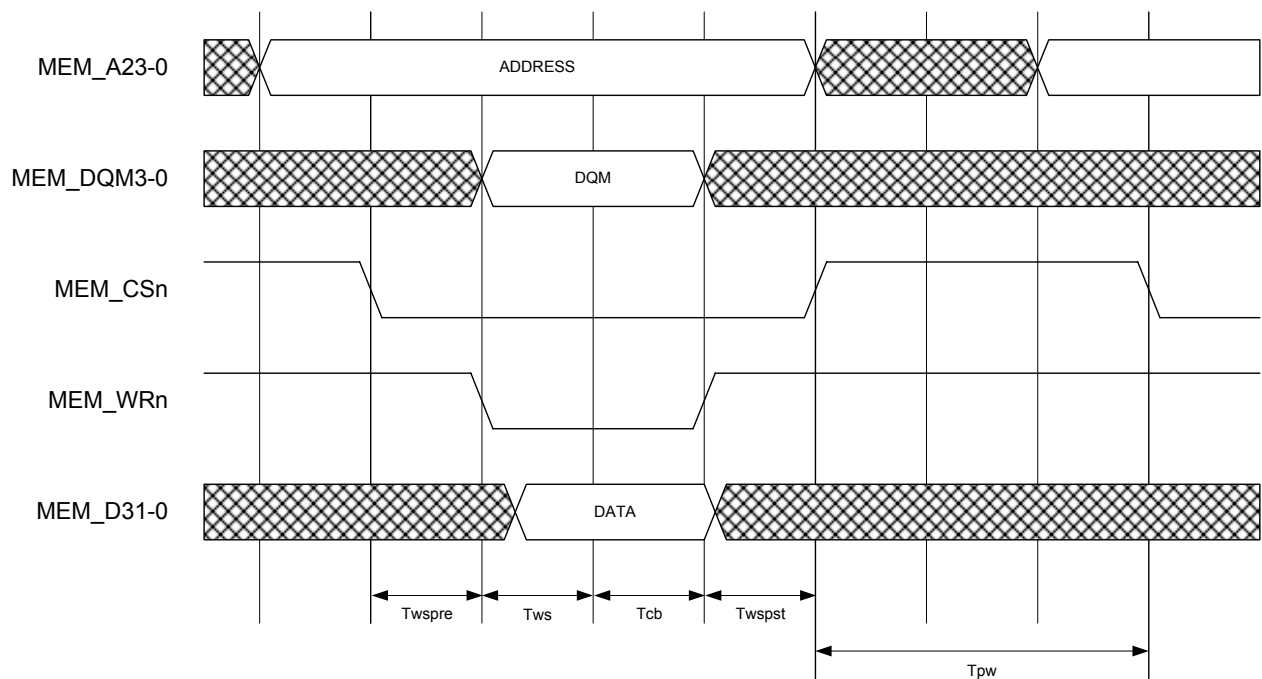
SRAM / FLASH Dword / Word / Byte read access to 16 Bit memory (no Pre and Post Wait-states)



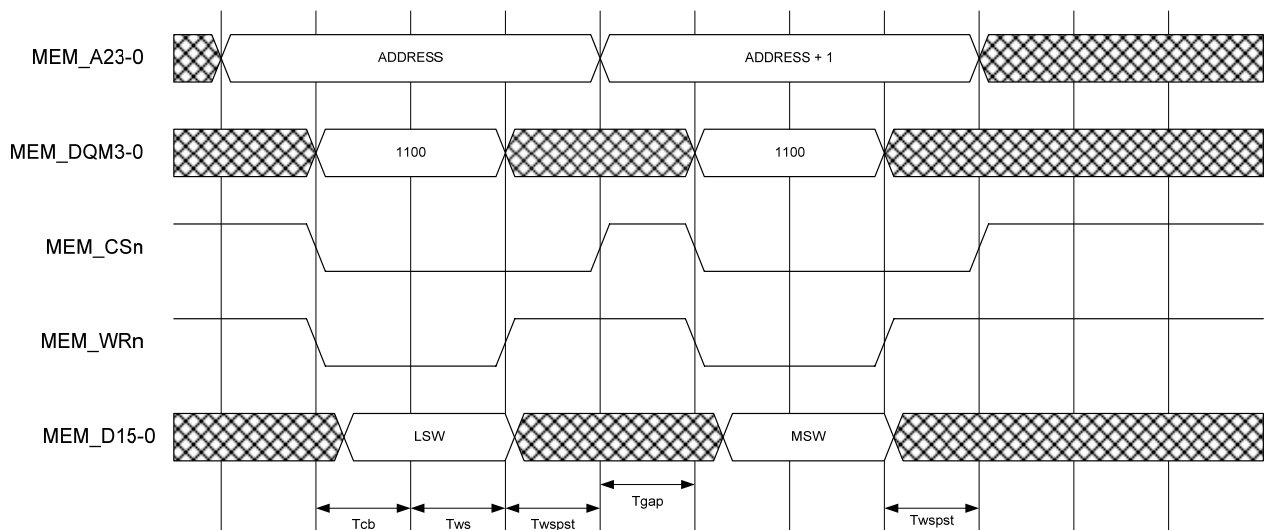
SRAM / FLASH Dword read access to 8 Bit memory



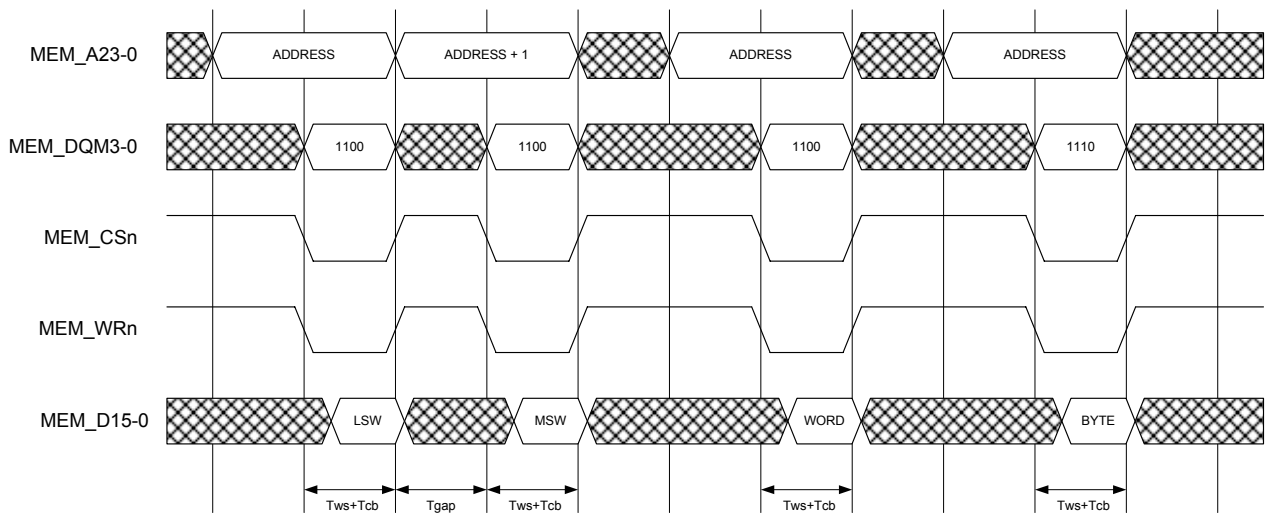
SRAM / FLASH Dword/Word/Byte read access to 8 Bit memory (no Pre and Post Wait-states)



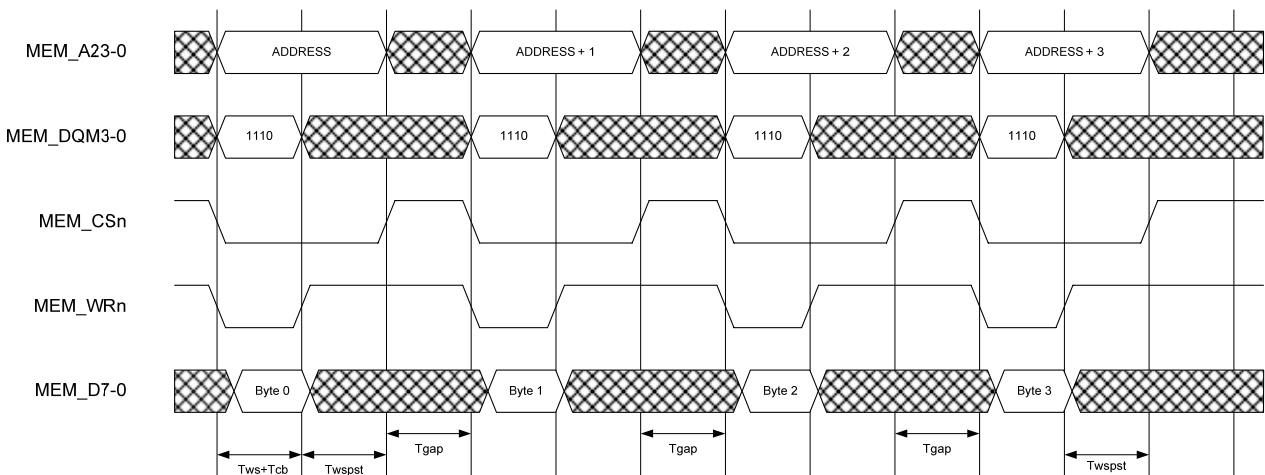
SRAM / FLASH Write Cycle



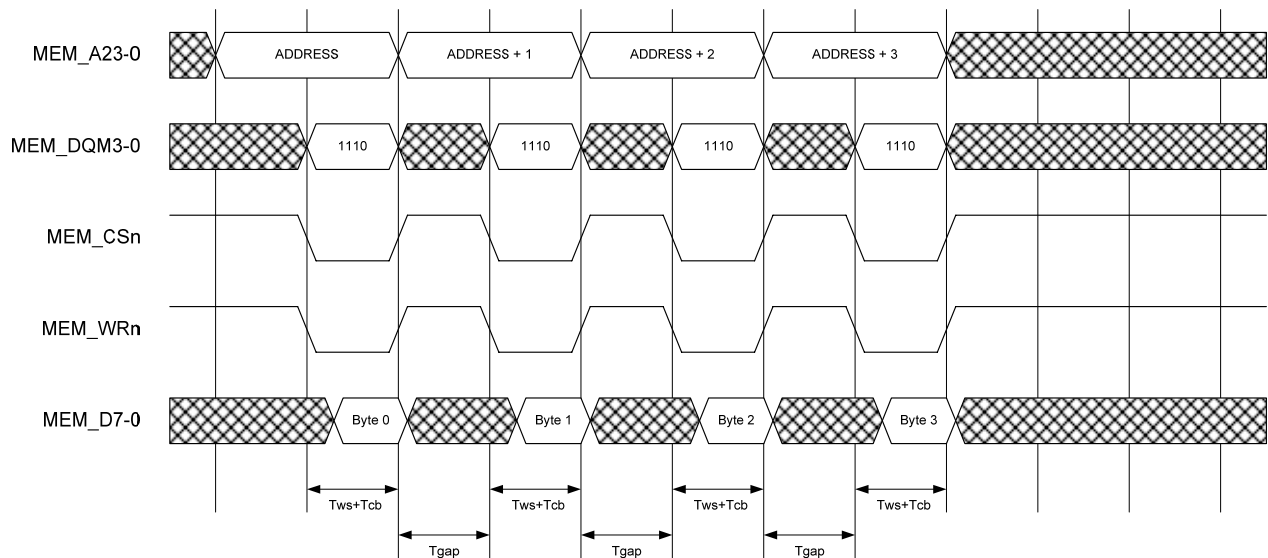
SRAM / FLASH Dword write access to 16 Bit memory



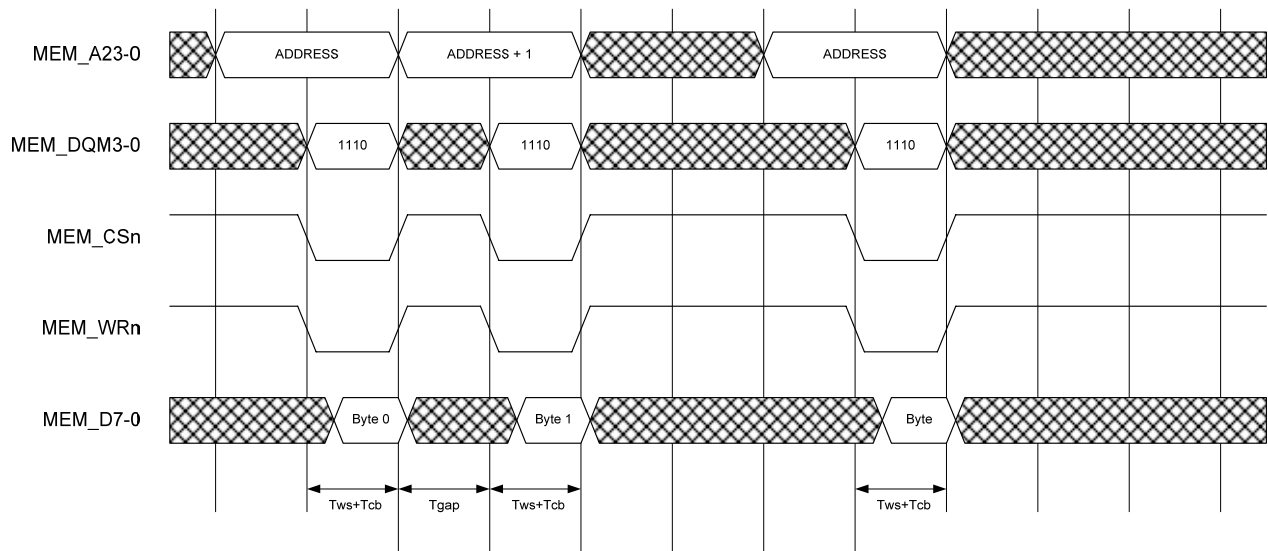
SRAM / FLASH Dword / Word / Byte write access to 16 Bit memory (no Pre and Post Wait-states)



SRAM / FLASH Dword write access to 8 Bit memory



SRAM / FLASH Dword write access to 8 Bit memory (no Pre and Post Wait-states)



SRAM / FLASH Word / Byte write access to 8 Bit memory (no Pre and Post Wait-states)

Parameter	min.	typ.	max.	Unit	Remarks
Tcb (cycle base time)	-	10	-	ns	
Twspre (pre cycle waitstate time)	0	-	30	ns	configurable in 10ns steps
Tws (waitstate time)	0	-	310	ns	configurable in 10ns steps
Twspst (post cycle waitstate time)	0	-	30	ns	configurable in 10ns steps
Tgap	-	10	-	ns	
Tpw (write to write pause)	10	-	30	ns	Either 10ns or 30ns (see below)
Tpr (read to read pause)	-	20	-	ns	only if mem waitstate enabled *

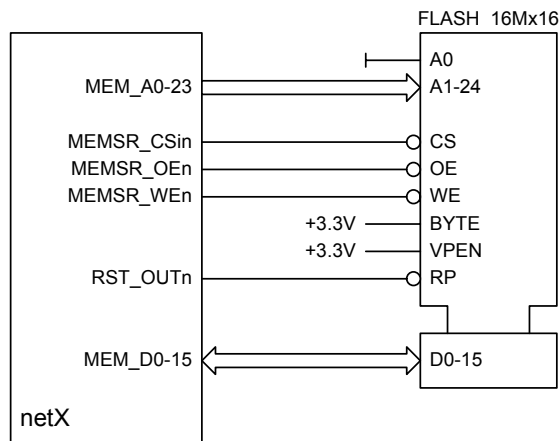
The access labeled "Pseudo Burst Read Cycle" is accomplished by using ARM load multiple commands (e.g. LDmia), allowing the consecutive read of up to 14 Dwords in a row. The same kind of cycle applies when executing firmware directly from FLASH or SRAM (instructions are fetched by burst type reads, that can however be longer than 14 dwords). These accesses require the AHBL waitstate for the memory interface (which is enabled by default after reset) to be disabled (-> "SDRAM timing fix"). If this waitstate is not disabled, the chip select signal will be de-asserted between consecutive read cycles with a minimum pause of 20ns between the cycles.

During write cycles the chip select signal will always be de-asserted between consecutive cycles, regardless of the AHBL waitstate, however the minimum pause between two cycles will drop from 30ns to 10ns with the waitstate disabled. Analog to the read cycle, the minimum pause can further only be achieved using ARM store multiple commands (e.g. STMIA).

For more detailed timing information (setup and hold times) see chapter 5.4.9.

2.9.1.3 Connecting SRAM / FLASH Components

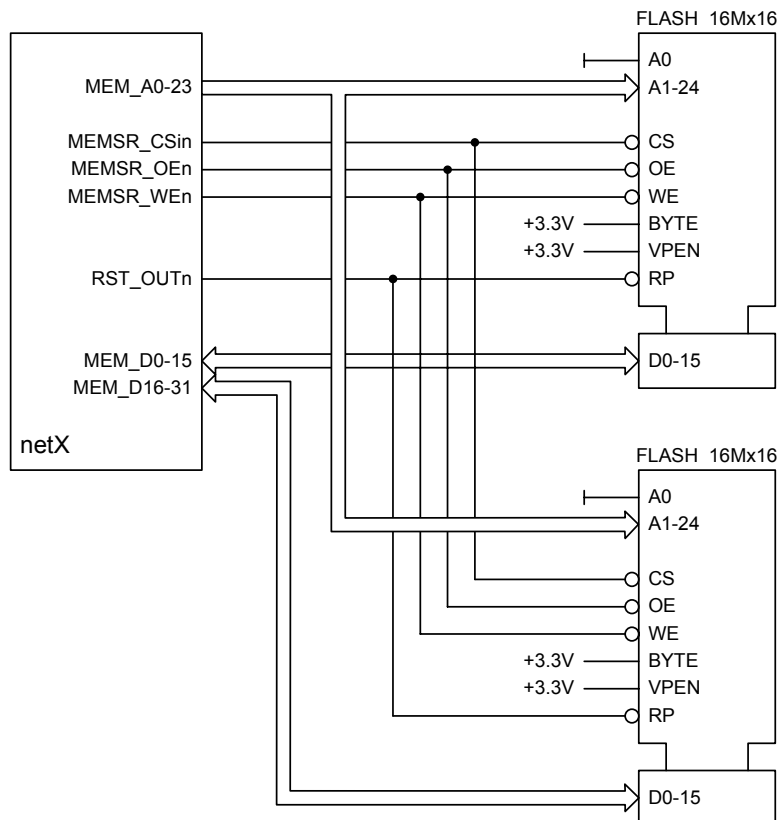
The following example schematics show how to connect SRAM and FLASH to the Memory Interface of the netX.



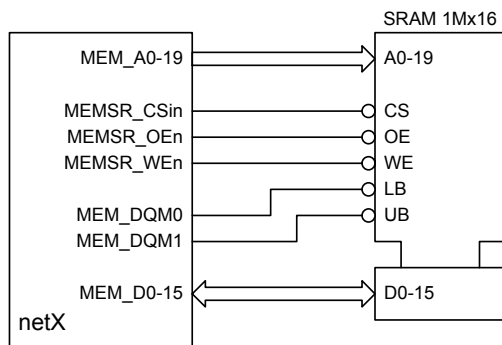
Note:

These examples assume the use of FLASH components that reserve address line A0 for Low / High Byte Selection (when operating in BYTE mode). However, this does not apply to all FLASH components on the market. Some components always expect 16-Bit addresses and hence require to connect the address lines one-to-one (A0 to A0, A1 to A1, etc.) Users should always consult the datasheet of their FLASH to determine the correct way to hook up the FLASH.

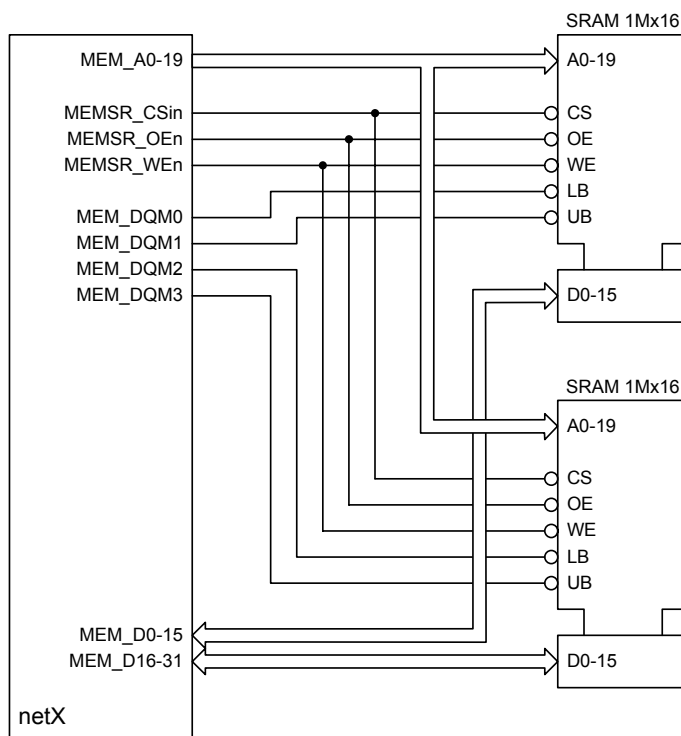
Interface with 16 Bit FLASH up to 32 Mbytes (no Byte access)



Interface with two 16 Bit FLASH (bus width 32 Bit) up to 64 Mbytes (no Byte or word access)



Interface with 16 Bit SRAM up to 2 Mbytes (byte access possible)



Interface with two 16 Bit SRAM (bus width 32 Bit) up to 4 Mbytes (byte and word access possible)

2.9.2 SDRAM

The SDRAM controller can drive all SDRAM Single Data Rate types from 16 MBit to 512 MBit, providing a 1 GByte address space from 0x80000000 to 0xBFFFFFFF.

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 size 16 Bit, 32 Bit
- CAS Latency 2 or 3
- Refresh-mode high and low priority
- Power save mode SDRAM-Self-refresh-Mode with disabled clock switch on / off SDRAM Controller

The SDRAM data bus width can be either 16 or 32 Bit. In order to achieve maximum memory performance, 32 Bit is recommended. However, depending on your performance requirements, 16 Bit may still be the better choice for some applications, due to reduced effort for the PCB layout and smaller board size.

The following table shows all supported memory combinations up to 256 Mbytes total memory.

SDRAM Memory Size	Organization	Number of Chips	Configuration	Total Memory Size
64 MBit	2 MBit x 32	1	1x32	8 MBytes
	4 MBit x 16	1	1x16	8 MBytes
128 MBit	4 MBit x 16	2	2x16	16 MBytes
	4 MBit x 32	1	1x32	16 MBytes
	8 MBit x 16	1	1x16	16 MBytes
	8 MBit x 16	2	2x16	32 MBytes
256 MBit	8 MBit x 32	1	1x32	32 MBytes
	16 MBit x 16	1	1x16	32 MBytes
	16 MBit x 16	2	2x16	64 MBytes
	16 MBit x 32	1	1x32	64 MBytes
512 MBit	32 MBit x 16	1	1x16	64 MBytes
	32 MBit x 16	2	2x16	128 MBytes
	64 MBit x 8	4	4x8	256 MBytes
	32 MBit x 16	1	1x16	64 MBytes

2.9.2.1 SDRAM Parameters

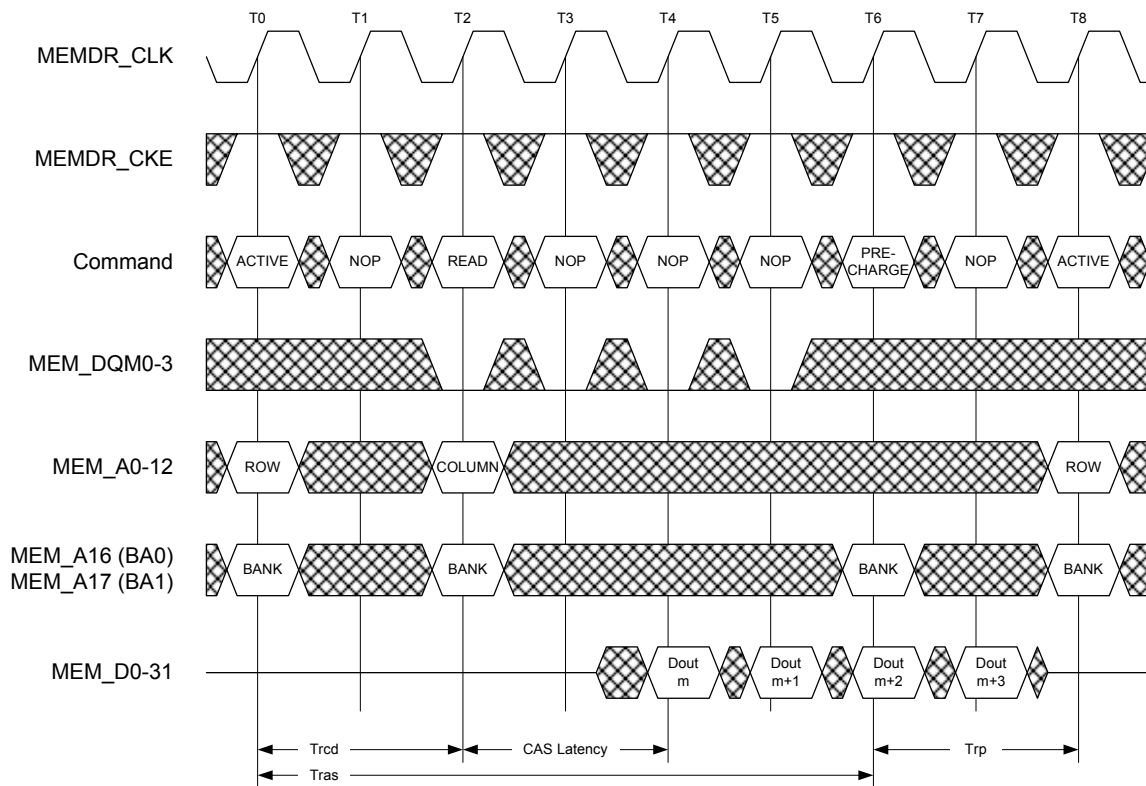
The SDRAM Controller runs with the 100 MHz system clock. There are a few parameters which have to be configured, according to the SDRAM components used. These are listed in the following table:

Parameter	Description	Value	Dimension
Trcd	ACTIVE to READ or WRITE delay / RAS to CAS delay	1-3	CYC
Twr	WRITE recovery time	1-3	CYC
Trp	PRECHARGE command period time	1-3	CYC
Tras	ACTIVE to PRECHARGE command time	3-10	CYC
Trfc	REFRESH to command time / AUTO REFRESH period	4-19	CYC
Trefi	Average periodic refresh interval	3.9 7.8 15.6 32.2	µs
CAS La- tency	CAS Latency	2-3	CYC

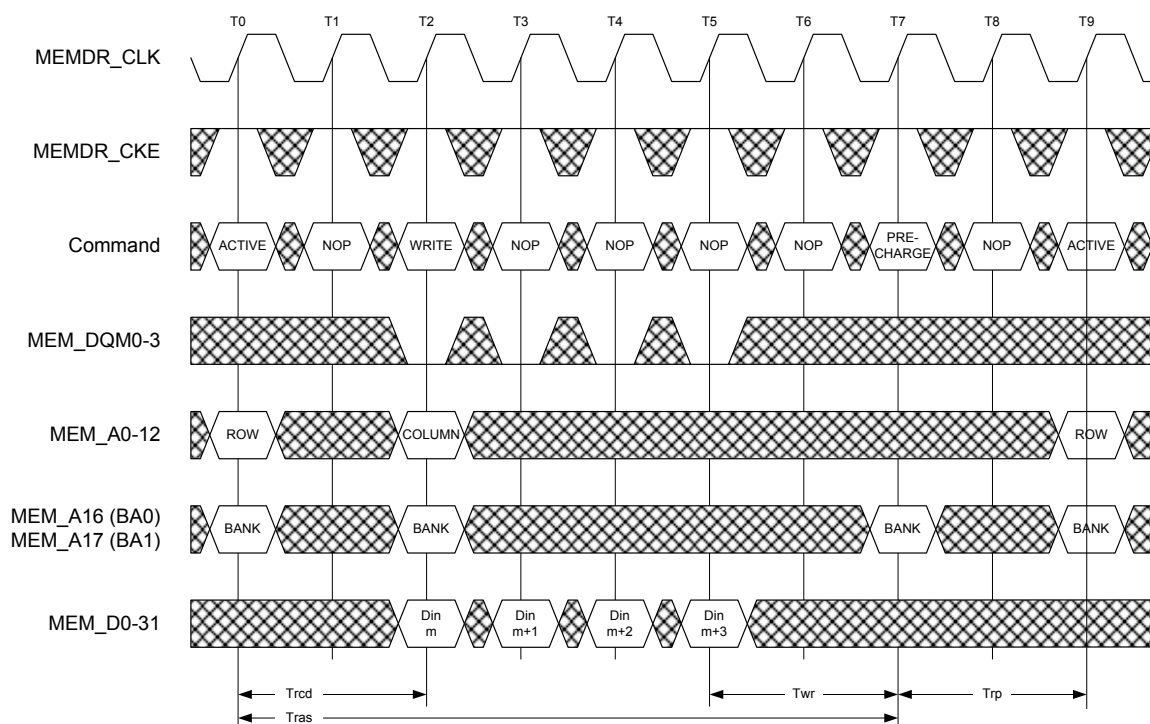
For details on the SDRAM configuration registers, please consult the “netX 500 Program Reference Guide”

2.9.2.2 SDRAM Timing

The following diagrams demonstrate the SDRAM parameters:



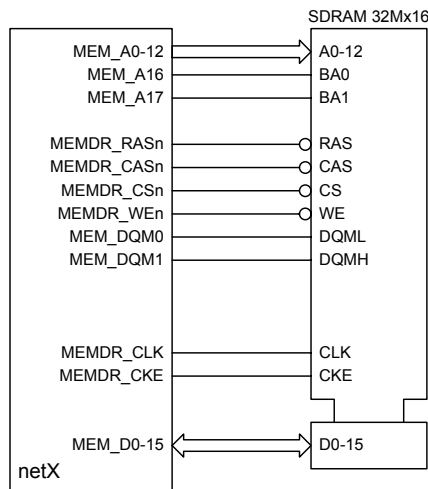
SDRAM read cycle



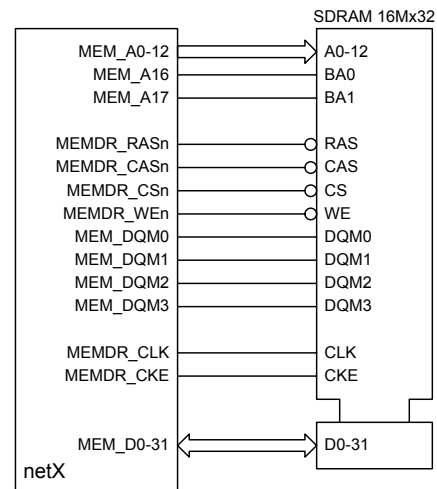
SDRAM write cycle

2.9.2.3 Connecting SDRAM Components

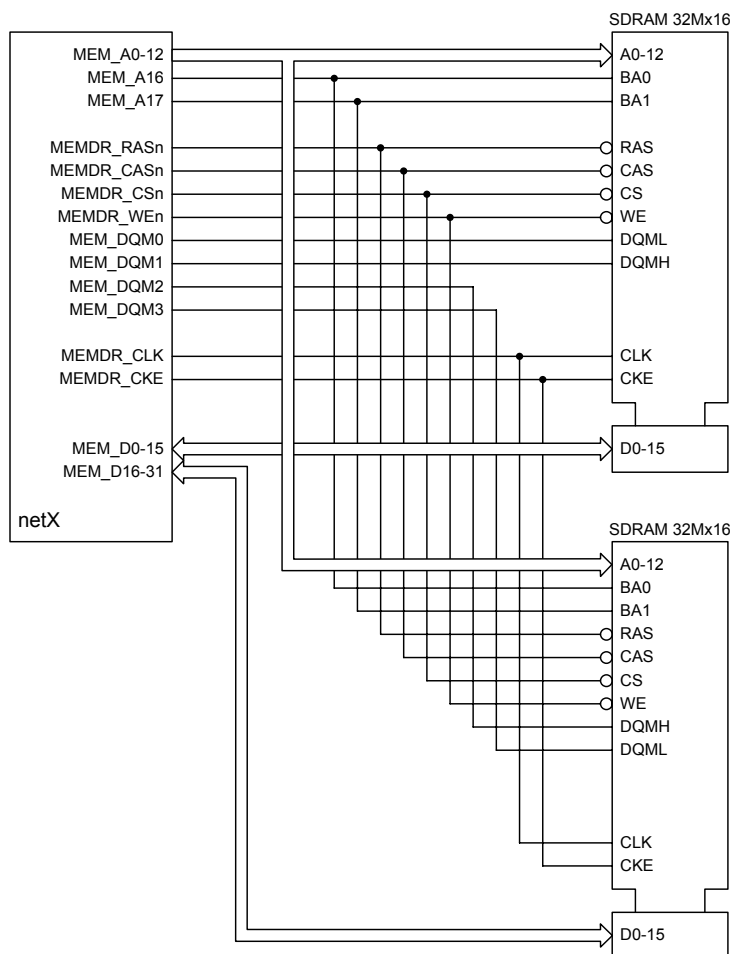
The following schematics show how to connect the SDRAM to netX for the different configurations.



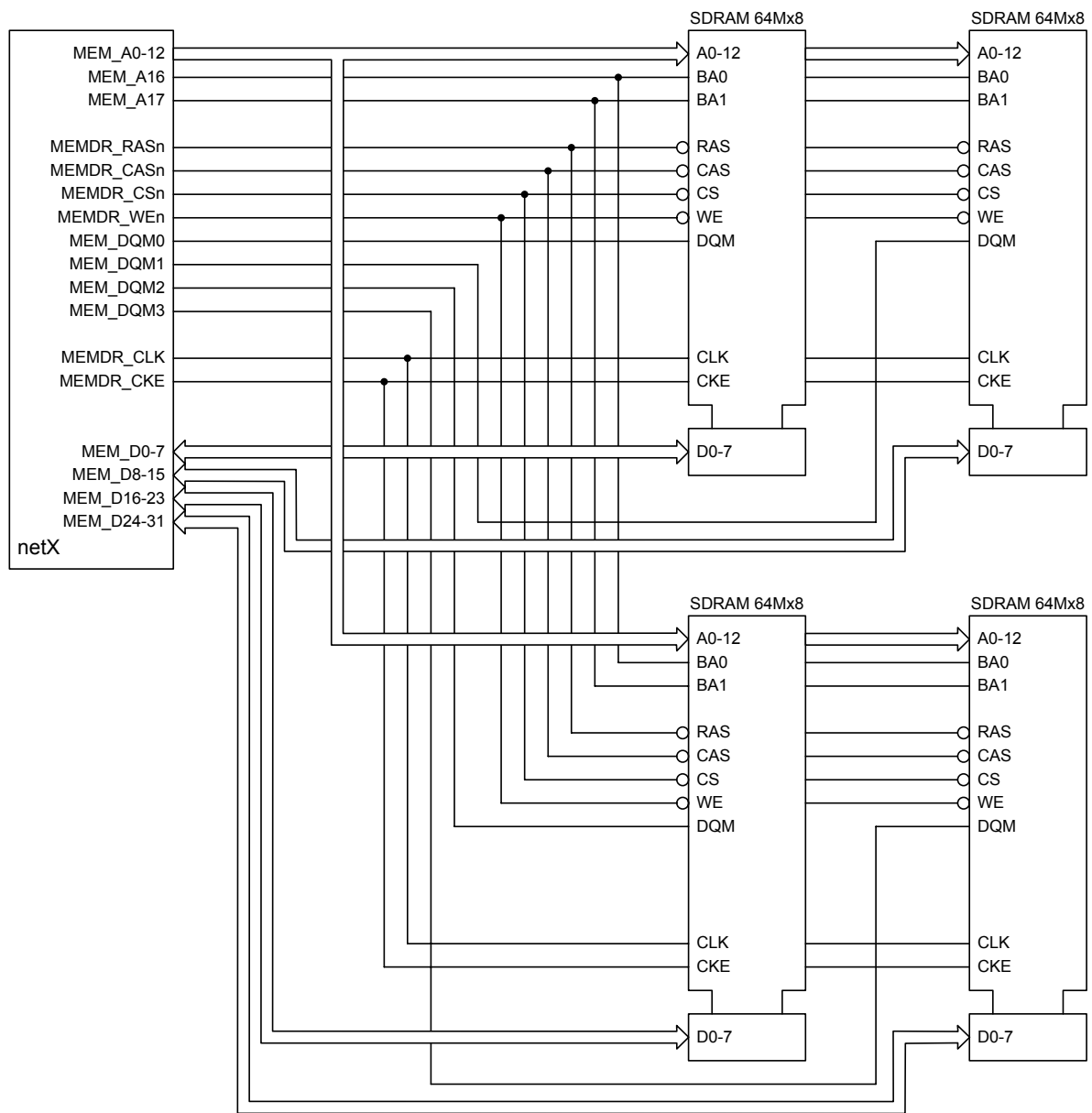
1x16:
Interface with 16 Bit SDRAM up to 64 MBytes



1x32:
Interface with 32 Bit SDRAM up to 64 MBytes



2x16:
Interface with two 16 Bit SDRAMs up to 128 Mbytes



4x8: Interface with four 8 Bit SDRAMs up to 256 MBytes

2.10 Extension Bus

The Extension Bus mode is one of the two basic modes, the host interface of the netX can be operated in. It is used for direct access of external peripherals or static memory from the netX. The following list provides an overview of the features of the Extension Bus:

- up to 4 chip select signals with independent programmable bus width size and timing
- 8 / 16 bit data bus width
- multiplexed or non multiplexed data bus
- 25 bit address range (32MByte) for each chip select, maximum address range 4 x 32 MByte
- Programmable Bus timing with wait states, setup and hold times
- support for external ready or wait signal
- unused interface pins can be used as general purpose input / output pins
- all Extension Bus signal lines can be made 5V input tolerant

2.10.1 Extension Bus Configuration

In order to use the Extension Bus, the netX host interface must be configured for 'Extension Bus Mode' in the `DPM_ARM_IF_CFG0` register. Further, it is important to also configure each required signal line of the Extension Bus for host interface mode in the `DPM_ARM_IO_MODE0` and `DPM_ARM_IO_MODE1` registers. Unused signal lines of the Extension Bus may be left configured for I/O mode (e.g. unused upper address lines or data lines 15-8 when using an 8 Bit device only), allowing to use them as additional PIO signals.

If the Extension Bus is enabled, there is no possibility for a host system to access the virtual Dual-Port memory of the netX chip.

The configuration of each Extension Bus memory area is done through the 'Extension Bus Configuration Chip Select' register. For a detailed register description see the 'Program Reference Guide'.

2.10.2 Extension Bus Address Space and netX Memory Allocation

The Extension Bus address space is accessible by the netX through four different memory base addresses. All internal accesses between `0x2000_0000` and `0x3FFF_FFFF` are mapped to the external peripherals or memory connected at the Extension Bus. Each of the four chip select signals uses a 32M byte memory window. The address lines `EXT_A25` and `EXT_A26` are internally decoded for the chip select generation. The upper address lines `EXT_A27` and `EXT_A28` are not decoded. The following table shows the allocation addresses in the netX memory range.

netX Start Address	netX End Address	Description	
<code>0x2800_0000</code>	<code>0x3FFF_FFFF</code>	-	Mirrored address windows of chip selects
<code>0x2600_0000</code>	<code>0x27FF_FFFF</code>	<code>EXT_CS3n</code>	32M byte address window
<code>0x2400_0000</code>	<code>0x25FF_FFFF</code>	<code>EXT_CS2n</code>	32M byte address window
<code>0x2200_0000</code>	<code>0x23FF_FFFF</code>	<code>EXT_CS1n</code>	32M byte address window
<code>0x2000_0000</code>	<code>0x21FF_FFFF</code>	<code>EXT_CS0n</code>	32M byte address window

2.10.3 Address and Data Byte Steering

The Extension Bus logic is capable of handling all 8, 16 or 32 bit netX accesses from netX to the external peripherals or memory devices. According to the bus width configuration of the Extension Bus an access from netX side is converted to external 8 or 16 bit accesses. For example a 32 bit access to an 8 bit device is converted to four 8 bit accesses.

The following table shows the address and data steering between netX and external devices.

Note:

When performing an access to an Extension Bus memory location by the ARM CPU, the CPU will be stopped, until the access is completed. Hence, 32 bit accesses to a slow 8 bit device may have a significant impact on the netX performance and will increase interrupt latency, or DMA transfer times.

Extension Bus Data Width	Address / Data Byte Steering		
8 Bit	netX Data 0-7	Extension Bus Data 0-7	ADR[1:0] = 00
	netX Data 8-15	Extension Bus Data 0-7	ADR[1:0] = 01
	netX Data 16-23 →	Extension Bus Data 0-7	ADR[1:0] = 10
	netX Data 24-31	Extension Bus Data 0-7	ADR[1:0] = 11
16 Bit (Byte access)	netX Data 0-7	Extension Bus Data 0-7]	ADR[1:0] = 00
	netX Data 8-15	Extension Bus Data 8-15	ADR[1:0] = 01
	netX Data 16-23 →	Extension Bus Data 0-7	ADR[1:0] = 10
	netX Data 24-31	Extension Bus Data 8-15	ADR[1:0] = 11
16 Bit (Word access)	netX Data 0-15	Extension Bus Data 0-15]	ADR[1:0] = 00
	netX Data 16-31 →	Extension Bus Data 0-15	ADR[1:0] = 10

Address and Data Byte Steering

2.10.4 Intel / Motorola Data Format

The netX system works with Intel memory format (little endian). There is no special mode for connecting to Motorola type (big endian) memory systems, hence any necessary data conversions must be performed by software, when accessing big endian components.

2.10.5 Multiplexed / Non-Multiplexed Data Bus

Besides devices with a separate memory and data bus, the Extension Bus also supports devices with a multiplexed data bus. The timing and polarity of the address latch enable signal and data hold times can be programmed in a wide range.

2.10.6 Data Ready or Data Acknowledge

When the external wait/ready function is enabled, the EXT_RDY signal is sampled at the rising system clock edge during the active access. It must be asserted at least 2.5 clock cycles prior to the rising edge of read or write strobe signal to add wait states to the current cycle. The polarity of the signal can be programmed, supporting devices with ready- as well as devices with wait signal generation.

2.10.7 End-Of-Cycle

When the end of the access cycle specified by the Trdwrcyc parameter is reached, the cycle will be stopped immediately and all active signals will be de-asserted.

Users should take care when setting timing parameters, as the versatility of the interface also allows settings that don't make sense. If, for example, the write delay time Twron is greater than the complete cycle time specified by the Trdwrcyc parameter, the EXT_WRn signal will never be asserted!

2.10.8 Pin Description of Extension Bus

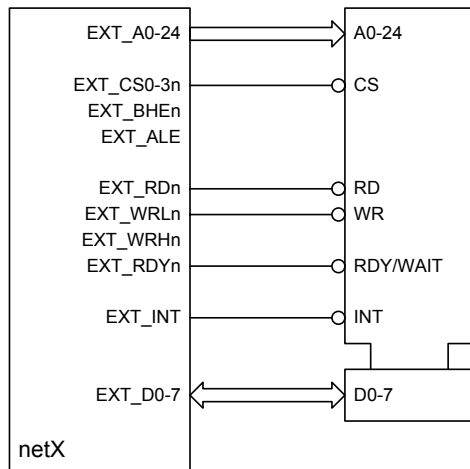
The following table provides an overview of all Extension Bus signals.

Pin Name	Intel	Motorola	Description
EXT_D0-15	DATA[15:0]	DATA[15:0]	Extension Bus Data 0-15
EXT_A0-24	ADDR[24:0]	ADDR[24:0]	Extension Bus Address 0-24
EXT_CS0n	CS0n	CS0n	Extension Bus Chip select 0
EXT_CS1n	CS1n	CS1n	Extension Bus Chip select 1
EXT_CS2n	CS2n	CS2n	Extension Bus Chip select 2
EXT_CS3n	CS3n	CS3n	Extension Bus Chip select 3
EXT_ALE	ALE	ASn	Extension Bus Address Latch Enable
EXT_BHEn	BHEn	BHEn	Extension Bus High Enable
EXT_RDn	RDn	RD/WRn	Extension Bus Read
EXT_WRLn	WRLn	DSLn	Extension Bus Write Low
EXT_WRHn	WRHn	DSHn	Extension Bus Write High
EXT_RDY	WAITn	READYn	Extension Bus Ready
EXT_IRQ	IRQn	IRQn	Extension Bus Interrupt Request

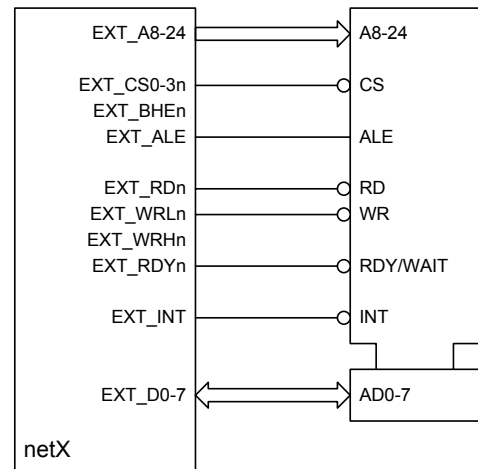
Extension Bus Interface signals

EXT_D0-15	Bi-directional data bus. When an 8 bit data bus is selected the high byte of the data bus can be used as programmable input / output pins.
EXT_A0_24	25 Bit address bus. Each address line can be individually enabled or disabled. So it is possible to use the higher address lines as programmable input / output pins if the complete address range is not required. The address line EXT_A0 is low when an even byte or word is accessed. It will be active for 8 bit and 16 bit data bus width.
EXT_CS0n EXT_CS1n EXT_CS2n EXT_CS3n	External chip selects for extension bus. These signals are decoded using internal address lines EXT_A27 and EXT_A28.
EXT_ALE	Address latch enable used for multiplexed address- / data-bus accesses. If active, it indicates a valid address driven on the data bus. Used for Intel and Motorola multiplexed bus modes.
EXT_BHEn	Byte High Enable. This signal is asserted when an odd byte or a word is accessed. It will be active for 8 bit and 16 bit data width.
EXT_RDn	Active low read signal, can be also configured as a direction signal EXT_RD/WRn for Motorola type interfaces.
EXT_WRLn	Active low write or write low signal, depending on configuration. This signal can be configured as the data strobe or low byte data strobe signal for Motorola type interfaces.
EXT_WRHn	This is the high byte write strobe signal and can be also configured as the high byte data strobe signal for Motorola type interfaces.
EXT_RDY	External data ready or data acknowledge input for access cycle extension. The polarity of this signal can be configured. The input signal is internally connected to high level when used as input / output pin.
EXT_IRQ	External interrupt request pins, level triggered. The interrupt lines are internally connected to a low level when the pins are programmed as input / output pins.

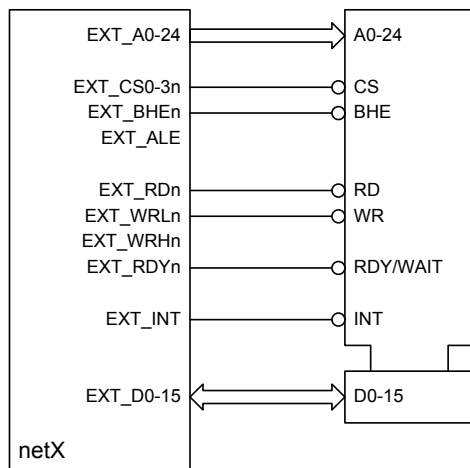
2.10.9 Extension Bus Component Connection



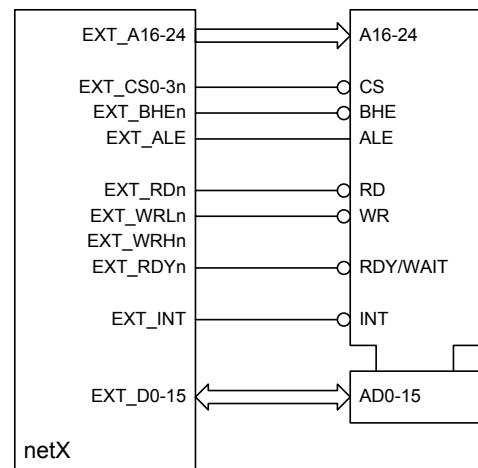
Intel, 8 Bit non multiplexed, one write signal



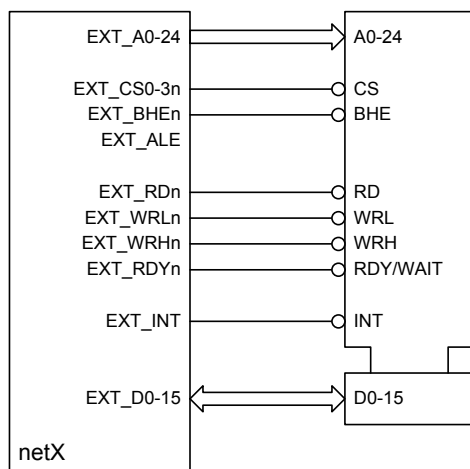
Intel, 8 Bit multiplexed, one write signal



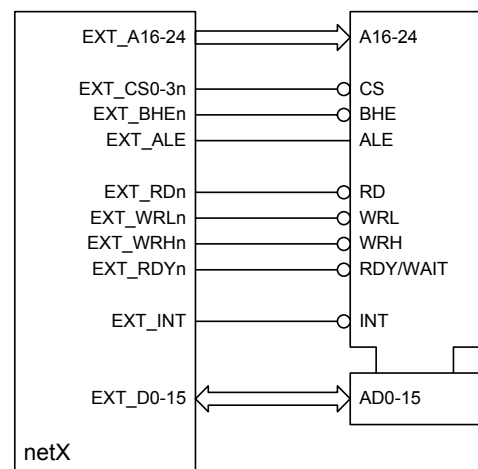
Intel, 16 Bit non multiplexed, one write signal



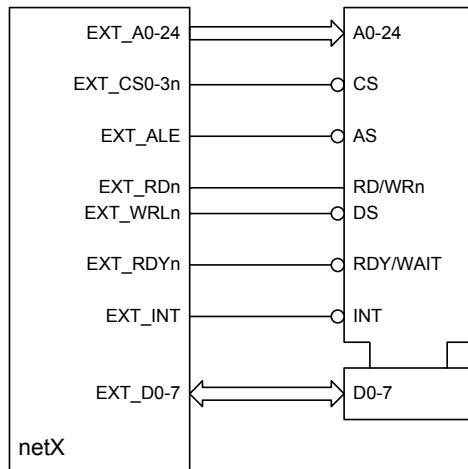
Intel, 16 Bit multiplexed, one write signal



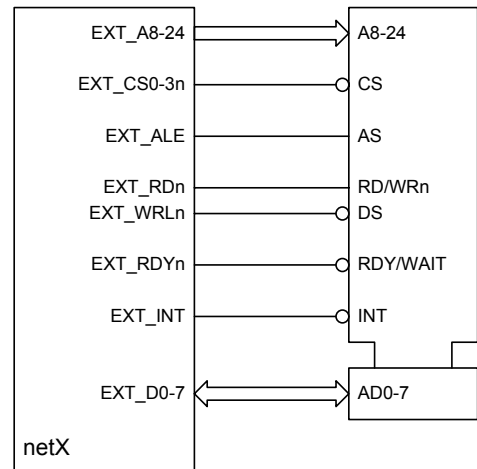
Intel, 16 Bit non multiplexed, write low/high signals



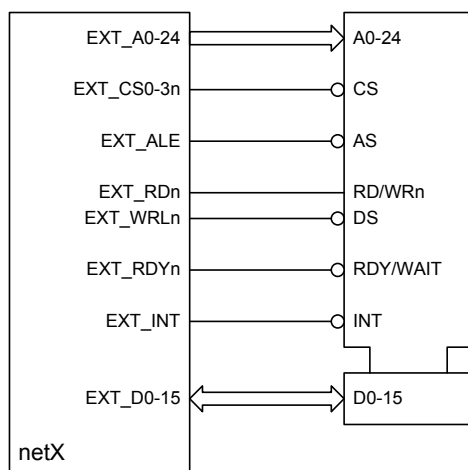
Intel, 16 Bit multiplexed, write low/high signals



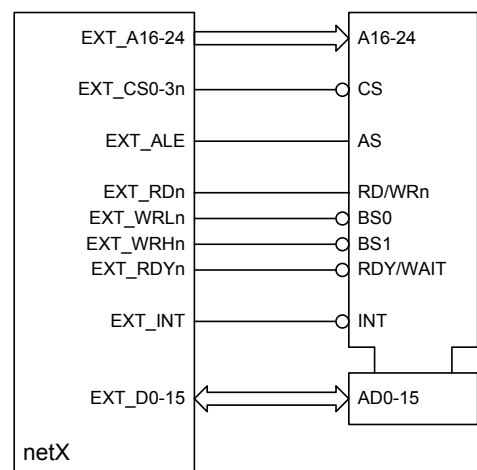
Motorola, 8 Bit non multiplexed, one write signal



Motorola, 8 Bit multiplexed, one write signal

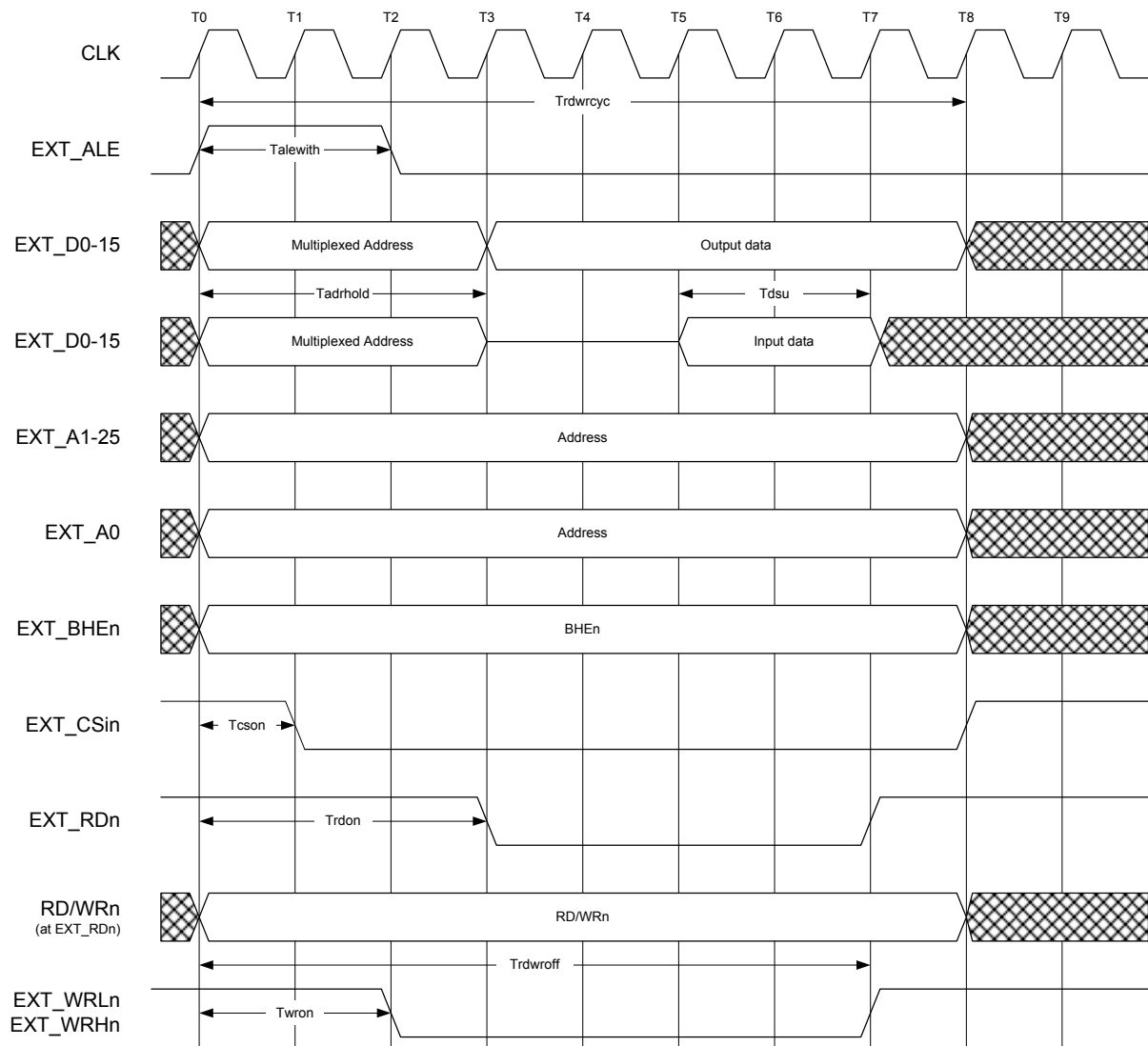


Motorola, 16 Bit non multiplexed, one data strobe



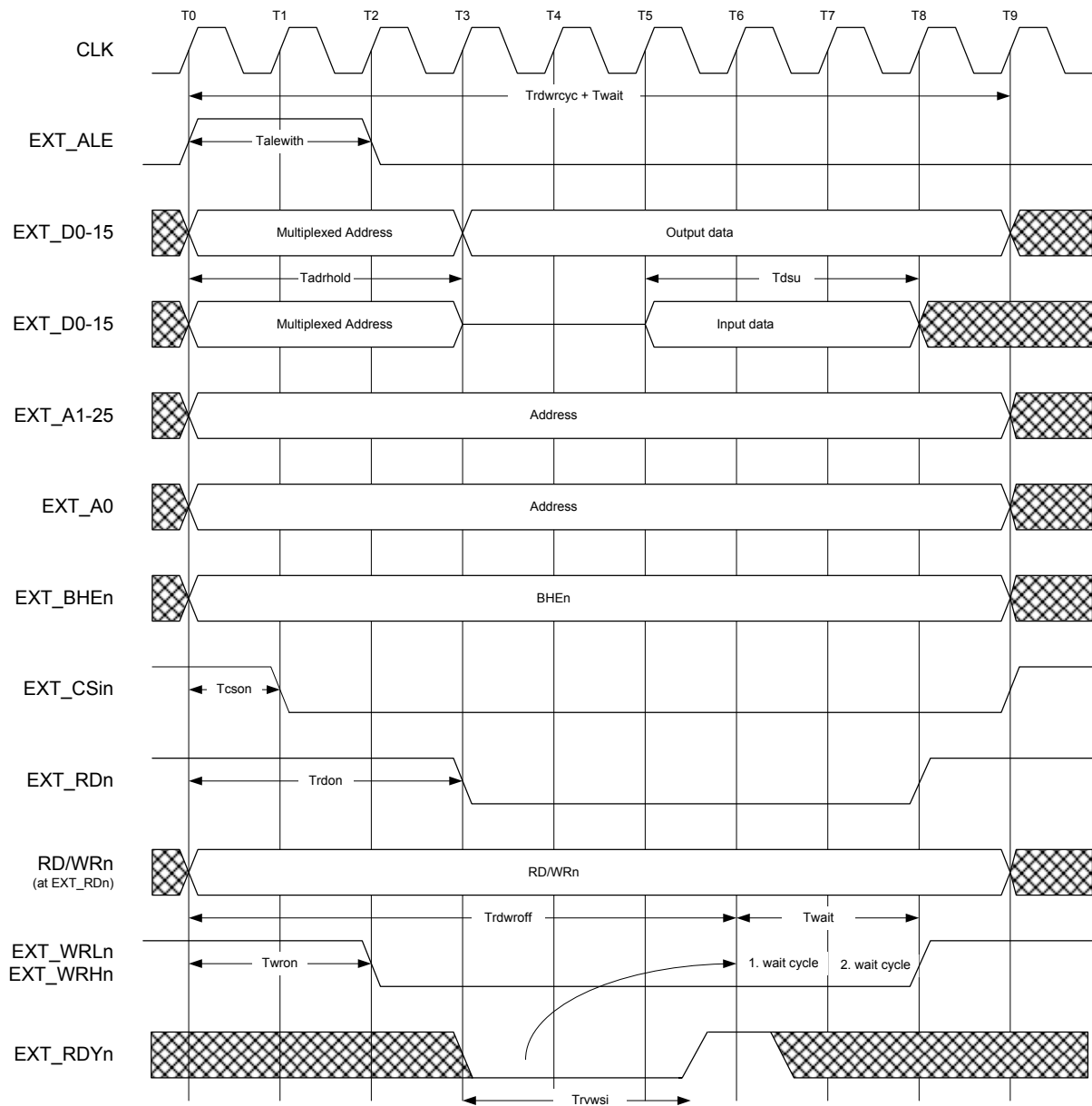
Motorola, 16 Bit multiplexed, two data strobes

2.10.10 Extension Bus Timing without Wait-states



Parameter	Description	Value	Dimension
Talewidth	Delay time from start cycle until ALE inactive	0-7	CYC
Tadrhold	Delay time from start cycle until invalid address at the data bus.	0-7	CYC
Tcson	Delay time from start cycle until Chip Select insertion	0-7	CYC
Trdon	Delay time from start cycle until RD low in system clocks	0-7	CYC
Twron	Delay time from start cycle until WR low in system clocks	0-7	CYC
Trdwroff	Delay time from start cycle until RD or WR inactive delay for accesses in system clocks	0-31	CYC
Trdwrcyc	Set the end of an access cycle in system clocks The values 0x00 and 0x01 are interpreted as 0x20	2-32	CYC
Tdsu	Data setup time for read accesses	1	CYC

2.10.11 Extension Bus Timing with Waitstates



Parameter	Description	Value	Dimension
Talewidth	Delay time from start cycle until ALE inactive	0-7	CYC
Tdrhold	Delay time from start cycle until invalid address at the data bus.	0-7	CYC
Tcson	Delay time from start cycle until Chip Select insertion	0-7	CYC
Trdon	Delay time from start cycle until RD low in system clocks	0-7	CYC
Twron	Delay time from start cycle until WR low in system clocks	0-7	CYC
Trdwroff	Delay time from start cycle until RD or WR inactive delay for accesses in system clocks	0-31	CYC
Trdwrcyc	Set the end of an access cycle in system clocks The value 00hex and 01hex are interpreted as 20hex	2-32	CYC
Tdsu	Data setup time for read accesses	1	CYC
Trwsi	Ready valid to waitstate insertion delay	2.5	CYC

2.11 Dual-Port memory

The Dual-Port Memory (DPM) interface is used for allowing data transfer between the netX chip and an external host system. Unlike standard DPM components, the netX DPM is a **virtual** Dual-Port memory, which appears as a 64k linear memory to the host side, while accesses to the DPM are redirected to one or up to eight different memory areas, located anywhere within the complete netX memory map, including also register areas. The interface of the netX DPM is widely configurable and can hence support virtually any common μ P interface like Intel 80186/188, Motorola MC68000, ColdFire, or ARM CPU based processors, just to name a few. The Dual-port memory structure is programmable from netX side and hence fully firmware specific. While any output signal of the netX DPM always uses 3.3V signaling voltage, all DPM inputs or I/Os can be made 5V tolerant by hooking up separate power pins to 5V. The following list provides an overview of the Dual-port memory features.

- 64 kByte maximum total size with a fixed structured control block at the highest 512 Bytes
- up to eight DPM memory areas of individual size, that can be mapped to different areas of netX memory space.
- up to sixteen handshake register pairs with programmable register width of 8 or 16 bit
- support for 8 or 16 bit data bus
- support for multiplexed or non multiplexed data bus
- programmable control lines that let external glue logic become obsolete
- interrupt and data ready generation
- signals lines can be made 5V input tolerant

2.11.1 Dual-Port Memory Interface Mode

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

If the Dual-port memory is enabled, there is no possibility to use the Extension Bus interface.

Note:

As long as not configured otherwise (firmware or DPM/Extension Bus boot mode), all host interface pins are switched into high impedance mode after power up or during and after reset. Users must take precautions to prevent these floating signals from causing a problem in their system, by adding pull-up or pull-down resistors to the appropriate DPM signals (e.g. interrupt line DPM_INT or data ready line DPM_RDY).

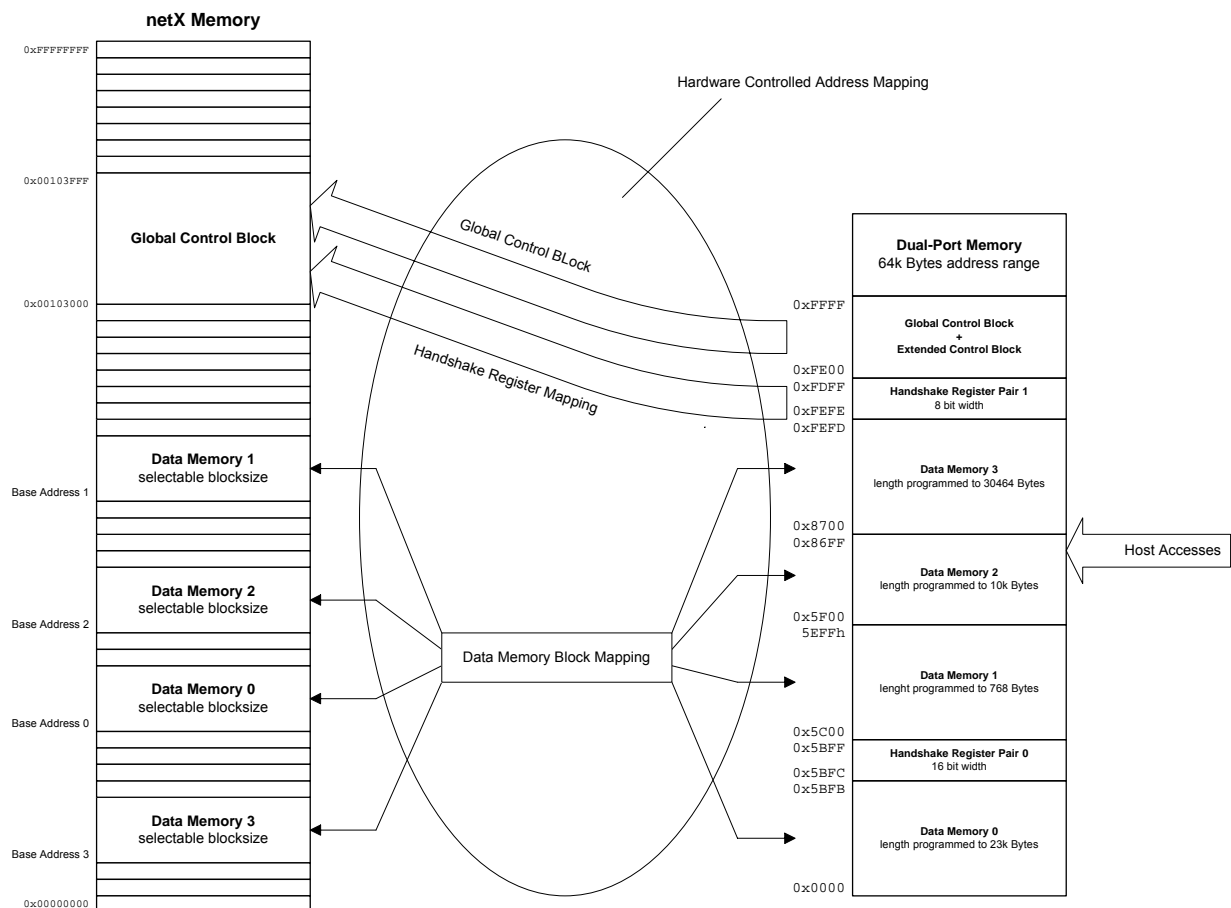
2.11.2 Dual-Port Memory Structure and Allocation

As mentioned in the previous chapter, the netX does not contain a real Dual-port memory as the user may know it. All addresses within the linear address range of the Dual-port memory are mapped to one or more netX memory areas by a programmable address decoder. The destination can be a special control register, an internal memory area or an external memory device. Only a small area of 512 bytes located at the end of the DPM is “hard-mapped” to the host accessed control register block.

Note:

Though they are destined for host access only, these host accessed control registers, as well as the host accessed handshake register block are also accessible from netX side. This possibility was implemented for debugging purposes and should hence only be used with great care! Reading or writing these control registers from netX side may cause undesired behavior. This applies particularly to accesses to the handshake register block, which affects the handshake interrupt handling (reading a host access handshake register from the netX side may clear a pending handshake interrupt linked to this handshake cell, before the host was able to respond to that interrupt)

The following figure shows an overview of the memory address mapping of the Dual-port memory.



The total address size of the linear accessed Dual-port memory is 64 kByte with a fixed structured control block at the highest 512 Bytes from 0xFE00 until 0xFFFF. The rest between address 0x0000 and 0xFDFF is available for data transfer. The following table shows the Dual-Port memory structure from host side. Keep in mind that the the lower area from 0x0000 until 0xFDFF is **only an example** because the structure can be programmed from netX side.

	Byte 3	Byte 2	Byte 1	Byte 0
0xFFFFC - 0xFF00	fixed global control block - Interrupts - Software Reset - System Status - Host Timer - Process Data Watchdog Timer			
0xFEFF - 0xFE00	extended control block			
0xFDFF - 0xA000	Unused area with nearly 24k bytes			
0x9FFF - 0x9FFC	Host → netX Handshake Flags (16 bit)		netX → Host Handshake Flags (16 bit)	
0x9FFB - 0x7800	Data Memory Block 8 with nearly 10k bytes			
0x77FF - 0x7000	Data Memory Block 7 with 2k bytes			
0x6FFF - 0x6000	Data Memory Block 6 with 4k bytes			
0x5FFF - 0x5000	Data Memory Block 5 with 4k bytes			
0x4FFF - 0x4FFC	Host → netX Handshake Flags	netX → Host Handshake Flags	Handshake Data Memory 2 Bytes	
0x4FFB - 0x2800	Data Memory Block 4 with nearly 10k bytes			
0x27FF - 0x2000	Data Memory Block 3 with 2k bytes			
0x1FFF - 0x1000	Data Memory Block 2 with 4k bytes			
0x0FFF - 0x0000	Data Memory Block 1 with 4k bytes			

Example of Dual-port memory structure

2.11.3 Global Control Block

The global control block hosts several DPM related registers for exchanging status information, interrupt handling, watchdog and timer functions. This 'hard mapped' 512 Byte block is located at DPM offset 0xFE00. The next chapters describe the functions that can be accessed through the global control block.

2.11.3.1 DPM Interrupts

The global control block contains interrupt status and mask registers for all DPM related interrupts. Sources of a netX side DPM interrupt can either be the five possible interrupt input signals (PIOs 35,36,40,47,72), the 16 handshake cells, a DPM watchdog timeout or a memory lock error (host access to unmapped DPM memory area). On the host side, possible interrupt sources are again the 16 handshake cells, DPM watchdog timeout, a system status change and host timer event.

Each interrupt request can be enabled or disabled independently and is "OR-ed" with any other enabled DPM interrupts to the global interrupt request for the host interface, which is routed to the VIC (Vectored Interrupt Controller) of the netX, respectively to the host interrupt output signal of the netX.

While all DPM interrupt sources have separate flag bits, their status can additionally be read through a common 8 Bit interrupt vector, that always reflects the active (and enabled) interrupt with the highest priority. When this interrupt is cleared, the vector displays the next active (and enabled) interrupt with the next lower priority. When no (enabled) interrupts are active, the vector has the value 0x00. An interrupt status flag is cleared by writing a one to the status flag (writing a zero has no effect). When a handshake register is read, the corresponding interrupt is cleared automatically.

One global interrupt request bit signals, that one or more interrupt request is active. While all interrupt status flags will always be set when the appropriate source is active (regardless if the interrupt has been enabled or not), the global Bit is only set when at least one of the active interrupts has also been enabled, which correspondingly applies to the reset vector.

For more details see the appropriate chapter of the 'netX 500/100 - Program Reference Guide'.

2.11.3.2 DPM Software Reset

The DPM Software Reset register allows the host processor to reset the netX chip. This is done by a special access procedure. After initiation, the chip will be reset after 1 ms (this delay ensures the proper completion of the current write cycle). The software reset is comparable to any other Reset except the Power On Reset (see chapter 2.5 (Reset) for details).

For initiation of the DPM software reset, the host has to write the sequence 0x00, 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40 and 0x80 to the 'DPMHS_RES_REQ' register.

For more details see the appropriate chapter of the 'netX 500/100 - Program Reference Guide'.

2.11.3.3 System Status

The DPM Global Control Block also contains a system status register ('SYS_STA') which is determined for exchanging status information between the netX and the host processor chip. Some bits of this register can only be written from netX side and some bits are only writeable from host side. This register also controls the state of the status LED 'RDY' and 'RUN' (see chapter 2.3). There is an 8 bit field for the NETX_STATUS_CODE which can be used by the netX software to provide status information to the host. If enabled, a system status interrupt (external interrupt to the host system) will be generated on any write access to the data bits 8 to 15 of the system status register.

For more details of the system status register see the 'netX 500/100 - Program Reference Guide'.

2.11.3.4 Host Timer

The host has the possibility to use this timer for cyclic interrupt generation or as a count down timer.

The timer is a 16 bit count down timer and can generate an interrupt at the host system side. There are two different timer modes: The first mode will let the timer stop after counting down to zero and set the interrupt event only once. In the second mode, the interrupt can be generated too but on reaching zero, the timer will be reloaded with the configured timeout value and will continue to run allowing a cyclic interrupt event to be generated.

Host Timer programming is done by setting the timer start count value TMR_START in the 'DPMHS_TMR_START' register, selecting the time base, the function mode and setting the start flag in the 'DPMHS_TMR_CTRL' register. When the start flag is set, the counter is loaded with the timer start count value and started. Setting the start flag again is also possible during timer operation and causes a reload of the timer, while clearing the start flag causes the timer to stop counting.

The clock divider register bits provide the possibility to select different clock time bases. The divider counter for time base generation will be also cleared when the host timer start flag is being set or the time base value is changed.

For more details of host timer programming see the 'netX 500/100 - Program Reference Guide'.

2.11.3.5 Process Data Watchdog Timer

- Watchdog Timer Host
- Watchdog Timer netX

Watchdog Timer for Host Supervision (Watchdog Host)

This 16 bit count down watchdog timer can be used for supervision of the host system. The netX chip can set a timeout value, which can be read from the host system. The timeout value can be changed by the netX any time. The watchdog timer is being disabled by setting the timeout value to zero, which is the default value after a power on reset.

Once the timer is running, the host system has to periodically trigger the timer within the watchdog timeout period to avoid generating the host timeout event which causes a host timeout interrupt on the netX side. Whenever the watchdog timer is triggered by the host system, the timer register will be loaded with the timeout value. Setting another timeout value from netX side will not change the current counter value of the watchdog timer. The new timeout value will be loaded when the host system performs the next triggering of the watchdog timer. After configuration by the netX, the timer will not start to run, until the host system triggers the watchdog the first time.

The time base of the watchdog timer is fixed to 100 µs. The timeout range can be selected between 100µs and 6.50 sec. The following formula is used for calculation of the timeout parameter:

$$\text{TIMEOUT_HOST} = \text{TIMEOUT_VALUE} \times 100 \mu\text{s}$$

For triggering the watchdog timer, there is a register which can be accessed from the host system. Triggering the watchdog timer is only possible with a special access code. This access code is generated by a pseudo random generator and is only valid for one access, which reduces the probability that a crashed host application still retriggers the watchdog. The following sequence must be completed to trigger the watchdog timer:

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

Watchdog Timer for netX Supervision (Watchdog netX)

This watchdog timer has an analog functionality to the watchdog timer for host supervision, however the direction is reversed (netX triggers watchdog and host interrupt is generated on a watchdog timeout).

For details on watchdog timer registers see the 'netX 500/100 - Program Reference Guide'.

2.11.3.6 Extension Control Block

This block is reserved for future extensions.

2.11.3.7 Data Memory Area / Data Memory Blocks

The complete DPM memory area can be divided into a maximum of eight memory blocks. From host view, these blocks are located at the address range between 0x0000 and 0xFDFE. The start address, where each Dual-port memory block is mapped in the internal netX address range, is programmable by the netX. Accesses to these data memory blocks are forwarded by the netX host interface to the netX memory range. The target of a data memory block can either be an internal memory area, a register area, or external memory (SDRAM/SRAM/FLASH).

There is no default mapping after a power on reset, yet the host interface isn't even enabled, hence the DPM area can only be accessed from the host, after the DPM has been configured by the netX firmware or the DPM boot mode (see chapter 2.3)

When the host system tries to access a memory address of the Dual-port memory which is not mapped, the access is aborted and, if enabled, an interrupt event occurs on the host side (Memory Lock Interrupt). While write accesses will be ignored, read accesses will always reveal the data 0x0BAD.

For details on how to configure the DPM mapping, please consult the netX 500/100 Program Reference Guide (chapter 5.2)

2.11.3.8 Handshake Registers

To allow synchronization of the data transfer between host system and netX, a total of 16 handshake register pairs with interrupt capability are available. Handshake registers can be 8 or 16 bit wide and are hence available for all selectable DPM bus widths. While both registers of a handshake register pair are readable by host and netX, the upper register of the pair can only be written by the host, while the lower register can only be written by the netX. Writing to a handshake register from the host side, can generate a host interface interrupt on netX side, while writing from the netX side can generate a host interrupt (signal DPM_INT). After receiving an interrupt, the processor (host CPU, respectively netX) has to either check the interrupt vector or the handshake flags to identify the handshake register(s) that caused the interrupt and either read the corresponding handshake register, which will automatically clear this handshake interrupt, or clear the corresponding handshake interrupt flag (by writing '1' to the flag bit).

The Dual-port memory offset address of each register pair is set by the netX. If the handshake register pair is located at the upper end of a data memory block, then the structure is equal to standard Dual-port memory devices which are commonly used to connect two microprocessor systems with each other.

The base address of the 8 or 16 bit handshake register pairs in the Dual-port memory must be a DWORD aligned address. These addresses can be set anywhere in the Dual-port memory address range, except the 512 Byte control block at the upper end of the DPM address range. The absolute address of a handshake register in the Dual-port memory depends on the register width. The following example shows the address assignment:

8 bit register width Host → netX handshake register = BASE_ADDRESS[15:2] + 03h
 netX → Host handshake register = BASE_ADDRESS[15:2] + 02h

16 bit register width Host → netX handshake register = BASE_ADDRESS[15:2] + 02h
 netX → Host handshake register = BASE_ADDRESS[15:2] + 00h

While it does not make sense, to set up 16 Bit Handshake registers with an 8 Bit wide DPM interface, 8 Bit handshake registers may also be used when operating the DPM interface in 16 Bit mode.

Any access of the host to a handshake register address will not be redirected to a netX data memory area, but will be mapped into the corresponding register within the handshake register block. If the 8 bit handshake register mode is selected, any access from host side to memory location at byte 0 and byte 1 of the DWORD aligned handshake pair base address will also be mapped into the handshake registers.

Though each handshake register pair comprises of only one physical 32 Bit register, there are three ways to access the register pairs. There is an address block for netX access, an address block for host access and the standard access path through the DPM. The register addresses for netX access are located between 0x00103500 and 0x0010353F. When accessing the handshake registers via these addresses, only the netX-to-host part of the register is writeable and reading a register automatically clears an appropriate netX side handshake interrupt. The register addresses for host access are located between 0x00103200 and 0x0010323F. When accessing the handshake registers via these addresses, only the host-to-netX part of the register is writeable and reading a register automatically clears an appropriate host side handshake interrupt. It is also this address range that will internally be used, when directly accessing the handshake register pair from host side through the DPM. If for example, handshake register pair 1 has been set up at DPM address 0x2000, an access to 0x2000 will be redirected to address 0x00103504, the host side address for handshake register pair 1.

(of course, the whole host side handshake address block could also be mapped to a DPM memory range. In that case, each handshake register could be accessed through two different DPM addresses).

As both address ranges, netX side and host side, are part of the netX address range, it is possible to also access the host side registers from the netX. This was implemented for debugging purposes and should be used with care to avoid undesired system behaviour (if a host handshake interrupt is pending and the netX reads the corresponding host side register, the interrupt may be cleared before the host had the chance to respond to the interrupt)! As the netX side handshake registers address block could

be mapped into a DPM memory area, the note above also applies to the host side. Through such a mapping, a host could read a handshake register through the netX path instead of the host path (and clear a pending netX interrupt, that was actually supposed to be cleared by the netX).

Due to the full programmability of the Dual-port memory structure, configurations are possible, where the global control block, one (or more) handshake register pair(s) and a data memory block overlap each other. A hardware priority decoder of all data memory blocks and registers will solve such a situation. The following table shows the priority assignment of the memory blocks:

Priority Level	Memory Block Name
highest	Global Control Block
	Handshake Register Pair 0
	:
	Handshake Register Pair 15
lowest	Data Memory Block [7:0]

Dual-port memory data priority level

Note:

There is no default handshake register mapping after a power on reset, hence the desired handshake registers must be configured by netX firmware, before they can be used.

2.11.3.9 Dual-Port Memory Configuration

The Dual-port memory interface configuration is done from netX side by the 'DPM_ARM_IF_CFG0' and 'DPM_ARM_IF_CFG1' registers. For more information about these registers, see the 'netX 500/100 - Program Reference Guide'.

2.11.4 DPM interface signals

The control signals of the DPM interface are user configurable, which allows to connect common host processors to the netX without external glue logic. The following table shows an overview of all interface pins.

Pin Name	Intel	Motorola	Description
DPM_D0-15	D[15:0]	D[15:0]	Dual port memory Data 0-15
DPM_A0	A[0]	A[0] / BE0n / LDSn	Dual port memory Address 0 or Byte Enable 0
DPM_A1-15	A[15:1]	A[15:1]	Dual port memory Address 1-15
DPM_A16-19	A[19:16]	A[19:16]	Dual port memory Address 16-19 (optional)
DPM_SELA12-19	-	-	Inputs for internal address comparator (A12-19)
DPM_ALE	ALE	AS	Dual port memory Address Latch Enable
DPM_CSn	CSn	CSn	Dual port memory Chip Select
DPM_BHEn	BHEn	EN / BE1n / UDSn	Dual port memory High Bus Enable
DPM_RDn	RDn	RD/WRn	Dual port memory Read
DPM_WRLn	WRLn	-	Dual port memory Write Low
DPM_WRHn	WRHn	-	Dual port memory Write High
DPM_RDY	WAITn	TA	Dual port memory Ready
DPM_INT	INT	INT	Dual port memory Interrupt

Host interface pins in DPM mode

DPM_0-15	Bi-directional data bus. The interface supports 8 and 16 bit data busses and can be operated in multiplexed or non multiplexed mode.
DPM_A0-19	Address bus. In multiplexed address mode the lowest 8 or 16 bit lines can be used as programmable input /output signals.
DPM_ALE	Address latch enable or address strobe. In multiplexed data bus mode this signal can be configured as active high / low or positive / negative edge triggered signal.
DPM_CSn	Chip select signal for Dual-port memory access. Instead of the chip select, an internal address comparator can also be used.
DPM_BHEn	Byte high enable signal. Controls the access to the upper byte of the Dual-port data bus.
DPM_RDn	Read signal (Intel type interface) or data direction signal RD/WRn (Motorola type interface).

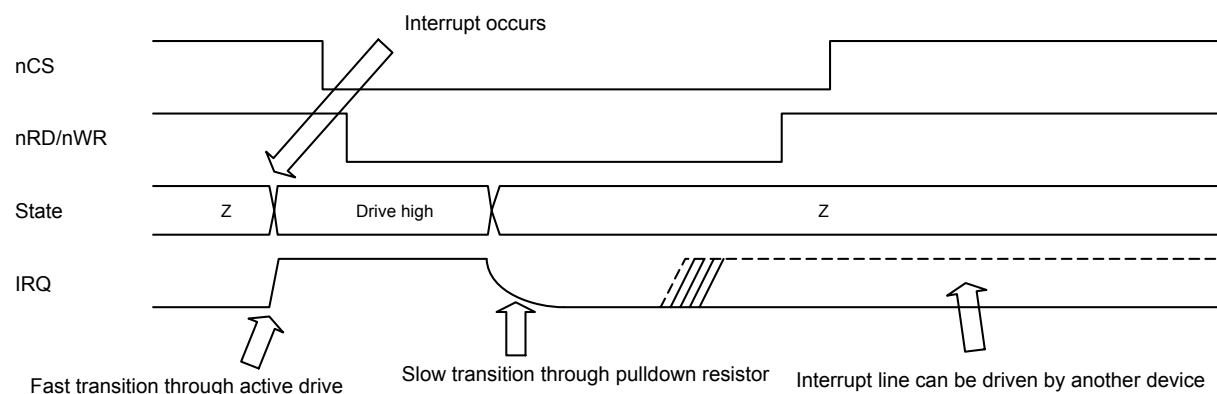
DPM_WRLn DPM_WRHn	Write strobe lines for high and low data byte. For 8 bit modes only the DPM_WRLn line is used
DPM_RDY	The ready signal or transfer-acknowledge signal controls the access end of each cycle to the netX. Several operating modes are programmable. The polarity and the driver type can be configured. It is possible to switch the signal between wait mode (waitstate insertion) and ready mode (transfer ready).
DPM_INT	The interrupt signal is the global interrupt request output to the host system. Several operating modes are programmable like polarity and driver mode.

2.11.5 Interrupts and Interrupt Signal

The electrical characteristic of the interrupt request line from netX to the host system is programmable. There are four different interrupt output modes:

- High Impedance
- Fixed high or low output level
- Push-Pull output, active high or active low
- Open Drain / Open Source output, depending on the interrupt polarity.

After power up, the interrupt pin will be in high impedance state until configured otherwise. When programmed as a normal interrupt output (push / pull), the signal is driven high or low when an interrupt request has occurred. In Open Drain / Open Source configuration it is necessary to connect an external pull-up or pull-down resistor for maintaining a valid (and inactive) signal level when no interrupt is active.



Interrupt signaling for open source output with external pull-down resistor

2.11.5.1 Data Ready Signal

The Data Ready Signal allows the netX to extend the current DPM access cycle until the requested data is available (read) or write data has been accepted. The signal mode and polarity, as well as the drive mode, are programmable:

Signal modes:

- WAIT/BUSY mode, active low (Mode 0)
- WAIT/BUSY mode, active high (Mode 1)
- READY mode, active low (Mode 2)
- READY mode, active high (Mode 3)

Drive modes:

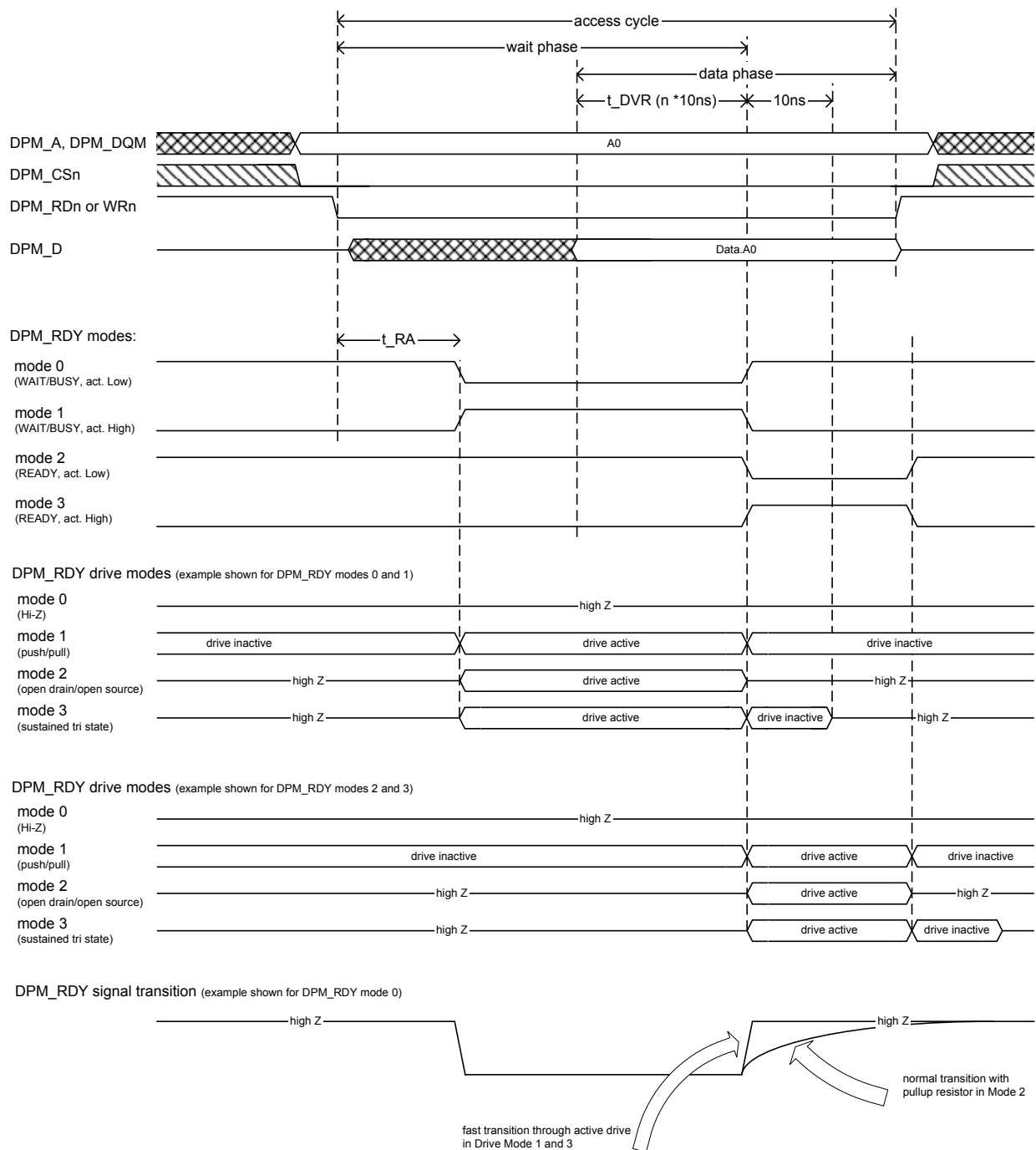
- High Impedance output (Mode 0)
- Push-Pull output (Mode 1)
- Sustained Tristate output (Mode 3)
- Open Drain / Open Source (depending on the configured polarity (Mode 2))

After power up, the DPM_RDY pin will be in high impedance state until configured otherwise. In push-pull mode, the signal is always driven. In Open Drain / Open Source mode the signal will be driven only during its active state, hence it is necessary to use external pull-up or pull-down resistors for maintaining a valid (and inactive) signal level when the DPM_RDY is not active. The sustained tristate output mode works similar to the Open Drain / Open Source output mode, however the signal remains being driven when entering the inactive state for one cycle before entering the high impedance state. This provides a faster signal edge than it could be achieved by a pull-down or pull-up resistor, while the signal may still be shared with other components.

In WAIT/BUSY mode, an active signal means, that the netX DPM is not yet ready and the host needs to extend the access. In READY mode an active signal means that the netX DPM is ready and the host may terminate the access. However, the READY mode is not just a negation of the WAIT mode (see the following diagram for the difference between WAIT/BUSY and READY)

When in WAIT/BUSY mode, the DPM_RDY signal will always become active on a read access, while single write accesses are usually accepted without activating the signal. In READY mode, the signal will be activated on any access.

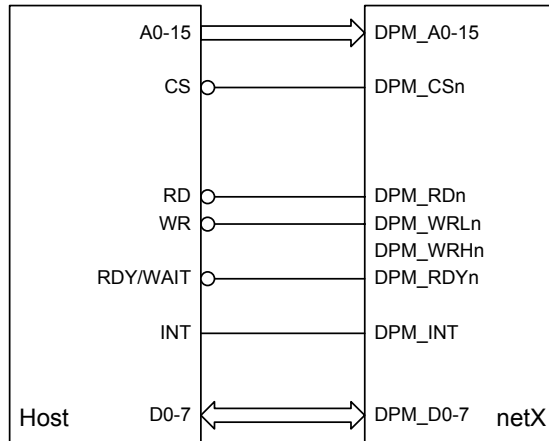
If enabled, an internal counter ensures, that the access time will not exceed 256 μ s, which would occur if an access from host side is mapped into external netX memory which will be always signaling not-ready (e.g. unconfigured or powered down SDRAM). In such cases, the current access will be aborted after 256 μ s, allowing the host processor to end the cycle, preventing system lock up conditions on the host side.



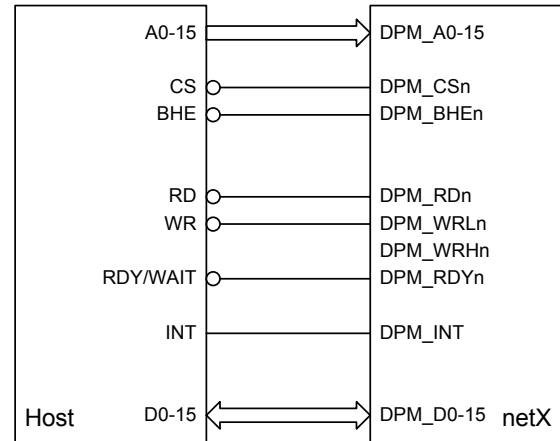
Comparison of different data ready configurations

2.11.5.2 Dual-Port Memory circuits

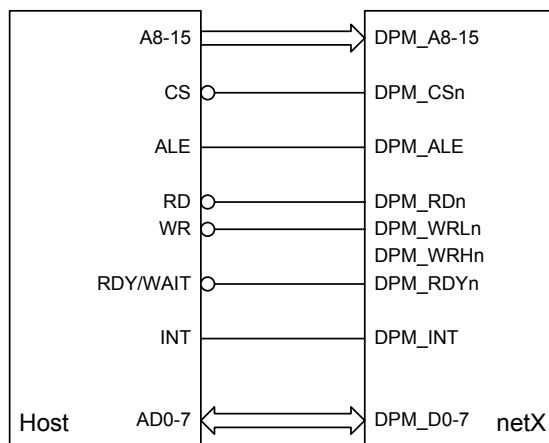
This chapter gives a short view of some standard interface circuits. See the 'netX 500/100 - Reference Program Guide' for default interface configuration register setting.



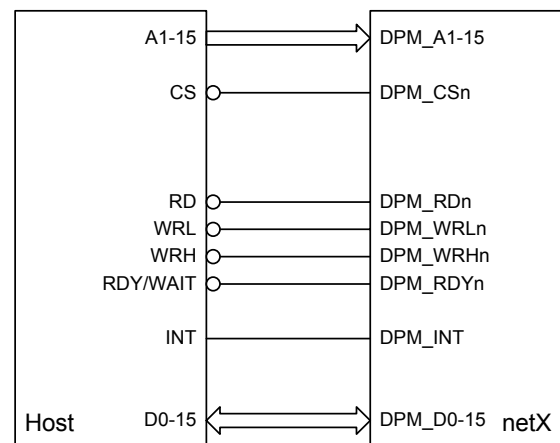
'Intel™ interface', 8 Bit, non multiplexed



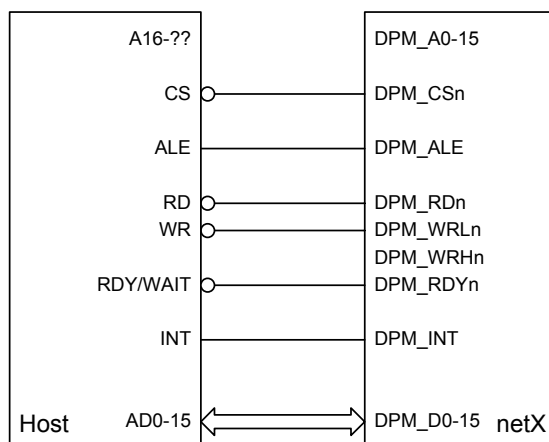
'Intel™ interface', 16 Bit, non multiplexed



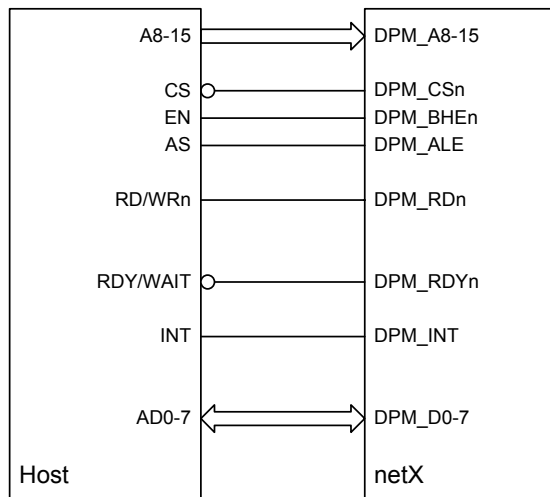
'Intel™ interface', 8 Bit, multiplexed



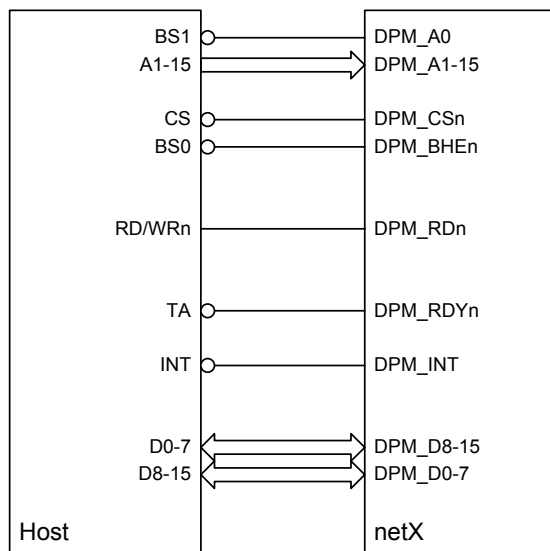
*'Intel™ interface', 16 Bit, non multiplexed,
2 write signals (Low Byte, High Byte)*



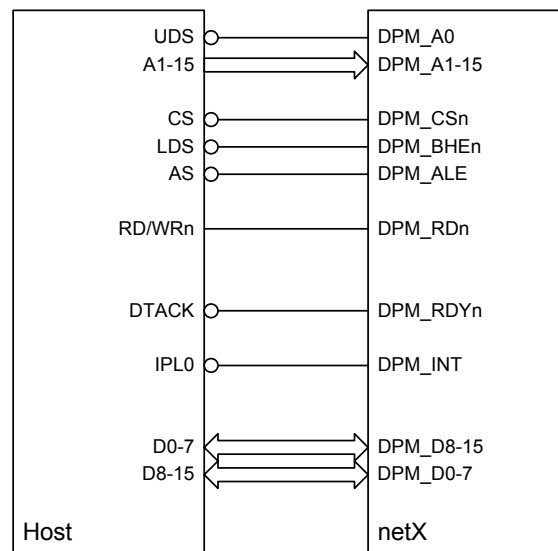
'Intel™ interface', 16 Bit, multiplexed



'Motorola™ interface', 8 Bit, multiplexed



Motorola™ ColdFire, 16 Bit, non multiplexed



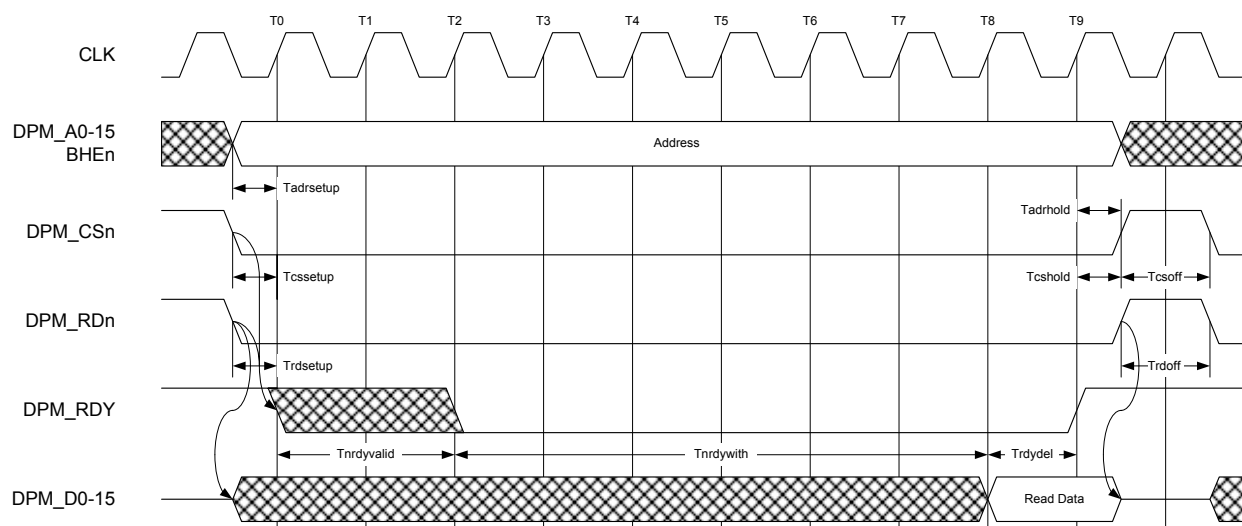
Motorola™ M68000, 16 Bit, non multiplexed

2.11.6 Dual-Port Memory Timing

Due to the virtual nature of the netX Dual-port memory, all DPM accesses are handled by a state machine and data is transferred to and from internal memory, registers, or external devices. The functionality of the state machine is described following.

A Read Cycle is started if the state machine detects a read cycle start sampled with the rising edge of the internal 100 MHz clock. For Intel modes this event could be the signals DPM_CS_n and DPM_RD_n at low level. For Motorola modes the direction and data strobe signals are sampled.

At this time the address at DPM_A0-15 and the signal BHE_n are also latched. Afterwards the state machine runs some clock cycles ($T_{nr\text{dyvalid}}$) for internal synchronization and arbitration until the DPM_RDY signal gets valid, which means the host CPU has to insert wait states before evaluating this signal. Now, some wait cycles ($T_{nr\text{dywidth}}$) are required to fetch the read data from the appropriate memory location. If the requested DPM address is mapped into external memory this time will be higher, depending on the access time of this memory and concurrent accesses to this memory from other system components (ARM, XC). The read data on the DPM_D0-15 lines will become valid with the rising edge of the signal DPM_RDY. In some cases the DPM_RDY signal has to provide a certain setup time for the read data, hence it can be delayed up to three CLK cycles ($T_{r\text{dydel}}$) by configuration. The diagram below shows a delay of one cycle. Sampling one of the signals DPM_CS_n or DPM_RD_n at high level will end the DPM Read Cycle, hence one of these signals has to be high for more than one CLK cycle ($T_{c\text{soff}}$ or $T_{r\text{doff}}$) to allow cycle end detection. The tri-state drivers of the data lines are directly controlled by the DPM_RD_n and DPM_CS_n signal without internal synchronization.

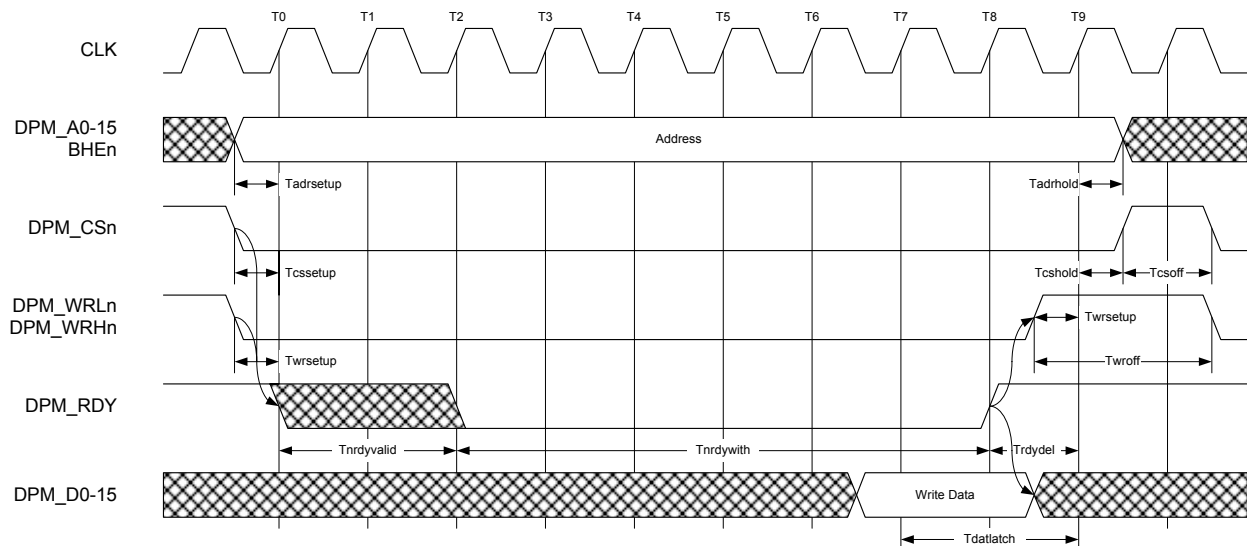


Dual-port memory Read Cycle

Parameter	Description	Value	Dimension
Tadrsetup	Address set up time until internal read cycle starts	0...1	CYC
Tcssetup	Chip select set up time until internal read cycle starts when the DPM_RD _n signal is low too	0...1	CYC
Trdsetup	Read set up time until internal read cycle starts when the DPM_CS _n signal is low too	0...1	CYC
Tnr\text{dyvalid}	Ready valid delay time after read cycle start	2...3	CYC
Tnr\text{dywidth}	Ready width time (see note below!)	4...8	CYC
Trdydel	Ready delay after valid read data. This parameter is configurable.	0...3	CYC
Tcshold	Chip select hold time after DPM_RDY is ready.	0	CYC
Tadrhold	Address hold time after DPM_RDY is ready.	0	CYC
Trdoff	Read off time before next cycle start	> 1	CYC
Tcsoff	Chip select off time Min. time till the internal Read Cycle stops.	> 1	CYC

Note:

The value for the *Tnrdywidth* parameter is valid only for DPM mappings to **internal** RAM. For DPM areas directing to host interface registers add two more cycles. For DPM areas mapped to SDRAM, timing is not predictable, hence when using DPM mappings to SDRAM, it is mandatory to make use of the *DPM_RDY* signal when interfacing the netX DPM to a host processor!

*Dual-port memory Write Cycle*

Parameter	Description	Value	Dimension
Twrsetup	Write set up time until internal write cycle starts when the DPM_CS _n signal is low too	0...1	CYC
Twrhold	Hold time for write data	0	CYC
Twroff	Write off time before next cycle start	> 1	CYC

A DPM Write Cycle works quite similar to the Read Cycle. If two write signals are used for high and low data bytes (DPM_WRL_n and DPM_WRH_n) it must be guaranteed that both signals will become valid within a half system clock cycle. Otherwise the state machine might detect only one write signal at cycle start and initiate a Byte Write instead of a Word Write Cycle.

The write data is continually latched during the write cycle. The host system has to check for the assertion of the DPM_RDY signal to extend the access cycle time. When the DPM_RDY signal has become (or remains) inactive, the write cycle can be finished by deactivation of the write signal. The positive edge write signal edge will eventually start an internal write cycle and the write data which was sampled on the rising write edge is stored.

During single write cycles the “not-ready-time” *T_{nrdywidth}* will typically be two cycles only, because the write data is buffered by the host interface module before it is being transferred to its actual memory location.

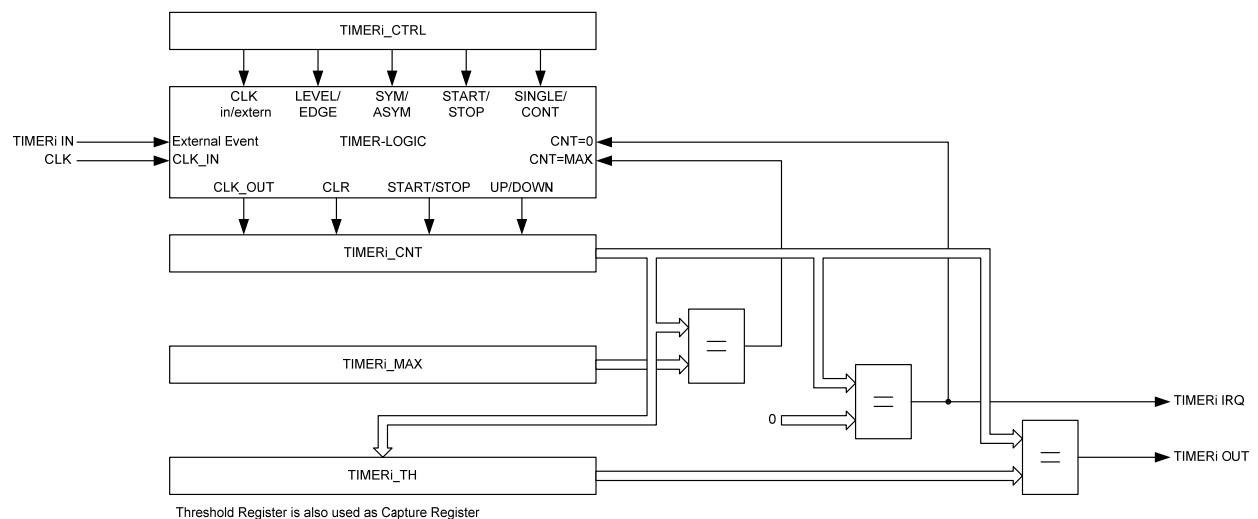
2.12 Timer

The netX provides five 32-Bit Counters which 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 with a threshold value

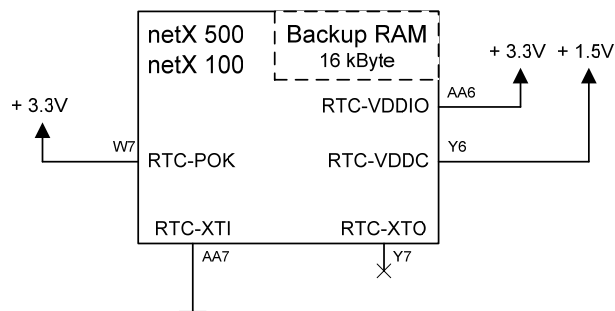
As external events, any GPIO can be assigned. This can be a rising or falling edge respectively a high or low level at the GPIO by setting the inverting bit at the GPIO configuration register.

The counter value can be read and overwritten any time.



Timer function diagram

With netX 100 designs or with netX 500 designs that do not make use of the RTC, the following schematic applies:



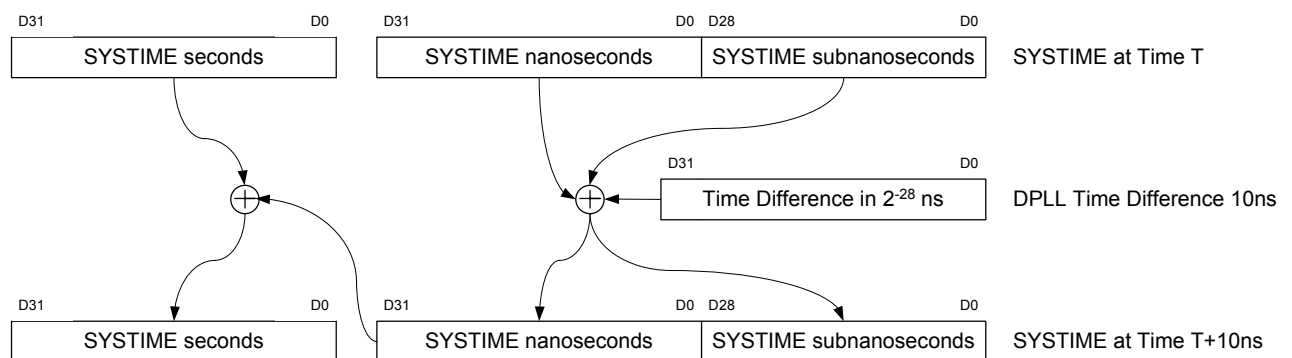
Circuit for netX 100 or for netX 500 when RTC is not used

2.14 IEEE 1588 System Time

The precision System Time derives from a counter, clocked with the 100 MHz system clock and has a resolution of 10 ns from the view of the application. Due to drift, aging or failure of the crystal this time can differ from a system wide master clock which is very often needed in Real-time Ethernet system.

The System Time is not realized by a standard counter but uses an adder which increases the current time value by a programmable number (nominally 10) every clock period. If the 100 MHz clock has a deviation to the master clock then the added value will slightly differ of 10 with a resolution of 2^{-28} ns to compensate the deviation. This can be calculated based on the protocol of IEEE 1588 or other Real-time Ethernet functions.

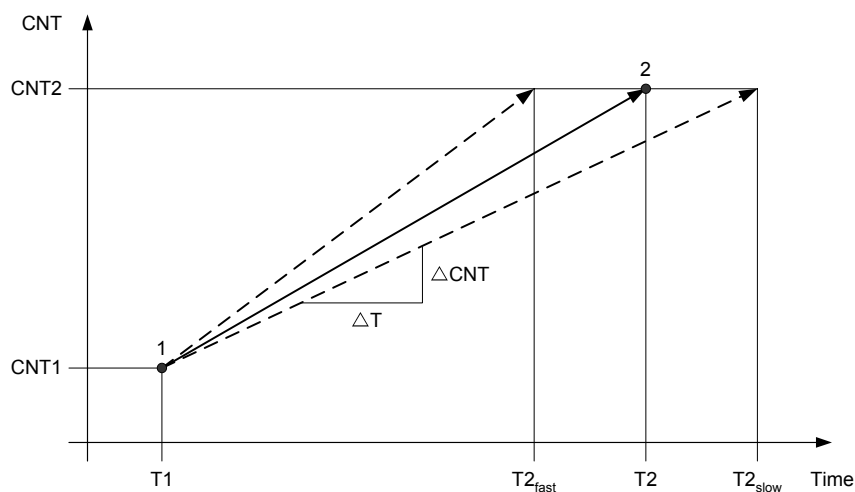
The System Time is provided in two 32-Bit registers. One register represents the seconds and the other represents the nanoseconds from time zero. The application has to read the seconds value first, because this will freeze the nanoseconds register to get a consistent System Time.



Calculation of the System Time

The following diagram shows how the time clock compensation works. With a clock period of $\Delta T = 10$ ns the value $\Delta \text{CNT} = 10$ will be added continuously to the System Time CNT1 to reach CNT 2 exactly at T2. If the clock runs too fast CNT2 will be reached after $T2_{\text{fast}}$ or if the clock is too slow, CNT2 is reached after $T2_{\text{slow}}$.

If ΔCNT is calculated exactly then CNT2 will be reached at T2. The procedure of an ongoing correction prevents the problems of a one step correction resulting in a large step of the System Time.



Ongoing correction of time failure

2.15 JTAG Debug Interface

2.15.1 Standard JTAG connector

The netX Debug Interface is based on the Joint Test Action Group (JTAG) IEEE Standard 1149.1 and supports debugging tools, compliant with this standard. It provides two different modes of operation: **ARM Debug mode** and **Boundary Scan mode**. By default, the netX JTAG interface operates in ARM Debug mode, passing all JTAG signals to the integrated ARM CPU. See chapter 2.15.3 for information, on how to activate the Boundary Scan mode.

The JTAG connector is a 20 pin Insulation Displacement Connector (IDC) keyed box header (2.54 mm male) that matches with IDC sockets mounted on a ribbon cable and provides the following signals:

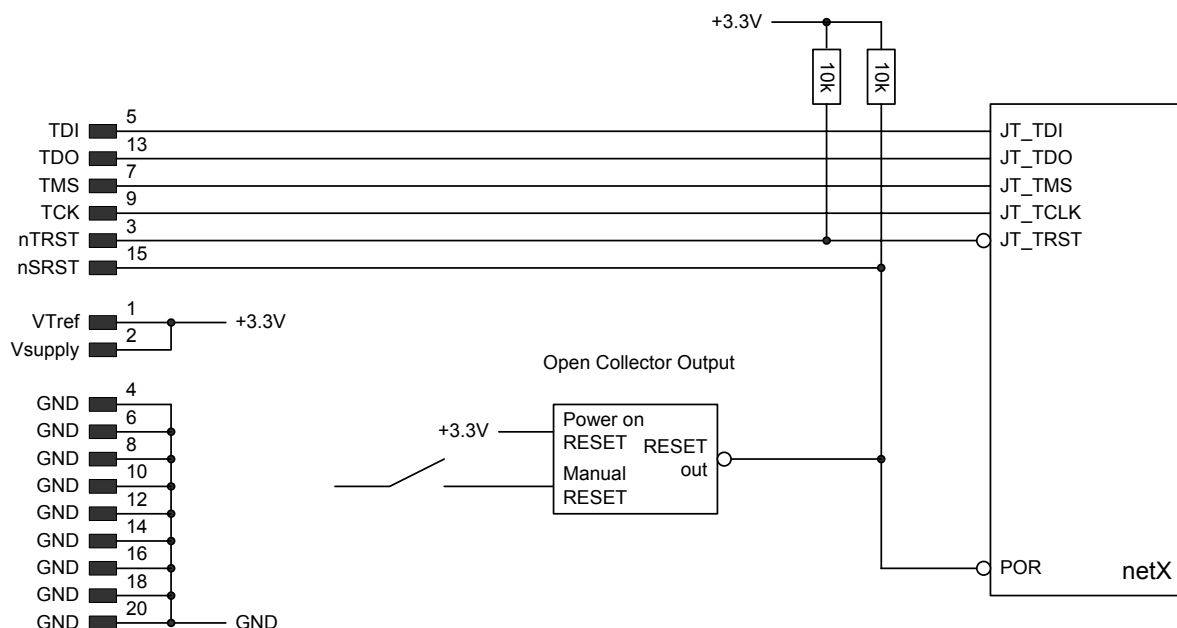
Pin	ARM Signals	netX Signals	Pin	ARM Signals	netX Signals
1	VTref	+3.3V	2	Vsupply	+3.3V
3	nTRST	JT_TRSTn	4	GND	VSS
5	TDI	JT_TDI	6	GND	VSS
7	TMS	JT_TMS	8	GND	VSS
9	TCK	JT_TCK	10	GND	VSS
11	RTCK	Not used	12	GND	VSS
13	TDO	JT_TDO	14	GND	VSS
15	nSRST	PORn	16	GND	VSS
17	DBGRRQ	Not used	18	GND	VSS
19	DBGACK	Not used	20	GND	VSS

Two different reset signals are involved when using the JTAG interface of the netX:

nSRST Open collector output from the debugger to the target system reset. Also input to the debugger, allowing a reset initiated on the target to be reported to the debugger.

nTRST Open collector output from the debugger to the reset signal on the netX JTAG port.

The target board must include pull-up resistors on both reset signals.



JTAG Interface with voltage supervisor chip (i.e. MAX 823)

2.15.2 Hilscher “mini-JTAG” Connector

For netX products with small board size that do not allow implementation of the standard 20 pin JTAG connector, while still requiring access to the JTAG interface, Hilscher has defined an 8 pin JTAG interface port for the connection of Flex Cables. Of course, users are free to use any suitable connector for a JTAG interface in their application, along with a custom cable adaptor. Implementing the Hilscher defined connector however releases the user from defining and building such an adaptor, as there is already an appropriate adapter for the “mini-JTAG” connector available from Hilscher.

This adapter provides a standard 20 pin JTAG connector to be used with common debugger devices and a Flex cable, that connects to the “mini-JTAG” port.

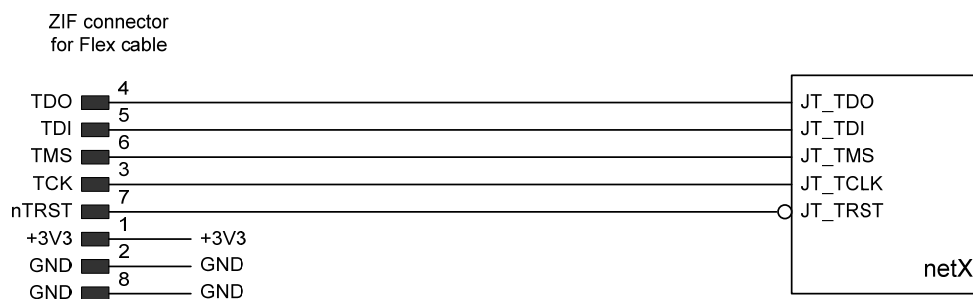
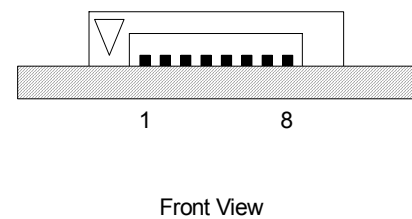
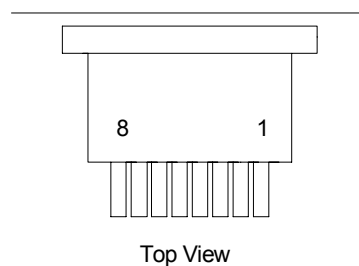
The adaptor also provides the required pull-up resistor on the JT_TRST signal, leaving the netX JTAG interface disabled (in Reset state), achieved through the internal pull-down on the JT_TRST pin, when not connected to the application.

For details on the connector (dimensions, recommended land pattern, etc) please consult the appropriate manufacturer datasheet).

Connector	Signals	Manufacturer	Product
ZIF Flex cable connector, vertical	8	JST	08FLT-SM1-TB
ZIF Flex cable connector, horizontal	8	JST	08FLZ-RSM1-TB

Connector Pinout:

Pin	Signal name
1	+3V3
2	GND
3	JT_TCK
4	JT_TDO
5	JT_TDI
6	JT_TMS
7	JT_TRST
8	GND



Note:

The “mini-JTAG” connector does not allow the debugger to reset the netX (Target Reset), as the appropriate signal (PORN) is not available on the connector! This means, that resetting and stopping the target before the bootloader has been started is not possible with this solution. Further, some debuggers have problems to properly connect the target when the PORN signal is not available, hence the use of this solution is only recommended when the design does not provide enough space for a standard JTAG connector or a custom solution that provides the PORN signal!

2.15.3 Boundary Scan mode

Besides the (default) ARM Debug mode, the netX JTAG interface also supports a second mode, allowing the user to run Boundary Scan tests on the netX, by the use of appropriate (third-party-) tools.

To activate the Boundary Scan mode, refer to the following table, indicating the required state of certain netX signals:

Signal	Pin number	State
TEST	K19	high
GPIO14	W13	high
GPIO8	AA15	low
GPIO9	Y15	low
GPIO10	AA14	low
GPIO11	Y14	low

As all above mentioned pins are equipped with internal pull-down resistors (50k), GPIOs 8-11 can be left unconnected, however TEST and GPIO14 need to be pulled high.

Due to the digital nature of Boundary Scan, some (analog-) netX pins can inherently not be controlled by Boundary Scan, while others are involved in the scan mode itself and are hence not accessible either. This applies to **all** power pins (VSS, VDDIO, VDDC), as well as **all** pins of the Ethernet PHYs, **all** pins of the AD-Converters, the USB port, the JTAG port, all test pins (TEST, TMC1, TMC2, TACT_RST), **all** oscillator- and RTC-pins, as well as GPIOs 8-14. For further details please consult the appropriate netX BSDL files, available for download from the Hilscher website.

2.16 Embedded Trace Macrocell ETM

The ETM is a Real-Time trace module capable of instruction and data tracing. The ETM is an integral part of the ARM microcontroller and works with special debug tools like the Hitex Tool chain for ARM. The ETM comprises the following main components:

Trace port Output signals that help to understand the operation of the processor.

Triggering and filtering facilities

An extensible specification enables to control tracing by specifying the exact set of triggering and filtering resources required for particular application. Resources include address comparators and data comparators, counters and sequencers.

The netX contains the ETM9 Rev 2a (ETM Architecture ETMv1.3) in medium configuration.

The connector for the ETM is standardized by ARM. Further information is available at www.arm.com. It is highly recommended to implement accordingly otherwise the debug tools will not work correctly.

The following table shows the pin assignment of the 38-pol. AMP Mictor connector 2-767004-2 as defined by ARM.

Pin	ARM Signals	netX Signals		Pin	ARM Signals	netX Signals
1	Nc			2	Nc	
3	Nc			4	Nc	
5	GND	VSS		6	TRACECLK	ETM_TCLK
7	DBGREQ	ETM_DRQ		8	DBGACK	ETM_DACK
9	nSRST	Not used		10	EXTTRIG	
11	TDO	JT_TDO		12	VTRef	VCCIO
13	RTCK	Not used		14	VCC	VCCIO
15	TCK	JT_TCLK		16	TRACEPKT[7]	ETM_TPKT07
17	TMS	JT_TMS		18	TRACEPKT[6]	ETM_TPKT06
19	TDI	JT_TDI		20	TRACEPKT[5]	ETM_TPKT05
21	nTRST	JT_TRSTn		22	TRACEPKT[4]	ETM_TPKT04
23	TRACEPKT[15]	ETM_TPKT15		24	TRACEPKT[3]	ETM_TPKT03
25	TRACEPKT[14]	ETM_TPKT14		26	TRACEPKT[2]	ETM_TPKT02
27	TRACEPKT[13]	ETM_TPKT13		28	TRACEPKT[1]	ETM_TPKT01
29	TRACEPKT[12]	ETM_TPKT12		30	TRACEPKT[0]	ETM_TPKT00
31	TRACEPKT[11]	ETM_TPKT11		32	TRACESYNC	ETM_TSYNC
33	TRACEPKT[10]	ETM_TPKT10		34	PIPESTAT[2]	ETM_PSTAT2
35	TRACEPKT[9]	ETM_TPKT09		36	PIPESTAT[1]	ETM_PSTAT1
37	TRACEPKT[8]	ETM_TPKT08		38	PIPESTAT[0]	ETM_PSTAT0

Note:

The AMP Mictor connector has four additional through-hole-pins in the center which have to be grounded for proper operation of the trace port!

For the PCB layout it is recommended to have the lines for the ETM signals as short as possible (the signal delay should be < 100ps). The length of the lines should be equal to avoid different signal delays. To improve signal quality, matching resistors can be placed in the signal lines (located as close as possible to the chip pins (<10mm)) to match the output impedance of the chip signal driver with the PCB trace impedance.

2.17 Vectored Interrupt Controller

The Vectored Interrupt Controller (VIC) supports 32 interrupt sources, whereas 16 can be vectored. The interrupt priority and the type of interrupt, IRQ or Fast IRQ, are configurable. All Interrupts can be masked.

Some of the Interrupts represent the result of a logical OR of up to 32 single interrupts of a function block. Hence the Interrupt service routine may have to check further registers to determine the actual source of an interrupt (e.g. resolving the GPIO interrupt to the GPIO input that caused it).

The following table shows the different interrupt sources:

Interrupt	Source	Standard Use	Remark
0	Reserved for Software Interrupt		ARM standard configuration
1	Timer / Counter 0	Real-time operating system timer	Timer / counter interrupt from GPIO module
2	Timer / Counter 1		Timer / counter interrupt from GPIO module
3	Timer / Counter 2		Timer / counter interrupt from GPIO module
4	System Time nanoseconds compare		Configurable, e.g. '1-second-IRQ'
5	System Time seconds compare	Windows CE	Configurable, e.g. '1-day-IRQ'
6	GPIO15		External interrupt from GPIO15
7	Watchdog		Watchdog expired
8	UART 0	general diagnostic port	
9	UART 1		
10	UART 2		
11	USB		USB Interface
12	SPI		SPI Interface
13	I2C		<i>Not yet implemented</i>
14	LCD		LCD Controller
15	HOST		Dual port memory and Extension Bus
16	GPIO		GPIO 0-14
17	XPEC0		Communication channel 0 / XP_IRQ(11:0)
18	XPEC1		Communication channel 1 / XP_IRQ(11:0)
19	XPEC2		Communication channel 2 / XP_IRQ(11:0)
20	XPEC3		Communication channel 3 / XP_IRQ(11:0)
21	SYNC0		Motion synch. channel 0 / XP_IRQ(15:12)
22	SYNC1		Motion synch. channel 1 / XP_IRQ(15:12)
23	SYNC2		Motion synch. channel 2 / XP_IRQ(15:12)
24	SYNC3		Motion synch. channel 3 / XP_IRQ(15:12)
25	PHY0 / PHY1		Internal Phy0 or Phy1
26	RTC_POK		Power fail
27	<i>n/a</i>		<i>Reserved</i>
28	<i>n/a</i>		<i>Reserved</i>
29	Timer / Counter 3		Timer / counter interrupt from GPIO module
30	Timer / Counter 4		Timer / counter interrupt from GPIO module
31	<i>n/a</i>		<i>Reserved</i>

IRQ Table, netX 500

The Vectored Interrupt Controller (VIC) provides a software interface to the interrupt system. In an ARM system, two levels of interrupts are available:

- Fast Interrupt Request (FIQ) for fast, low latency interrupt handling
- Interrupt Request (IRQ) for more general interrupts.

Generally, only one single FIQ source is used at a time in a system, to provide a true low-latency interrupt. This has the following benefits:

- The interrupt service routine can be executed directly without having to determine the source of the interrupt.
- Interrupt latency is reduced. The banked registers available for FIQ interrupts can be used more efficiently, because a context save is not required.

There are 32 interrupt lines. The VIC uses one bit position for each different interrupt source. The software can control each request line to generate software interrupts.

There are 16 vectored interrupts. These interrupts can only generate an IRQ interrupt. The vectored and non-vectored IRQ interrupts provide an address for an Interrupt Service Routine (ISR). Reading from the vector interrupt address register, VICVectAddr, provides the address of the ISR, and updates the interrupt priority hardware that masks out the current and any lower priority interrupt requests. Writing to the VICVectAddr register, indicates to the interrupt priority hardware that the current interrupt is serviced, allowing lower priority interrupts to become active.

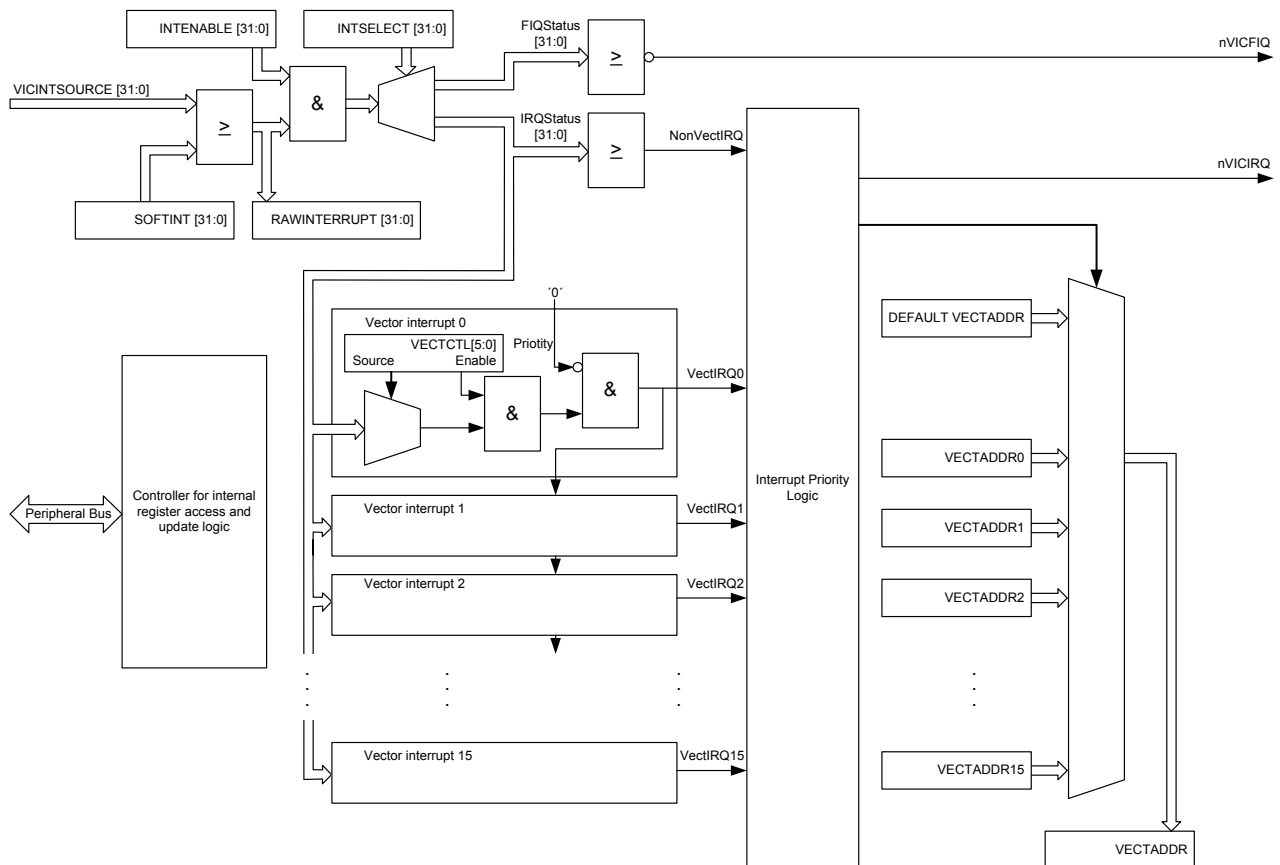
The FIQ interrupt has the highest priority, followed by interrupt vector 0 to interrupt vector 15. Non-vectored IRQ interrupts have the lowest priority. A programmed interrupt request allows to generate an interrupt under software control. This register is typically used to downgrade an FIQ interrupt to an IRQ interrupt.

The block diagram on the following page shows an overview of the VIC.

Note:

The priority of the FIQ over IRQ is set by the ARM. The VIC can raise both an FIQ and an IRQ at the same time.

The VIC is compatible to ARMPrimeCell VIC (PL190), hence appropriate documentation from ARM should also be consulted.



Block diagram of the Vectored Interrupt Controller

2.17.1 Interrupt generation

Interrupt request generation

For generation of FIQStatus[31:0] and IRQStatus[31:0], the interrupt requests from the peripherals are received and combined with the software interrupt requests. Then any undesired interrupt requests are masked out and the results are either routed to FIQStatus[31:0] or IRQStatus[31:0] (see block diagram).

Non-vectored FIQ interrupt (nVICFIQ) generation

By combining FIQStatus[31:0] (see block diagram) the non-vectored FIQ (nVICFIQ) which is connected to the ARM CPU, is generated.

Non-vectored IRQ interrupt generation

By combining IRQStatus[31:0] (see block diagram) the non-vectored IRQ is generated. This signal is used as input of the Interrupt Priority Logic.

Vectored interrupt generation

There are 16 vectored interrupt blocks which generate 16 vectored interrupt signals (VectIRQ0-15). The vectored interrupt blocks receive the IRQStatus[31:0] (Interrupt Requests) and set the VectIRQx if the following conditions are met:

- the selected interrupt is active
- the selected interrupt is currently highest requesting interrupt
- the selected interrupt is enabled in the vector control register (VICIntCntl[0-15])

Software interrupts

The software can control the source interrupt lines to generate software interrupts. These interrupts are generated before interrupt masking, in the same way as external source interrupts. Software interrupts are cleared by writing to the software interrupt clear register, VICSoftIntClear (see Software interrupt clear register, VICSoftIntClear). This is normally done at the end of the interrupt service routine.

2.17.2 Interrupt priority logic

The interrupt priority block prioritizes the following requests:

- Non-vectorized interrupt requests
- vectored interrupt requests

The highest-priority request generates an IRQ interrupt if the interrupt is not currently being serviced. The FIQ interrupt has the highest priority (outside the Interrupt priority logic), followed by interrupt vector 0 to interrupt vector 15. non-vectorized IRQ interrupts have the lowest priority.

2.17.3 Interrupt flow sequence

Vectored interrupt flow sequence:

The following procedure shows the sequence for the vectored interrupt flow:

- An interrupt occurs.
- The ARM processor branches to either the IRQ or FIQ interrupt vector.
- If the interrupt is an IRQ, read the VICVectAddr register and branch to the interrupt service routine. This can be done using an LDR PC instruction. Reading the VICVectorAddr register updates the hardware priority register of the interrupt controller.
- Stack the workspace so that IRQ interrupts can be re-enabled.
- Enable the IRQ interrupts so that a higher priority can be serviced.
- Execute the Interrupt Service Routine (ISR).
- Clear the requesting interrupt in the peripheral, or write to the VICSoftIntClear register if the request was generated by a software interrupt.
- Disable the interrupts and restore the workspace.
- Write to the VICVectAddr register. This clears the respective interrupt in the internal interrupt priority hardware.
- Return from the interrupt. This re-enables the interrupts.

Simple interrupt flow:

The following procedure shows how you can use the interrupt controller without using vectored interrupts or the interrupt priority hardware. For example, you can use it for debugging.

- An interrupt occurs.
- Branch to IRQ or FIQ interrupt vector.
- Branch to the interrupt handler.
- Interrogate the VICIRQ Status register to determine which source generated the interrupt, and prioritize the interrupts if there are multiple active interrupt sources. This takes a number of instructions to compute.
- Branch to the correct ISR.
- Execute the ISR.
- Clear the interrupt. If the request was generated by a software interrupt, the VICSoftIntClear register must be written to. Check the VICIRQ Status register to ensure that no other interrupt is active. If there is an active request go to Step 4 (Interrogate ...).
- Return from the interrupt.

Note:

If the simple flow is used, you must not read or write to the VICVectorAddr register.

2.18 UART

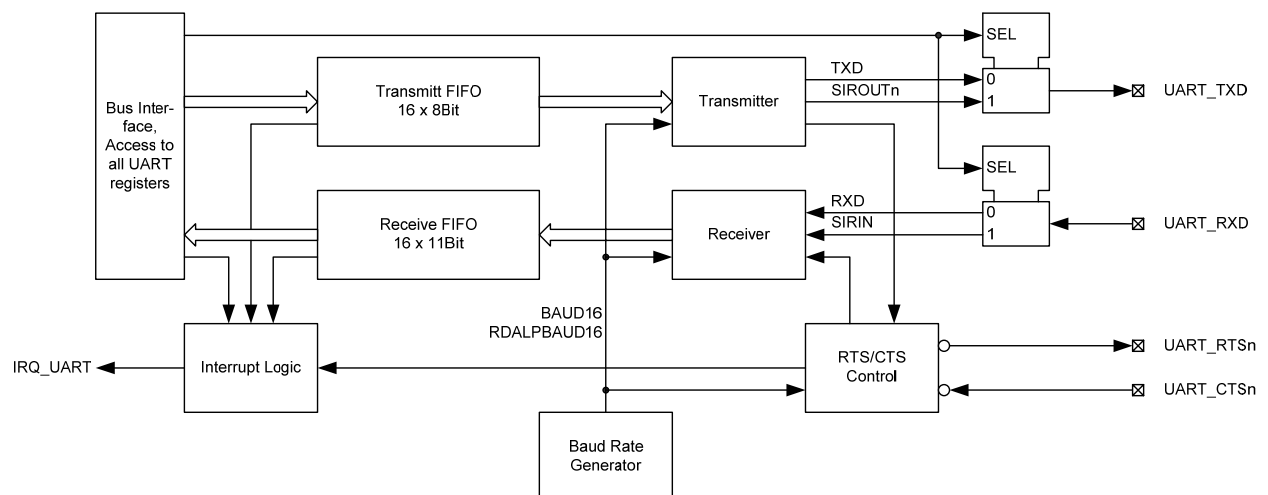
The three UARTs are 16550-compliant with 16 bytes transmit and receive FIFOs.

They can be configured to support speeds up to 3.125 MBaud. The interface supports configurations of:

- five, six, seven, or eight data-bit transfers
- one or two stop bits
- even, odd, or no parity
- IrDA SIR encoding and decoding

The request-to-send (RTS) and clear-to-send (CTS) modem control signals also are available with the interface for hardware flow control. Special features like stick parity and adjustable FIFO trigger level are implemented. UART0 is commonly used as diagnostic port, it is hence not recommended to use this port for other purposes, especially when using loadable firmware from Hilscher. The UARTs are shared with the General Purpose IOs and can be individually configured for UART or GPIO mode.

Attention! When UART0 is not connected and serial boot mode (via USB) is to be used with the application, please consider the notes in chapter 2.3 System LED and Boot Options!



Block diagram of the UART

The ARM CPU reads and writes data and control/status information via the peripheral bus interface. The UART module can generate four individually-maskable interrupts which are combined to a single interrupt so that the output is asserted if any of the individual interrupts are asserted and unmasked.

If a framing, parity or break error occurs during reception, the appropriate error bit is set, and is stored in the FIFO. If an overrun condition occurs, the overrun register bit is set immediately and FIFO data is prevented from being overwritten.

Baud rate generator

The baud rate generator contains free-running counters which generate the internal Baud16 or IrLP-Baud16 signal. Baud16 or IrLPBaud16 provide timing information for UART transmit and receive control. Baud16 is a stream of pulses with a width of 10 ns and a frequency of sixteen times the baud rate.

Transmit FIFO

The transmit FIFO is an 8-bit wide, 16-bit depth, first-in, first-out memory buffer. CPU data written across the bus interface is stored in the FIFO until read out by the transmit logic. The transmit FIFO can be disabled to act as a one-byte holding register.

Receive FIFO

The receive FIFO is an 11-bit wide, 16-bit depth, first-in, first-out memory buffer. Received data, and corresponding error bits, are stored in the receive FIFO by the receive logic until read out by the CPU across the bus interface. The FIFO can be disabled to act as a one-byte holding register.

Transmitter

The transmit logic performs parallel-to-serial conversion on the data read from the transmit FIFO. Control logic outputs the serial bit stream begins with a start bit, data bits, least significant bit (LSB) first, followed by parity bit, and then stop bits according to the programmed configuration in control registers.

Receiver

The receive logic performs serial-to-parallel conversion on the received bit stream after a valid start pulse has been detected. Parity, frame error checking and line break detection are also performed, and the data with associated parity, framing and break error bits is written to the receive FIFO.

Interrupt logic

Four individual maskable active HIGH interrupts are generated in the UART module and are combined to one interrupt output. This output is generated as an OR function of the individual interrupt requests. The single combined interrupt is used with the system interrupt controller that provides another level of masking on a per-peripheral basis. This allows use of modular device drivers which will always know where to find the interrupt source control register bits.

IrDA SIR Endec

The Transmitter and Receiver block contain an IrDA SIR protocol Endec. The SIR protocol Endec can be enabled for serial communication via signals nSIROUT and SIRIN to an infrared transducer instead of using the signals TXD and RXD. The SIR protocol Endec can both receive and transmit, but it is half-duplex only, so it cannot receive while transmitting, or vice versa.

The SIR transmit encoder modulates the Non Return-to-Zero (NRZ) transmit bit stream. The IrDA SIR physical layer specifies use of a Return To Zero, Inverted (RZI) modulation scheme which represents logic 0 as an infrared light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared Light Emitting Diode (LED).

In normal mode the transmitted pulse width is specified as three times the period of the internal x16 clock (Baud16), that is, 3 / 16 of a bit period.

In Low-power mode, the transmitted infrared pulse is set to 3 times the period of the internally generated IrLPBaud16 signal. The frequency of IrLPBaud16 signal is set by writing the appropriate divisor value to UARTILPR.

The active low encoder output is normally LOW for the marking state (no light pulse). The encoder outputs a high pulse to generate an infrared light pulse representing a logic 0 or spacing state.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the internal logic. The decoder input is normally HIGH (marking state) in the idle state (the transmit encoder output has the opposite polarity to the decoder input).

A start bit is detected when the decoder input is LOW. Regardless of being in normal or low-power mode, a start bit is deemed valid if the decoder is still LOW, one period of IrLPBaud16 after the LOW was first detected.

UART communication

Data received or transmitted is stored in two 16-byte FIFOs, the receive FIFO has an extra three bits per character for status information.

For transmission, data is written into the transmit FIFO. This causes a data frame to start transmitting with the parameters indicated in UARTLCR. Data continues to be transmitted until there is no data left in the transmit FIFO. The BUSY signal goes HIGH as soon as data is written to the transmit FIFO (that is, the FIFO is non-empty) and remains asserted HIGH while data is being transmitted. BUSY is negated only when the transmit FIFO is empty, and the last character has been transmitted from the shift register, including the stop bits. BUSY can be asserted HIGH even though the UART module may no longer be enabled.

When the receiver is idle (RXD continuously 1, in the marking state) and a LOW is detected on the data input (a start bit has been received), the receive counter, with the clock enabled by Baud16, begins running and data is sampled on the eighth cycle of that counter (half way through a bit period).

The start bit is valid if RXD is still LOW on the eighth cycle of Baud16, otherwise a false start bit is detected and it is ignored.

If the start bit was valid, successive data bits are sampled on every 16th cycle of Baud16 (that is, one bit period later) according to the programmed length of the data characters. The parity bit is then checked if parity mode was enabled.

Lastly, a valid stop bit is confirmed if RXD is HIGH, otherwise a framing error has occurred. When a full word has been received, the data is stored in the receive FIFO, with any error bits associated with that word.

Error bits

The three error bits are stored in bits 10:8 of the receive FIFO, and are associated to a particular character. There is an additional error which indicates an overrun error but it is not associated with a particular character in the receive FIFO. The overrun error is set when the FIFO is full and the next character has been completely received in the shift register. The data in the shift register is overwritten but it is not written into the FIFO.

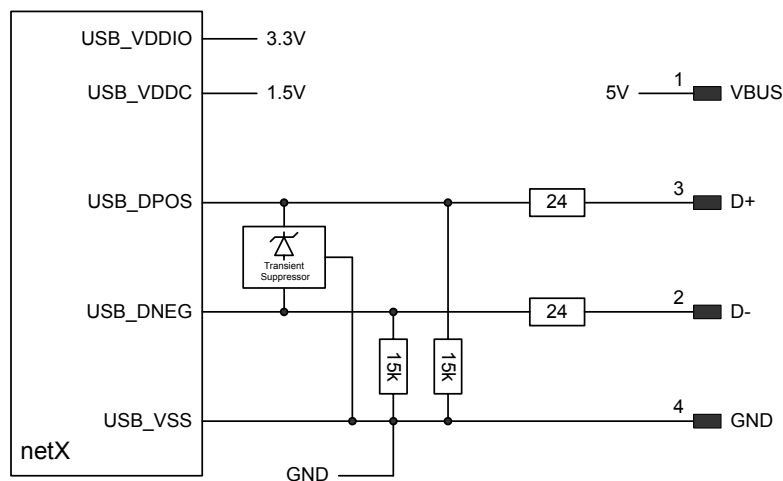
FIFO bits 7:0 : received data
FIFO bit 8 : framing error
FIFO bit 9 : parity error
FIFO bit 10 : break error

Disabling the FIFOs

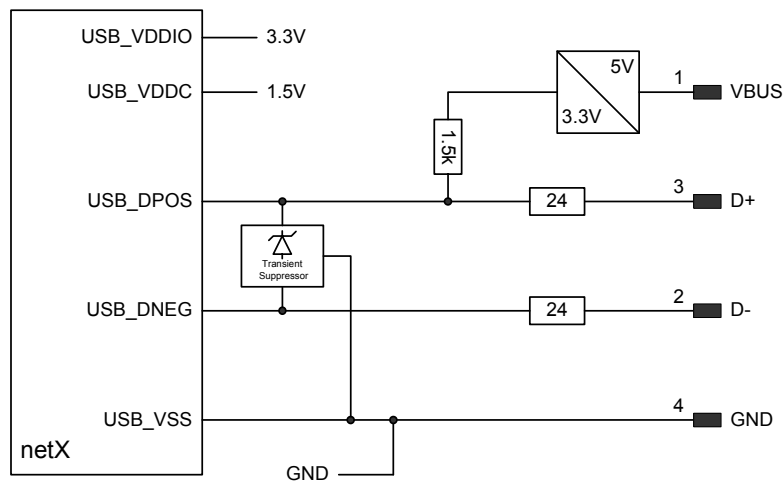
Additionally, it is possible to disable the FIFOs. In this case, transmit and receive sides of the UART module have 1-byte holding registers (the bottom entry of the FIFOs). The overrun bit is set when a word has been received and the previous one was not yet read.

2.19 USB

The integrated USB V 1.1 interface is fully compliant with the USB specification. It supports both full and low speed transfers and can act as a host or device. The USB unit includes an integrated transceiver and provides eight pipes. Their direction, transfer type and FIFO size can be configured at run-time. All low-level USB operations are realized in hardware. The software only has to manage the enumeration process and data transfer from and to the FIFOs.



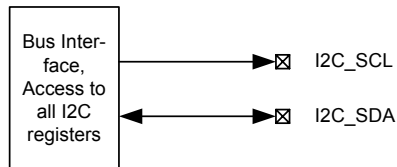
USB Downstream Port / Host with Full Speed 12 Mb/s with Receptacle "A"



USB Upstream Port / Device with Full-Speed 12 Mb/s with Receptacle "B"

2.20 I2C

The I2C Bus Interface Unit provides a general purpose 2-pin serial communication port, interfacing to a wide range of peripherals and memory components. The netX chip always acts as master on this port, which supports 8 different transfer speeds. The maximum clock frequency (I2C_SCL) is 1MHz.



Block diagram of I2C unit

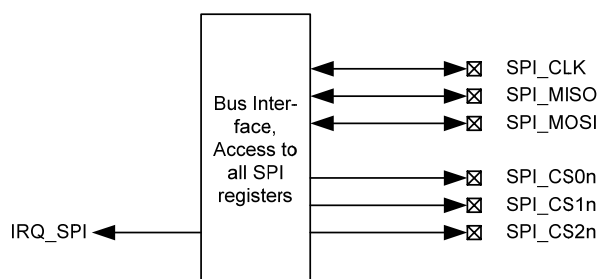
2.21 SPI

Beside the I2C, the SPI is the most common serial interface for peripherals and memory components. It can be also used to connect Smart Media Cards.

The SPI unit supports all four transfer modes with 16 different speeds and can be configured as master or slave. As a master, the maximum speed is 25 MHz. The interface is completely interrupt driven and provides separate 16x16 bit FIFOs for incoming and outgoing data. The filling level of the FIFOs can be read at any time. The IRQ output signal is the disjunction of 7 internal IRQ signals. Each of them is maskable by software and can be read and reset by software.

There are three select signals available to connect up to three different SPI devices without any additional glue logic. These chip select signals can be controlled by either the internal SPI communication state machine or directly by software.

The burst length can be programmed from 1 to 128. Also the burst delay is programmable (0-7 SPI clock cycles).



Block diagram of SPI Unit

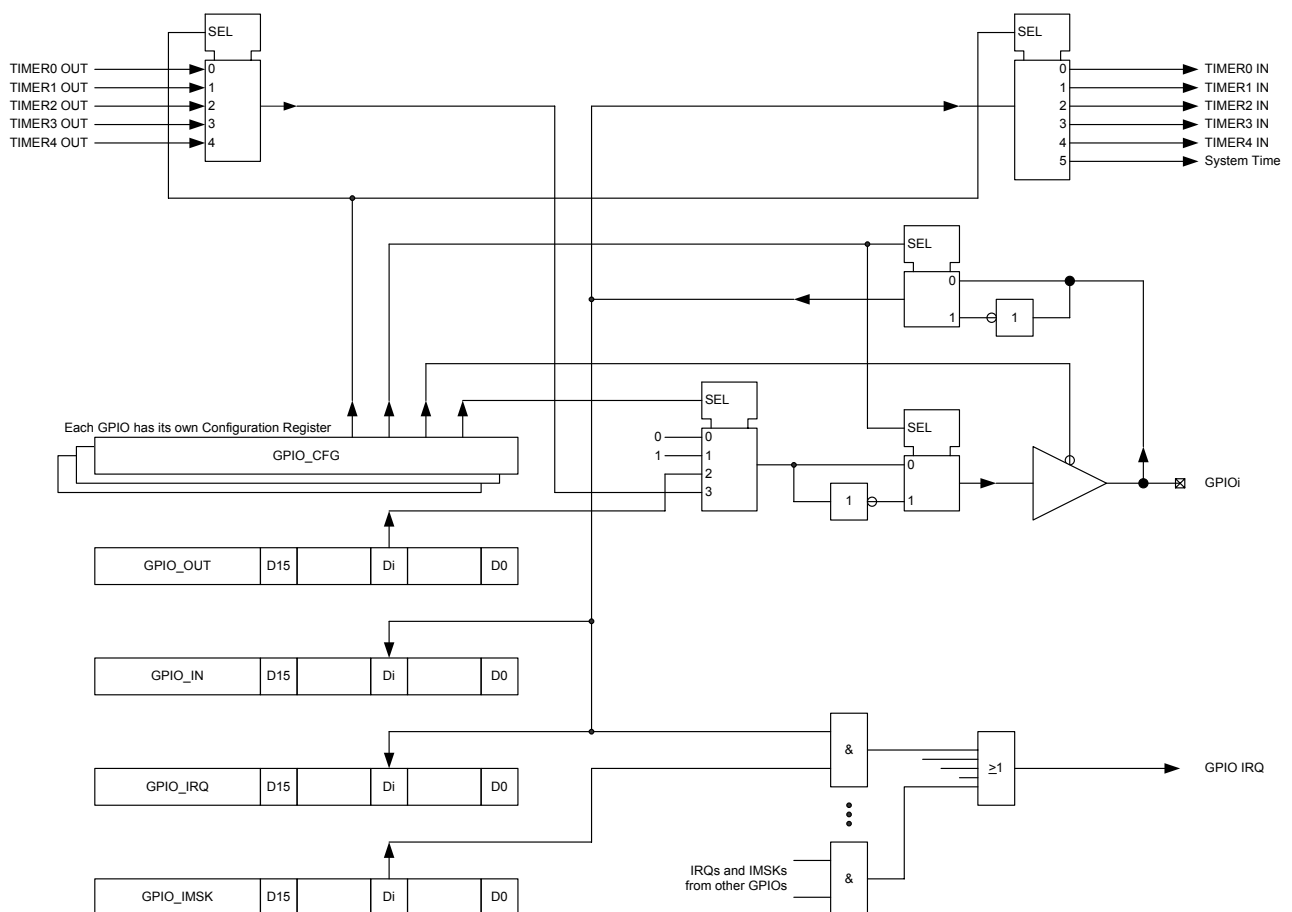
2.22 GPIO

The netX 500/100 provides a total of 32 general purpose IOs, which are shared with the UARTs and are also used as input or output signal along with the internal timers..

The GPIOs provide the following features:

- Can be programmed individually as input or output, inverted or non inverted
- Outputs can be set by individual registers as well as by a common register
- Each GPIO can be assigned to a System Timer, to be used as capture input or PWM output
- Each GPIO can generate an interrupt

Each GPIO has its own configuration register GPIO_CFGi to configure these functions and to read and write the IO individually. All inputs can be read together in the GPIO_IN register, respectively can be written through the GPIO_OUT register.



Block diagram showing GPIO functionality

2.23 PIO

The netX chip contains several programmable input / output lines. Each of the 83 PIO can be used as simple input or output without any additional features. The first 31 PIO pins (PIO0-PIO30) are shared with Motion Control pins, LCD pins and ETM pins. The 53 other Pins (PIO32-PIO84) are shared with Host Interface Pins. (PIO31 does not exist). PIOs 0 – 7 usually drive Fieldbus and RT Ethernet status LEDs in standard applications.

PIO 0-31

The first 31 PIO signals are controlled by three registers: 'PIO_IN - PIO Input Register', 'PIO_OUT - PIO Output Register' and 'PIO_OE - PIO Output Enable Register'.

PIO 32-84 Pins

These programmable input / output pins are multiplexed with the host interface and are hence also referenced as "HIF-PIOs". When the host interface is switched to 'I/O Mode', all pins work as normal input / output lines. When the Extension Bus or Dual-Port memory mode is selected, unused interface lines can be used as programmable input / output pins. It is possible to switch each pin between host interface operation and programmable input / output pin via the mode registers. The following list shows all relevant netX control registers. For a detailed description of all registers see the 'netX 500/100 Program Reference Guide'.

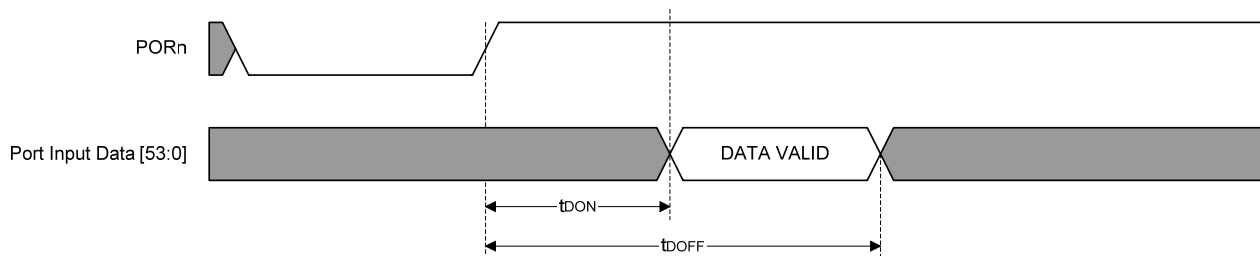
DPM_ARM_IF_CFG0	This register selects the different operating modes for the host interface. The interface may be switched to 'Disable', 'Extension Bus', 'Dual-Port Memory' or 'IO Mode' operation.
DPM_ARM_IO_DATA0 DPM_ARM_IO_DATA1	These data registers set the output data for all HIF-PIOs with enabled output drivers, respectively contain the sampled input data of all HIF-PIOs.
DPM_ARM_IO_DRV_EN0 DPM_ARM_IO_DRV_EN1	The driver enable register controls the output drivers of each pin. When the DPM or Extension Bus function is enabled, the setting of the driver enable bit is ignored. After power on reset all drivers are disabled.
DPMAS_IO_MODE0 DPMAS_IO_MODE1	<p>The mode registers allow to configure each pin to either I/O mode or Host interface mode. When the 'I/O Mode' is selected in Register DPM_ARM_IF_CFG0 (switches all pins to I/O mode), the settings of these mode registers have no effect. The mode of each pin can be changed anytime, also during operation.</p> <p>The IN_CONTROL[1:0] flags control the sampling of the input pin signal. Four modes can be selected.</p> <p>Mode '00' (reset condition) samples all input signal lines shortly after rising edge of the power on reset signal allowing configuration by external pull-up or pull-down resistors. The firmware can read the input data register after starting. In mode '01' the input latches are always enabled and the data is stored in internal flip flops, clocked by the system clock. When mode '1x' is selected the input flip flops are only enabled when the PIO77 pin has a high (mode 11) or low (mode 10) level. Data will be stored in the input flip flops, controlled by the internally synchronized PIO77 pin.</p>

Output Delay of PIO 32-84 Pins

The output register bits are directly connected to the output pins. The output data is always driven synchronous to the system clock. So there is a minimal path delay of approx. 1 system clock.

Power-On-Reset Sampling of PIO 32-84 Pins

After the Power On Reset signal is being released, the state of all 53 host interface pins is sampled and stored in two registers. This provides the possibility to select different device configurations by using (weak) pull-up or pull-down resistors and implementing the corresponding configurations in the firmware. The following figure shows the power on reset schematic of data sampling.



Input sampling during power on reset

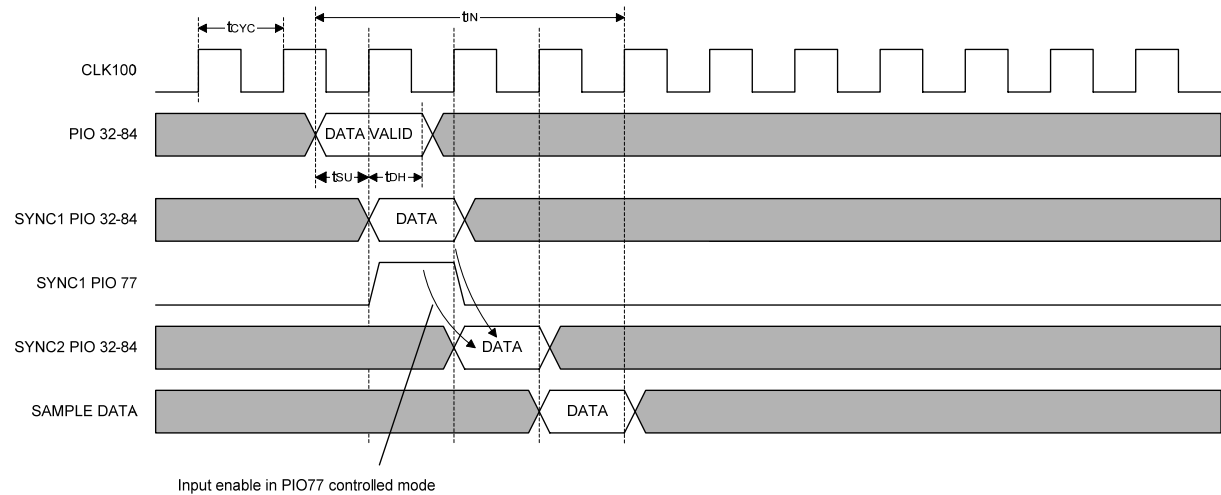
Parameter	Description	Value	Dimension
t_{DON}	Maximum time between deassertion of the Power On Reset Signal and valid (external) data supplied to the PIOs	11	ms
t_{DOFF}	Minimum time between deassertion of the Power On Reset signal and deactivation of (external) data supplied to the PIOs	11.5	ms

Note:

The Timing values and the figure above are only relevant when driving data signals from an external data source instead of using pull-up or pull-down resistors. In that case, users must take care, that the host interface configuration that might be done by the firmware or the boot loader (DPM boot mode or Extension Bus boot mode) will not collide with the data source connected to the PIOs!

Input Sampling of PIO 32-84 Pins

All input pins are synchronized to the internal system clock. Besides continuous data sampling, it is also possible to enable/disable all input flip flops via the PIO77 pin and initiate data sampling. When this external signal is active, the input flip flops are enabled. The signal will be synchronized to the system clock as well. The following figure shows the input signal delay at the two different sampling modes.



Input Sampling and Timing Diagram

Parameter	Description	Value	Dimension
t_{CYC}	Internal 100 MHz clock cycle	10	ns
t_{SU}	Input setup time before internal rising clock edge	< 1	CYC
t_{DH}	Hold time after internal rising clock edge	< 1	CYC
t_{IN}	Synchronization delay for input signals until valid internal sampling data	3...4	CYC

2.24 LCD

The LCD Controller provides all necessary control signals to interface directly to a variety of color and monochrome LCD panels:

- Active matrix TFT panels with up to 18-bit bus interface
- Single-panel monochrome STN panels, 4-bit and 8-bit bus interface
- Dual-panel monochrome STN panels, 4-bit and 8-bit bus interface per panel
- Single-panel color STN panels, 8-bit bus interface
- Dual-panel color STN panels, 8-bit bus interface per panel

The LCD Controller can be programmed to support a wide range of panel resolutions such as:

- 320 x 200
- 320 x 240
- 640 x 200
- 640 x 240
- 640 x 480

It supports 15 gray-level mono, 3375 color STN and 32K color TFT with a programmable timing for the different display panels.

The following table shows which data bits are used for the different types of display.

	STN color single panel	STN color dual panel	STN 4-Bit mono single panel	STN 4-Bit mono dual panel	STN 8-Bit mono dual panel	STN 8-Bit mono dual panel	TFT 18-Bit color
LCD_D0	PD[7]	UPD[7]	PD[3]	UPD[3]	PD[7]	UPD[7]	Intensity Bit
LCD_D1	PD[6]	UPD[6]	PD[2]	UPD[2]	PD[6]	UPD[6]	RED[0]
LCD_D2	PD[5]	UPD[5]	PD[1]	UPD[1]	PD[5]	UPD[5]	RED[1]
LCD_D3	PD[4]	UPD[4]	PD[0]	UPD[0]	PD[4]	UPD[4]	RED[2]
LCD_D4	PD[3]	UPD[3]			PD[3]	UPD[3]	RED[3]
LCD_D5	PD[2]	UPD[2]			PD[2]	UPD[2]	RED[4]
LCD_D6	PD[1]	UPD[1]			PD[1]	UPD[1]	Intensity Bit
LCD_D7	PD[0]	UPD[0]			PD[0]	UPD[0]	GREEN[0]
LCD_D8		LPD[7]		LPD[3]		LPD[7]	GREEN[1]
LCD_D9		LPD[6]		LPD[2]		LPD[6]	GREEN[2]
LCD_D10		LPD[5]		LPD[1]		LPD[5]	GREEN[3]
LCD_D11		LPD[4]		LPD[0]		LPD[4]	GREEN[4]
LCD_D12		LPD[3]				LPD[3]	Intensity Bit
LCD_D13		LPD[2]				LPD[2]	BLUE[0]
LCD_D14		LPD[1]				LPD[1]	BLUE[1]
LCD_D15		LPD[0]				LPD[0]	BLUE[2]
LCD_D16							BLUE[3]
LCD_D17							BLUE[4]

PD Panel data
UPD Upper panel data
LPD Lower panel data

The data bit PD[i] corresponds to the pixel position. Means PD[0] is the leftmost pixel on the panel and PD[7] is the rightmost pixel within the 8-Bit data.

Note:

The LCD Controller is based on the ARM PrimeCell LCD Controller (PL110), hence appropriate documentation from ARM may also be consulted.

2.25 Motion Control Functions

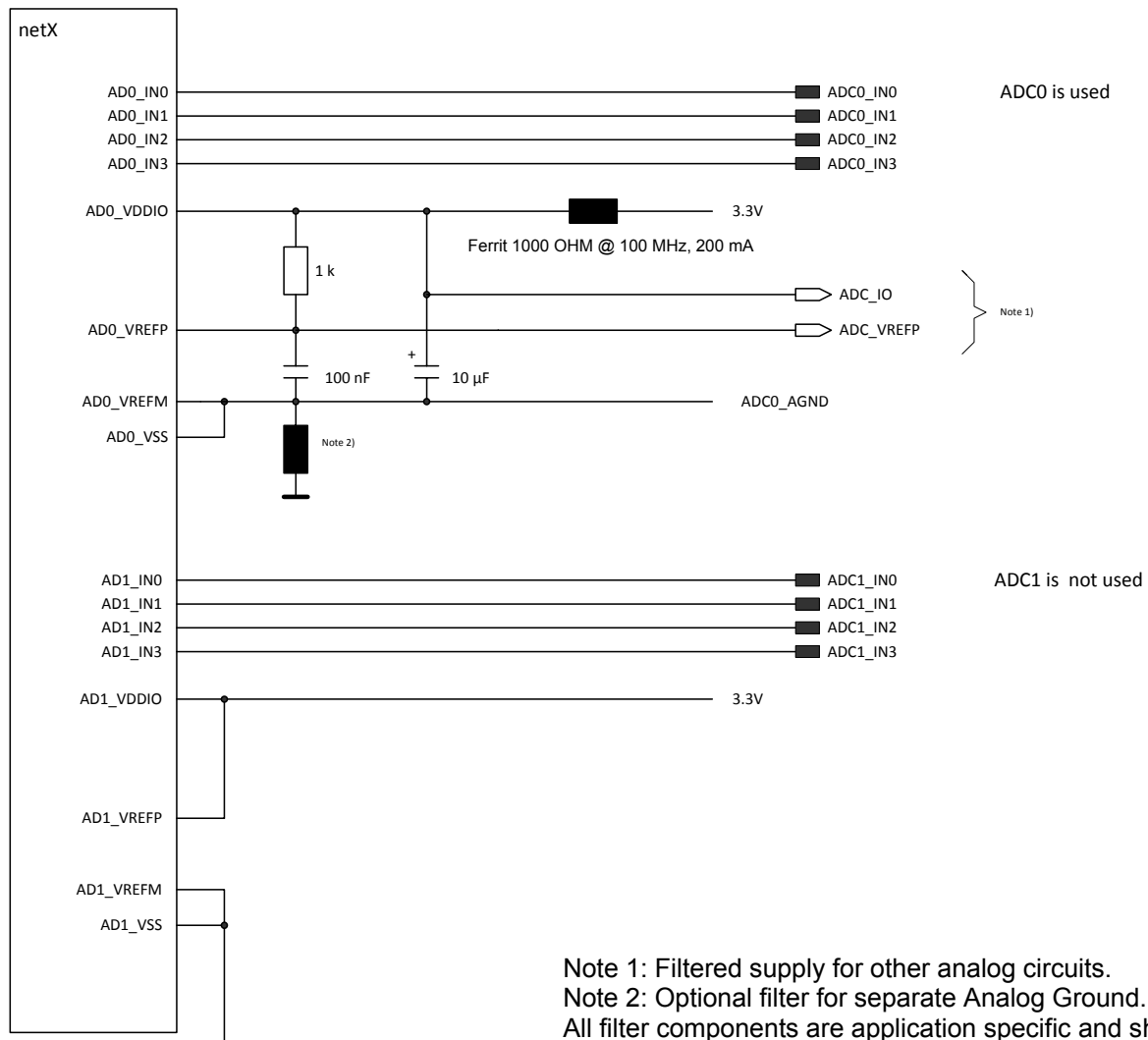
The netX is also designed for Motion Control applications by providing two three phase center aligned Motor PWM units, with an additional simple PWM for a resolver sensor and analog inputs to measure the phase currents or the position of an analog sensor. Additionally, two Encoder interfaces are available.

2.25.1 AD-Converter

The netX contains two 10 bit AD-Converters, each with a four channel input multiplexer and sample & hold unit. Both converters can work simultaneously to measure i.e. the currents of a motor power stage. The maximum sample rate is 1MHz for each AD-Converter.

The ADC unit is controlled by only one register, which has different bit divisions for read and write accesses and is accessible at four different addresses. For performance reasons (pre-processing of the sampled data) it is recommended, to access the ADC via one of the xPEC controllers, however it is also possible, to access the ADC from the ARM CPU. In that case, at least one of the xPECs must be switched off and the remaining xPECs must not access the ADC register!

The following picture shows an example schematic of an ADC application.



2.25.2 Motor PWM

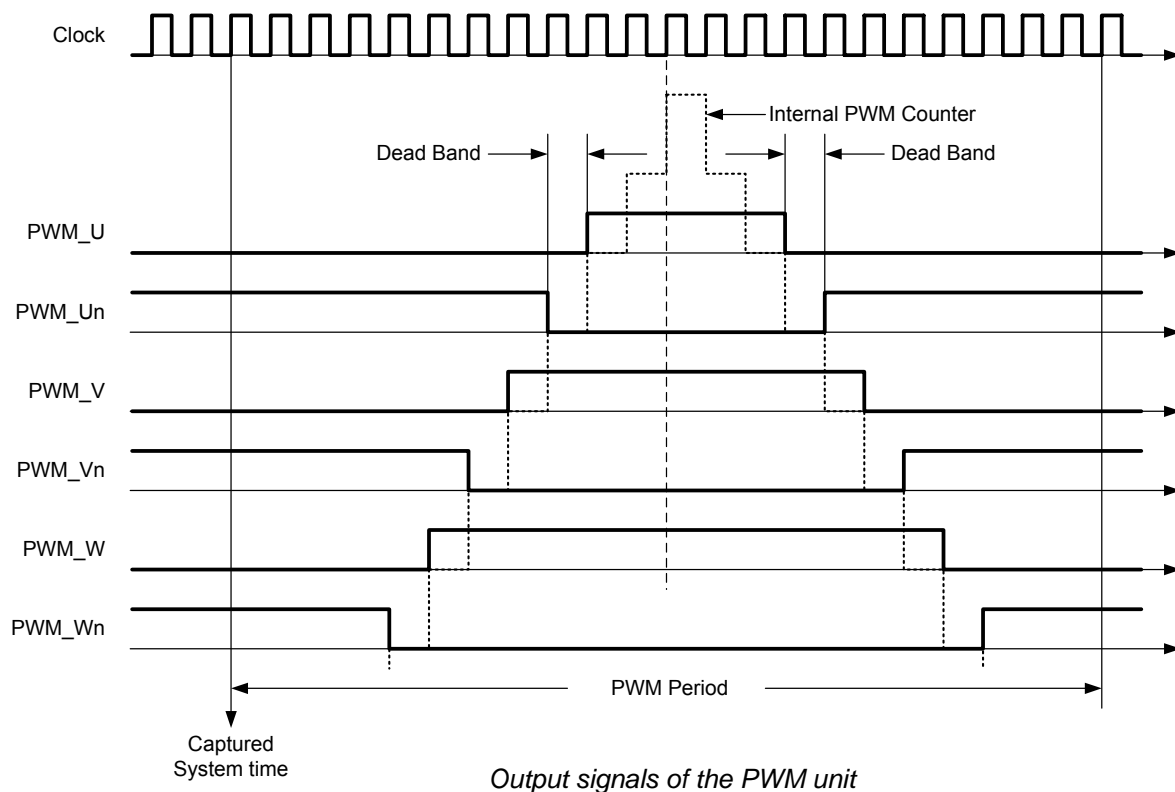
To allow the use of the netX in drive control applications, special PWM logic has been implemented. To get a maximum on flexibility and Real-time performance these functions are directly connected to the communication function blocks (xMACs).

A symmetric 16 Bit counter creates the PWM cycle by counting up till it reaches half of parameter TP and then counting down to zero. The value of this counter is compared with the half of parameters TU, TV and TW, switching the corresponding output U, V and W to low if the parameter is below the value of the counter and high if it is above or equal. Each output also has a corresponding complementary output (Un, Vn and Wn). The PWM signals can for example be used to control the three phase transistor bridge of a motor drive or to control three single phase motors. To avoid short circuit currents at the switching point, the positive edge of each complementary signal pair is delayed for the number of clock cycles configured by parameter TD, creating a dead zone where high side and low side transistors of a bridge are switched off.

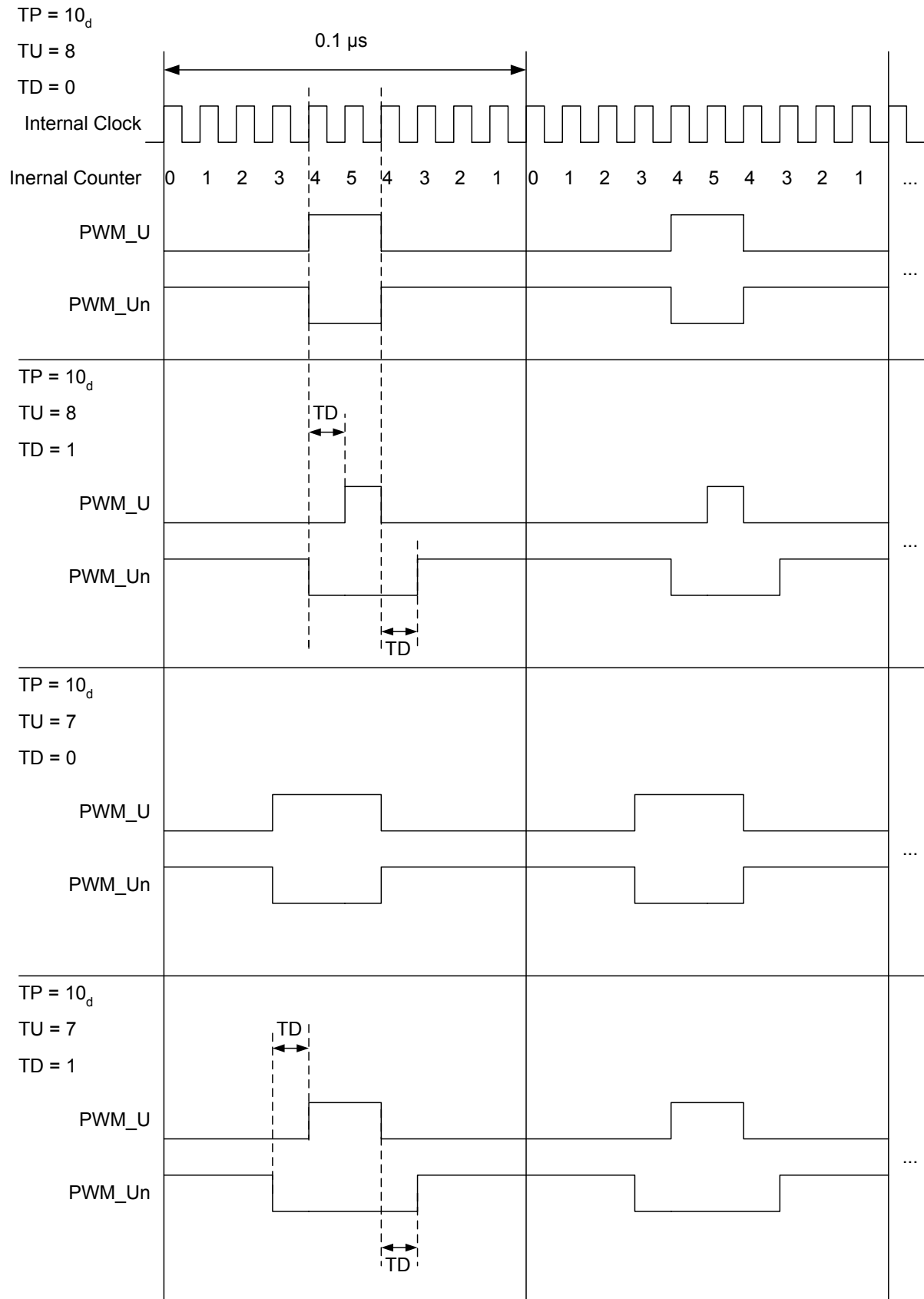
The following diagram refers to even values of TP, TU, TV and TW. An example of an odd value of TU can be found at the end of this chapter.

For safety reasons, the input signal PWM_FAILn was implemented. If this signal is low or left open (internal pull-down), all outputs of the PWM are immediately switched to low level.

The system time is always captured at the beginning of the PWM cycle. This value can be used to implement a digital PLL at XC level, to synchronize the PWM with other processes (e.g. motor current measurement) and control the phase of the PWM.



Examples:



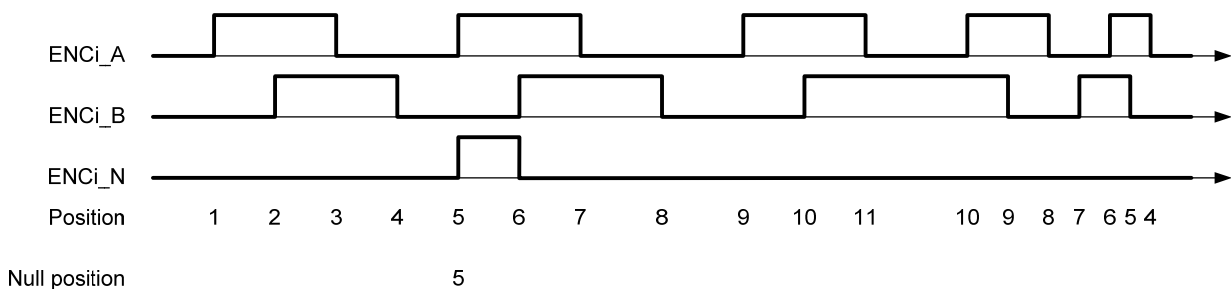
2.25.3 Resolver PWM

To use a resolver as a position sensor, a sine voltage has to be generated and sampled by the AD-converter synchronously. For this purpose, a second PWM unit was implemented, which has only one output signal, generated from a 16 Bit counter with a resolution of 10 ns.

2.25.4 Encoder interface

Two Quadrature Encoder input blocks with the signals A, B and N are available. They detect the rotating direction and count the position using both edges of the encoder signals, quadrupling the resolution of the Encoder signals. The counter direction is configurable.

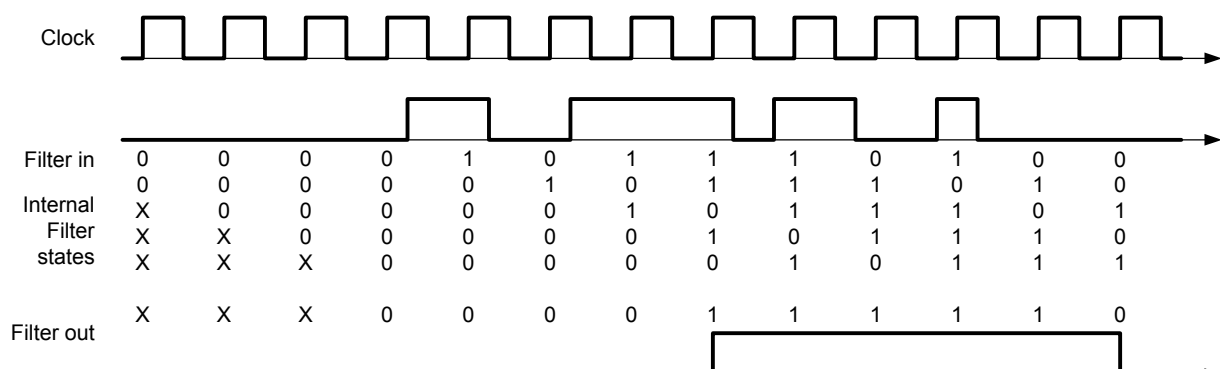
Four different configurable capture registers can either capture the actual position of ENC0 or ENC1 or the system time at the rising or falling edge of the MP0 or MP1 inputs. Further, each encoder block has three capture registers, capturing the current encoder position and system time (sampled with any encoder signal edge) and the position at the last null pulse.



Encoder Input signals and actual position value

To suppress noise glitches, the Encoder signals pass a digital filter, which is realized by sampling the input signals and comparing the last five values with each other. The Filter delivers the signal level at its output which has the majority of these five sampled signals.

To allow adapting to the maximum frequency of the encoder signals, the sample clock frequency can be set to 1, 2, 5, 10, 20, 50 or 100 MHz, or the filter can be switched off completely.



Digital Filtering of encoder input signals

2.26 Ethernet Interface

The netX contains two Ethernet MACs (realized by XMAC/XPEC channels 0 and 1) with integrated PHYs. They support the following modes / features:

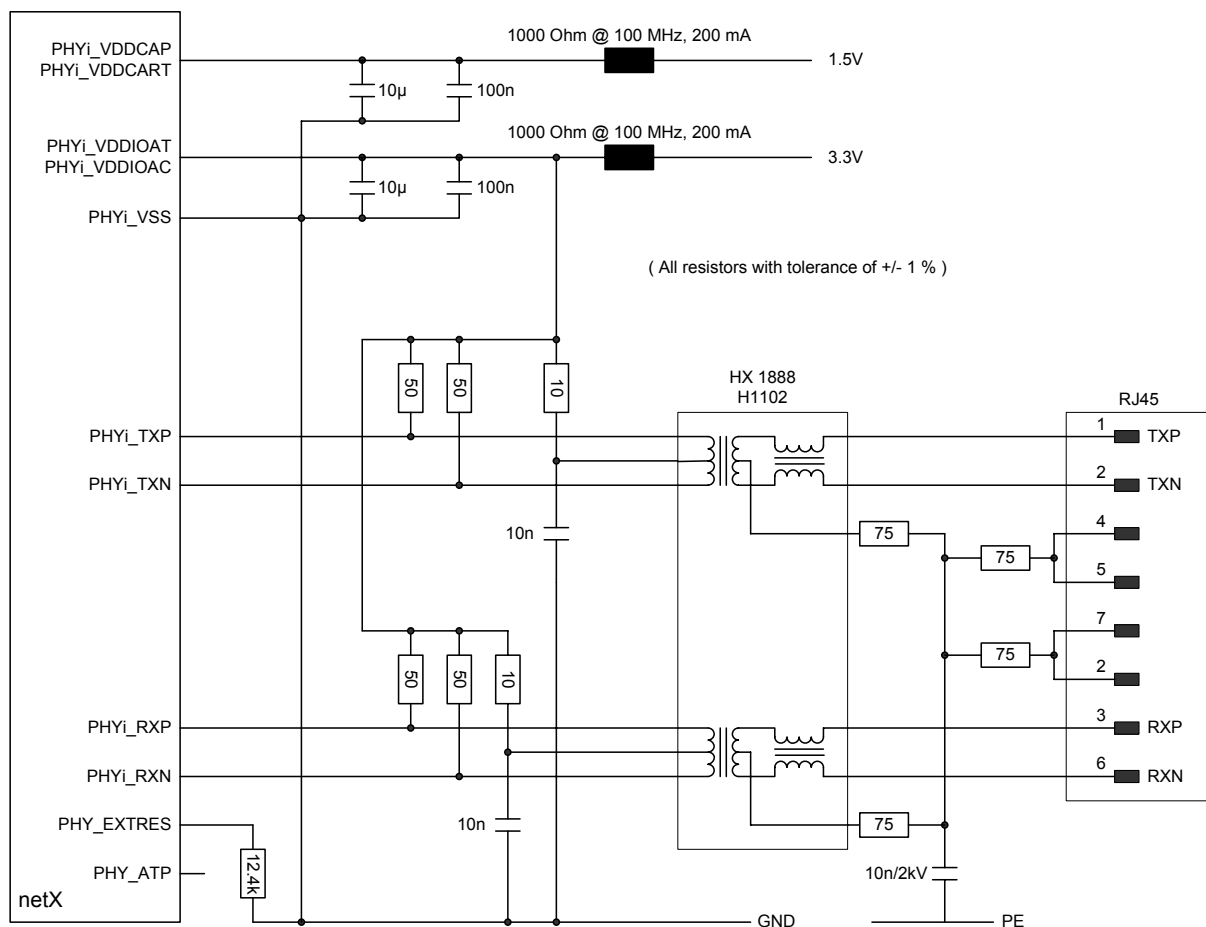
- 10Base-T / 100Base-TX
- 100Base-FX with external drivers
- Auto-Negotiation
- Auto-Crossover
- Auto-Polarity

They are fully compliant with IEEE 802.3 / 802.3u to run the following protocols:

- PROFINET RT
- Ethernet/IP
- Open Modbus on TCP/IP

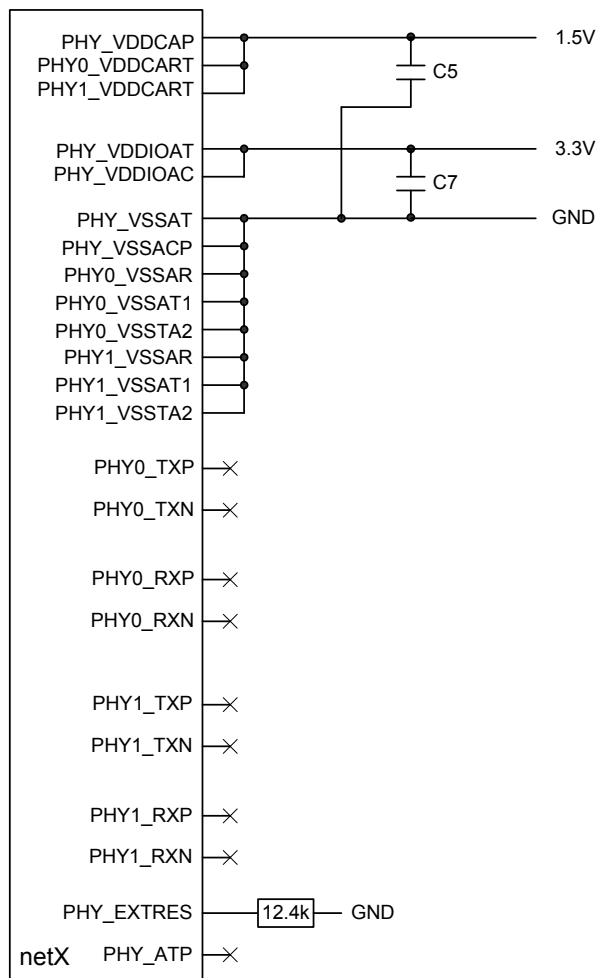
Additionally the Ethernet MAC includes special logic to support:

- Time synchronization based on IEEE 1588
- Powerlink
- EtherCAT
- SERCOS-III



Basic circuit for netX Ethernet interface (100Base-T, only one channel shown)

With applications that do not make use of the Ethernet interface, the PHY signals must be connected according to the following schematic (power must be supplied and reference resistor must be connected). For values of the capacitors, see reference [2].



Circuit when netX Ethernet interface is not used

2.26.1 Real Time Ethernet

Besides standard Ethernet functionality, the netX Ethernet channels also support all current Real Time Ethernet protocols. The different protocols make use of special functionality:

POWERLINK uses HUB-Functionality to forward the Ethernet-Telegrams. The access conflicts on the Ethernet are prevented with the help of the protocol stack running on the ARM 926. To increase performance, the xPEC answers a Poll-Request-Telegram immediately with a Poll-Response-Telegram, without the interrupt latency that would be encountered, if the ARM CPU was be involved.

EtherCAT forwards the Ethernet data immediately from one port to the other and extracts, respectively fills in the local data. Therefore the xMACs have direct connection between each other and have a special filter function to identify the local data inside Ethernet-Frames. The xPEC is responsible for the transfer in and out of the memory of the ARM.

Current Features:

- Slave implementation
- 1 kByte Process Data Memory
- 4 SyncManager units
- 3 FMMUs (Fieldbus Memory Management unit, support only byte-wise mapping)
- Mailbox-Functionality for CANopen over EtherCAT, Ethernet over EtherCAT
- Powerful AL Controller integrated

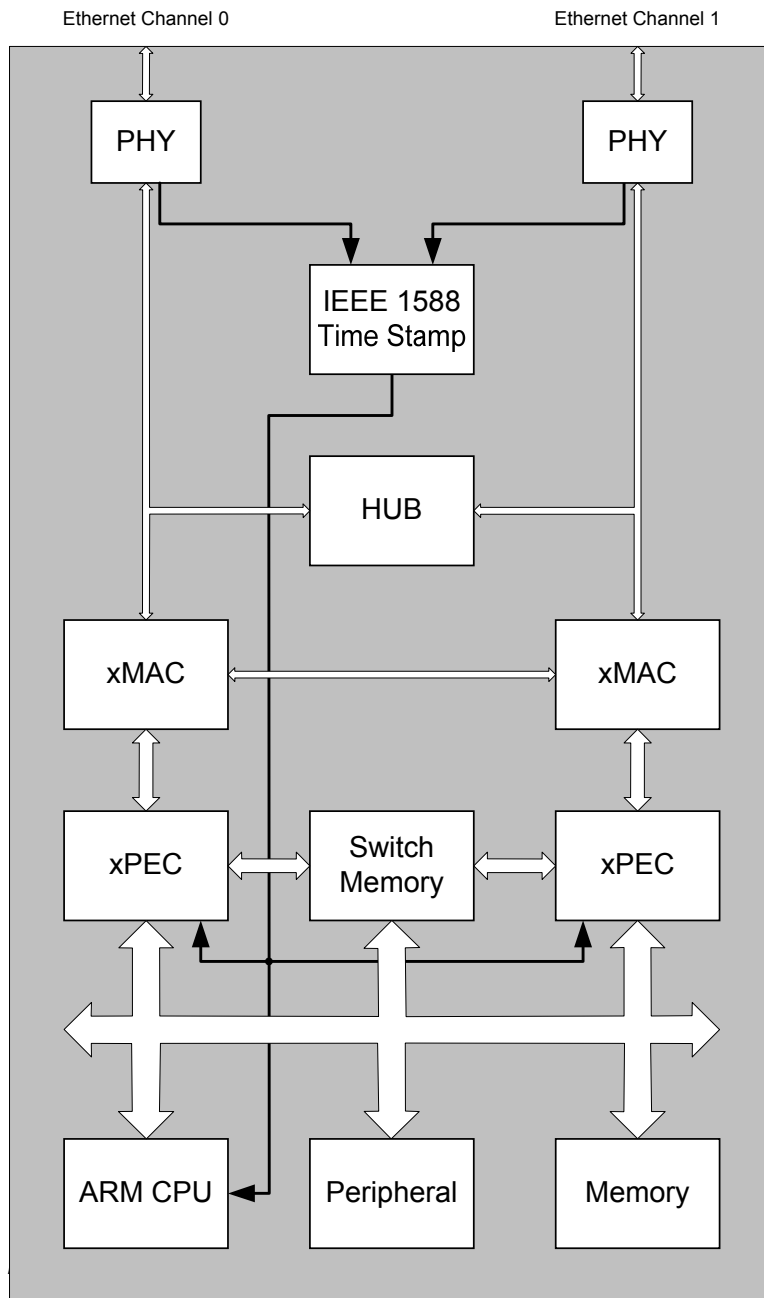
SERCOS-III uses the same filters and the state machines which are defined by the IGS e.V..

EtherNet/IP uses only the standard Ethernet-Functions of one channel. However for motion control with synchronization down to a microsecond, the distributed time according to the international Standard IEEE 1588 is used. This means to measure transmit and receive time by hardware at the MII interface between xMAC and PHY. Using a special protocol, this information is exchanged between the network devices and the local system time is adjusted to a common network time.

The Real-Time-Communication of PROFINET uses a Switch function and priority control according to IEEE 802.1Q. This is implemented by the xPECs, providing own local memory to manage the routing data and a state machine to control the telegram transfers.

For Motion Control, PROFINET uses isochronous transmission. This requires synchronization of the local time by IEEE 1588 and transmission of the telegrams results according to a time table. This time control is another mode of the xPEC state machine. Because the IRT specification is not yet completed, this function is currently disabled.

These features allow device manufactures to use a unified Hardware, activating the several special Real-Time-Ethernet functions by appropriate software as needed.



Block diagram of the special Real-Time Ethernet Features

Note: For better understanding, the data switch is not shown.

2.27 Fieldbus Interface

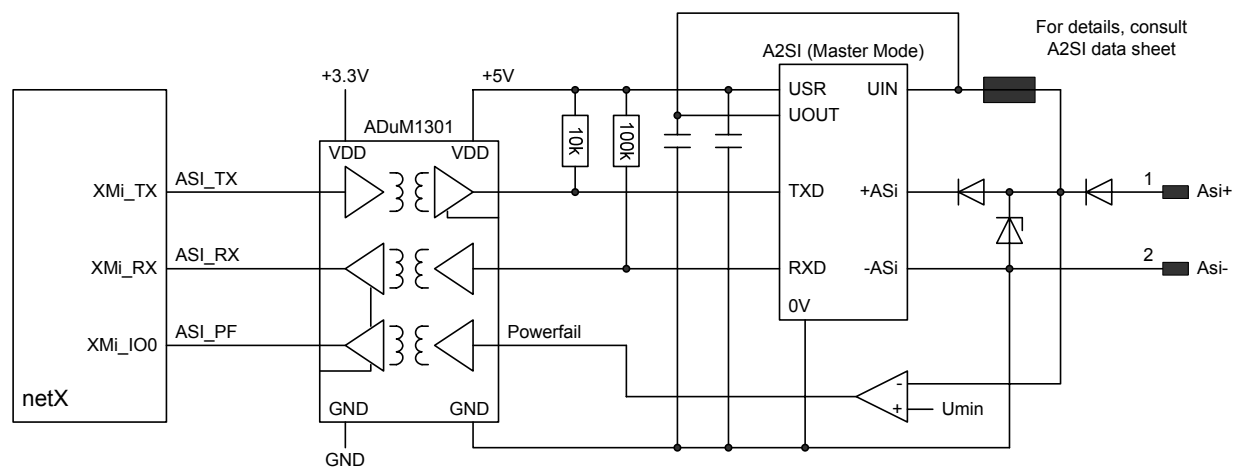
The XMAC/XPEC units of the netX can operate as fieldbus controllers (one XMAC and one XPEC per fieldbus channel), for virtually any existing and future fieldbus system, like:

- AS interface Master
- CANopen Master and / or Slave
- DeviceNet Master and / or Slave
- InterBus Master
- PROFIBUS-DP Master and / or Slave

Different systems can be combined.

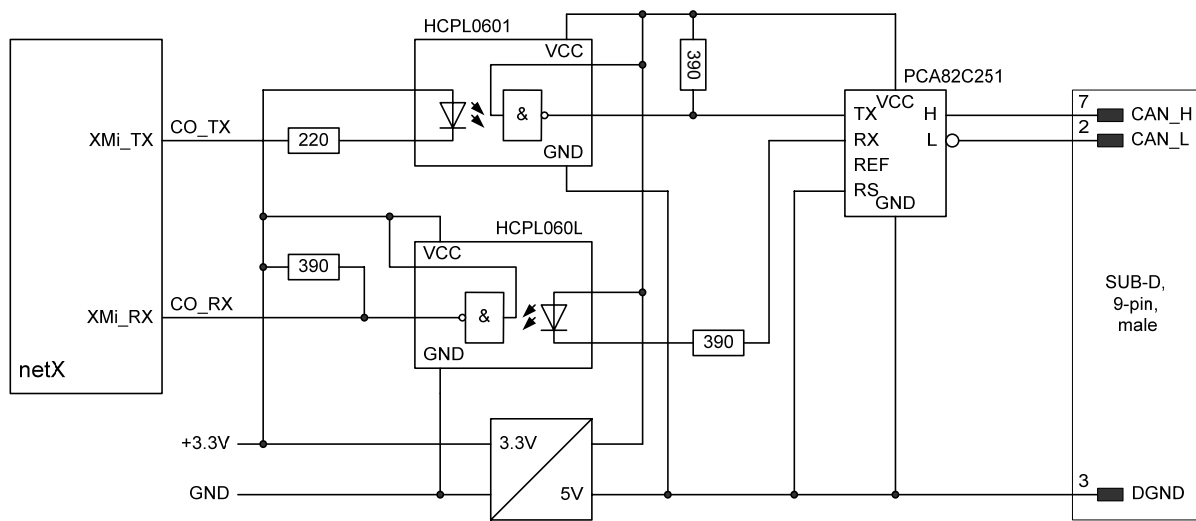
The following sub chapters show the typical external circuitry required for the implementation of a certain fieldbus interface with the netX. These schematics are for demonstration only. For detailed hardware design information, please consult the latest revisions of the reference schematics and the netX hardware design guide, available through the netX Download section at www.hilscher.com as well as the latest revisions of the appropriate fieldbus interface specifications.

2.27.1 AS interface Master



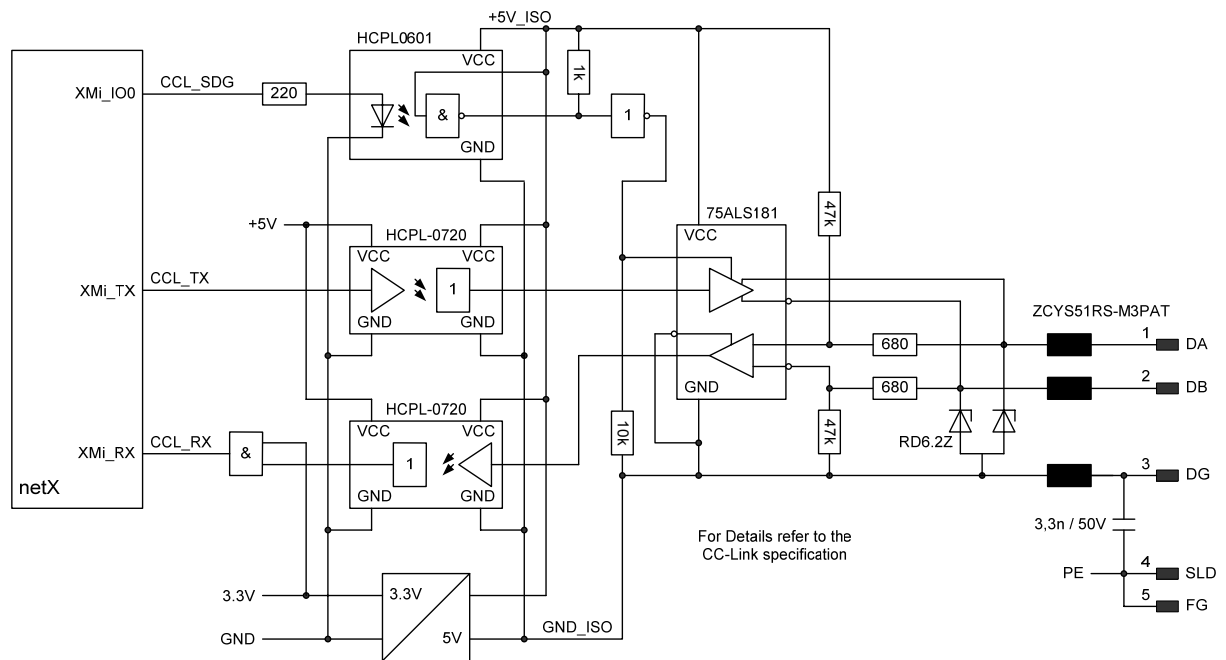
Basic circuit for netX AS interface Master

2.27.2 CANopen Interface



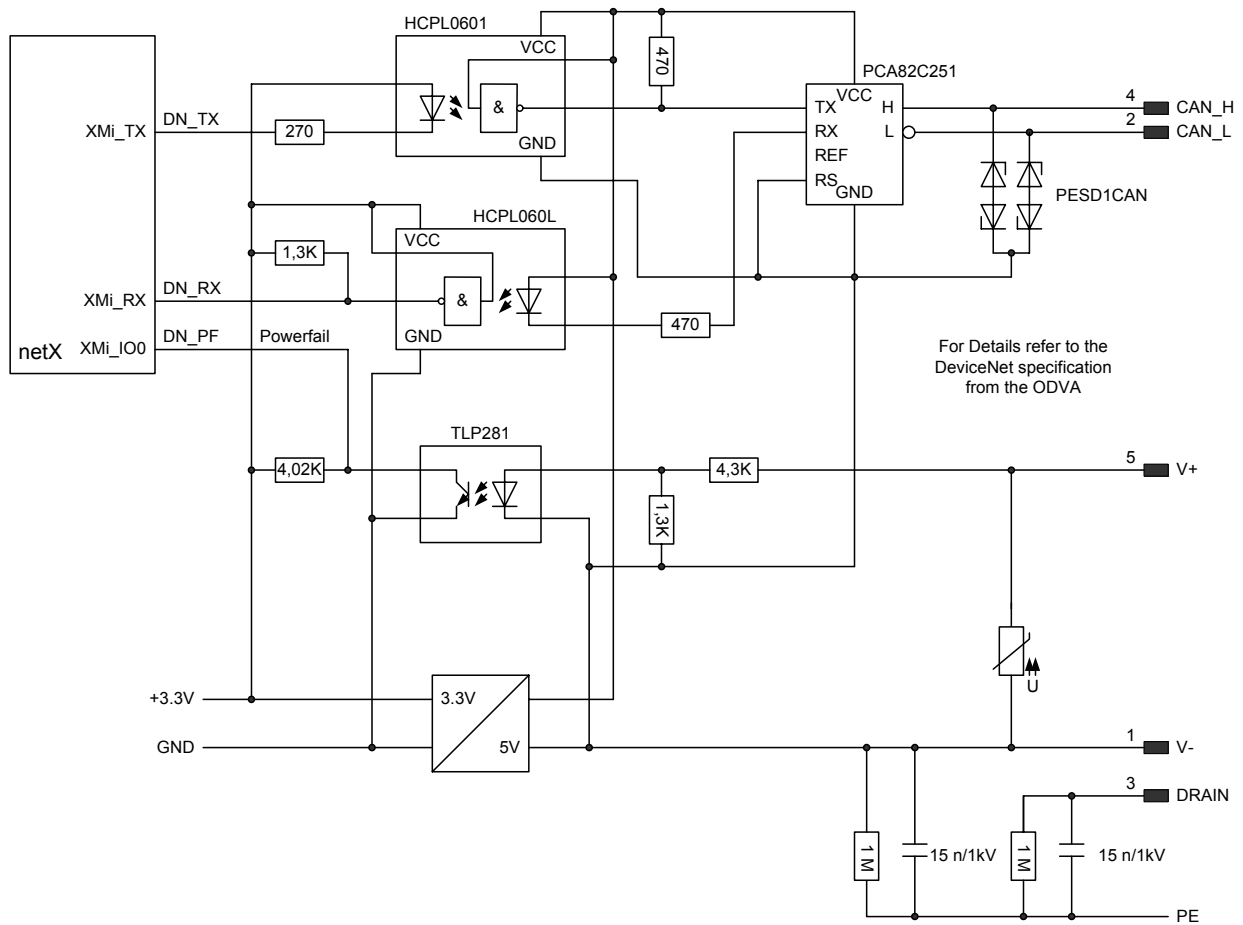
Basic circuit for netX CANopen interface

2.27.3 CC-Link Interface



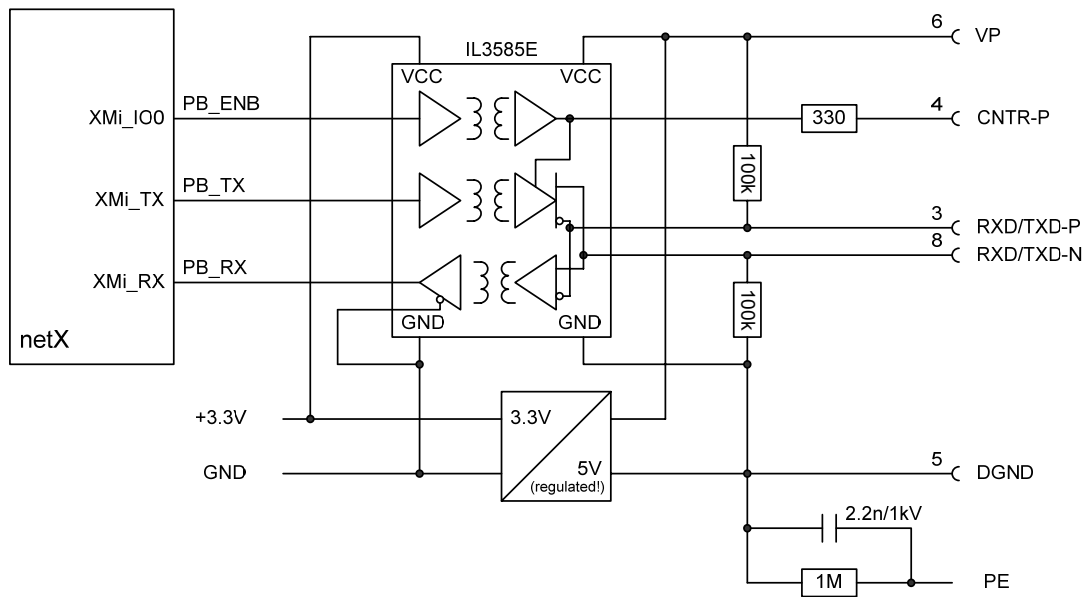
Basic circuit for netX CC-Link interface

2.27.4 DeviceNet Interface



Basic circuit for netX DeviceNet interface

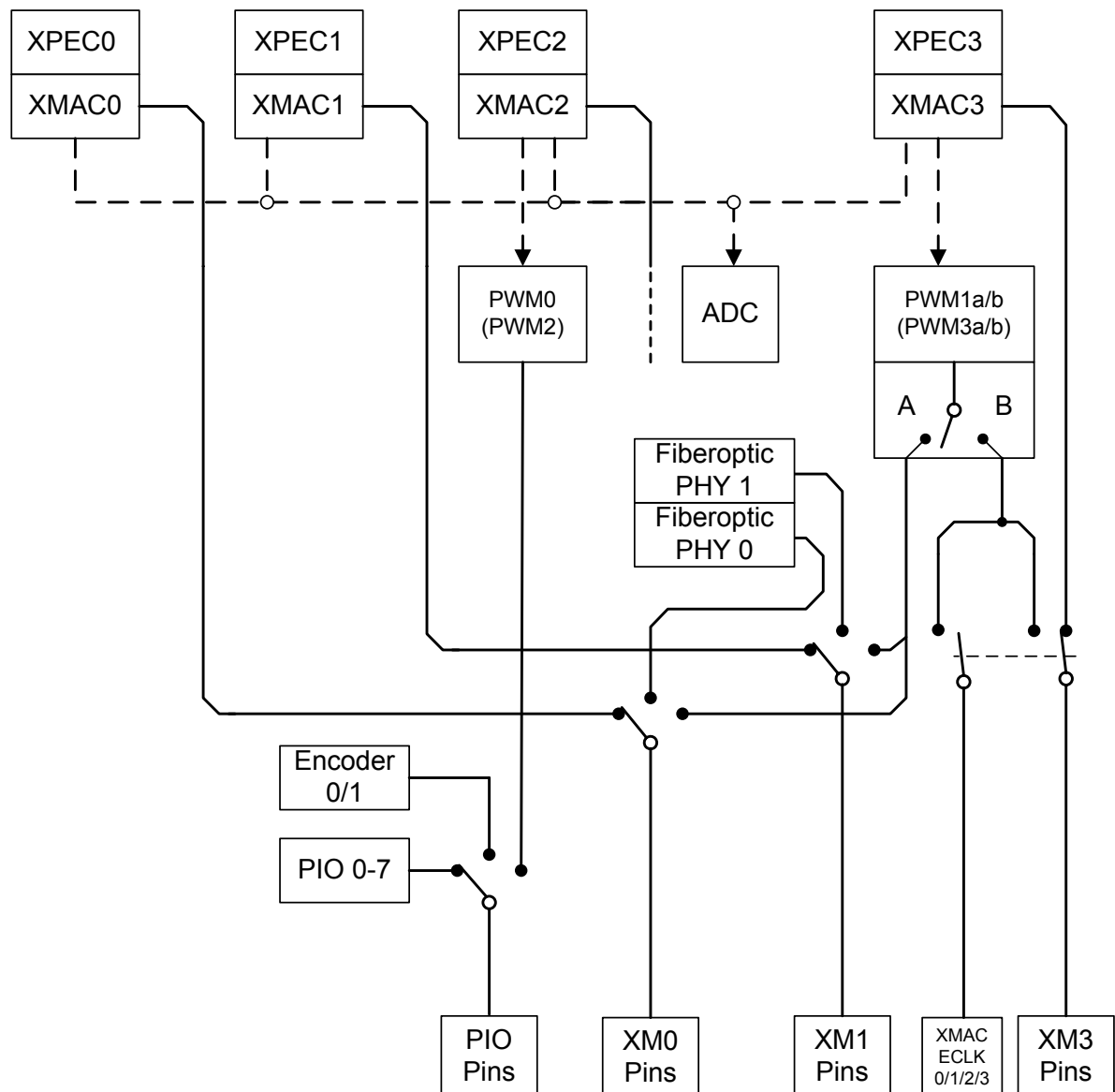
2.27.5 PROFIBUS Interface



Basic circuit for netX PROFIBUS interface

2.28 XMAC Resource Sharing

The previously described modules PWM, ADC, Encoder, PIO and the XMAC communication channels, share some control- and pin resources and can hence not be used independently. The following figure provides an overview about the available resources and how they are shared.

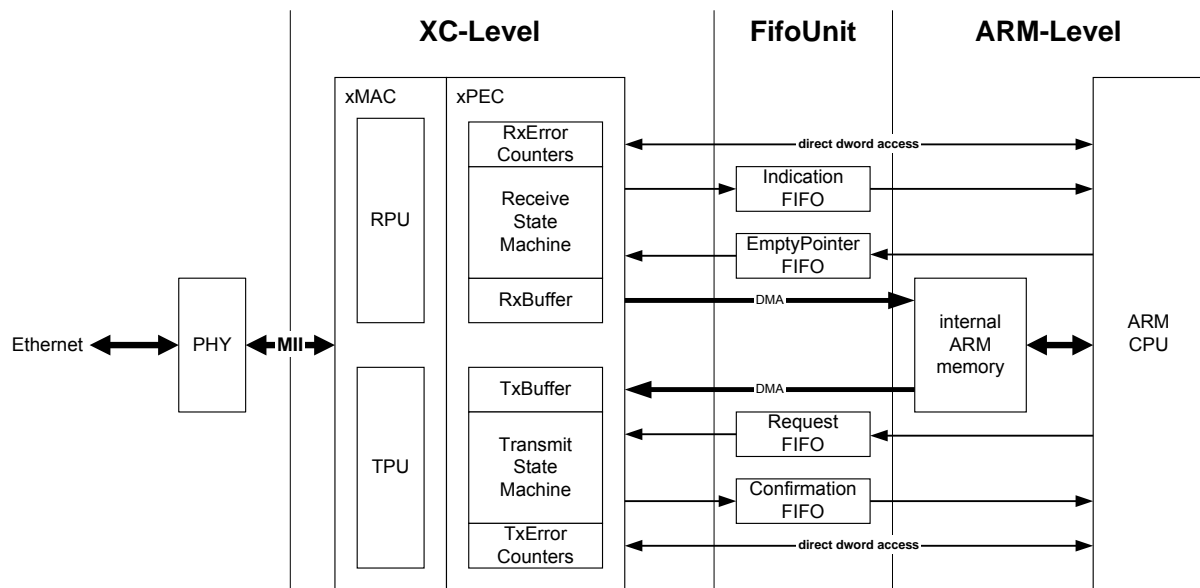


Notes:

- * PWM0 and PWM1 are independent modules (PWM1 has two pinning options)
- * PWM0 can only be controlled by XMAC2 (or by ARM CPU, if XMAC2 is disabled)
- * PWM1 can only be controlled by XMAC3 (or by ARM CPU, if XMAC3 is disabled)
- * ADC can be controlled by any XMAC (or by ARM CPU, if at least one XMAC is disabled)

3 Ethernet Communication Structure

The Ethernet communication between the ARM CPU and the xPEC/xMAC controllers uses special Memory with hardware supported Pointer FIFOs. These FIFOs provide a window into internal ARM Memory and define the starting points of the Ethernet Frames. If the ARM puts a Pointer into the FIFO, the state machine of Ethernet Controller is automatically started to read the Ethernet frame, generate the Checksum and send it out through the PHY. In the receiving direction, the ARM gets an interrupt whenever a Pointer is written into the FIFO by the Ethernet Controller. The data transfer is described in details in the following chapters.



Communication structure between ARM and Ethernet Controllers

3.1 Ethernet data transfer

FIFO-Unit attributes

- 32 separate FIFOs
- each FIFO has a width of 32 bit
- each FIFO depth is configurable
- depth limit: overall 2048 entries

Initialization

- configuration of FIFO depths
- filling of EmptyPointer-FIFO with pointers to internal netX memory blocks of 1536 Bytes size

Transmitting of Ethernet Packets (Request)

- netX takes an Empty Pointer out of Empty-Pointer-FIFO
- netX writes complete Ethernet frame excluding the Frame Check Sequence to internal netX memory block (1536 Byte long) indicated by Empty-Pointer
- netX puts two entries into Request-FIFO:
 - 1st entry: Request-Packet-Pointer
Pointer to Ethernet packet within netX memory (DataPointerEntry)
 - 2nd entry: Request-Packet-Length
Ethernet packet length (LengthEntry)
- xPEC/xMAC is informed automatically if Request-FIFO is not empty
- xPEC/xMAC takes out both entries of Request-FIFO, gets Ethernet packet data (located by Request-Packet-Pointer) via DMA into xPEC/xMAC memory and transmits Ethernet packet on medium (using internal PHYs)
- After transmission of Ethernet packet the xPEC/xMAC puts back Request-Packet-Pointer and Request-Packet-Length into Confirmation-FIFO (Request-Packet-Length entry contains information about transmission successful/failed)
- xPEC/xMAC informs ARM about a confirmation via interrupt
- netX takes out both Request-Packet-Pointer to Empty-Pointer-FIFO and discards the Request Packet Length

Reception of Ethernet Packets (Indication)

- xPEC/xMAC gets an indication of an incoming Ethernet packet
- xPEC/xMAC takes out an Empty-Pointer of Empty-Pointer-FIFO
- xPEC/xMAC writes complete incoming Ethernet via DMA into internal netX memory block (1536 Byte long) indicated by Empty-Pointer
- after reception finished xPEC/xMAC writes two entries into Indication-FIFO:
 - 1st entry: Indication-Packet-Pointer
Pointer to received Ethernet packet within netX memory (formerly Empty Pointer)
 - 2nd entry: Indication Packet Length
Ethernet Packet Length
- xPEC/xMAC informs ARM about Indication via interrupt
- netX takes Indication-Packet-Pointer and Indication Packet Length out of Indication-Fifo pointers and transfers the received data (located by first entry) to the appropriate application (i.e. TCP/IP stack)
- netX releases the Indication-Packet-Pointer due putting it back into the Empty-Pointer-FIFO and discards the Indication-Packet-Length

Error Counters

- The receive and transmit error counters (32 bit wrap-around counters) can be read by netX directly (dword access)

3.2 Error Counters

First block (stb):

The bit stb indicates that this block is the first block of an indication.

Last block (enb):

The bit enb indicates that this block is the last block of an indication.

Timestamp follows (tms):

The bit tms indicates that a timestamp follows as next block. This timestamp shows the content of register systime_ns at the moment the Start-Of-Frame-Delimiter was detected.

Frame number

This field contains an 8 bit wrap-around counter value to assign all data blocks of the current indication each other directly. After the reception of a complete frame the wrap-around counter will be incremented.

RAM segment address

The RAM segment address field contains the offset address of buffer within the internal RAM segment where the Ethernet frame data is located. The buffer size is user-specific and configurable on application level.

Receive error summary (RES):

If any error occurred during reception the receive error summary flag is set.

Runt frame (runt):

If the length of the received frame is below minFrameSize (64 bytes for 10/100 MBit) and the 32-Bit-CRC is valid the frame is declared as a "runt" frame.

CRC error (fcs err):

If the 32-Bit-CRC is invalid and the length of the received frame is between minFrameSize and maxFrameSize a FCS error is reported to upper layers by setting the CRC error bit.

Frame alignment error (fae):

If the received frame does not contain an integer number of octets the frame will be truncated by the nearest octet boundary. If the FCS field is in error in the frame the alignment error bit will be set.

Frame too long error (ftl):

If the length of the received frame and exceeds the maximum permitted frame size a frame too long error is reported to upper layers. The maximum permitted frame size is 1518 Bytes for Non-VLAN-tagged frames and 1522 Bytes for VLAN-tagged frames in 10/100 Mbit Speed.

Rx error (rx err):

If the MII-signal RX_ERR went up during reception an rx error is reported to upper layers.

4 Internal ROM

The internal 32 KByte ROM contains two program modules:

Boot loader	This program code is started directly after power on reset from the ARM CPU. It initializes the basic register set of netX and looks for the 2 nd part of the boot-loader.
Multi tasking Kernel	<p>This is the complete kernel of our operating system rcX which can be used license free on the netX. It has the following features:</p> <ul style="list-style-type: none">• Pre-emptive Multitasking Kernel• Services for Semaphore, Mutex, Event, Timer, Message, Queue, Memory• Dynamic creation of unlimited number of Objects• Optimized Hardware Abstraction Layer for netX peripheral functions• Supported by ARM Development Tool Chain of the company HITEX

4.1 Boot Function

4.1.1 The 1st Stage Bootstrap

The 1st Stage Bootstrap is the program code located in the netX on-chip Boot ROM which is executed following a reset of the chip. This code can not be changed, as it is a fixed part of the netX-Chip. The 1st Stage Bootstrap searches for 2nd Stage Bootstraps in either external storage devices or in internal RAM and checks for a correct boot block and a correct application checksum. Once found, the 2nd Stage boot loader Code is started from the appropriate device. If no 2nd Stage Bootstrap has been found, the 1st Stage Bootstrap either jumps to serial boot mode or waits for a firmware download (DPM boot mode).

4.1.2 The 2nd Stage Bootstrap

The 2nd Stage Bootstrap is the program code located in a bootable source being started by the 1st Stage Bootstrap. This code can be changed since it is located somewhere in a storage device and consists of a few bytes typically.

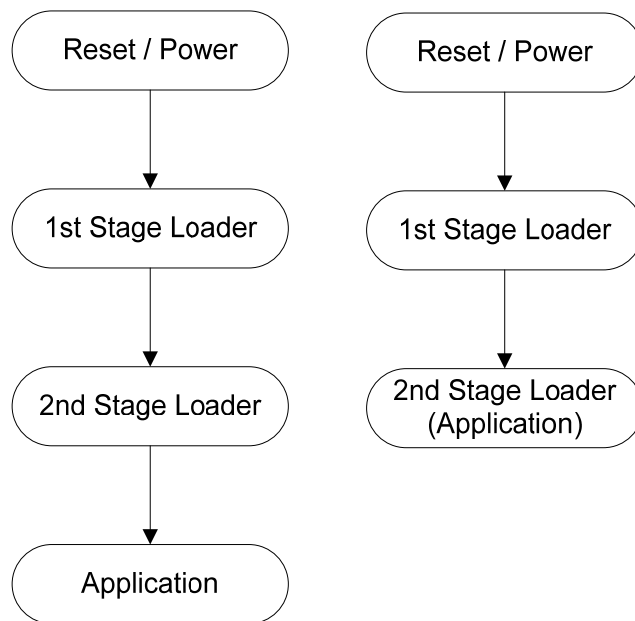
The 2nd Stage Bootstrap is responsible for locating, loading and starting the actual Application program code.

4.1.3 The Application Code

The Application Code is the last instance in the booting process that is finally started by the 2nd stage loader. It contains the main task routines that shall be executed on netX chip in the final stage.

In many Applications the 2nd Stage Bootstrap is already the Application Code itself. In this case the 2nd stage Bootstrap does not exist in its original sense and the Application code takes its place.

4.1.4 The general Bootstrap Sequence.



In cases, where the desired boot option can't be used directly (e.g. DPM boot mode is desired, but the host processor does not match the netX host interface configuration that is automatically used in this mode), a small 2nd stage loader in a small SPI memory can provide the appropriate solution by performing the required configuration and then continuing to boot from the final firmware source. Such a 2nd stage loader can also be used to extend the already numerous boot options of the netX (e.g. boot via Ethernet port).

4.1.5 The 1st Stage Bootstrap Options

To meet different system requirements, the 1st Stage Bootstrap offers a variety of different boot options. The following is a list of the different boot options that are implemented and a summary of their functional operation:

- **Serial Port Standard Boot Option:**
The 2nd Stage Bootstrap (or Application) code is received from the serial port 0 or USB and is loaded into internal or external RAM and started from there.
- **Memory Interface parallel FLASH Boot Option:**
The 2nd Stage Bootstrap (or Application) to be executed is located in an external parallel FLASH, connected to the memory interface and copied optionally to RAM and started at the indicated entry point (direct execution out of FLASH is also possible at cost of performance).
- **Serial FLASH/EEPROM Boot Option:**
The 2nd Stage Bootstrap (or Application) to be loaded is located in external serial FLASH or EEPROM, loaded to internal or external RAM at the specified memory address and started from there at the indicated entry point.
- **I2C Boot Option:**
The 2nd Stage Bootstrap (or Application) to be loaded is located in an external I2C Device, loaded to the internal or external RAM at the specified memory address and started from there at the indicated entry point.
- **MMC Boot Option:**
The 2nd Stage Bootstrap (or Application) to be loaded is located in an external MMC Device that is formatted either in FAT12, FAT16 or FAT32 and contains a file named "NETX.ROM", loaded to internal or external RAM at the indicated program memory address and started at the indicated entry point.
- **Extension Bus parallel FLASH Boot Option:**
The 2nd Stage Bootstrap (or Application) to be executed is located in an external parallel FLASH connected to the Extension Bus interface and copied optionally to RAM and started at the indicated entry point (direct execution out of FLASH is also possible at cost of performance).
- **Dual-Port-Memory Boot Option:**
The 2nd Stage Bootstrap (or Application) to be executed is loaded into the internal RAM by an external HOST via the Dual-Port Memory Interface and started from there.

4.1.6 The 1st Stage Boot Option Selection

Once the 1st Stage Bootstrap is started, it performs a series of checking operations to determine which boot options to use. The behavior of the 1st Stage Bootstrap is determined by a pre-selected boot mode, which is realized by applying certain signal levels to the RDY and RUN pins of the netX which are read upon a reset (see also chapter 2.3)

The bootstrap first checks for conditions that indicate if it should enter the Serial Boot Mode prior to all other modes. This guarantees that the user can always override any other boot option, provided by the target system by forcing the Serial Boot mode.

If the conditions for a certain boot option are not met, the loader steps to the next option and continues until it detects a valid boot image or eventually either waits in a loop or jumps to the Serial Boot Mode again. The flowchart shown in Figure 1-1 illustrates the process the 1st Stage Bootstrap uses to determine the desired boot option.

4.1.7 The 1st Stage Boot Option detection

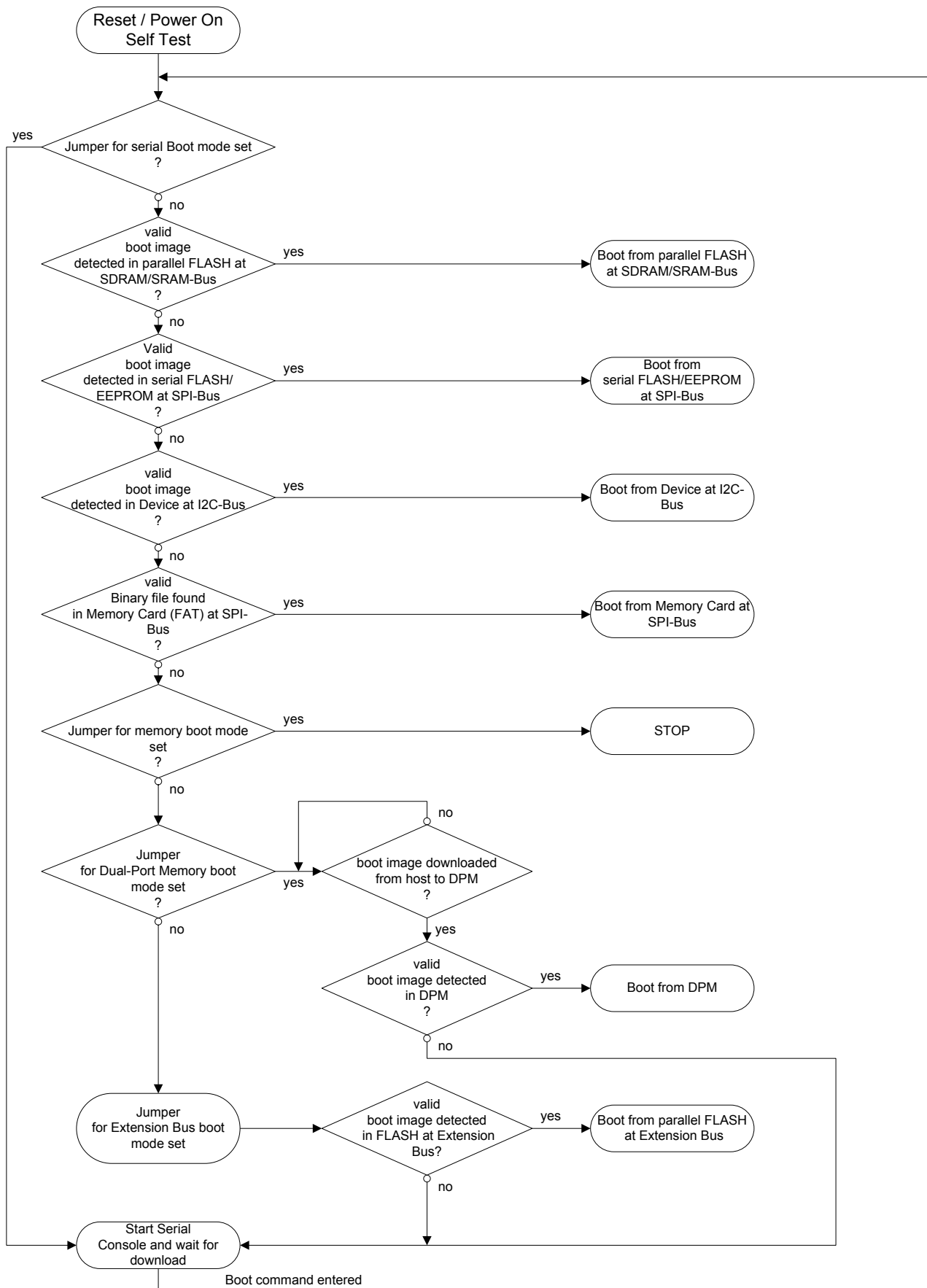
The following list describes the sequence in which the Bootstrap searches for 2nd Stage Bootstraps:

- 3.) Serial Boot Mode selected by the presence of a special jumper position.
- 4.) Memory interface parallel FLASH Boot option selected by the presence of a valid boot image within a FLASH memory connected to the memory bus (SDRAM/SRAM).
- 5.) Serial FLASH/EEPROM Boot option selected by the presence of a valid boot image within a serial Flash memory, connected to the SPI interface, using CS0.
- 6.) I2C Boot option selected by the presence of a valid boot image in an I2C memory.
- 7.) MMC Boot option selected by the presence of a valid boot image within the file "NETX.ROM" in the FAT8/16/32 formatted Root-Directory of an MMC, connected to the SPI interface, using CS1.
- 8.) Extension Bus parallel FLASH Boot option selected by the presence of a valid boot image in a FLASH memory, connected to the Extension Bus.
- 9.) Dual-Port-Memory HOST Boot option selected by the presence of a valid boot image in DPM.

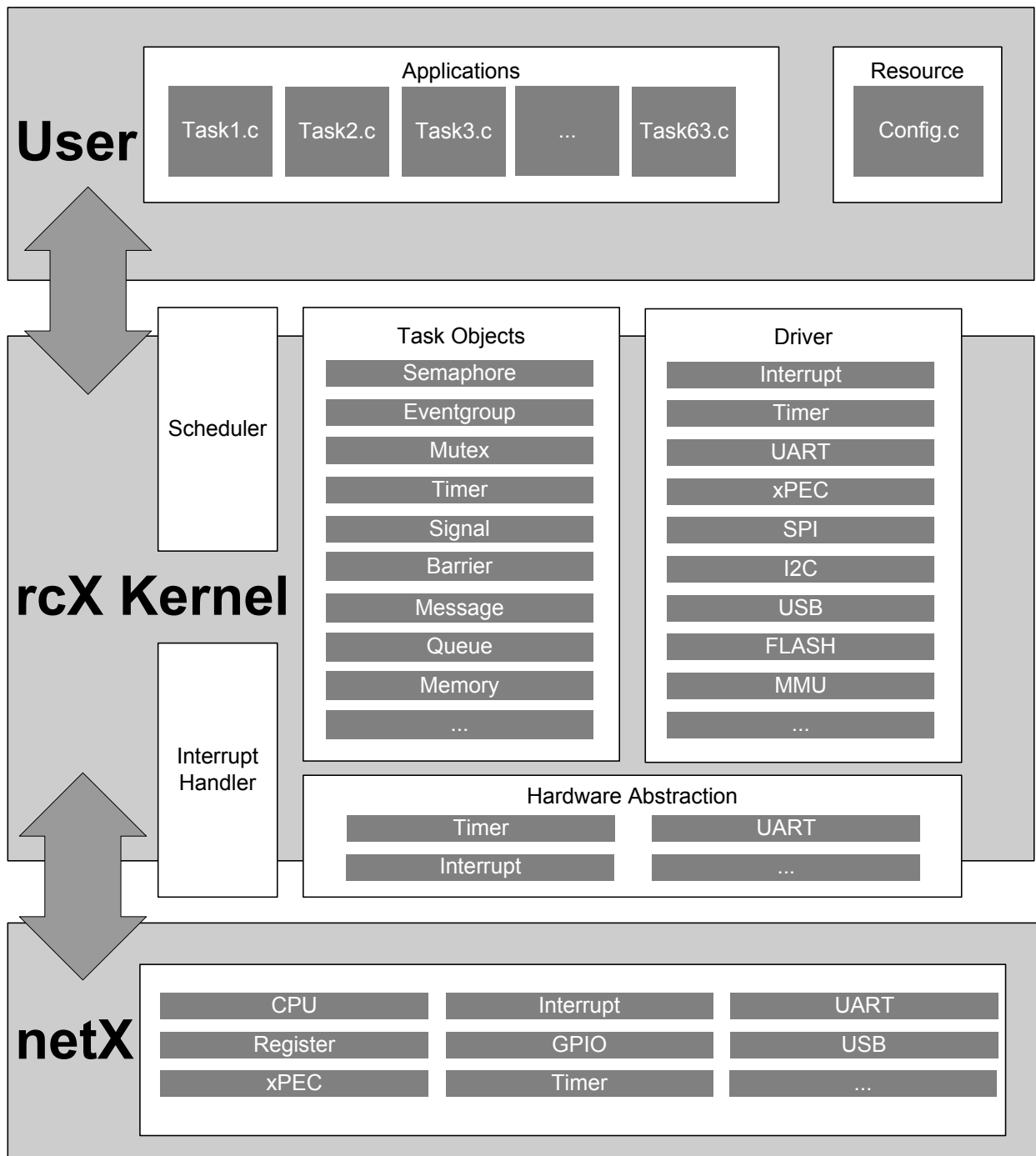
Notes:

Options 8 and 9 are only checked, if the appropriate Boot mode has been pre-selected.

When selecting DPM boot mode or Extension Bus boot mode, these modes will still never be entered, when a valid boot image (boot block and application checksum correct) is found in either a parallel Flash at SDRAM/SRAM interface, an SPI Flash, an I2C Memory or an MMC card.

The 1st Stage Loader Boot-Sequence as Flowchart:

4.2 High Performance Real-Time Operating System rcX



rcX Block diagram

The Architecture of rcX is organized to provide a Hard-Real-Time performance for embedded netX applications of small to medium sizes. The internal Kernel configuration covering the Object Data Structures, Inter-task Communication Paths and Time Management are highly optimized in terms of size and access speed. As a result, rcX provides a very low interrupt latency and fast task switching as main characteristics.

4.2.1 Pre-emptive Kernel

The rcX Kernel is a Pre-emptive full featured multitasking Operating System that can operate with a scaleable number of Task-Processes. Up to 255 Tasks can be managed in accordance to their priority. Background Interrupts are managed by rcX wrapper functions and lead the user to its encapsulated Interrupt Service Routines which permit for instance to make Kernel calls also at Interrupt Level.

4.2.2 Small Footprint

Although rcX has over 100 Application Functions available to the user, the Kernel has a very small memory footprint of about 24 KByte code (ROM) and 6 KByte (internal TCM) data memory.

4.2.3 Modularity

rcX is a modular RTOS that has been designed to meet the needs of all possible embedded netX applications. Along with the Kernel there are Drivers for all the different kinds of netX Peripherals. All modules are working fully independent of each other and thus it permits for leaving out those that are not needed in order to save memory space if necessary.

4.2.4 Dynamic Objects

All rcX Objects such as Tasks, Interrupts, Timers, and Inter-task Communication Paths are fully dynamic. They can be created and deleted at any time during runtime. These Objects can also be statically configured due to a “font-end” configuration file. This allows an easy, centralized configuration of your project and maintains the overview of the configured resources.

4.2.5 Rich API Functions

rcX is rich on API functions for the following Inter-task Communication Paths:

- Messages
- Queues
- Mutexes
- Semaphores
- Barriers
- Timers
- Eventgroups
- Signals
- Multiples

4.2.6 Tool Support

rcX is tightly integrated into one leading tool suite called HiTop available from the company Hitex aimed to make your debugging job easier. Internal Task states as well as states of the different kinds of rcX-Objects can be monitored. For netX, rcX supports compilers from GNU and ARM® and the debuggers from Hitex with Tanto and Tantino and ARM® with Realview-ICE®.

5 Electrical Specifications

5.1 Absolute Maximum Ratings

Stresses beyond the following ratings may cause permanent damage to the device. Please note, that these stress ratings do not imply functional operation of the device and that exposure to these ratings for extended periods of time may affect device reliability.

Symbol	Parameter	Conditions	Ratings	Unit
V _{DDC}	Power supply, core voltage		-0.5 to 2.0	V
V _{DDIO}	Power supply, IO voltage		-0.5 to 4.6	V
V _{DDH}	Power supply, Hostinterface voltage		-0.5 to 6.0	V
USB_V _{DDC}	USB power supply, core voltage		-0.5 to 2.0	V
USB_V _{DDIO}	USB power supply, IO voltage		-0.5 to 4.6	V
OSC_V _{DDC}	Oscillator power supply, core voltage		-0.5 to 2.0	V
RTC_V _{DDIO}	Real-time clock power supply, IO		-0.5 to 4.6	V
RTC_V _{DDC}	Real-time clock power supply core		-0.5 to 2.0	V
ADC_V _{DD}	AD-Converter power supply, digital		-0.5 to 2.0	V
ADC_V _{ADD}	AD-Converter power supply, analog		-0.5 to 4.6	V
ADC_V _{REFM}	AD-Converter ref. voltage low	V _{REFM} < V _{REFP}	-0.5 to 4.6	V
ADC_V _{REFP}	AD-Converter ref. voltage high	V _{REFP} < ADC_V _{ADD}	-0.5 to 4.6	V
ADC_V _{AIN}	Analog input voltage V _I	V _I < V _{ADD} + 0.5 V	-0.5 to 4.6	V
PHY_V _{DDCART}	PHY power supply		-0.5 to 2.0	V
PHY_V _{SSACP}	PHY power supply		-0.5 to 2.0	V
PHY_V _{DDIOAC}	PHY power supply		-0.5 to 4.6	V
PHY_V _{DDIOAT}	PHY power supply		-0.5 to 4.6	V
PHY_V _{DDD}	PHY digital power supply		-0.5 to 2.0	V
V _I	Input voltage	V _I < V _{DDIO} + 0.5 V	-0.5 to 4.6	V
V _O	Output voltage	V _O < V _{DDIO} + 0.5 V	-0.5 to 4.6	V
I _O	Output current	6 mA type	±21	mA
		9 mA type	±29	mA
		USB type	±37	mA
T _A	Operating ambient temperature	Note 1)	-40 to +85	°C
T _j	Junction temperature	Note 1)	-40 to +125	°C
T _{stg}	Storage temperature	Note 2)	-65 to +150	°C

Notes:

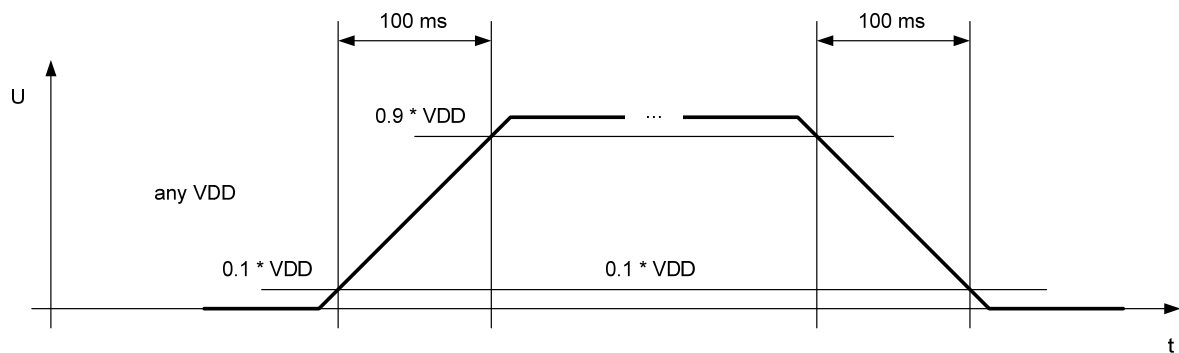
- 1) See also chapter 6.1
- 2) This temperature range has no relation to soldering conditions or solderability of the components. Please check chapter 6.3 for storage condition that maintain solderability.

5.2 Power Up Sequencing

All power supplies of the netX chip must be switched on or off within 100 ms time.

Important Note: To make sure that no I/O pins drive, the voltages must be applied in the following order: The core supply voltage must have reached at least approx. 0.8 V before the IO supply voltage is applied.

All voltages should have reached the $0.9 \cdot V_{DD}$ condition not later than 100ms after the first supply started to ramp up (time measured from $0.1 \cdot V_{DD}$ condition). The following figure shows details of powering up and down.



Note:

All power supply voltage pins have to be connected to the power supplies, even when certain peripheral parts of the netX chip are not used!

5.3 Power Consumption / Power Dissipation

The typical power consumption of the netX 500/100 is approximately 1.6 Watts (integrated PHY's in use) or 1.1 Watts (PHY's turned off). The following tables provide an overview of power consumption of single modules of the chip and some standard applications.

Power Consumption of netX 500/100 Chip:

Symbol	Unit	Min.	Typ.	Max.	Unit
P _{BASE}	Ground consumption, chip in idle state			0.72	W
P _{ARM}	ARM CPU	0.01	0.08	0.1	W
P _{SDRAM}	SDRAM Note 1)	0.03	0.1	0.28	W
P _{HIF}	Host interface, DPM mode 16 Bit Note 2) 3)	0	0.05	0.1	W
P _{XC}	Four communication channels (XMAC and XPEC) Note 2) 4)	0	0.24	0.28	W
P _{PHY}	Two Ethernet PHYs (100Base-TX) Note 5)	0		0.49	W
P _{LCD}	LCD Controller Note 6)	0	0.01	0.03	W
P _{MAX}	Max. chip power consumption Note 7)			2	W
P _{RES}	Max. chip power consumption while PORn is active			0.05	W

All values above assume nominal supply voltages ($V_{DDC} = 1.5V$, $V_{DDIO} = 3.3V$). Increasing the supply voltages to their maximum (1.65V and 3.6V) will increase power consumption by appr. 25%

Notes:

- 1) SDRAM IO power consumption strongly depends on capacitive load, access rate and type of access (write consumes significantly more power than read). Max. value is valid for one typical 32Bit SDRAM and one 16Bit FLASH component at stress test conditions (maximum possible (write) access rate (which can not be reached in a useful application).
- 2) Min values apply when modules are idle (not used).
- 3) Max. value is at maximum possible access rate and load capacity of 50pF on all host interface data signals.
- 4) If only one communication channel (fieldbus or Ethernet) is used, power consumption is ~ 25% of stated values (50% for two channels, 75% for three channels).
- 5) Each Ethernet port requires one communication channel and one PHY. If only one PHY is enabled, power consumption is 50% of stated value (max. value). Min. value applies when both PHYs are disabled.
- 6) LCD Controller available on netX 500 only. Max. value is valid at max. resolution (1024x768 -> not recommended, may cause bandwidth problems) and memory setup as described in Note 1.
- 7) At preceding conditions.

Please also consider the netX heat test report, available at www.hilscher.com (netX – Downloads – Testreports)!

netX power consumption at typical setups:

Symbol	Parameter	Min.	Typ.	Max.	Unit
P _{BOOT}	Serial boot mode, no external communication		0.8		W
P _{CE2ETH}	2 Port Full Duplex Ethernet, LCD, SDRAM, Windows CE Software		1.5		W
P _{CE1ETH}	1 Port Full Duplex Ethernet, LCD, SDRAM, Windows CE Software		1.3		W
P _{2ETH}	2 Port Full Duplex Ethernet, SDRAM, IO		1.5		W
P _{FB}	Fieldbus 12 MBaud, one channel,		1.0		W

Power supply current overview:

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
I _{DD_3V3}	complete power supply 3.3V Note 1)	3.3 V			350	mA
I _{DD_1V5}	complete power supply 1.5V Note 2)	1.5 V			800	mA
I _{DDC}	Power supply current, netX core	1.5 V	2		600	mA
I _{DDIO}	Power supply current, netX IO Note 3)	3.3 V	12		200	mA
I _{DDH}	Clamping diode current, host interface	3.3 V			10	mA
		5.0 V			10	mA
ADC_I _{VDDIO}	ADC IO current	3.3 V	0.6		2.5	mA
RTC_I _{VDDC}	RTC core current	1.5 V		0.1		μA
RTC_I _{VDDIO}	RTC IO current	3.3 V	4		50	μA
USB_I _{VDDC}	USB core current	1.5 V			5	mA
USB_I _{VDDIO}	USB IO current Note 4)	3.3 V			5	mA
OSC_I _{VDDC}	Oscillator core current	1.5 V			6	mA
PHY_I _{VDDC}	PHY core current, includes I _{VDDCART} and I _{VDDCAP} Note 5)	1.5 / 1.65 V		150	180	mA
PHY_I _{VDDIO}	PHY IO current, includes I _{VDDIOAC} and I _{VDDIOAT} Note 6)	3.3 / 3.6 V		130	150	mA
PHY_I _{EXTPU}	PHY IO ext. pull-up resistor current Note 6)	3.6 V			100	mA

Notes:

- 1) Assuming standard setup (1 * SDRAM, 1* par. FLASH, 1* ser. FLASH, 2 Ethernet Ports at 100Base-TX). It is recommended, to design the power supply for a worst case current of **0.4A @ 3.3V** (see also Note 3!)
- 2) Assuming that no other components than the netX 500/100 are powered by the 1.5V core supply. It is recommended, to design the power supply for a worst case current of **1A @ 1.5V**.
- 3) Actual max. value is widely application specific. Stated value does not include applications that drive current through a large number of IO pins (e.g. for directly driving LEDs).
- 4) The max. value does not include short circuit conditions with a worst case value of 170mA (assuming the short circuit is behind the serial resistors of D- / D+). If the system is to keep running during such conditions, the 3.3V system power supply must either be able to source the additional current or the USB IO power supply must be powered by a separate supply.
- 5) Stated current values apply, when both PHYs are enabled and operate in 100Base-TX mode (worst case for core power consumption). Max. current applies at max. voltage conditions.
- 6) Stated current values apply, when both PHYs are enabled and operate in 10Base-T mode (worst case for IO power consumption). Max. current applies at max. voltage conditions.

5.4 AC / DC Specifications

5.4.1 DC Parameters

The following table describes the standard pad cells of the netX. The special pad cells for crystals, USB, ADCs and PHYs are described in the following chapters.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{DDC}	Power supply voltage, core		1.425	1.5	1.65	V
V_{DDIO}	Power supply voltage, IO		3.0	3.3	3.6	V
V_{DDH}	Voltage for internal clamping diodes, host interface	3.3 V	3.0	3.3	3.6	V
		5.0 V	4.75	5.0	5.25	V
V_N	Negative trigger voltage	Schmitt trigger input	0.6		1.8	V
V_P	Positive trigger voltage	Schmitt trigger input	1.2		2.4	V
V_H	Hysteresis voltage	Schmitt trigger input	0.3		1.5	V
V_{IL}	Input voltage, low		0		0.8	V
V_{IH}	Input voltage, high	3.3 V	2.0		$V_{DDIO} + 0.5$	V
		5.0 V	2.0		$V_{DDH} + 0.5$	V
$slew_{ri}$	Input rise (normal input)		0.015			V/ns
$slew_{fa}$	Input fall time (normal input)		0.015			V/ns
$slew_{ri}$	Input rise (Schmitt input)		0.27			V/ms
$slew_{fa}$	Input fall time (Schmitt input)		0.27			V/ms
I_{OZ}	Off-state current	$V_O = V_{DDIO}$ or GND			±10	µA
I_{OS}	Output short circuit current				-250 Note 1)	mA
I_{LI}	Input leakage current	$V_I = V_{DDIO}$ or GND Normal input		±0.1	±10 Note 2)	µA
		$V_I = \text{GND}$ pull-up 50 kΩ	-37	-103	-253	µA
		$V_I = V_{DDIO}$ pull-down 50 kΩ	26	73	175	µA
I_{OL}	Output current, low	$V_{OL} = 0.4V$ 6 mA type	6.0			mA
			9.0			mA
I_{OH}	Output current, high	$V_{OH} = 2.4V$ 6 mA type	-6.0			mA
			-9.0			mA
V_{OL}	Output voltage, low	$I_{OL} = 0 \text{ mA}$			0.1	V
		$I_{OL} = 6 \text{ mA}$ (6 mA type)			0.4	V
		$I_{OL} = 9 \text{ mA}$ (9 mA type)			0.4	V
		$I_{OL} = 18 \text{ mA}$ (18 mA type)			0.4	V
V_{OH}	Output voltage, high	$I_{OH} = 0 \text{ mA}$	$V_{DDIO} - 0.1$			V
		$I_{OH} = 6 \text{ mA}$ (6 mA type)	2.4			V
		$I_{OH} = 9 \text{ mA}$ (9 mA type)	2.4			V
		$I_{OH} = 18 \text{ mA}$ (18 mA type)	2.4			V
C_{IN}	Pin capacitance	Input buffer	2	4	6	pF
C_{IO}	Pin capacitance	Output and bidirectional buffer	2	4	6	pF
R_{PU}	Resistance of internal 50k pull-up resistor	$V_{DDIO} = 3.3 \pm 0.3V$ $T_A = -40 \text{ to } +85^\circ\text{C}$	14.2	31.9	80.7	kΩ
R_{PD}	Resistance of internal 50k pull-down resistor	$V_{DDIO} = 3.3 \pm 0.3V$ $T_A = -40 \text{ to } +85^\circ\text{C}$	20.6	44.9	116.4	kΩ

Notes:

- 1) Output short circuit time no longer than one second and only one pin of the chip!
- 2) This value only applies to (also internally) unconnected standard pad cells. As most netX pins are equipped with internal pull-up or pull-down resistors, please also consider chapter 6.4 (Signal Definitions) to determine actual leakage current of a specific pin!

Symbol definition:

V_{DDC} , V_{DDIO} : Indicates the voltage range for normal logic operations that occur when $V_{SS} = 0$ V.

V_N : Indicates the input level at which the output level is inverted when the input is changed from the high-level side to the low-level side.

V_P : Indicates the input level at which the output level is inverted when the input is changed from the low-level side to the high-level side.

V_H : Indicates the differential between the positive trigger voltage and the negative trigger voltage.

V_{IL} : Indicates the voltage which could be applied to the input pins to guarantee a logical low input level.

V_{IH} : Indicates the voltage which could be applied to the input pins to guarantee a logical high input level.

$slew_{ri}$: slew rate rise time

$slew_{fa}$: slew rate fall time

I_{OZ} : Indicates the current that flows from the power supply pins when the rated power supply voltage is applied when a 3-state output has high impedance.

I_{OS} : Indicates the current that flows when the output pins are shorted (to GND pins) when output is at high level. The output short circuit must not exceed more than one second and is only for one pin.

I_{LI} : Indicates the current that flows via an input pin when a voltage is applied to that pin.

I_{OL} : Indicates the current that flows to the output pins when the rated low-level output voltage is being applied.

I_{OH} : Indicates the current that flows from the output pins when the rated high-level output voltage is being applied.

V_{OL} : Indicates the output voltage at low level and when the output pin is open.

V_{OH} : Indicates the output voltage at high level and when the output pin is open.

Output buffer output current (I_{OL} , I_{OH})

The cell base IC output current is defined at the conditions of $V_{OL} = 0.4$ V and $V_{OH} = 2.4$ V in a 3.3 V output buffer. However, since it is possible to use different V_{OL} and V_{OH} values in actual applications, the following coefficient should be used to estimate the I_{OL} and I_{OH} characteristics according to the use conditions.

$$V_{OL} = 0.4 \text{ to } 0.6 \text{ V}, V_{OH} = (V_{DD} - 0.4 \text{ V}) \text{ to } (V_{DD} - 0.6 \text{ V})$$

Since I_{OL} and I_{OH} change almost in proportion to the output voltage, direction approximation can be used. Approximation method:

$$I_{OL}' = I_{OL} \times V_{OL} / 0.4 \text{ (mA)}$$

$$I_{OH}' = I_{OH} \times (V_{DD} - V_{OH}) / 0.6 \text{ (mA)}$$

I_{OL} : I_{OL} specification when $V_{OL} = 0.4$ V

V_{OL} : V_{OL} value being used

I_{OH} : I_{OH} specification when $V_{OH} = 2.4$ V (3.3 V output buffer)

V_{OH} : V_{OH} value being used

The MIN, TYP and MAX designators in the graph show the curve under the following conditions:

Min.: $V_{DD} = 3.0$ V (3.3 V output buffer), $T_J = 125^\circ\text{C}$

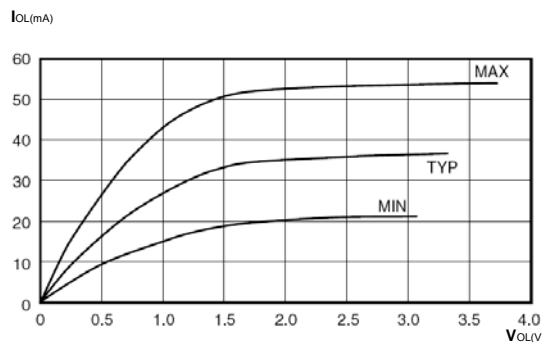
Typ.: $V_{DD} = 3.3$ V (3.3 V output buffer), $T_J = 25^\circ\text{C}$

Max.: $V_{DD} = 3.6$ V (3.3 V output buffer), $T_J = -40^\circ\text{C}$

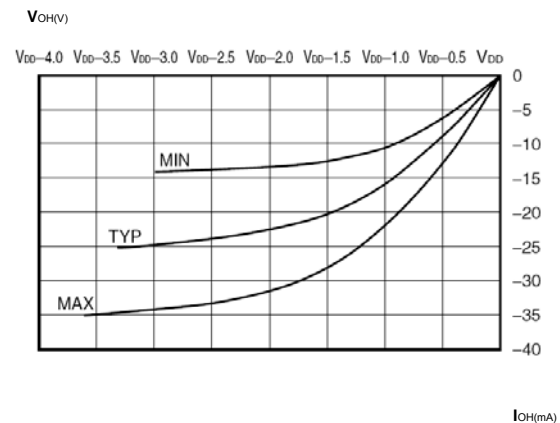
6mA output driver:

(typical block type: TDOPAC33NN06, TDOPAC33NL06)

(a) I_{OL} vs. V_{OL}



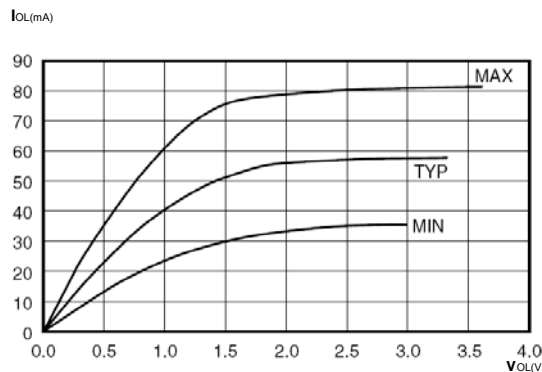
(b) I_{OH} vs. V_{OH}



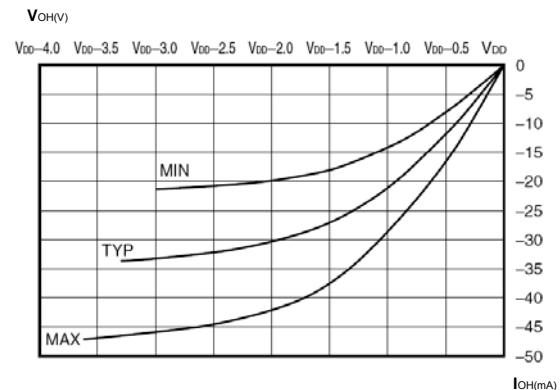
9mA output driver:

(typical block type: TDOPAC33NN09, TDOPAC33NL09)

(a) I_{OL} vs. V_{OL}

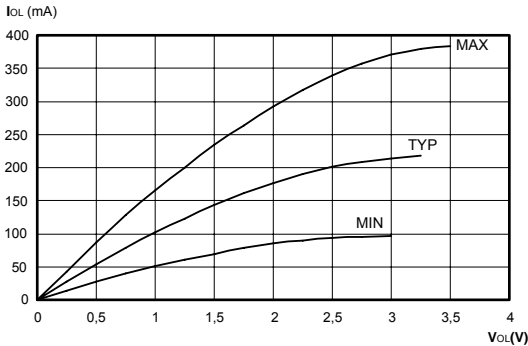


(b) I_{OH} vs. V_{OH}

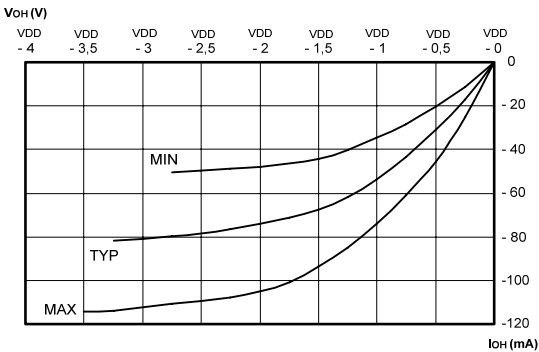


Host interface (18mA) output driver:
(typical block type: TDBIAPCUNLP36C)

(a) I_{OL} vs. V_{OL}



(b) I_{OH} vs. V_{OH}



5.4.2 System Oscillator

The system oscillator circuit along with the internal PLL, generates all internal clocks of the netX 500/100 chip. For clock generation, either a quartz crystal (oscillation mode: fundamental) with the internal oscillator circuit or a quartz oscillator connected to the clock input pin may be used.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
OSC_VDDC	Oscillator power supply core		1.425	1.5	1.65	V
V _{IL}	Input voltage, low	Note 1)	0		0.8	V
V _{IH}	Input voltage, high	Note 1)	2.0		V _{DDIO} + 0.5	V
V _{OL}	Output voltage, low	Note 1)			0.4	V
V _{OH}	Output voltage, high	Note 1)	2.0	3.0	V _{DDIO}	V
f _{CLK}	System clock frequency			25		MHz
	System clock tolerance		-100		+100	ppm
	System clock duty cycle		40	50	60	%
	System jitter tolerance			20		ps [RMS]
T _{CYC}	System clock cycle time			40		ns
T _{high}	System clock high time		14		20	ns
T _{low}	System clock low time		14		20	ns
	System clock slew rate		0.5			V/ns
C _{IN}	Pin capacitance	input buffer	2.5	3.5	4.5	pF
C _{OUT}	Pin capacitance	output buffer	2.7	3.7	4.7	pF
OSC_I _{VDDC}	Oscillator current	1.5 V			6	mA

Notes:

- 1) These values are DC parameters, that do not apply to the dynamical system of a crystal circuit. When using crystals, the circuit should be designed in a way that keeps the levels within the absolute maximum ratings (-0,5V to Vddio + 0,5V).

5.4.3 RTC Oscillator

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
RTC_V _{DDC}	RTC power supply core		1.425	1.5	1.65	V
RTC_V _{DDIO}	RTC power supply I/O		1.6 ^{Note 1}	3.3	3.6	V
RTC_I _{VDDIO}	RTC IO current	Normal operation		50		μA
		Power Down Buffered mode		4		
RTC_I _{VDDC}	RTC core current			< 0.1		μA
V _{IL}	Input voltage, low		0		0.8	V
V _{IH}	Input voltage, high		2.0		V _{RTC_VDDIO} + 0.5	V
V _{OL}	Output voltage, low				0.4	
V _{OH}	Output voltage, high			RTC_V _{DDIO} - 0.3		
f _{RTC}	RTC clock frequency			32.768		kHz
ESR _{crystal}	series resistance crystal			50	80	kΩ
C _{IN}	Pin capacitance	input buffer	2.5	3.5	4.5	pF
C _{OUT}	Pin capacitance	output buffer	2.7	3.7	4.7	pF

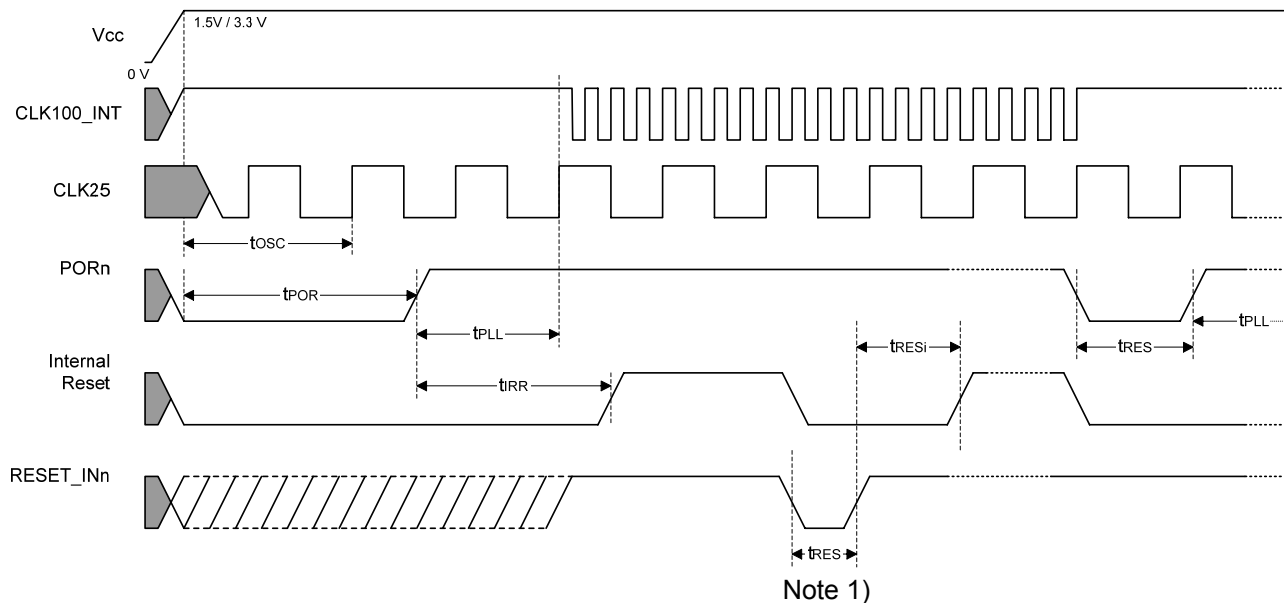
Note:

- 1) This value is only evaluated by some applications and can not be guaranteed.

5.4.4 Power On Reset / Reset Input

The netX provides two reset inputs, one asynchronous reset input with a Schmitt-trigger input cell that is to be used for the mandatory Power On Reset of the chip and one synchronous reset input that can be used to perform a system reset on the powered up and running chip.

The picture and table below show the behavior of the reset system and the timing requirements for the reset input signals.



Parameter	Description	Value	Dimension
t_{OSC}	Startup time for the internal oscillator system	0...10	ms 2)
t_{POR}	Minimum duration of external Power On Reset signal	> t_{OSC}	2)
t_{PLL}	Time between deassertion of $PORn$ and release of internal PLL	10,5	ms
t_{IRR}	Time between deassertion of $PORn$ and release of internal Reset	11,5	ms
t_{RES}	Minimum Reset pulse width for running chip	10	ns
t_{RESi}	Active time of internal Reset after releasing $RESET_INn$	640	ns (64 clks)

Notes:

- 1) The use of $RESET_INn$ is optional and shown in this diagram for reference only. It is not a required part of the startup reset sequence.
- 2) The Oscillator startup time depends on environment temperature and crystal type. The given values are typical values that have been evaluated with a certain crystal and may differ from the values experienced with the user's application. The external reset signal source (reset generator) may release the $PORn$ signal before the System oscillator has settled completely, however the values for t_{PLL} and t_{IRR} may diverge from the values above in that case, since they derive from counter values and the counters are clocked by the system oscillator.

5.4.5 USB

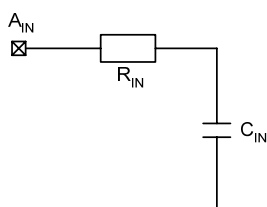
Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
USB_V _{DDC}	USB power supply, core		1.425	1.5	1.65	V
USB_V _{DDIO}	USB power supply, IO		3.0	3.3	3.6	V
USB_I _{VDDC}	USB core current	1.5 V			5	mA
USB_I _{VDDIO}	USB IO current	3.3 V			5	mA
V _{IH}	Input-high voltage	single end	2.0		V _{DDIO} +0.5	V
V _{IL}	Input-low voltage	single	0		0.8	V
V _{DI}	Differential input voltage	0.8 V < V _{IN} < 2.5 V	200			mV
V _{OH}	Output-high voltage	ext. 15 k Ω pull-up to V _{DDIO}	2.8		V _{DDIO}	V
V _{OL}	Output-low voltage	ext. 1.5 k Ω pull-down to GND	0		0.3	V
C _{PIN}	Pin capacitance		5		7	pF

5.4.6 ADC

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
ADC_VDD	Power supply digital		1.425	1.5	1.65	V
ADC_VREFP	Ref. voltage high		3.0	3.3	3.6	V
ADC_VREFM	Ref. voltage low		0	0	0	V
ADC_VDDIO	Power supply analog		3.0	3.3	3.6	V
V _{AIN}	Analog input voltage range		ADC _{VREFM}		ADC _{VREFP}	V
ADC_I _{VDDIO}	Current consumption Note 1)	one ADC	0.3	0.6	1.2	mA
ADC_I _{REF}	Reference current Note 2) Between ADC_VREFP - ADC_VREFM			60	150	μA
RES	Resolution		10			Bits
t _C	Conversion time		1			μs
f _C	Conversion rate				1	Msp/s
t _{AIN}	Setup time SEL[2:0] to conversion Note 3)		80			ns
DLE	Differential linearity error	V _{AIN} = 0 to AVDD		± 0.3 LSB	± 0.7 LSB	
ILE	Integral linearity error	V _{AIN} = 0 to AVDD		± 0.5 LSB	± 1.5 LSB	
ZSE	Zero scale error			± 0.5 LSB	± 1.0 LSB	
FSE	Full scale error			± 0.5 LSB	± 1.0 LSB	
A _{IN}	Analog input channels Note 4)		2 x 4 channels multiplexed			
R _S	Signal source impedance				1	kΩ
C _{IN}	Equivalent analog input capacitance Note 5)			9	11	pF
R _{IN}	Equivalent analog input resistance Note 5)			0.5	0.8	kΩ
	Pipeline delay (Latency)				16	Cycles

Notes:

- 1) Does not include the current flowing to the AVREF pin (I_{REF}). All current = I_{DD} + I_{REF}
- 2) DC current does not flow between AVREFP and AVREFM. This value is charge and discharge current of C array which flows when AD conversion.
- 3) When changing the channel setting before a conversion, a settling time of t_{AIN f} for the input MUX should be allowed before starting conversion.
- 4) The ACD part includes two independent converters with integrated 8-to-1 multiplexers. Only channels 0, 2, 4 and 6 of each ADC are wired to a pin.
- 5) Equivalent Circuit:



5.4.7 PHY

The power supplies of the PHYs must always be connected, even when they are not used.

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
PHY_V _{DDCART}	PHY power supply		1.425	1.5	1.65	V
PHY_V _{DDCAP}	PHY central analog power supply		1.425	1.5	1.65	V
PHY_V _{DDIOAC}	PHY central analog power supply		3.0	3.3	3.6	V
PHY_V _{DDIOAT}	PHY analog test power supply		3.0	3.3	3.6	V
P _{10BT}	for both Phy's	1.5 V		120	140	mW
		3.3 V		420	500	mW
P _{100BT}	for both Phy's	1.5 V		230	270	mW
		3.3 V		260	300	mW
P _{100FX}	for both Phy's	1.5 V		100	120	mW
		3.3 V		5	6	mW
PHY_R _{EXTRES}	Reference resistor	Must always be connected, 12.4 kΩ / 1%				
PHY_I _{VDDC}	PHY core current, includes I _{VDDCART} and I _{VDDCAP}	1.5 V	2	150	180	mA
PHY_I _{VDDIO}	PHY IO current, includes I _{VDDIOAC} and I _{VDDIOAT}	3.3 V	1	130	150	mA

Max. power and current values apply at max. voltage conditions

100BASE-TX:

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V _{100OUTH}	TX Output, High Level Differential Signal, TXP/TXN		0.95		1.05	V
V _{100OUTL}	TX Output, Low Level Differential Signal, TXP/TXN		-0.95		-1.05	V
V _{100OUTM}	TX Output, Mid. Level Differential Signal, TXP/TXN		-0.05		+0.05	V
V _{OVS}	TX Output, Overshoot Differential Signal, TXP/TXN		0		5	V
V _{100INTHON}	RX Input, Turn-on ThresholdLevel Differential Signal, RXP/RXN				1.0	V
V _{100INTHOFF}	RX Input Turn-off ThresholdLevel Differential Signal, RXP/RXN		0.20			V
t _r	Rise time, TXP/TXN		3		5	ns
t _f	Fall time, TXP/TXN		3		5	ns
	Duty cycle distortion, TXP/TXN		0		0.5	ns(pp)
	Transmit Jitter, TXP/TXN		0		1.4	ns(pp)

These specs are compliant with ANSI/IEEE802.3 Std.

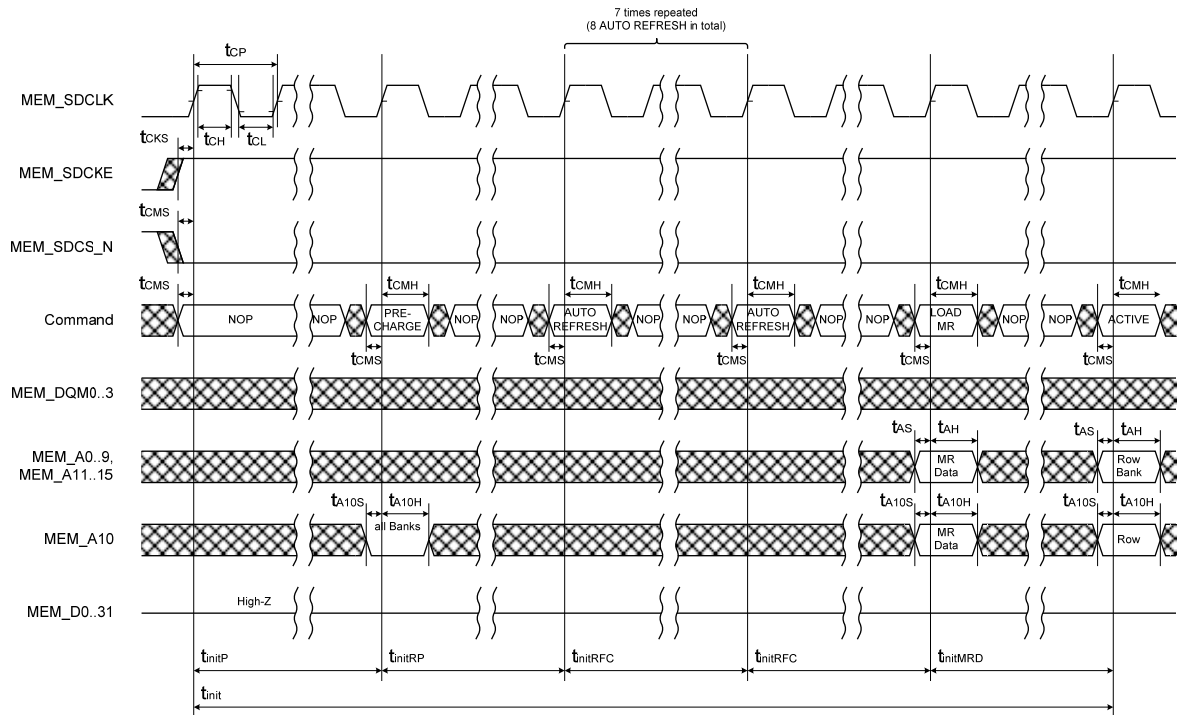
10BASE-T:

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V _{10OUT}	TX Output Amplitude Differential Signal, TXP/TXN		2.2		2.8	V
V _{10INTH}	RX Input Threshold Level Differential Signal, RXP/RXN		0.30		0.585	V

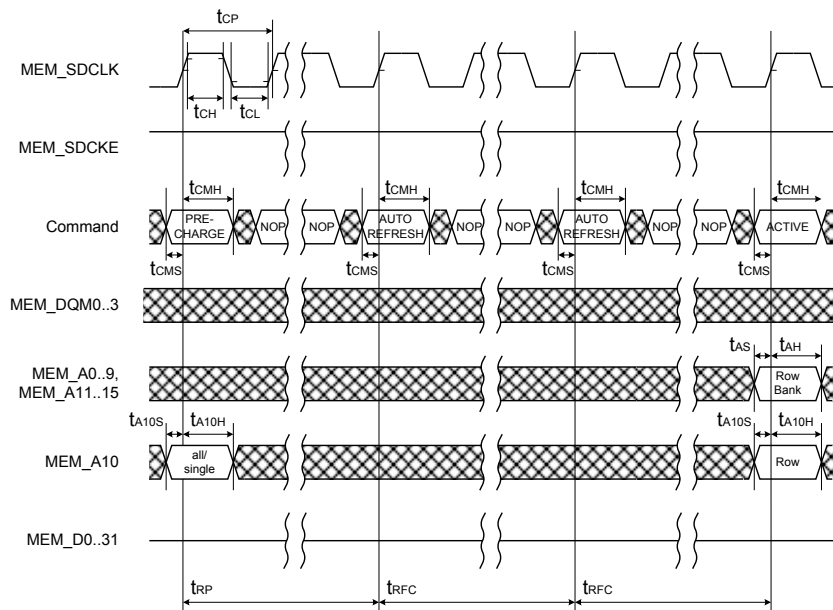
These specs are compliant with ANSI/IEEE802.3 Std.

5.4.8 SDRAM

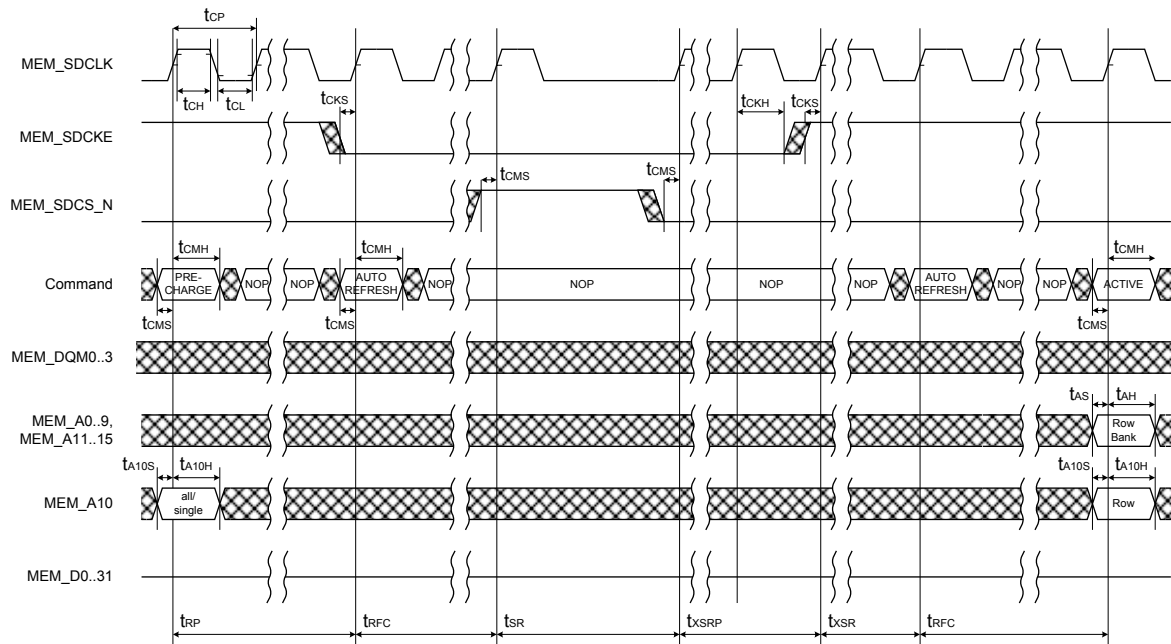
Initialization



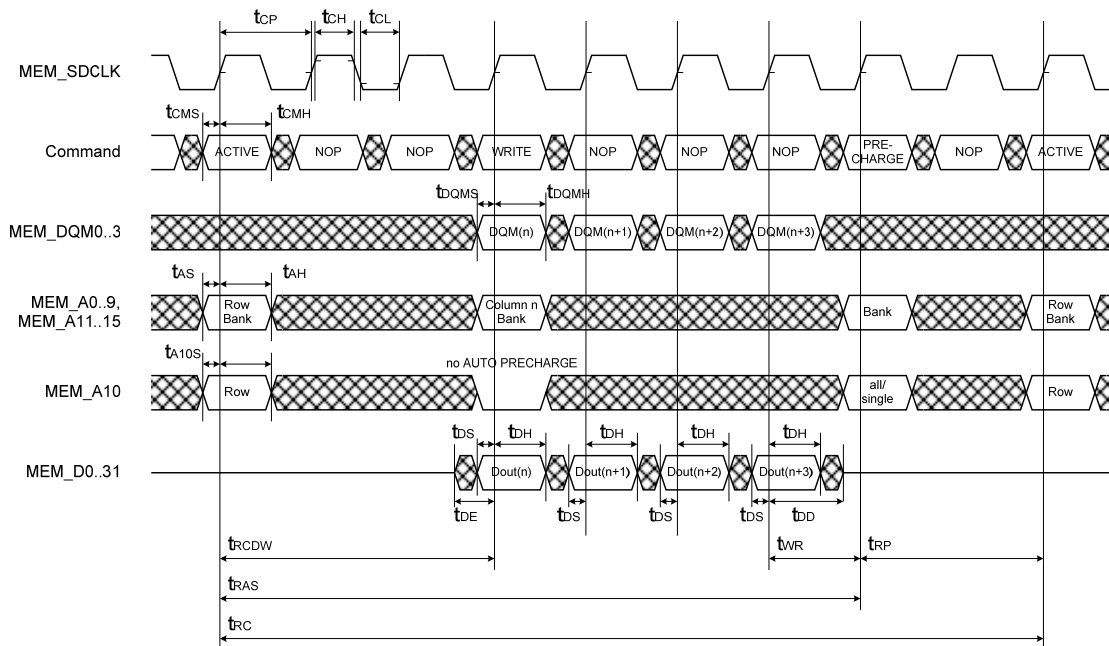
Auto refresh cycles



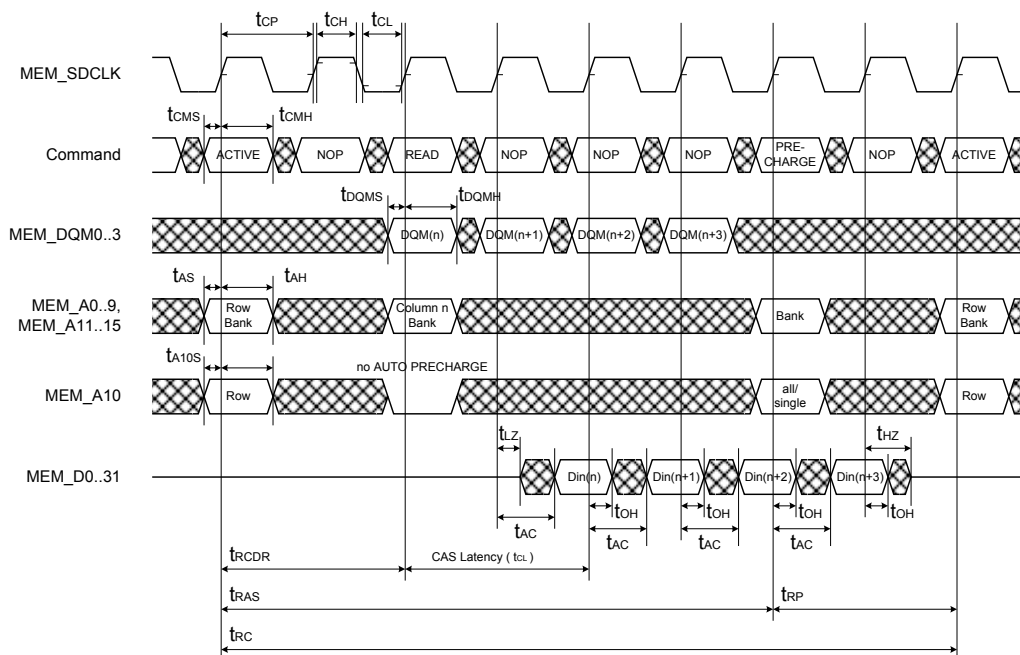
Self refresh mode, entry and exit



Write access



Read access



Electrical Characteristics of netX 500/100 SDRAM Memory Interface Part are related to following external capacitive loads:

IOs	Description	Load C_L	Unit
MEM_D0..31	Memory Interface Data Lines	50	pF
MEM_A0..23	Memory Interface Address Lines	50	pF
MEM_DQM0..3	Memory Interface Data Qualifier Mask	50	pF
MEM_SDCLK	SDRAM Clock	25	pF
MEM_SDCS_N MEM_SDWE_N MEM_SDRAS_N MEM_SDCAS_N MEM_SDCKE	SDRAM Control Signals	25	pF
MEM_MCS0..2_N MEM_MOE_N MEM_MWE_N	SRAM/FLASH Control Signals	25	pF

Following Table shows detailed IO timing of netX 500 /100 memory Interface. All Timings are related to the following SDRAM clock phase settings (to be done by programming *sdram_timing_ctrl* register):
mem_sdclk_phase: 3 *data_sample_phase*: 2

Symbol	Parameter	Min⁽¹⁾	Typ	Max⁽¹⁾	Unit
t _{CP}	SDRAM Clock (MEM_SDCLK) cycle time	10.0 ⁽²⁾	10.0 ⁽²⁾	⁽²⁾	ns
t _{CH}	SDRAM Clock (MEM_SDCLK) high level width	3.5		4.7	ns
t _{CL}	SDRAM Clock (MEM_SDCLK) low level width	3.8		4.8	ns
t _{CKS}	Clock Enable (MEM_SDCKE) setup time	5.0 ⁽³⁾		5.5 ⁽³⁾	ns
t _{CKH}	Clock Enable (MEM_SDCKE) hold time	2.3 ⁽³⁾		3.1 ⁽³⁾	ns
t _{CMS}	Command setup time MEM_SDCS_N MEM_SDWE_N MEM_SDRAS_N MEM_SDCAS_N	5.2 4.7 4.9 5.0		5.6 5.6 5.5 5.5	ns
t _{CMH}	Command hold time MEM_SDCS_N MEM_SDWE_N MEM_SDRAS_N MEM_SDCAS_N	2.1 2.4 2.4 2.3		3.0 3.1 3.2 3.1	ns
t _{AS}	Address (without MEM_A10) setup time	3.2		4.6	ns
t _{AH}	Address (without MEM_A10) hold time	2.9		3.6	ns
t _{A10S}	MEM_A10 setup time	3.4		5.7	ns
t _{A10H}	MEM_A10 hold time	3.1		3.7	ns

t_{DQMS}	Data Qualifier Mask (MEM_DQM*) setup time	3.4		5.7	ns
t_{DQMH}	Data Qualifier Mask (MEM_DQM*) hold time	2.3		3.6	ns
t_{DS}	netX Data-out setup time	2.1		5.3	ns
t_{DH}	netX Data-out hold time	2.9		3.6	ns
t_{DE}	netX Data-out enable (Low Z) time	2.5		5.5	ns
t_{DD}	netX Data-out disable (High Z) time	3.4		3.9	ns
t_{AC}	SDRAM Access time			7.5	ns
t_{OH}	SDRAM Data-out hold time	0.0			ns
t_{LZ}	SDRAM Data-out enable (Low Z) time	0.0			ns
t_{HZ}	SDRAM Data-out disable (High Z) time			14.9	ns

Notes:

1. netX SDRAM clock generation is compensating operating condition effects. Hence MIN timing does not appear necessarily at MIN operating conditions (high voltage, low temperature) and MAX timing does not appear necessarily at MAX operating conditions (low voltage, high temperature).
2. MEM_SDCLK is always running at the same frequency as netX internal system clock.
3. MEM_SDCKE is only used for SDRAM power down (Self Refresh mode or SDRAM disabled).

Timing Characteristics configurable by *sdrām_timing_ctrl* configuration register:

Symbol	Parameter	Min	Typ	Max	Unit
t_{RAS}	ACTIVE to PRECHARGE time	3..10			t_{CP}
t_{RC}	ACTIVE to ACTIVE (same bank) time	4..13 ⁽¹⁾			t_{CP}
t_{RCDR}	ACTIVE to READ (same bank) time	1..3 ⁽²⁾			t_{CP}
t_{RCDW}	ACTIVE to WRITE (same bank) time	2..4 ⁽²⁾			t_{CP}
t_{REFI}	Average periodic REFRESH interval (Refresh period divided by rows to refresh within refresh period)	3.9 ⁽³⁾	7.8 ⁽³⁾ 15.6 ⁽³⁾	31.2 ⁽³⁾	us
t_{RFC}	AUTO REFRESH period	4..19			t_{CP}
t_{RP}	PRECHARGE command period	1..3			t_{CP}
t_{RRD}	ACTIVE to ACTIVE different bank time	1..3 ⁽⁴⁾			t_{CP}
t_{WR}	WRITE recovery time	1..3			t_{CP}
t_{SR}	Self Refresh period	1			t_{CP}
t_{XSRP}	Self Refresh Clock active to exit period exit	4			t_{CP}
t_{XSR}	Self Refresh exit to command period	4..19 ⁽⁵⁾			t_{CP}
t_{CL}	CAS Latency	2		3	t_{CP}

Notes:

1. Minimum t_{RC} is $t_{RAS} + t_{RP}$.
2. ACTIVE to WRITE (same bank) time is ACTIVE to READ (same bank) time plus 1 additional cycle from memory controller internal arbitration.
3. Expecting 100MHz system clock.
4. netX 500/100 SDRAM Controller uses t_{RCDR} for t_{RP} .
5. For t_{XSR} programmed t_{RFC} settings will be taken.

SDRAM Initialisation and Power Up Timing Characteristics:

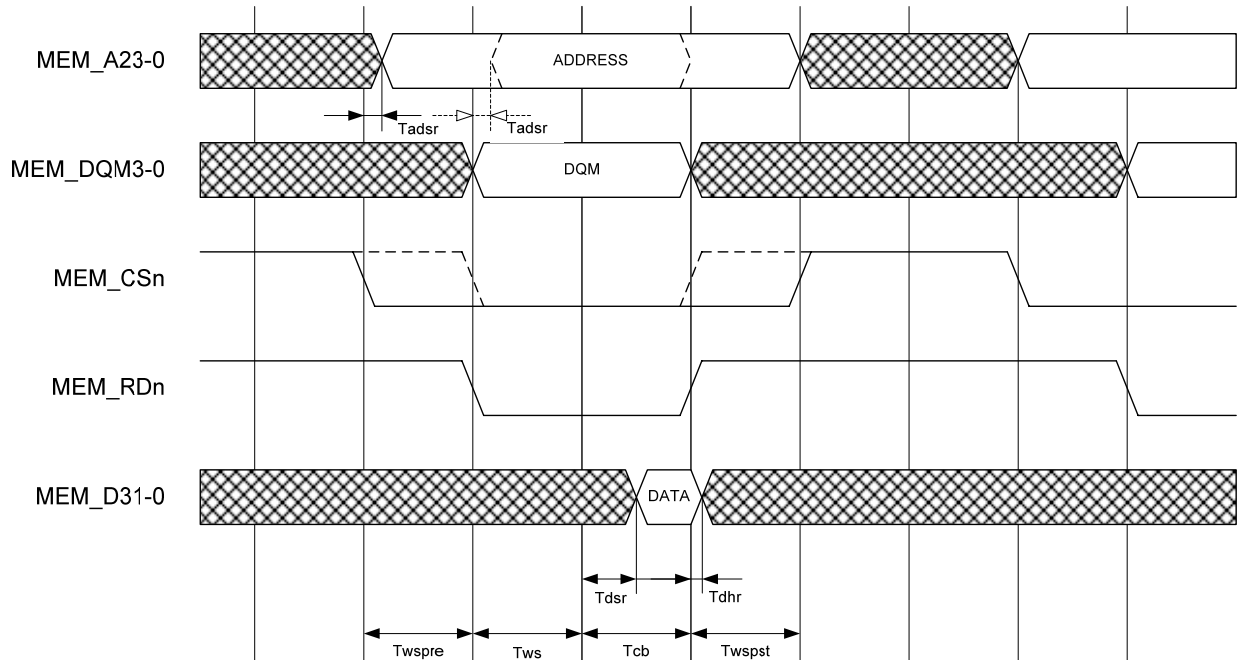
Symbol	Parameter	Min	Typ	Max	Unit
t_{init}	Whole SDRAM initialisation time till first ACTIVE	20317		20317	t_{CP}
t_{initP}	SDRAM Power Up time (NOP till first PRECHARGE)	20050		20050	t_{CP}
t_{initRP}	Initialisation PRECHARGE command period	16		16	t_{CP}
$t_{initRFC}$	Initialisation AUTO REFRESH period	23		32	t_{CP}
$t_{initMRD}$	Load MR to first command period	4 ⁽¹⁾			t_{CP}

Notes:

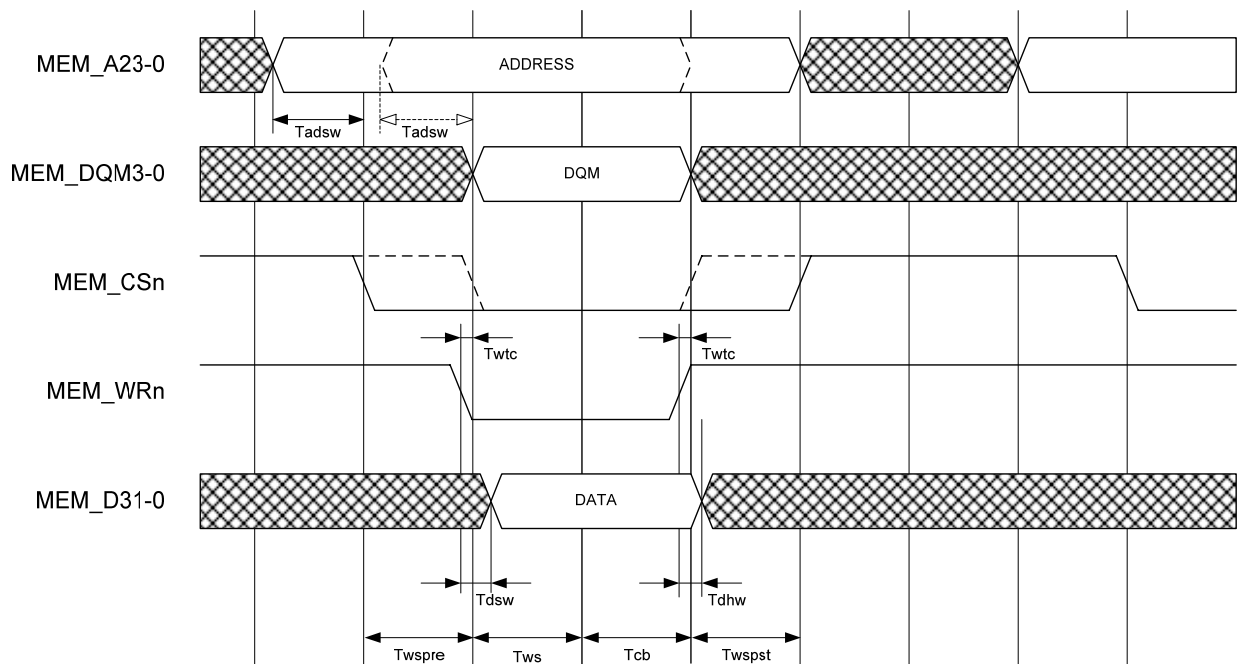
1. If longer $t_{initMRD}$ time is required, software can wait until *sdrām_ready* flag is set in *sdrām_general_ctrl* register before running first SDRAM memory access.

5.4.9 SRAM / FLASH

Read access:



Write access:



Parameter	Notes	min	max	Unit
Tadsr (address setup time, read): Time between assertion of CSn and valid address	1)	1	2	ns
Tadsw (address setup time, write): Time between valid address and assertion of CSn	1)	8	9	ns
Tdsr (data setup time, read): Time between assertion of RDn (+ Tws) and valid data from memory	1), 3)	0	7.8	ns
Tdsw (data setup time, write): Time between assertion of WRn and valid data to memory	-	1.3	2.4	ns
Tdhw (data hold time, write): Time between deassertion of WRn and deactivation of output buffers	-	Twtc (min)	Twtc (max)	ns
Tdhr (data hold time, read): Required hold time of memory data after deassertion of RDn	1), 3)	0	-	ns
Twtc (write to chip select time)	2)	0.05	0.2	ns
Twspre: pre cycle waitstates, configurable in 10ns steps	-	0	30	ns
Tws: waitstates, configurable in 10ns steps	-	0	310	ns
Twspst: post cycle waitstates, configurable in 10ns steps	-	0	30	ns

Notes:

- 1) All timings are based on the following load conditions:

- control signals (RDn, WRn, CSn, DQM) loaded with 25 pF
- address and data signals loaded with 50 pF

Since the significant factor in signal delays is the pad cell delay rather than the internal signal path delay, the actual timing is highly dependant on the actual capacitive signal load (see also table on next page).

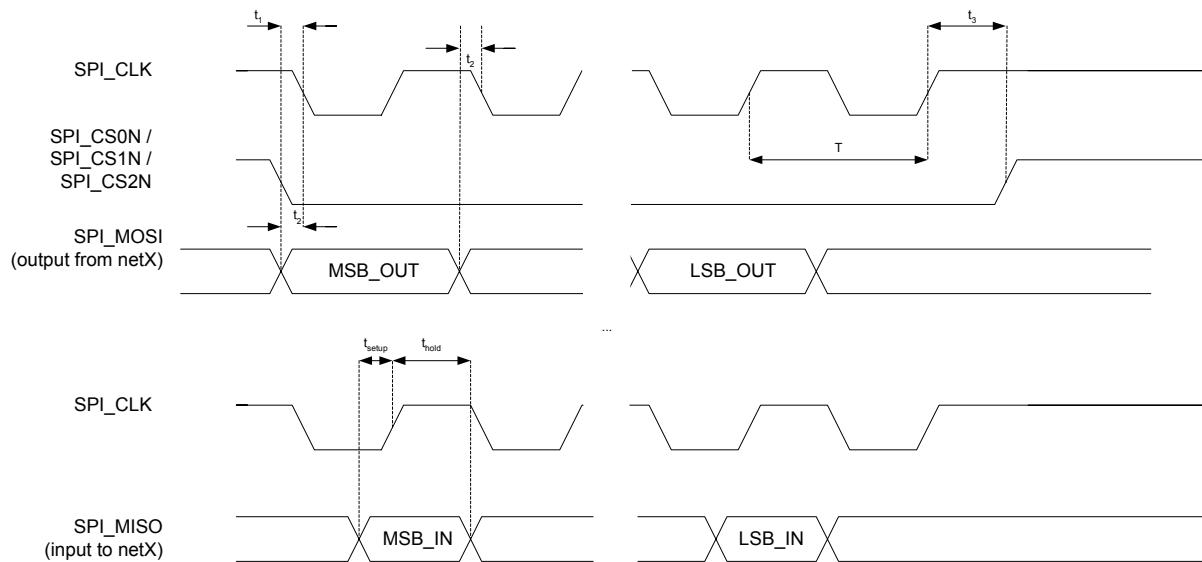
- 2) **Twtc** reflects the special clock path the write signal has been assigned to, to allow this signal (which causes data sampling in the accessed memory on its rising edge) to become inactive earlier than all other signals. While CSn and address will be extended by post cycle waitstates, data lines will always be deactivated with the write signal. Hence, scenarios, putting a capacitive load to WRn that is considerably high (WRn delay increases) while at the same time putting a considerably low capacitive load to the data lines (effective data hold time decreases), should be avoided, but are however very unlikely anyway (data inputs of memories usually have a higher input capacity than their control signal inputs)
- 3) Internal data sampling during read accesses takes place on the next rising edge of the internal 100MHz system clock after assertion of the read signal (no waitstates, Tws = 0) or after the last waitstate cycle (Tws) has ended. Due to the signal delay of the external RDn signal, the allowed maximum data setup time is shorter than 10ns; on the other hand, sampling has already happened on deassertion of the external RDn, hence the required data hold time is 0. (please also mind the delay between assertion of RDn and valid address, which will lead to a lower effective setup time in 0 waitstate scenarios, as with common memory components, the time from address to valid data is usually longer than the output enable time!)

The following table shows the minimum and maximum pad delays of the two different pad types, used with the memory interface for three different load conditions:

Pad type	Load	min.	max.
I/O (data)	10 pF	1.060 ns	1.875 ns
	25 pF	1.466 ns	2.627 ns
	50 pF	2.661 ns	4.823 ns
Output (control and address)	10 pF	0.895 ns	1.573 ns
	25 pF	1.464 ns	2.621 ns
	50 pF	2.402 ns	4.349 ns

5.4.10 SPI

In this example is CPOL=0 and CPHA=0: The master sets the output line on falling edge of SPI_CLK and the slave samples on rising edge of SPI_CLK.

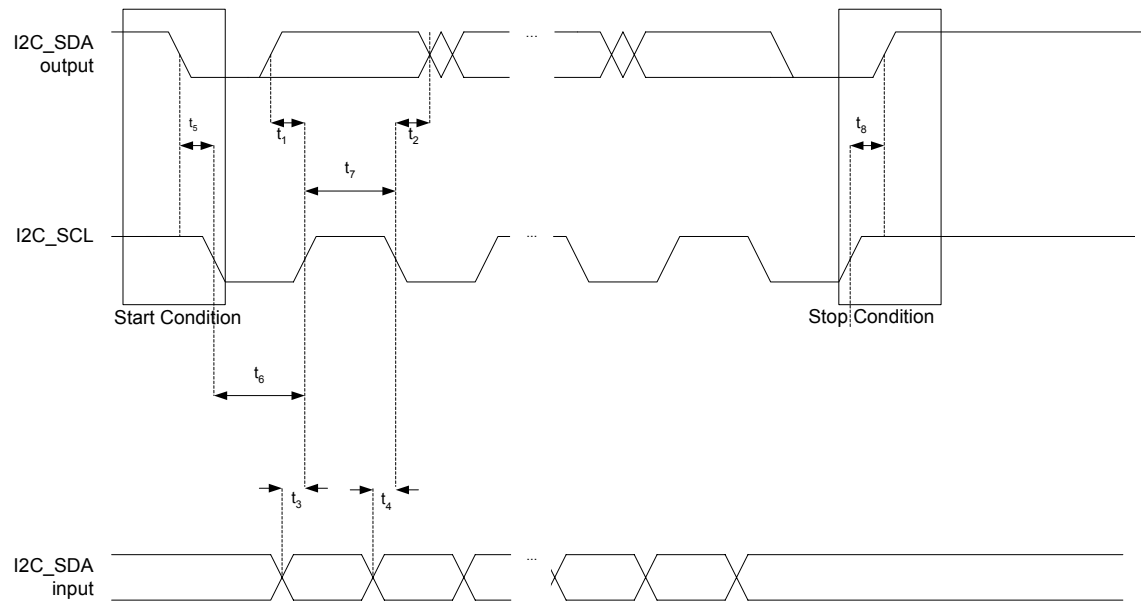


Symbol	Description	Condition	Min	Max	Unit
t_1	Chip select to clock delay ^{Note 1)}	$C_L = 10 \text{ pF}$	0	1 ^{Note 1)}	ns
t_2	Data out to SPI clock delay ^{Note 1)}	$C_L = 10 \text{ pF}$	0	1 ^{Note 1)}	ns
t_3	SPI clock inactive to chip select disabling ^{Note 1)}	$C_L = 10 \text{ pF}$	0	1 ^{Note 1)}	ns
t_{setup}	Setup Time		15		ns
t_{hold}	Hold Time		10		ns
T	SPI Cycle Time, programmable		40	20000	ns

Note:

- 1) All timing values are calculated for a load capacitance of 10 pF for each signal line. When a signal line has more load capacitance the signal delay increases by 0.1 ns/pF load capacitance for this signal line.

5.4.11 I²C



The following timing specification is for fast mode (400 kHz). The netX supports data rates up to 1 MHz.

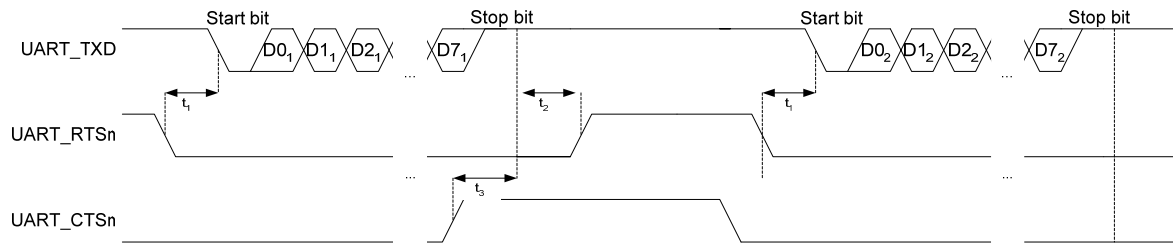
Symbol	Description	Condition	Min	Max	Unit
t_1	Data setup time Note 1)	Fast mode $f = 400 \text{ kHz}$ $C_L = 400 \text{ pF}$	1.23		μs
t_2	Data hold time		10		ns
t_3	Data setup time for posedge clock		10		ns
t_4	Data setup time for negedge clock		10		ns
t_5	Hold time for start condition Note 1)		1.24		μs
t_6	Low period of the clock Note 1)		1.2		μs
t_7	High period of the clock Note 1)			1.3	μs
t_8	Setup time for stop condition Note 1)		1.25		μs

Note:

- 1) The delay of each output signal depends on the connected load capacitance. The delays can be estimated by the following formula:
 $t_{\text{delay}} = 0.075 \cdot C_L \cdot \text{ns/pF} \quad (\text{ns})$, C_L : Load Capacitance (pF)

The timing values above are based on 400 pF load capacitance.

5.4.12 UART



Symbol	Description	Condition	Min	Max	Unit
t ₁	Programmable leading time	Bit Time	0	255	Bit Times
		System Time	0	2.55	µs
t ₂	Programmable trailing time	Bit Time	0	255	Bit Times
		System Time	0	2.55	µs
t ₃	Setup Time before the end of stop bit		70		ns

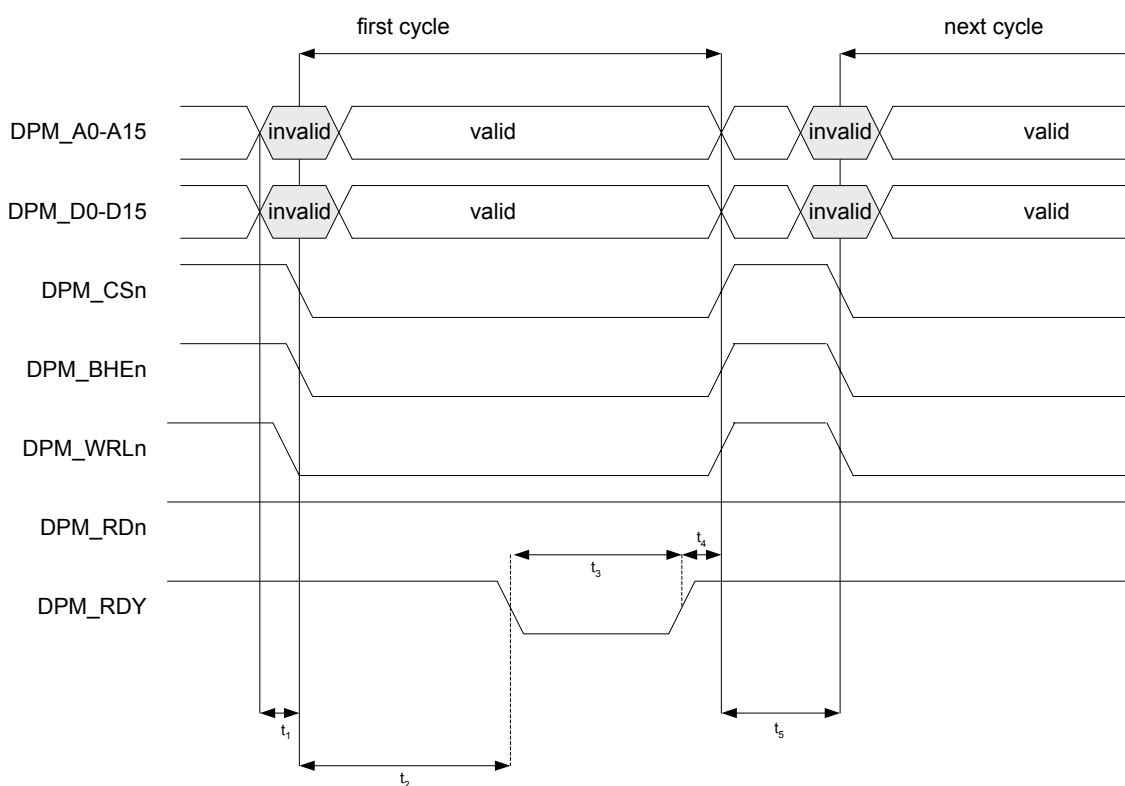
Note:

This example has the following settings:

- 1 Start Bit
- 1 Stop Bit
- 8 Data Bits
- Mode = '1'

5.4.13 Dual-port memory

Intel 16 bit write

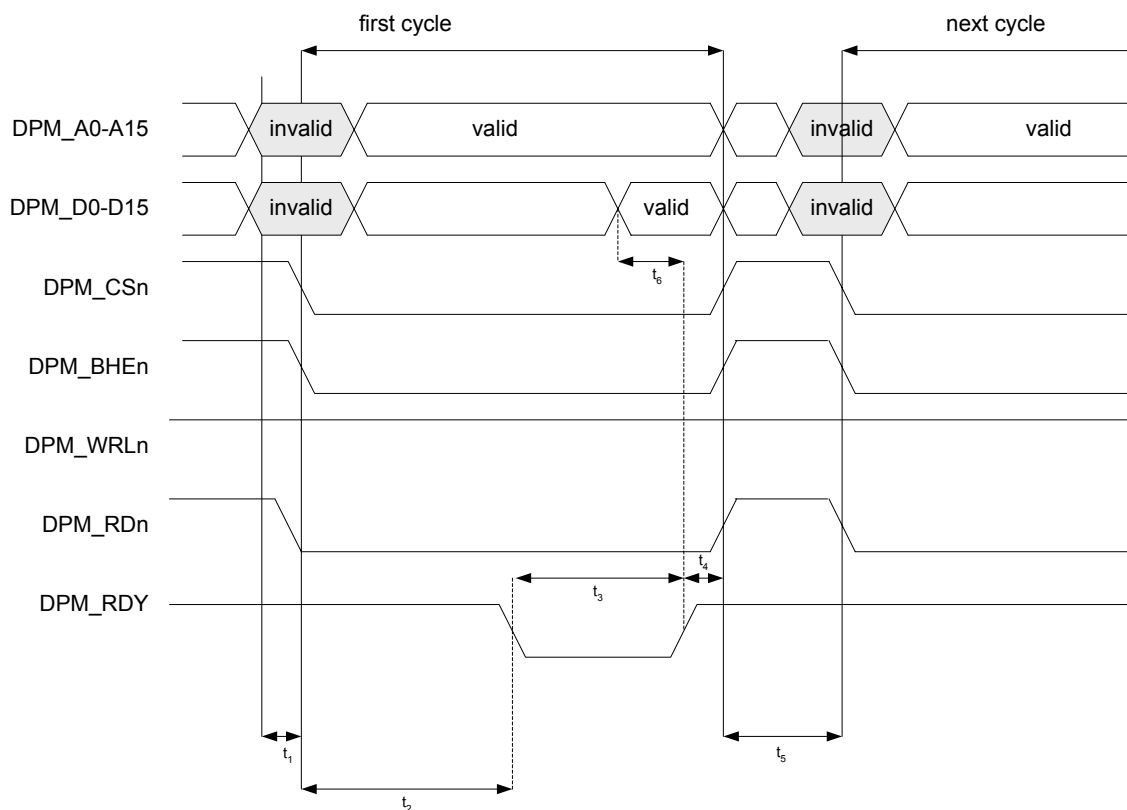


Symbol	Description	Condition	Min	Max	Unit
t ₁	The cycle starts when all control signals are active.		-5	5	ns
t ₂	Time from cycle start until DPM_RDY is active		10	30	ns
t ₃	Busy signal active time duration	Note 1)		80	ns
t ₄	When DPM_RDY goes high the cycle can be finished.		0		ns
t ₅	Time between two cycles		10		

Note:

- 1) The maximum time depends on the accessed memory area. The stated value only applies to DPM accesses targeting internal RAM. For accesses targeting host interface registers, add two more clock cycles (20ns). Max. value for accesses targeting SDRAM can not be specified (-> use of DPM_RDY signal is mandatory).
For maximum performance, the DPM_RDY signal should always be evaluated by the host CPU.
For write accesses, the DPM_RDY active time is usually significantly shorter than for read accesses, as the write value will be buffered by the DPM interface and then written to its actual memory destination.

Intel 16 bit read



Symbol	Description	Min	Max	Unit
t ₁	The cycle starts when all control signals are active	-5	5	ns
t ₂	Time from cycle start until DPM_RDY is active	10	30	ns
t ₃	Busy signal active time duration Note 1)		80	ns
t ₄	When DPM_RDY goes high the cycle can be finished.	0		ns
t ₅	Time between two cycles	10		ns
t ₆	Time from data signals valid until DPM_RDY goes high Note 2)	0	30	ns

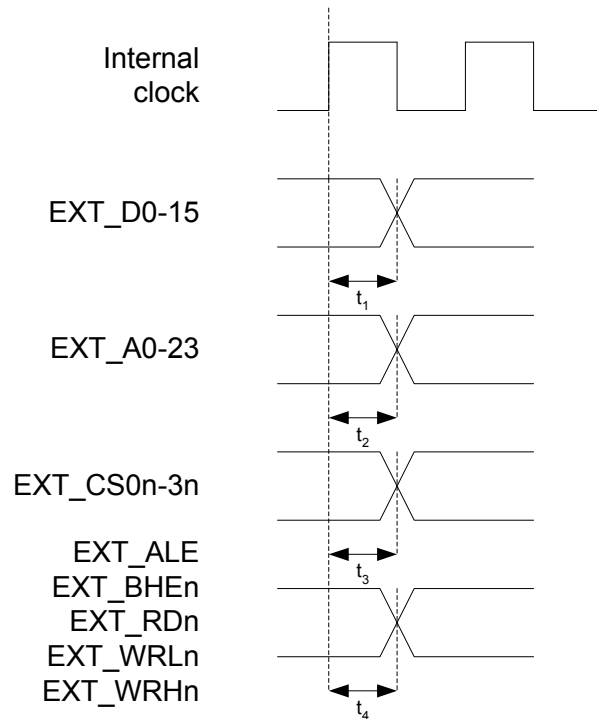
Note:

- 1) The maximum time depends on the accessed memory area. The stated value only applies to DPM accesses targeting internal RAM (with t₆ set to 0). For accesses targeting host interface Registers, add two more clock cycles (20ns). Max. value for accesses targeting SDRAM can not be specified (-> use of DPM_RDY signal is mandatory).
For maximum performance, the DPM_RDY signal should always be evaluated by the host CPU. For write accesses, the DPM_RDY active time is usually significantly shorter than for read accesses, as the write value will be buffered by the DPM interface and then written to its actual memory destination
- 2) t₆ is configurable (0/10/20/30 ns), increasing t₆ also increases t₃ accordingly.

5.4.14 Extension bus

Output Timing:

All output signals of the extension bus interface change with the rising edge of the internal 100 MHz system clock and are hence synchronous. However, the output delays are slightly different and depend on the load capacitance, connected to the outputs.



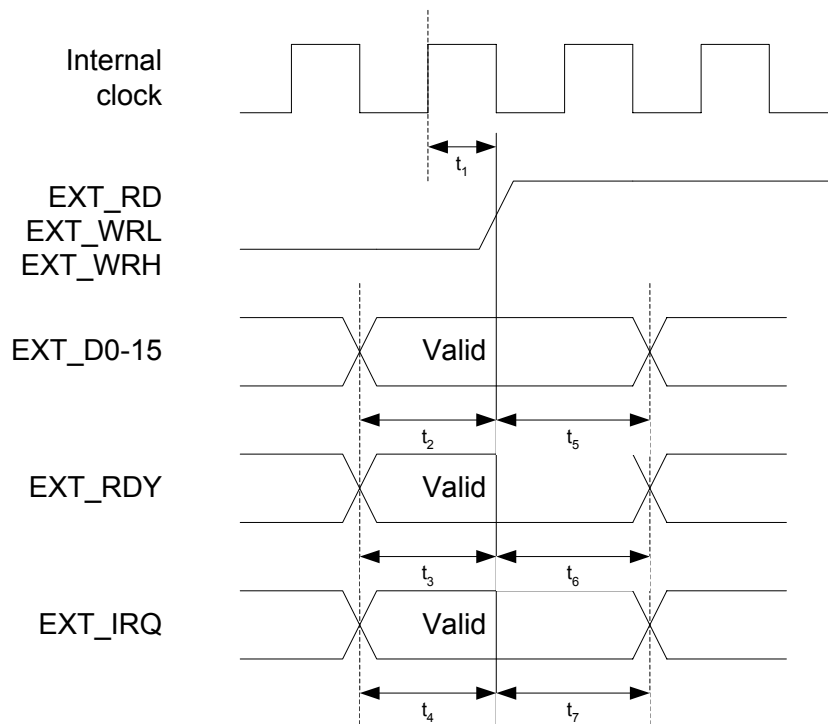
Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
t_1	Clock to data out time	$C_L = 50 \text{ pF}$			6.4	ns
t_2	Clock to address out time	$C_L = 50 \text{ pF}$			6.1	ns
t_3	Clock to chip select out time	$C_L = 50 \text{ pF}$			6.0	ns
t_4	Clock to control signal out time	$C_L = 50 \text{ pF}$			6.4	ns

Note:

The output delay of each line increases by approx. 0.2 ns/pF load capacitance

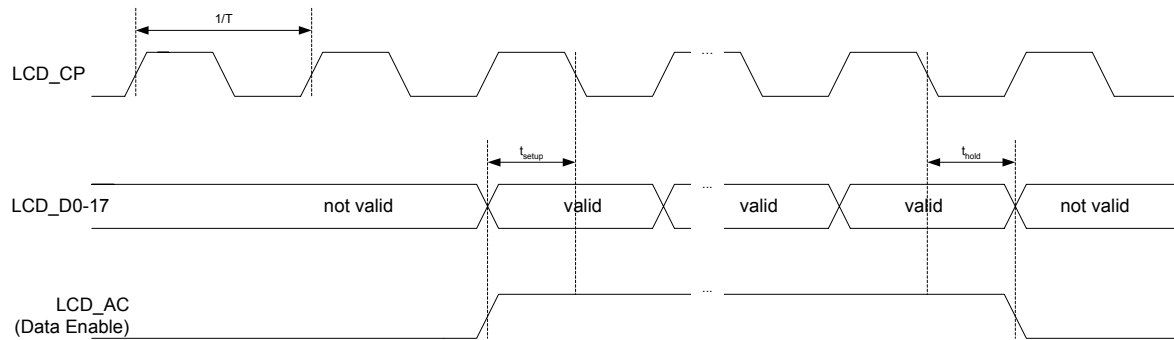
Input Timing:

All input signals of the extension bus interface are sampled on the rising edge of the internal 100 MHz system clock.



Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
t_1	Clock to control signal out time	$C_L = 50 \text{ pF}$			6.4	ns
t_2	setup time				$t_1 + 3$	ns
t_3	setup time				$t_1 + 25$	ns
t_4	setup time		0		10	ns
t_5	hold time		0			ns
t_6	hold time		0			ns
t_7	hold time		0			ns

5.4.15 LCD

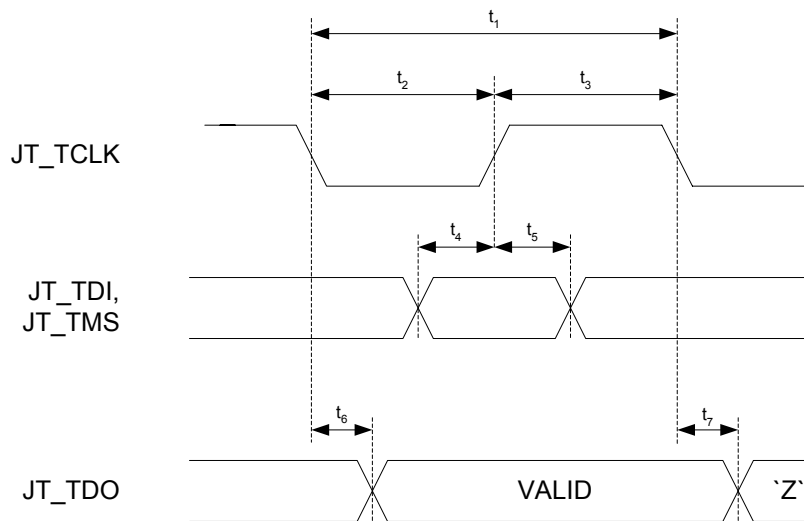


Symbol	Parameter	Min.	Typ.	Max.	Unit
$1/T$	Panel Clock Frequency, programmable	0,097		33 ^{Note1)}	MHz
t_{setup}	Setup Time	$(\frac{1}{2}) * T - 2$	$(\frac{1}{2}) * T$	$(\frac{1}{2}) * T + 2$	ns
t_{hold}	Hold Time	$(\frac{1}{2}) * T - 2$	$(\frac{1}{2}) * T$	$(\frac{1}{2}) * T + 2$	ns

Notes:

- 1) Higher frequencies (up to 100 MHz) are possible but may decrease chip performance.

5.4.16 JTAG



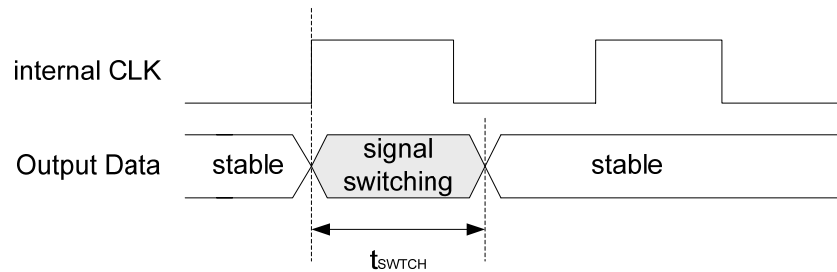
Symbol	Parameter	Min.	Typ.	Max.	Unit
t_1	Clock Period	90			ns
f_{TCLK}	JTAG Clock frequency ^{Note 1)}			11.11	MHz
t_2	Clock Low Time	40			ns
t_3	Clock High Time	40			ns
t_4	Setup Time before rising edge	0			ns
t_5	Hold Time after rising edge	15			ns
t_6	TCLK to TDO delay	12		17	ns
t_7	TCLK to High-Z delay	12		17	ns

Notes:

- 1) The maximum frequency is $CLOCK_{ARM} / 18 = 200 \text{ MHz} / 18 = 11.11 \text{ MHz}$.

5.4.17 Fieldbus / PWM / Encoder

These I/O signals are all synchronized to the internal clock. Because of the software controlled output data flow, only timing delays between the different output signals can be provided.



Symbol	Description	Min	Typ.	Max	Unit
t_{SWITCH}	Switching time between different output signals	-1	0	1	ns

5.5 Failure Rate (FIT)

Due to the comparatively low volume of netX controller production, Failure Rates for the complete chip are not available. However, as a guideline value, the known FIT values (as at December 2012) for the CB-12 process, the netX 50, netX 100 and netX 500 are based on, can be used:

	Failure Rate (FIT), Confidence Level = 60%		
Junction Temperature	Ea = 0.3eV	Ea = 0.5eV (Typ.)	Ea = 0.7eV
55 °C	22	5	1
70 °C	35	11	3
80 °C	47	18	6
90 °C	62	28	12
100 °C	80	42	22
110 °C	101	64	39
125 °C	143	112	86

6 Package Information

6.1 Package Thermal Specification

Absolute maximum junction temperature: $T_{j_max} = 125\text{ °C}$
 Absolute minimum junction temperature: $T_{j_min} = -40\text{ °C}$

Recommended operating ambient temperature:

Operating ambient temperature: $-25\text{ °C} \dots +85\text{ °C}$ @ typ. 1.5 Watt (with heat sink $R_{th} \leq 10\text{ K/W}$)
 Operating ambient temperature: $-25\text{ °C} \dots +70\text{ °C}$ @ typ. 1.5 Watt (without heat sink)

Thermal Characterization (based on 4 Layer PCB (NECEL condition / FR4)):

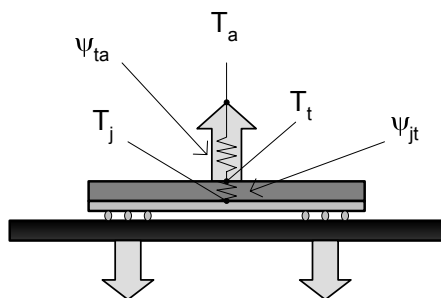
Symbol	Parameter	Air Flow [m/s]				Unit
		0	0.2	1	2	
θ_{ja}	Thermal resistance, junction-to-ambient	25.9	23.4	19.9	18.2	°C/W
ψ_{jt}	Thermal parameter, junction to the top center of the package surface	0.04	0.10	0.20	0.28	°C/W
ψ_{ta}	Thermal parameter, the top center of the package surface to ambient	25.9	23.3	19.7	17.9	°C/W
θ_{jc}	Thermal resistance, junction-to-case	4.26				°C/W

Case 1 - Without external heat sink:

Heat flow path exist not only on package top

$$T_t = T_a + \psi_{ta} \times P_{netX}$$

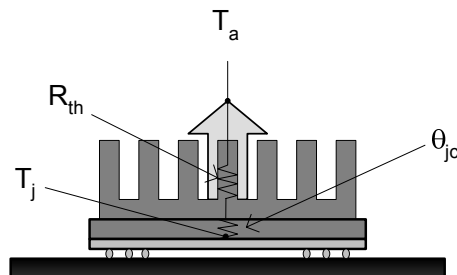
$$T_j \approx T_t$$



Case 2 - With external heat sink:

Package surface dominates heat flow path via heat sink

$$T_j = T_a + (\theta_{jc} + R_{th}) \times P_{netX}$$



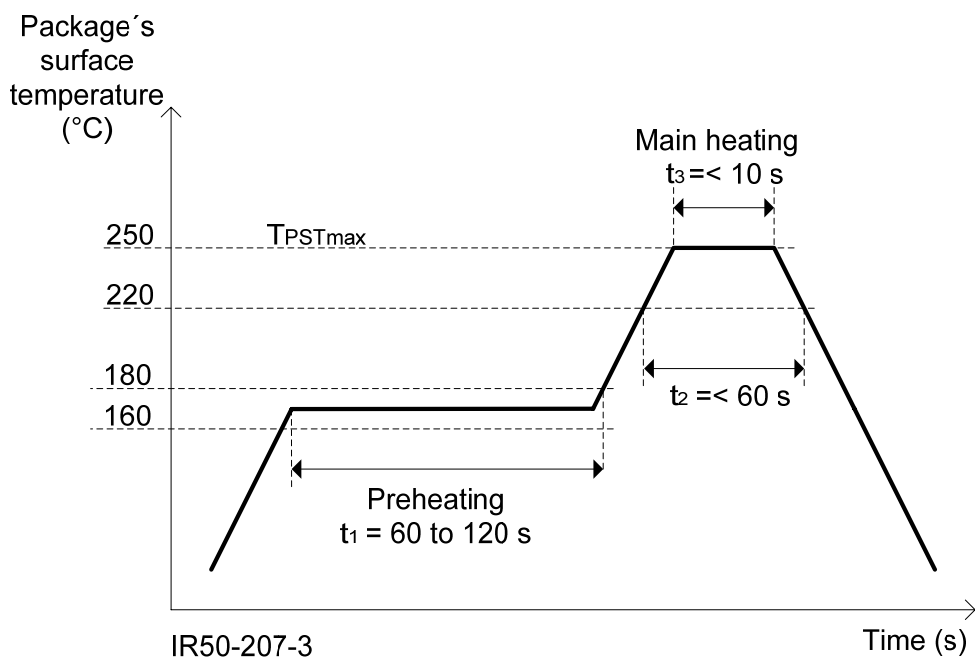
Please also consider the netX heat test report, available at www.hilscher.com (netX – Downloads – Testreports)!

6.2 Soldering Conditions

The following soldering parameters are recommended for infrared reflow soldering. The netX package is suitable for a Pb-free soldering process.

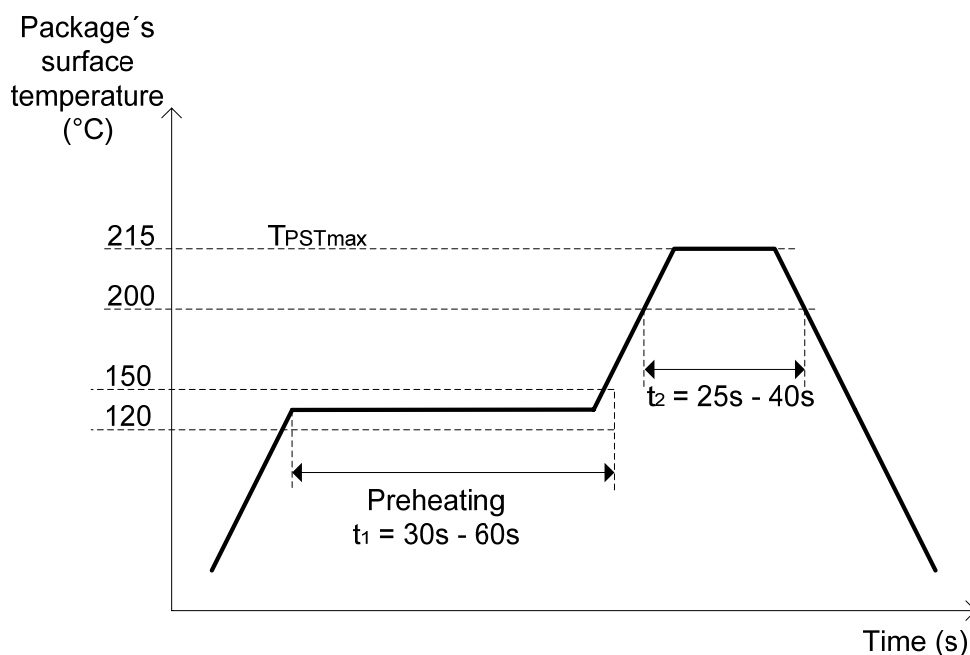
6.2.1 Infrared Reflow Soldering Characterization

Symbol	Parameter	Value
T_{PSTmax}	Maximum temperature, package's surface temperature	250 °C
t_1	Maximum time at maximum temperature	≤ 10 s
t_2	Maximum time of temperature higher than 220 °C	≤ 60 s
t_3	Preheating time at 160 to 180 °C	60 ... 120 s
	Maximum chlorine content of rosin flux	< 0.2 %



6.2.2 Vapour Phase Reflow Soldering (VPS) Characterization

Symbol	Parameter	Value
T_{PSTmax}	Maximum temperature, package's surface temperature	215 °C
t_1	Preheating time	30s - 60s
t_2	Time of temperature > 200°C	25s - 40 s
	Maximum chlorine content of rosin flux	0.2 %



6.3 General storage conditions

Parameter	Conditions	Min.	Max.	Unit
Storage temperature Note 1)	Sealed Drypack	5	30	°C
Storage humidity Note 1)	Sealed Drypack	20	70	%RH
Storage time Note 2)	Sealed Drypack		2	years
Storage temperature Note 3)	Open Drypack		< 25	°C
Storage humidity Note 3)	Open Drypack		< 65	%RH
Storage time after opening dry pack Note 3)	Open Drypack		7	days
Baking time	125°C	20	72	hours
Number of times of mounting			3	times

Notes:

- 1) Storing the sealed Drypacks at other conditions, may affect solderability of the components.
- 2) When storing sealed Drypacks for more than two years, it is recommended to check for oxidation of the solder balls and perform tests to approve solderability prior to using the components for production. The two year period starts at the seal date, printed on the Drypack label.
- 3) Open Drypacks / unpacked components may be stored at these conditions for 7 days before soldering. When exceeding one of the parameters temperature, humidity or time after opening a sealed Drypack, components must be tempered according to parameter "Baking time" before soldering

6.4 Signal Definitions

Signal	PAD Type	Description	
General			
PORn	IUS	Power on reset	
RSTINn	IO18C ('IC')	Reset input (pad used as input only, output can't be enabled)	
RSTOUTn	OZ6	Reset output	
RDY	IOU9	RDY-LED / boot start option	
RUN	IOU9	RUN-LED / boot start option	
WDGACT	IO18C ('OZC9')	Watch dog active (input used in PCI host mode only)	
CLKOUT	IO18C ('OZC9')	Clock out (pad used as output only, input is not connected)	
XTAL			
OSC_XTI	XTAL	25 MHz crystal input	
OSC_XTO	XTAL	25 MHz crystal output	
OSC_VSS	GND	Oscillator ground supply	
OSC_VDDC	PWR	Oscillator power supply 1.5 V	
JTAG			
JT_TRSTn	IDS	JTAG test reset	
JT_TMS	IUS	JTAG test mode select	
JT_TCLK	IUS	JTAG test clock	
JT_TDI	IUS	JTAG test data input	
JT_TDO	OZ6	JTAG test data output	
SPI			
SPI_CLK	IOU6	SPI clock	
SPI_CS0n	IOU6	SPI chip select 0	
SPI_CS1n	IOU6	SPI chip select 1	
SPI_CS2n	IOU6	SPI chip select 2	
SPI_MISO	IOU6	SPI master input slave output data	
SPI_MOSI	IOU6	SPI master output slave input data	
I2C			
I2C_SCL	IOZUS9 (5k pu)	I2C serial clock line	
I2C_SDA	IOZUS9 (5k pu)	I2C serial data line	
USB			
USB_DNEG	USB	USB D- line	
USB_DPOS	USB	USB D+ line	
USB_VSS	GND	USB ground supply	
USB_VDDC	PWR	USB power supply 1.5 V	
USB_VDDIO	PWR	USB power supply 3.3 V	
Test function			
TEST	ID	Activate test mode	Leave open for normal operating mode
TMC1	ID	Test mode 1	
TMC2	ID	Test mode 2	
TACT_TRST	IDS	Reset test controller	Connect to GND for normal operation mode
TCLK	IO18C ('IC')	Test clock (pad used as input only, output can't be enabled)	

Signal	PAD Type	Description
--------	----------	-------------

GPIOs shared with UARTs			
GPIO0	shared	IOD6	General Purpose IO 0
UART0_RXD		IOD6 ('ID')	UART 0 Receive Data
GPIO1	shared	IOD6	General Purpose IO 1
UART0_TXD		IOD6 ('OZD6')	UART 0 Transmit Data
GPIO2	shared	IOD6	General Purpose IO 2
UART0_CTSn		IOD6 ('ID')	UART 0 Clear to Send
GPIO3	shared	IOD6	General Purpose IO 3
UART0_RTSn		IOD6 ('OZD6')	UART 0 Ready to Send
GPIO4	shared	IOD6	General Purpose IO 4
UART1_RXD		IOD6 ('ID')	UART 1 Receive Data
GPIO5	shared	IOD6	General Purpose IO 5
UART1_TXD		IOD6 ('OZD6')	UART 1 Transmit Data
GPIO6	shared	IOD6	General Purpose IO 6
UART1_CTSn		IOD6 ('ID')	UART 1 Clear to Send
GPIO7	shared	IOD6	General Purpose IO 7
UART1_RTSn		IOD6 ('OZD6')	UART 1 Ready to Send
GPIO8	shared	IOD6	General Purpose IO 8
UART2_RXD		IOD6 ('ID')	UART 2 Receive Data
GPIO9	shared	IOD6	General Purpose IO 9
UART2_TXD		IOD6 ('OZD6')	UART 2 Transmit Data
GPIO10	shared	IOD6	General Purpose IO 10
UART2_CTSn		IOD6 ('ID')	UART 2 Clear to Send
GPIO11	shared	IOD6	General Purpose IO 11
UART2_RTSn		IOD6 ('OZD6')	UART 2 Ready to Send
GPIO12-14		IOD6	General Purpose IOs 12-14
GPIO15		IOD6	General Purpose IOs 15
IRQ		IOD6	Interrupt Request

PIOs shared with LCD Controller shared with ETM		
PIOs		
PIO8-30	IOD6	Peripheral IOs 8-30
LCD Controller		
LCD_LP	IOD6 ('OD6')	STN / TFT: Line / Horizontal Synchronization Pulse
LCD_FP	IOD6 ('OD')	STN / TFT: Frame / Vertical Synchronization Pulse
LCD_AC	IOD6 ('OD6')	STN / TFT: AC Bias Drive / Enable Data Output
LCD_CP	IOD6 ('OD6')	LCD Panel Clock
LCD_POWER	IOD6 ('OD6')	LCD Panel Power Enable
LCD_D0-17	IOD6 ('OD6')	LCD Panel Data 0-17
ETM		
ETM_TPKT0-15	IOD6 ('OD6')	ETM Trace packet 0-15
ETM_PSTAT0-2	IOD6 ('OD6')	ETM Pipe status 0-2
ETM_TSYNC	IOD6 ('OD6')	ETM Trace synchronization
ETM_TCLK	IOD6 ('OD6')	ETM Trace clock
ETM_DRQ	IOD6 ('ID')	ETM Debug request
ETM_DACK	IOD6 ('OD6')	ETM Debug acknowledge

Signal	PAD Type	Description
--------	----------	-------------

Real-time Clock

RTC_XTI	XTAL	RTC crystal input
RTC_XTO	XTAL	RTC crystal output
RTC_POK	IDS	Power OK
RTC_VDDIO	PWR	Real-time clock power supply 3.3 V
RTC_VDDC	PWR	Real-time clock power supply 1.5 V

Memory Interface

MEMSR_CS0-2n	O9	SRAM chip select
MEMSR_OEn	O9	SRAM output enable
MEMSR_WEn	O9	SRAM write enable
MEMDR_CSn	O9	SDRAM chip select
MEMDR_WEn	O9	SDRAM write enable
MEMDR_RASn	O9	SDRAM row address strobe
MEMDR_CASn	O9	SDRAM column address strobe
MEMDR_CKE	O9	SDRAM clock enable
MEMDR_CLK	IO9 ('OZ9')	SDRAM clock (pad used as output only)
MEM_DQM0	O9	Memory data qualifier mask D0-7
MEM_DQM1	O9	Memory data qualifier mask D8-15
MEM_DQM2	O9	Memory data qualifier mask D16-23
MEM_DQM3	O9	Memory data qualifier mask D24-31
MEM_D0-31	IO9	Memory data 0-31
MEM_A0-23	O9	Memory address 0-23

PIOs shared with Dual-Port Memory shared with Extension Bus**PIOs**

PIO32-84	IO18C	Peripheral IOs 32-84
----------	-------	----------------------

Dual-Port Memory

DPM_D0-15	IO18C	Dual port memory data 0-15
DPM_A0-15	IO18C ('IC')	Dual port memory address 0-15
DPM_ALE	IO18C ('IC')	Dual port memory address latch enable
DPM_CSn	IO18C ('IC')	Dual port memory chip select
DPM_BHEn	IO18C ('IC')	Dual port memory byte high enable
DPM_RDn	IO18C ('IC')	Dual port memory read
DPM_WRLn	IO18C ('IC')	Dual port memory write low
DPM_WRHn	IO18C ('IC')	Dual port memory write high
DPM_RDY	IO18C ('OZC18')	Dual port memory ready
DPM_INT	IO18C ('OZC18')	Dual port memory interrupt

Extension Bus

EXT_D0-15	IO18C	Extension bus data 0-15
EXT_A0-24	IO18C ('OC18')	Extension bus address 0-24
EXT_CS0-3n	IO18C ('OC18')	Extension bus chip select 0-3
EXT_ALE	IO18C ('OC18')	Extension bus address latch enable
EXT_BHEn	IO18C ('OC18')	Extension bus byte high enable
EXT_RDn	IO18C ('OC18')	Extension bus read
EXT_WRLn	IO18C ('OC18')	Extension bus write low
EXT_WRHn	IO18C ('OC18')	Extension bus write high
EXT_RDY	IO18C ('IC')	Extension bus ready
EXT_IRQ	IO18C ('IC')	Extension bus interrupt request

Signal	PAD Type	Description
--------	----------	-------------

PIOs shared with Encoders shared with PWM 0**PIOs**

PIO0-7	IOD6	Peripheral IOs 0-7
--------	------	--------------------

Encoders

ENC0_A	IOD6 ('ID')	Encoder 0 signal A
ENC0_B	IOD6 ('ID')	Encoder 0 signal B
ENC0_N	IOD6 ('ID')	Encoder 0 signal Null
ENC1_A	IOD6 ('ID')	Encoder 1 signal A
ENC1_B	IOD6 ('ID')	Encoder 1 signal B
ENC1_N	IOD6 ('ID')	Encoder 1 signal Null
ENC_MP0	IOD6 ('ID')	Encoder measure point 0
ENC_MP1	IOD6 ('ID')	Encoder measure point 1

PWM 0

PWM0E_U	IOD6 ('OD6')	PWM 0 phase U, shared option E
PWM0E_Un	IOD6 ('OD6')	PWM 0 phase U inverted, shared option E
PWM0E_V	IOD6 ('OD6')	PWM 0 phase V, shared option E
PWM0E_Vn	IOD6 ('OD6')	PWM 0 phase V inverted, shared option E
PWM0E_W	IOD6 ('OD6')	PWM 0 phase W, shared option E
PWM0E_Wn	IOD6 ('OD6')	PWM 0 phase W inverted, shared option E
PWM0E_FAILn	IOD6 ('ID')	PWM 0 failure, shared option E
PWM0E_RSV	IOD6 ('OD6')	PWM 0 resolver, shared option E

AD-Converter

ADC0_IN0	ANA	AD-Converter 0 channel 0 input
ADC0_IN1	ANA	AD-Converter 0 channel 1 input
ADC0_IN2	ANA	AD-Converter 0 channel 2 input
ADC0_IN3	ANA	AD-Converter 0 channel 3 input
ADC0_VREFM	AGND	AD-Converter 0 reference voltage low
ADC0_VREFP	APWR	AD-Converter 0 reference voltage high
ADC0_VSS	AGND	AD-Converter 0 ground supply
ADC0_VDDIO	APWR	AD-Converter 0 power supply 3.3 V
ADC1_IN0	ANA	AD-Converter 1 channel 0 input
ADC1_IN1	ANA	AD-Converter 1 channel 1 input
ADC1_IN2	ANA	AD-Converter 1 channel 2 input
ADC1_IN3	ANA	AD-Converter 1 channel 3 input
ADC1_VREFM	AGND	AD-Converter 1 reference voltage low
ADC1_VREFP	APWR	AD-Converter 1 reference voltage high
ADC1_VSS	AGND	AD-Converter 1 ground supply
ADC1_VDDIO	APWR	AD-Converter 1 power supply 3.3 V

Signal	PAD Type	Description
--------	----------	-------------

Fieldbus 0 shared with PHY 0 fiberoptic shared with PWM 1, shared option A

XM0_RX	IOU9 ('IU')	Fieldbus 0 Receive Data
XM0_TX	IOU9 ('OU9')	Fieldbus 0 Transmit Data
XM0_IO0	IOD9	Fieldbus 0 General IO 0
XM0_IO1	IOD9	Fieldbus 0 General IO 1

Fieldbus 1 shared with PHY 1 fiberoptic shared with PWM 1, shared option A

XM1_RX	IOU9 ('IU')	Fieldbus 1 Receive Data
XM1_TX	IOU9 ('OU9')	Fieldbus 1 Transmit Data
XM1_IO0	IOD9	Fieldbus 1 General IO 0
XM1_IO1	IOD9	Fieldbus 1 General IO 1

PHY 1 fiberoptic

FO1_RD	IOU9 ('IU')	PHY 1 Fiberoptic Receive Data
FO1_TD	IOU9 ('OU9')	PHY 1 Fiberoptic Transmit Data
FO1_SD	IOD9 ('ID')	PHY 1 Fiberoptic Signal Detect
FO1_EN	IOD9 ('OD9')	PHY 1 Fiberoptic Enable Indication

PWM 1, shared option A

PWM1A_Vn	IOU9 ('OU9')	PWM 1 Phase V inverted, shared option A
PWM1A_W	IOU9 ('OU9')	PWM 1 Phase W, shared option A
PWM1A_Wn	IOU9 ('OD9')	PWM 1 Phase W inverted, shared option A
PWM1A_RSV	IOU9 ('OD9')	PWM 1 Resolver, shared option A

Fieldbus 2 shared with PWM 1, shared Option B

XM2_RX	IOU9 ('IU')	Fieldbus 2 Receive Data
XM2_TX	IOU9 ('OU9')	Fieldbus 2 Transmit Data
XM2_IO0	IOD9	Fieldbus 2 General IO 0
XM2_IO1	IOD9	Fieldbus 2 General IO 1
XM2_ECLK	shared	IOD9
PWM1B_FAILn		IOD9 ('ID')
XM0_ECLK	shared	IOD9
PWM1B_RSV		IOD9 ('OD9')

Fieldbus 3 shared with PWM 1, shared Option B**Fieldbus 3**

XM3_RX	IOU9 ('IU')	Fieldbus 3 Receive Data
XM3_TX	IOU9 ('OU9')	Fieldbus 3 Transmit Data
XM3_IO0	IOD9	Fieldbus 3 General IO 0
XM3_IO1	IOD9	Fieldbus 3 General IO 1
XM3_ECLK	IOD9	Fieldbus 3 External Receive-/Transmit Clock
XM1_ECLK	IOD9	Fieldbus 1 External Receive-/Transmit Clock

PWM 1, shared Option B

PWM1B_U	IOU9 ('OU9')	PWM 1 Phase U, shared option B
PWM1B_Un	IOU9 ('OU9')	PWM 1 Phase U inverted, shared option B
PWM1B_V	IOD9 ('OD9')	PWM 1 Phase V, shared option B
PWM1B_Vn	IOD9 ('OD9')	PWM 1 Phase V inverted, shared option B
PWM1B_W	IOD9 ('OD9')	PWM 1 Phase W, shared option B
PWM1B_Wn	IOD9 ('OD9')	PWM 1 Phase W inverted, shared option B

Power Supply

VSS	34x	GND	Ground supply (except PHYs, ADCs, USB, OSC)
VDDC	18x	PWR	Power supply core 1.5 V (except PHYs, ADCs, USB, RTC, OSC)
VDDIO	21x	PWR	Power supply IO buffer 3.3 V (except PHYs, ADCs, USB, RTC)
VDDH	3x		Common cathode of upper clamping diodes of host interface pins. Connect to 3.3 V or 5.0 V according to input level.

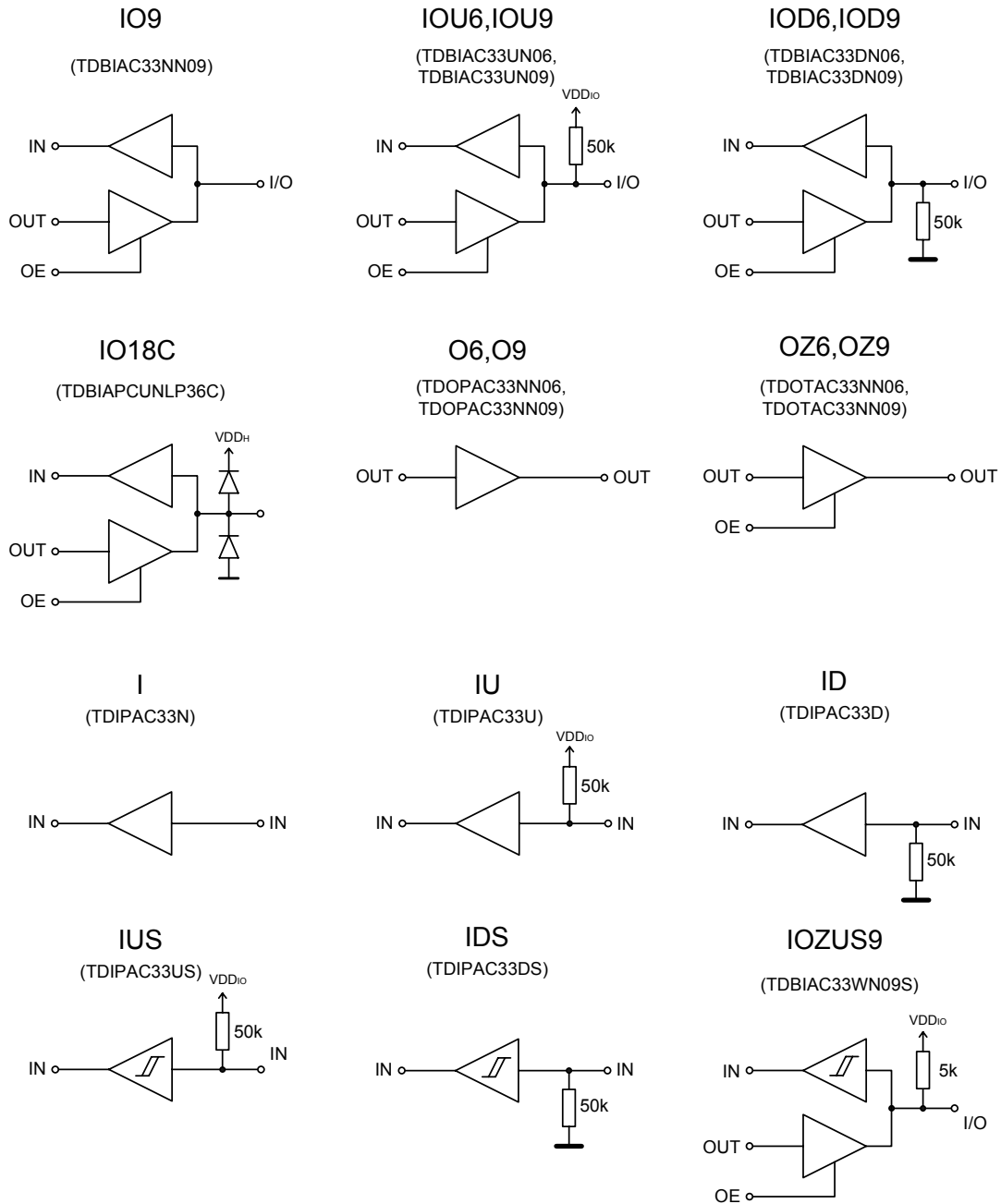
PAD Type Explanation:

Symbol	Description
I	Input
O	Output
Z	Output is tri-state-able or open drain
S	Input provides Schmitt trigger
U	Internal pull-up 50 k (I2C pins: pull-up 5k)
D	Internal pull-down 50 k
C	Internal clamping diodes to GND and VDDh
6	Output driver can source / sink 6 mA
9	Output driver can source / sink 9 mA
18	Output driver can source / sink 18 mA
XTAL	Crystal input or output
USB	USB pad
PHY	PHY pad
ANA	Analog pin
PWR	1.5V (Core) or 3.3V I/O
GND	Digital ground (0V)
APWR	Analog power (1.5V or 3.3V)
AGND	Analog ground (0V)

Notes:

- 1) PAD Types in brackets resemble the logical functionality of a buffer in a certain situation (when a pure input signal is assigned to an IO buffer, the buffer does of course not change in any way, however as its output driver is then automatically disabled, the buffer behaves like an input buffer only).

6.4.1 Schematic View of netX Pad Types



Notes:

The O6, OZ9, I and IU buffer types are not used in the netX 500/100.

6.5 Pin Table Sorted By Pin Numbers

Pin	Signal
A1	VDDIO
A2	VSS
A3	MEM_D22
A4	MEM_D23
A5	DPM_D02
A5	EXT_D02
A5	PIO81
A6	DPM_D05
A6	EXT_D05
A6	PIO76
A7	DPM_A00
A7	EXT_A00
A7	PIO73
A8	VDDH
A9	DPM_A05
A9	EXT_A05
A9	PIO64
A10	VDDH
A11	TCLK
A12	VDDC
A13	DPM_A13
A13	EXT_A13
A13	PIO48
A14	DPM_BHEn
A14	EXT_BHEn
A14	PIO43
A15	VDDH
A16	DPM_D13
A16	EXT_D13
A16	PIO37
A17	DPM_D10
A17	EXT_D10
A17	PIO33
A18	DPM_D08
A18	EXT_D08
A18	PIO32
A19	USB_VSS
A20	VSS
A21	VDDIO
B1	VDDIO
B2	MEM_D20
B3	MEM_D21
B4	MEM_D24
B5	DPM_D01
B5	EXT_D01
B5	PIO82
B6	DPM_D04
B6	EXT_D04
B6	PIO77
B7	DPM_D07
B7	EXT_D07
B7	PIO74
B8	DPM_A02
B8	EXT_A02
B8	PIO69
B9	DPM_A04
B9	EXT_A04
B9	PIO65
B10	DPM_A07
B10	EXT_A07
B10	PIO60
B11	DPM_A09
B11	EXT_A09
B11	PIO56
B12	VDDC
B13	DPM_A12
B13	EXT_A12
B13	PIO49
B14	DPM_WRHn
B14	EXT_WRHn
B14	PIO44
B15	DPM_D15
B15	EXT_D15
B15	PIO41
B16	DPM_D12
B16	EXT_D12
B16	PIO38
B17	DPM_D09
B17	EXT_D09
B17	PIO34

Pin	Signal
B18	CLKOUT
B19	USB_DNEG
B20	USB_DPOS
B21	VDDIO
C1	MEM_D18
C2	MEM_D19
C3	MEM_D25
C4	MEM_D26
C5	DPM_D00
C5	EXT_D00
C5	PIO83
C6	DPM_D03
C6	EXT_D03
C6	PIO78
C7	DPM_D06
C7	EXT_D06
C7	PIO75
C8	DPM_A01
C8	EXT_A01
C8	PIO70
C9	DPM_A03
C9	EXT_A03
C9	PIO66
C10	DPM_A06
C10	EXT_A06
C10	PIO61
C11	DPM_A08
C11	EXT_A08
C11	PIO57
C12	DPM_A10
C12	EXT_A10
C12	PIO53
C13	DPM_A11
C13	EXT_A11
C13	PIO50
C14	DPM_WRLn
C14	EXT_WRLn
C14	PIO45
C15	DPM_D14
C15	EXT_D14
C15	PIO42
C16	DPM_D11
C16	EXT_D11
C16	PIO39
C17	DPM_ALE
C17	EXT_ALE
C17	PIO35
C18	RSTInn
C19	USB_VDDC
C20	VDDC
C21	VDDC
D1	MEM_D16
D2	MEM_D17
D3	MEM_D27
D4	MEM_D28
D5	EXT_CS3n
D5	PIO84
D6	EXT_CS2n
D6	PIO79
D7	VDDIO
D8	EXT_A22
D8	PIO71
D9	EXT_A20
D9	PIO67
D10	EXT_A18
D10	PIO62
D11	EXT_A16
D11	PIO58
D12	DPM_A14
D12	EXT_A14
D12	PIO54
D13	DPM_CSn
D13	EXT_CS0n
D13	PIO51
D14	DPM_RDY
D14	EXT_RDY
D14	PIO46
D15	VDDIO
D16	WDGACT
D17	PIO36

Pin	Signal
D18	USB_VDDIO
D19	PHY1_VSSAR
D20	PHY1_VDDCART
D21	VDDC
E1	MEM_A01
E2	MEM_A00
E3	MEM_D29
E4	MEM_D30
E5	MEMDR_CSn
E6	EXT_CS1n
E6	PIO80
E7	VDDIO
E8	EXT_A23
E8	PIO72
E9	EXT_A21
E9	PIO68
E10	EXT_A19
E10	PIO63
E11	EXT_A17
E11	PIO59
E12	DPM_A15
E12	EXT_A15
E12	PIO55
E13	DPM_RDn
E13	EXT_RDn
E13	PIO52
E14	DPM_INT
E14	EXT_IRQ
E14	PIO47
E15	VDDIO
E16	PIO40
E16	EXT_A24
E17	VSS
E18	PHY_VSSACP
E19	PHY1_VSSAT1
E20	PHY1_RXP
E21	PHY1_RXN
F1	MEM_A03
F2	MEM_A02
F3	MEM_D31
F4	MEMDR_CKE
F5	MEMDR_RASn
F17	PHY_VDDCAP
F18	PHY_VDDIOAC
F19	PHY1_VSSAT2
F20	PHY1_TXP
F21	PHY1_TXN
G1	MEM_A05
G2	MEM_A04
G3	MEM_D08
G4	MEM_D00
G5	MEMDR_CASn
G17	PHY_ATP
G18	PHY0_VSSAT2
G19	PHY_EXTRES
G20	PHY0_TXN
G21	PHY0_TXP
H1	MEM_A07
H2	MEM_A06
H3	MEM_D09
H4	MEM_D01
H5	MEMDR_WEn
H17	PHY0_VDDCART
H18	PHY0_VSSAR
H19	PHY0_VSSAT1
H20	PHY0_RXN
H21	PHY0_RXP
J1	MEM_A09
J2	MEM_A08
J3	MEM_D10
J4	MEM_D02
J5	MEMDR_CLK
J9	VSS
J10	VSS
J11	VSS
J12	VSS
J13	VSS
J17	VDDC
J18	VDDC
J19	VSS

Pin	Signal
J20	PHY_VDDIOAT
J21	PHY_VSSAT
K1	MEMSR_CS0n
K2	MEM_A10
K3	MEM_D11
K4	MEM_D03
K5	MEM_DQM3
K9	VSS
K10	VSS
K11	VSS
K12	VSS
K13	VSS
K17	VDDC
K18	VDDC
K19	TEST
K20	TACT_TRST
K21	VSS
L1	VDDC
L2	VDDC
L3	MEM_A11
L4	VDDIO
L5	VDDIO
L9	VSS
L10	VSS
L11	VSS
L12	VSS
L13	VSS
L17	VDDIO
L18	VDDIO
L19	TMC2
L20	VDDC
L21	VDDC
M1	MEM_A12
M2	MEM_A13
M3	MEM_D12
M4	MEM_D04
M5	MEM_DQM2
M9	VSS
M10	VSS
M11	VSS
M12	VSS
M13	VSS
M17	ENC1_B
M17	PIO04
M17	PWM0E_W
M18	ENC1_A
M18	PIO03
M18	PWM0E_Vn
M19	ENC0_N
M19	PIO02
M19	PWM0E_V
M20	ENC0_B
M20	PIO01
M20	PWM0E_Un
M21	ENC0_A
M21	PIO00
M21	PWM0E_U
N1	MEM_A14
N2	MEM_A15
N3	MEM_D13
N4	MEM_D05
N5	MEM_DQM1
N9	VSS
N10	VSS
N11	VSS
N12	VSS
N13	VSS
N17	ENC1_N
N17	PIO05
N17	PWM0E_Wn
N18	FO0_EN
N18	PWM1A_FAILn
N18	XM0_IO1
N19	FO0_SD
N19	PWM1A_V
N19	XM0_IO0
N20	FO0_TD
N20	PWM1A_Un
N20	XM0_TX
N21	FO0_RD

Pin	Signal	Pin	Signal	Pin	Signal
N21	PWM1A_U	U21	XM3_RX	Y10	LCD_D09
N21	XM0_RX	V1	OSC_XTI	Y10	PIO17
P1	MEM_A16	V2	OSC_VDDC	Y11	ETM_PSTAT1
P2	MEM_A17	V3	ADC1_VREFM	Y11	LCD_D04
P3	MEM_D14	V4	ADC1_IN2	Y11	PIO12
P4	MEM_D06	V5	ADC0_IN3	Y12	VDDC
P5	MEM_DQM0	V6	ADC0_VREFP	Y13	GPIO13
P17	ENC_MP0	V7	VDDIO	Y14	GPIO11
P17	PIO06	V8	ETM_TPKT07	Y14	UART2_RTS
P17	PWM0E_FAILn	V8	LCDCP	Y15	GPIO09
P18	FO1_EN	V8	PIO29	Y15	UART2_TXD
P18	PWM1A_RSV	V9	ETM_DREQ	Y16	GPIO07
P18	XM1_IO1	V9	LCD_D16	Y16	UART1_RTS
P19	FO1_SD	V9	PIO24	Y17	GPIO05
P19	PWM1A_Wn	V10	ETM_TPKT13	Y17	UART1_TXD
P19	XM1_IO0	V10	LCD_D11	Y18	GPIO03
P20	FO01_TD	V10	PIO19	Y18	UART0_RTS
P20	PWM1A_W	V11	ETM_TPKT10	Y19	GPIO01
P20	XM1_TX	V11	LCD_D06	Y19	UART0_TXD
P21	FO1_RD	V11	PIO14	Y20	VSS
P21	PWM1A_Vn	V12	ETM_TPKT08	Y21	VDDIO
P21	XM1_RX	V12	LCD_D01	AA1	VDDIO
R1	MEM_A18	V12	PIO09	AA2	VSS
R2	MEM_A19	V13	GPIO15	AA3	ADC1_VREFP
R3	MEM_D15	V13	IRQ	AA4	ADC0_VREFM
R4	MEM_D07	V14	JT_TDI	AA5	ADC0_IN0
R5	MEMSR_CS1n	V15	VDDIO	AA6	RTC_VDDIO
R17	ENC_MP1	V16	SPI_CS2n	AA7	RTC_XTI
R17	PIO07	V17	SPI_CS1n	AA8	ETM_TPKT04
R17	PWM0E_RSV	V18	SPI_CS0n	AA8	LCD_LP
R18	XM2_IO1	V19	RSTOUTn	AA8	PIO26
R19	XM2_IO0	V20	PWM1B_W	AA9	PIO21
R20	XM2_TX	V20	XM3_ECLK	AA9	ETM_TPKT02
R21	XM2_RX	V21	PWM1B_Wn	AA9	LCD_D13
T1	MEM_A20	V21	XM1_ECLK	AA10	ETM_TPKT00
T2	MEM_A21	W1	OSC_XTO	AA10	LCD_D08
T3	MEMSR_WEn	W2	OSC_VSS	AA10	PIO16
T4	MEMSR_OEn	W3	ADC1_IN1	AA11	ETM_PSTAT2
T5	MEMSR_CS2n	W4	ADC1_VDDIO	AA11	LCD_D03
T17	VDDC	W5	ADC0_IN2	AA11	PIO11
T18	VDDC	W6	ADC0_VDDIO	AA12	VDDC
T19	TMC1	W7	RTC_POK	AA13	GPIO12
T20	PWM1B_FAILn	W8	ETM_TCLK	AA14	UART2_CTS
T20	XM2_ECLK	W8	LCD_AC	AA14	GPIO10
T21	PWM1B_RSV	W8	PIO28	AA15	GPIO08
T21	XM0_ECLK	W9	ETM_TPKT05	AA15	UART2_RXD
U1	MEM_A22	W9	LCD_D15	AA16	UART1_CTS
U2	MEM_A23	W9	PIO23	AA16	GPIO06
U3	VDDC	W10	ETM_TSYNC	AA17	GPIO04
U4	ADC1_VSS	W10	LCD_D10	AA17	UART1_RXD
U5	ADC1_IN3	W10	PIO18	AA18	UART0_CTS
U6	VDDIO	W11	ETM_TPKT11	AA18	GPIO02
U7	VDDIO	W11	LCD_D05	AA19	GPIO00
U8	ETM_TPKT06	W11	PIO13	AA19	UART0_RXD
U8	LCD_POWER	W12	ETM_PSTAT0	AA20	VSS
U8	PIO30	W12	LCD_D00	AA21	VDDIO
U9	ETM_TPKT15	W12	PIO08		
U9	LCD_D17	W13	GPIO14		
U9	PIO25	W14	I2C_SDA		
U10	ETM_TPKT14	W15	I2C_SCL		
U10	LCD_D12	W16	SPI_MOSI		
U10	PIO20	W17	SPI_MISO		
U11	ETM_TPKT12	W18	SPI_CLK		
U11	LCD_D07	W19	PORn		
U11	PIO15	W20	RDY		
U12	ETM_TPKT09	W21	RUN		
U12	LCD_D02	Y1	VDDIO		
U12	PIO10	Y2	VSS		
U13	JT_TCLK	Y3	ADC1_IN0		
U14	JT_TDO	Y4	ADC0_VSS		
U15	VDDIO	Y5	ADC0_IN1		
U16	JT_TMS	Y6	RTC_VDDC		
U17	JT_TRSTn	Y7	RTC_XTO		
U18	PWM1B_Vn	Y8	ETM_DACK		
U18	XM3_IO1	Y8	LCD_FP		
U19	PWM1B_V	Y8	PIO27		
U19	XM3_IO0	Y9	ETM_TPKT03		
U20	PWM1B_Un	Y9	LCD_D14		
U20	XM3_TX	Y9	PIO22		
U21	PWM1B_U	Y10	ETM_TPKT01		

Note: Because of shared signals some Pins appears up to three times at the Pin Table

6.6 Pin Table Sorted By Signals

Signal	Shared with		Pin
ADC0_IN0			AA5
ADC0_IN1			Y5
ADC0_IN2			W5
ADC0_IN3			V5
ADC0_VDDIO			W6
ADC0_VREFM			AA4
ADC0_VREFP			V6
ADC0_VSS			Y4
ADC1_IN0			Y3
ADC1_IN1			W3
ADC1_IN2			V4
ADC1_IN3			U5
ADC1_VDDIO			W4
ADC1_VREFM			V3
ADC1_VREFP			AA3
ADC1_VSS			U4
CLKOUT			B18
DPM_A00	PIO73	EXT_A00	A7
DPM_A01	PIO70	EXT_A01	C8
DPM_A02	PIO69	EXT_A02	B8
DPM_A03	PIO66	EXT_A03	C9
DPM_A04	PIO65	EXT_A04	B9
DPM_A05	PIO64	EXT_A05	A9
DPM_A06	PIO61	EXT_A06	C10
DPM_A07	PIO60	EXT_A07	B10
DPM_A08	PIO57	EXT_A08	C11
DPM_A09	PIO56	EXT_A09	B11
DPM_A10	PIO53	EXT_A10	C12
DPM_A11	PIO50	EXT_A11	C13
DPM_A12	PIO49	EXT_A12	B13
DPM_A13	PIO48	EXT_A13	A13
DPM_A14	PIO54	EXT_A14	D12
DPM_A15	PIO55	EXT_A15	E12
DPM_ALE	PIO35	EXT_AALE	C17
DPM_BHEn	PIO43	EXT_BHEn	A14
DPM_CSn	PIO51	EXT_CSn	D13
DPM_D00	PIO83	EXT_D00	C5
DPM_D01	PIO82	EXT_D01	B5
DPM_D02	PIO81	EXT_D02	A5
DPM_D03	PIO78	EXT_D03	C6
DPM_D04	PIO77	EXT_D04	B6
DPM_D05	PIO76	EXT_D05	A6
DPM_D06	PIO75	EXT_D06	C7
DPM_D07	PIO74	EXT_D07	B7
DPM_D08	PIO32	EXT_D08	A18
DPM_D09	PIO34	EXT_D09	B17
DPM_D10	PIO33	EXT_D10	A17
DPM_D11	PIO39	EXT_D11	C16
DPM_D12	PIO38	EXT_D12	B16
DPM_D13	PIO37	EXT_D13	A16
DPM_D14	PIO42	EXT_D14	C15
DPM_D15	PIO41	EXT_D15	B15
DPM_INT	PIO47	EXT_IRQ	E14
DPM_RDn	PIO52	EXT_RDn	E13
DPM_RDY	PIO46	EXT_RDY	D14
DPM_WRHn	PIO44	EXT_WRHn	B14
DPM_WRLn	PIO45	EXT_WRLn	C14
ENC0_A	PWM0E_U	PIO00	M21
ENC0_B	PWM0E_Un	PIO01	M20
ENC0_N	PWM0E_V	PIO02	M19
ENC1_A	PWM0E_Vn	PIO03	M18
ENC1_B	PWM0E_W	PIO04	M17
ENC1_N	PWM0E_Wn	PIO05	M17
ENC_MP0	PWM0E_FAILn	PIO06	P17
ENC_MP1	PWM0E_RSV	PIO07	R17
ETM_DACK	PIO27	LCD_FP	Y8
ETM_DREQ	PIO24	LCD_D16	V9
ETM_PSTAT0	PIO08	LCD_D00	W12
ETM_PSTAT1	PIO12	LCD_D04	Y11
ETM_PSTAT2	PIO11	LCD_D03	AA11
ETM_TCLK	PIO28	LCD_AC	W8
ETM_TPKT00	PIO16	LCD_D08	AA10
ETM_TPKT01	PIO17	LCD_D09	Y10
ETM_TPKT02	PIO21	LCD_D13	AA9
ETM_TPKT03	PIO22	LCD_D14	Y9
ETM_TPKT04	PIO26	LCD_LP	AA8
ETM_TPKT05	PIO23	LCD_D15	W9
ETM_TPKT06	PIO30	LCD_POWER	U8
ETM_TPKT07	PIO29	LCD_CP	V8
ETM_TPKT08	PIO09	LCD_D01	V12
ETM_TPKT09	PIO10	LCD_D02	U12
ETM_TPKT10	PIO14	LCD_D06	V11
ETM_TPKT11	PIO13	LCD_D05	W11

Signal	Shared with		Pin
ETM_TPKT12	PIO15	LCD_D07	U11
ETM_TPKT13	PIO19	LCD_D11	V10
ETM_TPKT14	PIO20	LCD_D12	U10
ETM_TPKT15	PIO25	LCD_D17	U9
ETM_TSYNC	PIO18	LCD_D10	W10
EXT_A00	PIO73	DPM_A00	A7
EXT_A01	PIO70	DPM_A01	C8
EXT_A02	PIO69	DPM_A02	B8
EXT_A03	PIO66	DPM_A03	C9
EXT_A04	PIO65	DPM_A04	B9
EXT_A05	PIO64	DPM_A05	A9
EXT_A06	PIO61	DPM_A06	C10
EXT_A07	PIO60	DPM_A07	B10
EXT_A08	PIO57	DPM_A08	C11
EXT_A09	PIO56	DPM_A09	B11
EXT_A10	PIO53	DPM_A10	C12
EXT_A11	PIO50	DPM_A11	C13
EXT_A12	PIO49	DPM_A12	B13
EXT_A13	PIO48	DPM_A13	A13
EXT_A14	PIO54	DPM_A14	D12
EXT_A15	PIO55	DPM_A15	E12
EXT_A16	PIO58		D11
EXT_A17	PIO59		E11
EXT_A18	PIO62		D10
EXT_A19	PIO63		E10
EXT_A20	PIO67		D9
EXT_A21	PIO68		E9
EXT_A22	PIO71		D8
EXT_A23	PIO72		E8
EXT_A24	PIO40		E16
EXT_ALE	PIO35	DPM_ALE	C17
EXT_BHEn	PIO43	DPM_BHEn	A14
EXT_CSn	PIO51	DPM_CSn	D13
EXT_CS1n	PIO80		E6
EXT_CS2n	PIO79		D6
EXT_CS3n	PIO84		D5
EXT_D00	PIO83	DPM_D00	C5
EXT_D01	PIO82	DPM_D01	B5
EXT_D02	PIO81	DPM_D02	A5
EXT_D03	PIO78	DPM_D03	C6
EXT_D04	PIO77	DPM_D04	B6
EXT_D05	PIO76	DPM_D05	A6
EXT_D06	PIO75	DPM_D06	C7
EXT_D07	PIO74	DPM_D07	B7
EXT_D08	PIO32	DPM_D08	A18
EXT_D09	PIO34	DPM_D09	B17
EXT_D10	PIO33	DPM_D10	A17
EXT_D11	PIO39	DPM_D11	C16
EXT_D12	PIO38	DPM_D12	B16
EXT_D13	PIO37	DPM_D13	A16
EXT_D14	PIO42	DPM_D14	C15
EXT_D15	PIO41	DPM_D15	B15
EXT_IRQ	PIO47	DPM_INT	E14
EXT_RDn	PIO52	DPM_RDn	E13
EXT_RDY	PIO46	DPM_RDY	D14
EXT_WRHn	PIO44	DPM_WRHn	B14
EXT_WRLn	PIO45	DPM_WRLn	C14
FO0_EN	XM0_IO1	PWM1A_FAILn	N18
FO0_RD	XM0_RX	PWM1A_U	N21
FO0_SD	XM0_IO0	PWM1A_V	N19
FO0_TD	XM0_TX	PWM1A_Un	N20
FO01_TD	XM1_TX	PWM1A_W	P20
FO1_EN	XM1_IO1	PWM1A_RSV	P18
FO1_RD	XM1_RX	PWM1A_Vn	P21
FO1_SD	XM1_IO0	PWM1A_Wn	P19
GPIO00	UART0_RXD		AA19
GPIO01	UART0_TXD		Y19
GPIO02	UART0_CTS		AA18
GPIO03	UART0_RTS		Y18
GPIO04	UART1_RXD		AA17
GPIO05	UART1_TXD		Y17
GPIO06	UART1_CTS		AA16
GPIO07	UART1_RTS		Y16
GPIO08	UART2_RXD		AA15
GPIO09	UART2_TXD		Y15
GPIO10	UART2_CTS		AA14
GPIO11	UART2_RTS		Y14
GPIO12			AA13
GPIO13			Y13
GPIO14			W13

Signal	Shared with		Pin
GPIO15	IRQ		V13
I2C_SCL			W15
I2C_SDA			W14
IRQ	GPIO15		V13
JT_TCLK			U13
JT_TDI			V14
JT_TDO			U14
JT_TMS			U16
JT_TRSTn			U17
LCD_AC	PIO28	ETM_TCLK	W8
LCD_D00	PIO08	ETM_PSTAT0	W12
LCD_D01	PIO09	ETM_TPKT08	V12
LCD_D02	PIO10	ETM_TPKT09	U12
LCD_D03	PIO11	ETM_PSTAT2	AA11
LCD_D04	PIO12	ETM_PSTAT1	Y11
LCD_D05	PIO13	ETM_TPKT11	W11
LCD_D06	PIO14	ETM_TPKT10	V11
LCD_D07	PIO15	ETM_TPKT12	U11
LCD_D08	PIO16	ETM_TPKT00	AA10
LCD_D09	PIO17	ETM_TPKT01	Y10
LCD_D10	PIO18	ETM_TSYNC	W10
LCD_D11	PIO19	ETM_TPKT13	V10
LCD_D12	PIO20	ETM_TPKT14	U10
LCD_D13	PIO21	ETM_TPKT02	AA9
LCD_D14	PIO22	ETM_TPKT03	Y9
LCD_D15	PIO23	ETM_TPKT05	W9
LCD_D16	PIO24	ETM_DREQ	V9
LCD_D17	PIO25	ETM_TPKT15	U9
LCD_FP	PIO27	ETM_DACK	Y8
LCD_LP	PIO26	ETM_TPKT04	AA8
LCD_POWER	PIO30	ETM_TPKT06	U8
LCD_CP	PIO29	ETM_TPKT07	V8
MEM_A00			E2
MEM_A01			E1
MEM_A02			F2
MEM_A03			F1
MEM_A04			G2
MEM_A05			G1
MEM_A06			H2
MEM_A07			H1
MEM_A08			J2
MEM_A09			J1
MEM_A10			K2
MEM_A11			L3
MEM_A12			M1
MEM_A13			M2
MEM_A14			N1
MEM_A15			N2
MEM_A16			P1
MEM_A17			P2
MEM_A18			R1
MEM_A19			R2
MEM_A20			T1
MEM_A21			T2
MEM_A22			U1
MEM_A23			U2
MEM_D00			G4
MEM_D01			H4
MEM_D02			J4
MEM_D03			K4
MEM_D04			M4
MEM_D05			N4
MEM_D06			P4
MEM_D07			R4
MEM_D08			G3
MEM_D09			H3
MEM_D10			J3
MEM_D11			K3
MEM_D12			M3
MEM_D13			N3
MEM_D14			P3
MEM_D15			R3
MEM_D16			D1
MEM_D17			D2
MEM_D18			C1
MEM_D19			C2
MEM_D20			B2
MEM_D21			B3
MEM_D22			A3
MEM_D23			A4

Signal	Shared with		Pin
MEM_D24			B4
MEM_D25			C3
MEM_D26			C4
MEM_D27			D3
MEM_D28			D4
MEM_D29			E3
MEM_D30			E4
MEM_D31			F3
MEM_DQM0			P5
MEM_DQM1			N5
MEM_DQM2			M5
MEM_DQM3			K5
MEMDR_CASn			G5
MEMDR_CKE			F4
MEMDR_CLK			J5
MEMDR_CSn			E5
MEMDR_RASn			F5
MEMDR_WEn			H5
MEMSR_CS0n			K1
MEMSR_CS1n			R5
MEMSR_CS2n			T5
MEMSR_OEn			T4
MEMSR_WEn			T3
ENC_MP0	PWM0E_FAILn	PIO06	P17
ENC_MP1	PWM0E_RSV	PIO07	R17
OSC_VDDC			V2
OSC_VSS			W2
OSC_XTI			V1
OSC_XTO			W1
PHY_ATP			G17
PHY_EXTRES			G19
PHY_VDDCAP			F17
PHY_VDDIOAC			F18
PHY_VDDIOAT			J20
PHY_VSSACP			E18
PHY_VSSAT			J21
PHY0_RXN			H20
PHY0_RXP			H21
PHY0_TXN			G20
PHY0_TXP			G21
PHY0_VDDCART			H17
PHY0_VSSAR			H18
PHY0_VSSAT1			H19
PHY0_VSSAT2			G18
PHY1_RXN			E21
PHY1_RXP			E20
PHY1_TXN			F21
PHY1_TXP			F20
PHY1_VDDCART			D20
PHY1_VSSAR			D19
PHY1_VSSAT1			E19
PHY1_VSSAT2			F19
PIO00	PWM0E_U	ENC0_A	M21
PIO01	PWM0E_Un	ENC0_B	M20
PIO02	PWM0E_V	ENC0_N	M19
PIO03	PWM0E_Vn	ENC1_A	M18
PIO04	PWM0E_W	ENC1_B	M17
PIO05	PWM0E_Wn	ENC1_N	N17
PIO06	PWM0E_FAILn	ENC_MP0	P17
PIO07	PWM0E_RSV	ENC_MP1	R17
PIO08	LCD_D00	ETM_PSTAT0	W12
PIO09	LCD_D01	ETM_TPKT08	V12
PIO10	LCD_D02	ETM_TPKT09	U12
PIO11	LCD_D03	ETM_PSTAT2	AA11
PIO12	LCD_D04	ETM_PSTAT1	Y11
PIO13	LCD_D05	ETM_TPKT11	W11
PIO14	LCD_D06	ETM_TPKT10	V11
PIO15	LCD_D07	ETM_TPKT12	U11
PIO16	LCD_D08	ETM_TPKT00	AA10
PIO17	LCD_D09	ETM_TPKT01	Y10
PIO18	LCD_D10	ETM_TSYNC	W10
PIO19	LCD_D11	ETM_TPKT13	V10
PIO20	LCD_D12	ETM_TPKT14	U10
PIO21	LCD_D13	ETM_TPKT02	AA9
PIO22	LCD_D14	ETM_TPKT03	Y9
PIO23	LCD_D15	ETM_TPKT05	W9
PIO24	LCD_D16	ETM_DREQ	V9
PIO25	LCD_D17	ETM_TPKT15	U9
PIO26	LCD_LP	ETM_TPKT04	AA8
PIO27	LCD_FP	ETM_DACK	Y8

Signal	Shared with		Pin
PIO28	LCD_AC	ETM_TCLK	W8
PIO29	LCD_CP	ETM_TPKT07	V8
PIO30	LCD_POWER	ETM_TPKT06	U8
PIO32	EXT_D08	DPM_D08	A18
PIO33	EXT_D10	DPM_D10	A17
PIO34	EXT_D09	DPM_D09	B17
PIO35	EXT_ALE	DPM_ALE	C17
PIO36			D17
PIO37	EXT_D13	DPM_D13	A16
PIO38	EXT_D12	DPM_D12	B16
PIO39	EXT_D11	DPM_D11	C16
PIO40	EXT_A24		E16
PIO41	EXT_D15	DPM_D15	B15
PIO42	EXT_D14	DPM_D14	C15
PIO43	EXT_BHEn	DPM_BHEn	A14
PIO44	EXT_WRHn	DPM_WRHn	B14
PIO45	EXT_WRLn	DPM_WRLn	C14
PIO46	EXT_RDY	DPM_RDY	D14
PIO47	EXT_IRQ	DPM_INT	E14
PIO48	EXT_A13	DPM_A13	A13
PIO49	EXT_A12	DPM_A12	B13
PIO50	EXT_A11	DPM_A11	C13
PIO51	EXT_CS0n	DPM_CS0n	D13
PIO52	EXT_RDn	DPM_RDn	E13
PIO53	EXT_A10	DPM_A10	C12
PIO54	EXT_A14	DPM_A14	D12
PIO55	EXT_A15	DPM_A15	E12
PIO56	EXT_A09	DPM_A09	B11
PIO57	EXT_A08	DPM_A08	C11
PIO58	EXT_A16		D11
PIO59	EXT_A17		E11
PIO60	EXT_A07	DPM_A07	B10
PIO61	EXT_A06	DPM_A06	C10
PIO62	EXT_A18		D10
PIO63	EXT_A19		E10
PIO64	EXT_A05	DPM_A05	A9
PIO65	EXT_A04	DPM_A04	B9
PIO66	EXT_A03	DPM_A03	C9
PIO67	EXT_A20		D9
PIO68	EXT_A21		E9
PIO69	EXT_A02	DPM_A02	B8
PIO70	EXT_A01	DPM_A01	C8
PIO71	EXT_A22		D8
PIO72	EXT_A23		E8
PIO73	EXT_A00	DPM_A00	A7
PIO74	EXT_D07	DPM_D07	B7
PIO75	EXT_D08	DPM_D08	C7
PIO76	EXT_D05	DPM_D05	A6
PIO77	EXT_D04	DPM_D04	B6
PIO78	EXT_D03	DPM_D03	C6
PIO79	EXT_CS2n		D6
PIO80	EXT_CS1n		E6
PIO81	EXT_D02	DPM_D02	A5
PIO82	EXT_D01	DPM_D01	B5
PIO83	EXT_D00	DPM_D00	C5
PIO84	EXT_CS3n		D5
PORn			W19
PWM0E_FAILn	PIO06	ENC_MP0	P17
PWM0E_U	PIO00	ENC0_A	M21
PWM0E_Un	PIO01	ENC0_B	M20
PWM0E_V	PIO02	ENC0_N	M19
PWM0E_Vn	PIO03	ENC1_A	M18
PWM0E_W	PIO04	ENC1_B	M17
PWM0E_Wn	PIO05	ENC1_N	N17
PWM0E_RSV	PIO07	ENC_MP1	R17
PWM1A_FAILn	XM0_IO1	FO0_EN	N18
PWM1A_U	XM0_RX	FO0_RD	N21
PWM1A_Un	XM0_TX	FO0_TD	N20
PWM1A_V	XM0_IO0	FO0_SD	N19
PWM1A_Vn	XM1_RX	FO1_RD	P21
PWM1A_W	XM1_TX	FO1_TD	P20
PWM1A_Wn	XM1_IO0	FO1_SD	P19
PWM1A_RSV	XM1_IO1	FO1_EN	P18
PWM1B_FAILn	XM2_ECLK		T20
PWM1B_U	XM3_RX		U21
PWM1B_Un	XM3_TX		U20
PWM1B_V	XM3_IO0		U19
PWM1B_Vn	XM3_IO1		U18
PWM1B_W	XM3_ECLK		V20
PWM1B_Wn	XM1_ECLK		V21

Signal	Shared with		Pin
PWM1B_RSV	XM0_ECLK		T21
RDY			W20
RSTInn			C18
RSTOUTn			V19
RTC_POK			W7
RTC_VDDC			Y6
RTC_VDDIO			AA6
RTC_XTI			AA7
RTC_XTO			Y7
RUN			W21
SPI_CLK			W18
SPI_CS0n			V18
SPI_CS1n			V17
SPI_CS2n			V16
SPI_MISO			W17
SPI_MOSI			W16
TACT_TRST			K20
TCLK			A11
TEST			K19
TMC1			T19
TMC2			L19
UART0_CTS	GPIO02		AA18
UART0_RTS	GPIO03		Y18
UART0_RXD	GPIO00		AA19
UART0_TXD	GPIO01		Y19
UART1_CTS	GPIO06		AA16
UART1_RTS	GPIO07		Y16
UART1_RXD	GPIO04		AA17
UART1_TXD	GPIO05		Y17
UART2_CTS	GPIO10		AA14
UART2_RTS	GPIO11		Y14
UART2_RXD	GPIO08		AA15
UART2_TXD	GPIO09		Y15
USB_DNEG			B19
USB_DPOS			B20
USB_VDDC			C19
USB_VDDIO			D18
USB_VSS			A19
WDGACT			D16
XM0_ECLK	PWM1B_RSV		T21
XM0_IO0	PWM1A_V	FO0_SD	N19
XM0_IO1	PWM1A_FAILn	FO0_EN	N18
XM0_RX	PWM1A_U	FO0_RD	N21
XM0_TX	PWM1A_Un	FO0_TD	N20
XM1_ECLK	PWM1B_Wn		V21
XM1_IO0	PWM1A_Wn	FO1_SD	P19
XM1_IO1	PWM1A_RSV	FO1_EN	P18
XM1_RX	PWM1A_Vn	FO1_RD	P21
XM1_TX	PWM1A_W	FO1_TD	P20
XM2_ECLK	PWM1B_FAILn		T20
XM2_IO0			R19
XM2_IO1			R18
XM2_RX			R21
XM2_TX			R20
XM3_ECLK	PWM1B_W		V20
XM3_IO0	PWM1B_V		U19
XM3_IO1	PWM1B_Vn		U18
XM3_RX	PWM1B_U		U21
XM3_TX	PWM1B_Un		U20

Signal	Pin
VSS	A2, A20, E17, J9, J10, J11, J12, J13, J19, K9, K10, K11, K12, K13, K21, L9, L10, L11, L12, L13, M9, M10, M11, M12, M13, N9, N10, N11, N12, N13, Y2, Y20, AA2, AA20
VDDC	A12, B12, C20, C21, D21, J17, J18, K17, K18, L1, L2, L20, L21, T17, T18, U3, Y12, AA12
VDDIO	A1, A21, B1, B21, D7, D15, E7, E15, L4, L5, L17, L18, U6, U7, U15, V7, V15, Y1, Y21, AA1, AA21
VDDH	A8, A10, A15

Note: Because of shared signals some Pins appears up to three times at the Pin Table.

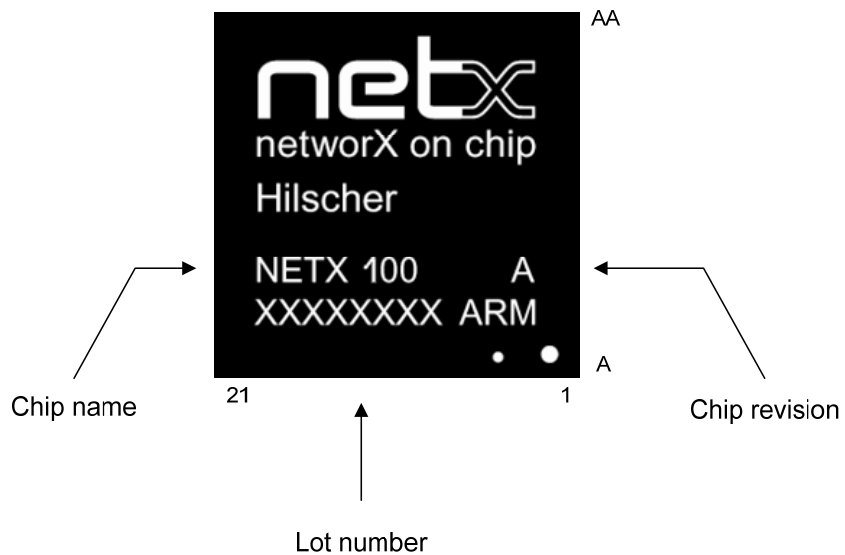
6.7 Pin Overview

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
A	VDDIO	VSS	MEM_D22	MEM_D23	PIO81 EXT_D2 DPM_D2	PIO76 EXT_D5 DPM_D5	PIO73 EXT_A0 DPM_A0	VDDH	PIO64 EXT_A5 DPM_A5	VDDH	TCLK	VDDC	PIO48 EXT_A13 DPM_A13	PIO43 EXT_BHEh DPM_BHEh	VDDH	PIO37 EXT_D13 DPM_D13	PIO33 EXT_D10 DPM_D10	PIO32 EXT_D8 DPM_D8	USB_VSS	VSS	VDDIO
B	VDDIO	MEM_D20	MEM_D21	MEM_D24	PIO82 EXT_D1 DPM_D1	PIO77 EXT_D4 DPM_D4	PIO74 EXT_D7 DPM_D7	PIO69 EXT_A2 DPM_A2	PIO65 EXT_A4 DPM_A4	PIO60 EXT_A7 DPM_A7	PIO56 EXT_A9 DPM_A9	VDDC	PIO49 EXT_A12 DPM_A12	PIO44 EXT_WRHh DPM_WRHh	PIO41 EXT_D15 DPM_D15	PIO38 EXT_D12 DPM_D12	PIO34 EXT_D9 DPM_D9	CLKOUT	USB_DNEG	USB_DPOS	VDDIO
C	MEM_D18	MEM_D19	MEM_D25	MEM_D26	PIO83 EXT_D0 DPM_D0	PIO78 EXT_D3 DPM_D3	PIO75 EXT_D6 DPM_D6	PIO70 EXT_A1 DPM_A1	PIO66 EXT_A3 DPM_A3	PIO61 EXT_A6 DPM_A6	PIO57 EXT_A8 DPM_A8	PIO53 EXT_A10 DPM_A10	PIO50 EXT_A11 DPM_A11	PIO45 EXT_WRLh DPM_WRLh	PIO42 EXT_D14 DPM_D14	PIO39 EXT_D11 DPM_D11	PIO35 EXT_ALE DPM_ALE	RSTIn	USB_VDDC	VDDC	VDDC
D	MEM_D16	MEM_D17	MEM_D27	MEM_D28	PIO84 EXT_CS3h	PIO79 EXT_CS2h	VDDIO	PIO71 EXT_A22	PIO67 EXT_A20	PIO62 EXT_A18	PIO58 EXT_A16	PIO54 EXT_A14 DPM_A14	PIO51 EXT_CS0h DPM_CS0h	PIO46 EXT_RDY DPM_RDY	VDDIO	WDGACT	PIO36	USB_VDDIO	PHY1_VSSAR	PHY1_VDDCART	VDDC
E	MEM_A1	MEM_A0	MEM_D29	MEM_D30	MEMDR_CSn	PIO80 EXT_CS1h	VDDIO	PIO72 EXT_A23	PIO68 EXT_A21	PIO63 EXT_A19	PIO59 EXT_A17	PIO55 EXT_A15 DPM_A15	PIO52 EXT_RD0h DPM_RD0h	PIO47 EXT_RD0 DPM_INT	VDDIO	PIO40 EXT_A24	VSS	PHY_VSSACP	PHY1_VSSAT1	PHY1_RXP	PHY1_RXN
F	MEM_A3	MEM_A2	MEM_D31	MEMDR_CKE	MEMDR_RASn												PHY_VDDCAP	PHY_VDDIOAC	PHY1_VSSAT2	PHY1_TXP	PHY1_TXN
G	MEM_A5	MEM_A4	MEM_D8	MEM_D0	MEMDR_CASn												PHY_ATP	PHY0_VSSAT2	PHY_EXTRES	PHY0_TXN	PHY0_TXP
H	MEM_A7	MEM_A6	MEM_D9	MEM_D1	MEMDR_Wen												PHY0_VDDCART	PHY0_VSSAR	PHY0_VSSAT1	PHY0_RXN	PHY0_RXP
J	MEM_A9	MEM_A8	MEM_D10	MEM_D2	MEMDR_CLK	VSS					VSS	VSS	VSS	VSS							
K	MEMSR_CS0h	MEM_A10	MEM_D11	MEM_D3	MEM_DQM3	VSS					VSS	VSS	VSS	VSS							
L	VDDC	VDDC	MEM_A11	VDDIO	VDDIO	VSS					VSS	VSS	VSS	VSS							
M	MEM_A12	MEM_A13	MEM_D12	MEM_D4	MEM_DQM2	VSS					VSS	VSS	VSS	VSS							
N	MEM_A14	MEM_A15	MEM_D13	MEM_D5	MEM_DQM1	VSS					VSS	VSS	VSS	VSS							
P	MEM_A16	MEM_A17	MEM_D14	MEM_D6	MEM_DQM0												PIO4_ENC1_B PWM0E_W	PIO3_ENC1_A PWM0E_Vn	PIO2_ENC0_N PWM0E_V	PIO1_ENC0_B PWM0E_Wh	PIO0_ENC0_A PWM0E_U
R	MEM_A18	MEM_A19	MEM_D15	MEM_D7	MEMSR_CS1h												PIO5_ENC1_N PWM0E_Wh	XM0_I01 FO0_EN PWM1A_FALn	XM0_I00 FO0_S0 PWM1A_V	XM0_TX FO0_T0 PWM1A_Wh	XM0_RX FO0_R0 PWM1A_U
T	MEM_A20	MEM_A21	MEMSR_Wen	MEMSR_OEn	MEMSR_CS2h												PIO6_ENC_MP0 PWM0E_FALn	XM1_I01 FO1_EN PWM1A	XM1_I00 FO1_S0 PWM1A_Wh	XM1_TX FO1_T0 PWM1A_W	XM1_RX FO1_R0 PWM1A_Vh
U	MEM_A22	MEM_A23	VDDC	ADC1_VSS	ADC1_IN3	VDDIO	VDDIO	PIO30 LCD_POWER ETM_TPKT6	PIO25 LCD_D17 ETM_TPKT15	PIO20 LCD_D12 ETM_TPKT14	PIO15 LCD_D7 ETM_TPKT12	PIO10 LCD_D2 ETM_TPKT9	JT_TCLK	JT_TDO	VDDIO	JT_TMS	JT_TRSTn	XM3_I01 PWM1B_Vh	XM3_I00 PWM1B_V	XM3_TX PWM1B_Wh	XM3_RX PWM1B_U
V	OSC_XT1	OSC_VDDC	ADC1_VREFM	ADC1_IN2	ADC0_IN3	ADC0_VREFP	VDDIO	PIO29 LCD_CP ETM_TPKT7	PIO24 LCD_D16 ETM_DREG	PIO19 LCD_D11 ETM_TPKT13	PIO14 LCD_D6 ETM_TPKT10	PIO9 LCD_D1 ETM_TPKT8	GPIO15_IRQ	JT_TDI	VDDIO	SPL_CS2h	SPL_CS1h	SPL_CS0h	RSTOUTn	XM0_ECLK PWM1B_W	XM1_ECLK PWM1B_Wh
W	OSC_XT0	OSC_VSS	ADC1_IN1	ADC1_VDDIO	ADC0_IN2	ADC0_VDDIO	RTC_POK	PIO28 LCD_LP ETM_TCLK	PIO23 LCD_D15 ETM_TPKT3	PIO18 LCD_D10 ETM_TSYNC	PIO13 LCD_D5 ETM_TPKT11	PIO8 LCD_D0 ETM_PSTAT0	GPIO14	I2C_SDA	I2C_SCL	SPL_MOSI	SPL_MISO	SPL_CLK	PORn	RDY	RUN
Y	VDDIO	VSS	ADC1_IN0	ADC0_VSS	ADC0_IN1	RTC_VDDC	RTC_XT0	PIO27 LCD_FP ETM_DACK	PIO22 LCD_D14 ETM_TPKT3	PIO17 LCD_D9 ETM_TPKT1	PIO12 LCD_D4 ETM_PSTAT1	VDDC	GPIO13	GPIO11_UART2_RTS	GPIO9_UART2_TXD	GPIO7_UART1_RTS	GPIO5_UART1_TXD	GPIO3_UART0_RTS	GPIO1_UART0_TXD	VSS	VDDIO
AA	VDDIO	VSS	ADC1_VREFP	ADC0_VREFM	ADC0_IN0	RTC_VDDIO	RTC_XT1	PIO26 LCD_LP ETM_TPKT4	PIO21 LCD_D13 ETM_TPKT2	PIO16 LCD_D8 ETM_TPKT0	PIO11 LCD_D3 ETM_PSTAT2	VDDC	GPIO12	GPIO10_UART2_CTS	GPIO8_UART2_RXD	GPIO6_UART1_CTS	GPIO4_UART1_RXD	GPIO2_UART0_CTS	GPIO0_UART0_RXD	VSS	VDDIO

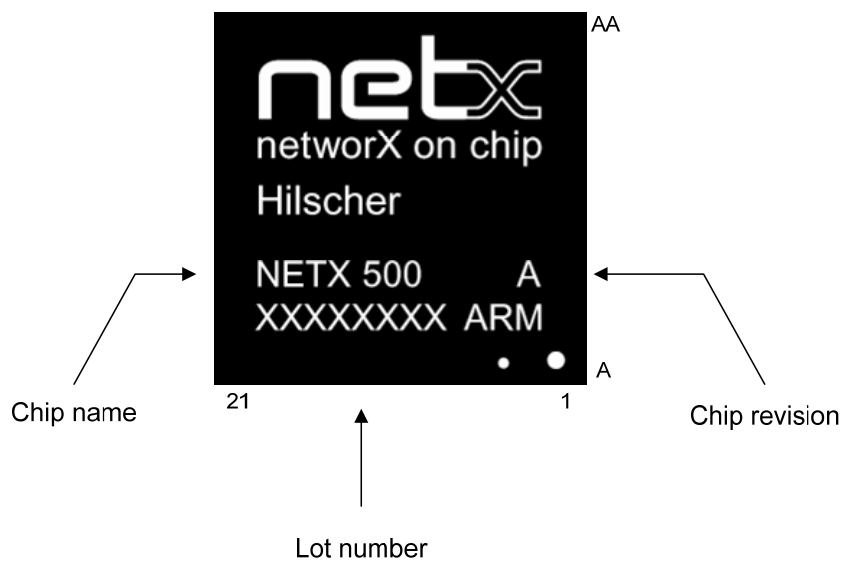
Pining netX 100 and netX 500 Top view

6.8 Device Marking

6.8.1 netX 100

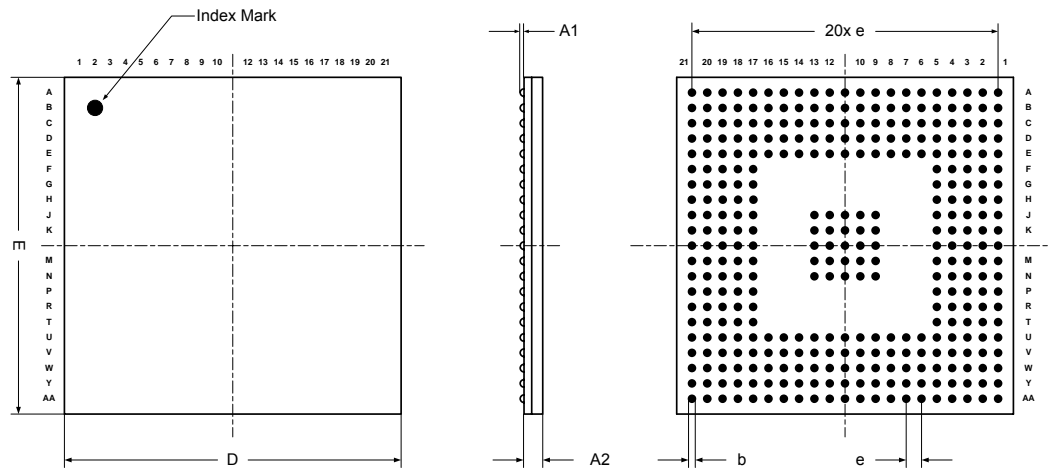


6.8.2 netX 500



6.9 Mechanical Dimensions / Physical Dimensions

The netX 100 and netX 500 come in a 345 pin PBGA package:



Mechanical Dimension of the netX 100 and netX 500

Symbol	Min.	Typ.	Max.
A1	0,42 mm	0,50 mm	0,58 mm
A2	1,23 mm	1,33 mm	1,43 mm
b	0,50 mm	0,60 mm	0,70 mm
E	21,93 mm	22,00 mm	22,07 mm
e	0,95 mm	1,00 mm	1,05 mm
D	21,93 mm	22,00 mm	22,07 mm

6.10 Material composition

6.10.1 Solder balls

Solder ball weight: 327 mg

Material	CAS No.	Amount per ball	Concentration
Copper	7440-50-8	1.7 mg	~ 0.5%
Tin	7440-31-5	315 mg	~ 96.5 %
Silver	7440-22-4	9.8 mg	~ 3 %

6.11 Ordering Information

Ordering Number:

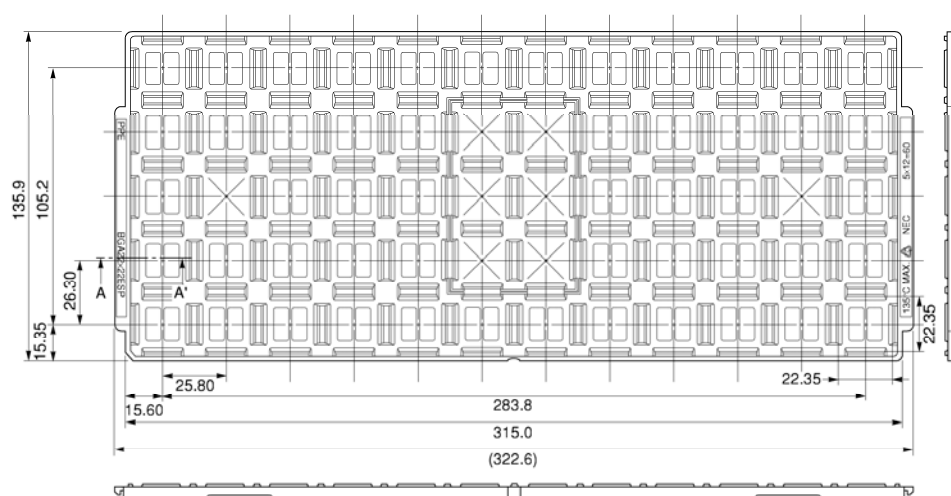
2210.000	NETX 500	netX 500 Network Controller (single chip)	
2210.100	NETX 500 (BOX)	- / -	15 pcs.
2210.200	NETX 500 (TRAY)	- / -	60 pcs.
2210.300	NETX 500 (PACKAGE)	- / -	300 pcs.
2220.000	NETX 100	netX 100 Network Controller (single chip)	
2220.100	NETX 100 (BOX)	- / -	15 pcs.
2220.200	NETX 100 (TRAY)	- / -	60 pcs.
2220.480	NETX 100 (PACKAGE480)	- / -	480 pcs.

6.12 Packing

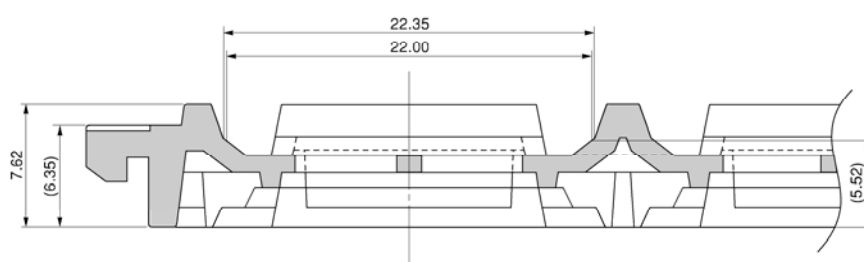
The smallest standard packing unit for the netX 500 and netX 100 is a tray, containing 60 netX devices, protected by a sealed bag (Dry pack). Smaller quantities for sampling or evaluation are repacked at Hilscher.

TRAY CONTAINER

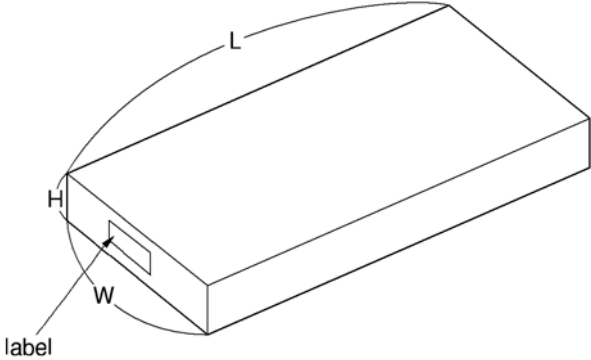
UNIT : mm

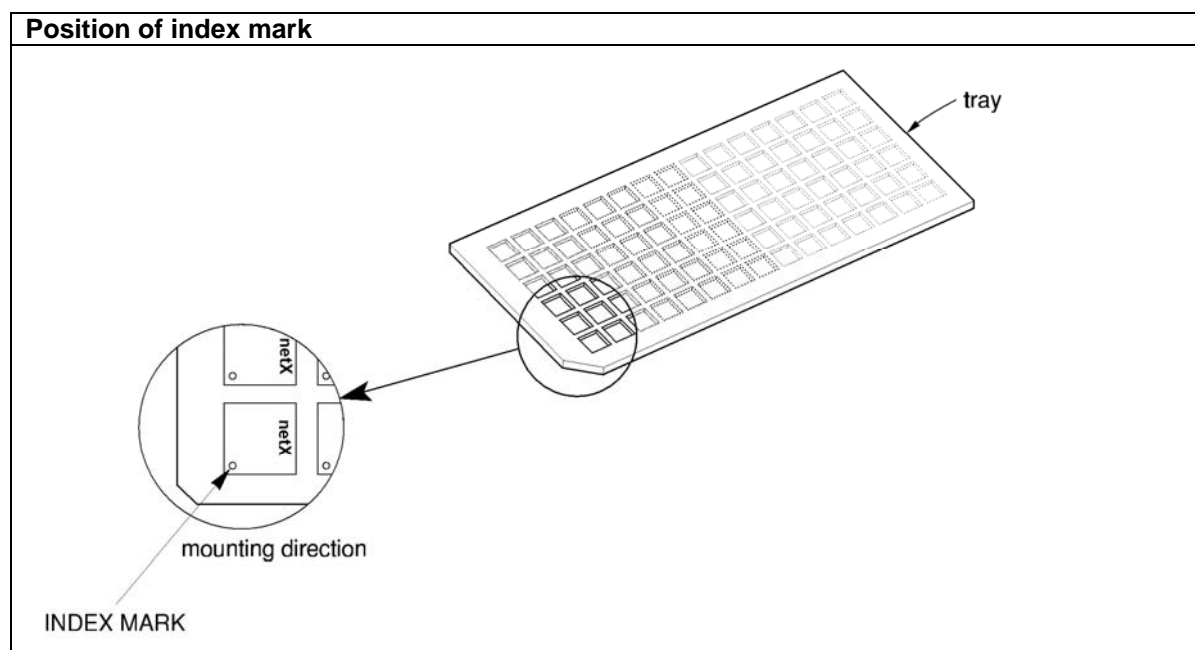


< SECTION A – A' >



Applied Package	Quantity (pcs)	Tray	BGA22×22ESP
320-pin Plastic BGA (22×22)	60 MAX.	Material	Carbon PPE
345-pin Plastic BGA (22×22)		Heat Proof Temp.	135°C MAX.
385-pin Plastic BGA (22×22)		Surface resistance	less than $1 \times 10^{12} \Omega / \square$
389-pin Plastic FBGA (22×22)		The tolerance of tray's dimensions are based on JEDEC STANDARD.	

Box (containing 5 trays)	Dimensions	Label
	153 x 340 x 75 mm (W x L x H)	Position: side of box Part number, quantity, lot number

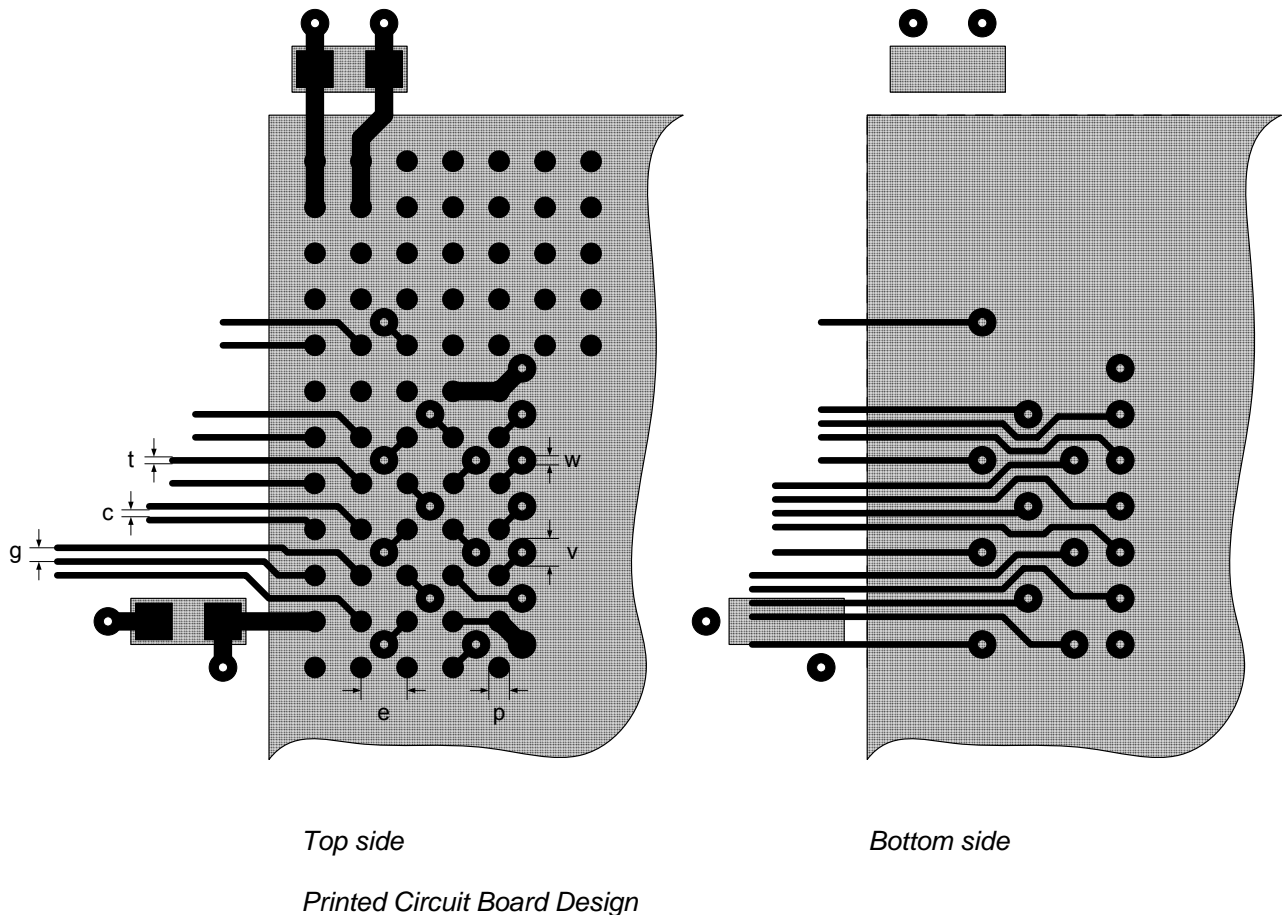


6.13 Moisture Sensitivity Level

The Moisture Sensitivity Level (MSL) is MSL 3, 168 h, 30 °C / 60%RH.

7 Printed Circuit Board Design

Using a PBGA package, the netX requires small traces and vias on the PCB. However, as a result of the proper chip pinout design, four layers and 0.15 mm traces respectively 0.2 mm through whole vias are sufficient for most applications which can hence be realized by inexpensive standard printed circuit board technology.



Dimension	Description	mm	mil
c	Clearance	0,15	6
e	Pitch	1,00	39.37
g	Grid	0.15	6
p	Pad	0,45	18
t	Trace Width	0.15	6
v	Via Diameter	0,60	24
w	Drill Hole	0,20	8
	PCB max. width	2,00	79

Note:

Vias within the chip footprint area should be exactly centered between the pins to avoid possible soldering problems during manufacturing!

7.1 netX Trace Wiring

When routing the netX traces keep in mind, that there are some critical signal pairs. Especially the Ethernet differential pairs must be routed close to each other and must be of equal length.

Care must be taken when routing the SDRAM interface lines. The SDRAM clock line is a 100 MHz signal line and all the SDRAM bus lines should have equal length and capacitive load. The use of 32 Bit SDRAMs instead of 16 Bit RAM pairs is recommended. When also external FLASH and SRAM devices are to be connected, the SDRAM device has to be placed near the netX chip, while the FLASH device can be placed in a greater distance because its timing is usually less critical. With some applications, serial impedance matching resistors for all SDRAM signals may improve signal quality. We strongly recommend the use of CAD systems, that support impedance controlled routing to detect and eliminate signal integrity problems in the first place.

The main quartz oscillator must be placed close to the netX chip, allowing the signal lines to be directly connected to the netX by short traces. The RTC quartz is less critical.

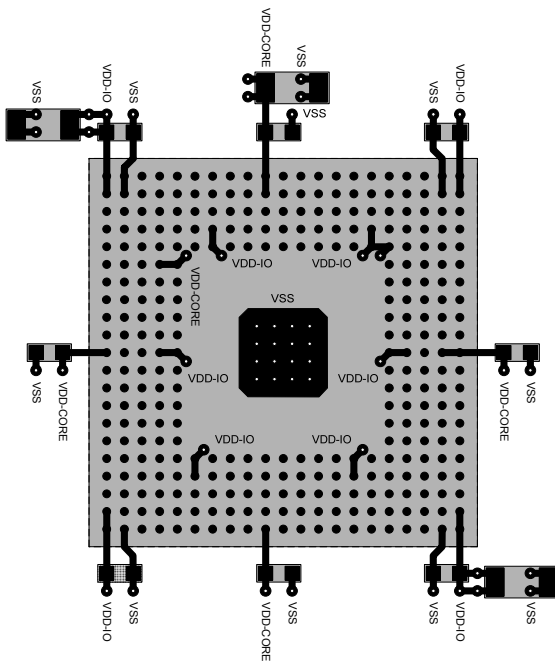
Generally, netX power connections must be as short as possible.

7.2 V_{cc} Pin Requirements / Decoupling Capacitors

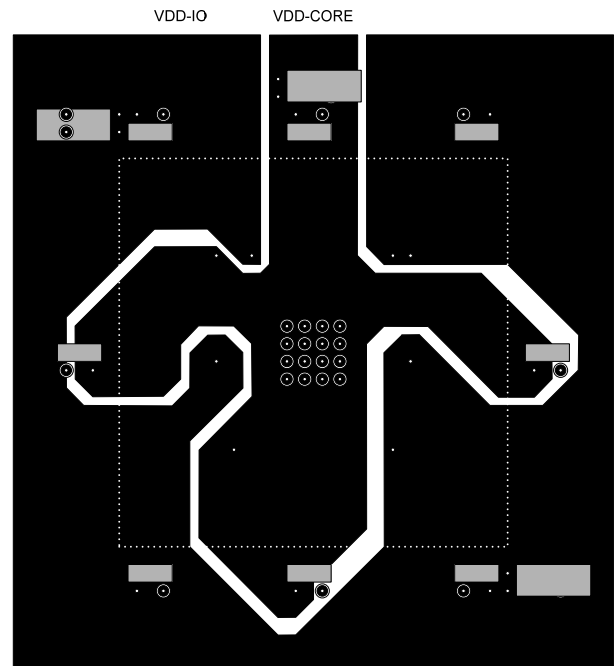
Each corner of the netX accommodates VDD-IO pads and GND pads. VDD-CORE pads are located in the middle of every side. They are all located at the outer PAD ring. This allows placement of the decoupling capacitor close to the power pads on the component side of the printed circuit boards.

The decoupling capacitor should be X7R ceramic type with 100 nF / 6.3 V, which are available in 0603 package. Additional three X5R or X7R ceramic capacitors of 10 uF / 6.3 V should be placed around the netX. They are available in 1206 or 0805 package.

The following picture shows a proposal for the placement of the decoupling capacitor beside the netX. Further, an example for the inner VDD-IO and VDD-CORE power planes is shown. The VSS plane covers one complete plane of the printed circuit board.



Placement of the decoupling capacitors



Inner plane for VDD-IO and VDD-CORE

7.3 Reference PCB Layout Design

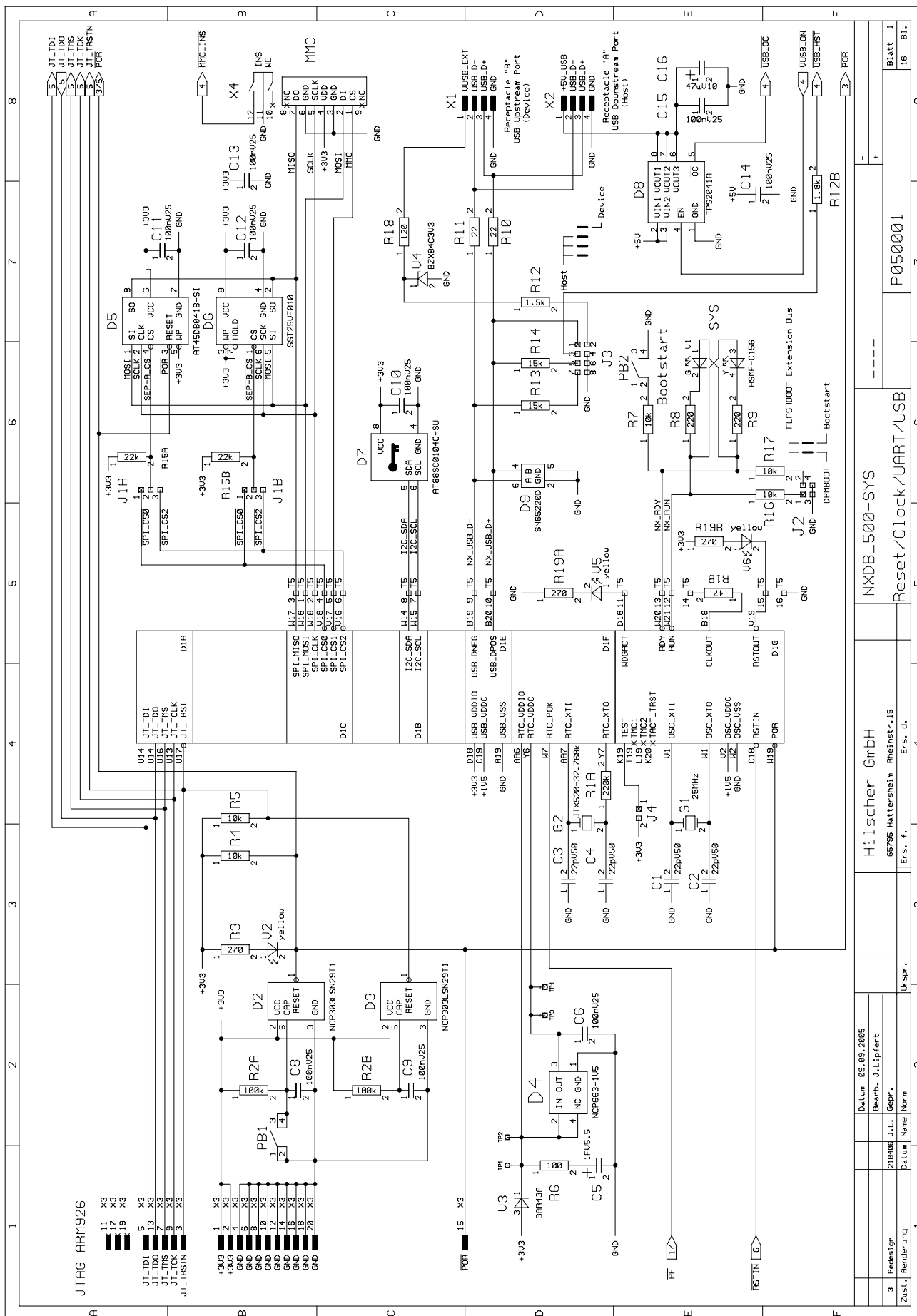
Please refer to the Gerber Data of the NXSB100 evaluation board, available through the Hilscher website (netX Downloads at www.hilscher.com)

8 Reference Schematics

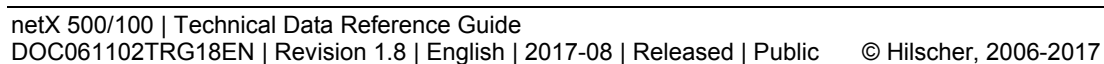
The following pages show the complete reference schematics of the NXDB 500-SYS development board. This board is available from 'Hilscher Gesellschaft für Systemautomation' and has the following ordering number:

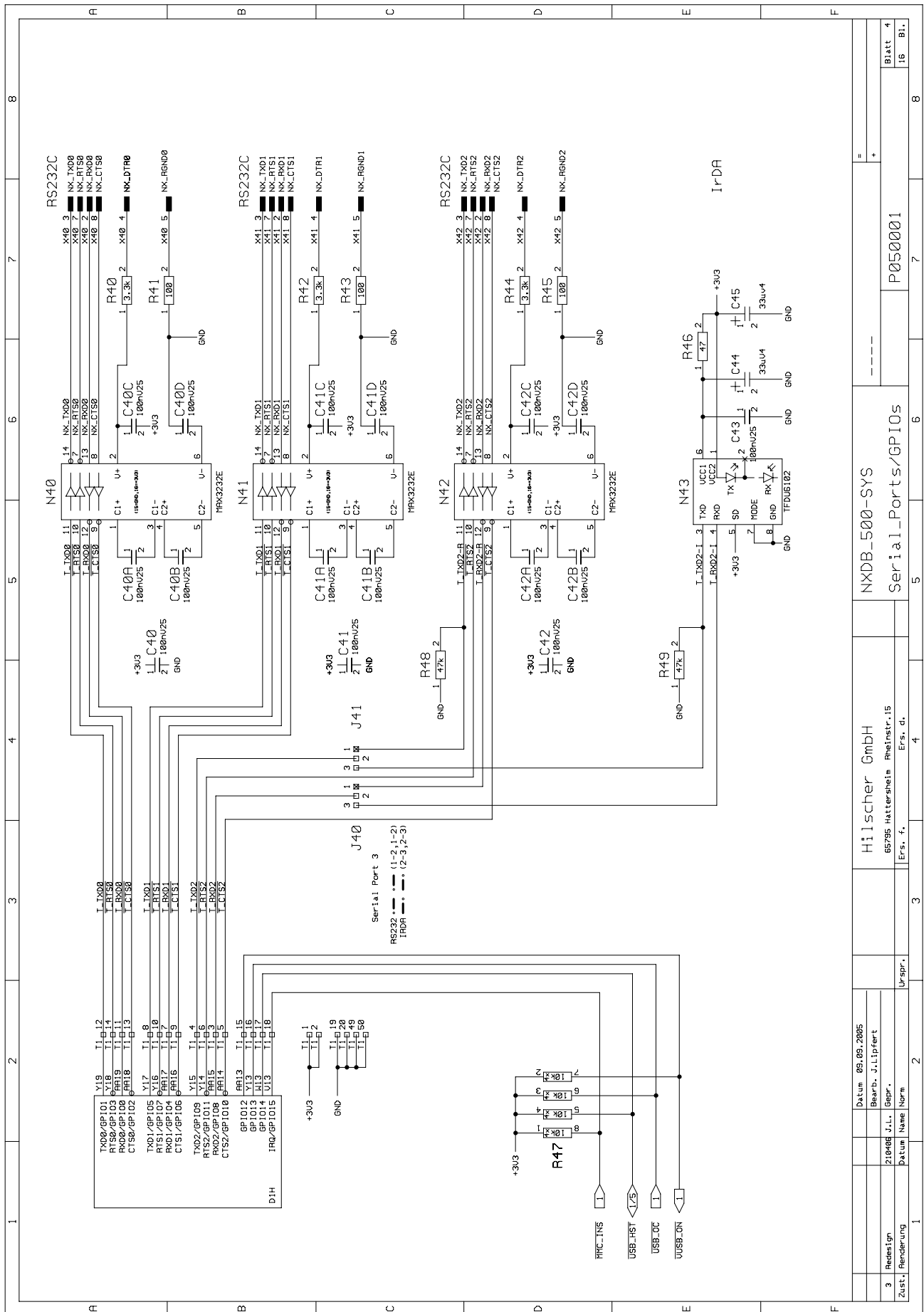
5711.010	NXDB 500-SYS	netX 500 System Development Board
----------	--------------	-----------------------------------

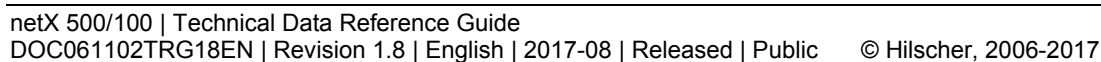
For the most recent revision of the reference schematics please check the Hilscher website (netX Downloads) at www.hilscher.com !

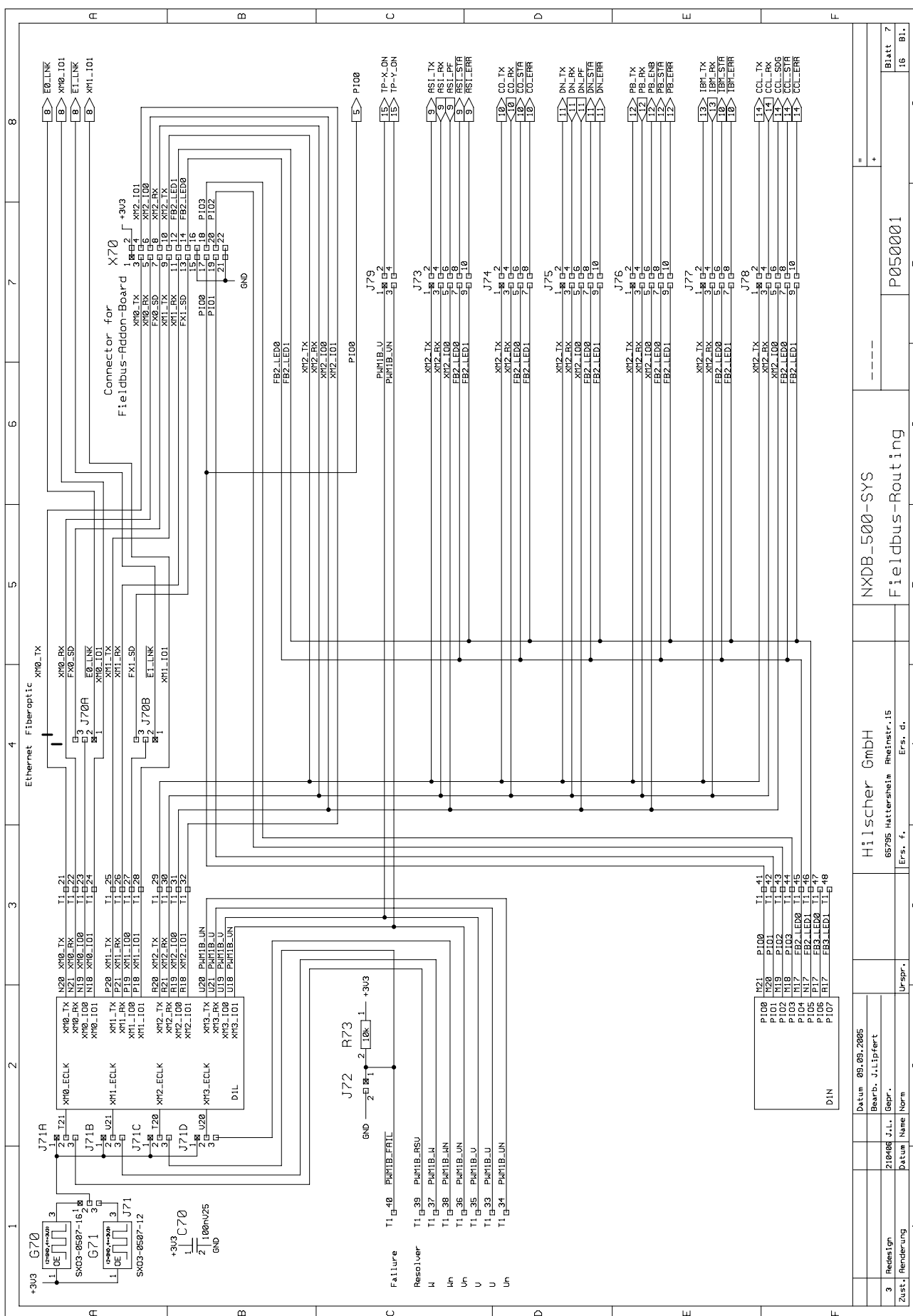


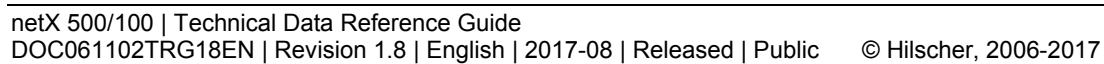


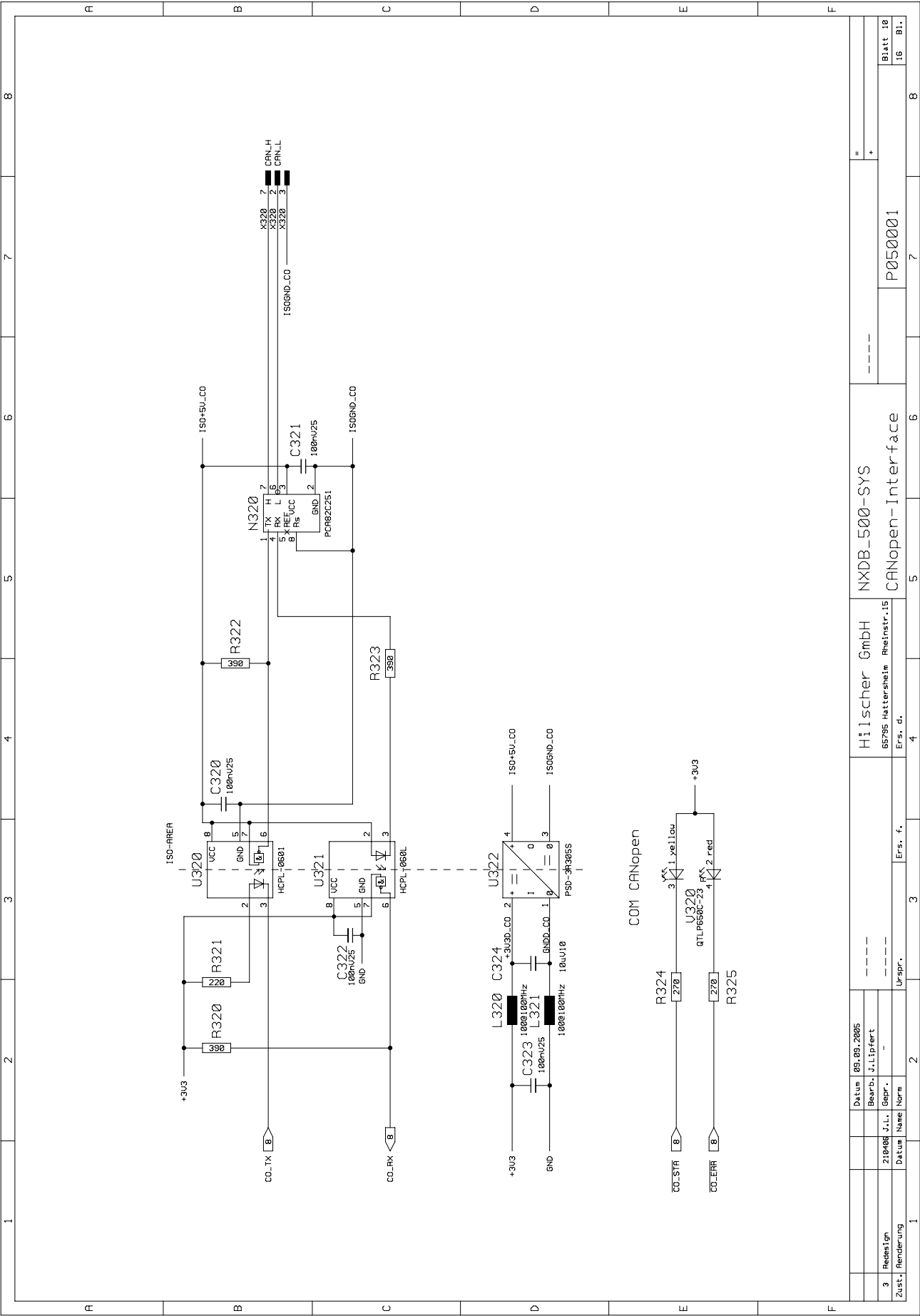


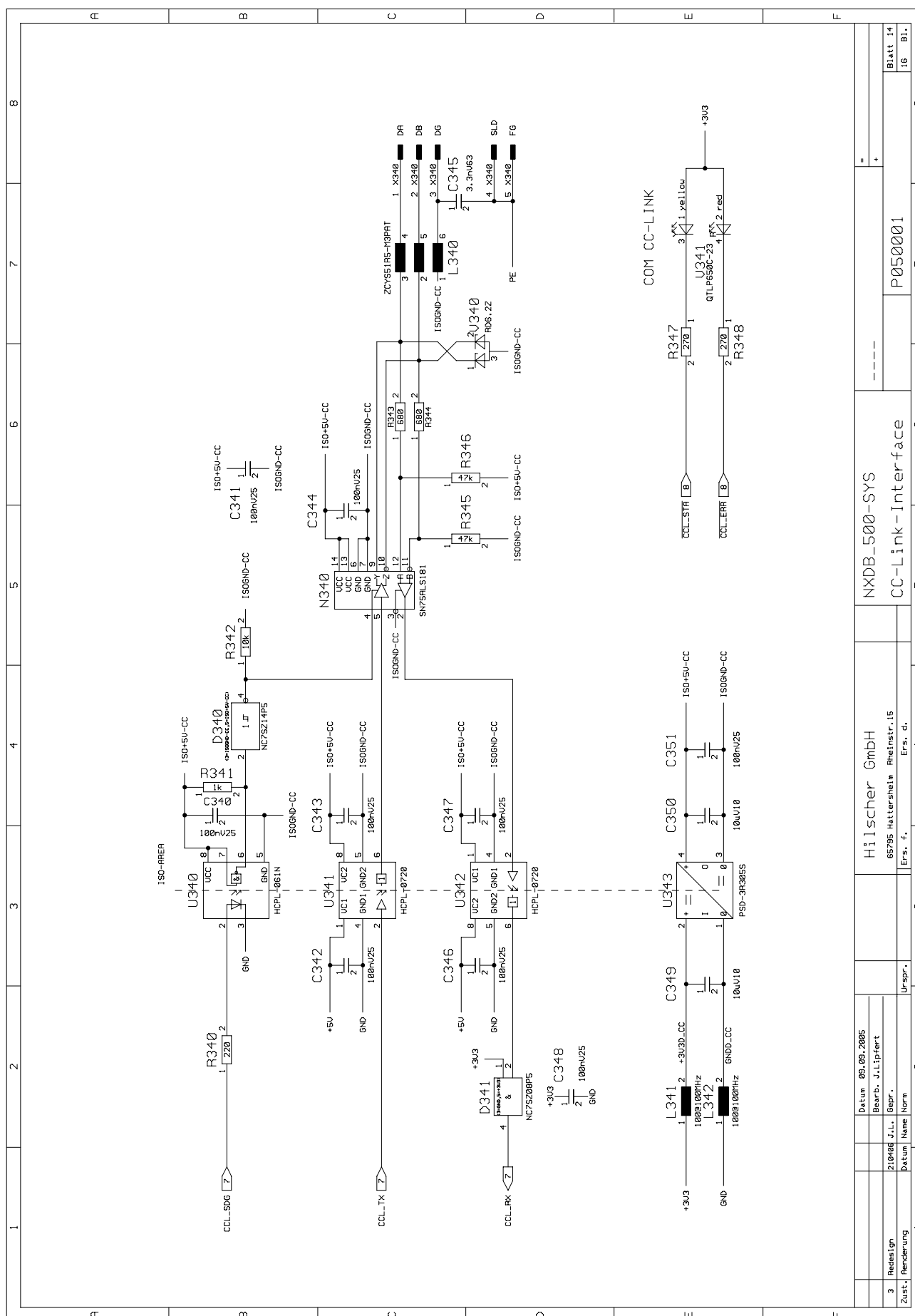


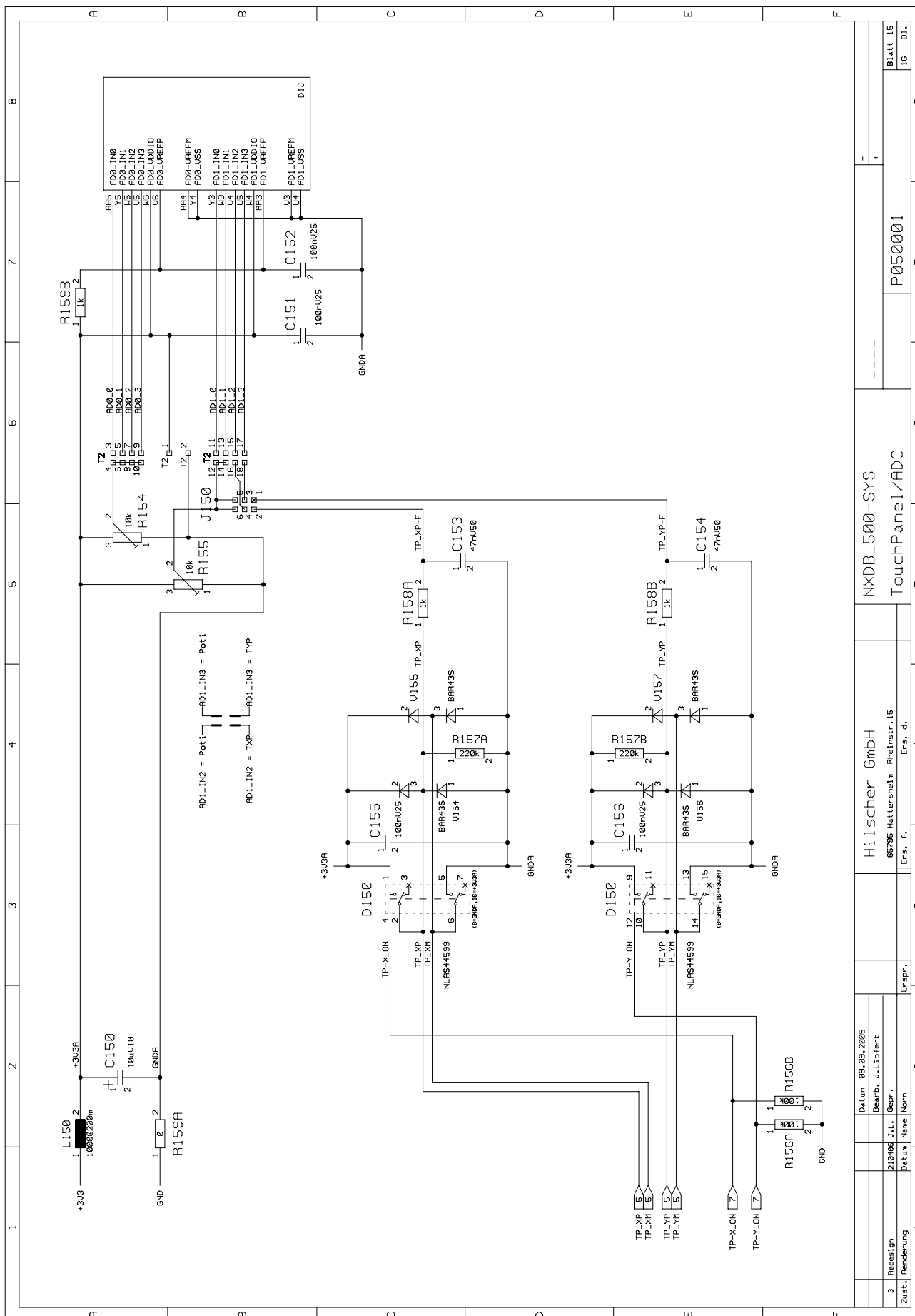












8.1 Bill of Materials of Reference Design

Please check the Hilscher website (netX Downloads) at www.hilscher.com for a detailed BOM of the reference design and further design documents.

9 Revision History

netX 500/100 Technical Data Reference Guide

Rev.	Date	Changes	Who
	14-07-2008	created	JL
	18-07-2008	Chapter 1.2 added	JL
	18-07-2008	Chapter 5.5 added	JL
	18-07-2008	Chapter 6.8 added	JL
	22-07-2008	Chapter 2.28 added	JL
(1.3p)	22-07-2008	Non public Pre-Release	JL
	01-09-2008	Chapter 2.27.2, corrected basic circuit figure	JL
	13-11-2008	Chapter 5.4.8 , replaced completely	JL
	13-11-2008	Chapter 2.11.6, added min. value for Tnrdyvalid, revised note	JL
	13-11-2008	Chapter 5.4.13, revised t2, t3, t6 and Note 1), added Note 2)	JL
	14-11-2008	Chapter 2.26, added circuit for unused Ethernet interface	JL
	14-11-2008	Chapter 2.13, added circuit for netX 100 or netX 500 with unused RTC	JL
	20-11-2008	Chapter 5.4.1, added output characteristics of host interface buffers	JL
	05-12-2008	Changed host interface buffer spec from IO9C (9mA) to IO18C (18mA)	JL
1.3	08-12-2008	Released	JL
	19-12-2012	Chapter 5.5, values from December 2012	HH
1.4	04-05-2015	Released	HH
	04-05-2015	Section 6.11: 'NETX 100 (PACKAGE480)' added	HH
	04-05-2015	Section 9: History about Product Brief removed.	HH
1.5	27-07-2015	Section 6.12: Box dimensions updated.	HH
1.6	24-11-2015	Section 1.4 added.	HH
	24-11-2015	Section 6.6: ENC_MP0 and ENC_MP1 added to table.	HH
	24-11-2015	Section 6.13 added.	HH
1.7	2015-12-16	Section 6.6: Signal DPM_ALE added.	HH
	2015-12-16	Section 6.7: Correction V20 is PWM1B_W, V21 is PWM1B_Wn, R17 is PWM0E_RSV, T21 is PWM1B_RSV	HH
1.8	2017-08-11	Section 6.4: Signals XM0 added to table.	HH
	2017-08-11	Section 5.2: order of supply voltages added.	HH

10 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.
Pune, Delhi, Mumbai
Phone: +91 8888 750 777
E-Mail: info@hilscher.in

Italy

Hilscher Italia S.r.l.
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.
Seongnam, Gyeonggi, 463-400
Phone: +82 (0) 31-789-3715
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