# Design-In Guide

# netX 51/52

Hilscher Gesellschaft für Systemautomation mbH

www.hilscher.com

DOC120102DG07EN | Revision 7 | English | 2015-08 | Released | Public

# Table of Contents

**1 Introduction ... 4**
1.1 About this document ... 4
1.2 List of revisions ... 5
1.3 References to documents ... 5
1.4 Legal notes ... 6
1.4.1 Copyright ... 6
1.4.2 Important notes ... 6
1.4.3 Exclusion of liability ... 7
1.4.4 Warranty ... 7
1.4.5 Export regulation ... 7
1.4.6 Registered trademarks ... 8

**2 Basic concepts ... 9**
2.1 netX 51/52 - introduction ... 9
2.2 netX 51 or netX 52 – Which netX is the right one? ... 10
2.3 netX 51/52 – Basic design structures ... 11
2.4 Design checklist ... 12
2.4.1 Checklist for loadable firmware ... 12
2.4.2 Checklist for linkable object module ... 12
2.5 Design examples ... 13
2.5.1 Communication interface ... 13
2.5.2 Gateway ... 14
2.5.3 Encoder interface ... 15
2.5.4 IO-Link Proxy for all real-time Ethernet systems ... 16

**3 Basic circuits ... 17**
3.1 Power supply ... 17
3.1.1 Core voltage regulator ... 17
3.2 System clock ... 18
3.3 Power-on reset and reset in ... 19
3.4 netX 51/52 chip mode selection ... 20
3.5 Boot options ... 21
3.5.1 Configured by the RDY/RUN pins ... 21
3.5.2 Configuration and parameterization by the security EEPROM ... 22
3.5.3 System RDY/RUN LED ... 24
3.5.4 Host interface mode during boot up ... 25
3.6 External memory ... 26
3.6.1 Serial memory interface ... 27
3.6.1.1 SPI Flashs ... 27
3.6.1.2 QSPI Flash ... 28
3.6.1.3 MMC/SD card ... 30
3.6.2 Parallel memory interface ... 32
3.6.2.1 SRAM and Flash ... 32
3.6.3 SDRAM ... 35
3.7 Host interface ... 36
3.7.1 Dual-port memory ... 37
3.7.1.1 8/16/32-bit data width and dual-port memory size ... 37
3.7.1.2 Control lines ... 38
3.7.1.3 Non-multiplexed mode ... 39
3.7.1.4 Multiplexed mode ... 40
3.7.1.5 Ready/Busy signal ... 41
3.7.2 Serial port memory (SPI/QSPI access to DPM) ... 42
3.7.3 Extension bus ... 44
3.7.3.1 Non-multiplexed mode ... 44
3.7.3.2 Multiplexed mode ... 46
3.7.4 MII port 2 acts as a PHY or MAC interface ... 48
3.7.5 SDRAM memory extension ... 50
3.7.6 MMIO signals ... 51
3.7.7 PIO signals ... 51
3.7.8 External pull-ups/pull-downs, unused signals ... 51
3.8 Multiplexed IO matrix (MMIO) ... 52
3.9 General purpose IOs ... 55
3.10 Serial interfaces ... 56

3.10.1 UARTs ... 56
3.10.2 USB ... 57
3.10.3 SPI ... 58
3.10.4 I2C ... 58
3.10.5 CAN ... 59
3.10.6 Ethernet diagnostic port ... 59
3.11 IO-Link ... 60
3.12 Fieldbus interfaces ... 61
3.12.1 CANopen interface ... 62
3.12.2 CC-Link interface ... 63
3.12.3 DeviceNet interface ... 64
3.12.4 PROFIBUS interface ... 65
3.12.5 Fieldbus status LEDs ... 66
3.12.6 Fieldbus address switches ... 66
3.13 Real-time Ethernet (RTE) interface ... 67
3.13.1 Twisted pair ... 67
3.13.2 Fiber optic ... 70
3.13.2.1 Transceivers with internal AC-termination ... 70
3.13.2.2 Diagnostic monitoring interface ... 70
3.13.3 Unused Ethernet PHYs ... 73
3.13.4 Ethernet status LEDs ... 74
3.13.4.1 Ethernet communication status LEDs ... 74
3.13.4.2 Real-time Ethernet protocol status LEDs ... 75
3.13.5 RTE device address switches ... 77
3.13.6 Real-time Ethernet synchronization signals ... 78
3.13.7 Special real-time Ethernet requirements ... 79
3.13.7.1 Fast StartUp – Special feature of PROFINET ... 79
3.13.7.2 QuickConnect$^{TM}$ – Special feature of EtherNet/IP ... 79

**4 Debug and test interfaces ... 80**
4.1 JTAG interface ... 80
4.1.1 Hilscher's 'mini-JTAG' connector ... 81
4.2 ETM interface ... 83
4.3 Boundary scan test ... 84

**5 Resource overview ... 85**
5.1 rcX kernel ... 85
5.2 Memory requirements of Hilscher stacks ... 86

**6 General design considerations ... 87**
6.1 Thermal behavior ... 87
6.1.1 Basics ... 87
6.1.2 Estimates ... 87
6.1.3 Recommendations ... 88
6.1.4 Rules of thumb ... 89
6.2 EMC behavior ... 90
6.2.1 Layer stack ... 90
6.2.2 Decoupling capacitors ... 91
6.2.3 Reset lines ... 92
6.2.4 Clock circuits ... 92
6.2.5 Ethernet interface ... 93
6.2.6 Memory bus ... 94
6.2.7 Planes ... 95
6.2.8 VIAs and signal fan out under netX 51/52 ... 96

**7 Reference section ... 97**
7.1 Crystals ... 97
7.2 Serial Flash memory / EEPROM ... 97
7.3 Parallel Flash memory ... 98
7.4 SDRAM ... 98
7.5 Other components ... 98

**8 Appendix ... 99**
8.1 List of tables ... 99
8.2 List of figures ... 99
8.3 Contacts ... 101

# 1 Introduction

## 1.1 About this document

This document is directed to hardware developers creating a hardware design with a communication controller of the Hilscher's netX family. It does not explain netX technology and features, which is covered by the corresponding Technical Data Reference Guides.

It describes the standard circuitry around all netX interfaces like memory interface (SDRAM, FLASH) USB, UARTs, XMACs (Ethernet and field bus), LCD, as well as power supply, reset and clock circuits along with the standard netX I/O resources (PIOs, GPIOs) that have been assigned a default functionality at Hilscher. This includes Status LEDs, control signals and sync signals for RTE applications.

Although the system designer is basically completely free to select any available I/Os for I/O purposes, it is important that he complies with the standard port definitions, whenever loadable firmware (LFW) from Hilscher is to be used, as this kind of firmware necessarily assumes compliance with Hilscher standard assignments.

**Note:** Designers should be aware, that not all components supported by netX hardware (e.g. parallel FLASH) are necessarily also supported by existing software/firmware or tools from Hilscher! Hence it is strongly recommended, to consult the feature table in the following chapter, to make sure that all desired hardware features of the planned design are eventually supported by the firmware that will run on the design. This applies not only, but particularly to customers planning to use loadable firmware (LFW) from Hilscher instead of doing own firmware development.

Resources that are currently not supported by loadable firmware (LFW) or where no drivers/code are yet available may still already be supported by existing Hilscher devices (e.g. Gateways). It is hence recommended to check with Hilscher sales, if there is already an existing solution for your problem. Further, Hilscher offers several custom design services for netX hardware and software, as well as manufacturing services, providing an easy way to your custom product. For detailed information and quotes, please contact Hilscher sales.

Hilscher also offers a schematic review service, allowing your hardware design to be checked by netX experienced hardware engineers. Hilscher Sales will be happy to provide an individual quote for this service, after receiving your schematics (PDF format).

**Note:** Before starting a design, it is strongly recommended, to consult the latest Errata Sheets (available on the Hilscher website www.hilscher.com) of the netX controllers!

## 1.2 List of revisions

| Rev | Date | Name | Chapter | Revision |
|---|---|---|---|---|
| 5 | 2014-12-16 | HJH / SH / HH | All | Complete review and update with additional details. |
| 6 | 2015-01-23 | HH | 3.7.1.1 | Correction: High signal to select 8-bit mode, low signal to select 16-bit mode. |
| 7 | 2015-08-26 | HN/HH | 2.3 | Section *netX 51/52 – Basic design structures*: Figure 3 updated. |
| | | | 2.5.1 | Section *Communication interface*: Figure 5: Resistors added (correction). |
| | | | 2.5.3 | Section *Encoder interface*: Figure 9: Level shifter added (correction). |
| | | | 3.7.4 | Section *MII port 2 acts as a PHY or MAC interface*: Figure 49 updated. |
| | | | 3.7.4 | Section *MII port 2 acts as a PHY or MAC interface*: Table 15 signals MDC and MDIO added. |
| | | | 3.10.2 | Section *USB*: Figure 53 corrected. |
| | | | 3.12.2 | Section *CC-Link interface*: Figure 57 updated. |
| | | | 3.12.3 | Section *DeviceNet interface*: Figure 58 updated. |
| | | | 3.12.5, 3.12.6, 3.13.4.1, 3.13.4.2, 3.13.5 | Resistor value range added and MMIO numbers corrected. |
| | | | 3.13.2.2 | Section *Diagnostic monitoring interface*: Figure 64 corrected. |
| | | | 3.13.6 | Section *Real-time Ethernet synchronization signals*: EtherNet/IP Adapter added. |
| | | | 4.1 | Section *JTAG interface*: Figure 69 corrected. |
| | | | 5.2 | Section *Memory requirements of Hilscher stacks*: Table updated. |

*Table 1: List of revisions*

## 1.3 References to documents

This document refers to the following documents:

[1] Hilscher Gesellschaft für Systemautomation mbH: Technical Data Reference Guide, netX 51/52,Revision 2, English, 2013.

[2] Hilscher Gesellschaft für Systemautomation mbH: Dual-Port Memory Interface, English, Revision 12, 2013-03

[3] Hilscher Gesellschaft für Systemautomation mbH: netX Security Memory, Implementation and use, English, Revision 2, 2009-10

[4] Hilscher Gesellschaft für Systemautomation mbH: Migration Guide netX 50 to netX 51/52, English, Revision 5, 2013-08

[5] Hilscher Gesellschaft für Systemautomation mbH: Programming Reference Guide netX 51/52, English, Revision 4, 2015-01

*Table 2: References to documents*

## 1.4 Legal notes

### 1.4.1 Copyright

### 1.4.2 Important notes

The user manual, accompanying texts and the documentation were created for the use of the products by qualified experts, however, errors cannot be ruled out. For this reason, no guarantee can be made and neither juristic responsibility for erroneous information nor any liability can be assumed. Descriptions, accompanying texts and documentation included in the user manual do not present a guarantee nor any information about proper use as stipulated in the contract or a warranted feature. It cannot be ruled out that the user manual, the accompanying texts and the documentation do not correspond exactly to the described features, standards or other data of the delivered product. No warranty or guarantee regarding the correctness or accuracy of the information is assumed.

We reserve the right to change our products and their specification as well as related user manuals, accompanying texts and documentation at all times and without advance notice, without obligation to report the change. Changes will be included in future manuals and do not constitute any obligations. There is no entitlement to revisions of delivered documents. The manual delivered with the product applies.

Hilscher Gesellschaft für Systemautomation mbH is not liable under any circumstances for direct, indirect, incidental or follow-on damage or loss of earnings resulting from the use of the information contained in this publication.

### 1.4.3 Exclusion of liability

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

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

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

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

### 1.4.4 Warranty

Although the hardware and software was developed with utmost care and tested intensively, Hilscher Gesellschaft für Systemautomation mbH does not guarantee its suitability for any purpose not confirmed in writing. It cannot be guaranteed that the hardware and software will meet your requirements, that the use of the software operates without interruption and that the software is free of errors. No guarantee is made regarding infringements, violations of patents, rights of ownership or the freedom from interference by third parties. No additional guarantees or assurances are made regarding marketability, freedom of defect of title, integration or usability for certain purposes unless they are required in accordance with the law and cannot be limited. Warranty claims are limited to the right to claim rectification.

### 1.4.5 Export regulation

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

### 1.4.6 Registered trademarks

CANopen® is a registered trademark of CAN in AUTOMATION - International Users and Manufacturers Group e.V. (CiA), Nuremberg.

CC-Link® is a registered trademark of Mitsubishi Electric Corporation, Tokyo, Japan.

DeviceNet® and EtherNet/IP® are trademarks of ODVA (Open DeviceNet Vendor Association, Inc).

EtherCAT® is a registered trademark and a patented technology of Beckhoff Automation GmbH, Verl, Germany, formerly Elektro Beckhoff GmbH.

Modbus® is a registered trademark of Schneider Electric.

Powerlink® is a registered trademark of B&R, Bernecker + Rainer Industrie-Elektronik Ges.m.b.H, Eggelsberg, Austria

PROFIBUS® und PROFINET® are registered trademarks of PROFIBUS International, Karlsruhe.

Sercos interface® is a registered trademark of Sercos International e. V., Suessen, Germany.

I2C is a registered trademark of NXP Semiconductors, formerly Philips Semiconductors.

All other mentioned trademarks are property of their respective legal owners.

# 2 Basic concepts

## 2.1 netX 51/52 - introduction

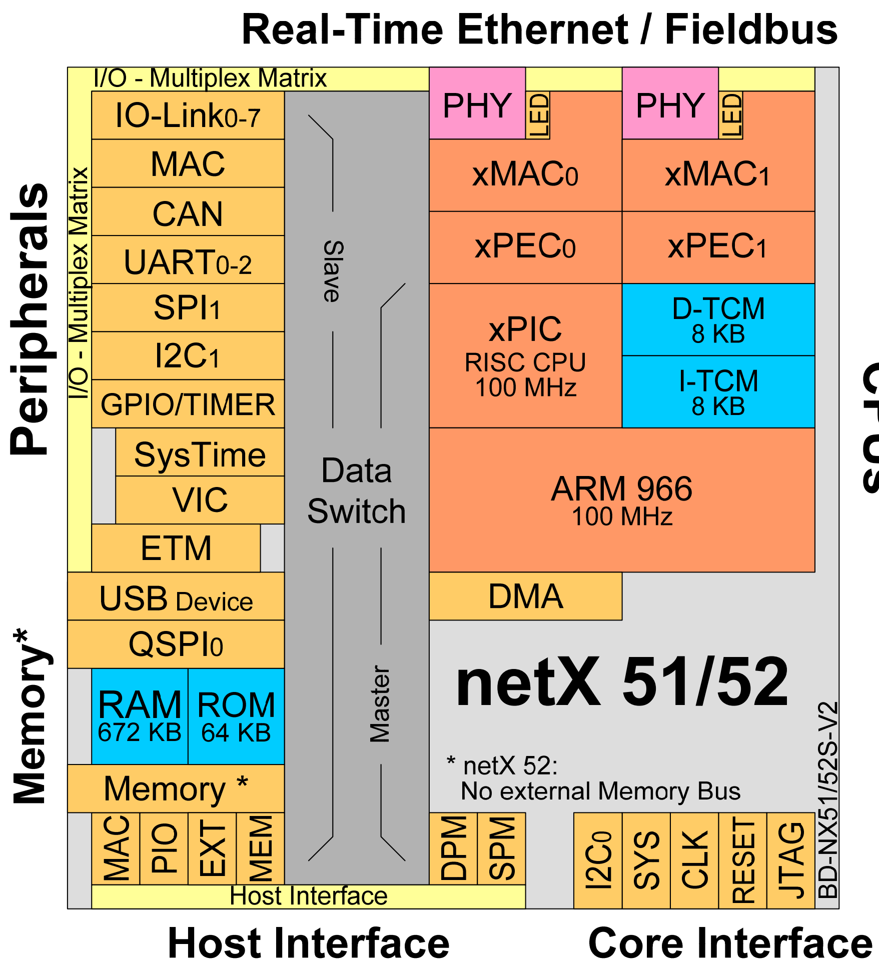


*Figure 1: netX 51/52 block diagram*

The netX 1/52 is a highly integrated network controller with two communication channels. The integrated PHYs can be programmed to run any important real-time Ethernet system. Apart from the 32 bit/100 MHz main CPU, it has an additional 32-bit/100 MHz RISC CPU for handling some background tasks. This helps achieve very fast bus cycles down to 100 us without influencing the protocol stack running on the ARM CPU.

It has a 672 KB internal SRAM and a QSPI/SPI interface to connect a serial Flash which contains the program code. Moreover, netX 51 has a memory interface for SDRAM, Flash or SRAM to enlarge the internal memory.

It incorporates many peripheral functions like 8-channel IO-Link master, CAN controller, a third Ethernet channel mainly used for diagnostic purposes. Of course, all standard interfaces like UART, USB device, SPI, and I2C master or slave are available, too. The multiplex matrix reduces the number of peripheral signals to the number required for the current application.

Peripherals and CPUs are connected via a switch matrix where data transfers simultaneously take place as long as they have a different destination or source area.

A special feature is the very flexible host interface which can be configured to work as DPM, intelligent serial port memory, extension bus or memory controller for SRAM/SDRAM.

## 2.2 netX 51 or netX 52 – Which netX is the right one?

The following schematic gives an overview of the signals and the differences between netX 51 and netX 52. To have the right understanding: both devices contain the same piece of silicon. The netX 52 does not have so many peripheral I/O signals and no external memory bus. Thus, the housing is smaller and the chip is cheaper.

The netX 51/52 needs a 3.3 V power supply for the I/Os and a 1.5 V for the core voltage. Only a few external components are needed e.g. a crystal, a reset generator and additional memory depending on the application.

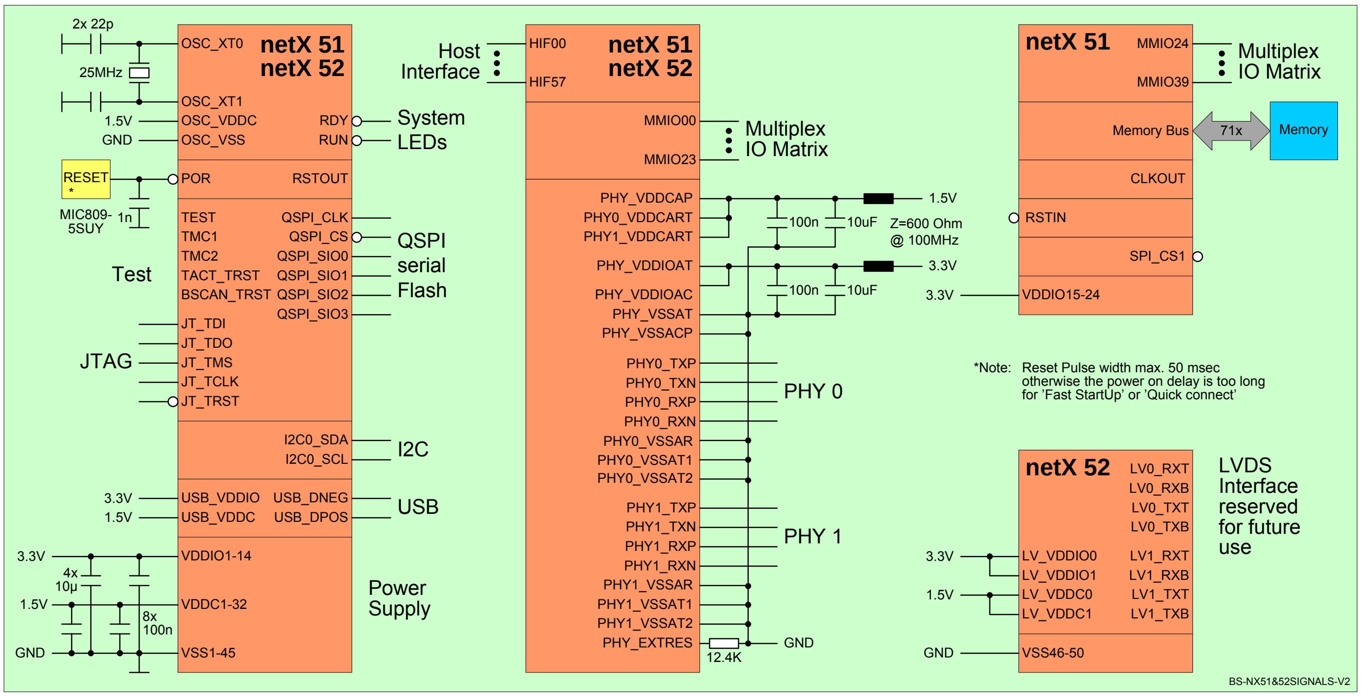


*Figure 2: netX 51/52 signals*

- The silicon inside the netX 51 or 52 is the same!
- netX 51 has a larger housing of 19x19 mm with 1 mm pitch and 324 pins plus
  - a memory bus to enlarge the memory with an external SDRAM, SRAM or Flash
  - a multiplex IO matrix with 40 I/Os: 16 more than netX 52, for that reason it can drive 8 IO-Link channels instead of 5 like netX 52
  - three signals: Clock out, reset in and SPI chip select 1
- netX 52 has a small housing of 15x15 mm with 0.8 mm pitch and 244 pins
  - netX 52 is pin-compatible with netX 6
  - netX 52 is cheaper than netX 51 due to a smaller housing and less pins

➔ For a pure communication interface: netX 52 is the right choice

  - the protocol stacks as loadable firmware do not need an external SDRAM
  - easier layout, smaller footprint, lower costs for components
  - using standard schematic for finalizing the design quickly without risk

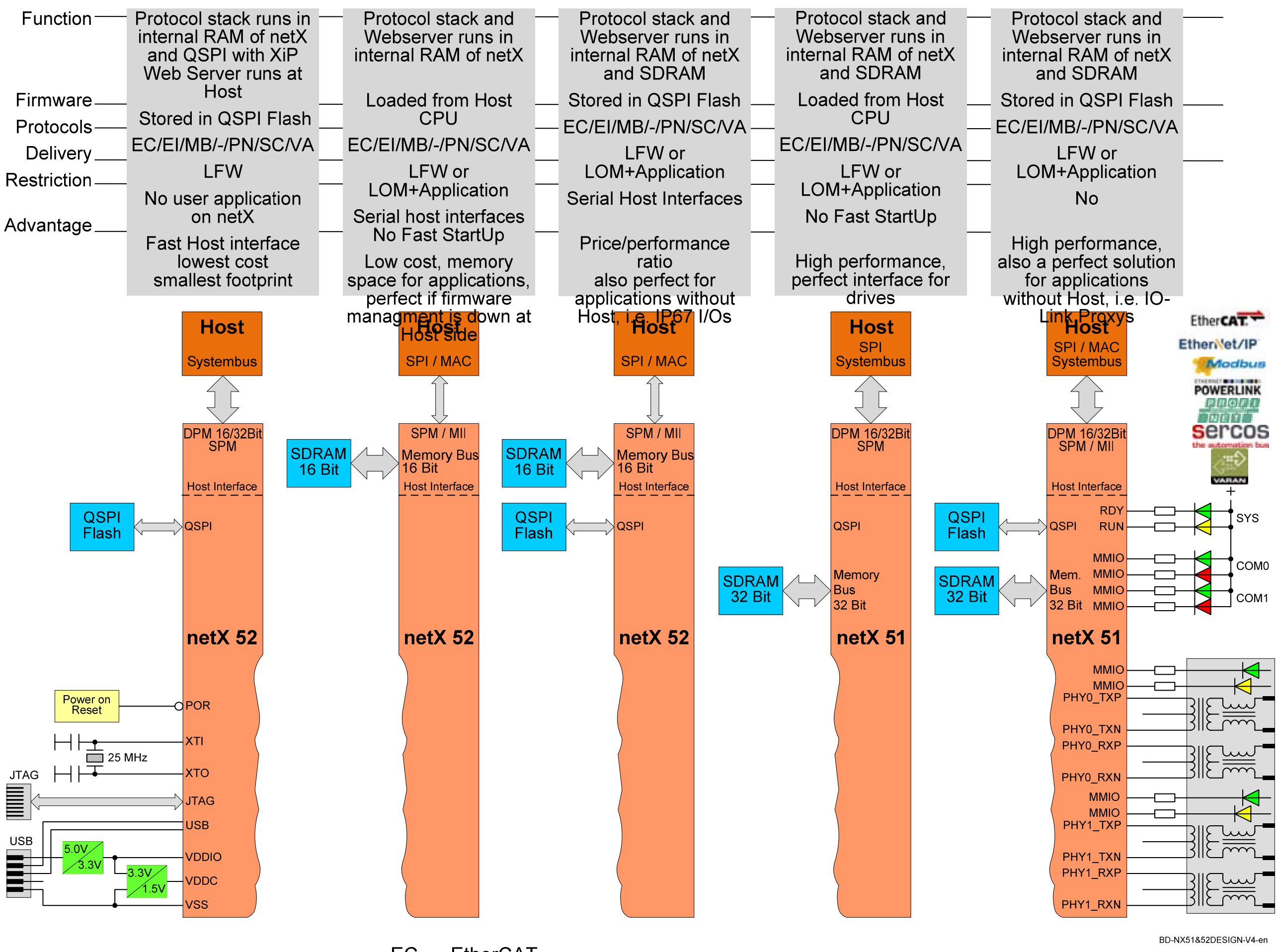
➔ Running additional application software on the chip: Use netX 51 to have more memory space

➔ For a stand-alone application, i.e. an IP67 IO block where the host interface can be configured as memory bus to connect a SDRAM: netX 52 might be a good option.

## 2.3 netX 51/52 – Basic design structures

If you start developing a design with netX 51/52, you have to decide on whether you want to use netX 52 or whether you want to add a lot of application code to the protocol stack. In that case, the netX 51 would be the right choice.

The following diagram shows basic design structures:



| | | | | | |
|---|---|---|---|---|---|
| Function | Protocol stack runs in internal RAM of netX and QSPI with XiP Web Server runs at Host | Protocol stack and Webserver runs in internal RAM of netX | Protocol stack and Webserver runs in internal RAM of netX and SDRAM | Protocol stack and Webserver runs in internal RAM of netX and SDRAM | Protocol stack and Webserver runs in internal RAM of netX and SDRAM |
| Firmware | Stored in QSPI Flash | Loaded from Host CPU | Stored in QSPI Flash | Loaded from Host CPU | Stored in QSPI Flash |
| Protocols | EC/EI/MB/-/PN/SC/VA | EC/EI/MB/-/PN/SC/VA | EC/EI/MB/-/PN/SC/VA | EC/EI/MB/-/PN/SC/VA | EC/EI/MB/-/PN/SC/VA |
| Delivery | LFW | LFW or LOM+Application | LFW or LOM+Application | LFW or LOM+Application | LFW or LOM+Application |
| Restriction | No user application on netX | Serial host interfaces No Fast StartUp | Serial Host Interfaces | No Fast StartUp | No |
| Advantage | Fast Host interface lowest cost smallest footprint | Low cost, memory space for applications, perfect if firmware management is down at Host side | Price/performance ratio also perfect for applications without Host, i.e. IP67 I/Os | High performance, perfect interface for drives | High performance, also a perfect solution for applications without Host, i.e. IO-Link Proxys |

| | | | |
|---|---|---|---|
| EC | EtherCAT | LFW | Ladbare Firmware |
| EI | EtherNet/IP | LOM | Linkable Object Module |
| MB TCP | Open Modbus/ | | |
| PL | POWERLINK | | |
| PN | PROFINET | MII | Multi Media Interface |
| SC | Sercos | DPM | Dual-Port Memory |
| VA | Varan | SPM (SPI) | Serial-Port Memory |

*Figure 3: netX 51/52 – Basic design structures*

Most customers are running Hilscher protocol stacks on the netX. If you also want to do this, you have to follow instructions on how to develop the hardware that our protocol stacks can run without any modification.

You will find most of these instructions in this manual. On the next pages, we provide various design examples. We also have an excellent support team. If you do not have the resources, we can develop and produce the requested device for you.

## 2.4 Design checklist

### 2.4.1 Checklist for loadable firmware

☑ Do I have a possibility to load my firmware from e.g. Flash, host controller, Ethernet?

☑ Are the boot options and system LEDs set up correctly?

☑ How do I get my firmware into my device USB, UART, host, Ethernet?

☑ If I need additional memory, is it set up correctly?

☑ Is my fieldbus / real-time Ethernet Interface connected correctly?

☑ Are my fieldbus / real-time Ethernet LEDs connected?

☑ Is my host controller associated properly?

☑ Do I have all basic components like power supply, power-on reset, system clock?

### 2.4.2 Checklist for linkable object module

☑ Do I have a possibility to load my firmware from e.g. Flash, host controller, Ethernet?

☑ Are the boot options and system LEDs set up correctly?

☑ How do I get my firmware into my device USB, UART, host, Ethernet?

☑ Is my external memory set up correctly?

☑ Is my fieldbus / real-time Ethernet interface connected correctly?

☑ Are my fieldbus / real-time Ethernet LEDs connected?

☑ Do I have all basic components like power supply, power-on reset, system clock?

☑ Do I need some debug options?

☑ Are all of my peripherals e.g. CAN, GPIOs, I2C, SPI, UARTs connected?

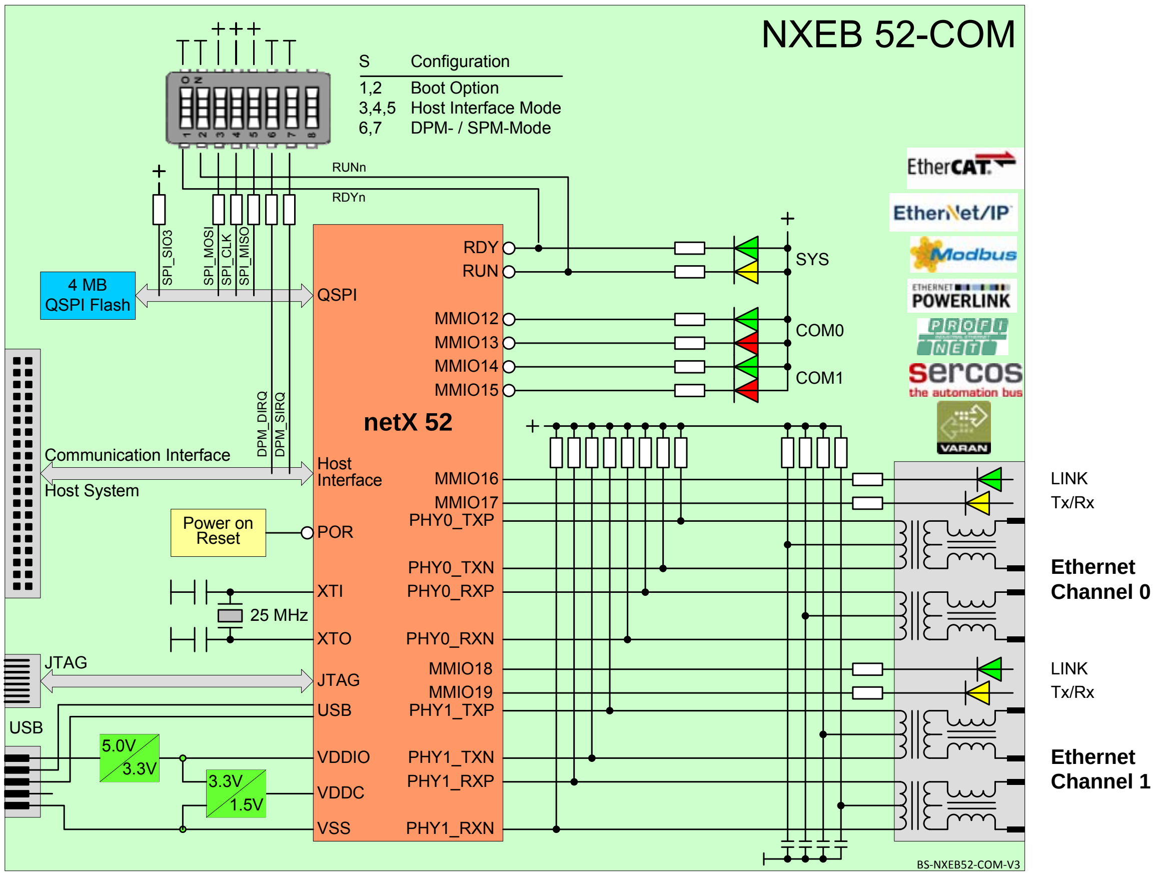
## 2.5 Design examples

### 2.5.1 Communication interface

- Standard netX 52 core for all real-time Ethernet systems
- netX 52 with all core components on 25 x 25 mm²

*Figure 4: NXEB 52-COM*

This is the most frequently used design for a general communication interface. It needs only a few external components. The host interface can be configured from a high speed 32-bit parallel DPM to a very fast SPI connection. Alternatively, you can configure a UART, CAN or some digital I/Os on the same pins. In a final design, resistors will substitute the DIP switch.



| S | Configuration |
|---|---|
| 1,2 | Boot Option |
| 3,4,5 | Host Interface Mode |
| 6,7 | DPM- / SPM-Mode |

*Figure 5: NXEB 52-COM block diagram*

## 2.5.2 Gateway

- Gateway between real-time and standard Ethernet
- Use MII port at the host interface in combination with 8 MB SDRAM

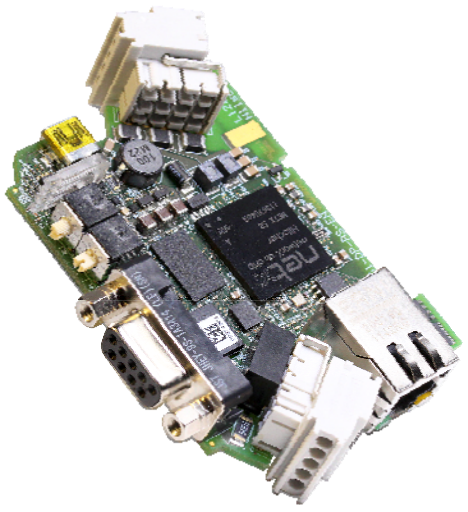


*Figure 6: NXEB 52-GATEWAY*

The third Ethernet port is often used as a local diagnostic port. It can also be used as a type of gateway to transfer standard TCP/IP frames into the non-real-time channel of the real-time Ethernet system e.g. EtherCAT or Sercos.

The dedicated CAN port is a nice option for gateways between a legacy system and the new real-time Ethernet world.

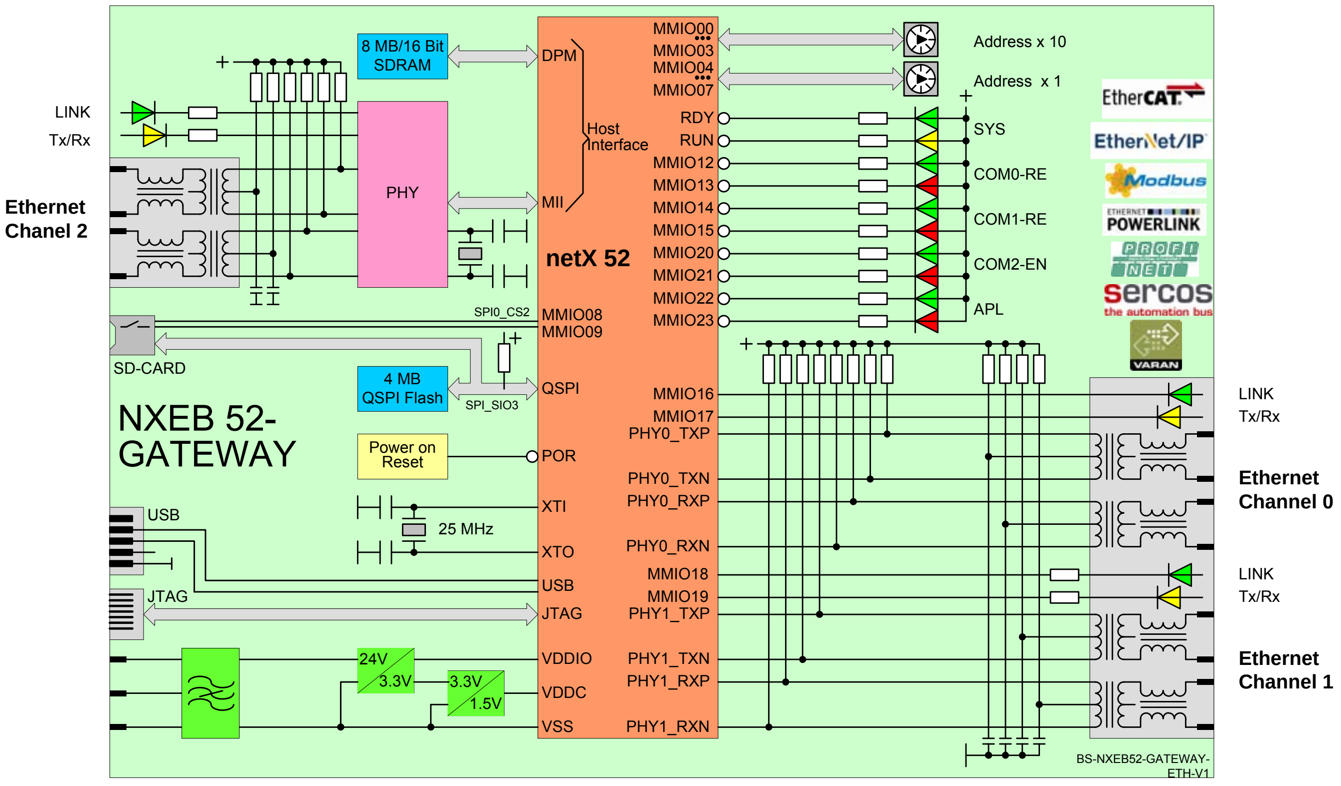


*Figure 7: NXEB 52-GATEWAY block diagram*

### 2.5.3 Encoder interface

- Compact 53 mm diameter
- BiSS interface implemented on xPIC
- External SDRAM via host interface due the encoder profile
- 85 °C operating temperature with cooling interface to the encoder housing

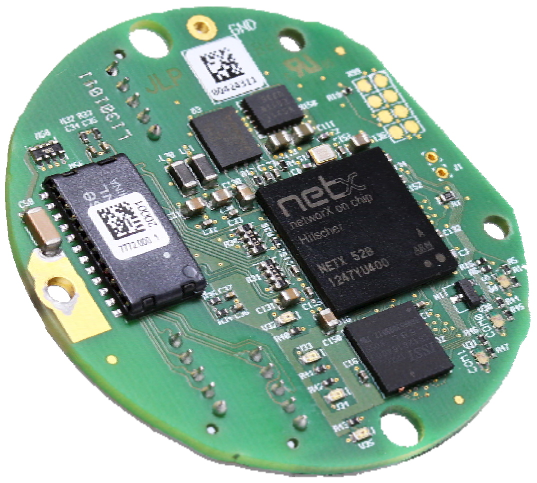


*Figure 8: NXEB 52-ENCODER*

By implementing the *BiSS*$^R$ protocol on the xPIC we have realized a glue-less connection between the netX 51/52 and the *BiSS*$^R$ chip.

The special requirement of an operating temperature of +85°C can be met with a closed temperature coupling between the netX and the encoder housing.

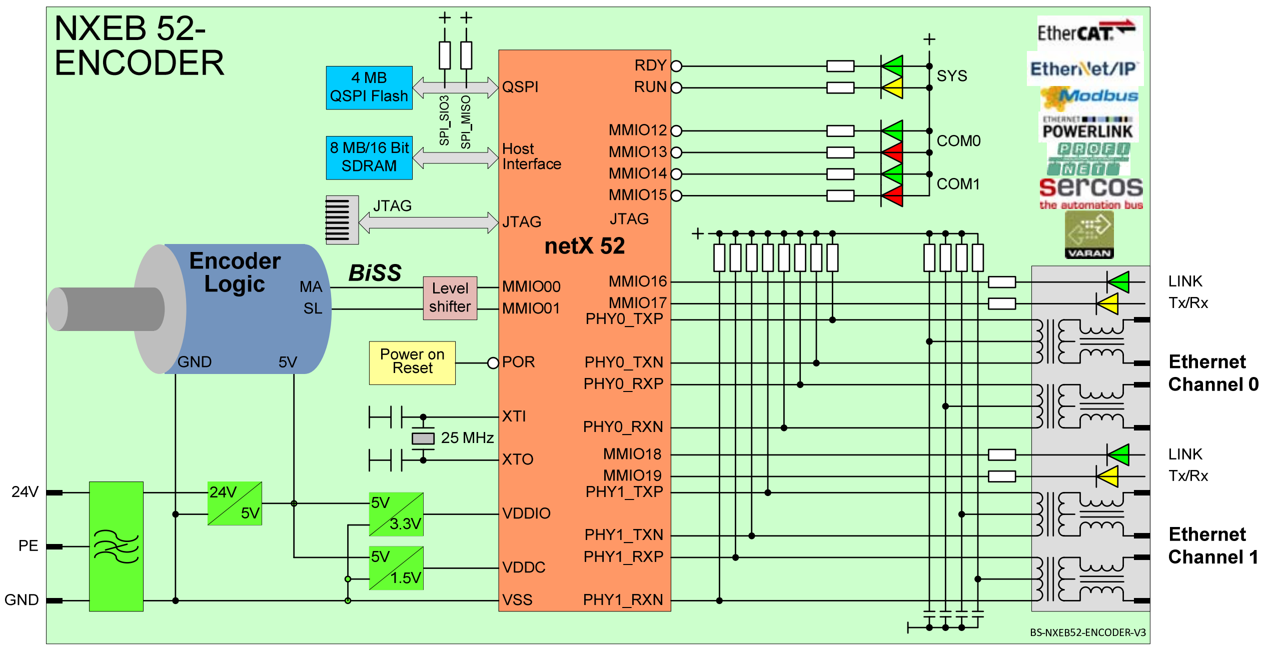


*Figure 9: NXEB 52-ENCODER block diagram*

### 2.5.4 IO-Link Proxy for all real-time Ethernet systems

- IO-Link gateway based on netX 51
- One hardware design for all real-time Ethernet systems
- 8 channels, IO-Link V1.1, implemented on xPIC
- Performance

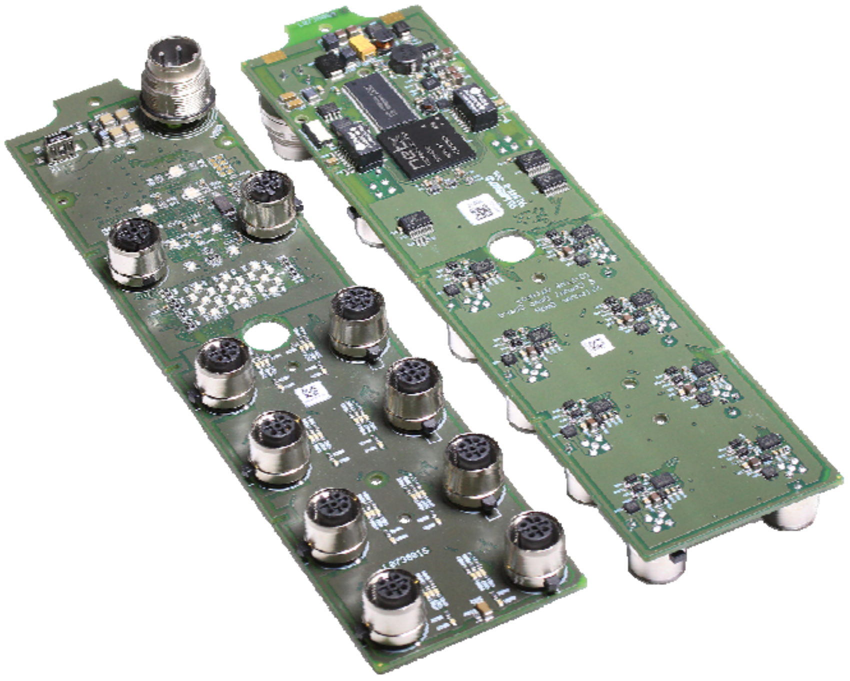


*Figure 10: IO-Link Gateway with netX 51*

In this design example we also use the xPIC to run the lower protocol layer. This results in a very cost-effective design for IP67 IO-Link IO-blocks. Beside the compact hardware design, the proxy functionality is available between IO-Link and any important real-time Ethernet system.

# 3 Basic circuits

## 3.1 Power supply

netX 51/52 requires an I/O voltage supply of 3.3 V and a core voltage supply of 1.5 V.

For worst case scenarios, the 3.3 V supply should be able to deliver a current of 350 mA for the netX part (netX, memory, etc.) of the circuit. If the core voltage regulator is also supplied by the 3.3 V rail, the max. current will increase accordingly.

The core supply should be able to deliver 900 mA.

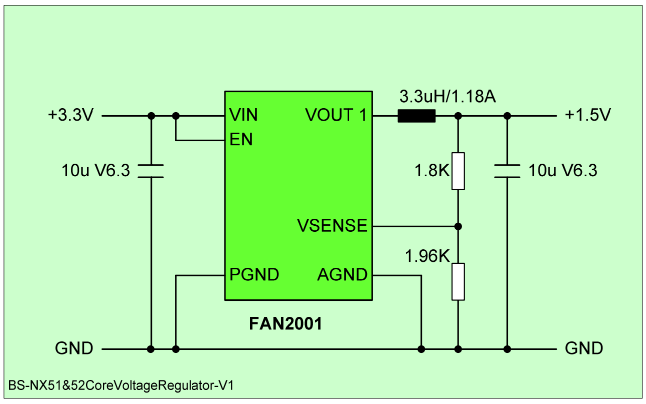
Lower max. currents may be sufficient for certain applications e.g. applications that do not use the Ethernet ports. For details on power consumption see the Technical Data Reference Manual of the netX chip (see reference [1]).

It is recommended to implement a separate core voltage supply for the netX even if a 1.5 V supply is required by other parts of the system (e.g. FPGA). The output voltage of this separate supply should be “programmable” either by approx. digital input pins or by external resistors. This allows you to continue using the same design after the netX type used has gone through a die shrink process, which always results in a reduced core voltage.

### 3.1.1 Core voltage regulator

A Hilscher standard circuit for the core voltage regulator uses a FAN2001 (Fairchild Semiconductor) step-down DC-DC converter and is in Figure 11. The complete part name is FAN2001MPX. The inductor is a CR32NP-3R3 (Sumida). The capacitors are ceramic types (X5R/X7R), resistors 1%.

An alternative part, especially for designs with space constraints, is the EN5312Q (Enpirion), which delivers 1A, has an integrated inductor and provides 3 digital inputs for setting the output voltage, requiring only two 10 µF ceramic caps as additional external components (see www.enpirion.com for detailed information).



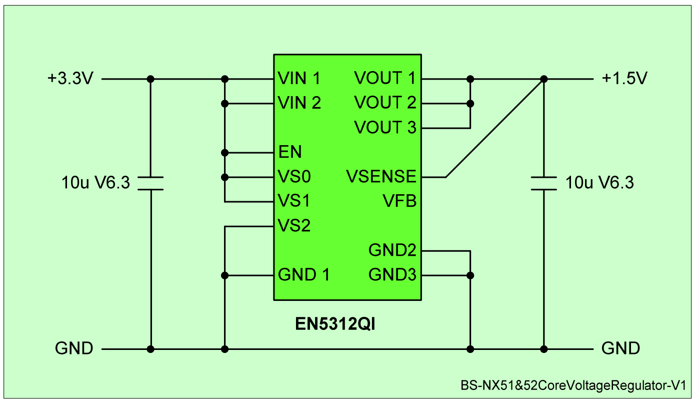
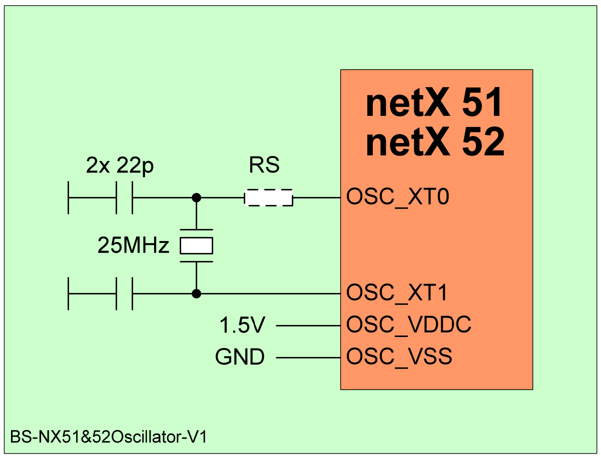


*Figure 11: netX core voltage regulator*

## 3.2 System clock

netX 51/52 uses an internal oscillator with an external crystal or an external oscillator for generating the 25 MHz base clock. This clock will be stabilized by a PLL which generates all internal clocks of the chip.

Figure 12 shows the circuits for the system clock generation:



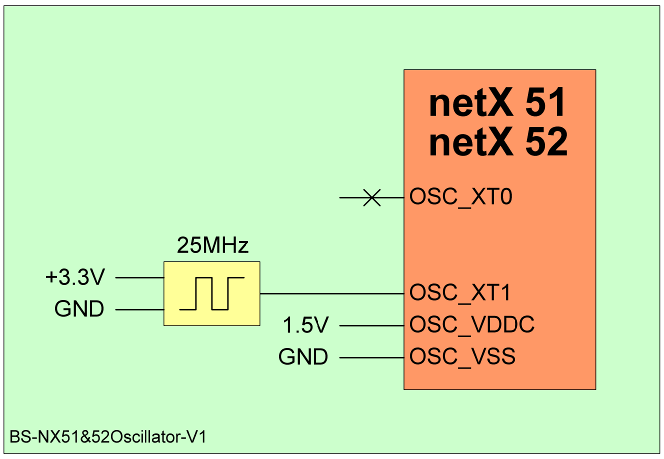


*Figure 12: netX 51/52 system oscillator circuit*

The values of the capacitors and the serial resistor (Rs) depend on the used crystal. If the same crystal is used all Hilscher netX products are equipped with, the resistor Rs will not be used and is replaced by a wire. The capacitors should have a value of 22 pF.

If a different crystal is used, you have to consult the data sheet of the crystal to determine the appropriate values.

The Hilscher standard netX system crystal is an ABM7-25.000 MHz-D2Y-T manufactured by Abracon.

Alternatively, an external oscillator can be used and is connected according to the schematic (Figure 12, right).

**Note:** An external oscillator or a different crystal you wish to use must have a frequency of 25 MHz and a max. tolerance of +/- 100 ppm throughout the complete temperature range, the design will be specified for!

Q1: We already have crystals or oscillators in store with a different frequency or a higher tolerance. Can we also use these parts for the system oscillator?

A1: No. The 25 MHz clock is the base for all other netX clocks. It influences any timing around the netX, like SDRAM timing, Baud rates, Ethernet timing, etc. Deviating from the specified frequency will most likely result in a system that does not work properly.

## 3.3 Power-on reset and reset in

The netX 51/52 provides a Schmitt-trigger power-on reset (PORn), netX 51 an additional Reset Input (RSTINn). The POR signal should be generated by a dedicated reset circuit which also checks that the VDDIO voltage has reached the min. level of 3 V.

Reset generator components are often available with a push/pull or an open drain output. If the design uses the JTAG Interface of the netX, select an open drain type, since this allows you to simply connect the reset signal from the JTAG connector (which is also specified as open drain) to the output of the reset generator. Of course, a pull-up resistor must be attached to the PORn signal if open drain reset sources are used.

The optional RSTINn, commonly used by an external host processor to reset the netX, also provides a Schmitt-trigger gate. The netX 51 provides an internal pull-up resistor (nominal 50 kΩ) and could be left open if not used. However, it is recommended to tie it to VDDIO (+3.3V), since this can improve EMC behavior.

When placing the components during PCB design, place the reset source(s) near the reset inputs of the netX to keep the traces of the reset signals short. Routing reset signals all over the PCB may result in bad EMC behavior of the design, since ESD may cause undesired resets of the chip. We recommend connecting a 1 nF ceramic capacitor to GND and PORn close to the netX PORn pin. The following figure shows the standard reset circuits:

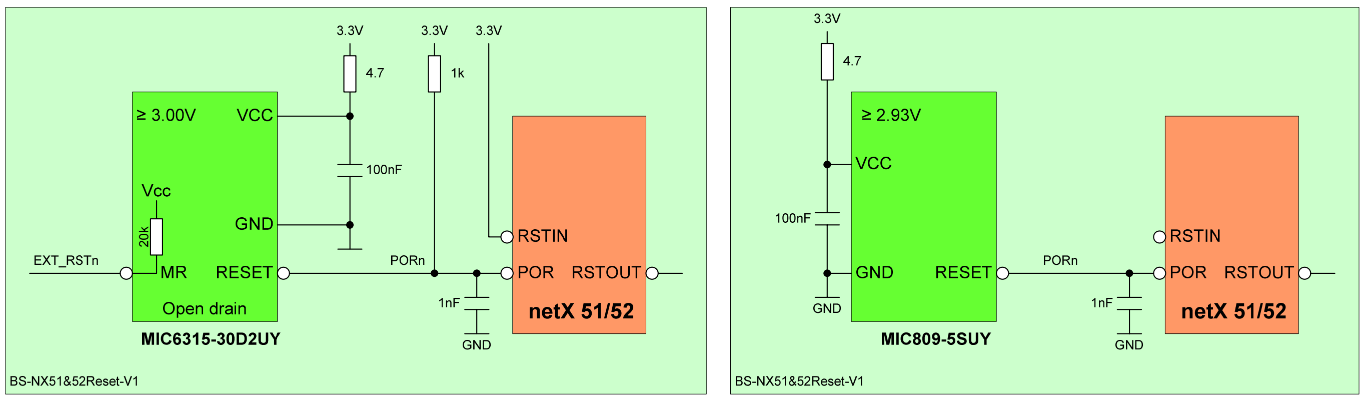


*Figure 13: netX 51/52 reset circuits*

| Reset device specification | MIC6315-30D2UY | MIC809-5SUY |
|---|---|---|
| Min. | 20 ms | 30 ms |
| Typ. | 28 ms | - |
| Max. | 44 ms | 66 ms |
| Reset level | 3.0 V ± 2,5 % (2.925 V … 3.075 V) | 2.93 V ± 2.4 % (2.85 V … 3.00 V) |
| Temperature range | –40 °C … +85 °C | –40 °C … +85 °C |

*Table 3: Reset device specification*

**Note:** In case of PROFINET (using the Fast StartUp function) and EtherNet/IP (using the QuickConnect function) the delay in the reset circuit must always be observed. The delay should NEVER exceed the max. of 66 ms. Make sure that the core voltage is stable and over 1.425 V before the PORn signal is released. An example for such a component is the MIC6315-30D2UY from the company Micrel.

Section *JTAG interface* on page 80 shows the reset circuit in combination with the JTAG interface.

## 3.4 netX 51/52 chip mode selection

As described in the previous chapter, netX 51 and netX 52 contain the same piece of silicon, i.e. the same die. Additional netX 51 can upgrade a netX 50 design by a direct replacement on the same PCB. These three modes have to be configured by two resistors.

On netX 51, the ROM code uses the pins MEM_A18/QSPI_SIO2 and MEM_A19/SQPI_SIO3 to read the configured chip mode and on netX 52 the pins QSPI_SIO2 and QSPI_SIO3. Both pins are sampled after a negative edge at the power-on reset and the internal PLL reaches a stable state:

| Chip Mode | netX 51 | MEM_A18/ QSPI_SIO2 | MEM_A19/ QSPI_SIO2 | Pin J14 | Pin J15 | Pin E14 | Pin D14 | Version register | USB ID |
|---|---|---|---|---|---|---|---|---|---|
| | netX 52 | QSPI_SIO2 | QSPI_SIO3 | - | - | Pin A17 | Pin A18 | | |
| netX 50 compatible | | open (low) | open (low) | MEM_A22 | MEM_A23 | High impedance | High impedance | 0x243 | 0x18 |
| netX 51 | | pull-up (high) | open (low) | MEM_A18 | MEM_A19 | DPM_A16 / EXT_A22 | DPM_A17 / EXT_A23 | 0x643 | 0x18 |
| netX 52 | | open (low) | pull-up (high) | - | - | DPM_A16 / EXT_A22 | DPM_A17 / EXT_A23 | 0xA43 | 0x19 |
| Undefined / do not use | | pull-up (high) | pull-up (high) | - | - | - | - | - | - |

*Table 4: netX 51/52 chip mode selection*

This achieves QSPI signals on netX 51, two more address signals at the host interface and changes at the version register and the USB ID. MEM_A18/19 substitute MEM_A22/23 because the highest memory addresses are normally not used.

**Note:** If netX 51 is used as a direct replacement for netX 50, you must use the netX 50-compatible mode. Otherwise you risk damage to the netX 51 because on a netX 50-based PCB pin E14 is connected with +1.5 V and pin D14 with GND. For more information on the direct replacement of netX 50, refer to the Migration Guide

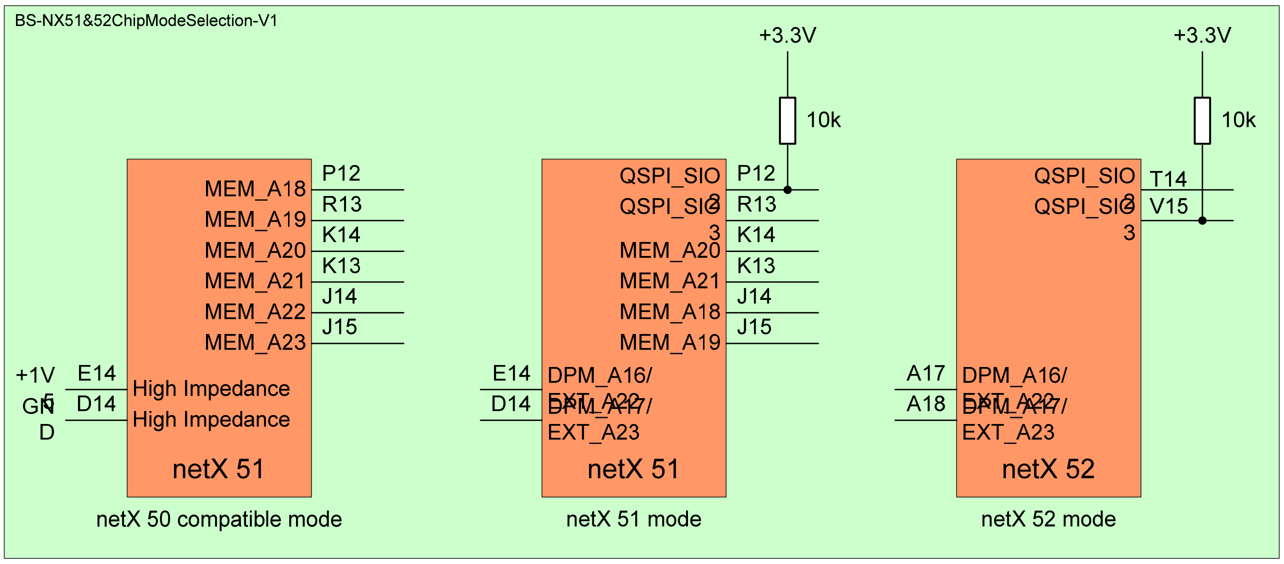


*Figure 14: netX 51/52 chip mode selection*

## 3.5 Boot options

Right after turning on the power or pressing the reset button, the netX searches for a bootable code image. Usually this image is a second stage boot loader or a firmware. The netX checks a sequence of devices for a bootable image: the boot sequence. The devices in the boot sequence are checked one after the other. The ROM loader checks device by device and starts the first valid image found.

### 3.5.1 Configured by the RDY/RUN pins

The state of the RDY/RUN pins after any type of reset defines the boot sequence.

Figure 15 shows an example circuit to realize the states for the different boot options without the Security EEPROM.

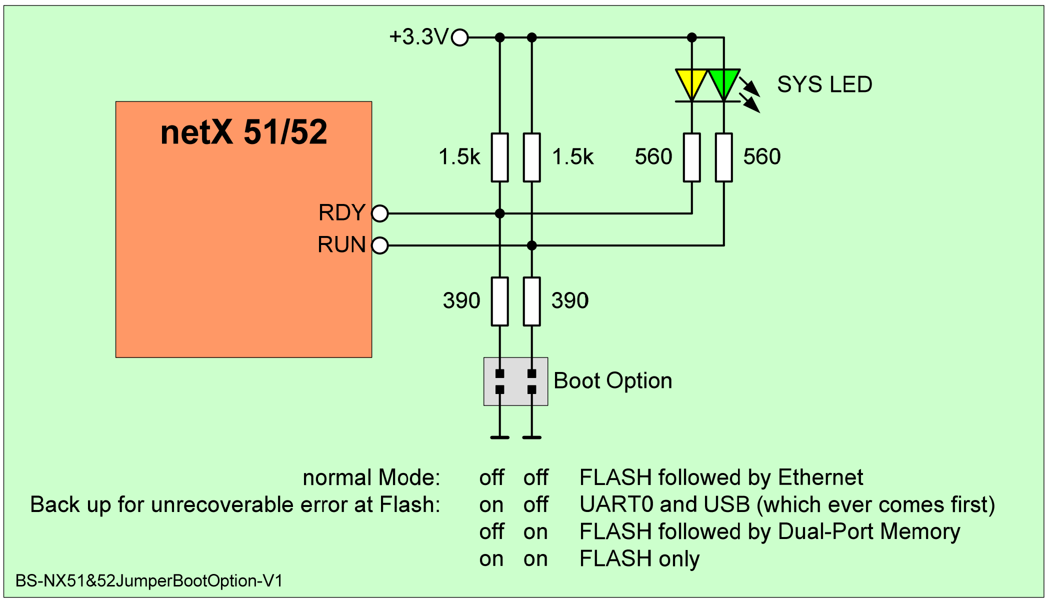

+3.3V
SYS LED
netX 51/52
1.5k
1.5k
560
560
RDY
RUN
390
390
Boot Option
normal Mode:
Back up for unrecoverable error at Flash:
off off FLASH followed by Ethernet
on off UART0 and USB (which ever comes first)
off on FLASH followed by Dual-Port Memory
on on FLASH only
BS-NX51&52JumperBootOption-V1

*Figure 15: netX 51/52 RDY/RUN circuit*

| | **Flash / Ethernet boot mode** | **Serial boot mode** | **Flash / DPM boot mode** | **Flash boot mode** |
|---|---|---|---|---|
| **netX 51** | Par. Flash memory bus | UART0 or USB | Par. Flash memory bus | Par. Flash memory bus |
| | QSPI CS0 Flash | | QSPI CS0 Flash | QSPI CS0 Flash |
| | SPI CS0 EEPROM | | SPI CS0 EEPROM | SPI CS0 EEPROM |
| | SPI CS1 MMC/SD card | | SPI CS1 MMC/SD card | SPI CS1 MMC/SD Card |
| | Ethernet | | Dual-Port Memory | None |
| **netX 52** | Par. Flash host interface | UART0 or USB | Par. Flash host interface | Par. Flash host interface |
| | QSPI CS0 Flash | | QSPI CS0 Flash | QSPI CS0 Flash |
| | SPI CS0 Flash | | SPI CS0 EEPROM | SPI CS0 EEPROM |
| | SPI CS2 MMC/SD card | | SPI CS2 MMC/SD card | SPI CS2 MMC/SD Card |
| | Ethernet | | Dual-Port Memory | None |

*Table 5: Boot sequence and devices for the different boot options*

**Note:** The Flash/Ethernet boot mode is most common. You can program a complete empty Flash or EEPROM via Ethernet with the firmware. Thereafter the device will always start this firmware. The serial boot mode is used as a backup function.

### 3.5.2 Configuration and parameterization by the security EEPROM

The security EEPROM is a further possibility to define the current boot option. It is divided into parts that only Hilscher can read and write by help of a security algorithm mainly used for master or other software licenses. Other parts with defined configuration parameters e.g. MAC address, SDRAM set up parameter or device information like serial number, production date and manufacture code. In addition, there is also a 32-byte part which can be completely defined by the user. The security EEPROM is mandatory in master application using Hilscher master protocol stacks. Since netX 51/52 are primarily designed for slave applications, a security EEPROM is not required. If you have enough space on your PCB, it might be a good design strategy to include a security EEPROM, but do not assemble it as long as it is not necessary. An explanation of the purpose and use of the security EEPROM is available on the Hilscher website in document Dual-Port Memory [2] and AppNote security EEPROM [3].

On netX 51/52, the security EEPROM must not be connected to the I2C interface! The RDY and RUN pins are used instead, see Figure 16.

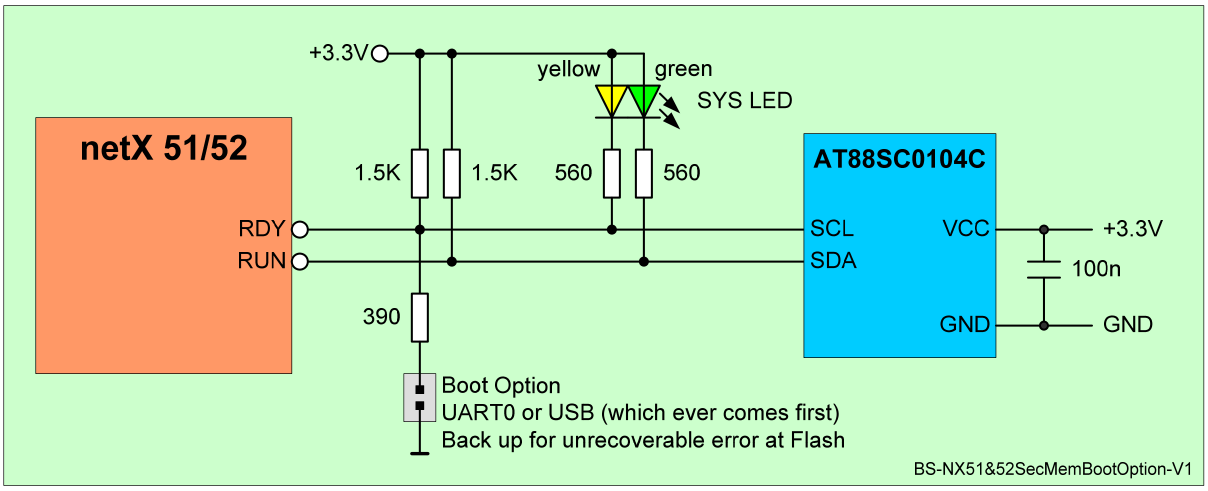

+3.3V
yellow
green
SYS LED
netX 51/52
RDY
RUN
1.5K
1.5K
560
560
390
AT88SC0104C
SCL
SDA
VCC
GND
+3.3V
100n
GND
Boot Option
UART0 or USB (which ever comes first)
Back up for unrecoverable error at Flash
BS-NX51&52SecMemBootOption-V1

*Figure 16: Sample schematic, netX 51/52 security memory and RDY/RUN LED*

You can use type AT88SC0104C only.

**Note:** If you want to use a security EEPROM, order it from Hilscher because it is a special customized version of the AT88SC0104C. Only Hilscher can correctly write the license and other crucial information to the security memory in a netX-readable format.

**Note:** All other I2C components in a netX 51/52 design have to be connected to the I2C interface of netX 51/52. The I2C emulation on the RDY and RUN pins is used by the first stage loader only and is not accessible for users!

Q1: Now that there is only a button for the serial boot mode, how can I enter the DPM boot mode or the extension bus boot mode on my design?

A1: The netX 51/52 can store the boot mode in the user area of the security EEPROM.

Q2: I prefer doing things in hardware. Can’t I still use the jumpers as with the other circuits?

A2: No, you cannot use the jumpers along with the EEPROM. I2C accesses do not allow you to drive signals actively high. Hence, the RDY and RUN pins must always have strong pull-ups when the EEPROM is to be accessed. The netX 51/52 ROM loader will not even look for a security EEPROM unless it detects a high level on the RDY and RUN pins.

Q3: Now that the boot mode is stored in the EEPROM, do I still need the serial boot mode button?

A3: The serial boot mode allows you to (re-)flash the firmware and to set or modify the boot mode stored in the EEPROM. Therefore it is strongly recommended.

Q4: Why are the system LEDs not connected antiparallel as with the other circuits?

A4: The diodes need to be connected differently because the antiparallel circuit has too much impact on the quality of the I2C signals.

Q5: I do not want to use the security memory. Where does the firmware get the MAC address from?

A5: The MAC address could be stored in a flash sector called device label, or set up through an rcX packet.

### 3.5.3 System RDY/RUN LED

For displaying system status, a system LED (dual LED or two single LEDs) is defined:

| LED | Color | Description |
|---|---|---|
| RDY | Yellow | netX with operating system is running |
| RUN | Green | User application is running without errors |

*Table 6: RDY / RUN LED status*

Designers could use LEDs with other colors, but we recommend using the Hilscher definition. Especially when interpreting blink codes for troubleshooting, it is helpful if customer and support see the same colors.

**Note:** In netX schematics, the RDY and RUN pins usually have a negating circle. However, the polarity of these pins (when used as outputs) depends on a register setting. Apart from the two bits that enable the output driver (pin configured as output) and set the level of the pin, there is a third bit for each pin that determines the polarity (active high or active low). Thus, it actually depends on these polarity bits, if the (output-) pins are active low or active high!

These polarity bits do not affect the pins when used as inputs (RDY and RUN inputs are never inverted).

The RDY and RUN pins are always shown as active low because this level will switch on the corresponding LED according the given schematic.

### 3.5.4 Host interface mode during boot up

A special case occurs if the boot sequence contains devices connected to the host interface. The host interface offers a variety of functions with just a few pins. Each pin has several functions. To prevent physical damage to the netX, the host interface should only use the functions that are really connected. Using other functions with the connection might cause permanent damage.

Example 1: A board has a DPM interface connected to the host interface pins. The host interface should be used as a DPM, but not as a memory interface.

Example 2: A board has a parallel flash connected to the host interface pins. The host interface should be used as a memory interface, but not as a DPM.

The netX offers both, DPM and parallel Flash, as boot devices. A way is required to determine which functions of the host interface can be used.

At this point the strapping options become important. They configure what is really connected to the host interface. When the netX executes the boot sequence for DPM mode, it first consults the strapping options. If, and only if, the device matches the settings of the strapping options, the netX looks for a bootable image. If the strapping options do not match, the device will be ignored.

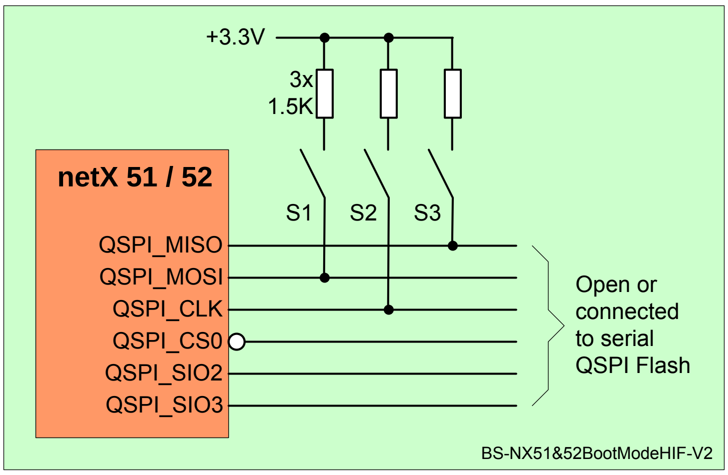


*Figure 17: How to configure the host interface mode during boot up*

| SPI0_MOSI | SPI0_CLK | SPI0_MISO | Device at host interface which netX boots from |
|---|---|---|---|
| none | none | none | Ignore host interface during boot. |
| none | none | pull up | 16-bit SDRAM and up to 4 MByte address range |
| pull up | none | pull up | 32-bit SDRAM and up to 4  MByte address range |
| none | pull up | none | 16-bit SRAM and up to 4 MByte address range |
| none | pull up | pull up | From host CPU via serial dual-port memory (SPM) |
| pull up | none | none | From host CPU via 8-bit parallel dual-port memory and up to 2 KByte address range. |
| pull up | pull up | pull up | From host CPU via 8-bit parallel dual-port memory and up to 64 KByte address range |
| pull up | pull up | none | Reserved. Do not use it. |

*Table 7: Host interface mode*

## 3.6 External memory

Basically, the netX 51/52 provide three different interfaces to connect external memory. All interfaces are working independently from each other:

| | |
|---|---|
| Serial memory interface | for QSPI Flash, SPI EEPROMs and MMC/SD-cards<br>This is the favorite place to store the program code.<br>QSPI with the Execution-in-Place (XiP) function can also run directly the program code for some low priority background tasks.<br>Do not use SPI EEPROMs for new designs! |
| Parallel memory interface | available only on netX 51 for Flash/SRAM/SDRAM<br>Mostly used with a cheap 8 MB/32-bit SDRAM to enlarge the internal memory with a good overall performance. |
| Host memory interface | for Flash, SRAM, SDRAM<br>It takes a lot of signals from the host interface and you have to check which functions are still available. For details, see chapter Host Interface. |

When connecting memory components to the parallel Flash, SRAM or SDRAM interface, designers should always recalculate the capacitive load of these interface signals. The limitations are:

| | |
|---|---|
| Parallel memory interface | 10 pF for data lines and 15 pF for all other signals<br>As standard you can only connect one memory device. |
| Host memory interface | 30 pF on all signals |

The capacitive load directly influences the signal timing (the higher the load, the longer the signal delay) which has a limited scope with SDRAMs. Since the allowed range of operating conditions (min./max. voltage, min./max. temperature) further influences signal timing, capacity limits need to be defined that ensure a safe operation throughout the whole voltage and temperature range.

If these capacity limits are exceeded, this may, to a certain extent, be compensated by two clock phase parameters of the SDRAM interface. Such “out-of-spec-designs” are imaginable, but require careful evaluation!

**Note:** If you are using our standard firmware, consider that it supports only the memory devices listed in the reference section of this manual. Of course, other memories can also be used, but not without prior evaluation and tests. To clarify details, contact our netX Support!

### 3.6.1 Serial memory interface

The netX 51/52 has a dedicated SPI/QSPI controller to be used for program code.

An additional SPI controller can be mapped to the MMIOs that are supposed to be used with external peripheral devices.

SPI/QSPI uses high clock rates. This means: Keep the traces short.

#### 3.6.1.1 SPI Flashs

SPI Flashs have already been on the market for a long time. We, thus, strongly recommend not to use these components for new designs due to the following disadvantages:

| | |
|---|---|
| Longer program load time | since they have only a single wire transfer and the bandwidth is only one fourth of QSPI Flash |
| Non Execution-in-Place | i.e. they cannot enlarge the size of the executable code, especially in netX 52 applications |

**Important:**

If you are using standard loadable firmware (LFW) without SDRAM for netX 52 designs, you cannot use SPI Flashs! See next section about *QSPI Flash* on page 28.

If you want to use other Flash memories, please contact Hilscher before.

While an SPI Flash can be connected to three different chip select signals, it must be connected to chip select 0 (SPI0_CS0n), if the design is to be able to boot from this SPI Flash.

**Note:** SPI0_CS2n is available on MMIO pins.

A standard component used by Hilscher was the AT45DB321 from ATMEL providing a capacity of 4 MB. For connection, see Figure 18:

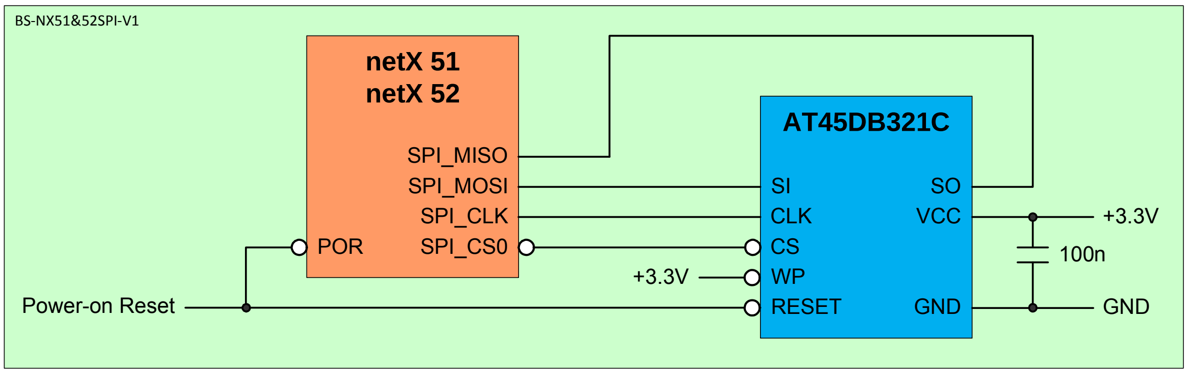


*Figure 18: netX 51/52 SPI Flash*

**Note:** ATMEL does not continue with their serial EPROM business which was taken over from Adesto. The new component number is AT45DB321E.

### 3.6.1.2 QSPI Flash

Instead of a standard SPI Flash, a QSPI Flash may be used on netX 51/52 designs. It allows you to reduce the time required for loading large firmware images and to reduce boot up time. These Flashes can be switched to a second mode that allows the simultaneous serial transmission of 4 bit.

This feature has been implemented for designs that run PROFINET and are required to provide the so-called “Fast Startup” (FSU) feature. It is available on netX 51 and netX 52 for read and write.

The currently available solution allows a startup time for PROFINET slave firmware of less than 500 ms.

The supported Flash type is a W25Q32V from Winbond Electronics.

**Important:**

Since netX 52 uses XiP to execute loadable firmware, we highly recommend Windbond W25Q32. If you want to use other Flash memories, please contact Hilscher before.

Table 8 shows the pin connection for netX 51 and netX 52:

| | **Pin assignment** | **SPI_CSn QSPI_CSn** | **SPI_CLK QSPI_CLK** | **SPI_MOSI QSPI_SIO0** | **SPI_MISO QSPI_SIO1** | **QSPI_SIO2** | **QSPI_SIO3** |
|---|---|---|---|---|---|---|---|
| netX 51 | Standard | U14 | V15 | V16 | U15 | P12 | R13 |
| netX 52 | Standard | U15 | V16 | U16 | T15 | T14 | V15 |

*Table 8: QSPI pin assignment netX 51/52*

Figure 19 shows how to connect this Flash to netX 51 and netX 52:

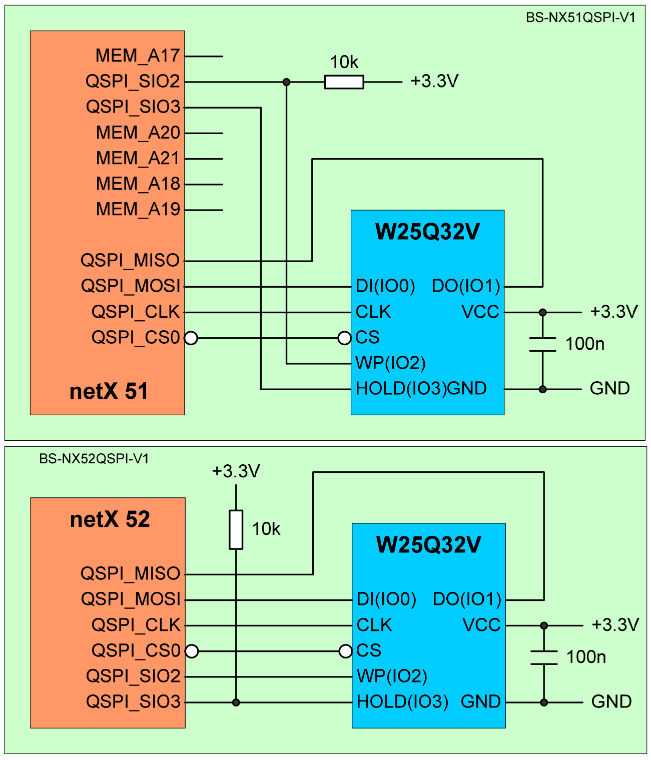
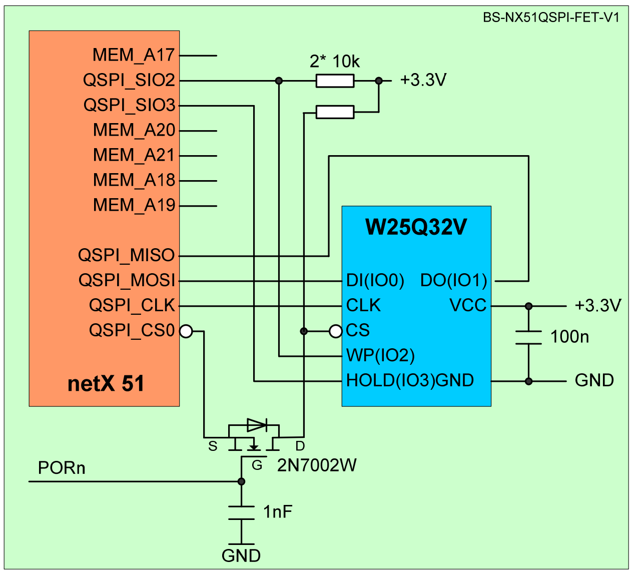
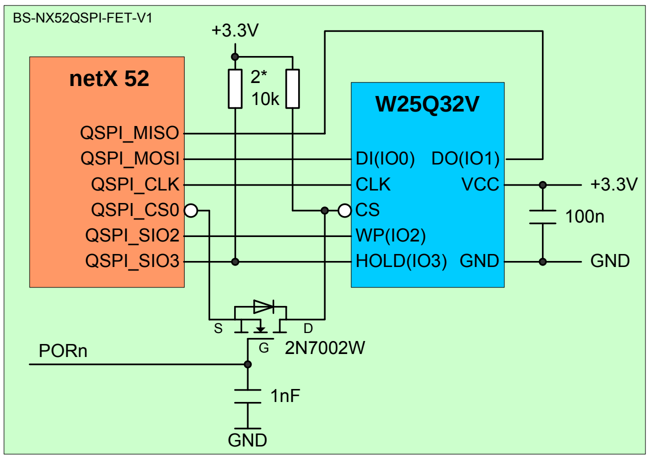


*Figure 19: netX 51/52 with QSPI Flash*

Continued on the next page.





**Note:** The FET is used to meet the datasheet requirement of this QSPI device: The CS signal should follow the power supply. According to experience and examinations the FET is left out in the Hilscher standard designs, this is not 100% compliant with the datasheet, but has not resulted in problems for many years.

### 3.6.1.3 MMC/SD card

Instead of an SPI Flash a MMC/SD card can be connected via SPI to the netX, which even allows you to boot a firmware image stored on such a card. To detect insertion or removal of the MMC/SD card during operation, an insertion signal has been defined which has to be pulled high if an MMC/SD card is in the socket. If the MMC/SD card is connected to the SPI bus in addition to a Flash memory, we recommend decoupling the MMC/SD card with switchable bus drivers.

| **Function** | **netX 51 pin name and number** | | **netX 52 pin name and number** | |
|---|---|---|---|---|
| MMC/SD card insert | GPIO15 at MMIO16 | U2 | GPIO08 at MMIO08 | N2 |
| SPI Chip Select | SPI0_CS1n | T14 | SPI0_CS2n at MMIO09 | P2 |

*Table 9: MMC/SD card insertion and CSn signal*

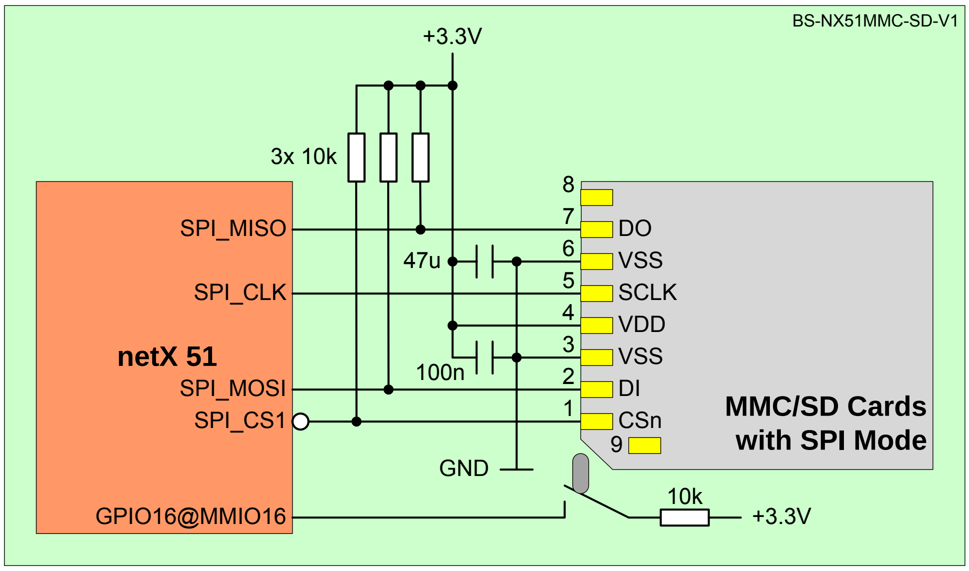


*Figure 20: netX 51 MMC/SD card*

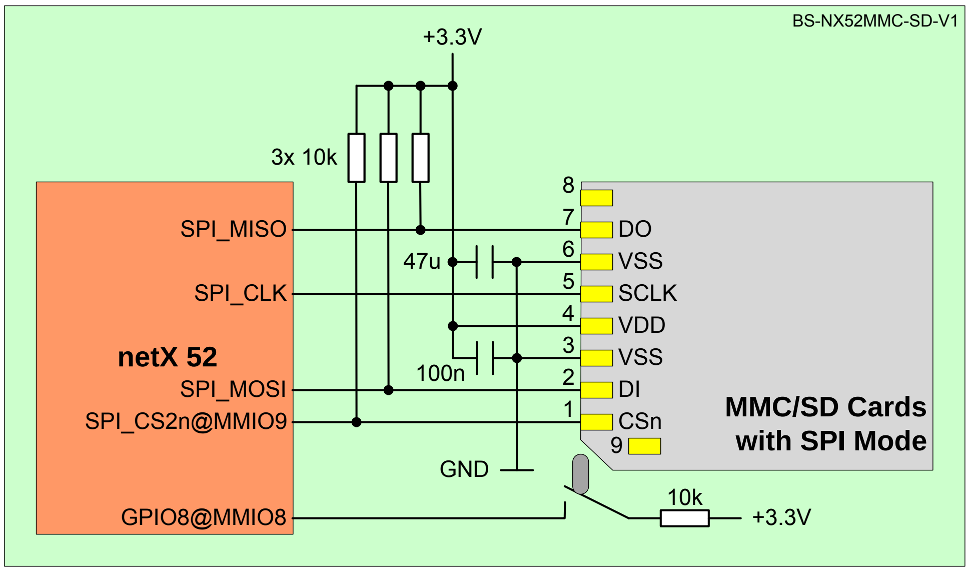


*Figure 21: netX 52 MMC/SD card*

**Note:** The card selection signals use the internal pull-down resistor at MMIO to have a defined low-level if there is no card. The pull-up resistors at DO, DI, CSn and the 47u capacitor are necessary according the SD card specification.

If the MMC/SD card is connected to the SPI bus in addition to a Flash memory, we recommend decoupling the MMC/SD card with a switchable bus driver.

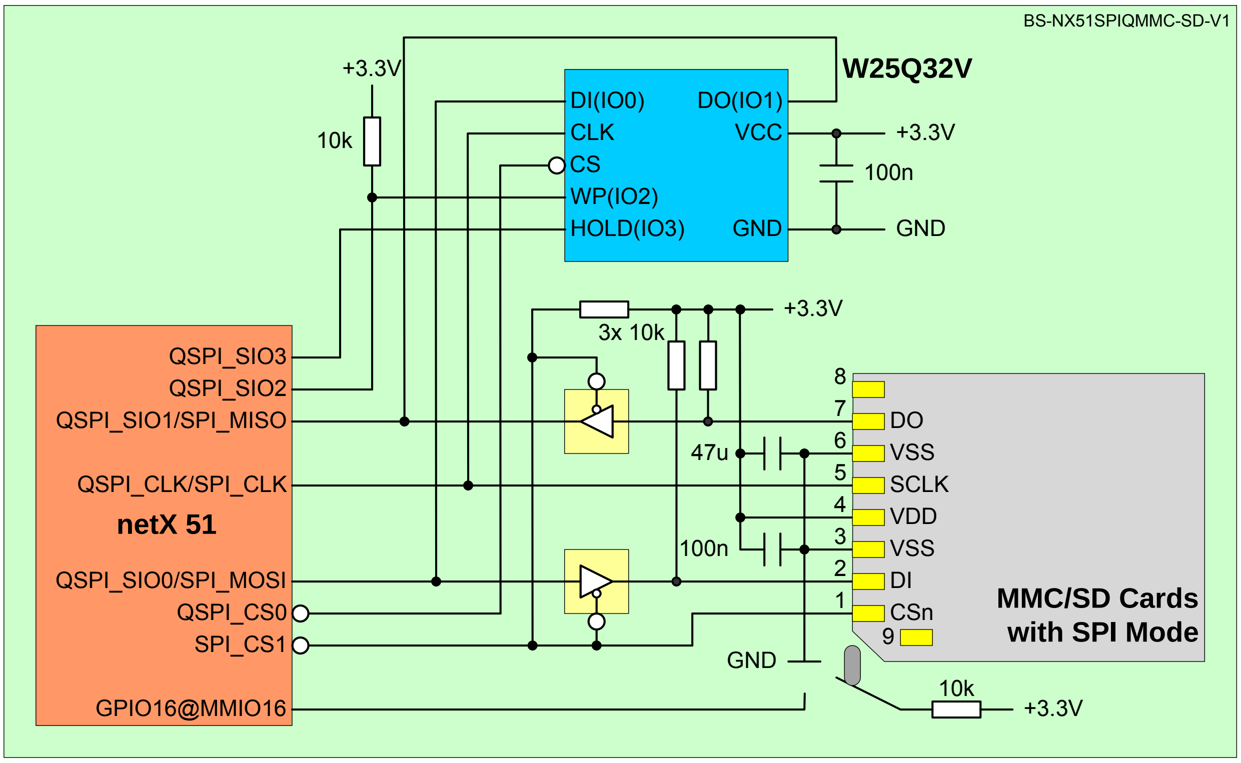


*Figure 22: MMC/SD card and Flash memory on SPI bus*

**Note**: Drivers are required for parallel operation.

### 3.6.2 Parallel memory interface

netX 51/52 contain two completely independent external memory interface controllers. The first one is connected to the memory interface of netX 51, the second one to the host interface of netX 51/52. However, the SDRAM controller of the external memory interface differs from that of the host interface: The external memory interface SDRAM controller is a high-performance multi-channel and multi-cache controller. The SDRAM controller of the host interface is a single-channel SDRAM-light controller containing 16 or 32 data bits.

#### 3.6.2.1 SRAM and Flash

For large firmware images for applications executing code directly from the Flash, the use of a parallel Flash is inevitable. Parallel Flash connected to the netX 51 may be 8-, 16- or 32-bit wide; two 16-bit components may be paired for 32-bit wide access.

**Note:** rcX does not support a Flash file system at a parallel Flash.

For performance reasons, 32-bit components should be used when executing code directly from the Flash although 16-bit wide components are very common.

The netX 51 SRAM/Flash memory controller provides three different chip-select signals (MEM_CS[2:0]) allowing to select three different memory components or pairs of components (two paired 16-bit Flashes use a common chip select signal), each with its own set of parameters (timing and bus width).

If the design is to boot from a parallel Flash, chip-select 0 must be used for selecting this Flash.

The memory controller is designed to never “waste” any address lines regardless of the bus width setting. Hence, in 8-bit mode, address line A0 is used for low and high byte selection, while in 16-bit mode A0 selects low and high word and in 32-bit mode, A0 is simply the LSB of a DWORD address.

For that reason, consult the data sheet of the desired Flash component to determine the correct way of hooking up the address lines of the Flash.

Many (16-bit) Flash components (e.g. TE28F128J...) use address line A0 for low/high byte selection when operating the component in 8-bit mode. They do not use A0 at all when used in 16-bit mode. Hence, with such components A0 of the Flash must be grounded (to prevent floating), while A0 of the netX is connected to A1 of the Flash, A1 to A2, A2 to A3, etc. The following schematics show examples:

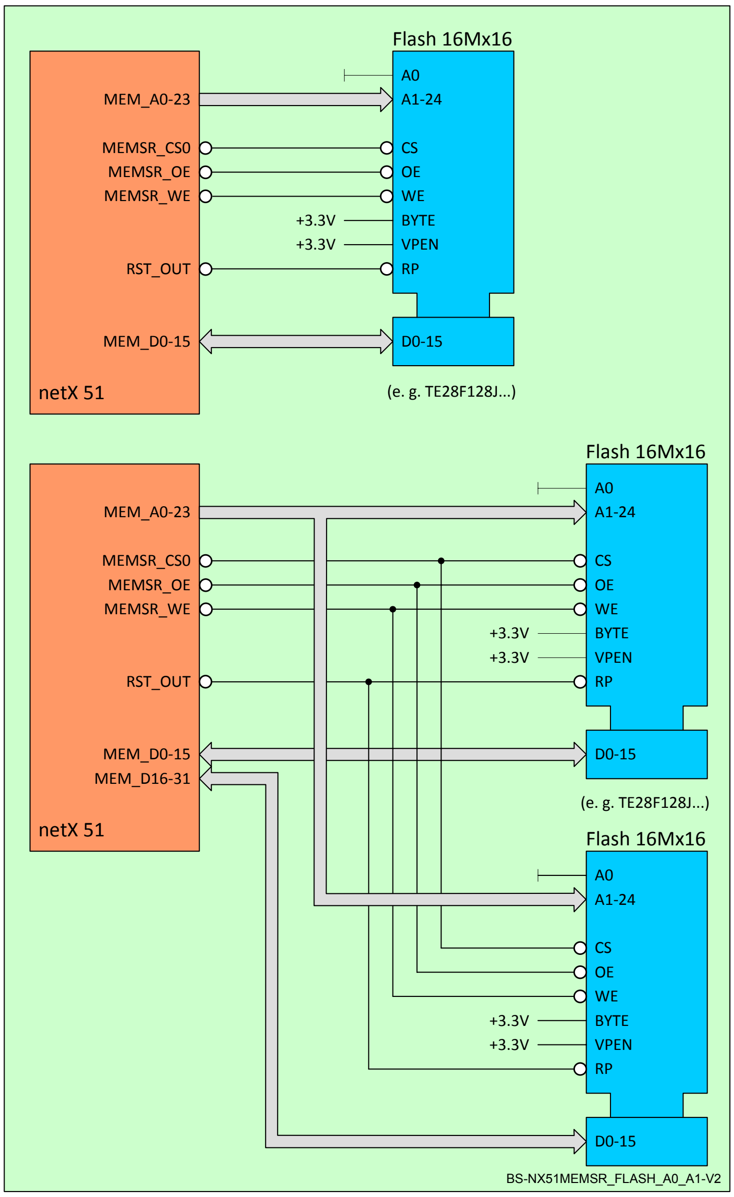


*Figure 23: netX 51 Flash - Address line A0 for low/high byte selection*

Other Flash components (e.g. S29GL256P...) always use A0 as the LSB of a word (16 bit) address. Hence, the address lines of such components must be connected straight forward as shown in Figure 24:

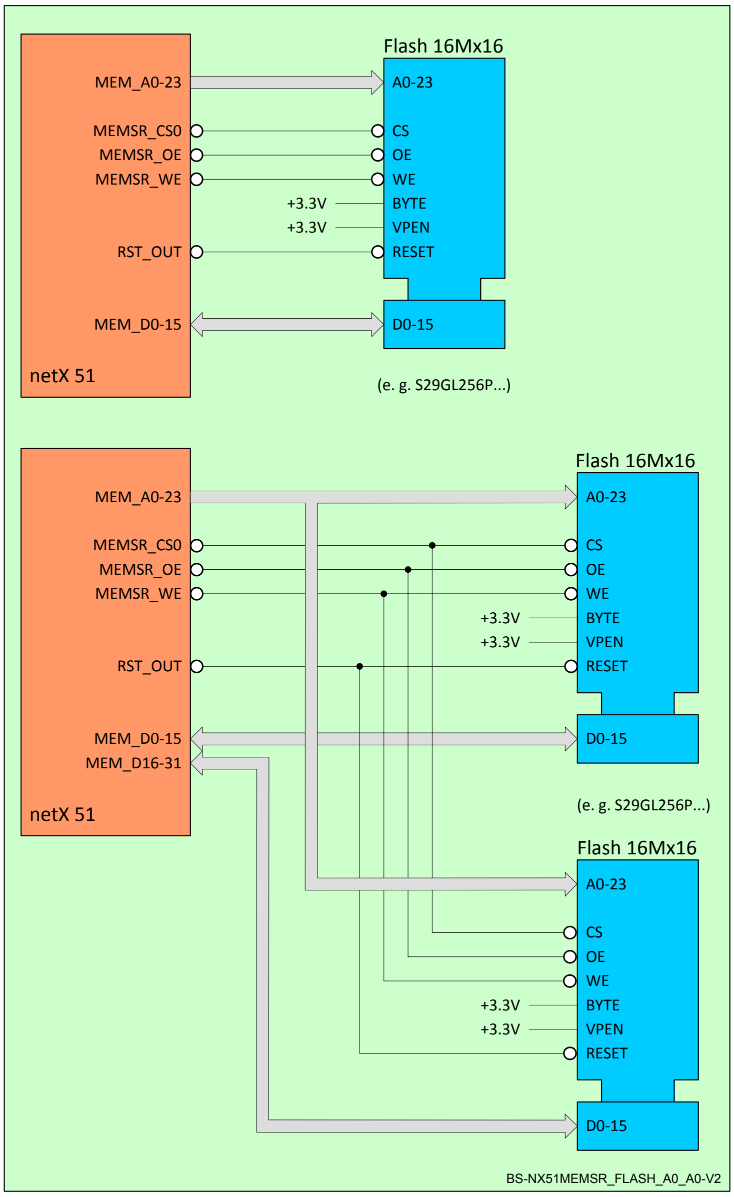


*Figure 24: netX 51 Flash - A0 as the LSB of a word address*

Q1: Is it possible to access the parallel Flash connected to the netX by another processor (e.g. host processor) while the netX does not use the Flash or is held in reset?

A1: This will work only with additional components. Since memory interfaces are usually not designed for multi-master access, the netX always drives all memory interface signals, except the data lines, even while it is held in reset state. Hence, accessing any memory components by another processor is possible only if the memory components can be isolated from the netX memory interface by appropriate bus switches.

### 3.6.3 SDRAM

The netX 51 has a separate memory bus to support an SDRAM.

netX 52 needs no external SDRAM, but an SDRAM can be connected to the host interface. In this case the DPM will be limited to a Serial Port Memory.

SDRAM components connected to netX may be 16-bit or 32-bit wide. Two 16-bit components may be paired to allow a 32-bit wide access. Using two 8-bit components (paired for 16-bit) or four 8-bit components (32-bit) is also possible.

For using SDRAM, designs of 32-bit width are generally recommended to make full use of the memory controller performance. The use of one 32-bit wide component instead of two 16-bit (or four 8-bit) components is further recommended due to an easier PCB design and reduced load capacity (two 16-bit components usually add twice the load to the address and control signals as a comparable 32-bit component).

Q1: In the meantime DDR-3 RAM or higher is state of the art. Why are the netX chips only equipped with an outdated SDRAM interface?

A1: Well, SDRAM isn't really outdated. DDR RAM technology was invented for the short-lived PC market on which it is commonly accepted that components have extremely short life cycles, a limited operating condition range, and substantial power consumption. Since DDR RAMs work with internal PLLs they cannot be used on older (slower) memory interfaces. DDR RAM technology is not suitable for the embedded / industrial market where customers usually look for an availability of several years. Further, even powerful embedded processor technology like ARM cannot necessarily compete with common PC processors in terms of processing power. Thus, it would make little sense to connect such processors to DDR RAMs anyway.

Connecting SDRAM to the netX is pretty straight forward, besides address lines A16 and A17, which are used for the bank select signals BA0 and BA1.

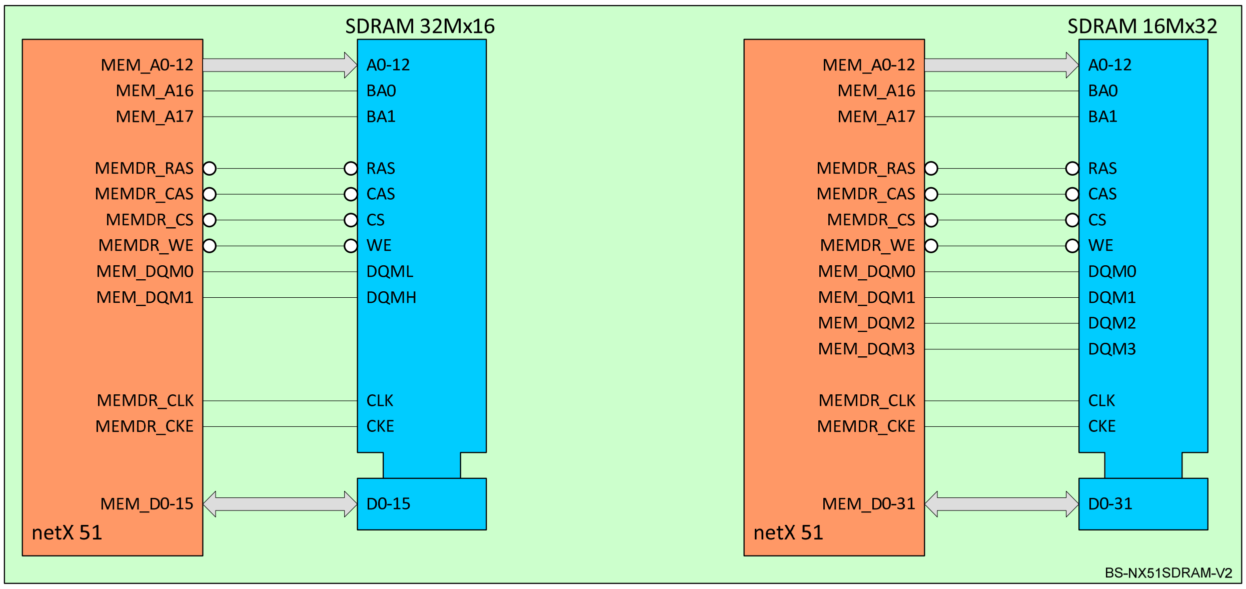


*Figure 25: netX 51 SDRAM 1 * 16 bit, 1 * 32 bit*

## 3.7 Host interface

The netX 51/52 host interface (HIF) has 58 signals in total and provides various connections:

- 8/16/32-bit dual-port memory (DPM): section *Dual-port memory* on page 37
- Dual-port memory via SPI/QSPI (SPM): section *Serial port memory (SPI/QSPI access to DPM)* on page 42,
- 8/16/32-bit extension bus (EXT): section *Extension bus* on page 44,
- Media independent interface (MII): section *MII port 2 acts as a PHY or MAC interface* on page 48
- 16/32-bit SDRAM
- 8x MMIO for GPIO, UART, CAN, I2C, SPI

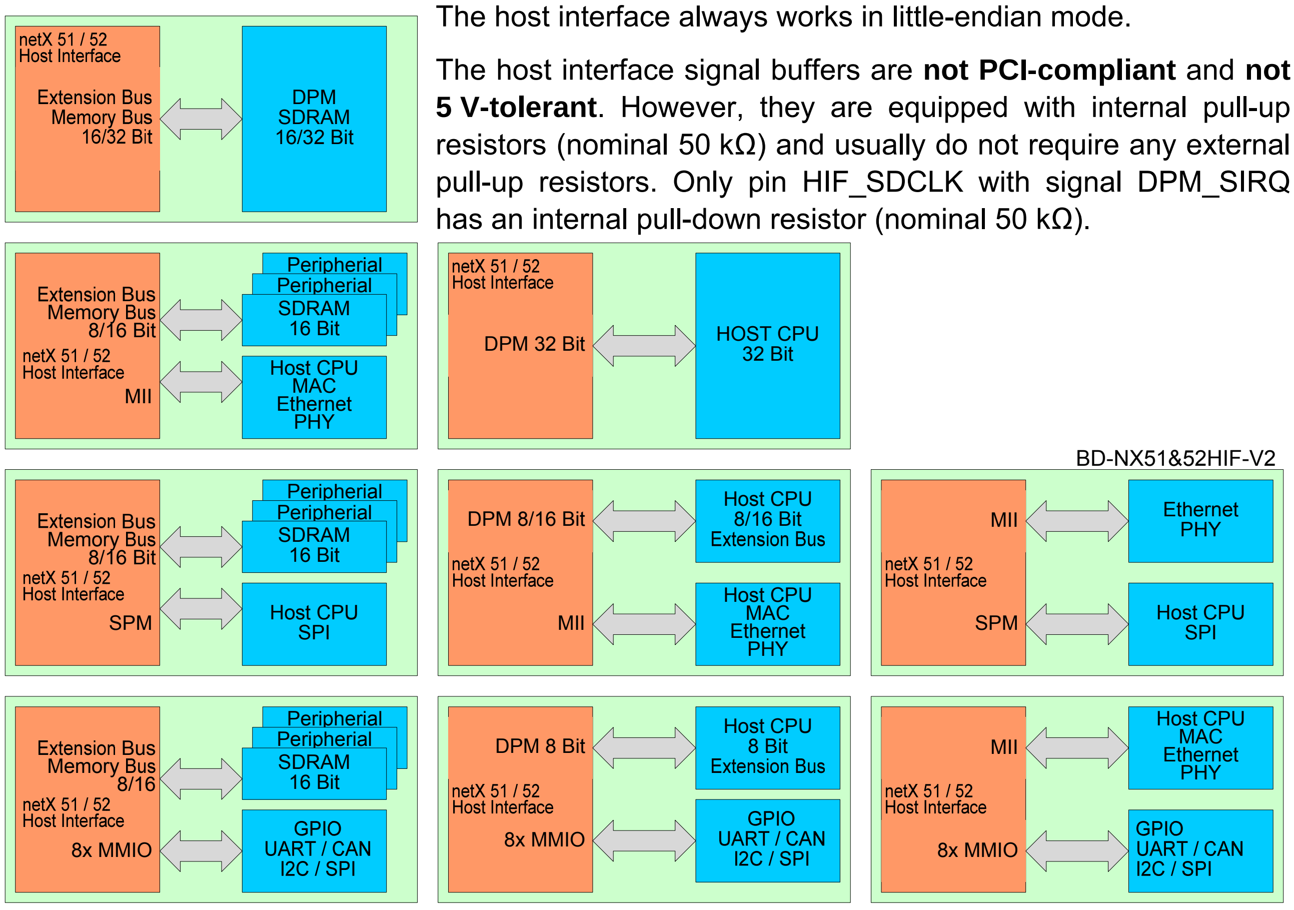
Table 10 shows the signals which can be combined on the host interface.

| | | **Dual-port memory** | | | **Extension bus** | | | **SDRAM** | | **MII** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 8 Bit | 16 Bit | 32 Bit | 8 Bit | 16 Bit | 32 Bit | 16 Bit | 32 Bit | |
| **MII** | | ✓ | ✓ | - | ✓ | ✓ | - | ✓ | - | - |
| **SPM** | | - | - | - | ✓ | - | - | ✓ | - | ✓ |
| **8x MMIO** | | ✓ | - | - | ✓ | ✓ | - | ✓ | - | ✓ |
| **SDRAM** | 16 Bit | - | - | - | ✓ | ✓ | ✓ | - | - | ✓ |
| | 32 Bit | - | - | - | ✓ | ✓ | ✓ | - | - | - |

*Table 10: Signal combination on host interface*

The host interface always works in little-endian mode.

The host interface signal buffers are **not PCI-compliant** and **not 5 V-tolerant**. However, they are equipped with internal pull-up resistors (nominal 50 kΩ) and usually do not require any external pull-up resistors. Only pin HIF_SDCLK with signal DPM_SIRQ has an internal pull-down resistor (nominal 50 kΩ).



netX 51 / 52 Host Interface
Extension Bus Memory Bus 16/32 Bit
DPM SDRAM 16/32 Bit
Extension Bus Memory Bus 8/16 Bit
netX 51 / 52 Host Interface
MII
Peripherial
Peripherial
SDRAM 16 Bit
Host CPU MAC Ethernet PHY
netX 51 / 52 Host Interface
DPM 32 Bit
HOST CPU 32 Bit
BD-NX51&52HIF-V2
Extension Bus Memory Bus 8/16 Bit
netX 51 / 52 Host Interface
SPM
Peripherial
Peripherial
SDRAM 16 Bit
Host CPU SPI
DPM 8/16 Bit
netX 51 / 52 Host Interface
MII
Host CPU 8/16 Bit Extension Bus
Host CPU MAC Ethernet PHY
MII
netX 51 / 52 Host Interface
SPM
Ethernet PHY
Host CPU SPI
Extension Bus Memory Bus 8/16
netX 51 / 52 Host Interface
8x MMIO
Peripherial
Peripherial
SDRAM 16 Bit
GPIO UART / CAN I2C / SPI
DPM 8 Bit
netX 51 / 52 Host Interface
8x MMIO
Host CPU 8 Bit Extension Bus
GPIO UART / CAN I2C / SPI
MII
netX 51 / 52 Host Interface
8x MMIO
Host CPU MAC Ethernet PHY
GPIO UART / CAN I2C / SPI

*Figure 26: Host interface connection possibilities*

## 3.7.1 Dual-port memory

### 3.7.1.1 8/16/32-bit data width and dual-port memory size

In DPM mode, netX 51/52 support 8, 16 or 32-bit data width. In general, the min. DPM size is 2 KB. With 8 or 16-bit data width the max. size is 1 MB, with 32-bit data width the max. DPM size is 256 KB.

Before you start your design, observe the following information:

#### 8/16-bit mode

The second stage boot loader offers a detection function to select parallel dual-port memory mode (8 or 16 bit) or serial port mode. Therefore, the second stage boot loader evaluates the signal at pin DPM_DIRQ and DPM_SIRQ during startup. Table 11 lists the selected mode depending on the signal state.

| DPM_DIRQ signal during startup | DPM_SIRQ signal during startup | Mode |
|---|---|---|
| 1 (high) | 1 (high) | Parallel dual-port memory mode: 8-bit mode |
| 1 (high) | 0 (low) | Parallel dual-port memory mode: 16-bit mode |
| 0 (low) | X | Serial port mode |

*Table 11: DPM_SIRQ signal during startup*

netX 51/52 has an internal pull-down resistor (nominal 50 kΩ) at DPM_SIRQ and an internal pull-up resistor (nominal 50 kΩ) at DPM_DIRQ. A pull-up resistor $R_{pu}$ with a recommended value of 2.2 kΩ is required to generate a high signal if the DPM_SIRQ is not connected to the host CPU.

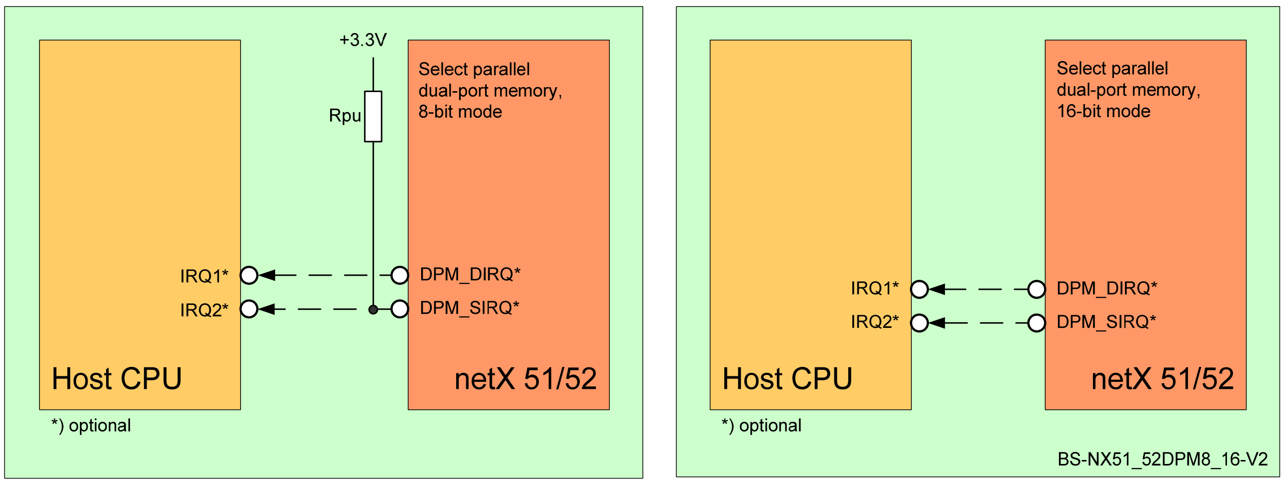


*Figure 27: Using the second stage boot loader to select parallel dual-port memory with 8 or 16-bit mode*

**Note:** If DPM_SIRQ or DPM_DIRQ is connected to the host CPU, pay attention to the internal resistor of the pin of the host CPU for dimensioning a pull-up or pull-down resistor to generate a proper signal according to Table 11.

#### 32-bit mode

You can use netX in 32-bit dual-port memory mode. This requires you to configure the netX for this mode. If you use the second stage boot loader, you have to set parameters in the second stage boot loader to activate the 32-bit mode using the Tag List Editor tool.

### Dual-port memory size

The default size of the DPM is 64 KB. If loadable firmware is used on the netX a size of 64 KB is the recommended size for a design and the host can access all communication channels located in the DPM. A dual-port memory size of 16 KB (address lines DPM_A0 to DPM_A13) is possible with loadable firmware, but the host can access one communication channel only! Address line DPM_14 and DPM_A15 have to be connected with a pull-down resistor of 1 k to GND to access the lowest 16 KB of the DPM. Be aware, if the host can access 16 KB of the DPM only, you will lose functionality and future extensions.

The address lines DPM_A0 to DPM_A15 are located at the same netX pins for all three modes 8, 16 and 32-bit mode and address a DPM size of up to 64 KB.

### Dual-port memory size of 128 KB and more

If you want to create a design with a DPM size of 128 KB (which requires DPM_A16) or a size of 256 KB (which requires DPM_A16 and DPM_A17), make sure to use different pins for 8/16-bit mode and for 32-bit mode.

If you want to create a design with a DPM size of 512 KB (DPM_A0 to DPM_A18) or a size of 1 MB (DPM_A0 to DPM_A19), remember that you can use 8 or 16-bit mode only.

| **Dual-port memory** | | **8/16-bit mode** | | **32-bit mode** | |
|---|---|---|---|---|---|
| **Size** | **Address** | **netX 51 pin** | **netX 52 pin** | **netX 51 pin** | **netX 52 pin** |
| 128 KB | DPM_A16 | B13 | C12 | E14 | A17 |
| 256 KB | DPM_A17 | A14 | C11 | D14 | A18 |
| 512 KB | DPM_A18 | A13 | A11 | - | |
| 1 MB | DPM_A19 | C12 | C10 | - | |

*Table 12: netX pin for a DPM size of 128 KB and more*

#### 3.7.1.2 Control lines

In DPM mode, the host interface can be controlled by separate read (RDn) and write (WRn, WRLn, WRHn) signals ("Intel mode") or by a combined RD/WRn signal indicating the direction of the access and byte strobe signals ("Motorola mode"). In "Intel mode", either a single write signal can be used (WRLn) or two write signals (WRLn, WRHn) for writing to the low byte (WRLn) and high byte (WRHn) separately.

In case of a 16-bit Host systems with word access, the address line A0 should be connected to GND. The following table shows the decoding logic for byte and word access.

| **BHEn** | **A0** | **Function** |
|---|---|---|
| 0 | 0 | word access |
| 0 | 1 | access high byte |
| 1 | 0 | access low byte |
| 1 | 1 | no access |

*Table 13: Function table of 16-bit decode logic*

### 3.7.1.3 Non-multiplexed mode

The following schematics show some examples for common setups in non-multiplexed mode:

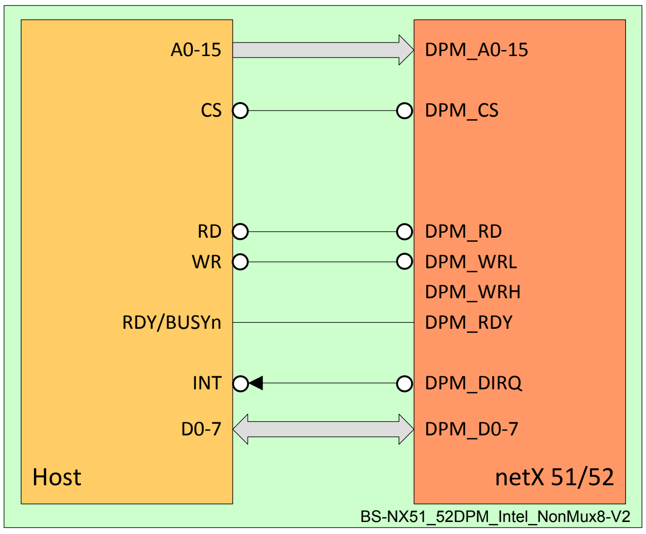


*Figure 28: Intel ™ interface, 8-bit, non-multiplexed*

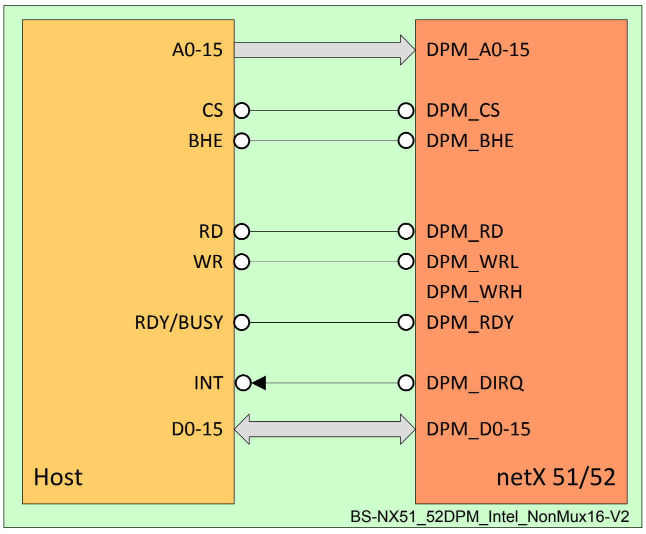


*Figure 29: Intel ™ interface, 16-bit, non-multiplexed*

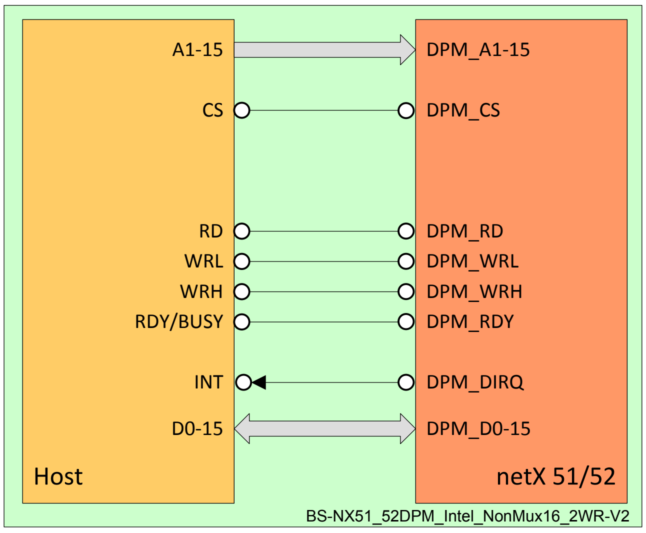


*Figure 30: Intel ™ interface, 16-bit, non-multiplexed, 2 write signals (low byte, high byte)*

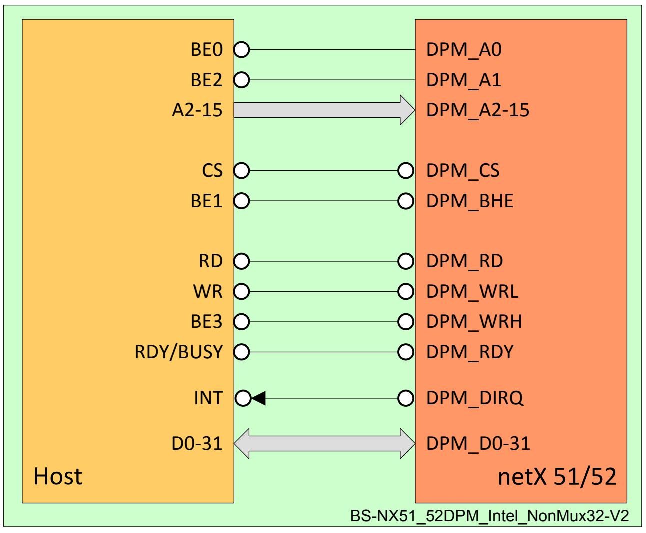


*Figure 31: Intel ™ interface, 32-bit, non-multiplexed*

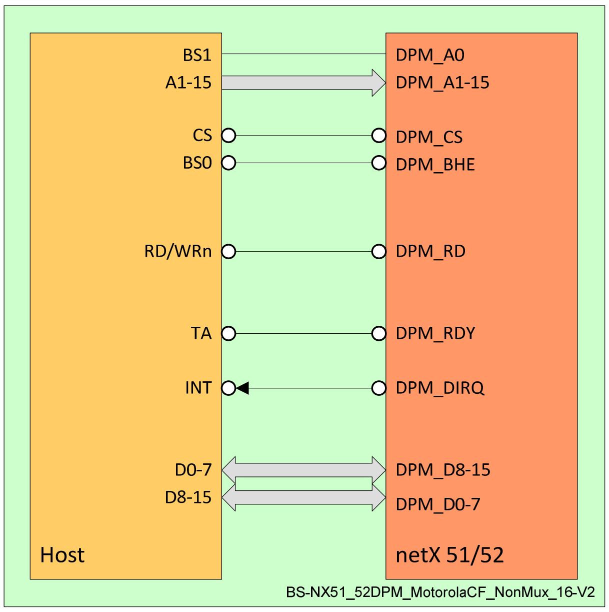


*Figure 32: Motorola $^{TM}$ ColdFire, 16-bit, non-multiplexed*

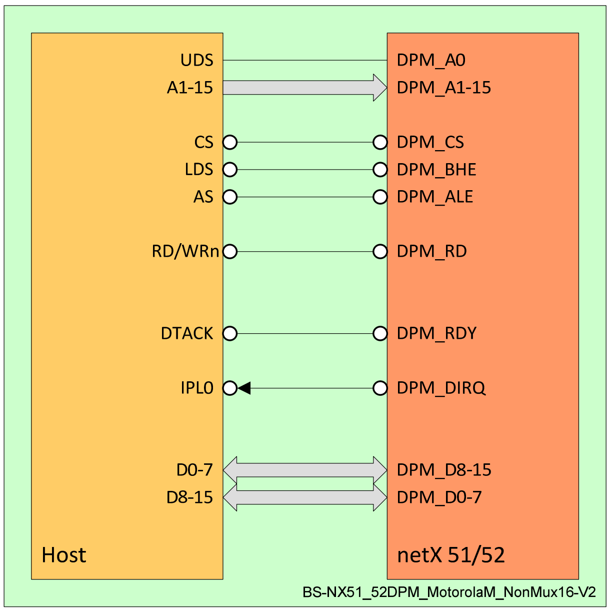


*Figure 33: Motorola $^{TM}$ M68000, 16-bit, non-multiplexed*

### 3.7.1.4 Multiplexed mode

The netX host interface can also be operated in multiplexed mode in which the data lines are alternatingly used for data and the lower address signals. The following schematics show some examples for common setups in multiplexed mode:

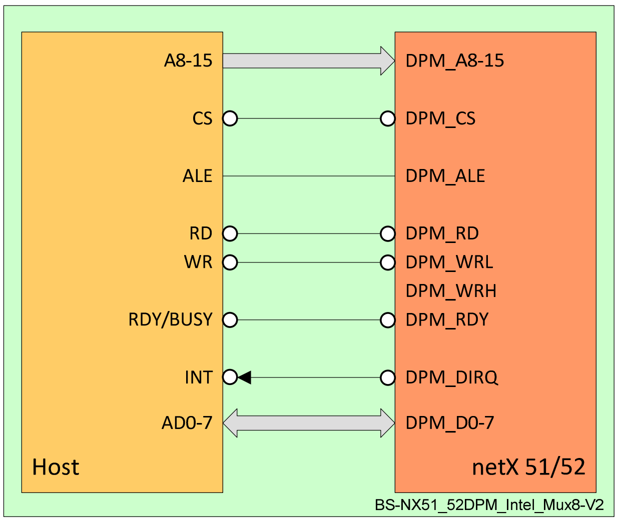


*Figure 34: Intel $^{TM}$ interface, 8 bit, multiplexed*

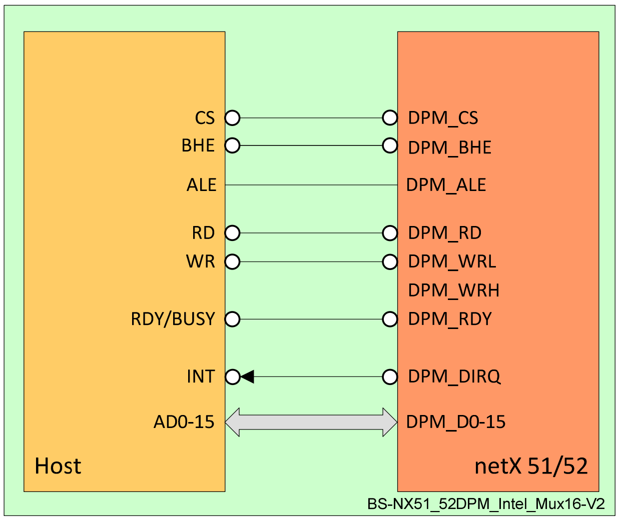


*Figure 35: Intel $^{TM}$ interface, 16 bit, multiplexed*

#### 3.7.1.5 Ready/Busy signal

In the Dual-Port memory mode of the host interface the netX provides an internal memory area which can be read or written from the internal ARM CPU and from an external host CPU. As a real DPM, the netX uses a READY/BUSYn signal to resolve access conflicts if both CPU try to access the memory area simultaneously.

To make sure that the host processor will read valid data or the netX successfully accepts the written data from the host, while keeping access cycles as short as possible, it is **absolutely necessary** for the host processor to support the Ready/Busy signal in connection with netX.

To allow interfaces without glue logic, the netX Ready/Busy signal supports two different modes:

| | |
|---|---|
| Ready mode: | The active DPM_RDY signal indicates that there is no access conflict. |
| Busy mode: | The active DPM_BUSYn signal indicates that the netX is still busy and the cycle has to be delayed. |

The Ready/Busy signal can be configured as an active high or active low signal, as push/pull or open drain, open source with sustained tri-state option (signal edge is actively driven).

**Note:** Our standard firmware uses the Busy mode with an active low signal. All schematics of this manual show the Ready/Busy signal as DPM_BUSYn.
In the netX 51/52 Technical Reference Guide this signal often appears in the timing diagrams as RDY (Ready mode / active high).

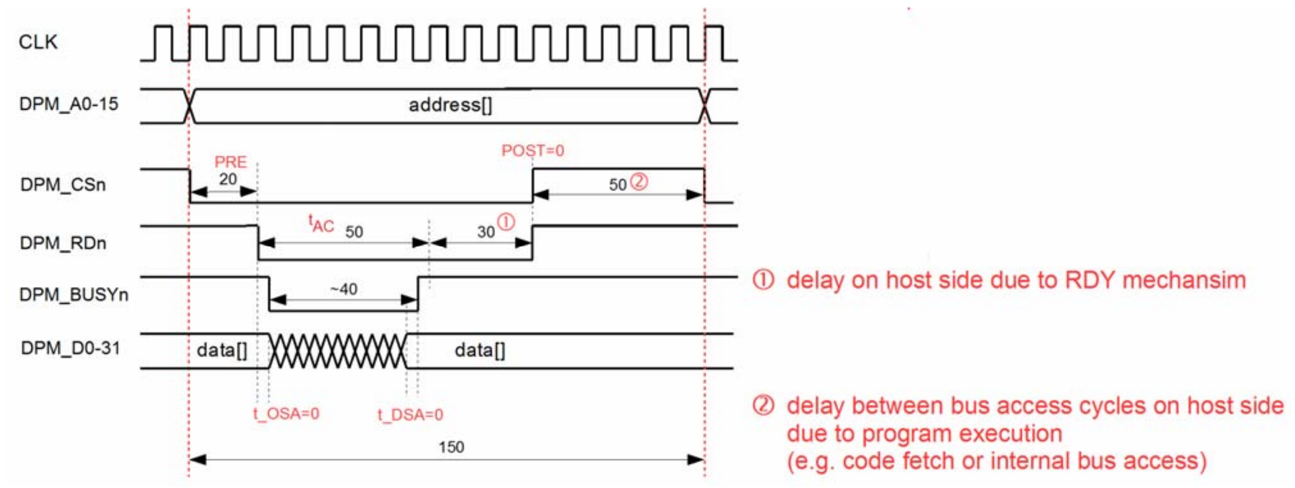

CLK
DPM_A0-15
address[]
DPM_CSn
PRE
20
POST=0
50 ②
DPM_RDn
tAC 50
30 ①
DPM_BUSYn
~40
DPM_D0-31
data[]
data[]
t_OSA=0
t_DSA=0
150
① delay on host side due to RDY mechansim
② delay between bus access cycles on host side due to program execution (e.g. code fetch or internal bus access)

*Figure 36: DPM read access by using DPM_BUSYn signal*

If your host CPU does not support a Ready/Busy signal, you can access the DPM without it if you observe the guaranteed min. access time $t_{AC}$ of 55 ns. In that case you have to set the host CPU pulse width of DPM_RDn on 60 ns plus the required host CPU data set up time. This results in a short cycle time of about 100 ns.

### 3.7.2 Serial port memory (SPI/QSPI access to DPM)

The netX 51/52 support the connection between host CPU and netX DPM via SPI and QSPI where the netX is the slave. Table 14 shows the SPI/QSPI serial DPM signals:

| **Pin** | | **SPI signal** | **QSPI signal** | **Comment** |
|---|---|---|---|---|
| **netX 51** | **netX 52** | | | |
| G16 | D18 | SPM_CLK | SPM_CLK | SPI or QSPI: serial clock input |
| H16 | F17 | SPM_CSn | SPM_CSn | SPI or QSPI: chip-select input |
| H15 | G16 | SPM_MOSI | SPM_SIO0 | SPI: master out slave in data input<br>QSPI: serial data bit 0 |
| J16 | G18 | SPM_MISO | SPM_SIO1 | SPI: master in slave out data output<br>QSPI: serial data bit 0 |
| D18 | C17 | - | SPM_SIO2 | QSPI only: serial data bit 2 |
| C18 | B17 | - | SPM_SIO3 | QSPI only: serial data bit 3 |
| G18 | C18 | SPM_DIRQn | SPM_DIRQn | Optional DIRQ for host (e.g. for data IRQ) |
| G15 | B18 | SPM_SIRQn | SPM_SIRQn | Optional SIRQ for host (e.g. for service IRQ) |

*Table 14: SPI/QSPI to DPM (serial DPM) pin assignment*

The host CPU sends read and write commands via the SPI/QSPI connection to the netX controller. A state machine within the netX translates these commands into parallel read and write access to the DPM without interfering with the ARM CPU and sends the answer back to the host CPU.

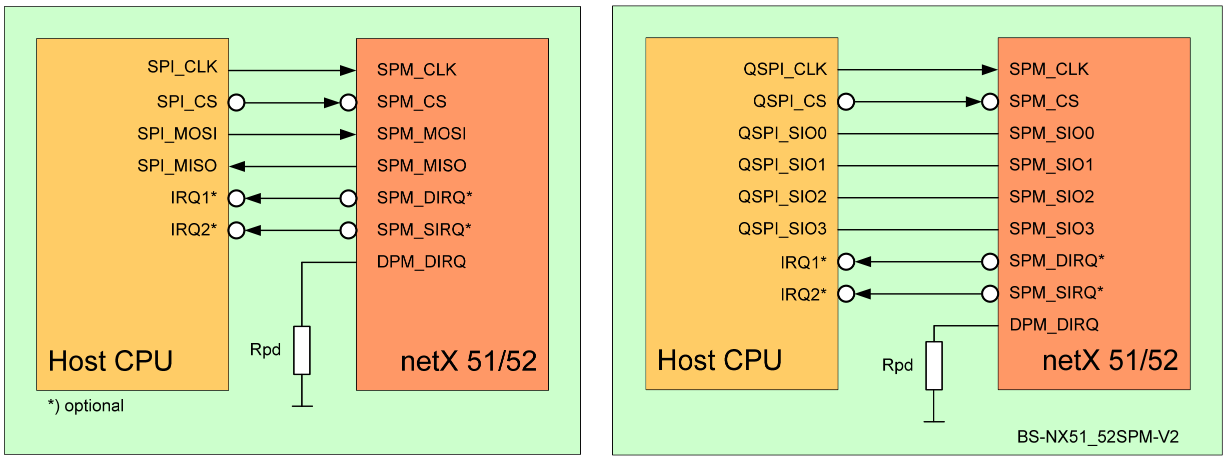


*Figure 37: SPI/QSPI to DPM interconnection*

The second stage boot loader offers a detection function to select parallel dual-port memory mode or serial port memory mode. If you want to activate the serial port memory mode, a low signal is required at pin DPM_DIRQ during startup. A pull-down resistor $R_{pd}$ with a recommended value of 2.2 kΩ is required to generate this low signal. Note that a high signal at pin DPM_DIRQ during startup activates the parallel dual-port memory mode.

If you do not want to use the detection function of the second stage boot loader, you can omit resistor $R_{pd}$, but you still have to activate the serial port mode. You can do this, e.g. by configuring the second stage boot loader.

The SPI clock is defined up to 125 MHz and the QSPI clock up to 33 MHz. Due to the lack of a speed SPI controller, we have tested the connection up to 80 MHz. The following diagram provides information on the typical transfer rate:

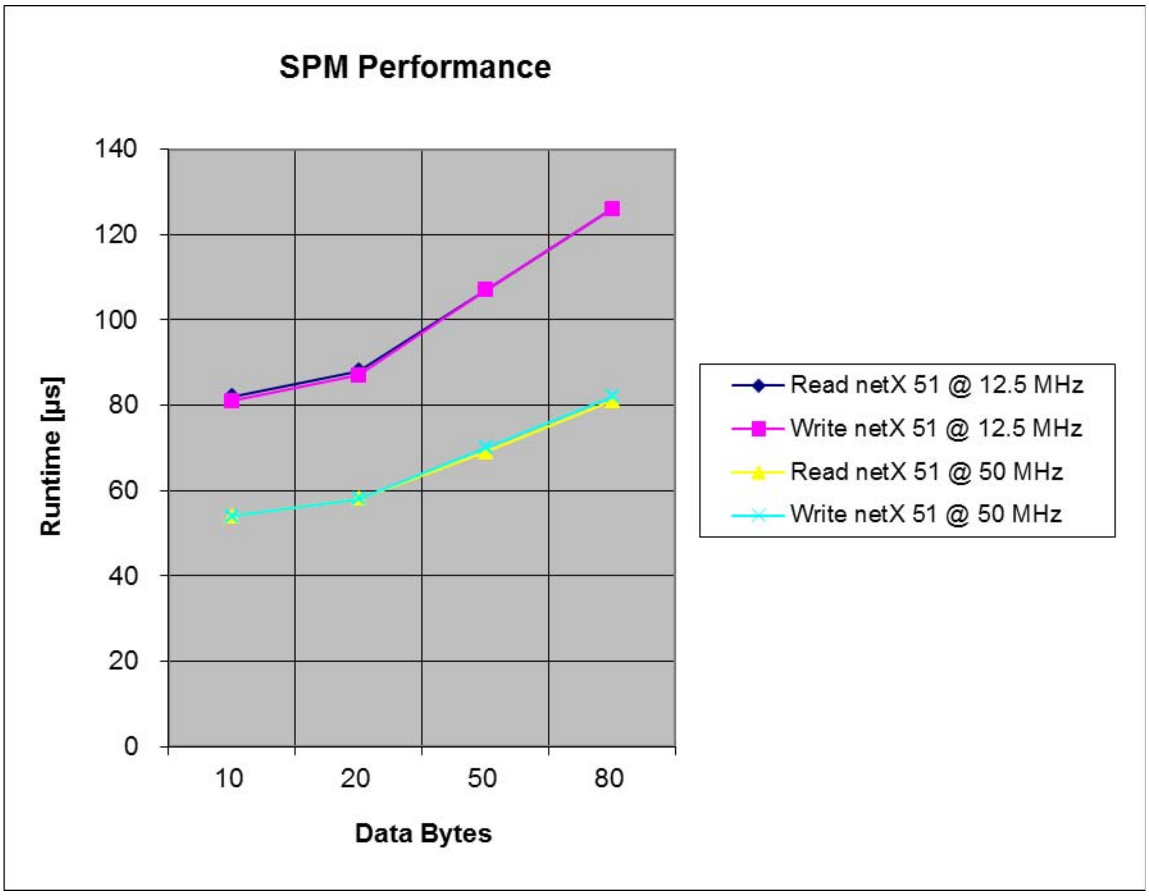


*Figure 38: SPM performance*

### 3.7.3 Extension bus

In the Extension Bus mode, the netX provides an active parallel 8, 16 and 32-bit wide asynchronous bus interface. It supports a Ready or Wait signal that allows external components to extend data cycles beyond the programmed cycle timing.

The Extension Bus can either use separate Read (EXT_RDn) and Write (EXT_WRn/WRLn, EXT_WRHn) signals ("Intel mode") or a combined R/WRn signal indicating the direction of the access along with Byte strobe signals ("Motorola mode"). In "Intel mode", either a single Write signal can be used (EXT_WRLn), combined with a Byte Enable Signals (EXT_BHEn) or two Write signals (EXT_WRLn, EXT_WRHn) for writing to the low Byte (EXT_WRLn) and high Byte (EXT_WRHn) separately.

The Extension Bus can be used to connect parallel peripherals like SRAMs, flashes, DPMs, etc. and can operate in non-multiplexed or multiplexed mode. The four different chip select signals allow connecting four completely different devices because each chip-select area is configured separately.

The netX also provides the possibility to boot from a memory device connected to the Extension Bus. In that case, the device must be connected to use the EXT_CS0n chip select signal and the Extension Bus boot mode must be selected.

#### 3.7.3.1 Non-multiplexed mode

The following schematics show some examples for common setups in non-multiplexed mode:

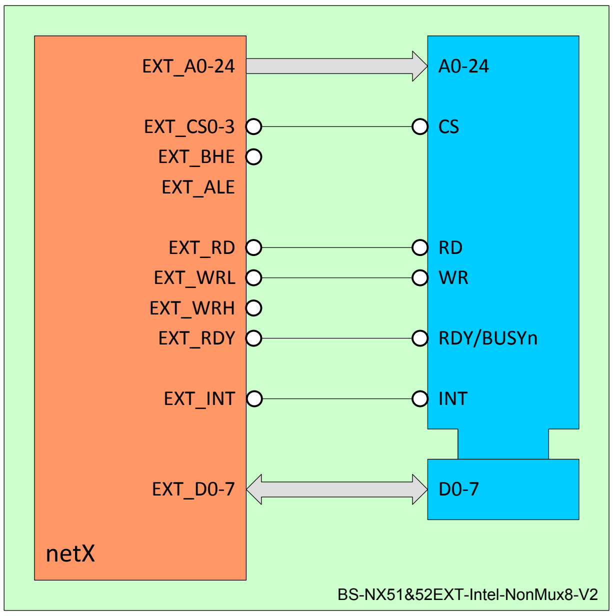


*Figure 39: netX extension bus Intel$^{TM}$ type interface circuit, 8 bit, non-multiplex*

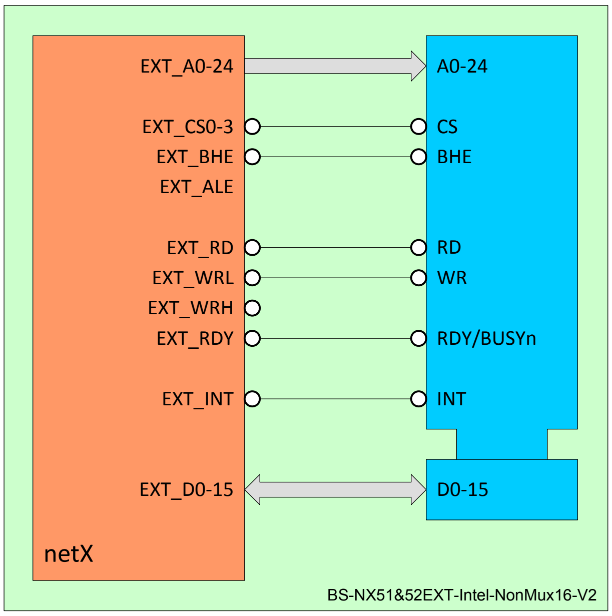


*Figure 40: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, non-multiplex*

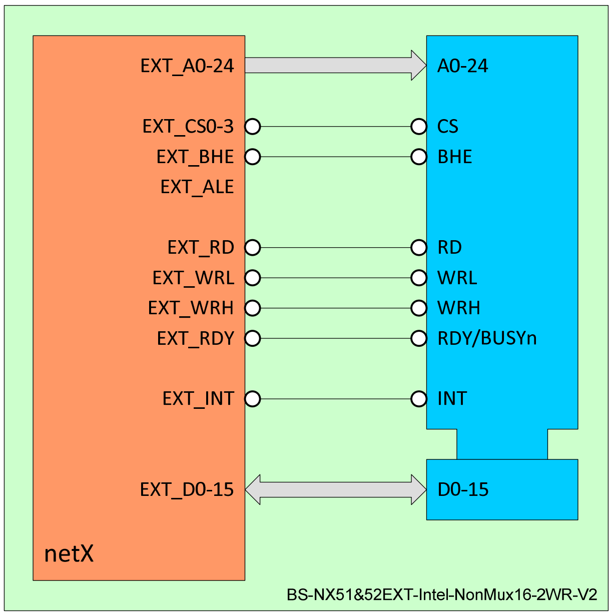


*Figure 41: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, non-multiplex, write high/low*

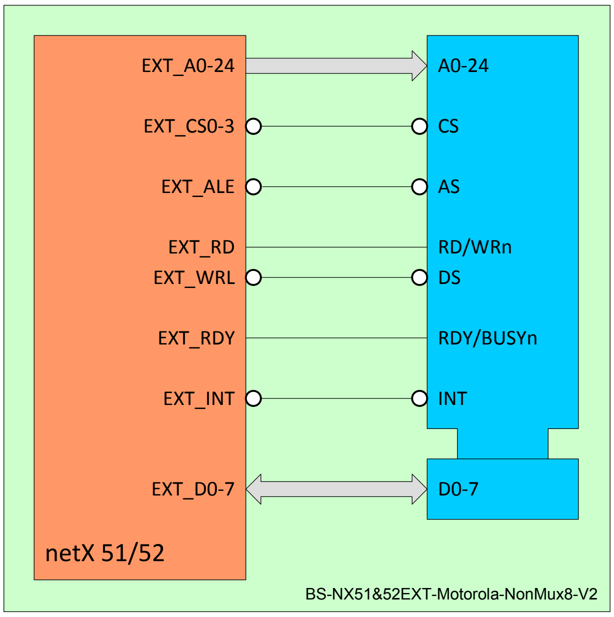


*Figure 42: netX extension bus Motorola$^{TM}$ type interface circuits, 8 bit, non-multiplex*

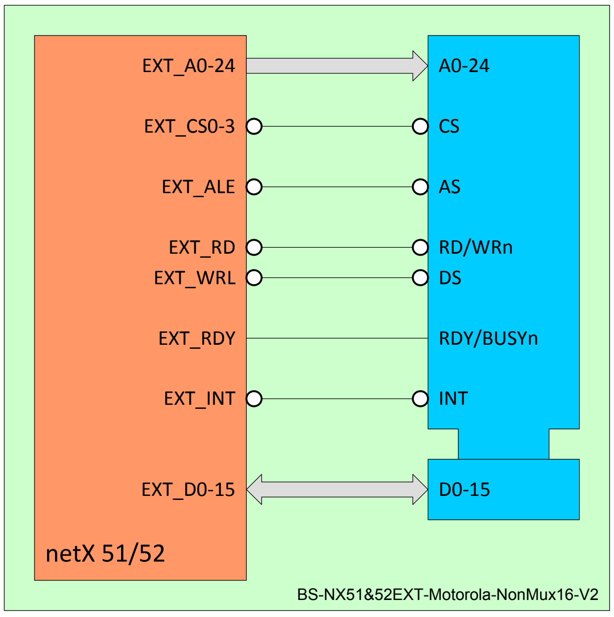


*Figure 43: netX extension bus Motorola$^{TM}$ type interface circuits, 16 bit, non-multiplex*

### 3.7.3.2 Multiplexed mode

The netX host interface can also be operated in multiplexed mode in which the data lines are alternatingly used for data and the lower address signals. The following schematics show some examples for common setups in multiplexed mode:

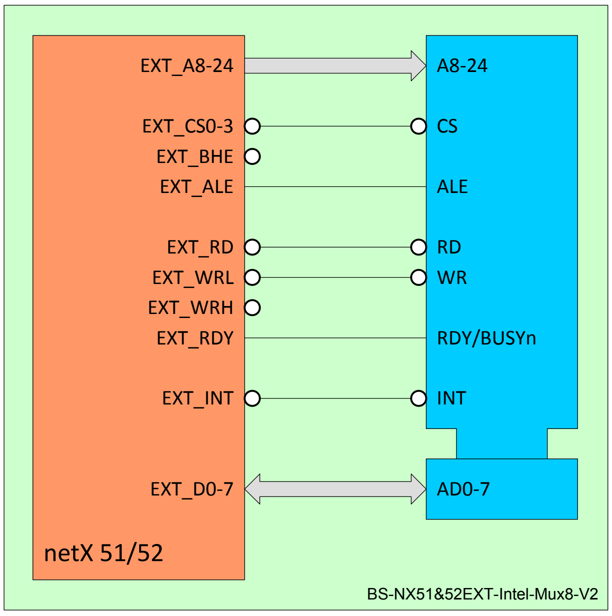


*Figure 44: netX extension bus Intel$^{TM}$ type interface circuits, 8 bit, multiplex*

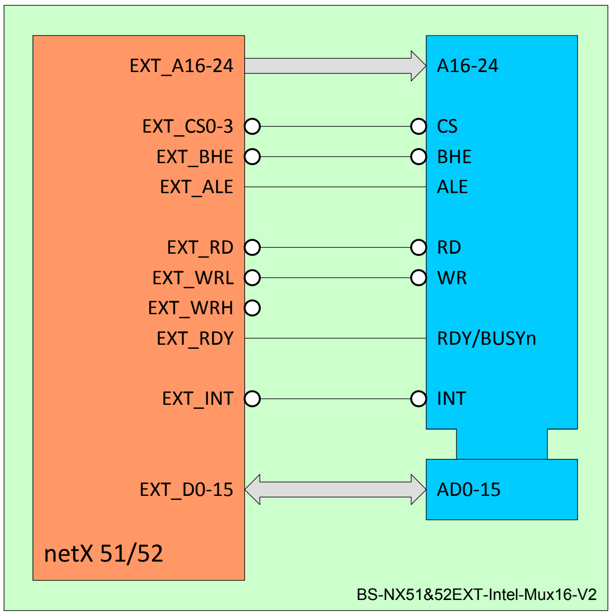


*Figure 45: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, multiplex*

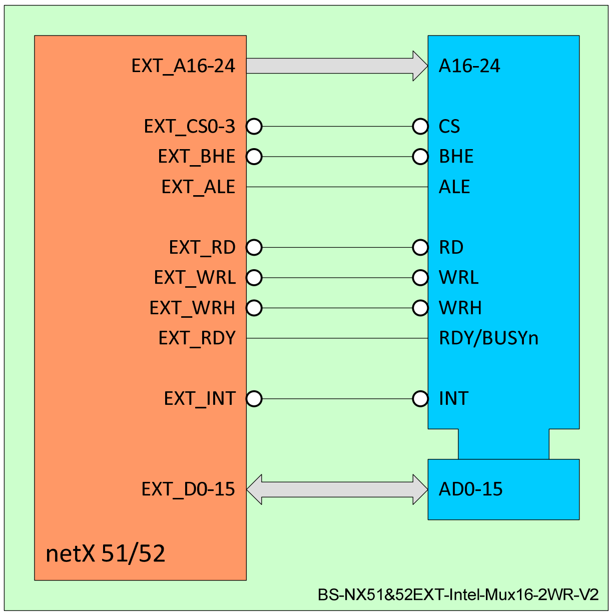


*Figure 46: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, multiplex, 2 write signals*

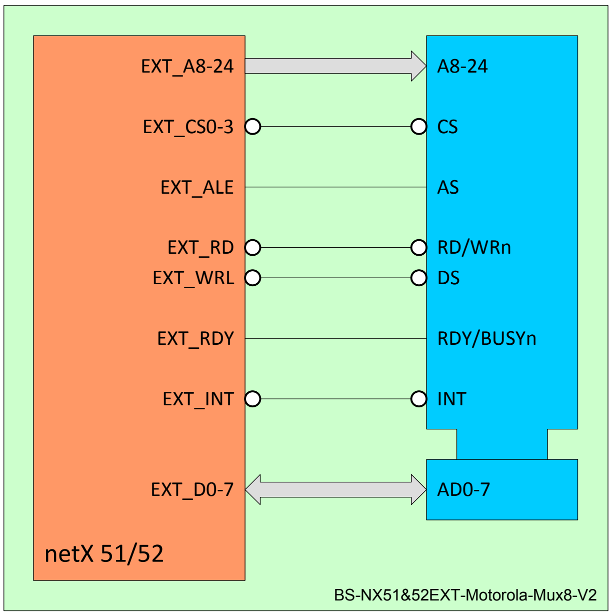


*Figure 47: netX extension bus Motorola$^{TM}$ type interface circuits, 8 bit, multiplex*

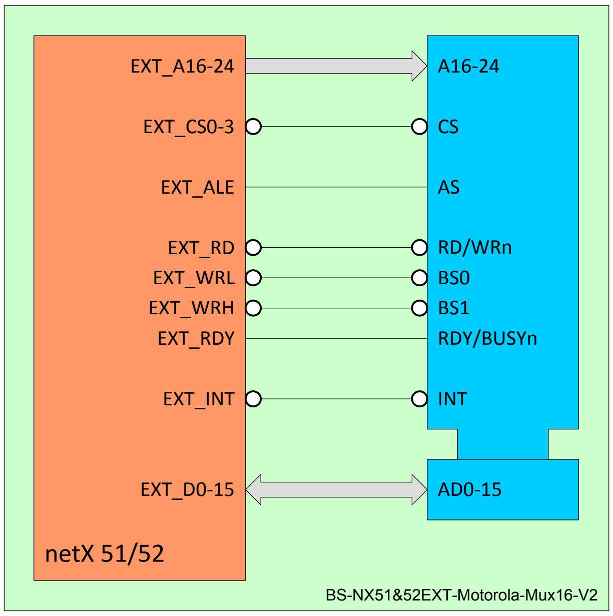


*Figure 48: netX extension bus Motorola$^{TM}$ type interface circuits, 16 bit, multiplex*

### 3.7.4 MII port 2 acts as a PHY or MAC interface

The netX 51/52 supports a third Ethernet channel which works together with the xPIC controller. Its MII interface can simulate a PHY to connect a MAC from an external Host CPU or it acts as a MAC to connect an external PHY for a third Ethernet Port.

Figure 49 shows the connection of UPD60610 PHY.

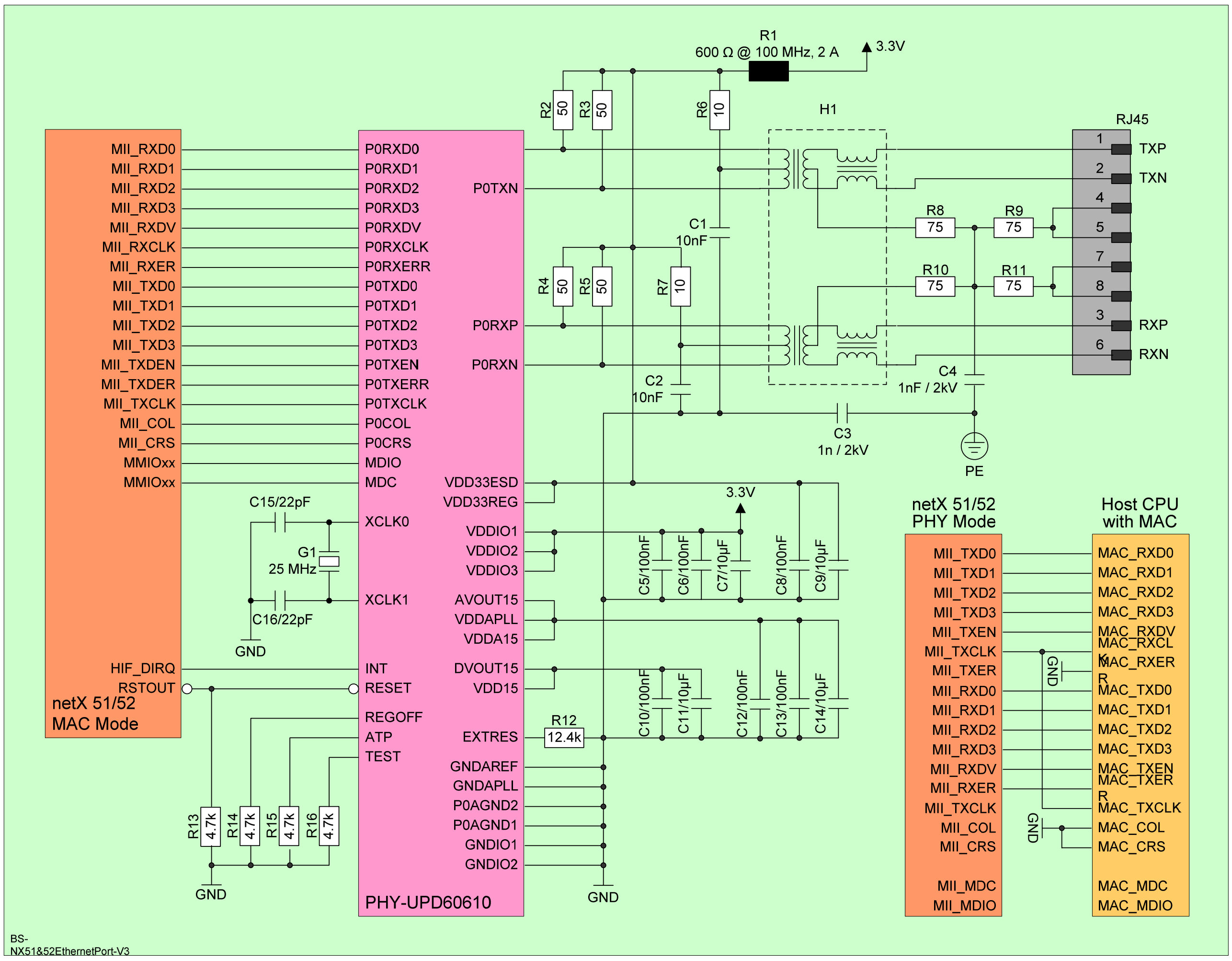


*Figure 49: Third Ethernet PHY on host interface*

*Right lower corner: Connection between netX in PHY mode and host CPU*

There are three ways to connect the external PHY/MAC to netX according to other functions on the host interface:

| | |
|---|---|
| Mode 1 | MII together with a host controller via 8/16-bit DPM or SPM<br>Interrupt is connected via HIF_DIRQn. |
| Mode 2 | MII together with an external 16-bit SDRAM<br>Interrupt is connected via MMIO44/SPM_DIRQn |
| Mode 3 | MII as an alternative to DPM signals<br>Interrupt is connected via HIF_DIRQn. |

The signals MDIO and MDC can be mapped to every free MMIO.

Table 15 gives an overview of the pin assignment (MII signals only).

| **Pins at netX 51** | **Pins at netX 52** | **HIF signals** | **Mode 1: With DPM or SPM** | **Mode 2: With 16-bit SDRAM** | **Mode 3: Alternative to DPM** |
|---|---|---|---|---|---|
| C7 | A3 | HIF_A0 | | | MII3_TXER |
| B8 | B6 | HIF_A1 | | | MII3_COL |
| C8 | C6 | HIF_A2 | | | MII3_CRS |
| C10 | C7 | HIF_A2 | | | MII3_RXD0 |
| A10 | B8 | HIF_A4 | | | MII3_RXD1 |
| B9 | A7 | HIF_A5 | | | MII3_RXD2 |
| C11 | B9 | HIF_A6 | | | MII3_RXD3 |
| D11 | C9 | HIF_A7 | | | MII3_RXDV |
| C13 | C13 | HIF_A8 | | | MII3_TXD0 |
| B12 | A10 | HIF_A9 | | | MII3_TXD1 |
| C14 | B12 | HIF_A10 | | | MII3_TXD2 |
| A17 | A14 | HIF_A11 | | | MII3_TXD3 |
| B15 | B13 | HIF_A12 | | | MII3_TXDEN |
| A16 | A13 | HIF_A13 | | | MII3_TXCLK |
| E14 | A17 | HIF_A16 | MII1_COL | | |
| D14 | A18 | HIF_A17 | MII1_CRS | | |
| J16 | G18 | HIF_D8 | | MII2_COL | |
| H15 | G16 | HIF_D9 | | MII2_CRS | |
| H16 | F17 | HIF_D10 | | MII2_TXER | |
| G16 | D18 | HIF_D11 | | MII2_MDIO (MMIO43)(1) | |
| G18 | C18 | HIF_D12 | | MII2_MDC (MMIO44)(1) | |
| G15 | B18 | HIF_D13 | | MII2_RXD1 | |
| D18 | C17 | HIF_D14 | | MII2_RXD2 | MII3_MDIO (MMIO46)(1) |
| C18 | B17 | HIF_D15 | | MII2_RXD3 | MII3_MDC (MMIO47)(1) |
| J18 | G17 | HIF_D16 | MII1_TXER | | |
| H17 | F18 | HIF_D17 | | | |
| H18 | E18 | HIF_D18 | MII1_RXDV | | |
| G17 | E17 | HIF_D19 | | | |
| B13 | C12 | HIF_D20 | MII12_TXCLK | MII12_TXCLK | |
| A14 | C11 | HIF_D21 | MII12_TXD0 | MII12_TXD0 | |
| A13 | A11 | HIF_D22 | MII12_TXD1 | MII12_TXD1 | |
| C12 | C10 | HIF_D23 | MII12_TXD2 | MII12_TXD2 | |
| B10 | A9 | HIF_D24 | MII12_TXD3 | MII12_TXD3 | |
| A11 | A8 | HIF_D25 | MII12_TXEN | MII12_TXEN | |
| A9 | A6 | HIF_D26 | MII1_RXD0 | | |
| C9 | B7 | HIF_D27 | MII1_RXD1 | | |
| B6 | A2 | HIF_D28 | MII12_RXCLK | MII12_RXCLK | |
| A7 | A4 | HIF_D29 | MII12_RXER | MII12_RXER | |
| A2 | D3 | HIF_D30 | MII1_RXD2 | | |
| J17 | H17 | HIF_D31 | MII1_RXD3 | | |

| Pins at netX 51 | Pins at netX 52 | HIF signals | Mode 1: With DPM or SPM | Mode 2: With 16-bit SDRAM | Mode 3: Alternative to DPM |
|---|---|---|---|---|---|
| A18 | A16 | HIF_BHE1 | | | MII3_RXER |
| C15 | B16 | HIF_BHE3 | | MII2_RXDV | |
| B18 | C15 | HIF_RDY | | | MII3_RXCLK |
| C17 | D16 | HIF_DIRQn | | MII2_RXD0 | |
| xx | xx | MMIOxx | MII1_MDIO (MMIOxx)(1) | | |
| xx | xx | MMIOxx | MII1_MDC (MMIOxx)(1) | | |

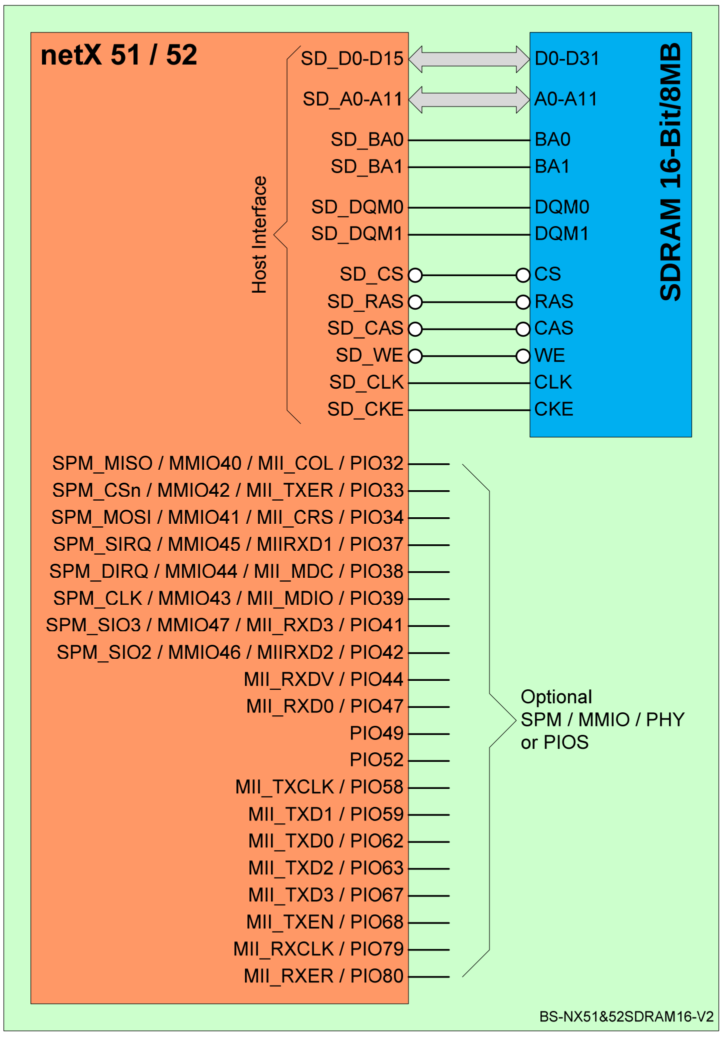
*Table 15: Third PHY MII pin connection*

**Note** (1): The signals MDIO and MDC can be mapped to every free MMIO. The positions given in the list are mere recommendations.

## 3.7.5 SDRAM memory extension

The host interface memory controller supports a 16 and 32-bit SDRAM. netX 51 can use the SDRAM connected to host interface as xPIC memory and the SDRAM on memory interface to the ARM CPU.

In case of netX 52 we recommend connecting an SDRAM at the host interface if you want to use linkable object modules (LOM) together with the additional application code.



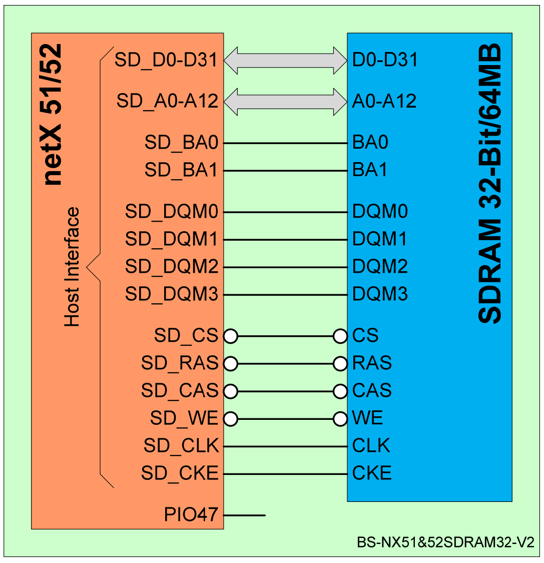


*Figure 50: 16-bit (left) / 32-bit (right) SDRAM at host interface*

### 3.7.6 MMIO signals

Up to eight pins of the host interface can be configured as MMIO40 to MMIIO47 and every MMIO can be configured as one signal of the internal peripherals. It is a great feature to have an I2C, SPI, UART, CAN or GPIO as part of the host interface.

### 3.7.7 PIO signals

Every pin of the host interface can become a general PIO pin. E.g., if you use the serial DPM with its four SPM signals for connection to a Host CPU, you have 54 additional PIO pins.

### 3.7.8 External pull-ups/pull-downs, unused signals

As already mentioned at the beginning of chapter *Host interface* on page 36, the netX 51/52 provides internal pull-up resistors on nearly all host interface pins. Hence, any host interface pin that is not used in a design may simply be left open. The only exception is the CLOCKOUT signal. It should be equipped with an external pull-up or pull-down (e.g. 10k) if not used to avoid any cross-currents from a floating signal.

As most host interface control signals are active low, the internal pull-ups usually provide the required inactive level if the netX is in reset state or if the host interface has not yet been configured by the firmware (all host interface pins are switched to input mode upon reset). However, if for example an active high interrupt signal is used, an external pull-down resistor may be required on this signal to avoid an undesired active state (this may also apply to the RDY/BUSYn signal which has a programmable polarity as well).

## 3.8 Multiplexed IO matrix (MMIO)

netX 51/52 have a lot of internal peripherals like I2C, SPI, UART, CAN or GPIO which have to be connected to external signals. Normally not all of them are used in a certain application. Therefore you can select a part of those signals and connect them via a multiplex matrix with the MMIO pins at the housing.

Most of the signals can be connected to any of these MMIOs. Some groups of signals with hard timing requirements like fiber optic interface or ETM are fix related to some MMIOs.

**Note:** For your PCB, carefully check the flexible and MMIO mapping which is fixed. Otherwise a redesign may be required!
All MMIOs have internal pull-downs (typically 50 K). Consider this if you connect external LEDs (anode at netX) and switches (one side to +3.3V) which will prevent slight shining respectively save resistors in your design.

| MMIOs | On netX 51 | On netX 52 |
|---|---|---|
| MMIOs pins | 40 | 24 |
| Additional at host interface | 8 | 8 |
| Additional at CLKOUT | 1 | - |
| Fiber Optic at MMIO0-7 | 8 signals, 41 MMIOs left | 8 signals, reduce MMIOs to 24 |
| Third Ethernet port at MMIOs fixed MMIOs | 18 signals as MAC, 31 MMIOs<br>14 signals as PHY, 35 MMIOs | 18 signals as MAC, 31 MMIOs<br>14 signals as PHY, 35 MMIOs |
| ETM as fixed MMIOs | 23 signals, 26 MMIOs left | - |

For detailed table with all mappable signals, see reference [1].

After power-up, the UART signals are mapped automatically to special MMIOs. This gives you the possibility to download the firmware via the UART0 in the ‘Serial boot mode’. For details, refer to chapter *Configured by the RDY/RUN pins* on page 21.

| Signal | Type | Description | netX 51 | netX 52 |
|---|---|---|---|---|
| UART0_CTS | I | UART0 Clear To Send | MMIO32 | MMIO22 |
| UART0_RTS | OZ | UART0 Request To Send | MMIO33 | MMIO23 |
| UART0_RXD | I | UART0 Receive Data | MMIO34 | MMIO20 |
| UART0_TXD | OZ | UART0 Transmit Data | MMIO35 | MMIO21 |
| UART1_CTS | I | UART1 Clear To Send | MMIO36 | - |
| UART1_RTS | OZ | UART1 Request To Send | MMIO37 | - |
| UART1_RXD | I | UART1 Receive Data | MMIO38 | - |
| UART1_TXD | OZ | UART1 Transmit Data | MMIO39 | - |

The following table shows the default configuration of the MMIO matrix as it is used by our protocol stacks. This fits to our development boards NXHX 51-ETM or NXHX 52-JTAG. Therefore, if you use a ‘loadable firmware’, you can run it directly on these development boards. If you create your own design with another mapping, you have to use the ‘Tag List Editor’ to rearrange the right mapping. In general, you can map any peripheral signal to any MMIO pin with this tool without having to recompile the firmware.

**netX 51**

| **Signal** | **netX 51 / NXHX 51-ETM** | |
|---|---|---|
| | **Peripheral** | **Pin** |
| XM0_RX | XM0_RX | MMIO00 |
| XM0_TX | XM0_TX | MMIO01 |
| XM0_IO1 | XM0_IO1 | MMIO02 |
| XM0_IO0 | XM0_IO0 | MMIO03 |
| XM1_TX | XM1_TX | MMIO04 |
| XM1_IO1 | XM1_IO1 | MMIO05 |
| XM1_RX | XM1_RX | MMIO06 |
| XM1_IO0 | XM1_IO0 | MMIO07 |
| | | MMIO08 |
| | | MMIO09 |
| | | MMIO10 |
| | | MMIO11 |
| LINK0n | PHY0_LINK | MMIO12 |
| ACT0n | PHY0_ACT | MMIO13 |
| LINK1n | PHY1_LINK | MMIO14 |
| ACT1n | PHY1_ACT | MMIO15 |
| | | MMIO16 |
| | | MMIO17 |
| | | MMIO18 |
| | | MMIO19 |
| | | MMIO20 |
| | | MMIO21 |
| | | MMIO22 |
| | | MMIO23 |
| | | MMIO24 |
| | | MMIO25 |
| | | MMIO26 |
| | | MMIO27 |
| COM0_Gn | PIO0 | MMIO28 |
| COM0_Rn | PIO1 | MMIO29 |
| COM1_Gn | PIO2 | MMIO30 |
| COM2_Rn | PIO3 | MMIO31 |
| UART0_CTSn | UART0_CTSn | MMIO32 |
| UART0_RTSn | UART0_RTSn | MMIO33 |
| UART0_RXD | UART0_RXD | MMIO34 |
| UART0_TXD | UART0_TXD | MMIO35 |
| | | MMIO36 |

| **Signal** | **netX 51 / NXHX 51-ETM** | |
|---|---|---|
| | **Peripheral** | **Pin** |
| | | MMIO37 |
| | | MMIO38 |
| | | MMIO39 |

## netX 52

| **Signal** | **netX 52 / NXHX 52-JTAG** | |
|---|---|---|
| | Peripheral | Pin |
| XM0_RX | XM0_RX | MMIO00 |
| XM0_TX | XM0_TX | MMIO01 |
| XM0_IO1 | XM0_IO1 | MMIO02 |
| XM0_IO0 | XM0_IO0 | MMIO03 |
| | | MMIO04 |
| | | MMIO05 |
| | | MMIO06 |
| | | MMIO07 |
| | | MMIO08 |
| | | MMIO09 |
| | | MMIO10 |
| | | MMIO11 |
| COM0_Gn | PIO0 | MMIO12 |
| COM0_Rn | PIO1 | MMIO13 |
| COM1_Gn | PIO2 | MMIO14 |
| COM2_Rn | PIO3 | MMIO15 |
| LINK0n | PHY0_LINK | MMIO16 |
| ACT0n | PHY0_ACT | MMIO17 |
| LINK1n | PHY1_LINK | MMIO18 |
| ACT1n | PHY1_ACT | MMIO19 |
| UART0_RXD | UART0_RXD | MMIO20 |
| UART0_TXD | UART0_TXD | MMIO21 |
| UART0_CTSn | UART0_CTSn | MMIO22 |
| UART0_RTSn | UART0_RTSn | MMIO23 |

## 3.9 General purpose IOs

General Purpose IOs (GPIO) are highly versatile multifunctional peripherals:

- 32 GPIOs which can be programmed or combined individually with 5 GPIO counters/timers with their own threshold/capture register. This allows many different PWM and capture functions:
  - Counting on edge or level, once or continuously
  - Counting external events on edge or level
  - Counting PWM: symmetric (triangle) or asymmetric (saw tooth)
  - Generating PWM signal (DC/DC) depending on the level of GPIO input
  - Generating IRQ in watchdog mode: If the input does not change within the programmed time
  - Generating interrupt on external edge or level
- GPIO outputs can be set or read in common via one register
- All GPIOs are accessible via two channels on two address ranges
  - ARM and xPIC can work parallel without influencing each other mutually
- GPIOs are shared with IO-Link pins
  - Each IO-Link channel needs 4 pins

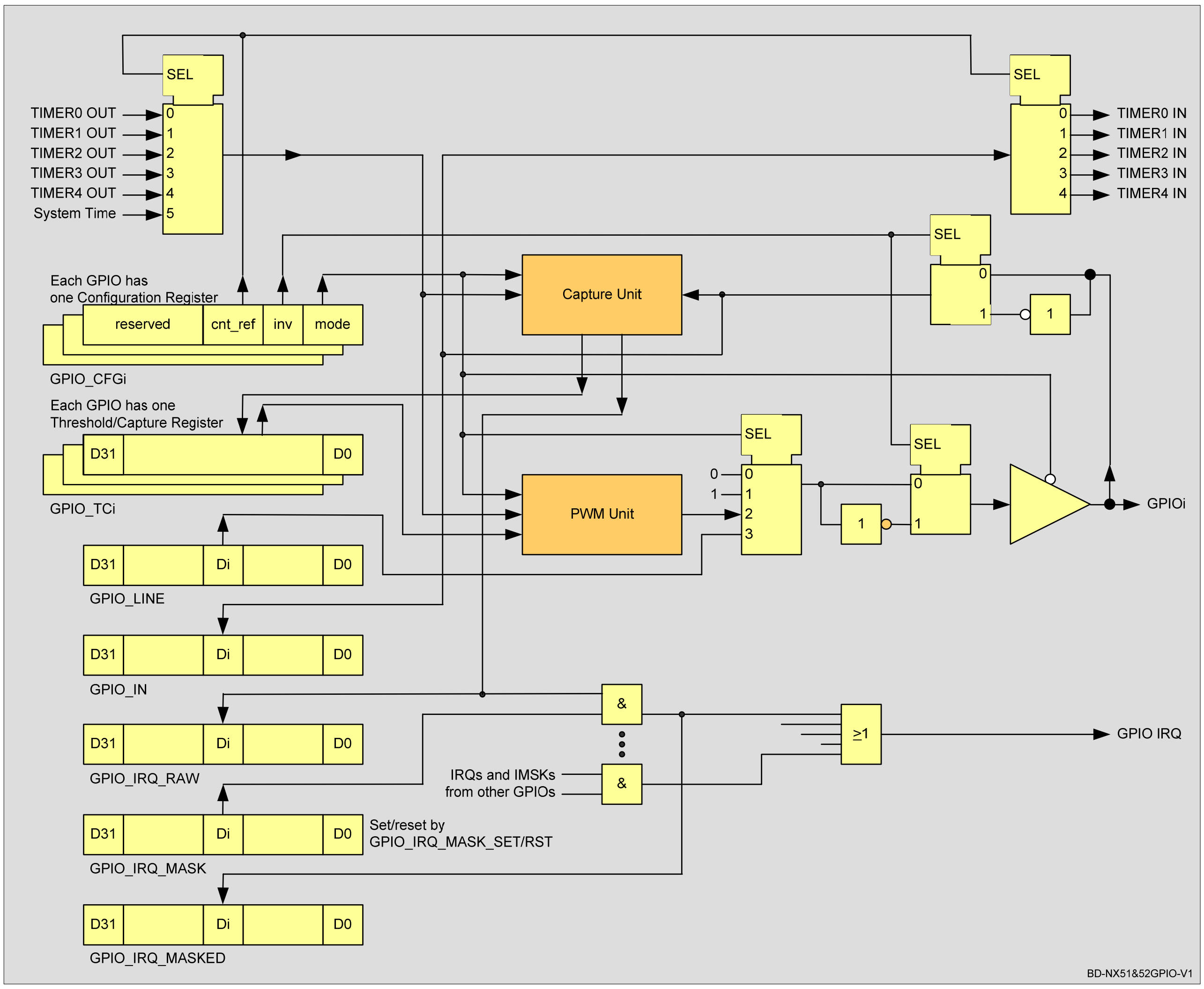


*Figure 51: GPIO block diagram*

## 3.10 Serial interfaces

### 3.10.1 UARTs

The netX provides a total of three UARTs (each with RX, TX, RTSn, CTSn) that interface directly to common RS-232 or RS-485 transceivers, see Figure 52. UART0 must be used if the serial boot mode via UART is to be available. UART0 may also be used as a diagnostic port by Hilscher firmware.

This example shows the connection for UART0. UART1 and UART2 are connected equally. On netX 51/52, the UARTs are accessible only via the multiplex matrix. Their signals can thus be mapped to any of the 48/32 MMIO pins.

**Note:** With UART0 we strongly recommend you to use the MMIOs shown in this example since this is the default mapping after power-up. The 'serial boot mode' via UART0 can be used with this default mapping only!
Due to an improved ROM boot loader, netX 51 does not require any external pull-ups on UART0 signals if UART0 is not used.

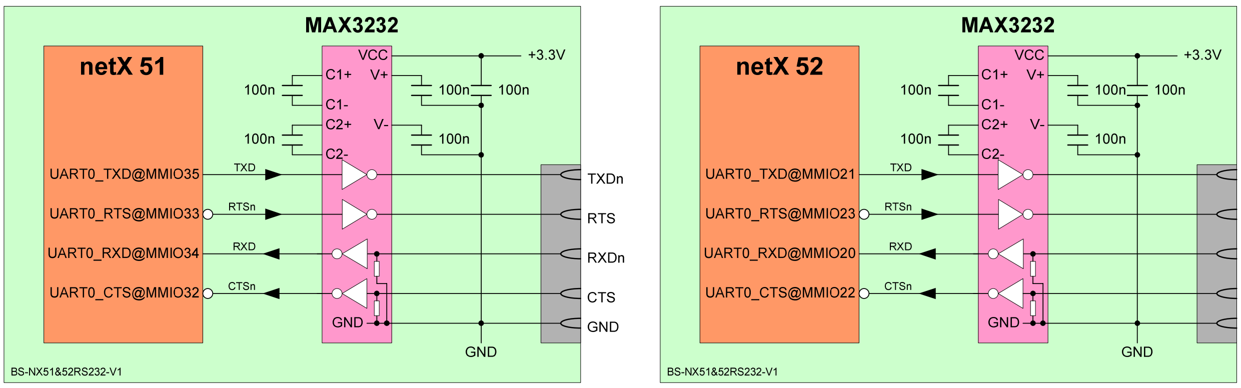


*Figure 52: netX 51/52 UART0 with RS-232 transceiver*

### 3.10.2 USB

netX 51/52 provide a USB interface which is compliant with USB 1.1 and which can be used in device mode (Downstream Port).

The netX USB interface allows you to connect the netX to a PC which can download and flash firmware, read and modify register values, and run hardware test applications by the help of freely available Hilscher software tools.

For using this handy and yet simple debug and service connection, we always recommend the implementation of a USB device port if board size constraints permit.

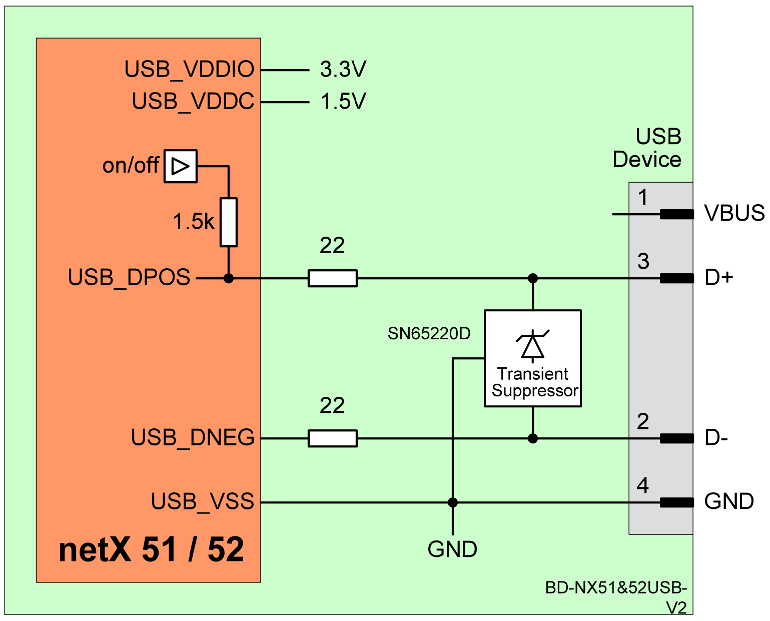


*Figure 53: netX 51/52 USB DOWNstream port (device mode)*

### 3.10.3 SPI

netX 51/52 have an additional SPI controller which is available at the MMIO matrix.

Basic technical data for SPI 1 - master and slave functionality

- SPI clock max. 50 MHz in master mode and 33 MHz in slave mode
- Clock polarity high or low, clock phase 0 or 1
- Data frame size from 4-bit to 16-bit data words
- 16-word deep transmit and receive FIFO
- DMA support

### 3.10.4 I2C

netX 51/52 have two I2C controllers. I2C0 is available with its signals I2C0_SCL and I2C0_SDA on dedicated pins and at the MMIO matrix. The I2C1 controller is available at the MMIO matrix.

The basic technical data are for I2C 0 / 1 - master and slave functionality

- 50 kHz to 3.4 MHz, 7-bit and 10-bit slave address
- 16-byte master and slave FIFO
- I2C sequence controller for acknowledge polling or long data transfer
- DMA support

### 3.10.5 CAN

Even today CAN is a very popular network. We have implemented a dedicated CAN controller in our netX 51/52. Of course, you can configure the communication channel to run as CAN controller, but in doing so you will lose the real-time Ethernet capability.

Basic technical data about compatible SJA 1000 functions

- Fully compliant with CAN standard rev. 2.0A and rev. 2.0 B
- Supports 11-bit and 29-bit identifier
- Bit rates up to 1 MBit/s
- Extended receive buffer (64-byte FIFO)
- Error counters with read / write access
- Programmable error warning limit
- Single-shot transmission (no re-transmission)
- Listen only mode (no acknowledge, no active error flags)
- Bit rate detection, using acknowledged and unacknowledged frames
- Acceptance filter extension (4-byte code, 4-byte mask)
- Abort transmission possible

### 3.10.6 Ethernet diagnostic port

The third Ethernet port is frequently used as a general diagnostic port. It does not have any hard real-time functionality. Since it is shared with some host interface signals, it is also described in a chapter of the host interface.

## 3.11 IO-Link

netX 51 has an IO-Link controller that supports IO-Link version V1.1 with 8 channels.

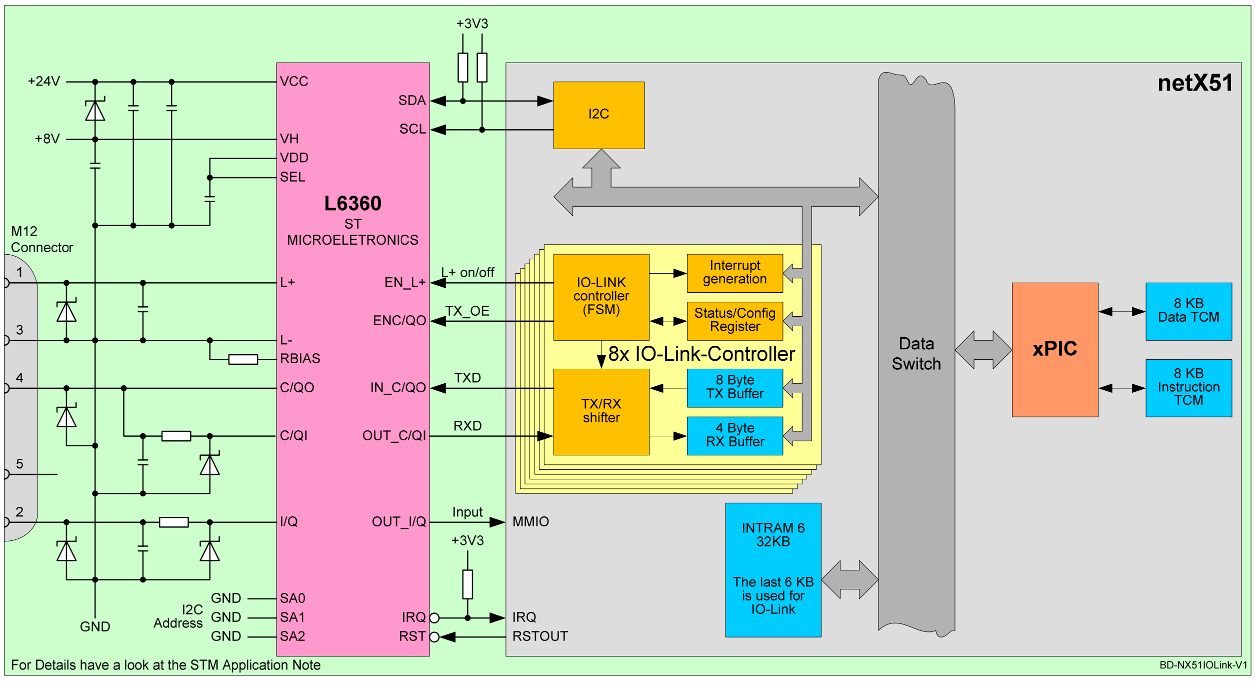


*Figure 54: IO-Link connections*

netX 52 has an IO-Link controller that supports IO-Link version V1.1 with 4 channels.

## 3.12 Fieldbus interfaces

The netX controllers are equipped with flexible communication processors (xPEC/xMAC units, also referred to as XC units). They allow realizing virtually any fieldbus interface on the market by simply adding the appropriate physical layer circuit.

netX 51/52 provide two XC units.

All common fieldbus interfaces are serial interfaces with a transmit and a receive signal. Some of them use an additional control signal or status signal. Owing to this, the xMAC units that are directly connected to the fieldbus physical layer circuit provide an XMi_TX, an XMi_RX and two I/O signals (XMi_IO0 and XMi_IO1). Current fieldbus interfaces use only one of them. On netX 51/52, these I/O signals have already been extended to a total of 5 signals for future use.

Each XC unit also provides a clock input/output signal (XMi_ECLK) which allows synchronizing external hardware to the XC clock or feeding an external clock to the XC unit. Both options are currently not used.

The XMAC signals on the netX 51/52 are routed via the multiplex matrix and connected to the MMIO pins. For the TX signal and the optional external clock, there is a direct pin sharing option that connects these signals directly to certain MMIO pins and bypasses the multiplex matrix (which is clocked by 100 MHz). This allows a higher resolution. We therefore strongly recommend adopting the proposed MMIO assignment for the XMAC signals as shown in Figure 55.

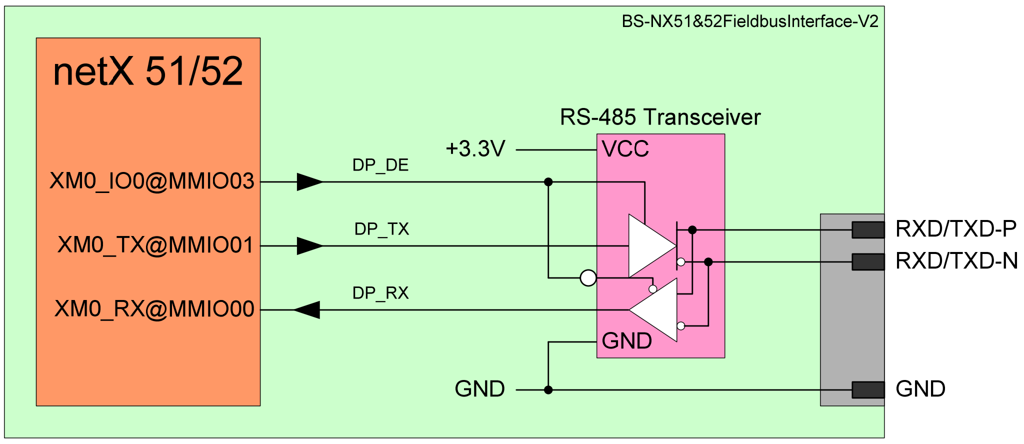


*Figure 55: netX 51/52 fieldbus interface*

### 3.12.1 CANopen interface

A CANopen interface can be implemented as shown in the following schematics.

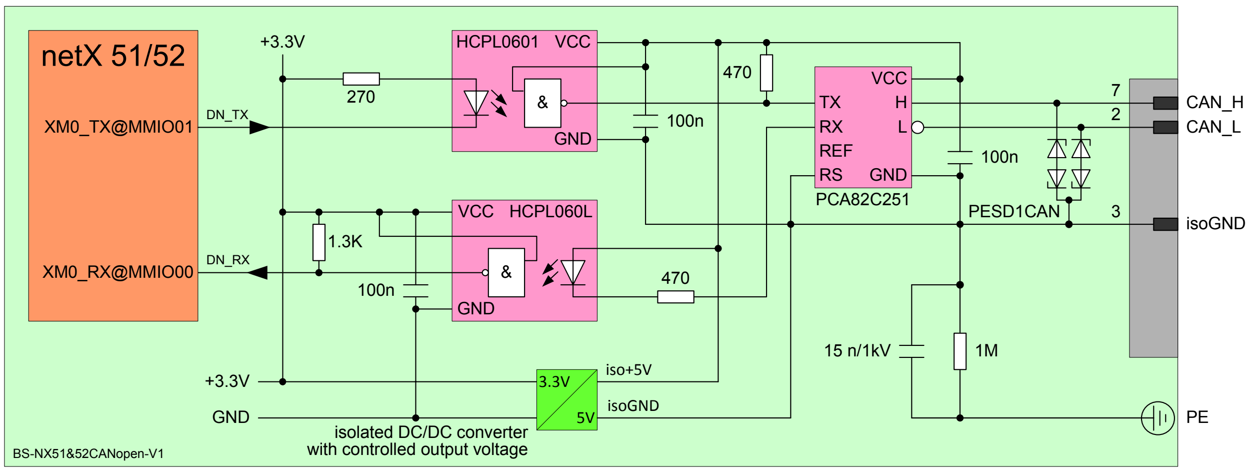


*Figure 56: Basic circuit of netX CANopen interface*

## 3.12.2 CC-Link interface

A CC-Link interface can be implemented as shown in the following schematics based on the reference schematics from the CC-Link specification.

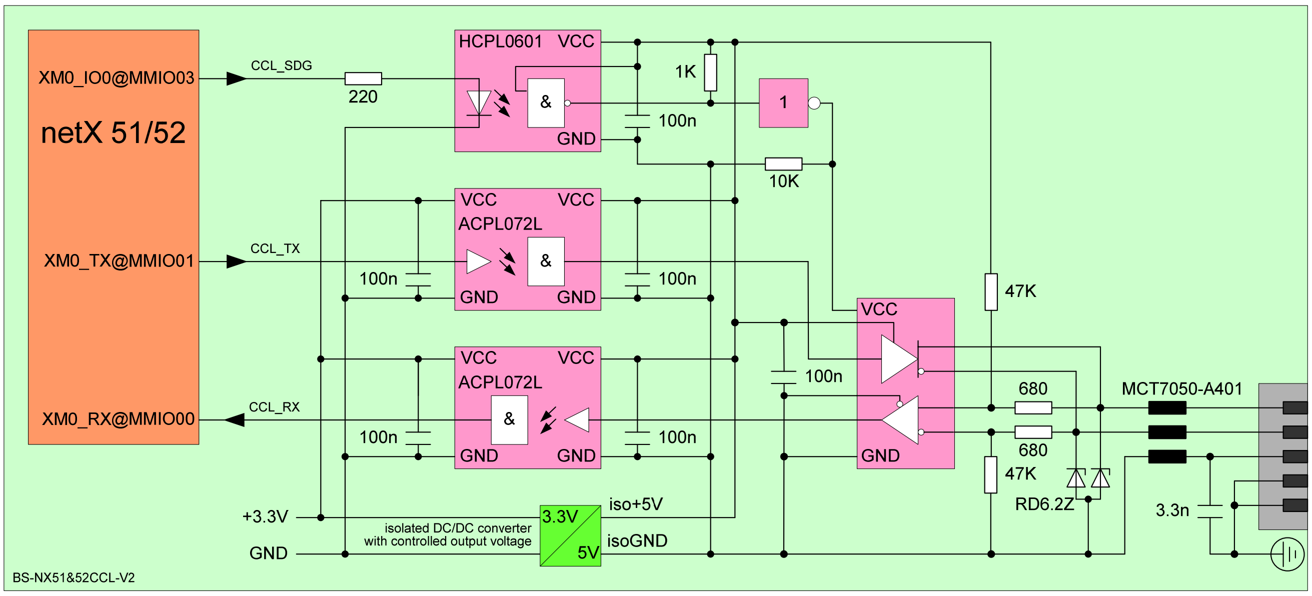


*Figure 57: Basic circuit for netX CC-Link interface*

### 3.12.3 DeviceNet interface

A DeviceNet interface can be implemented according to the following simplified schematics.

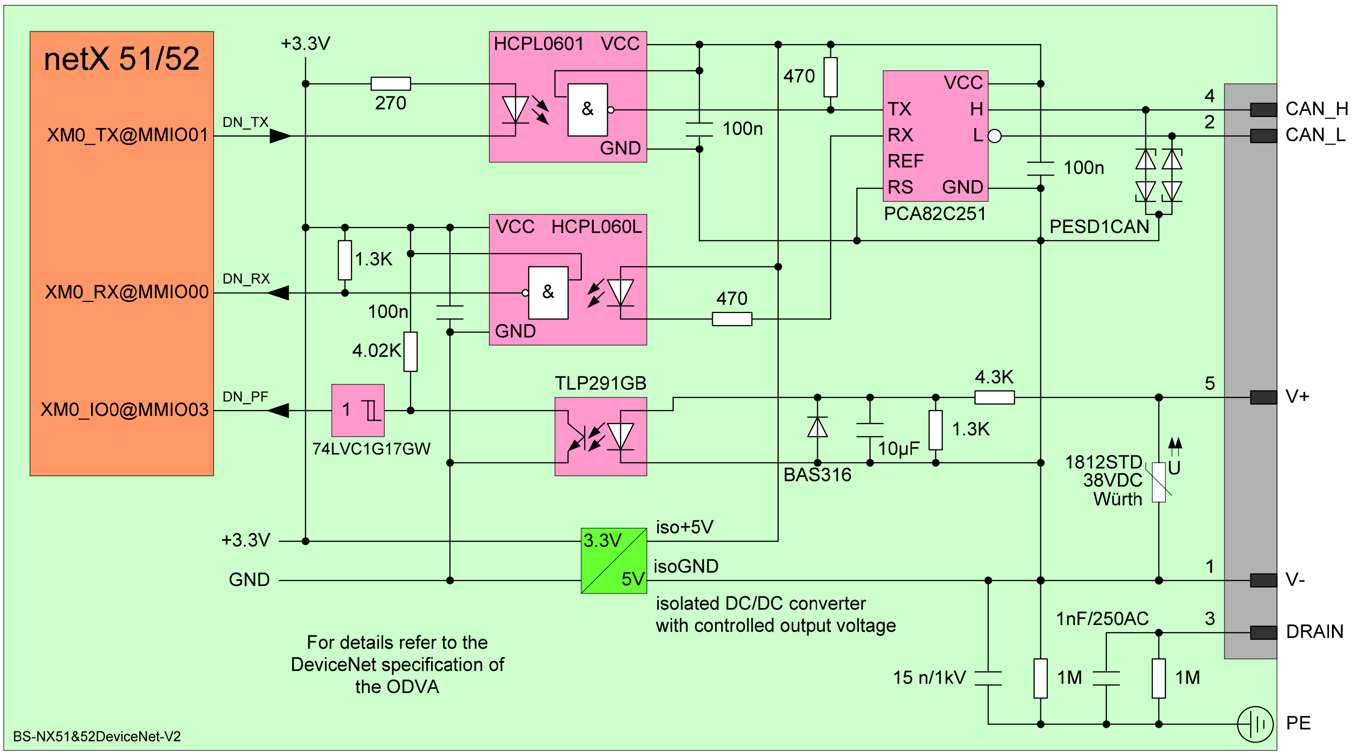


*Figure 58: Basic circuit for netX DeviceNet interface*

### 3.12.4 PROFIBUS interface

A PROFIBUS interface can be implemented according to the following simplified schematics.

**Note:** The (isolated) 3.3 V to 5 V DC-DC converter should be a regulated model or it should be equipped with an appropriate downstream LDO regulator. Otherwise, the secondary voltage may be too high resulting in out-of-spec signal levels on the PROFIBUS line!

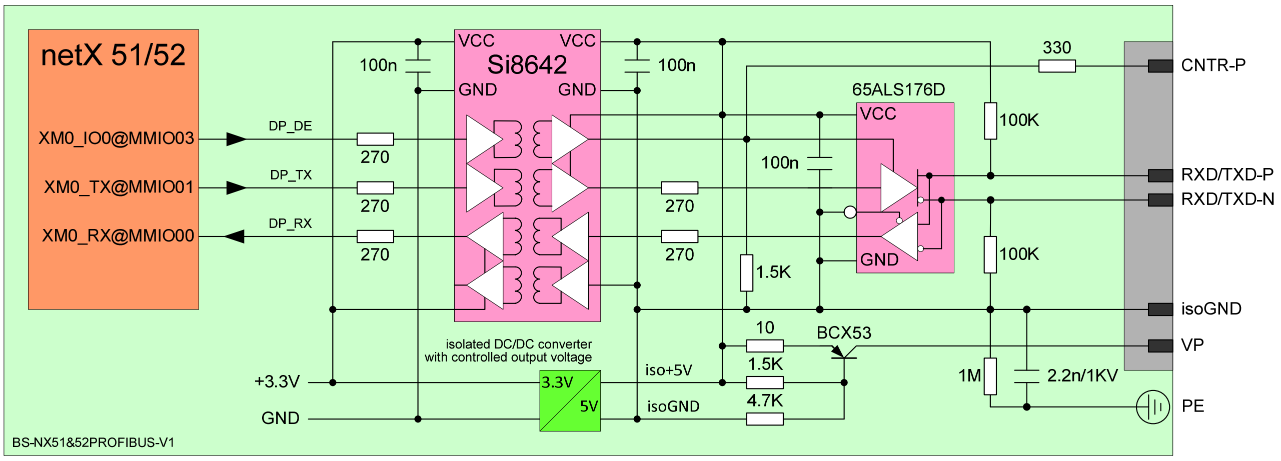


*Figure 59: Basic circuit for netX PROFIBUS interface*

### 3.12.5 Fieldbus status LEDs

Up to 2 fieldbus ports are possible. Two status LED signals are defined. Their function depends on the type of fieldbus interface.

| **Function** | **Signal** | **Pin name and number** | | | |
|---|---|---|---|---|---|
| | | netX 51 | | netX 52 | |
| Fieldbus 0, status 0 | PIO00 | MMIO28 | T9 | MMIO12 | R2 |
| Fieldbus 0, status 1 | PIO01 | MMIO29 | U10 | MMIO13 | U2 |
| Fieldbus 1, status 0 | PIO02 | MMIO30 | T10 | MMIO14 | T2 |
| Fieldbus 1, status 1 | PIO03 | MMIO31 | U11 | MMIO15 | R3 |

*Table 16: Status LEDs for fieldbus ports*

On the netX 51/52, all PIO signals are connected to the internal multiplex matrix, i.e. the fieldbus status LEDs must be connected to MMIO pins. MMIO 28 – 31 on netX 51 and MMIO 12 – 15 on netX 52 must always be used for the connection to PIO signals 0 – 3, whenever the design is to be operated with Hilscher loadable firmware.

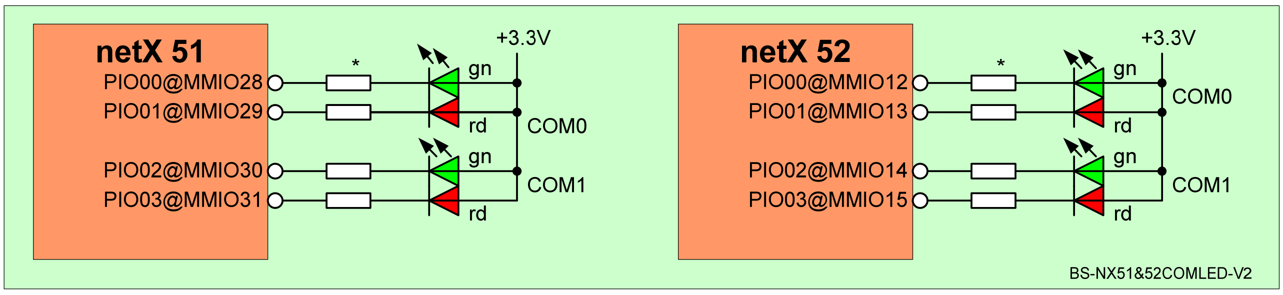


*Figure 60: Fieldbus status LEDs*

**Note *:** MMIO28…31: Adaption of the LED resistance to the brightness considering the driver strength. Usually 220 Ω to 470 Ω.

### 3.12.6 Fieldbus address switches

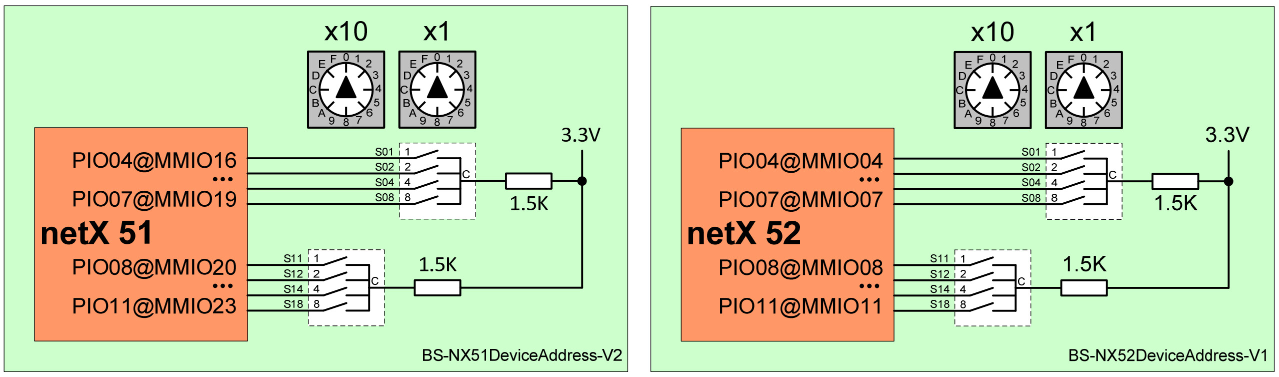


*Figure 61: Fieldbus address switches*

**Note:** This feature is not supported in our standard protocol stack, but its implementation is planned in the near future.
netX 52 has less MMIO pins. If signals overlap, i.e. sync signals, you have to use other MMIOs or the HIF pins.

## 3.13 Real-time Ethernet (RTE) interface

The netX 51/52 have two integrated physical layer units (PHYs) for Ethernet communication that allows you to build systems with two Ethernet ports using only a few external components like pull-ups and transformer(s). You can operate the PHYs in the modes:

- Twisted pair (10BASE-T / 100BASE-TX)
- Fiber optic (100BASE-FX)

### 3.13.1 Twisted pair

For 10BASE-T or 100BASE-TX Ethernet communication, connect the PHY as shown in Figure 62.

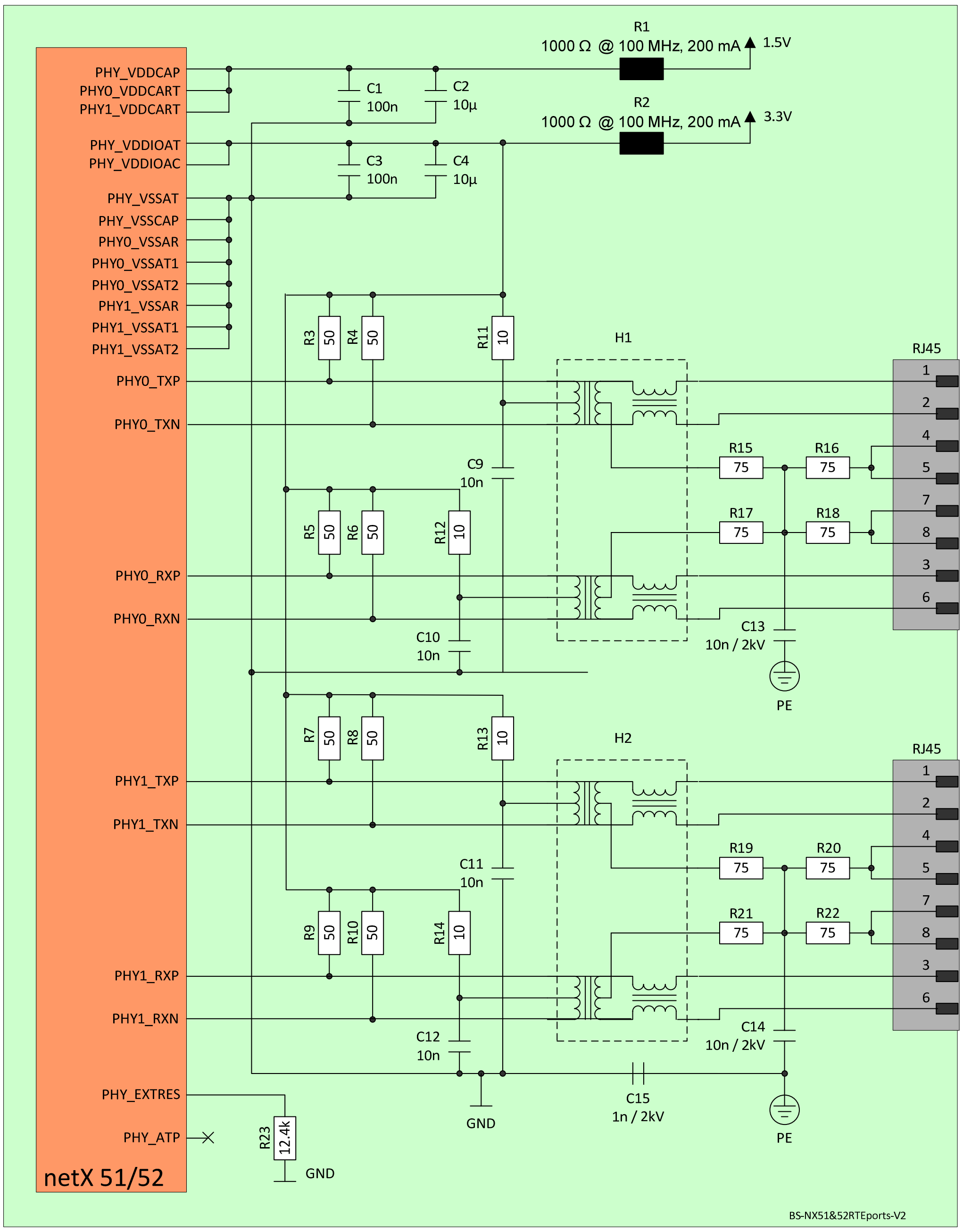


*Figure 62: netX 51/52 2-channel Ethernet circuit (Twisted pair)*

| Component | Value | Tolerance | Rating |
|---|---|---|---|
| R1, R2 | 600 Ω @ 100 MHz | | 2 A |
| R3, R4, R5, R6, R7, R8, R9, R10 | 50 Ω | 1% | 125 mW |
| R11, R12, R13, R14 | 10 Ω | 1 % | 63 mW |
| R15, R16, R17, R18, R19, R20, R21, R22 | 75 Ω | 1 % | 63 mW |
| R23 | 12.4 kΩ | 1 % | 63 mW |
| C1, C3 | 100 nF | | 6.3 V |
| C2, C4 | 10 µF | | 6.3 V |
| C9, C10, C11, C12 | 10 nF | 20 % | |
| C13, C14 | 1 nF | | 2 kV |
| C15 | 1 nF | | 2 kV |
| H1, H2 | H1102, HX1188 (Pulse Engineering.) | | |

*Table 17: Ethernet circuit component specification*

The selected Ethernet transformer(s) H1, H2, (Table 17 lists three examples) have to be 1:1 ratio types with center tap. They should be symmetric, i.e. transmit and receive path may be swapped. This is necessary to support the auto-crossover feature that is mandatory for most real-time Ethernet protocols!

Instead of a separate transformer, secondary side resistors (75 Ω) and RJ45 jack, integrated jacks can be used that combine all components (plus Status LEDs) in the housing of the jack. They are available as single-channel or 2-channel models.

Hilscher commonly uses a 2-channel integrated jack that also includes the PE capacitor (C3a / C3b) and is available from Pulse Engineering or ERNI:

| | |
|---|---|
| Pulse Engineering: | J8064D628ANL |
| ERNI: | 203313 |

If one Ethernet port is required only, connect this port according to Figure 63.

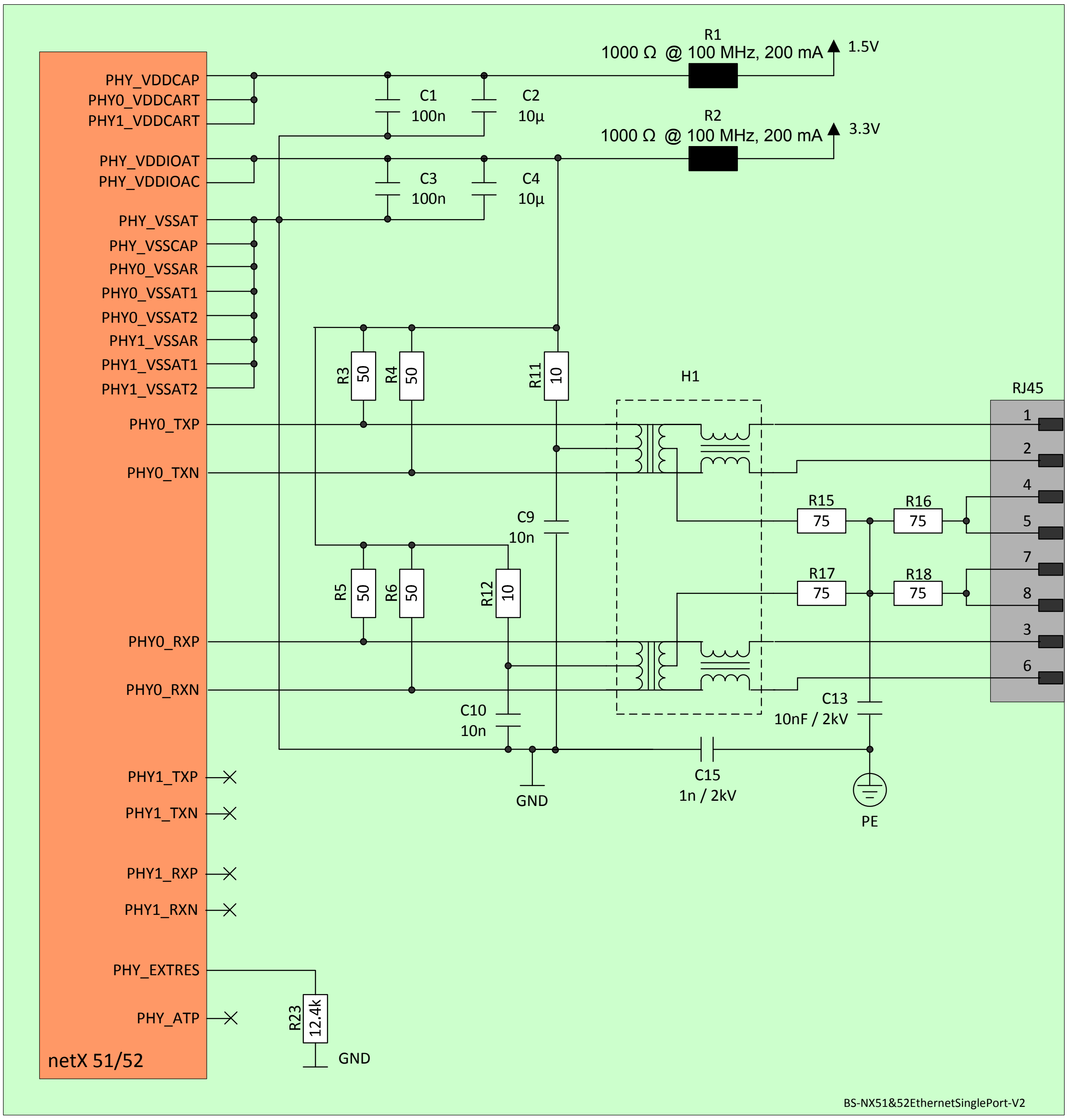


*Figure 63: netX 51/52 single-channel Ethernet circuit (Twisted pair)*

Q1: We do not stock 12.4 kΩ resistors and/or 50 Ω resistors. Can’t we use 12 kΩ/49 Ω/51 Ω instead?

A1: The specified resistor values are taken directly from the specs of the internal PHY. Using out-of-spec resistor values will result in an out-of-spec Ethernet interface for the proper and reliable function of which we cannot give a guarantee.

### 3.13.2 Fiber optic

For 100BASE-FX Ethernet communication, the netX requires external optical transceivers. Since these transceivers usually work with LVPECL (Low Voltage Positive Emitter Coupled Logic) levels, connect appropriate signal converters between netX and transceivers. Place the signal converters as close as possible to the corresponding netX pins. The traces of the differential signal lines between buffers and transceivers should provide an impedance of 50 Ω (100 Ω differential impedance). The signal lines require a Thevenin termination as shown in the following schematic.

#### 3.13.2.1 Transceivers with internal AC-termination

Some fiber optic transceivers already provide an internal AC-termination on the TXDATA input lines which results in the following circuit:

For netX fiber optic designs that are to be used with real-time Ethernet protocols, especially PROFINET, the fiber optic transceivers must be equipped with a Digital Diagnostics Monitoring Interface (DMI) providing status information. This is required for AIDA and only available at QFBR-5978AZ from Avago Technologies.

**Note:** If you develop, produce or sell products with DMI, you have to sign a contract with Siemens on a buying option for transceivers for fiber optics, Digital Diagnostic Monitoring Interface

#### 3.13.2.2 Diagnostic monitoring interface

Connecting I2C components to a common signal bus and addressing them individually via their I2C device address is usually possible. However, this is not possible for QFBR-5978AZ devices because they do not have a hardware-configurable device address. QFBR-5978AZ devices all use the same device address which tolerates other I2C components (sensors, memories, etc.), but not a second QFBR-5978AZ.

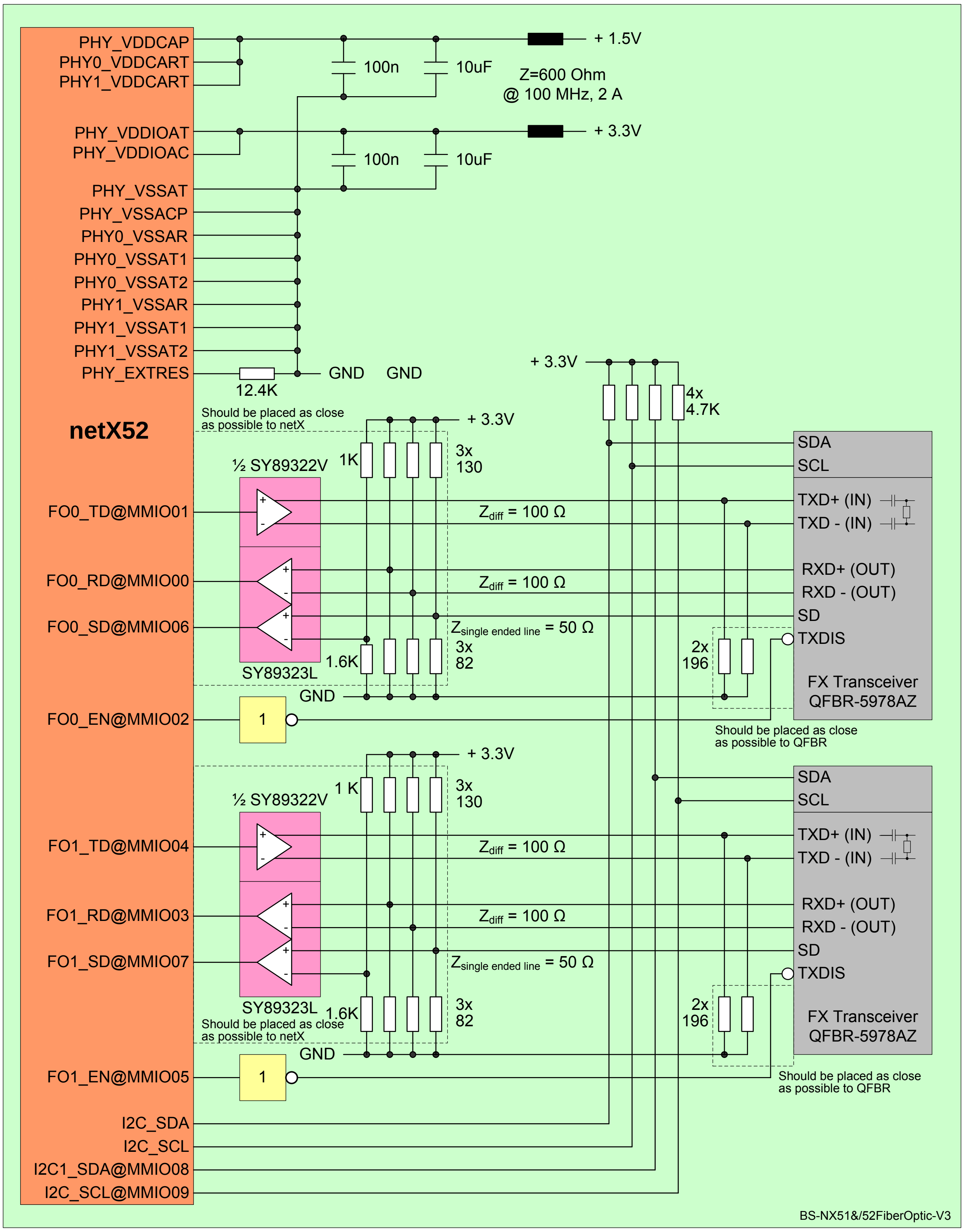


*Figure 64: netX 51/52 Ethernet circuit (Fiber optic)*

**Note:** The termination resistors shall be placed as close as possible to the end of the signal lines and connected with short traces (no stubs).

The BIAS resistors on the TXD signals (circuit for internally AC-terminated transceivers) are to be placed close to the beginning of the signal line, i.e. close to the LVTTl-to-LVPECL level translators.

For information on component placement, see the following chapter.

The LVTTL-to-LVPECL signal converters usually have thermal PADs for a proper heat dissipation. Observe the thermal design notes in the datasheet of the appropriate devices.

The power supply (+3.3 V) for the fiber optic transceivers should be filtered according to the manufacturer's recommendation (consult the datasheet of the transceiver).

### 3.13.3 Unused Ethernet PHYs

If the internal Ethernet PHYs are not used in a netX design, the signal pins (PHY0/1_RXP/RXN, PHY0/1_TXP/TXN) should simply be left open. However, all power supply pins must be connected, as well as the reference resistor, as shown in the following schematic.

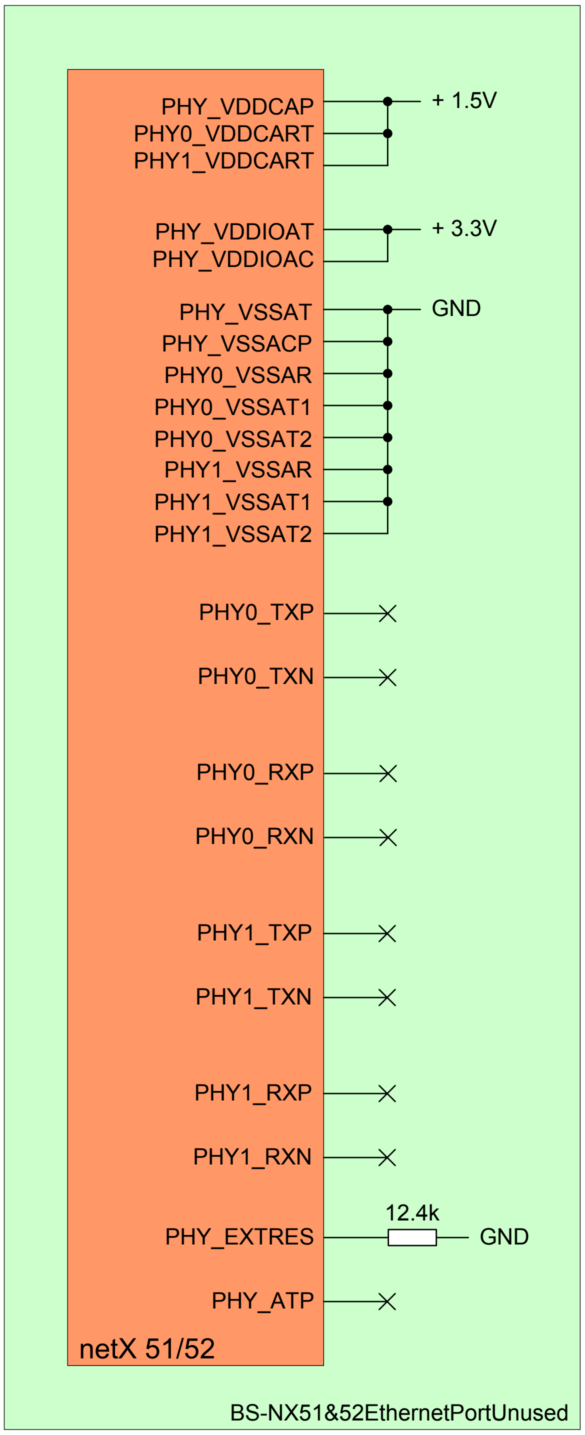


*Figure 65: netX 51/52 Ethernet circuit (PHYs not used)*

### 3.13.4 Ethernet status LEDs

#### 3.13.4.1 Ethernet communication status LEDs

Each of the netX Ethernet ports provides two status LED signals:

| | |
|---|---|
| LINKn | the link status LED is lit when a link has been established on the corresponding Ethernet port |
| ACTIVITYn | the yellow activity LED flickers when data is received or transmitted on the corresponding port |

The following table shows the standard pin assignment for the status LEDs:

| **Function** | **LED color** | **Pin name and number** | | | |
|---|---|---|---|---|---|
| | | netX 51 | | netX52 | |
| Ethernet Port 0, Link | green | XM0_IO0 @ MMIO12 | R1 | XM0_IO0 @ MMIO16 | T4 |
| Ethernet Port 0, Activity | yellow | XM0_IO1 @ MMIO13 | U1 | XM0_IO1 @ MMIO17 | V1 |
| Ethernet Port 1, Link | green | XM1_IO0 @ MMIO14 | T1 | XM1_IO0 @ MMIO18 | U4 |
| Ethernet Port 1, Activity | yellow | XM1_IO1 @ MMIO15 | V1 | XM1_IO1 @ MMIO19 | U3 |

*Table 18: Status LEDs for Ethernet ports*

netX 51/52 use the IO signals of XMAC0 and XMAC1 for Ethernet status signaling.

For operating the design with Hilscher loadable firmware, always use the schematics below.

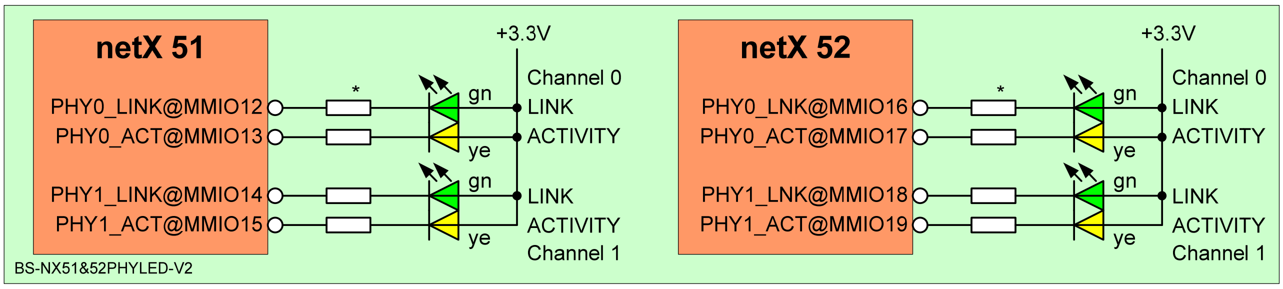


*Figure 66: netX 51/52 Ethernet status LED circuit*

**Note *:** MMIO12…15 or MMIO16…19: Adaption of the LED resistance to the brightness considering the driver strength. Usually 220 Ω to 470 Ω.

### 3.13.4.2 Real-time Ethernet protocol status LEDs

In addition to the standard Ethernet communication status LEDs, the different RTE protocols have defined up to four further LEDs.

**Note:** We absolutely recommend you to use 2 duo-color red/green LEDs. You can use them in your hardware for all major RTE systems. Our protocol stacks will properly control these LEDs so that you will pass the certification process.

The signals used for driving these LEDs (PIO0–3) are identical with the signals used for the fieldbus status LEDs on XMAC0 and XMAC1 (see section *Fieldbus status LEDs* on page 66 for pin assignment details). However, this is not really a conflict since RTE applications usually require both Ethernet ports and a fieldbus stack cannot run on XMAC0 or XMAC1 while an Ethernet application is using these XMACs.

| **Firmware** | **Label** | **Dual LED** | | | | | **Label** | **Meaning** |
|---|---|---|---|---|---|---|---|---|
| | | **COM 0** | | | **COM 1** | | | |
| | | **PIO0** | **PIO1** | | **PIO2** | **PIO3** | | |
| PROFINET | SF | | red | | | red | BF | SF: System Failure<br>BF: Bus Failure |
| EtherCAT | RUN | green | | | | red | ERR | RUN: Run<br>ERR: Error |
| POWERLINK | BS | green | | | | red | BE | BS: Bus Status<br>BE: Bus Error |
| SERCOS Slave | S | green | red | | | | - | S:<br>orange = red and green at the same time. |
| EtherNet/IP | MS | green | red | | green | red | NS | MS: Module Status<br>NS: Network Status |
| Open Modbus/TCP | RUN | green | | | | red | ERR | RUN: Run<br>ERR: Error |

*Table 19: Status LEDs for real-time Ethernet applications*

The definition of all RTE status LED I/Os is active low.

The following schematic shows how to connect the LEDs.

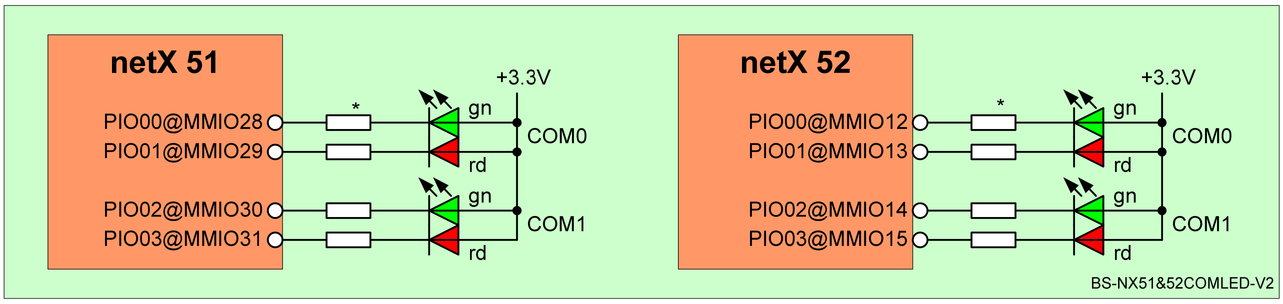


*Figure 67: netX 51/52 RTE status LED schematic*

**Note *:** MMIO28…31 or MMIO28…31: Adaption of the LED resistance to the brightness considering the driver strength. Usually 220 Ω to 470 Ω

For COM0 and COM1 indicators/light pipes, use two red/green duo LEDs or two pairs of single LEDs. Place the LEDs of each pair close to each other.

**Note:** On the netX 51/52, all PIO signals are connected to the internal MMIO Matrix, i.e. the RTE status LEDs must be connected to MMIO pins. MMIO 28–31 on netX 51 and MMIOs 12–15 on netX 52 must always be used for the connection to PIO0–3 signals, whenever the design is to be operated with Hilscher loadable firmware.

### 3.13.5 RTE device address switches

It is generally not mandatory to have address switches on a real-time Ethernet device. Without PROFINET you will find address switches on the slave devices. Therefore, we recommend having two rotary switches ranging from 00 to FF on the device. This gives you an address range from 01 to 159 and many unused additional addresses. These addresses can be used for special functions e.g. for resetting the configuration to the default setting of the manufacturer. All addresses and special functions will be read directly after power-on reset and will be saved internally or started directly.

| Real-time Ethernet | Type of address | Address range | | Special functions or reserved |
|---|---|---|---|---|
| | | Switch | Value | |
| EtherCAT | Device address | | | 00 |
| EtherNet/IP | Last part of the IP address | 01 – 09 | 01 – 09 | 0A – 0F |
| Modbus/TCP | Last part of the IP address | 10 – 19 | 10 – 19 | 1A – 1F |
| POWERLINK | Node ID | 20 – 29 | 20 – 29 | 2A – 2F |
| PROFINET | not used | 30 – 39 | 30 – 39 | 3A – 3F |
| Sercos | Device address | 40 – 49 | 40 – 49 | 4A – 4F |
| | | 50 – 59 | 50 – 59 | 5A – 5F |
| | | 60 – 69 | 60 – 69 | 6A – 6F |
| | | 70 – 79 | 70 – 79 | 7A – 7F |
| | | 80 – 89 | 80 – 89 | 8A – 8F |
| | | 90 – 99 | 90 – 99 | 9A – 9F |
| | | A0 – A9 | 100 – 109 | AA – AF |
| | | B0 – B9 | 110 – 119 | BA – BF |
| | | C0 – C9 | 120 – 129 | CA – CF |
| | | D0 – D9 | 130 – 139 | DA – DF |
| | | E0 – E9 | 140 – 149 | EA – EF |
| | | F0 – F9 | 150 – 159 | FA – FF |

*Table 20: RTE device address switches (overview)*

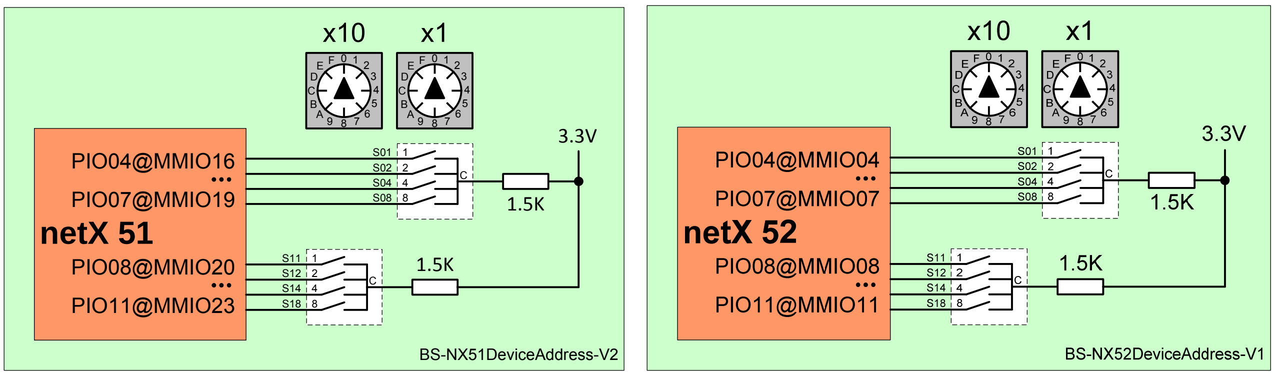


*Figure 68: RTE device address switches*

**Note:** This feature is not supported in our standard protocol stack, but its implementation is planned in the near future.
netX 52 has less MMIO pins. If signals overlap, i.e. sync signals, you have to use other MMIOs or the HIF pins.

### 3.13.6 Real-time Ethernet synchronization signals

Apart from running standard Ethernet protocols, the netX can run all current real-time Ethernet (RTE) protocols. The physical interface for RTE protocols is identical to the standard Ethernet port as described in the preceding chapters, except for some additional synchronization signals that may be required depending on protocol and application. Both are described in the following sections.

For RTE synchronization purposes netX 51/52 provide four additional signals, two input and two output signals.

| RTE protocol | Master / Slave | Sync signal | Function | Type | Remarks |
|---|---|---|---|---|---|
| Sercos | Slave | XC_SAMPLE0 @ MMIO08 | - | In | - |
| | | XC_SAMPLE1 @ MMIO09 | - | In | - |
| | | XC_TRIGGER0 @ MMIO10 | CON_CLK | Out | Configurable |
| | | XC_TRIGGER1 @ MMIO11 | DIV_CLK | Out | Configurable |
| | Master | XC_SAMPLE0 @ MMIO08 | - | In | - |
| | | XC_SAMPLE1 @ MMIO09 | - | In | - |
| | | XC_TRIGGER0 @ MMIO10 | CON_CLK | Out | Configurable |
| | | XC_TRIGGER1 @ MMIO11 | - | Out | - |
| EtherCAT | Slave | XC_SAMPLE0 @ MMIO08 | Latch 0 | In | - |
| | | XC_SAMPLE1 @ MMIO09 | Latch 1 | In | - |
| | | XC_TRIGGER0 @ MMIO10 | Sync 0 | Out | - |
| | | XC_TRIGGER1 @ MMIO11 | Sync 1 | Out | - |
| EtherNet/IP | Slave | XC_SAMPLE0 @ MMIO08 | - | In | - |
| | | XC_SAMPLE1 @ MMIO09 | - | In | - |
| | | XC_TRIGGER0 @ MMIO10 | TimeSync | Out | Configurable |
| | | XC_TRIGGER1 @ MMIO11 | - | Out | - |
| PROFINET IRT | IO-Controller / IO-Device | XC_SAMPLE0 @ MMIO08 | - | In | - |
| | | XC_SAMPLE1 @ MMIO09 | - | In | - |
| | | XC_TRIGGER0 @ MMIO10 | IO_Output | Out | Trigger for outputs valid, also used for certification (start of red phase) |
| | | XC_TRIGGER1 @ MMIO11 | IO_Output | Out | Trigger for sample inputs |
| POWERLINK | Controlled Node | XC_SAMPLE0 @ MMIO08 | - | In | - |
| | | XC_SAMPLE1 @ MMIO09 | - | In | - |
| | | XC_TRIGGER0 @ MMIO10 | SoC | Out | Configurable “Start of Cycle” |
| | | XC_TRIGGER1 @ MMIO11 | - | Out | - |

*Table 21: Additional RTE sync signals netX 51/52*

### 3.13.7 Special real-time Ethernet requirements

#### 3.13.7.1 Fast StartUp – Special feature of PROFINET

Designs with netX 51/52 that run PROFINET and require the Fast StartUp feature have to use a QSPI Flash as described in section *QSPI Flash* on page 28 and a reset circuit with a signal delay of less than 66 ms as described in section *Power-on reset and reset in* on page 19.

#### 3.13.7.2 QuickConnect$^{TM}$ – Special feature of EtherNet/IP

Designs with netX 51/52 that run Ethernet/IP and require the QuickConnect$^{TM}$ feature have to use a QSPI Flash as described in chapter *QSPI Flash* page 28 and a reset circuit with a signal delay of less than 66 ms as described in section *Power-on reset and reset in* on page 19.

# 4 Debug and test interfaces

## 4.1 JTAG interface

netX 51/52 are equipped with a standardized JTAG Interface that allows loading, flashing and starting firmware, debugging of firmware and provides access to the Boundary Scan Test mode of the chips. Though this interface is rarely used during the operation of final netX products, it is strongly recommended to have at least retrofittable access to this interface on any netX design, especially on prototypes, even on designs on which the customer intends to use Hilscher loadable firmware instead of writing his own software. The debugging of a prototype can be somewhere between cumbersome and impossible if the JTAG interface is not accessible. For developing your own netX software, a JTAG interface is essential anyway. Furthermore, automatic testing systems used in production testing usually need access to this interface, so the JTAG interface pins should at least be connected to test points that can be accessed by prober systems.

Figure 69 shows the standard netX JTAG circuit.

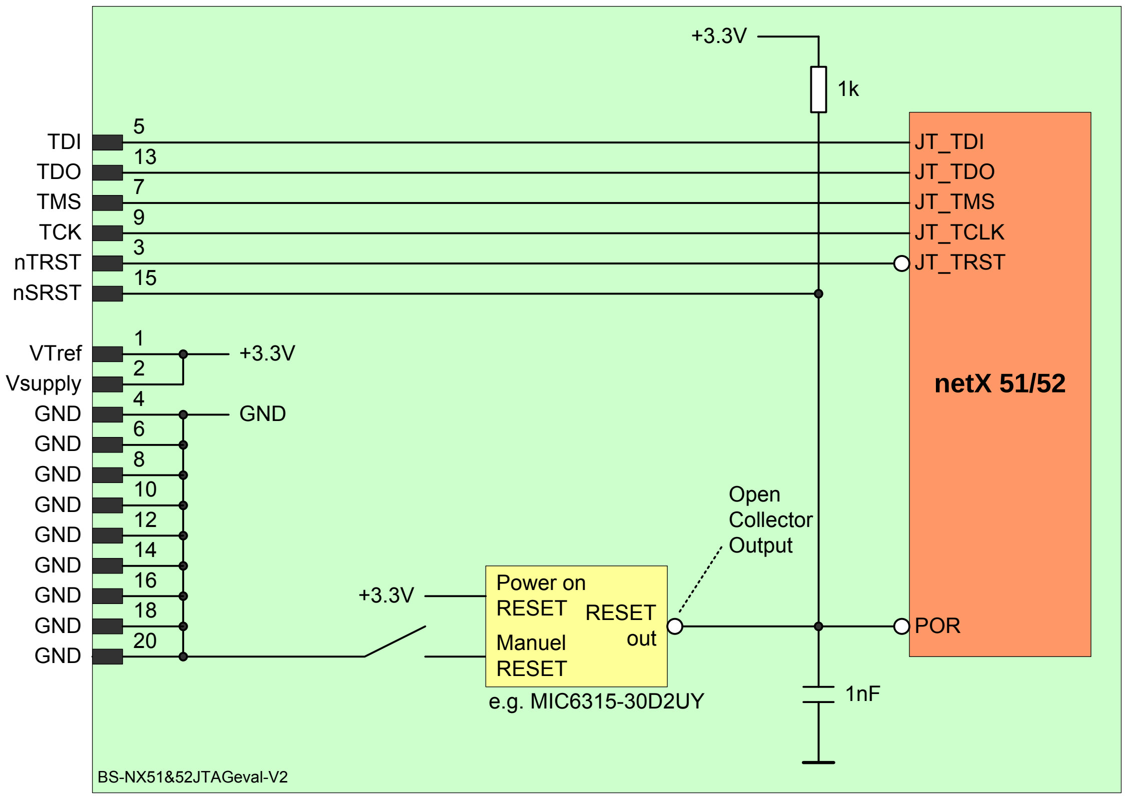


*Figure 69: netX 51/52 JTAG circuits*

Whenever there is enough space on the PCB of the design, a standard 20-pin shrouded header with 0.1” or 2.54 mm pitch should be used for the JTAG interface, since this allows connecting standard JTAG debugging units to be plugged to the board instantly without the need for any special cable adapters.

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 | DBGRQ | Not used | | 18 | GND | VSS |
| 19 | DBGACK | Not used | | 20 | GND | VSS |

*Table 22: 20-pin JTAG connector pin assignment*

In designs that do not require the use of the JTAG interface, the JTAG signals may be left unconnected. The internal pull-down on the JT_TRSTn will then constantly hold the JTAG interface in reset state.

## 4.1.1 Hilscher's 'mini-JTAG' connector

For netX products with small board size that do not allow the implementation of the standard 20-pin JTAG connector, but still require 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 adapter. The implementation of the connector defined by Hilscher relieves the user from defining and building such an adapter because an appropriate Hilscher adapter for the "mini-JTAG" connector is already available.

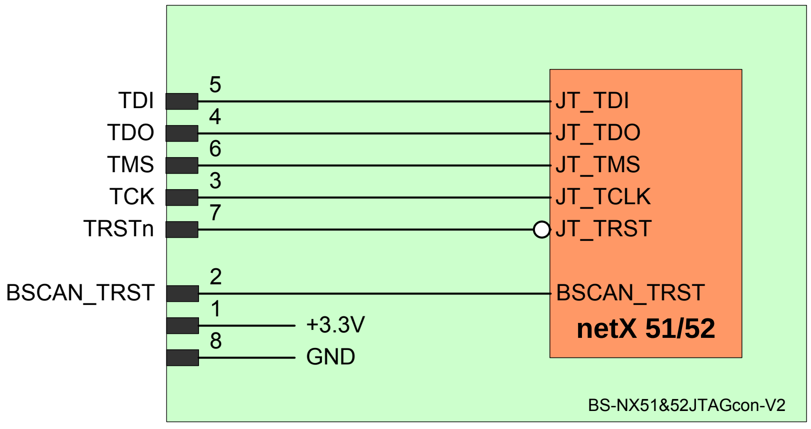


*Figure 70: "mini-JTAG" connector for debugging and Boundary Scan Test*

**Note:** The "mini-JTAG" connector does not allow the debugger to reset the netX target because of the missing power-on reset (PORn/nSRST) signal. The intention was to have a small low-cost connector for debugging, not for software development.

This adapter provides a standard 20-pin JTAG connector to be used with common debugger devices and a flex cable for connection to the “mini-JTAG” port.

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

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

*Table 23: ZIF connector for "mini-JTAG"*

Connector pin assignment:

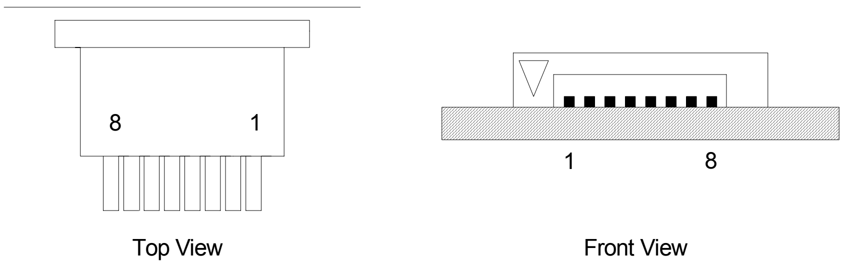
| Pin | Signal name | |
|---|---|---|
| 1 | +3V3 |  |
| 2 | BSCAN_TRST | |
| 3 | TCK | |
| 4 | TDO | |
| 5 | TDI | |
| 6 | TMS | |
| 7 | TRSTn | |
| 8 | GND | |

*Table 24: "mini-JTAG" ZIF pin assignment*

## 4.2 ETM interface

The ETM interface (Embedded Trace Macro cell) provided by the internal ARM CPU of the netX 51 extends the debugging capabilities provided by the JTAG interface. We generally recommend implementing the ETM interface when building netX evaluation boards, but most likely it will not be necessary for debugging prototype hardware. Even the 38-pin ETM board connectors (AMP Mictor 2-767004-2) are costly, let alone appropriate ETM debugging units. Thus, implementing an ETM interface is usually only of interest to customers who write their own software for their netX design.

The ETM connector is standardized by ARM. For more information, see www.arm.com. It is highly recommended to implement the ETM interface accordingly, otherwise the debug tools will not work correctly.

The following table shows how to wire the ETM connector with the netX.

| Pin | ARM signal | netX signal | netX 51 pin | Pin | ARM signal | netX signal | netX 51 pin |
|---|---|---|---|---|---|---|---|
| 1 | N.C. | | | 2 | N.C. | | |
| 3 | N.C. | | | 4 | N.C. | | |
| 5 | GND | VSS | | 6 | TRACECLK | ETM_TCLK | V2 |
| 7 | DBGRQ | ETM_DRQ | T6 | 8 | DBGACK | ETM_DACK | V6 |
| 9 | nSRST | Not used | | 10 | EXTTRIG | | |
| 11 | TDO | JT_TDO | L5 | 12 | VTRef | VCCIO | |
| 13 | RTCK | Not used | | 14 | VCC | VCCIO | |
| 15 | TCK | JT_TCLK | K5 | 16 | TRACEPKT[7] | ETM_TPKT07 | U11 |
| 17 | TMS | JT_TMS | K6 | 18 | TRACEPKT[6] | ETM_TPKT06 | T10 |
| 19 | TDI | JT_TDI | L6 | 20 | TRACEPKT[5] | ETM_TPKT05 | U10 |
| 21 | nTRST | JT_TRSTn | J5 | 22 | TRACEPKT[4] | ETM_TPKT04 | T9 |
| 23 | TRACEPKT[15] | ETM_TPKT15 | V14 | 24 | TRACEPKT[3] | ETM_TPKT03 | U9 |
| 25 | TRACEPKT[14] | ETM_TPKT14 | T13 | 26 | TRACEPKT[2] | ETM_TPKT02 | V9 |
| 27 | TRACEPKT[13] | ETM_TPKT13 | U13 | 28 | TRACEPKT[1] | ETM_TPKT01 | R8 |
| 29 | TRACEPKT[12] | ETM_TPKT12 | V13 | 30 | TRACEPKT[0] | ETM_TPKT00 | U7 |
| 31 | TRACEPKT[11] | ETM_TPKT11 | T12 | 32 | TRACESYNC | ETM_TSYNC | V3 |
| 33 | TRACEPKT[10] | ETM_TPKT10 | U12 | 34 | PIPESTAT[2] | ETM_PSTAT2 | V7 |
| 35 | TRACEPKT[9] | ETM_TPKT09 | T11 | 36 | PIPESTAT[1] | ETM_PSTAT1 | T7 |
| 37 | TRACEPKT[8] | ETM_TPKT08 | V12 | 38 | PIPESTAT[0] | ETM_PSTAT0 | U6 |

*Table 25: ETM signals*

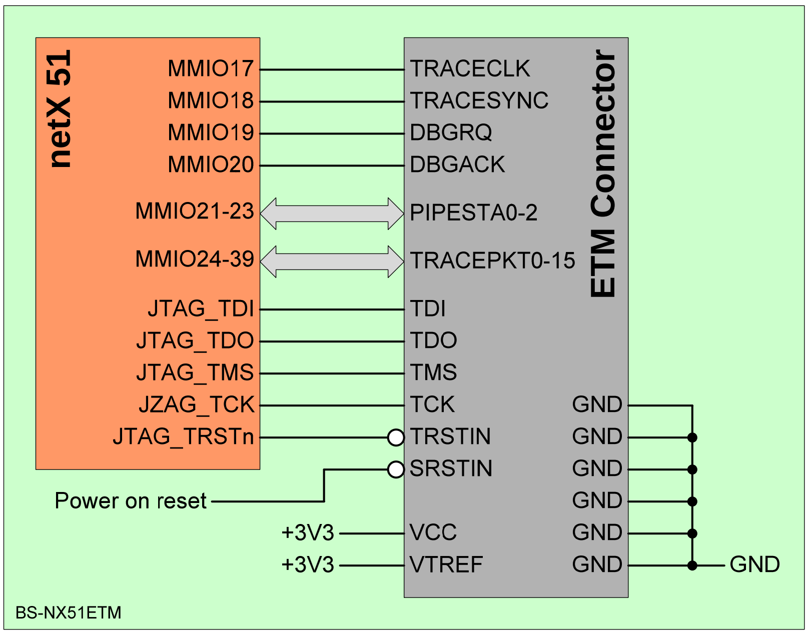


*Figure 71: ETM interface*

**Note:** In the center of the AMP Mictor connector there are four additional through-hole pins which have to be grounded for proper operation of the trace port!
For the PCB layout it is recommended to keep the lines for the ETM signals as short as possible. The signal delay should be < 100 ps. 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 (<10 mm)) to match the output impedance of the chip signal driver with the PCB trace impedance.

**Note:** For netX 51 ETM signals are shared signals which can be mapped to MMIO signal pins. While using ETM, you cannot use any other signals on the MMIO pin at the same time.

**Note:** netX 52 can be used with JTAG only.

## 4.3 Boundary scan test

For automated production testing of a final netX products, it may be desirable to make use of the netX Boundary Scan Test feature. All necessary signals are available at the ‘mini-JTAG connector. For mass production it makes sense to have this connector as eight pads on the bottom side of the PCB to access these signals via testing needles.

For entering the Boundary Scan Test mode, the test equipment has to pull the signal at BSCAN_TRST to high level.

# 5 Resource overview

The following tables list netX hardware resources and functions and provide information on existing software support for these features. “No driver available” means that Hilscher currently does not provide a driver or any special functions for easy access to the corresponding resource. However, this resource may still be used if the user himself develops the appropriate code or integrates third party products (e.g. Flash file system for parallel Flash).

## 5.1 rcX kernel

| Resource/functionality | Loadable firmware (LFW) | Linkable object modules (LOM) |
|---|---|---|
| USB device | for firmware update | for firmware update / debug |
| UART0 | for firmware update | for firmware update / debug |
| UART1 | not supported | supported |
| UART2 | not supported | supported |
| SDRAM | min. 8 MB required | min. 8 MB required for standard application |
| Security EEPROM | Master: mandatory<br>Slave: optional | Master: mandatory<br>Slave: optional |
| SPI Flash | Required (min. size see Table 27) | Required (min. size see Table 27) |
| Quad SPI | - requires second stage boot loader<br>- for loading FW only (FSU) | - requires second stage boot loader<br>- for loading FW only (FSU) |
| Second SPI interface | not supported | - no Flash file system<br>- only limited components |
| MMC/SD card | not supported | no driver available |
| Parallel Flash | not supported | - no Flash file system<br>- only limited components |
| Fieldbus slave (1 channel) | see Table 27 | see Table 27 |
| Ethernet ports | real-time Ethernet protocols (see separate list) | Standard Ethernet and real-time Ethernet protocols (see separate list) |
| Host interface | Dual-port memory interface | Dual-port memory interface, extension bus |
| Gateway functionality | not supported | user programmable |
| 2 Channel fieldbus | not supported | supported |
| I/O-Link | not supported | no driver available |

*Table 26: List of resources – rcX kernel*

## 5.2 Memory requirements of Hilscher stacks

The following table lists all fieldbus and RTE protocols that are currently (status of August 2015) available as loadable firmware from Hilscher, along with the required size of the SPI Flash holding the appropriate firmware.

The Flash size information

- is based on the code sizes of the current firmware releases
- includes the additional memory required for the Hilscher's second stage boot loader (currently 52 KB) and the Flash disk
- leaves some headroom for future extensions
- is generally rounded up to the next available Flash size step

If linkable object modules are to be used, users must provide the additional memory required for their user application (where applicable) and additional functionality e.g. integrating a Web server.

By now, the smallest available SDRAM components have a min. memory size of 8 MB which meets the requirements of all current protocols and netX 51 designs.

netX 52 designs using loadable firmware need no external SDRAM.

| **Protocol** | **Size** |
|---|---|
| CANopen Slave | 350 KB |
| DeviceNet Slave | 300 KB |
| CC-Link Slave | 250 KB |
| PROFIBUS Slave | 300 KB |
| EtherCAT Slave | 500 KB |
| EtherNet/IP Adapter | 550 KB |
| Open Modbus/TCP | 450 KB |
| POWERLINK Controlled Node (with integrated hub) | in preparation |
| PROFINET RT/IRT IO-Device (with integrated switch) | 750 KB |

*Table 27: Flash sizes for loadable firmware*

**Note:** We recommend you to choose a Flash of min. 4 MB since some stacks have to store remanent data in addition. A stack with remanent data together with the second stage boot loader requires about 1.5 MB. If you want to update a firmware, the firmware will be transferred completely into the Flash before the old firmware is deleted. 2 MB might not be enough for that.

# 6 General design considerations

## 6.1 Thermal behavior

### 6.1.1 Basics

Since netX 51/52 designs are often used in industrial environments, suitability for high temperature ranges is a frequent issue. Depending on the interfaces and chip type used, the power dissipation of netX 51/52 designs may range from 0.8W (netX 51/52 with fieldbus interface) to 2.15W (netX 51/52 with 2 Port Full Duplex Ethernet and 1 MAC (with external PHY), SDRAM). This leads to an appropriate warming of the netX silicon. Although the chip's junction temperature is limited to 125°C, higher temperatures will cause malfunction and permanent damage. Thus, it is always desirable to keep the junction temperature as low as possible because a semiconductor's statistical lifetime generally decreases with rising temperature. In order not to nullify the gained reduction of power dissipation (by the formation of additional heat that needs to be dissipated), hardware designers should not use the possible headroom (3.6 V I/O and 1.65 V core), but make sure that all netX power supply circuits deliver nominal voltage levels (3.3 V I/O and 1.5 V core).

BGA packages used with all current netX chips mainly offer the heat two ways to dissipate:

1. through the package balls into the copper of the PCB (mainly power and ground planes)
2. from the chip surface (top) to the environment

The resulting thermal resistance of the first path strongly depends on the characteristics of the PCB, i.e. the thermal behavior of a design also strongly depends on the PCB. The only possibility to decrease the influence of the PCB is to use a heat sink. This can considerably reduce the thermal resistance of the second heat dissipation path and improve the overall thermal behavior of the design.

### 6.1.2 Estimates

The Technical Data Reference Guides (chapter "Thermal package specification") of the netX chips provide the following formula that allows the calculation of the chip junction temperature ($T_j$) at a given environment temperature ($T_a$), power ($P_{netX}$) and thermal resistance ($R_{th}$) of the heat sink:

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

You will find the chip-specific value of $\theta_{ic}$ in the above-mentioned chapter of the respective Technical Data Reference Guide.

**Note:** The formula above only allows you to estimate the possible junction temperature of a particular design, as there is still an influence of the PCB characteristics. The parameters have been evaluated using certain test boards and may thus not be applied directly to a specific design. This applies even more to the second formula, for operation without heat sink!

### 6.1.3 Recommendations

Hilscher netX hardware is usually designed for a max. junction temperature of approx. 100°C, the recommended value for customer designs. The FIT-rate (FIT = Failure in Time, see the respective chapter of the Technical Data Reference Guide) for the silicon process the netX are based on, shows a significant rise in temperature between 100 °C and the absolute max. junction temperature of 125 °C, which was the reason for choosing the 100 °C.

However, since the absolute max. junction temperature is 125°C, it is at a device manufacturer's discretion to follow this recommendation or to accept a higher junction temperature and trade in a decrease of the MTBF of his devices for a higher temperature range specification.

A major factor of the thermal behavior of a netX design is the size of the PCB. Design experience at Hilscher shows that designs with a power density of more than 0.15 W/cm$^2$ are critical (assuming a junction temperature limit of 100 °C and a max. ambient temperature of 70 °C and considering the common max. temperatures of peripheral components like SDRAM, FLASH, reset generator, etc.). The board size should be chosen accordingly. To calculate the power density of a design, simply divide the power dissipation by the area of the PCB.

Two examples of the Hilscher netX product line are described below.

Example 1 is well within the power density limit, example 2 is at the limit:

**Example 1: Mini PCI card with Ethernet**

| | |
|---|---|
| Dimensions: | 44.6 mm x 59.8 mm |
| Area: | 26.67 cm² |
| Power consumption: | 1.75 W, heat sink, 70 °C |
| Power density: | 0.07 W/cm² |

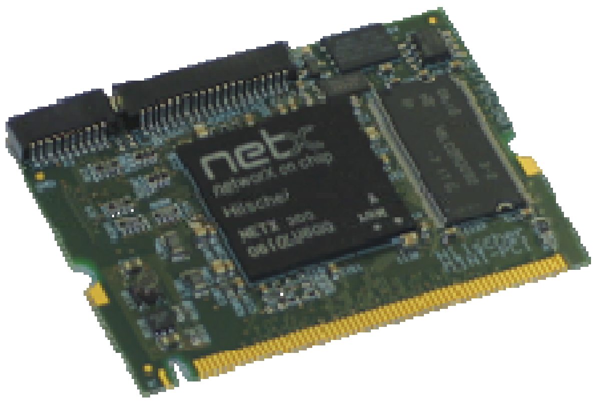


*Figure 72: CIFX 90-RE*

**Example 2: netIC with Ethernet**

| | |
|---|---|
| Dimensions: | 21.0 mm x 42.0 mm |
| Area: | 8.82 cm² |
| Power consumption: | 1.3 W, heat sink, 70 °C |
| Power density | 0.15 W/cm² |

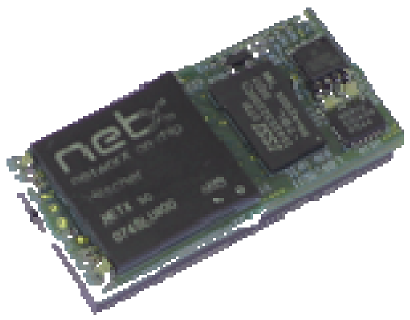


*Figure 73: NIC 50-RE*

### 6.1.4 Rules of thumb

Assuming the recommended junction temperature limit of 100 °C, we can provide the following rules of thumb based on Hilscher's design experience with netX chips:

- Design areas of 45 x 60 mm with heat sink usually work up to 70 °C.
- Design areas of 45 x 60 mm without heat sink usually work up to 55 °C with parts covering a temperature range from -40 °C to +85 °C.
- Design areas of 25 x 45 mm without heat sink usually work up to 70 °C with parts covering a temperature range from -40 °C to +105 °C.
- Using internal PHYs, the netX temperature rises by approx. 15 °C.
- Using the heat sinks recommended by Hilscher, the max. temperature of the netX case decreases by approx. 15 °C (netX 51), 0.5 m/s air flow.
- In case of very small designs and the use of netX 52, a heat sink will improve the temperature behavior only slightly (≤ 5 °C).
- The above rules assume an unimpeded convection of the PCB. If a small closed cabinet is used, the max. ambient temperature (inside the cabinet) must be decreased by approx. 15 °C.
- If the power density exceeds 0.15 W/cm², it becomes critical to make a netX design which operates up to 70 °C. For this purpose you have to use automotive parts covering a temperature range from -40 °C to +105 °C.
- Avoid placing semiconductor components on the bottom side of the PCB within the netX chip area. If resistors or ceramic capacitors allow operating temperatures of up to 100 °C (X7R ceramic), they may be placed under the netX.

## 6.2 EMC behavior

EMC design is a quite complex issue: Countless pages in countless books and papers have already been filled with descriptions, but it is not the intention of this document to further enrich this variety of publications. Nevertheless, we would like to give you some basic instructions and useful hints for the PCB designer routing a netX design.

### 6.2.1 Layer stack

We recommend to using a 6-layer board (4 signal layers, one power and one ground plane layer) although netX designs using 4-layer PCBs are possible depending on the complexity of the design. 4 signal layers provide enough space to keep the power and ground planes free from any signal traces and allow shielding areas on top and bottom layers, which contributes to a satisfying EMC behavior of the design. The following figure shows an approved 6-layer stack for netX designs:

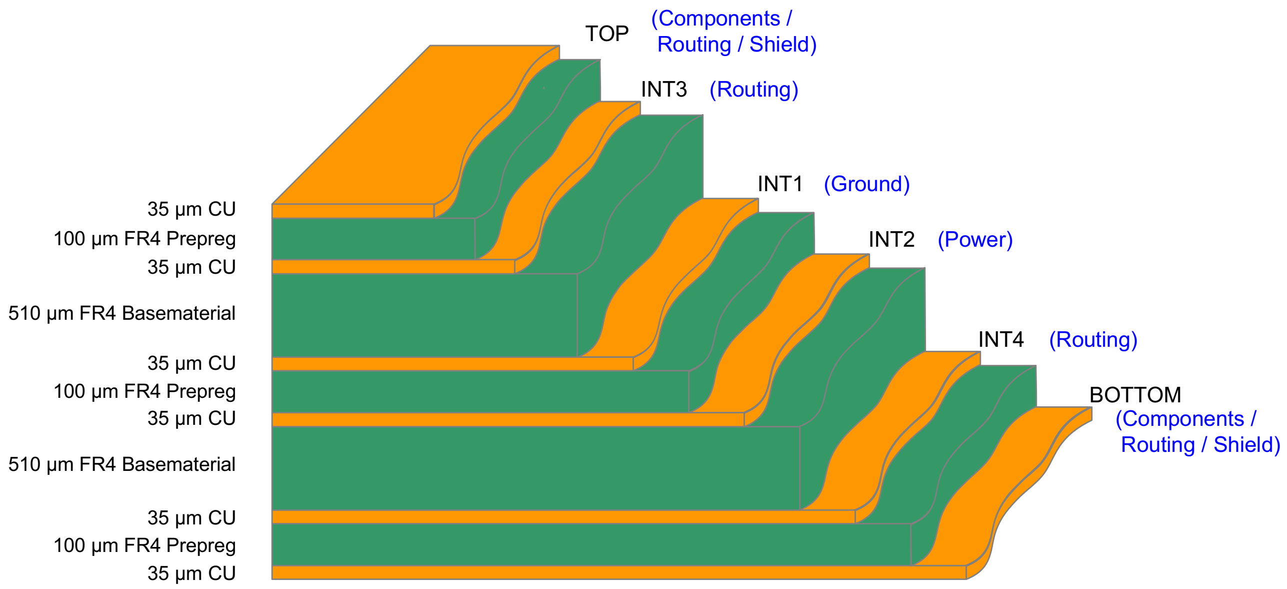


*Figure 74: Approved netX PCB layer stack*

### 6.2.2 Decoupling capacitors

As with any digital design, the use of a sufficient number of decoupling capacitors is important to provide a stable operation of the design and to avoid unnecessary emission. The following picture shows an example of how to arrange decoupling capacitors around a netX 51/52.

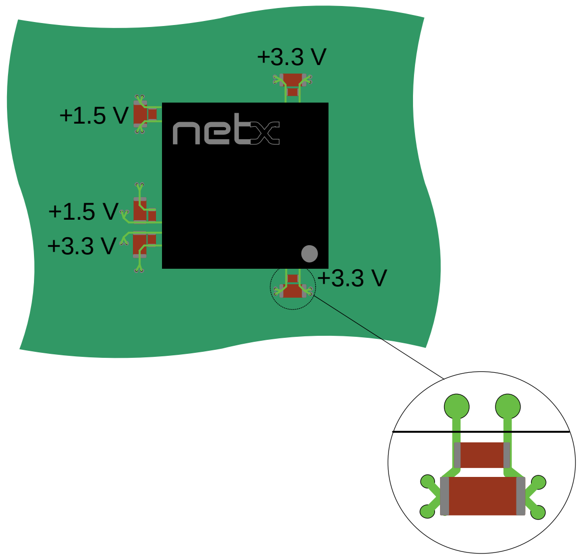


*Figure 75: netX 51/52 decoupling caps*

As shown in the figure, the power path goes from vias (located as close as possible to the caps) to the caps and from there directly to the netX power pins. However, only power pins on the two outer BGA rings should be connected that way. All inner power pins should be connected to the plane directly.

The best locations for the decoupling caps are on the bottom side (opposite) of the PCB, close to the power supply pins, as only a few of these pins are located on the outer BGA ring. If a double-sided mounting of components is not an option, place the decoupling capacitors close to and around the netX 51/52, connect the netX power and ground pins to the power and ground planes, connect the caps separately and always keep the traces as short as possible.

### 6.2.3 Reset lines

As already mentioned in section *Power-on reset and reset in* on page 19, reset signal lines should be kept as short as possible and should be equipped with a 1 nF ceramic capacitor (connected to the reset signal and ground). Reset signal lines should be located close to the netX reset input pin to reduce the risk of undesired resets due to noise or electrostatic discharge.

### 6.2.4 Clock circuits

Any oscillator pins on the netX chips are located on the outer BGA ball rings allowing you to keep the traces to a quartz crystal as short as possible.

The following picture shows a recommendation for placing and routing the oscillator components:

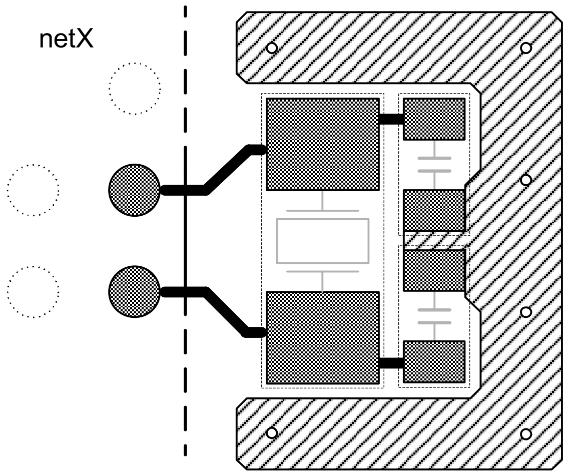


*Figure 76: Oscillator circuit with ground shield*

### 6.2.5 Ethernet interface

When routing the two signal pairs TXP/TXN and RXP/RXN of each netX Ethernet channel, some special requirements have to be fulfilled:

- Each signal pair must be routed as a separate pair of traces which should be kept as short as possible (place magnetics and termination components as close as possible to the netX).
- Traces of a pair must be routed adjacent to each other with constant spacing and equal length.
- The distance between signal pairs should be at least 5 times the spacing of the pair traces.
- Traces must be impedance-controlled maintaining a differential impedance of 100 Ohm.
- Minimize layer changes. If a layer change is inevitable, change layer with both traces at equal distance from start of trace. Avoid changing to layers that use a different reference plane.
- Avoid connectors in the signal traces. The traces should begin and end on the same PCB. If a connector is inevitable, use impedance-controlled connectors to minimize any discontinuities in trace impedance.
- Areas where Ethernet signals are routed should be free from any other signals in adjacent layers. Should it be necessary to route other signals in the Ethernet area, only use layers that are separated from the Ethernet signal layer(s) by a power or ground plane.
- Use the schematic from chapter 3.10.1 along with the recommended components.

The following pictures show two examples of setups that keep the differential impedance of the signal pair around 100 Ohm:

a) Edge-Coupled Surface Micro strip

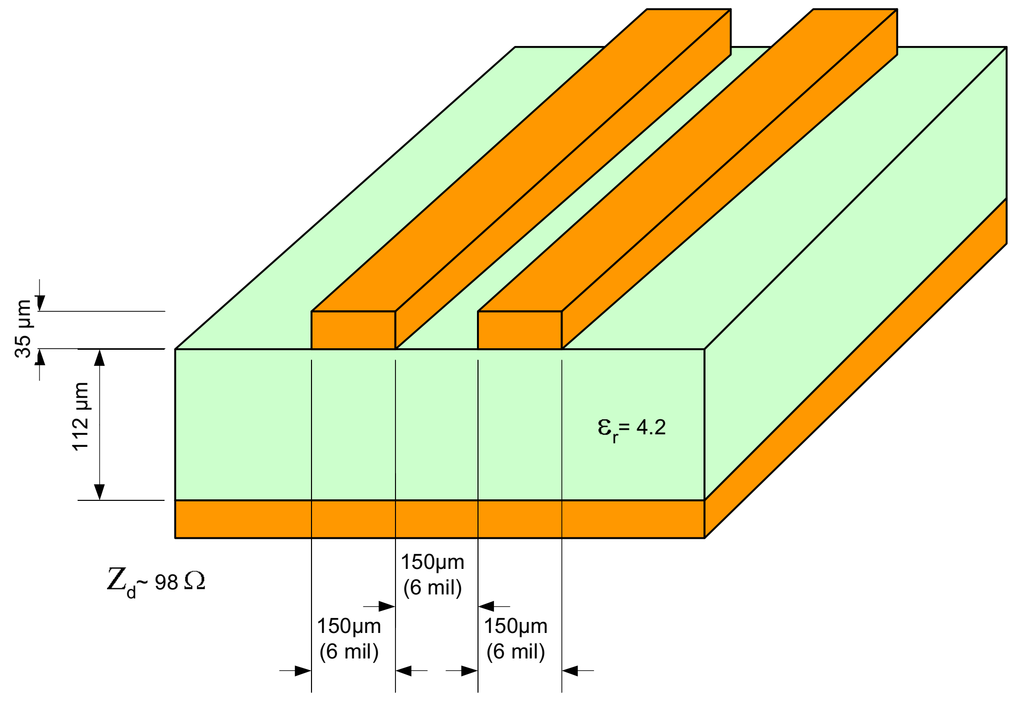


*Figure 77: Edge-Coupled Sourface Micro Strip*

To improve shielding the Ethernet traces can be routed on an inner layer using part of the top layer as shield:

b) Edge-Coupled Offset Strip line

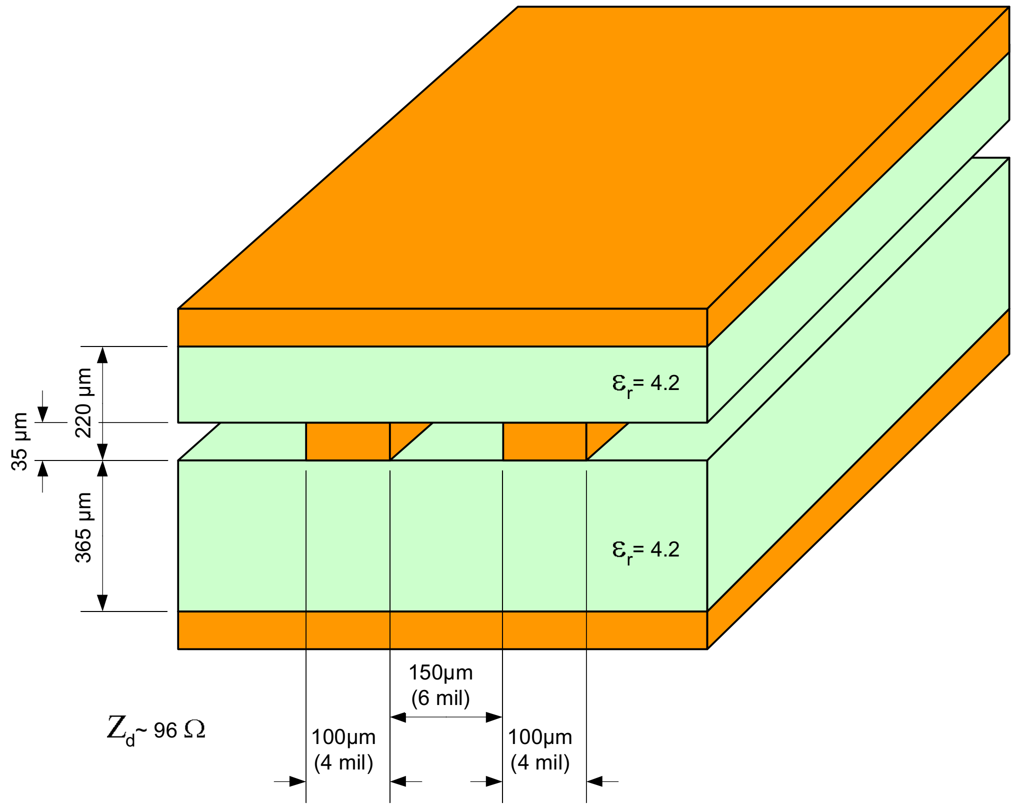


*Figure 78: Edge-Coupled Offset Strip Line*

## 6.2.6 Memory bus

When connecting SDRAM and/or parallel FLASH/SRAM etc., route the connection in a bus structure (no tree) with the bus starting at the netX, as shown in the following picture. The bus should be as short as possible and the length of the SDRAM clock signal trace should match the length of the longest SDRAM signal trace.

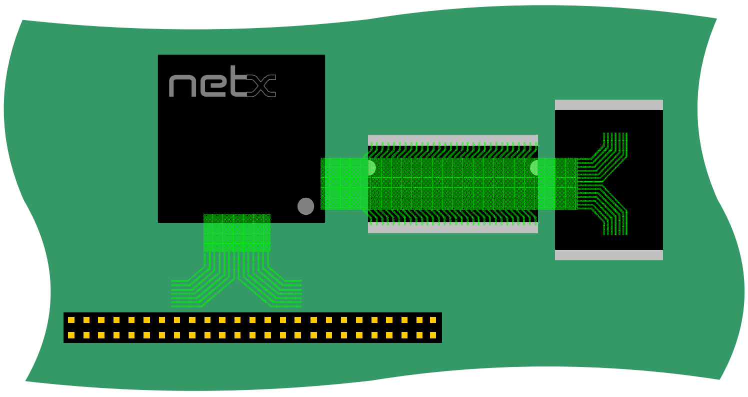


*Figure 79: netX Memory Bus*

### 6.2.7 Planes

When routing signal lines, make sure they run over a contiguous return path (power or ground plane) max. 2 layers away without being interrupted by large gaps, as this always increases emission. If you cannot avoid splitting planes, keep the traces well within the plane area. Do not route them at the edge or even outside the plane, as shown below:

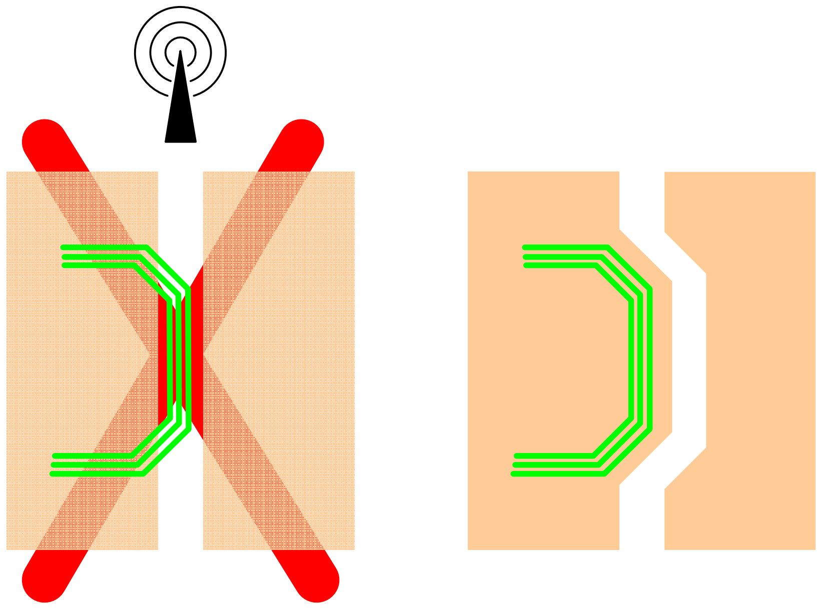


*Figure 80: Routing Example*

## 6.2.8 VIAs and signal fan out under netX 51/52

Coming in a PBGA package, the netX requires the use of small traces and vias on the PCB. Designs using external memory (SDRAM or PFlash) need 4 signal layers along with 6 mil standard technology for fanning out all signals. Designs using internal RAM only, might be realized with two signal layers as the SDRAM signal balls reside on the inner ball rings of the package. Standard "dog bone style" routing as shown below can be used to fan out the netX signals.

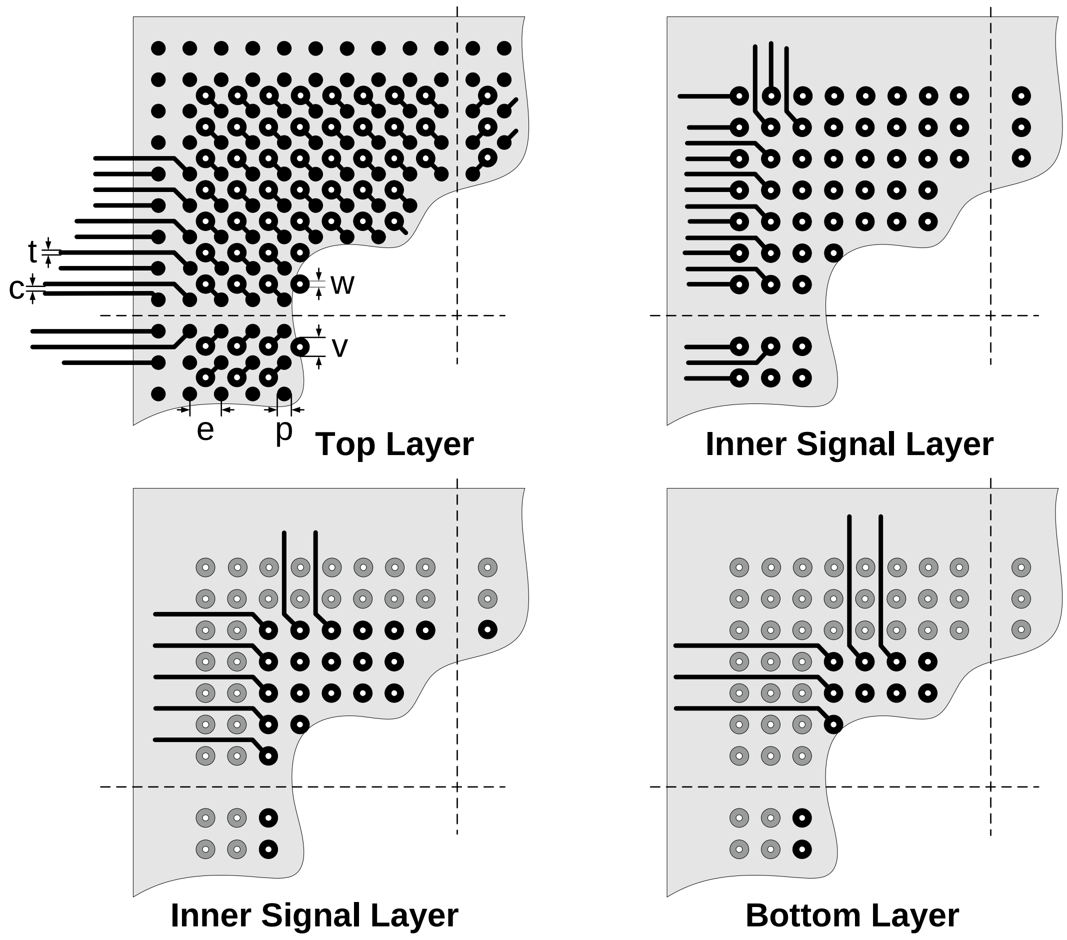


*Figure 81: VIA position and signal fan out under netX 51/52*

| Dimension | Description | netX 51 | | netX 52 | |
|---|---|---|---|---|---|
| | | mm | mil | mm | mil |
| c | Clearance | 0.15 | 6 | 0.1 | 4 |
| e | Pitch | 1.00 | 39.37 | 0.8 | 31.5 |
| p | Pad | 0.45 | 18 | 0.45 | 18 |
| t | Trace width | 0.15 | 6 | 0.1 | 4 |
| v | Via diameter | 0.50 | 20 | 0.4 | 16 |
| w | Drill hole | 0.20 | 8 | 0.15 | 6 |

*Table 28: VIA dimensions*

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

# 7 Reference section

The following chapters list some key components for netX 51/12 designs that have been successfully evaluated by Hilscher and, where applicable, are supported by Hilscher tools.

## 7.1 Crystals

| Manufacturer | Part number |
|---|---|
| Jauch | Q25,0-JXS22-12-10/20-T1-FU-LF |
| Abracon | ABM7-25.000MHZ-D2Y-T |

*Table 29: Reference crystals*

## 7.2 Serial Flash memory / EEPROM

| Manufacturer | Device | Type | Size | Frequency | Temperature | Housing |
|---|---|---|---|---|---|---|
| Adesto | AT45DB321E | SPI | 4 MB | 85 MHz | -40…+85 °C | 8P/SOIC, 8P/DFN, 8P/UBGA |
| Macronix | MX25L3235E | QSPI | 4 MB | 104 MHz | -40…+85 °C | 8P/SOP, 8P/WSON |
| STMicroelectronics | M25PE80 | SPI | 1 MB | | | |
| | M45PE40 | SPI | 512 KB | | | |
| | M45PE80 | SPI | 1 MB | | | |
| Winbond | W25Q32B | QSPI | 4 MB | 50 MHz | -40…+85 °C -40…+105 °C | 8P/PDP, 8P/SOIC, 8P/WSON |
| | W25Q32F | QSPI | 4 MB | 50 MHz | -40…+85 °C | 8P/PDP, 8P/SOIC, 8P/WSON |

*Table 30: Reference serial Flash memory/EEPROM*

**Note 1:** The ROM loader supports QSPI with the Winbond serial Flash memory only. The Windbond serial Flash memory works in QSPI mode only, not in serial mode (SPI).

**Note 2:** If you want to use the QSPI with Macronix, make sure you use a new version of the second stage loader on your hardware to support this memory!

## 7.3 Parallel Flash memory

| Manufacturer | Size | Part number |
|---|---|---|
| Spansion | 16 MB | S29GL128P90TFIR1 |
| | 32 MB | GL256N10FFI01 |

*Table 31: Reference parallel Flash memory*

## 7.4 SDRAM

| Manufacturer | Size | Part number |
|---|---|---|
| ISSI | 8 MB | IS42S32200C1 (32 bit) |
| | 32 MB | IS42S32800B (32 bit) |
| | 64 MB | IS42S32160B (32 bit) |
| Micron | 8 MB | MT48LC2M32B2 (32 bit) |
| | 16 MB | MT48LC4M32B2 (32 bit) |
| | 32 MB | MT48LC8M32B2 (32 bit) |
| | 64 MB | MT48LC16M32 (16 bit) |

*Table 32: Reference SDRAM*

## 7.5 Other components

| Part number | Manufacturer | Function |
|---|---|---|
| QFBR-5978AZ | Avago | Fiber Optic Transceiver |
| L6360 | STMicroelectronics | IO-Link Transceiver |
| PCA82C251 | | CAN Transceiver |
| 75ALS181 | | CC-Link Transceiver |
| 65ALS167D | | PROFIBUS Transceiver |
| FAN 2001 | Fairchild | 1.5V DC/DC Regulator |
| EN5312Q | Enpirion | 1.5V DC/DC Regulator |
| MIC6315-30D2UY | Micrel | Reset Generator |
| MIC809-5SUY | Micrel | Reset Generator |
| 203313 | ERNI | Ethernet Transformer |
| J8064D628ANL | Pulse Engineering | Ethernet Transformer |
| ICK S 18x18x6.5 SA | Fischer Elektronik Hilscher GmbH | Heat sink for netX 51 |

*Table 33: Other components*

# 8 Appendix

## 8.1 List of tables

Table 1: List of revisions ..... 5
Table 2: References to documents ..... 5
Table 3: Reset device specification ..... 19
Table 4: netX 51/52 chip mode selection ..... 20
Table 5: Boot sequence and devices for the different boot options ..... 21
Table 6: RDY / RUN LED status ..... 24
Table 7: Host interface mode ..... 25
Table 8: QSPI pin assignment netX 51/52 ..... 28
Table 9: MMC/SD card insertion and CSn signal ..... 30
Table 10: Signal combination on host interface ..... 36
Table 11: DPM_SIRQ signal during startup ..... 37
Table 12: netX pin for a DPM size of 128 KB and more ..... 38
Table 13: Function table of 16-bit decode logic ..... 38
Table 14: SPI/QSPI to DPM (serial DPM) pin assignment ..... 42
Table 15: Third PHY MII pin connection ..... 50
Table 16: Status LEDs for fieldbus ports ..... 66
Table 17: Ethernet circuit component specification ..... 68
Table 18: Status LEDs for Ethernet ports ..... 74
Table 19: Status LEDs for real-time Ethernet applications ..... 75
Table 20: RTE device address switches (overview) ..... 77
Table 21: Additional RTE sync signals netX 51/52 ..... 78
Table 22: 20-pin JTAG connector pin assignment ..... 81
Table 23: ZIF connector for "mini-JTAG" ..... 82
Table 24: "mini-JTAG" ZIF pin assignment ..... 82
Table 25: ETM signals ..... 83
Table 26: List of resources – rcX kernel ..... 85
Table 27: Flash sizes for loadable firmware ..... 86
Table 28: VIA dimensions ..... 96
Table 29: Reference crystals ..... 97
Table 30: Reference serial Flash memory/EEPROM ..... 97
Table 31: Reference parallel Flash memory ..... 98
Table 32: Reference SDRAM ..... 98
Table 33: Other components ..... 98

## 8.2 List of figures

Figure 1: netX 51/52 block diagram ..... 9
Figure 2: netX 51/52 signals ..... 10
Figure 3: netX 51/52 – Basic design structures ..... 11
Figure 4: NXEB 52-COM ..... 13
Figure 5: NXEB 52-COM block diagram ..... 13
Figure 6: NXEB 52-GATEWAY ..... 14
Figure 7: NXEB 52-GATEWAY block diagram ..... 14
Figure 8: NXEB 52-ENCODER ..... 15
Figure 9: NXEB 52-ENCODER block diagram ..... 15
Figure 10: IO-Link Gateway with netX 51 ..... 16
Figure 11: netX core voltage regulator ..... 17
Figure 12: netX 51/52 system oscillator circuit ..... 18
Figure 13: netX 51/52 reset circuits ..... 19
Figure 14: netX 51/52 chip mode selection ..... 20
Figure 15: netX 51/52 RDY/RUN circuit ..... 21
Figure 16: Sample schematic, netX 51/52 security memory and RDY/RUN LED ..... 22
Figure 17: How to configure the host interface mode during boot up ..... 25
Figure 18: netX 51/52 SPI Flash ..... 27
Figure 19: netX 51/52 with QSPI Flash ..... 28
Figure 20: netX 51 MMC/SD card ..... 30
Figure 21: netX 52 MMC/SD card ..... 30
Figure 22: MMC/SD card and Flash memory on SPI bus ..... 31
Figure 23: netX 51 Flash - Address line A0 for low/high byte selection ..... 33
Figure 24: netX 51 Flash - A0 as the LSB of a word address ..... 34
Figure 25: netX 51 SDRAM 1 * 16 bit, 1 * 32 bit ..... 35

Figure 26: Host interface connection possibilities .... 36
Figure 27: Using the second stage boot loader to select parallel dual-port memory with 8 or 16-bit mode .... 37
Figure 28: Intel $^{TM}$ interface, 8-bit, non-multiplexed .... 39
Figure 29: Intel $^{TM}$ interface, 16-bit, non-multiplexed .... 39
Figure 30: Intel $^{TM}$ interface, 16-bit, non-multiplexed, 2 write signals (low byte, high byte) .... 39
Figure 31: Intel $^{TM}$ interface, 32-bit, non-multiplexed .... 39
Figure 32: Motorola $^{TM}$ ColdFire, 16-bit, non-multiplexed .... 40
Figure 33: Motorola $^{TM}$ M68000, 16-bit, non-multiplexed .... 40
Figure 34: Intel $^{TM}$ interface, 8 bit, multiplexed .... 40
Figure 35: Intel $^{TM}$ interface, 16 bit, multiplexed .... 40
Figure 36: DPM read access by using DPM_BUSYn signal .... 41
Figure 37: SPI/QSPI to DPM interconnection .... 42
Figure 38: SPM performance .... 43
Figure 39: netX extension bus Intel$^{TM}$ type interface circuit, 8 bit, non-multiplex .... 44
Figure 40: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, non-multiplex .... 45
Figure 41: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, non-multiplex, write high/low .... 45
Figure 42: netX extension bus Motorola$^{TM}$ type interface circuits, 8 bit, non-multiplex .... 45
Figure 43: netX extension bus Motorola$^{TM}$ type interface circuits, 16 bit, non-multiplex .... 45
Figure 44: netX extension bus Intel$^{TM}$ type interface circuits, 8 bit, multiplex .... 46
Figure 45: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, multiplex .... 46
Figure 46: netX extension bus Intel$^{TM}$ type interface circuits, 16 bit, multiplex, 2 write signals .... 46
Figure 47: netX extension bus Motorola$^{TM}$ type interface circuits, 8 bit, multiplex .... 47
Figure 48: netX extension bus Motorola$^{TM}$ type interface circuits, 16 bit, multiplex .... 47
Figure 49: Third Ethernet PHY on host interface .... 48
Figure 50: 16-bit (left) / 32-bit (right) SDRAM at host interface .... 50
Figure 51: GPIO block diagram .... 55
Figure 52: netX 51/52 UART0 with RS-232 transceiver .... 56
Figure 53: netX 51/52 USB DOWNstream port (device mode) .... 57
Figure 54: IO-Link connections .... 60
Figure 55: netX 51/52 fieldbus interface .... 61
Figure 56: Basic circuit of netX CANopen interface .... 62
Figure 57: Basic circuit for netX CC-Link interface .... 63
Figure 58: Basic circuit for netX DeviceNet interface .... 64
Figure 59: Basic circuit for netX PROFIBUS interface .... 65
Figure 60: Fieldbus status LEDs .... 66
Figure 61: Fieldbus address switches .... 66
Figure 62: netX 51/52 2-channel Ethernet circuit (Twisted pair) .... 67
Figure 63: netX 51/52 single-channel Ethernet circuit (Twisted pair) .... 69
Figure 64: netX 51/52 Ethernet circuit (Fiber optic) .... 71
Figure 65: netX 51/52 Ethernet circuit (PHYs not used) .... 73
Figure 66: netX 51/52 Ethernet status LED circuit .... 74
Figure 67: netX 51/52 RTE status LED schematic .... 75
Figure 68: RTE device address switches .... 77
Figure 69: netX 51/52 JTAG circuits .... 80
Figure 70: "mini-JTAG" connector for debugging and Boundary Scan Test .... 81
Figure 71: ETM interface .... 84
Figure 74: Approved netX PCB layer stack .... 90
Figure 75: netX 51/52 decoupling caps .... 91
Figure 76: Oscillator circuit with ground shield .... 92
Figure 77: Edge-Coupled Sourface Micro Strip .... 93
Figure 78: Edge-Coupled Offset Strip Line .... 94
Figure 79: netX Memory Bus .... 94
Figure 80: Routing Example .... 95
Figure 81: VIA position and signal fan out under netX 51/52 .... 96

## 8.3 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