SysKonnect

SysKonnect
a business unit of
Schneider & Koch & Co.
Datensysteme GmbH
Siemensstraße 23
D-76275 Ettlingen
Tel. 0 72 43 / 50 2100
Fax 0 72 43 / 50 29 89

October 16, 1995
Version 1.0

SK-NET FDDI-<x>P

Technical Manual

SK internal documentation

Revision History

Revision	Released at	Changes from Previous Release	Remarks
1.0	October 16, 1995	n.a.	First Release after design closing. No prototype available yet.

Related Documentation

"AM79C850 Supernet 3" Preliminary documentation, Issue Date March 1995, Publication # 19574 Rev. Afrom Advanced Micro Devices (AMD)

"Flash Memory Products : 1992/1993 Data Book/Handbook" from Advanced Micro Devices (AMD)

"PCI Local Bus Specification" Revision 2.1, June 1, 1995 from PCI Special Interest Group

"PCI Bios Specification" Revision 2.1, August 26, 1994 from PCI Special Interest Group

"FDDI Station Management : Draft Proposed American National Standard" Rev 7.2, June 25, 1992, X3T9/92-067 and editorial changes -> Rev. 7.3, X3T9.5/93-085

"Fibre Distributed Data Interface (FDDI) - Part1: Token Ring Physical Layer Protocol" ISO 9314-1 :1989

"Fibre Distributed Data Interface (FDDI) - Part2: Token Ring Media Access Control (MAC)" ISO 9314-2 : 1989

"Fibre Distributed Data Interface (FDDI) - Part 3: Token Ring Physical Layer Medium Dependent (PMD)" ISO/IEC 9314 -3 : 1990

"FDDI Twisted Pair Physical Layer Medium Dependent (TP-PMD) : Proposed American National Standard" Rev. 2.1 03/01/94

"FDDI Low-Cost Fibre Physical Layer Medium Dependent (LCF-PMD) : Draft Proposed American National Standard for Information Systems" Rev. 2.0, 03/01//94

Current Release

Approved by		
Hardware Engineer	C.-F. Ruf	
Software Engineer (if applicable)		
Technical Writer	W. Frank	

Table of Contents

List of Tables

SysKonnect

October 16, 1995
Version 1.0

SK-NET[1] FDDI-<x>P Technical Manual

This manual describes the hardware of the network interface card "**SK-NET FDDI-<x>P**". It contains a complete description of the card's programming interface. This will enable the software engineer to develop driver and diagnosis software according to the specifications mentioned below.

1.0 Introduction

The "**SK-NET FDDI-<x>P**" is a highly integrated FDDI controller for PCI Local Bus systems designed to address high-performance system application requirements. The bus master architecture with integrated buffer management provides high system throughput besides minimum CPU and system bus load.

1. SK, SK-NET are trademarks of Schneider & Koch & Co. Datensysteme GmbH and SysKonnect, Inc. All other brand or product names are trademarks or registered trademarks of their respective holders.
SK-NET FDDI-FP, SK-NET FDDI-UP, etc. are interim names. In the market the NICs will be known as SK-<xxxx>, where <xxxx> is a four digit number.

The Network Interface Card (NIC) is designed to meet the PCI Local Bus Specification Version 2.1

The “**SK-NET FDDI-<x>P**”is a Universal Board i.e. it can be operated in both 3.3V and 5V PCI bus connectors respectively. The card’s dimensions are those of a PCI short length add-in-board. It is designed for 33MHz bus clock and a 32 bit wide data path.Thus, a max. transfer rate of 132 Mbit/s can be achieved on the PCI bus interface.

“**SK-NET FDDI-<x>P**” complies with the ANSI X3T9.5[1] specifications for FDDI. This includes Media Access Control (MAC), Station Management (SMT), Physical Layer Control (PLC) and several Physical Medium Dependent (PMD) specifications. There are different NIC versions with different PMDs for attachment to various FDDI cabling plants.

- ❑ Multimode optical fiber (62.5/125 µm or 50/125 µm)
- ❑ Low-cost fibre Medium Interface Connector (LCF-MIC; Power budget is that of multimode fibre cabling)
- ❑ Category 5 cabling as specified in the TP-PMD standard

The “**SK-NET FDDI-<x>P**” supports also the optional “restricted token monitoring” as described in the Ring Management (RMT) part of the SMT specifications. And the Supernet 3 architecture supports the handling of synchronous traffic additionally to the handling of asynchronous network traffic.

The “**SK-NET FDDI-<x>P**” card runs with the third generation of the AMD SUPERNET™ FDDI controller[2]: *Supernet 3* and with an Application Specify Integrated Circuit (ASIC) designed by SysKonnect. The card’s main hardware components are:

- ❑ ASIC interfacing AMD’s Supernet 3 to the PCI bus providing:
  - 32 bit bus interface
  - Supporting big/little endian in data pathes for receive and transmit queues
  - High performance bus master architecture with integrated Buffer Management Unit (BMU) for minimum CPU and bus load
  - Individual 128Byte FIFOs for receive queue 1 and 2, for the asynchronous and the synchronous transmit queue providing buffering between PCI Local Bus and buffer memory
  - Supporting misaligned data transfers between system memory and FIFOs
  - Supporting advanced PCI protocol (Memory Write and Invalidate, Memory Read Line and Multiple for cache line sizes up to 16)
  - Parity generating and checking at PCI local bus interface and internal FDDI data paths

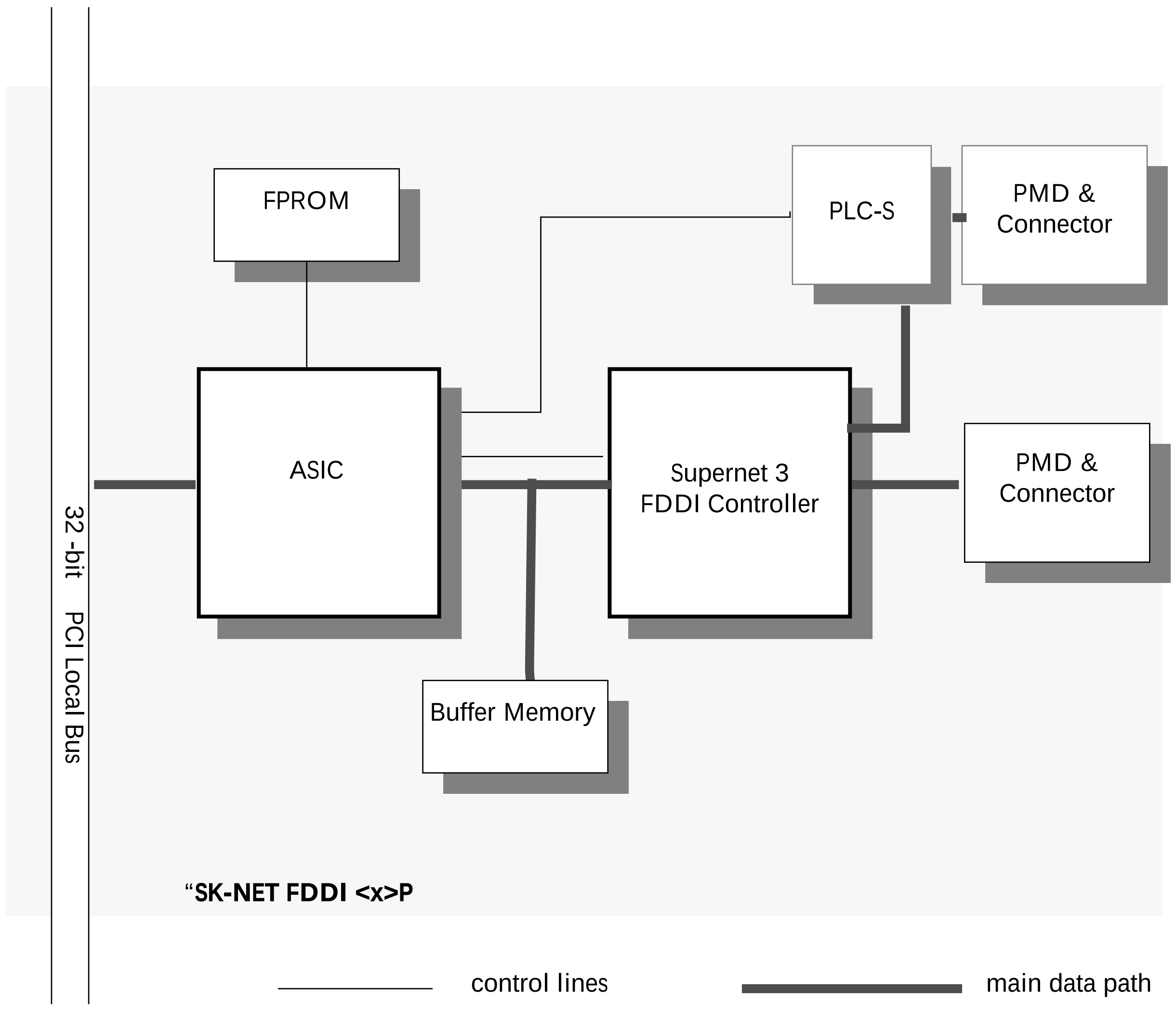
1. Since 1994 The ANSI X3T9.5 group works as ANSI X3T12 committee.

2. For detailed information see: “The SUPERNET™ 2 Family for FDDI: 1991 Data Book” from Advanced Micro Devices.

- ❏ 128Byte buffer memory for receive and transmit queues on-board
- ❏ Advanced Micro Devices' SUPERNET3 chip "AM79C850" providing:
    - Buffer memory clock frequency range 12.5 MHz to 25 MHz
    - Full duplex operation: 200 Mbit/s continuous data rate
    - Full support for synchronous transmission
    - Pointers to claim and beacon frame
    - Integrated buffer memory management
    - ANSI compliant TP-PMD Stream Cipher Scrambling/De-scrambling
    - Hardware Physical Connection Management support
    - Low power consumption - reduction of more than 25% from SUPERNET 2 solution
- ❏ An additional PLC -S on the Dual Attachment Station (DAS) version of the NIC.
    - ANSI compliant TP-PMD Stream Cipher Scrambling/De-scrambling
- ❏ A connector for an external optical bypass on DAS
- ❏ One (two on DAS) Transceiver for attachment to the network. One of
    - Multimode optical fiber (62,5/125 µm or 50/100 µm)
    - Low Cost Fiber MIC - multimode fiber power budget
    - Twisted Pair according to the ANSI X3T12 TP-PMD specification.

2.0 Physical Blocks

This survey describes the structure of the "**SK-NET FDDI-<x>P**" card and the functions of essential card elements. The block diagram displays the



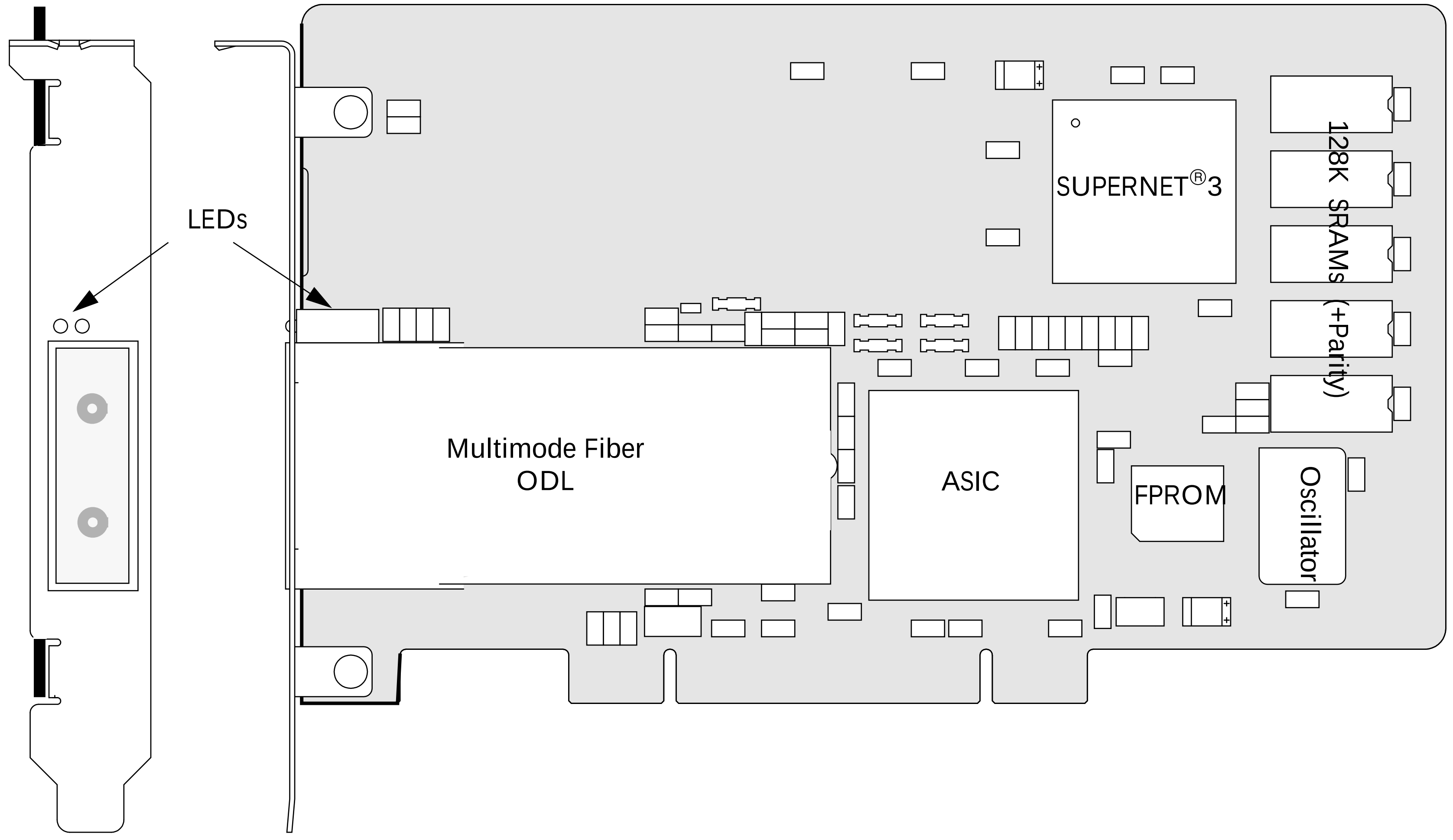
controller's main devices that must be programmed or accessed to operate the card.

- ❏ Application Specific Integrated Circuit (ASIC)
- ❏ Reprogrammable Flash EPROM (FPROM)
- ❏ Supernet 3 FDDI Controller
- ❏ 128 KByte Buffer Memory
- ❏ Optional PLC (on DAS NICs only)

The following figures show the arrangement of devices for two versions of the NIC as example. (There are more version e.g. the Low-Cost Fibre NIC.)

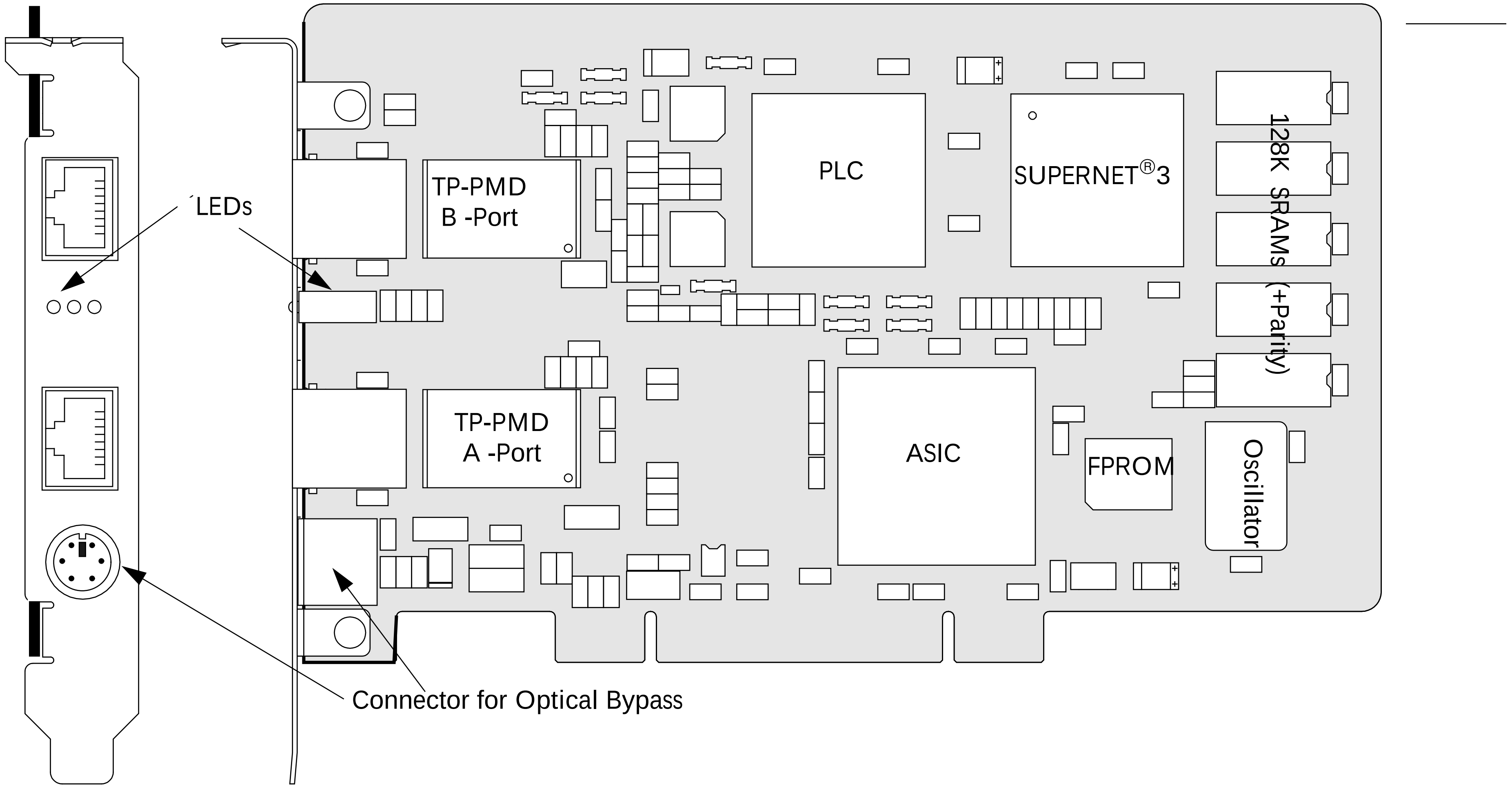
The **SK-NET FDDI-FP** an FDDI PCI network interface card for multimode optical fiber**.** Single Attachment Station with FDDI multimode fiber MIC (Medium Interface Connector)

The usage of LEDs is software controlled. For the proposed usage of LED look at section 5.2.3 *LED Register* on page 62



The **SK-NET FDDI-TP** an FDDI PCI network interface card for Category 5 Twisted Pair cabling according to the ANSI TP-PMD specification**.** Dual Attachment Station with two RJ45 jacks for attachment of TP-PMD Category 5 cabling

The usage of LEDs is software controlled. For the proposed usage of LED look at section 5.2.3 *LED Register* on page 62



2.1 Functional Description

The following gives a short description of the block diagram’s components. A more detailed discussion of the logical units most of them implemented in the ASIC is subject of the following chapters. For detailed information about the SUPERNET 3 internals you must refer to AMD’s manuals. Information about SUPERNET 3 beyond the interface to the ASIC is out of the scope of this manual.

2.1.1 ASIC

The Application Specific Integrated Circuit (ASIC) directly interfaces the PCI local bus on one side and the FDDI devices on the other side. It comprises the main control units of the network interface card. The initialization values for the registers implemented in the ASIC are hardwired or can be overwritten by initialization values from the FPROM. The ASIC contains the following main logical units:

- ❏ Bus Interface Unit (BIU) interfacing the PCI Local Bus
- ❏ Buffer Management Unit (BMU) accessing data structures in the host memory
- ❏ Host Processor Interface (HPI)
- ❏ Configuration Registers
- ❏ Control Register File

2.1.2 Buffer Memory

The “**SK-NET FDDI-<x>P**” controller comes with a 128-Kbyte SRAM for frame buffering. The buffer memory is controlled by the SUPERNET 3.

2.1.3 SUPERNET 3

The SUPERNET 3 chip AM79C850 controls access to the medium. This implementation complies with the ISO 9314 specification. The SUPERNET 3 is derived from its predecessor the SUPERNET 2 chipset. SUPERNET 3 integrates and enhances the functions of its predecessors. On the “**SK-NET FDDI-<x>P**” network interface card the FORMAC Plus operates at 25 MHz in tagged[1] mode.

In the card‘s DAS version the MAC unit has no access to data transferred on the secondary ring. The SUPERNET 3 puts data or receives data from the primary ring. The secondary ring can be used as a redundant backup ring only. This can also be seen from the figures showing the data path on the FDDI part of the NIC, starting on the following page.

2.1.4 PLC-S

Only on the DAS (Dual Attachment Station) versions of “**SK-NET FDDI-<x>P**” an additional PLC-S (Physical Layer Controller) is mounted on the NIC[2]. As the PLC -S in the SUPERNET 3 chip the additional PLC -S supports ANSI compliant TP-PMD Stream Cipher Scrambling/Descrambling

1. Tag/non-tag mode are features of the SUPERNET 2 chipset. The SUPERNET 3 is operated in the tag mode. Non-tag mode is no longer supported.

2. The PLC-S for the SAS is integrated in the SUPERNET 3.

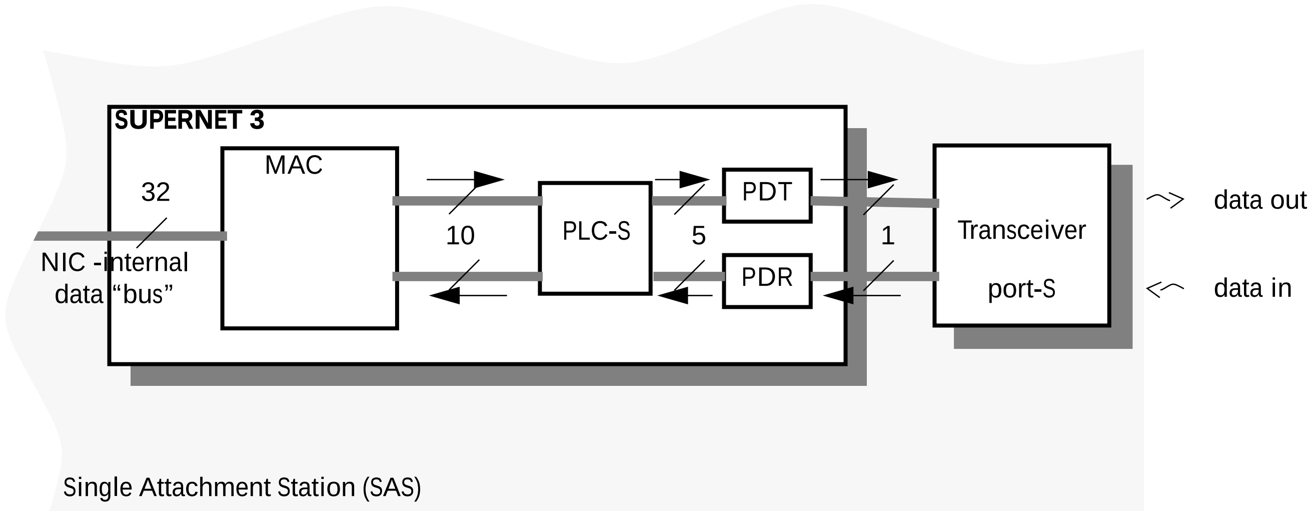
2.1.5 Transceivers

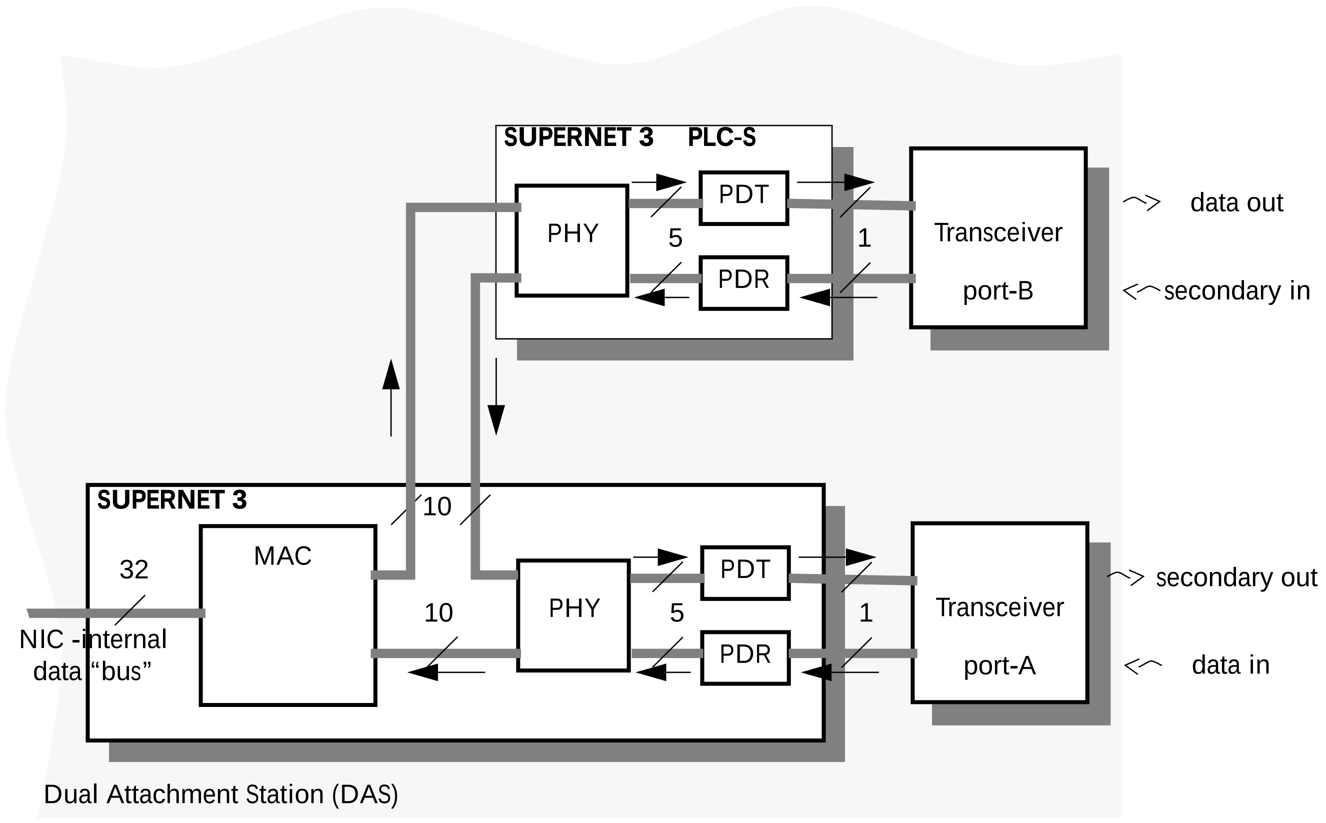
The are several transceivers foreseen for different cabling plants:

- ❑ Multimode Fibre ODL (Optical Data Link) for 62.5µm/125µm (or 50µm/125µm) core/cladding ratio
  Connector: FDDI MIC (Medium Interface Connector)
- ❑ Multimode Fibre ODL, however with
  Connector: LCF MIC (Duplex SC Connector)
- ❑ Cooper interface for Category 5 cabling according to X3T12 TP-PMD specification.
  Connector: RJ45 jack

2.1.6 Data Path in the NIC's FDDI Domain

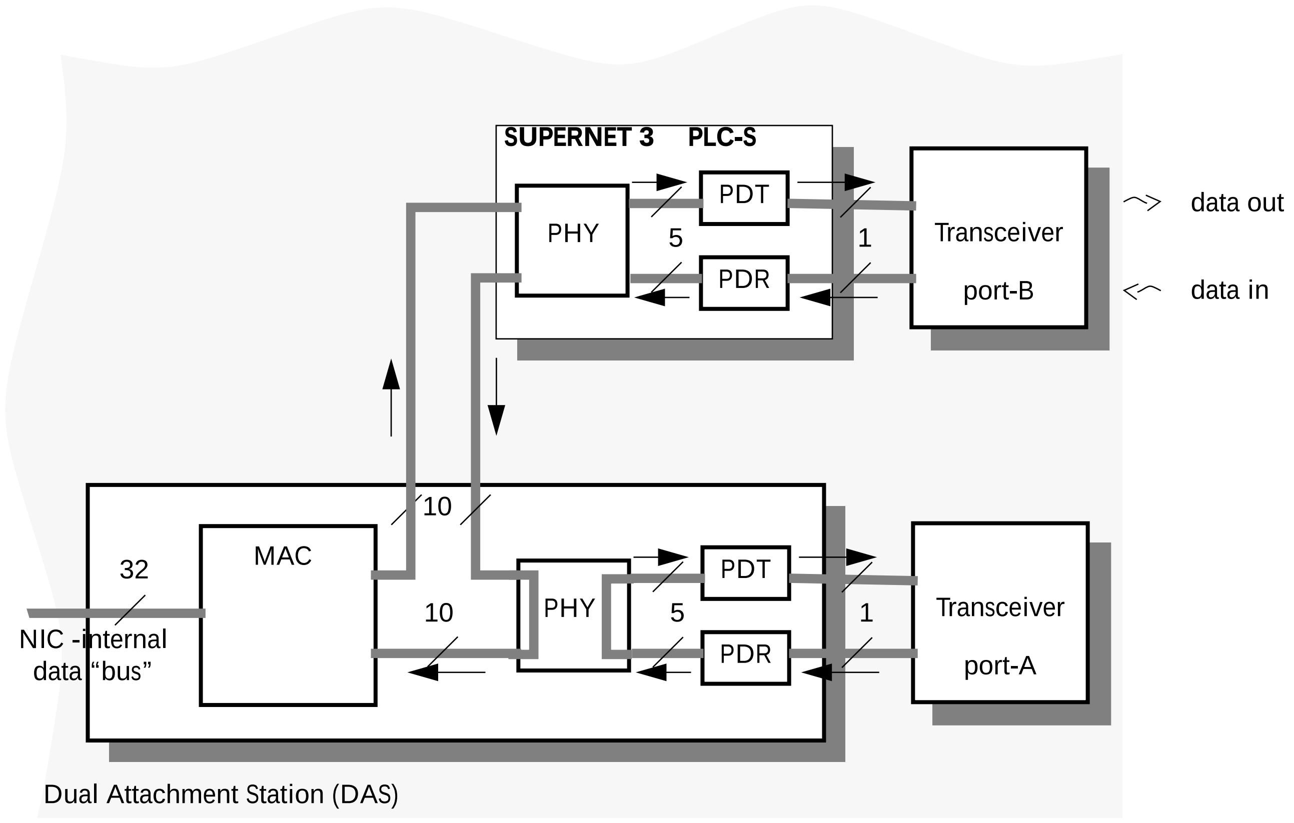
Details on data paths between SUPERRNET 3, the additional PLC (Physical Layer Controller) on DAS, and the transceivers - the attachment of the SUPERNET 3' receive and transmit buses, are sketched in the following.





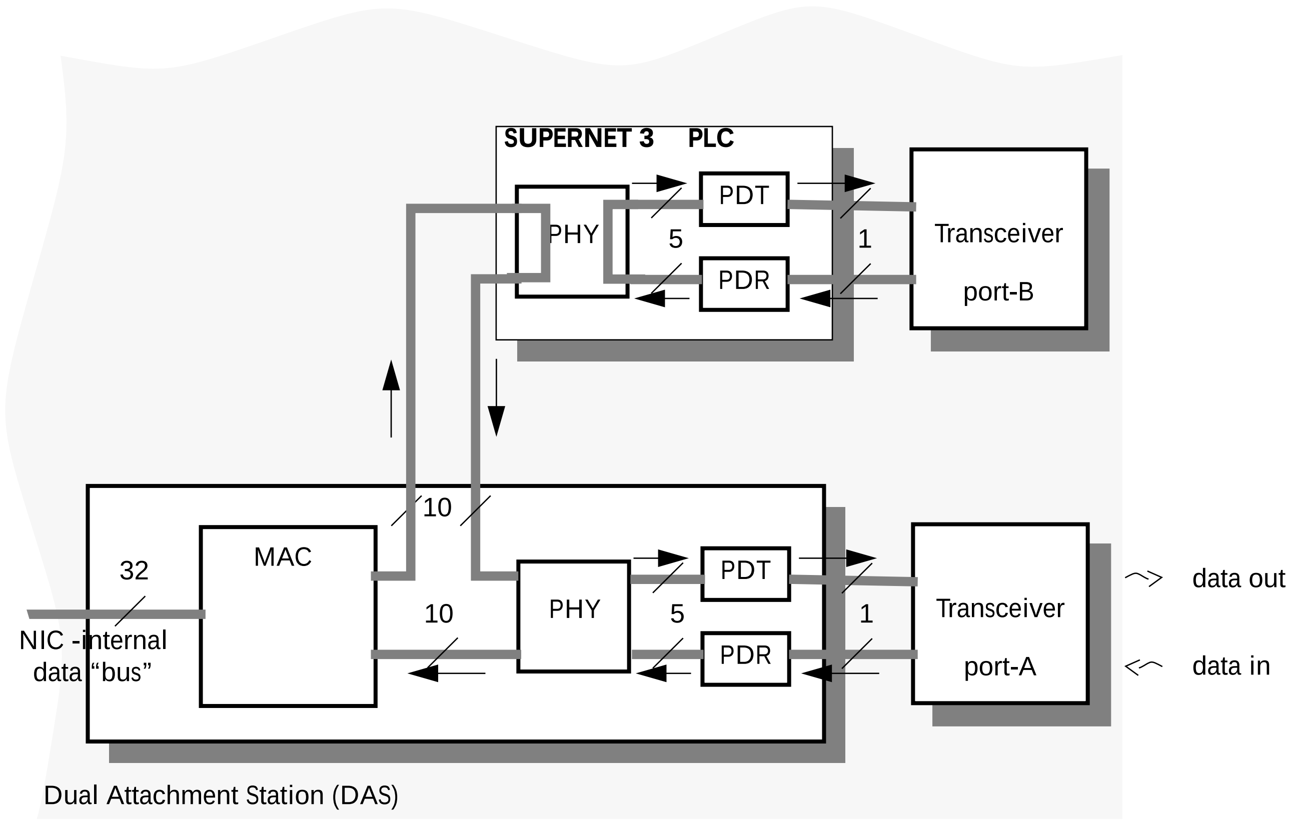
THRU State

THRU is the normal operational state of the NIC's DAS version. Data is received at the station's A port and transmitted at the station's B port.



WRAP-A State

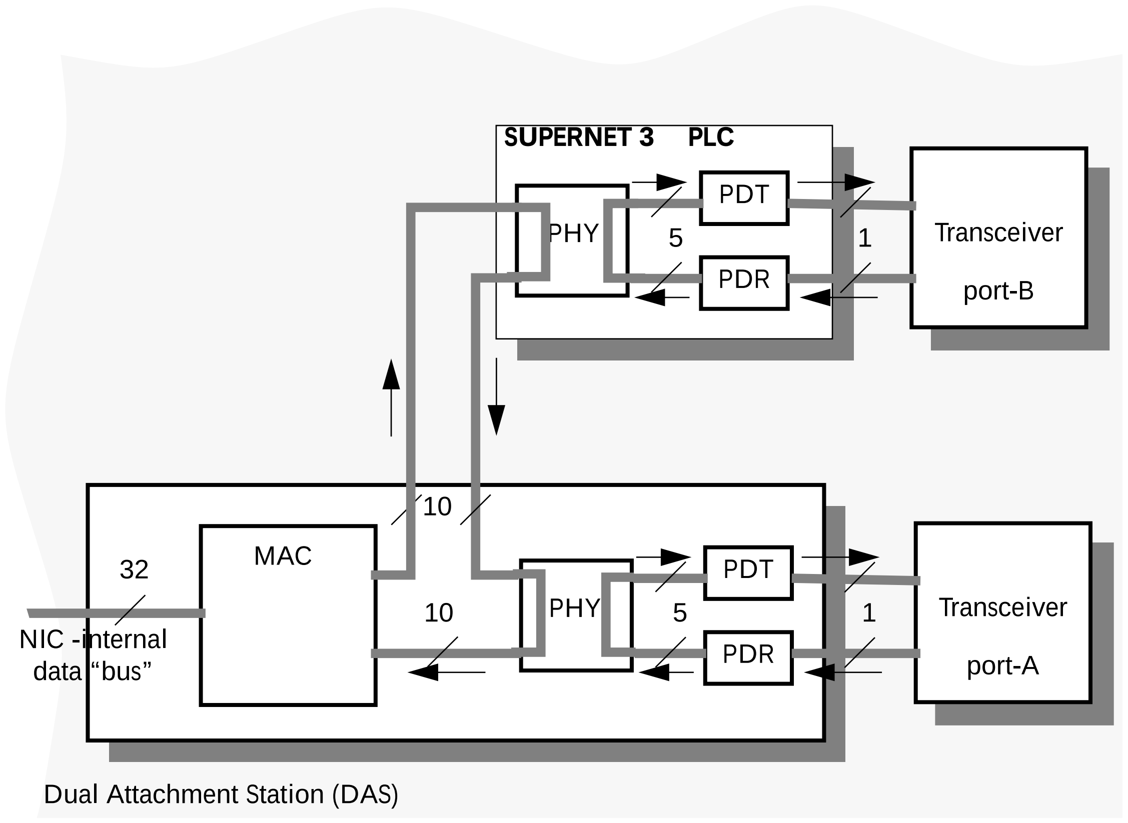
The DAS NIC is operated in WRAP A state, e.g. if the A port can not connect to the neighbor port or if the connection was broken. Data is received at the B port, handled by the MAC unit and transmitted at the B port.



WRAP-B State

The DAS NIC is operated in WRAP B state, e.g. if the B port can not connect to the neighbor port or if the connection was broken. Data is received at the A port, handled by the MAC unit and transmitted at the A port.

In the isolated state, there are loopbacks at all PLCs on the NIC. For example for the DAS:



Isolated State

3.0 Programmable Resources Overview

The PCI specification provides three different information spaces and different methods to access and control a NIC via the PCI bus:

- ❏ Configuration Space (Hardware access methods are Configuration Read and Configuration Write cycles)
- ❏ The Memory Space (There are several hardware access Methods on the PCI bus e.g. Memory Read, Memory Read Multiple, ...)
- ❏ I/O Space (translates to I/O Read, I/O Read Write cycles on the PCI bus)

3.1 Document Conventions

Starting with this chapter more and more tables will be shown. They are used to list the usage and settings of register, of single bits in registers, etc. To keep tables more compact some abbreviations, special notation will be used:

Register Descriptions		
Write:		
	ne	no effect
	yes	writable
	sh	special handling as described in the text or the table
	exec	execution of this command - may refer to single bits - the command is executed if the corresponding bit is written as 1.
Read:		
	aw	as written
	value	as defined by itself
Commands Commands are executed, if the appropriate bit is set. Read value as defined.		
Exclusive Commands (e.g. Start/Stop, ON/OFF (usually two bits) Commands are executed, if appropriate bit is set to 1. Setting both commands to 1, has no effect. Status is readable: 0b01 or 0b10.		

Table 1: Notations

3.2 Configuration Space Basics

Providing configuration information and supporting access to configuration information is mandatory for a PCI NIC. The configuration space comprises information and writable registers to identify the NIC, to store the memory and I/O addresses assigned by the system's start-up routines — all the information required to map the NIC seamless into the PCI bus system.

The Configuration Space is usually accessed using BIOS routines as described in the PCI BIOS specification. The Configuration Space of e.g. a NIC is addressed by selecting the IDSEL (detected by reading the unique Device ID)+ an address in the 256 Byte configuration space address range. This translates to Configuration Read/Configuration Write cycles executed on the PCI bus hardware level.

The "**SK-NET FDDI-<x>P**" is responding to type 0 configuration accesses, i.e. AD<1..0> = "00", IDSEL# asserted. The NIC does not support burst transfers from/to the configuration space. To burst transfer PCI bus cycles it responds with a disconnect with the first data transfer.

The Configuration Space can be accessed with 8-bit, 16-bit, or 32-bit transfers. On read transactions all data is driven as defined for full 32-bit accesses independent of BE<3..0>*.

(As described later the Configuration Space of the "**SK-NET FDDI-<x>P**" can be accessed also via I/O or Memory accesses. Not recommended — for test purposes only.)

The "**SK-NET FDDI-<x>P**" provides optional access to an Expansion ROM. The base address is assigned by the PCI system and deposited in the configuration space.

The base addresses for I/O and Memory access are also assigned by the PCI system and deposited in the configuration space.

Besides this mandatory configuration information in the configuration space header, the "**SK-NET FDDI-<x>P**" provides optional access to an 32-bit register called "*Our_Register*" in the device dependent region of the NIC's configuration space.

Our_Register contains settings that are important for initialization and not for run time. For example *Our_Register* contains a flag that enables/disables booting from a Boot ROM.

Intention

Values of *Our_Register* may be configured after installation of the NIC, when it is installed and configured the first time. The PCI specification Vers. 2.1 specifies an mechanism similar to the configuration of EISA NICs in EISA bus system.

From the *UDF* bit in the configuration space header the PCI system recognizes that User Defined Features (UDF) - e.g. booting from a Boot PROM or not - are available for this NIC. These features will be provided to the user installing the NIC by a configuration utility. The user can select between the various offered options. The configuration utility is provided by the computer system. It reads the optional features from a PCI Configuration File (PCF) provided by the NIC manufacturer.

(For this feature the *UDF* bit must be set to 1 in the FPROM. This is not foreseen for the first series of "**SK-NET FDDI-<x>P**" NICs but can be changed by setting the initialization value of the FPROM correspondingly. Additionally a PCF must be provided.)

Access to the Expansion ROM information and to *Our_Register* is optional, i.e. there are flags to enable/disable access to these registers. If they are disabled they behave like reserved registers.

A detailed description of the NIC's configuration space will be given in chapter chapter *4.0 Configuration Space* on page 44.

3.3 I/O Space and Memory Space Basics

The "**SK-NET FDDI-<x>P**" registers can be mapped in both I/O space[1] and Memory space. It depends on some initialization values whether the NIC requests I/O mapping, Memory mapping or both.
As designed the "**SK-NET FDDI-<x>P**" could also be operated mapped to the I/O space only, or mapped to the Memory space only. The PCI specification "highly recommends" mapping to the Memory space over mapping to the I/O space. As default the "**SK-NET FDDI-<x>P**" is mapped to both spaces simultaneously.. The NIC claims

- ❑ 256 Byte of 32bit I/O space
  and
- ❑ 2 KByte of 32bit Memory space

A memory base address and an I/O base address are assigned to the NIC by the system software. The assigned base addresses are written to the configuration space header.

Thus, you can access the NIC's registers, e.g. control/status registers, timer, interrupt mask and source register, etc. either via I/O accesses or Memory accesses.

All the registers of the "**SK-NET FDDI-<x>P**" are comprised in the Control Register File. Also the Configuration Space registers are mapped in the Control Register File.

If the NIC is mapped into both, the Control Register File can be accessed via I/O accesses and via Memory accesses. With both types of access you operate on the same registers.

The Control Register File comprises all control information necessary to operate the NIC. For example some control functions:

- ❑ Control of the various state machines realized in the ASIC
- ❑ Control of data transfer between the host memory and the on-board buffer memory.
- ❑ Control of the SUPERNET 3 chipset
- ❑ Interrupt sources and interrupt mask
- ❑ On-board timer
- ❑ ... .
- ❑ All you need to operate the card

1. Originally, the NIC was designed not to request I/O space by default. Thus there's a default value of "0" in bit 0 of the I/O base address register. However this "default value" will be overwritten by a default value of "1" specified in the FPROM. Both I/O and Memory accesses are foreseen by the hardware of the "**SK-NET FDDI-<x>P".** It's just a question of initialization and setting default values via the FPROM**.**

3.3.1 I/O Access

The registers mapped in the I/O space have to be accessed with the minimum data width, i.e. 8-bit, 16-bit, or 32-bit transfers as the registers are defined.

If the I/O resources are targeted for a burst operation on the PCI bus, they respond with a disconnect with the first data transfer. On read transactions, all data is driven as defined for full 32-bit accesses independent of BE<3..0>*.

3.3.2 Memory Access

There are two accessible resources:

- ❏ Memory mapped registers (Control Register File)
- ❏ Expansion ROM (FPROM)

The registers mapped in the Memory space have to be accessed with the minimum data width, i.e. 8-bit, 16-bit, or 32-bit transfers as the registers are defined. The Expansion ROM can be accessed with 8bit, 16bit, or 32bit transfers.

If the Memory space is targeted for a burst operation on the PCI bus, it responds with a disconnect with the first data transfer. On read transactions, all data is driven as defined for full 32-bit accesses independent of BE<3..0>*.

3.4 Principles of Operation

When looking more detailed at the Control Register File all the registers controlling single actions as switching LEDs on/off, clearing interrupts, resetting the NIC will be explained when listing the registers. This chapter deals with some main design ideas of the “**SK-NET FDDI-<x>P**”.

- ❏ Transfer of FDDI frames from the SUPERNET 3 chipset to the host memory and vice versa — how to transfer data along the PCI bus. This includes description of
  - Buffer Management Unit (BMU)
  - Queues, FIFOs, Host[1] Processor Interface
- ❏ Initialization — FPROM Loader
- ❏ Reset Behavior
- ❏ Parity Generation

The main task of the ASIC is to transfer data between the FDDI part of the NIC operating at 25 MHz along the PCI bus operating at a max. of 33 MHz. To complicate this, there may be concurrent request to transfer data in different queues (synchronous/asynchronous) in both directions, and ... ...

To accomplish this task the Bus Interface Unit (BIU) Master state machine implemented in the ASIC operates as bus master on the PCI bus. The Buffer Management Unit (BMU) also implemented in the ASIC controls read and write action on the host memory.
(Please note that the BMU operates with addresses in the host memory, not on the on-board buffer memory as one may guess from the name. The on--board buffer memory is managed by the SUPERNET 3 chip.)

Roughly speaking the BMU cares for addresses and data fragment length, while the BIU master state machine, several independent FIFOs and control logic care that data is transferred as contiguous fragments from/to the FDDI side (25 MHz) over the PCI bus (33MHz) to/from the host memory.

Receive FIFOs gather “low” bandwidth data on the FDDI side sending them to the host memory in PCI burst transfers.
Transmit FIFOs are filled by “high” bandwidth PCI bursts. They forward data to the lower bandwidth FDDI devices.

3.5 Buffer Management Unit

The “**SK-NET FDDI-<x>P**” provides various ways to transfer data. E.g. single registers of the SUPERNET 3 chipset can be read/written allowing transfer of single words. This may be useful for diagnosis software.

For normal operation the recommended method is preparing data structures in the host memory -called descriptors- according to the rules described in the following. Then invoke the NIC’s BMU to read the descriptor, and transfer the data fragments or total frames from/to the buffer memory which is controlled by the SUPERNET 3 chip.

Frame, data handling in the buffer memory is done by the SUPERNET 3. On the other side, in the host memory, the driver must prepare data structures, deposit or read data and handle the descriptors.

1. The expression ‘host’ may be misleading in this context. It is derived from the SUPERNET 3 terminology. Think of it as the SUPERNET’s interface to the host side.

The main information required by the BMU is deposited in the 4×32 bit descriptors. Since the NIC supports four data queues:

- ❏ Receive Queue 1
- ❏ Receive Queue 2
- ❏ Asynchronous Transmit Queue
- ❏ Synchronous Transmit Queue

there are four independent BMUs. Concurrent actions are controlled by an arbiter attached to them.

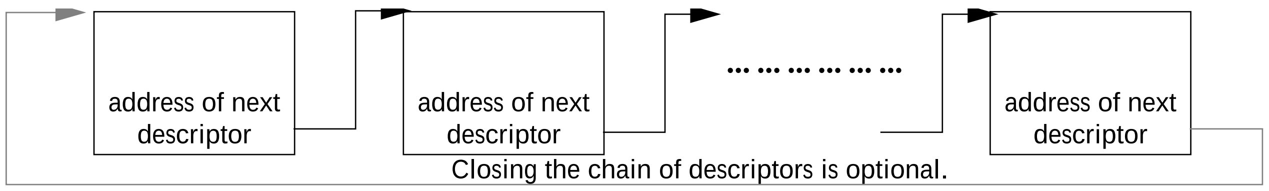
3.6 Descriptor Chains

For each queue at least one descriptor structure is required in the host memory. Usually for each queue a chain of descriptors is build in the host memory. The descriptor structure provides the following informations to the BMU:

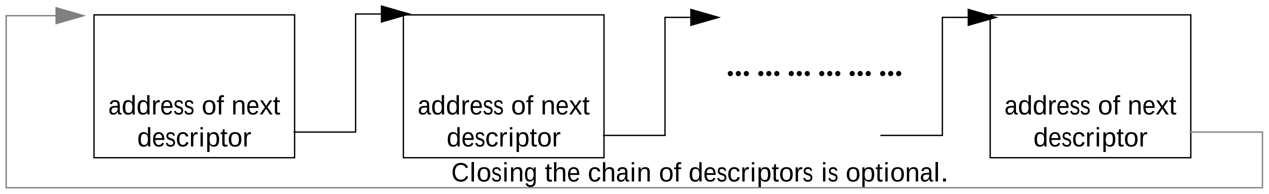
- ❏ An Own_Bit indicating whether the host (driver software) or the BMU owns the current descriptor
- ❏ Buffer address (address of data to be transferred)
- ❏ Buffer length (number of bytes to be transferred)
- ❏ Address of next descriptor in the chain.

Thus, for each of the four queues a chain of descriptors should be build, preferably at driver initialization time. It's also possible (not necessary) to chain the descriptors in a descriptor ring as sketched in the following figure.

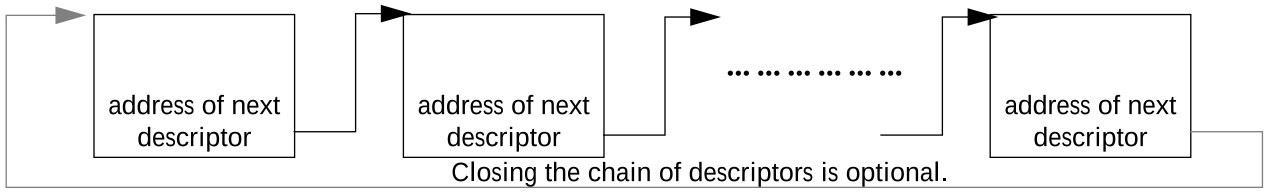
Chained Descriptors for Receive Queue 1


address of next descriptor
address of next descriptor
... ...
address of next descriptor
Closing the chain of descriptors is optional.

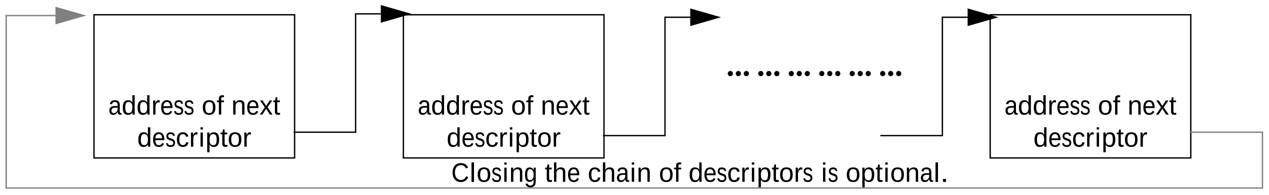
Chained Descriptors for Receive Queue 2


address of next descriptor
address of next descriptor
... ...
address of next descriptor
Closing the chain of descriptors is optional.

Chained Descriptors for Synchronous Transmit Queue


address of next descriptor
address of next descriptor
... ...
address of next descriptor
Closing the chain of descriptors is optional.

Chained Descriptors for Asynchronous Transmit Queue


address of next descriptor
address of next descriptor
... ...
address of next descriptor
Closing the chain of descriptors is optional.

After setting up the descriptor chains, the address of the first descriptor of each queue must be written in the corresponding Control Register File once, to initialize the Current Receive/Transmit Descriptor Address. Later on the BMU reads descriptor addresses and updates the information in the Control Register File - together with additional information and control bits to handle the BMU mechanism. See section 8.0 *Appendix A: Control Register File* and Table 5: *Receive /Transmit Queue Register Structure* on page 27.

The number of descriptors for a queue is theoretically unlimited. The size of a message descriptor is 4 × 4 bytes. The descriptors have to be located on 16 byte boundaries.

Hardware details: Buffer management is implemented for each queue separately, i.e. the BMU is four independent BMUs. The BMUs are structured similar. The BMUs for the Receive Queues are identical in detail and the BMUs for the Transmit Queues are identical in detail.

Own_Bit

The Own_Bit indicates ownership i.e. the right to change the descriptor contents.

Value	Meaning
0	host is owner of the descriptor After writing/updating the descriptor the host relinquishes the descriptor to the BMU by setting the Own_Bit to “1”.
1	BMU is owner of the descriptor After writing/updating the descriptor the BMU relinquishes the descriptor to the host by setting the Own_Bit to “0”.

Table 2: Descriptor: Own_Bit

After building, initializing the descriptor chains, receive queue descriptors should be owned by the BMU, transmit queue descriptors should be owned by the host. And the BMU Control/Status registers of each queue should contain the address of a descriptor, the “current descriptor”.

Besides the Own_Bit some bits of the descriptor are set only by the host (while he is the owner) some bits are set only by the BMU owning the descriptor.

3.6.1 Descriptor Entries in Detail

First, descriptors for transmit queues slightly differ from receive queue descriptors. The table lists the entries of the TBCTRL (Transmit Buffer Control) registers

Bit	Name	Description	Address	Defined by
TBCTRL	**Transmit Buffer Control register**		base	
31	Own_Bit	= 0 host is owner = 1 the BMU is owner After setting all other bytes of a descriptor, the host sets the Own_Bit to “1” for relinquishing the descriptor to the BMU. After writing all other bytes of a descriptor, the BMU sets the Own_Bit to “0” for relinquishing the descriptor to the host.		host BMU
30	STF	Start of Frame indicates that this is the first descriptor used by the BMU for this frame. True, if = 1		host
29	EOF	End of Frame indicates that this is the last descriptor used by the BMU for this frame. If EN_IRQ_EOF and EOF are set, an interrupt IRQ_EOF is generated after relinquishing this descriptor. True, if = 1		host
28	EN_IRQ_EOB	If EN_IRQ_EOB is set, an interrupt IRQ_EOB is initiated after relinquishing this descriptor. True, if = 1		host
27	EN_IRQ_EOF	Enable IRQ_EOF, see EOF. True, if = 1		host
26..24	SW<2..0>	May be used for software purposes. No effect on hardware.		host
23...16	CHECK	If the BMU is reading a descriptor owned by the BMU with a value different to 0x55, an interrupt IRQ_CHECK_TX is generated and no further descriptors are read (Supervisor state machine defaults to Idle).		host

Table 3: Transmit Descriptors

Bit	Name	Description	Address	Defined by
15..0	TBBC	Transmit Buffer Byte Count. Defines the number of data, which are read from host memory.		host
31..0	TXDSCR	Transmit Descriptor as defined by SUPERNET 3. Appended to transmit data of this buffer, if EOF is set. Note: Software has to ensure consistence with settings of byte counts for that frame.	base + 0x4	host, if EOF is set
31..0	TBADR	Transmit Buffer Address (Address in the host memory where the BMU gets data from)	base + 0x8	host
31..0	NTDADR	Next Transmit Descriptor Address	base + 0xc	host

Table 3: Transmit Descriptors

Bit	Name	Description	Address	Defined by
RBCTRL	**Receive Buffer Control register**		base	
31	Own_Bit	= 0 host is owner = 1 the BMU is owner After setting all other bytes of a descriptor, the host sets the Own_Bit to “1” for relinquishing the descriptor to the BMU. After writing all other bytes of a descriptor, the BMU sets the Own_Bit to “0” for relinquishing the descriptor to the host.		host BMU
30	STF	=1 indicates that with this data fragment (or frame) a new frame starts = 0 data fragment is not a start of frame fragment Start of Frame is coming with two functions: - Defined by the host: If set, the BMU is allowed to use this descriptor for the start of a received frame. If not set, the BMU does nothing but relinquishing the descriptor back to the host and reading the next descriptor. This way the BMU reads descriptors until one with STF set is found. - Defined by the BMU: Start of Frame is set, if the descriptor is holding the start of a received frame. Valid, when the Own_Bit is set to “0” by the BMU.		host BMU
29	EOF	End of Frame indicates that this is the last descriptor used by the BMU for this frame. If EN_IRQ_EOF and EOF are set, an interrupt IRQ_EOF is generated after relinquishing this descriptor. Valid, when Own is set to “0” by the BMU.		BMU
28	EN_IRQ_EOB	= 0 interrupt disabled = 1 interrupt enabled If EN_IRQ_EOB is set, an interrupt IRQ_EOB_RCV_1 is generated after relinquishing this descriptor.		host

Table 4: Receive Descriptor

Bit	Name	Description	Address	Defined by
27	EN_IRQ_EOF	= 0 disable interrupt IRQ_EOF = 1 enable IRQ_EOF, see EOF.		host
26	Dev_0	= 0 normal operation mode = 1 do not actually transfer data to the system memory		host
25..24	SW<1..0>	May be used for software purposes. No effect on hardware		host
23...16	CHECK	If the BMU is reading a descriptor owned by the BMU with a value different to 0x55, an interrupt IRQ_ERR_<xxx> is generated and no further descriptors are read (Supervisor state machine defaults to idle state).		host
15..0	RBBC	Receive Buffer Byte Count The host is defining the length of the buffer provided to the BMU. Max. length is restricted by the max. FDDI frame length. The BMU is rewriting the real number of bytes written to the buffer (valid, when the Own_Bit is set to “0” by the BMU)		host NIC
31..0	RFSW	Receive Frame Status Word as defined by SUPERNET 3 (including length). Valid, when own is set to “0” by the BMU and EOF is set.	base + 0x4	NIC
31..0	RBADR	Receive Buffer Address	base + 0x8	host
31..0	NRDADR	Next Receive Descriptor Address	base + 0xc	host

Table 4: Receive Descriptor

STF and EOF

The STF and EOF bits are required since a frame may be stored in several non-contiguous fragments in the host memory. To transmit this frame each fragment requires its own descriptor. The EOF information in transfer descriptors is forwarded to the SUPERNET 3. Additionally, the end of the frame transfer can be indicated by an interrupt.

When receiving frames, i.e. transferring a frame from the buffer memory to the host memory, the STF, EOF bits are used to guarantee that the first fragment of a frame is written to a new contiguous host memory area. For new frames or fragments containing the start of a frame, the BMU accepts only receive descriptors with STF set to “1”. Receive descriptors with STF = “0” are relinquished to the host and the next receive descriptor is read. The design idea was to enable driver software to write frame headers in the memory areas they want e.g. to the start of new contiguous buffers.

EN_IRQ_EOB

The design idea for the EN_IRQ_EOB interrupt was to give the driver software control over fragments from received frames e.g. the software provides only small buffers for the frame headers only. The BMU reads a receive descriptor with the STF set to “1” pointing to a small buffer area in the host memory. The first fragment, the frame buffer, is transferred, and the BMU reads the next descriptor. However, this descriptor is still owned by the host. Thus no more frame fragments are transferred. Driver software meanwhile interprets the frame header. Then the driver software sets up the receive descriptors, starting with the one just read by the BMU. With knowledge about the frame header driver software can provide space for the rest of the frame as appropriate. Then driver software relinquishes the descriptor(s) to the BMU and gives a Start <xxx> command the involved BMU. This causes the BMU to transfer the rest of the frame.

DEV_0

The receive descriptor’s DEV_0 bit is used to have the BMU reading this descriptor performing all the operations but without actually transferring data to the host memory. Setting the DEV_0 flag prevents that data are on the PCI bus, however the entire BMU mechanism is handled as usual, e.g. depending on the interrupt mask, a interrupt may occur.

Check

The Check bits are provided to control the integrity of the descriptors and the BMU transfer mechanism.

RFSW, TXDSCR

The RFSW (Receive Status Word) and the TXDSCR (Transmit Descriptor) contain the address information from or from the SUPERNET 3 - where to read or write the frame on the FDDI side.

The following is important for driver software:

☞ **Take care to write the 32 bit containing the Own_Bit at last in a single action thus relinquishing ownership to the BMU.**

3.6.2 BMU Operation

The BMU comprises four interacting state machines:

- ❏ Supervisor state machine
- ❏ Descriptor read state machine
- ❏ Descriptor write state machine
- ❏ Transfer state machine

Supervisor State Machine

The Supervisor state machine is issuing requests to the descriptor read and write state machines and the transfer state machines. These state machines are reporting the execution of their transaction to the supervisor.

The supervisor is requesting a descriptor read, if the command Start <xxx> is given or if the next descriptor is needed after servicing the current descriptor.

If the descriptor read state machine reads a descriptor owned by the BMU, the request for transferring data is issued to the transfer state machine.

If the transfer from/to the buffer is terminated, a descriptor write is requested.

Descriptor Read State Machine

The descriptor read state machine is loading the address counter with the current descriptor address, setting the number of bytes to be transferred to 16 and issues a request to perform bus master transfer.

Note: the 32-bit word containing the Own_Bit is read first.

Transfer State Machine

The Transfer SM loads the address counter with the current receive/transmit buffer address. (buffer in the host memory). It writes the number of bytes to be transferred to buffer byte counter and issues a request to perform bus master transfer of the actual data.

Descriptor Write State Machine

The descriptor write state machine is running up to two transfers:
If a fragment is received containing the end of a frame the descriptor write state machine first writes the Receive Frame Status Word (RFSW) in the descriptor.
For this the descriptor write state machine loads the address counter with the (current descriptor address + 4), sets the number of bytes to be transferred to 4 and issues a request to start bus master transfer. After this the descriptor write state machine has to relinquish the descriptor back to the host. Relinquishing is done in a second independent transfer.

Descriptor Write state machines in any of the four queues relinquish the descriptor back to the host. Writing the 32bit word containing the Own_Bit relinquishes the descriptor to the host:

The descriptor write state machine loads the address counter with the current descriptor address, sets the number of bytes to be transferred to 4 and issues a request to perform bus master transfer.

☞ Closing this transaction, **Current Descriptor Address** is loaded with **Next Descriptor Address**.

3.6.3 BMU Related Registers

The following registers are provided to control the operation of each BMU assigned to a queue:

- ❏ Receive Queue 1 register block
- ❏ Receive Queue 2 register block
- ❏ Asynchronous Transmit Queue register block
- ❏ Synchronous Transmit Queue register block

All these four blocks have the same structure (shown in the following table). Their location, addresses in the Control Register File is listed in chapter *8.0 Appendix A: Control Register File* on page 78

	Byte <3>	Byte <2>	Byte <1>	Byte <0>
<31..0>	Current Receive/Transmit Descriptor (for control purposes only)			
<31..0>				
<31..0>				
<31..0>				
<31..0>	Current Receive/Transmit Descriptor Address			
<31..0>	Current Address Counter			
<31..0>	Current Byte Counter			
<31..0>	BMU Control/Status Register			
<31..0>	Flag Register			
<31..0>	Test Register 1			
<31..0>	Test Register 2			
<31..0>	Test Register 3			

Table 5: Receive /Transmit Queue Register Structure

Current receive/transmit descriptors are provided in the register blocks for test purposes only.

The current receive/transmit descriptor address is written by the descriptor write state machine as described before. It is read by the descriptor read state machine. During driver initialization, after setting up the descriptor chains, the current receive/transmit descriptors must be initialized with the address of the first descriptor. Afterwards, the descriptor write state machine is setting the current descriptor address.

Current address counter and current byte counter are provided for the transmit state machine.

The only register in each block that must be written regularly during normal operation is the BMU Control/Status register. It contains the Start <xxx> flag. Writing the Start <xxx> flag causes the supervisor state machine to initiate reading of the current descriptor.

Note that there are four BMU Control/.Status Registers and four Flag Register (shown in the Table 7: *<xxx> Queue Flag Register*), one for each of the four queues. The register structure for each is identical, except for bit3. This location is reserved in the transmit queues, in the receive queues it is used to clear the parity interrupt.

Bit	Name	Description	Write	Read	Reset (SW)
31..22	Reserved				
21	Reset Descriptor Clear	Set/Clear Reset for Descriptors	exec	0b10	0
20	Reset Descriptor Set			0b01	1
19	Reset FIFO Clear	Set/Clear Reset for Descriptors	exec	0b10	0
18	Reset FIFO Set			0b01	1
17	Run HPI state machine	Reset state machine to Idle/Release state machine	exec	0b10	0
16	Reset HPI state machine			0b01	1
15	Run Supervisor state machine	Reset state machine to Idle/Release state machine	exec	0b10	0
14	Reset Supervisor state machine			0b01	1
13	Run Descriptor Read state machine	Reset state machine to Idle/Release state machine	exec	0b10	0
12	Reset Descriptor Read state machine			0b01	1
11	Run Descriptor Write state machine	Reset state machine to Idle/Release state machine	exec	0b10	0
10	Reset Descriptor Write state machine			0b01	1
9	Run Transfer state machine	Reset state machine to Idle/Release AM	exec	0b10	0
8	Reset Transfer state machine			0b01	1
7..5	Reserved				
4	Start <xxx>	Start Transfer of <xxx> Queue	exec	0	0

Table 6: BMU Control/Status Registers

Bit	Name	Description	Write	Read	Reset (SW)
3	In Receive Queues:	reserved			
	In Transmit Queues: CLR_IRQ_PAR	Clear Interrupt Parity	exec	0	0
2	CLR_IRQ_EOB	Clear Interrupt EOB	exec	0	0
1	CLR_IRQ_EOF	Clear Interrupt EOF	exec	0	0
0	CLR_IRQ_ERR	Clear Interrupt ERR	exec	0	0

Table 6: BMU Control/Status Registers

IRQ_ERR, IRQ_EOF, IRQ_EOB were already mentioned when describing the descriptors for the queues. The interrupt source can be found by reading the interrupt source register. Pending interrupts are cleared by writing the Control/Status Registers.

Generation of the Parity interrupt is described with the parity checking mechanisms of the "**SK-NET FDDI-<x>P**" (see section 3.11 *Parity Generation/Control* on page 42)

The sequence how to deassert the Reset of the various state machines is described in more detail in section 3.10 *Reset Behavior* on page 42

Before describing the effects of the Start <xxx> command and how the BMUs perform subsequent operations, the Queue Flag Register is shown for completeness. The meaning of the watermark will become clear when describing the NIC's FIFOs.

Bit	Name	Description	Write	Read	Reset (SW)
31..5	Reserved				
4..0	Watermark	Level for requesting data transfer (in 32bit words)	yes	aw	0

Table 7: <xxx> Queue Flag Register

☞ **For the transmit queues the Watermark must not be set to 0.**

3.6.4 Subsequent BMU Operations

The Start <xxxx> command initiates the BMU of a queue causing the supervisor state machine to initiate reading the current descriptor, doing transfer...

Transmit Queues

For the transmit queues things are obvious. Every time the driver software has data to transmit, the descriptors and the related buffers are set up and with one Start <xxx> command, the BMU is initiated.

The supervisor state machine in the BMU causes the descriptor read state machine to read the current descriptor and invokes the transfer state machine. Transfer is done and on completion the descriptor write state machine performs the following:

- ❏ The descriptor write state machine sets the Own_Bit of the current descriptor to “0”, relinquishing the current descriptor to the host.
- ❏ Then it overwrites the current descriptor address with the next descriptor address from the used descriptor.

After relinquishing the used buffer, the BMU goes on, causing the descriptor read state machine to read the current descriptor. ...

The BMU repeats this until the first descriptor not owned by the BMU is read. Then the BMU goes idle waiting for the next Start <xxx>.

Receive Queues

The rules how the BMUs behave are the same as for the transmit queues. However, if not triggering SUPERNET 3 interrupts or other events, driver software does not know when to give the Start <xxx> command.

If Start <xxx> is given the BMU will perform transfers reading descriptors until the frame is transferred. If the driver software does not make use of the STF, EOF, and EOB flags to perform controlled fragmentation and transfers, usually there will be enough descriptors available owned by the BMU.

Especially for the receive queues, usually it might happen that the BMU owns a descriptor but there are not data to transfer to the host memory. In this case the BMU rests in this state and as soon as there are data, the BMU transfers them to the host memory.

☞ **BMUs are started with a Start <xxx> command. They will go to idle state if they do not find a descriptor owned by the BMU**

☞ **IDLE BMUs do not poll for a changing Own_Bit. They must be invoked with a Start <xxx> command.**

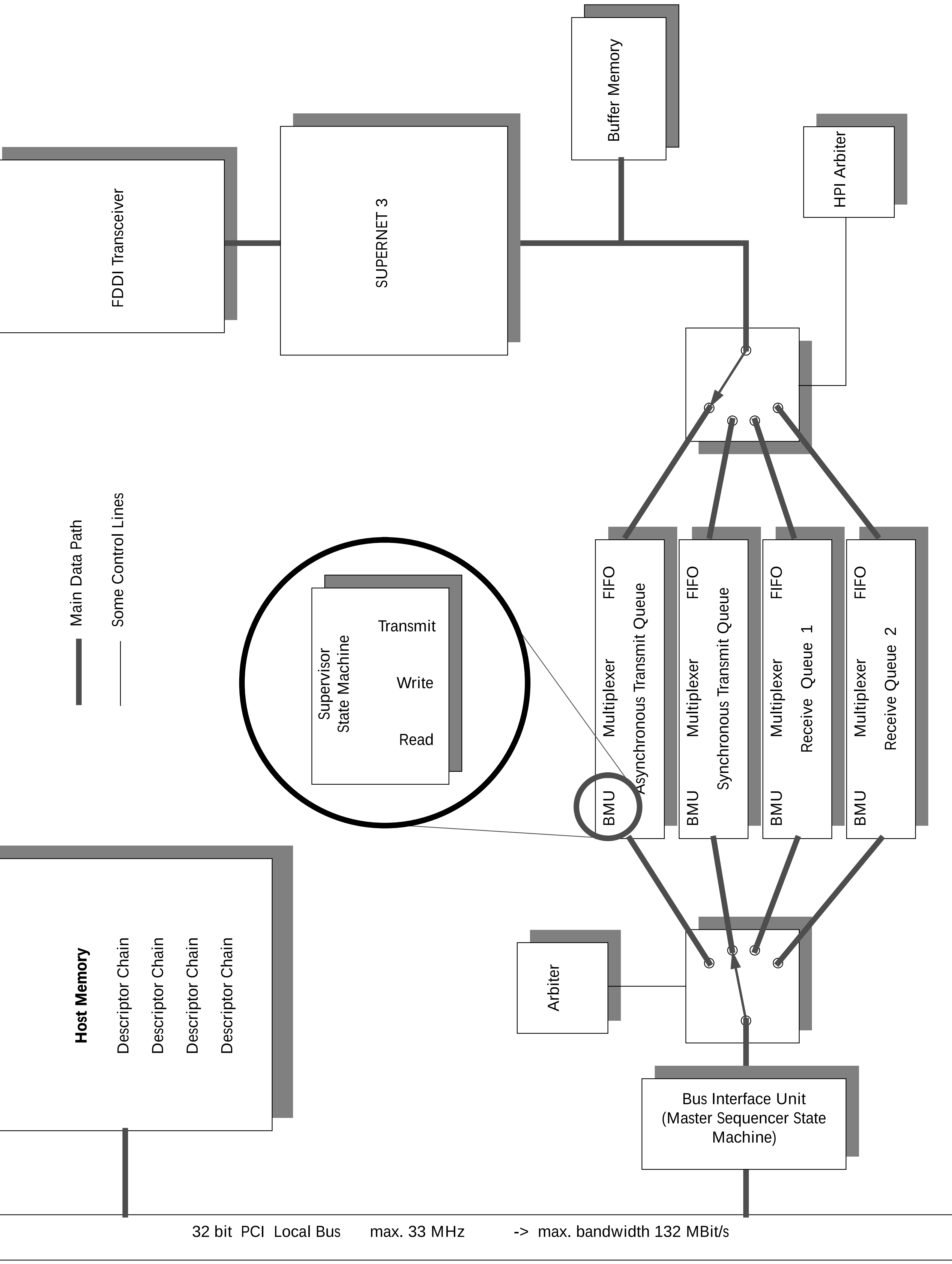
And important for receive BMUs:

☞ **BMUs owning a descriptor but having no data, wait for data and transfer them as data are available.**

Software may send Start <xxx> commands periodically. Except for small system load, giving Start <xxx> commands to the receive BMUs, while there’s no data, or sending more Start <xxx> commands than necessary does not matter. (The same is true for the transmit BMUs, however for transmitting frames the driver software controls how many frames and fragments it will send and ‘know’ when to send a Start <xxx> command.)

3.6.5 Bus Master Operation

The Master Sequencer State machine implemented in the ASIC handles the NIC’s bus master operations. It is controlled by giving an address and a guaranteed number of bytes to be transferred (plus minor additional information) from one the four queues.


FDDI Transceiver
SUPERNET 3
Buffer Memory
HPI Arbiter
Main Data Path
Some Control Lines
Supervisor State Machine
Transmit
Write
Read
BMU
Multiplexer
FIFO
Asynchronous Transmit Queue
BMU
Multiplexer
FIFO
Synchronous Transmit Queue
BMU
Multiplexer
FIFO
Receive Queue 1
BMU
Multiplexer
FIFO
Receive Queue 2
Host Memory
Descriptor Chain
Descriptor Chain
Descriptor Chain
Descriptor Chain
Arbiter
Bus Interface Unit (Master Sequencer State Machine)
32 bit PCI Local Bus max. 33 MHz -> max. bandwidth 132 MBit/s

If not already owner of the PCI bus, the Master sequencer state machine requests the bus. Ownership of the bus is released for several reasons:

- ❏ Number of bytes to transfer is zero
- ❏ Cache line size boundary is reached and number of bytes to be transferred is below cache line size.
- ❏ Latency counter expired

The default transfer is burst transfer, unless disabled by the Disable_Burst flag.

Hardware details: The following PCI bus cycle commands for transfer on the PCI bus are supported: Memory Write, Memory Write And Invalidate, Memory Read, Memory Read Line and Memory Read Multiple.
Memory Write and Invalidate is used instead of Memory Write, if the guaranteed number of bytes to be transferred is higher than Cache Line Size.
Memory Read Line is used instead of Memory Read, if the guaranteed number of bytes to be transferred is higher than one Cache Line Size.
If the guaranteed number of bytes to be transferred is higher than two Cache Line Sizes at least, Memory Read Multiple is used instead of Memory Read Line.
The command Memory Write And Invalidate, Memory Read Line and Memory Read Multiple may be disabled individually be setting the appropriate bits in Our_Register (Configuration Space).

A Target Retry is serviced by retrying the terminated cycle.

Software intervenience required: If a cycle is terminated by Target or Master Abort, this is reported to RT-ABORT or RMABORT (status register), interrupts IRQ_TRABORT or RMABORT are set and the master sequencer state machine is locked.
This has to be solved by resetting the state machine. (Reset_Master).

Dev_0

There is a specific mode of master accesses preventing transfers of unwanted frames to the host memory. This is signaled by the DEV_0 flag in the currently serviced descriptor.
In this mode, no real access to the PCI bus is requested, but the interface to the queues simulating ‘normal’ bus access.

The number of transferred bytes is reported back to the initiator on a per cycle base.

Since there are at least four queues that may initiate a bus master transfer an arbiter is needed to solve concurrent requests. Servicing one of the queues is reported to the arbiter by the sequencer state machine.

3.6.6 Arbiter

Requests from the queues for accessing the PCI bus are signaled as requests to the arbiter. Granting the bus to a queue is feed back, and interface signals are routed between the queue and the master sequencer.

Concurrent requests are serviced with a rotating priority scheme. Arbitration decisions are taken, while the bus is released for any reason. The queues are limiting the bus usage by the settings of their watermarks.

3.7 Data Transfer Block Diagram

The following diagram provides an overview over the state machines, and the involved logical and physical units. It's provided to show the data path and arrangement of control units along this path.

Transfer of data over the PCI bus and the involved control units had been described in the previous sections. The following will provide information about the Multiplexers, FIFOs, the HPI Arbiter, the SUPERNET and the PLC-S.

3.7.1 Multiplexers

The Multiplexers are provided for positioning of data in the right byte lanes on misaligned transfers. And if requested by the Byte_Swap flag, the multiplexers will swap data from little-endian representation (default) to big-endian representation.

As shown in the diagram there are four multiplexers, one for each queue. Multiplexers are controlled by the corresponding BMUs. During normal operation mode there are no other registers, flags that control the multiplexers, except for the Byte_Swap flag in Our_register (see Table 36: *Our_Register* on page 57).

For testing purposes the multiplexer position can be controlled. See chapter *9.0 Appendix B Testing the NIC* on page 84.

3.7.2 FIFOs

The four FIFOs are provided for a contiguous transfer of data between areas with different bandwidth characteristics: max. 132 MByte/sec on the PCI side versus max. 50 MByte/s at the SUPERNET 3's Host Processor Interface. Nevertheless the FIFOs are synchronized versus both the BMU clock on the PCI side and the Host Processor Interface (HPI) clock on the FDDI side.

The FIFOs are implemented in the ASIC. The FIFO width is 37 bit: 32 bit data + 4 parity bits + 1 tag bit. The FIFO depth is 34 words.

Accesses at the PCI side are 32 bit wide; accesses at the SUPERNET 3 side are 37bit wide. FIFOs are providing bus requests and length information to the state machines controlling access to the PCI bus.

Each FIFO may be cleared by its individual Reset_FIFO. For each FIFO there's a programmable watermark value, stored in the <xxx> Queue Flag register (see page 29). The watermark value indicates either the minimum number of 32 bit words available in the FIFO (receive FIFOs) or the number of free 32 bit words available to be filled (transmit FIFOs).

There are two types of FIFOs: Receive FIFOs in the receive queues and Transmit FIFOs.

Receive FIFOs

Each Receive FIFO is providing the following flags:

- ❏ *Allmost_FULL*, based on 32bit words
- ❏ *EOF*, if the FIFO is holding the end of frame (the tag bit is set
- ❏ *Watermark,* if watermark is reached. Watermark is counted in 32bit words.

A receive FIFO requests transfers, if either watermark is reached or EOF is set. The guaranteed number of available bytes if forwarded to the sequencer state machine. Once initiated, this number is decreased by each transferred byte. The initial value is the watermark in bytes or the real content of the FIFO, if EOF is set.
If a transfer is broken, it may be requested again if either watermark is reached or the EOF flag is set.

This limits the length of a transfer out of one receive queue to a maximum given by the watermark and forces the master frontend to PCI bus related state machines to lock on cache line size boundaries (if watermark is greater than a cache line size)

The watermark value defines the maximum burst size on the PCI bus. The watermark value should be higher than two cache line sizes (if feasible) with a maximum of 24.

The receive FIFOs are tracking the start and the end of a frame. The flags STF and EOF are written to the bit fields in a descriptor, if the first or last byte is clocked out of the FIFO.

If the last byte of a frame is clocked out of the FIFO, the receive status word is written to the descriptor and the EOF is reported.

Transmit FIFOs

Each Transmit FIFO is providing the following flags:

- *EMPTY*, based on 32bit words
- *EOF* , if the FIFO is holding an end of frame (tag bit is set)
- *Watermark*, if watermark (in 32 bit words) is reached.

A transmit FIFO requests transfer if there are free bytes in the FIFO. For transmit FIFOs the watermark value is the minimum number of free bytes in the FIFO. If the watermark value is exceeded, the guaranteed number of free bytes is forwarded to the PCI side state machines. Once initiated, the number is decreased by each transferred byte. The initial value is the watermark in bytes.

If transfer is broken, the transfer may be requested again, if the watermark value is again exceeded.

This limits the length of a transfer to one of the transmit queue to a maximum given by the watermark and forces the PCI side state machines to lock on cache line size boundaries (if watermark is greater than a cache line size).

The watermark value defines the maximum burst size on the PCI bus. The watermark value should be higher than two cache line sizes (if feasible) with a maximum of 24.

☞ **For the transmit queues the Watermark must not be set to 0.**

The transmit FIFOs are tracking the start and the end of a frame. After the last byte of a frame is clocked into the FIFO, the SUPERNET 3 transmit descriptor is appended. Tag bit are set for the SUPERNET 3 transmit descriptor and the last word.

While tagged words are in the FIFO, no further data are clocked into the FIFO. Thus, only bytes from one frame can be in a transmit FIFO.

3.7.3 Host Processor Interface (HPI)

The Host Processor Interface (HPI) as described here is implemented in the ASIC. With its two receive state machines and the two transmit state machines it interfaces the HPI of the SUPERNET 3. The HPI arbiter included in the HPI controls concurrent accesses to/from the FIFOs.

Receive State Machines

The HPI receive state machines are provided to keep the receive FIFOs filled. The receive state machine writes data out of the buffer memory to the receive FIFOs while

- ❏ Receive data are available (RDATA is set)
- ❏ FIFO is not full
- ❏ No End of Frame in FIFO (a tag bit was written to FIFO and not read out yet)
- ❏ Buffer memory bus is allocated

Note: the RXFBB <1..0> bits of mode register 2 in the SUPERNET 3 has to be set to 00 (first received byte stored to byte 0, LSB (mode register 2) has to be set to 0 (MSB is received first).

Transmit State Machines

The HPI transmit state machines are provided to keep the transmit FIFOs empty. The transmit state machine writes data out to the buffer memory while:

- ❏ There’s space remaining in the buffer memory’s transmit queue controlled by the SUPERNET 3 (QCTRL<2..0>)
- ❏ FIFO is not empty
- ❏ Buffer memory bus is allocated

☞ **If the SUPERNET 3 reports a transmit underrun condition, the related transmit state machine is locked. This can only be resolved by resetting this state machine.**

HPI Arbiter

The HPI arbiter controls concurrent accesses to/from the four queues.

On concurrent requests, the buffer memory is allocated to the receive and the transmit queue alternately.
Concurrent receive queues are serviced alternately. Concurrent transmit queues are serviced alternately, if the SUPERNET 3 is not transmitting. If the SUPERNET 3 is transmitting out of a queue, this queue is serviced with priority.

On concurrent requests a preemption counter is started, limiting the number of accesses of a running queue to 32. If the preemption counter expires or a queue is stopping transfers with their own rules, the memory buffer bus is allocated to the next requesting queues, following the arbitration scheme described above.

3.8 SUPERNET 3

AMD’s SUPERNET 3 chip is the FDDI controller of the “**SK-NET FDDI-<x>P**”. It comprises the two main controllers: The Medium Access Controller (formerly FORMAC+) and the PHY Layer Controller PLC-S.

Buffer Memory

The buffer management unit of the SUPERNET controls the 128 KByte buffer memory. The buffer memory is provided for temporary storing of frames. Operating in the tag mode, the buffer memory looks like an enhancement to the SUPERNET 3’s internal FIFOs.
The buffer memory has a depth of 32 K, 32 bit data width, 4 parity bits and 1 tag bit.

All 256 registers of the SUPERNET 3 are mapped into the Control Register FIle. PCI bus accesses are translated to SUPERNET 3 using it’s Node Processor Interface. All four interrupt lines MINTR<4..0>* of the SUPERNET 3 are routed to the NIC’s interrupt source register. The reset line RST* of the SUPERNET 3 is controlled by the NIC’s SW_Reset* line.

Data transfers from/to the NIC’s FIFOs to/from the Buffer Memory are running on the Host Processor Interface of the SUPERNET 3. This interface is clocked with 25 MHz (BMCLK) resulting in a max. transfer bandwidth of 50 MByte/sec for the Host Processor Interface.

3.8.1 PLC-S, DAS NIC

The additional PLC-S mounted on the DAS FDDI NICs, may be recognized by driver software from reading the DAS_Available Flag in the Control Register (DAS) register (Table 39: *Control Register (DAS)* on page 62).

The registers of the PLC-S are mapped into the Configuration Register File. The interrupt line INT* of the PLC-S is routed to the interrupt source register. The PLC-S’s reset line RST* is controlled by the NIC’s SW_Reset* line. Receive data lines from the additional PLC-S are routed to the ‘RB-Bus’ of the SUPERNET 3. The switching functions of the bypass connector are controlled through the ‘Control Register (DAS)’.

3.9 Initialization

There are three sources where the NICs registers get their initial values after power_on or reset of the network interface card. Later driver software should perform some initialization:

- ❏ There are hardwired initialization values defined in the ASIC
- ❏ Initialization values are read from the Flash EPROM (FPROM) after RST# is deasserted overwriting the hardwired initialization
- ❏ Some registers in the configuration space e.g. the base address registers are written by the PCI system software.
- ❏ Initialization by the software

3.9.1 Hardwired Values

All registers are set to some default value after power_on or RST# asserted on the PCI bus. If not tagged otherwise, the RESET value given in the tables throughout this manual is the hardwired reset value. If for some reasons, design alternatives, the NICs initialization may be customized, there’s a note saying that this initialization value may be rewritten out of the FPROM.

As described in the following any register value (except the FPROM loader start address) can be set to customized initialization value before PCI system software, the BIOS, configures the NIC.

3.9.2 Flash EPROM (FPROM)

The 128KByte Flash EPROM may be mapped in the memory address space with sizes of 16k, 32k, 64k or128k. The page size is defined by the Page Size<2..0> bits hold by Our_Register in the configuration space.

Default is mapping of the full 128KByte FPROM.

Mapping of the FPROM is controlled by the EN_FPROM flag in Our_Register. If this flag is not set, the ROM Base Address Register is not presented to the Control Register File and the Flash EPROM may be not mapped into the memory address space.

If mapped, the base address is defined in the Expansion ROM Base Address Register. The Expansion ROM Base Address Register is also holding the page size and the ROMEN flag which controls enabling of the Expansion ROM. The mapped page is selected by setting PAGE<3..0> in Our_Register..

The FPROM loader provides another mechanism than memory mapping to read data, especially configuration data from the FPROM.

The FPROM may contain Boot code or whatever but the last 16k sector is holding data for initialization of the NIC power_on or reset.

Reprogramming the FPROM

For programming of the Flash EPROM no additional 12V power supply and switching of the programming voltage is required. However

☞ **The FPROM must be programmed carefully, according to the manufacturer‘s algorithms[1]. Any deviation may cause failure of or damage to the FPROM.**

3.9.3 FPROM Loader

The FPROM loader is provided for the following tasks:

- ❏ Setting initialization values defined in the FPROM after booting/reset
- ❏ Reprogramming the FPROM
- ❏ Providing access to Boot Code (Expansion Boot ROM)

Loading Initialization Values

After a Reset from the PCI bus' RST# line, e.g. after starting the computer system, the Flash EPROM loader loads data -initialization values - from the FPROM into the Configuration Space Register or Control Register File.

Loading of the FPROM data is started by deassertion of the RST# line. FPROM data may also be used to set initialization values in the configuration space. But after starting the computer system the PCI system software starts configuring the entire system including the "**SK-NET FDDI-<x>P**".

1. If the AM29F010 FPROM is mounted on the board, detailed information can be found in: "Flash Memory Products : 1992/1993 Data Book/Handbook" by Advanced Micro Devices (AMD).

Therefore accesses to any resource of the NIC are terminated by Target Retry Cycles on the PCI bus while initialization values are loaded from the FPROM.

If started, the FPROM loader reads subsequent entries from the end of the 128KByte FPROM in decreasing order. Data and address, where to write data must be provided in the FPROM as shown in the table below. The FPROM loader reads 8 byte entries from the FPROM decreasing the FPROM address counter and writing the registers.

31..24	**23..16**		**15..8**	**7..0**	**Address**
key = 0x55	reserved	BE<3..0>	Address_upper	Address_lower	0x 1f ff c
Data<3>	Data<2>		Data<1>	Data<0>	0x1f ff 8

Table 8: Data Structure of Initialization Values in the FPROM

The key byte contains 0x55 for valid initialization data. If no valid key is read the FPROM loader terminates.

Address_upper and Address_lower contain the address where in the 2KByte Control Register File the subsequent data will be written. The loader is capable of accessing all registers in the Control Register File. (For registers not in the Configuration Space, their respective Reset must be cleared first.)

The Address column shows the FPROM address where the FPROM loader is reading data from. For the first access this is the last 4 byte word in the 128KByte FPROM at address 0x1FFFC.

The byte enable bits BE<3..0> indicate which of the data bytes in the entry are valid ones. The registers of the Control Register File may be written with 32bit, 16bit, or 8bit data words.

The FPROM address counter is initiated from the FPROM address register/-counter. Since the FPROM provides an 8bit access port, the address counter is initiated by the cards hardware with 0x1FFFF, the upper byte of the 128KByte is read first. (Nevertheless, the FPROM loader reads the FPROM in units of 8bytes.)

Programming the FPROM

For reprogramming the FPROM the FPROM address register and data register are provided.

Bit	Name	Description	Write	Read	Reset (HW)
FPROM Address Register/Counter					
31..17	Reserved				
16..0	Address Register/Counter	Defines 17-bit Flash FPROM address	yes	aw	0x1ffff
FPROM Data Register					
31..24	Reserved				
23..16	Reserved				
15..8	Reserved				
7..0	Data Port	Programming FPROM Data Port	exec	value	0
FPROM Loader Control Register					
31..12	Reserved				
11	Loader Test On	Testmode On/Off	exec	0b10	0
10	Loader Test Off			0b01	1
9	Loader Step	Decrement FPROM Address Counter.	exec	0	0
8	Loader Start	Starts loading of Flash EPROM at the location defined by the Address Register.	exec	0	0
7..0	Reserved				

Table 9: FPROM Address Register/Counter

☞ **The FPROM must be programmed carefully, according to the manufacturer‘s algorithms[1]. Any deviation may cause failure of or damage to the FPROM.**

The Loader Control Register is treated again in chapter *9.0 Appendix B Testing the NIC* on page 84.

3.9.5 Initialization by Software

Further initialization must be performed by the driver software, e.g. setting the interrupt mask, setting up the descriptor chains and announcing the first descriptor addresses to the BMU, ...

1. If the AM29F010 FPROM is mounted on the board, detailed information can be found in: “Flash Memory Products : 1992/1993 Data Book/Handbook” by Advanced Micro Devices (AMD).

Software must clear the various resets following the sequence

- ❏ Release SW_Reset
- ❏ Release Master_Reset
- ❏ Release HPI_SM_Reset
- ❏ Release Resets of all queue devices, which are intended to be used (Queue Control Registers)
  - Setup descriptors in system memory (at least for queues, which are intended to be used, at least first descriptor).
  - Release the resets of all components of the queues, which are intended to be used.
  - Initialize descriptors, Current Descriptor Address set to first descriptor in system memory.
  - Unlock queues.
- ❏ Initialize SUPERNET 3

The following initialization settings and decisions for the SUPERNET 3 should be done:

Mnemonics	Description	Recommended Setting
SNGLFRM	Single-Frame Receive Mode	This mode is not supported by SUPERNET3
FULL/HALF	Full duplex (if high)/Half duplex operation	Optional, both operation modes are supported by hardware
SELRA	Select RA Bus (if high) or RB Bus	Internal PHY (S-Port/A-Port) is routed to RA-Bus, the external PHY (B-Port) is routed to RB-Bus. **Note**: see also **MENDAS**.

Table 10: Initialization Mode Register 1

Mnemonics	Description	Recommended Setting
BMMODE		Set to tag mode (high) Default after Reset (SUPERNET 3)
CHKPAR	Check/generate parity on buffer memory data bus	Set to high, enabled
PARITY	Parity type	Set to high, even parity
ENHSRQ	Enable host requests	Set to high, enable
ENNPRQ	Enable node processor DMA requests	Set to low, disable

Table 11: Initialization Mode Register 2

Mnemonics	Description	Recommended Setting
SYMCTL		Not supported by SUPERNET 3
LSB	Least Significant Byte	Set to high, LSB is transmitted/received first
RXFBB<1..0>	Receive Frame Byte Boundary	Set to 0x00
AFULL<3..0>	Almost Full	If the number of free 32bit words in a transmit queue is less than twice the value programmed to **AFULL**, transfers to this queue are stopped. Set to 3 (6 free 32bit words) Note: A transmit queue must not be shorter than the longest expected transmit frame plus **AFULL** x 2 (in 32bit words).

Table 11: Initialization Mode Register 2

Mnemonics	Description	Recommended Setting
MENRS	Enable enhanced Receive status encoding	Set to high, enabled
MENXS	Enable enhanced Transmit status encoding	Set to low, disabled
MENAFULL	Enable enhanced QCTRL encoding for A_FULL	Set to low, disabled
MENQCTRL	Enable enhanced QCTRL encoding	Set to low, disabled
MENDAS	Enable DAS connections	Set to high, if DAS extension available (check DAS Available flag) Note: see also SELRA
MENTRCMD	Enable the Asynchronous queue 1 to ...	Set to low, disabled
MENFLOC<1..0>	Enables the FC location within the frame data long word.	set to 0x00

Table 12: Initialization Mode Register 3

Mnemonics	Description	Recommended Setting
RTHR		set receive threshold to the maximum of 240 bytes
XTHR		set transmit threshold to 0

Table 13: Frame Threshold Register

Additionally, enable parity on external PLC-S connection: Set ENA_-PAR_CHK in the PLC_CNTRL_A register of internal and external PLC-S.

3.10 Reset Behavior

There are several Reset lines provided on the “**SK-NET FDDI-<x>P**” NIC, e.g. the descriptor state machines can be reset individually. For the main Reset lines there exists a reset hierarchy described in the following

The power on reset HW_Reset is setting SW_Reset. SW_Reset is setting all individual Resets including SUPERNET 3 Reset and PLC-S Reset.

☞ **While SW_Reset is asserted, only the SW_Reset in the Control Register may be accessed.**

SW_Reset must be cleared, before clearing all individual Resets (even Master_Reset and HPI_SM_Reset in the Control Register).

Note the following when asserting SW_Reset

In order to guarantee the minimum pulse width of SW_Reset, a dummy read access should be inserted between subsequent write accesses setting and clearing SW_Reset. This read access may read the Control/Status Register.

3.11 Parity Generation/Control

The “**SK-NET FDDI-<x>P**” controls data integrity nearly on the entire data path through the Network Interface Card (NIC) by means of parity checking. The following describes mainly the check points and the parity checks by the ASIC.

Parity generation, checking and reporting for the data pathes between SUPERNET 3 and PLC-S (DAS extension) are provided by the means of these devices (see manuals).
Parity generation, checking and reporting for data read/written from/to memory buffer are provided by the SUPERNET 3 and the ASIC respectively. Parity for write data are generated by both devices, parity of read data is checked by both devices on their own accesses.

The SUPERNET is reporting parity errors by its own means.

If the ASIC is detecting a parity error reading buffer memory data, interrupts IRQ_PAR_RCV_1 or IRQ_PAR_RCV_2 are generated, running operations are continued.

Parity on the PCI bus is generated, checked and reported following the PCI specification.

Read data parity is generated for all read accesses to adapter resources (Flash EPROM, SUPERNET 3, PLC-S, ASIC-Registers) in the ASIC. Write data parity is checked for all write accesses to adapter resources (Flash EPROM, SUPERNET 3, PLC-S, ASIC-Registers) in the ASIC.

Address parity is checked for all address phases running on the bus.

If a write data parity error is detected, Parity_Error is set. Bus signal PERR* is asserted, if Parity_Report_Response_Enable is set.

If an address parity error is detected, Parity_Error is set. Bus signal SERR* is asserted and Signaled_Error is set, if SERR* enable and Parity_Report_Response_Enable are set.

Bus Master Access

For bus master accesses the following is performed:

Write data parity is generated for all write accesses to system memory. Read data parity is checked for all read accesses from system memory. Address parity is generated for all address phases generated on the bus.

If a read data parity error is detected, Parity_Error is set. The bus signal PERR* is asserted and Data_Parity_Error_Detected is set, if Parity_Report_Response_Enable is set.

If on a write access PERR* is sampled asserted, Parity_Error is set. Data_Parity_Error_Detected is set, if Parity_Report_Response_Enable is set.

If Data_Parity_Error_Detected is set, interrupt IRQ Master_Error is set. If Parity_Error is set, interrupt IRQ_Status is set (see also in the Interrupt Register).

4.0 Configuration Space

The “**SK-NET FDDI-<x>P”** supports the 256Byte configuration space as defined by the PCI specification revision 2.1.
Some of the configuration capabilities, e.g. User Defined Features (UDF), are foreseen by the hardware, but will not be used in the first series of “**SK-NET FDDI-<x>P”** NICs. Some features, e.g. the Subsystem Vendor ID and the Subsystem Device ID are foreseen but are not applicable to the “**SK-NET FDDI-<x>P**” NIC. (Subsystem information may be used e.g. by other manufacturers that use the ASIC or the NIC with their own drivers.)

The following table depicts the layout of the configuration space:

	Register				
	Byte 4	**Byte 3**	**Byte2**	**Byte 1**	**Address**
Header Portion	Device ID		Vendor ID		0x00
	Status		Command		0x04
	Class Code			Revision ID	0x08
	BIST	Header Type	Latency Timer	Cache Line Size	0x0c
	Base Address (1st)				0x10
	Base Address (2nd)				0x14
	Reserved				0x18
	Reserved				0x1c
	Reserved				0x20
	Reserved				0x24
	Reserved				0x28
	Subsystem ID		Subsystem Vendor ID		0x2c
	Expansion ROM Base Address				0x30
	Reserved				0x34
	Reserved				0x38
	Max_Lat	Min_Gnt	Interrupt Pin	Interrupt Line	0x3c
Device Dependent Region	Our_Register				0x40
	Reserved				0x44 ... 0xfc

Table 14: Configuration Registers

All multi-byte numeric fields follow little-endian order, if accessed by PCI configuration cycles.

Write operations to reserved or unimplemented registers are completed normally on the bus and the data are discarded. Read operations to reserved or unimplemented registers are completed normally on the bus and a data value of 0 is returned.

The configuration space is read/written by the PCI system software to determine a conflict-free configuration of the NIC and providing a seamless integration in the PCI bus system. The configuration space is also mapped in the Control Register File and can be accessed via I/O and Memory access.
In the I/O space the lower half[1] is mapped to block 3. In the Memory space the lower half is mapped to address (0x180 + memory base address). These mappings may be useful to verify the contents of the configuration space. In test mode the configuration information can also be written.
However, the recommended way to access the configuration space is using BIOS functions as described in the PCI BIOS specification.

Now the meaning of different registers in the configuration space is described in more detail.

4.0.1 Device-, Vendor, Subsystem-ID, Class Code, RevisionID

The NIC is uniquely identified as "**SK-NET FDDI-<x>P**" by the Vendor ID and the Device ID. The Class Code bytes provide more information about the type of PCI device.

The PCI SIG has allocated the Vendor ID 0x1148 to SysKonnect.

Bit	Name	Description	Write	Read	Reset value
15..0		Identifies SK as manufacturer of the "SK-NET FDDI-<x>P"	ne	0x1148	0x1148

Table 15: Vendor ID

(The Vendor ID reset value may be redefined by a value specified in the FPROM)

The Device ID register is a 16-bit register that uniquely identifies the NIC within SK's product line.

Bit	Name	Description	Write	Read	Reset value
15..0		Identifies the "SK-NET FDDI-<x>P" within SK's product line.	ne	0x4000	0x4000

Table 16: Device ID

1. The "upper half" of the 256 byte configuration space is reserved.

(The Device ID reset value may also be redefined by a value specified in the FPROM)

The Subsystem Vendor ID register may be used for customizing OEM versions. The subsystem Vendor ID is assigned by the PCI SIG.

Bit	Name	Description	Write	Read	Reset value
15..0		Identifies the subsystem vendor. 0 indicates, that the device is not supporting subsystem identification.	ne	0	0

Table 17: Subsystem Vendor ID

(The Subsystem Vendor ID reset value may also be redefined by a value specified in the FPROM)

The Subsystem ID register may be used for customizing OEM versions.

Bit	Name	Description	Write	Read	Reset value
15..0		Identifies the subsystem. 0 indicates, that the device is not supporting subsystem identification.	ne	0	0

Table 18: Subsystem ID

(The Subsystem ID register's reset value may also be redefined by a value specified in the FPROM)

The Class Code Register is used to identify the generic function of the NIC. The register is broken in three byte-size fields.

Bit	Name	Description	Write	Read	Reset value
7..0		Specifies the programming interface. Fixed value.	ne	0	

Table 19: Programming Interface (lower byte of class code)

Bit	Name	Description	Write	Read	Reset value
7..0		Identifies the network controller as an FDDI controller. Fixed value.	ne	0x02	

Table 20: Sub Class Register (middle byte of class code)

Bit	Name	Description	Write	Read	Reset value
7..0		Broadly classifies the function of the NIC as network controller. Fixed value.	ne	0x02	

Table 21: Base -Class Register (upper byte of class code)

The Revision identifier specified by the manufacturer. Revision 1.0 is coded as 0x10.

Bit	Name	Description	Write	Read	Reset value
7..0		Specifies the NIC‘s revision number/Rev. 1.0.	ne	0x10	0x10

Table 22: Revision ID Register

(The Revision ID reset value may also be redefined by a value specified in the FPROM)

4.0.2 Command Register

The Command Register is used to control the overall functionality of the "**SK-NET FDDI-<x>P**" NIC. It controls the NIC's ability to generate and response to PCI bus cycles.

Bit	Name	Description	Write	Read	Reset value
15 ..10	Reserved				
9	FBTEN	Fast Back-to-Back enable. If =1, fast back-to-back transactions to different agents are allowed (The NIC operating as master is running fast back--to-back write cycles) If =0, fast back-to-back transactions are only allowed to the same agent (The NIC operating as master is not running fast back-to-back write cycles)	yes	aw	0
8	SERREN	**SERR#** enable, controls the assertion of **SERR#** pin. If =1, **SERR**# is enabled If =0, **SERR**# is disabled	yes	aw	0
7	ADSTEP	Address/Data stepping. Fixed value, NIC does not use address/-data stepping.	ne	0	
6	PERREN	Parity Report Response Enable. If =1, Parity error reporting is enabled	yes	aw	0
5	VGASNOOP	VGA palette snoop. Fixed value.	ne	0	
4	MWIEN	Memory Write and Invalidate Cycle enable if = 1, Memory Write and Invalidate Cycles enabled, if = 0, Memory Write must be used instead.	yes	aw	0

Table 23: Command Register

Bit	Name	Description	Write	Read	Reset value
3	SCYCEN	Special Cycle enable. Fixed value = 0, NIC ignores all Special Cycle operations.	ne	0	
2	BMEN	Bus Master enable if = 1, bus master accesses are enabled if = 0, bus master accesses are disabled.	yes	aw	0
1	MEMEN	Memory Space access enabled if = 1, memory accesses are responded, if = 0, memory accesses are not responded.	yes	aw	0
0	IOEN	IO Space access enabled if = 1, IO accesses are responded, if = 0, IO accesses are not responded.	yes	aw	0

Table 23: Command Register

To logically disconnect the NIC from all PCI bus cycles except configuration cycles, a value of ZERO should be written to this register.

4.0.3 Status Register

The Status Register contains status information for the PCI bus related events.

Reads to this register behave normally, writes are slightly different in that bits can be reset, but not set. A bit is reset whenever the register is written, and the data in the corresponding bit location is a ONE (‘sh’ = special handling).

Bit	Name	Description	Write	Read	Reset value
15	PERR	Parity Error. Is set whenever a parity error is detected (data or address), even if parity error handling is disabled (**PERREN**).	sh	value	0
14	SERR	Signaled SERR#. Is set whenever a address parity error is detected and both, SERREN and PERREN are enabled.	sh	value	0
13	RMABORT	Received Master Abort. Is set when a master transaction is terminated with a master abort sequence.	sh	value	0
12	RTABORT	Received Target Abort. Is set when a NIC’s master transaction is terminated with a Target Abort sequence.	sh	value	0

Table 24: Status Register

Bit	Name	Description	Write	Read	Reset value
11		Reserved			
10..9	DEVSEL	**DEVSEL#** Timing. Fixed value = 01b (medium), **DEVSEL#** is asserted two CLK periods after **FRAME#** is asserted.	ne	01b	
8	DATAPERR	Data Parity Error detected. Set, if a data parity error is detected running master cycles and **PERREN** is set.	sh	value	0
7	FB2BCAP	Fast Back-to-Back Capable Fixed value = 1, target is capable of accepting fast back-to-back transactions	ne	1	
6	UDF	UDF supported	ne	0	0
5..0		Reserved			

Table 24: Status Register

(The Status Register's reset value may also be redefined by a value specified in the FPROM)

4.0.4 Cache Line Size, Latency, Header Type, and BIST

The following describes the register settings either assigned by the PCI system, e.g. the cache line size or by the NIC announcing a capability.

Cache Line Size

Specifies the system cache line in units of 32-bit words. The value is assigned by the PCI system software.

The cache line size is restricted to be a power of 2. The most significant "1" in this register is used to set the cache line size. Any other "1's" are ignored. The NIC supports cache line sizes of 2, 4, 8, or16 32bit words.

The NIC as bus master uses this field as criteria for starting of transfers to or from to complete cache lines with the Memory Write and Invalidate, Read Line and Read Multiple commands and to know when to disconnect burst accesses at cache line boundaries.

Bit	Name	Description	Write	Read	Reset value
7..0		The cache line size is restricted to be a power of 2. The most significant "1" in this register is used to set the cache line size. Any other "1's" are ignored.	yes	aw	0

Table 25: Cache Line Size Register

(The Cache Line Size registers' reset value may be redefined by a value specified in the FPROM -- not recommended)

Latency Timer Register

Specifies the maximum time the NIC can continue with bus master transfers after the system arbiter has removed GNT*.The time is specified in units of PCI bus clocks.

The working copy of the timer will start counting down when the NIC asserts FRAME* for the first time during a bus mastership period. The timer will freeze at ZERO. When the timer is ZERO and GNT* is deasserted by the system arbiter, the NIC will finish the current data phase and then immediately release the bus.

Bit	Name	Description	Write	Read	Reset value
7..0		max. resting time for bus master transfers after *GNT** was deasserted in units of PCI bus clocks	yes	aw	0

Table 26: Latency Timer Register

(The Latency Timer registers' reset value may be redefined by a value specified in the FPROM -- not recommended)

Header Type:

The Header Type Register describes the format of the PCI Configuration Space locations 0x10 to 0x3c thus identifying a device to be a single or a multi function device.

Bit	Name	Description	Write	Read	Reset value
7		Single function/multi function device. Fixed value = 0, the NIC is a single function device.	ne	0	
6..0		PCI Configuration Space layout. Fixed value = 0, the layout of the PCI Configuration Space locations 0x10 to 0x3c is as shown in the table above.	ne	0	

Table 27: Header Type Register

Providing a Built-in Self Test is an optional capability for PCI devices. The "**SK-NET FDDI-<x>P**" does not provide BIST.

Bit	Name	Description	Write	Read	Reset value
7..0		Fixed value, the NIC is using the interrupt pin INTA*. Fixed value.	ne	0x01	

Table 28: Interrupt Pin Register

4.0.5 Interrupt, Min_Gnt, Max_Latency

The Interrupt Line Register is used to communicate interrupt line routing information. POST[1]software writes the routing information into this register as it initializes and configures the system.

The value in this register tells which input of the system interrupt controller(s) the devices's interrupt pin is connected to. Device drivers and operating systems can use this information to determine priority and vector information.

The Interrupt Line Register is not modified by the NIC. It has no effect on the operation of the device. It's a special service for software engineers writing driver or diagnosis software.

Bit	Name	Description	Write	Read	Reset value
7..0		input port of the system interrupt controller(s)	yes	aw	0

Table 29: Interrupt Line Register

Bit	Name	Description	Write	Read	Reset value
7..0		Fixed value, the NIC is using the interrupt pin INTA*.	ne	0x01	

Table 30: Interrupt Pin Register

The Min_Gnt read-only register specifies the NIC's desired settings for Latency Timer value.

Bit	Name	Description	Write	Read	Reset value
7..0		The value specifies in units of 1/4 microseconds the burst period needed by the NIC assuming a clock rate of 33MHz. (32 words x 33.33ns) = 1.07us 1.07us/.0.25us = 4	ne	0x04	0x04

Table 31: Min_Gnt Register

(The Min_Gnt registers' reset value may be redefined by a value specified in the FPROM)

1. Power On Self Test is run by the PCI system.

The Max_Lat read-only register specifies the NIC’s desired settings for Latency Timer value.

Bit	Name	Description	Write	Read	Reset value
7..0		The value specifies in units of 1/4 microseconds how often the NIC needs to gain access to the PCI bus assuming a clock rate of 33MHz. (32 words x 4)/12.5MB/s = 10µs 10µs/0.25µs = 40	ne	0x28	0x28

Table 32: Max_Lat Register

(The Max_Lat registers’ reset value may be redefined by a value specified in the FPROM)

4.0.6 Base Address Register

The base addresses for the I/O space and the memory space are assigned by the PCI system software. The assigned values are written to the 32bit base address registers.

I/O and memory space base address registers have different values in bit 0 of the register: a “0” indicating memory space, a “1” indicating I/O space.

The PCI specification reserves more bytes for Base Address Register information than the “**SK-NET FDDI-<x>P**” requires. The unused Base Address Registers are read as **“0”.**

The first Base Address Register is a 32-bit register that determines the location of the NIC in memory space, if memory mapping is used.

Bit	Name	Description	Write	Read	Reset value
31...11	MEMBASE	Memory base address most significant 21 bits.	yes	aw	0
10..4	MEMSIZE	Memory size requirements. Fixed value, indicates memory space requirement of 2048 bytes.	ne	0	
3	PREFEN	Prefetchable. Fixed value, indicates that prefetching is not allowed. (Memory Write Byte Merging is not tolerable)	ne	0	
2..1	Memory Type	Memory type. Indicates, that this base register is 32 bit wide and that mapping can be done anywhere in the 32 bit memory space. Memory type may be reloaded from the Flash EPROM (further memory types).	ne	0	0
0	MEMSPACE	Memory space indicator. Fixed value, indicates, that this base address register describes an memory base address.	ne	0	

Table 33: Base Address Register (1st)

The Base Address Register 1’s reset value should be redefined by a value specified in the FPROM.

The second Base Address Register determines the location of the NIC in the host's I/O space. If EN_IO Mapping is disabled, this location is treated like reserved locations.

Bit	Name	Description	Write	Read	Reset value
31...8	IOBASE	I/O base address most significant 24 bits.	yes	aw	0
7...2	IOSIZE	I/O size requirements. Fixed value. Reading back a value of "0" in bits 7...2 indicates I/O space requirement of 256 bytes.	ne	0	
1	Reserved	Reserved location.	ne	0	
0	IOSPACE	I/O space indicator. Indicating that this Base Address Register describes an I/O base address.	ne	1	0

Table 34: Base Address Register (2nd)

(The Base Address Register 2's reset value may be redefined by a value specified in the FPROM - not recommended)

4.0.7 Expansion ROM Base Address

The Expansion ROM Base Address Register is a 32 bit register that determines the base address and size information of optional expansion ROM.

If EN_EPROM is disabled, this location is treated like a *reserved* register.

Bit	Name	Description	Write	Read	Reset value
31...17	Rombase	ROM base address significant 15 bits.	yes	aw	0

Table 35: Expansion ROM Base Address Register

Bit	Name	Description	Write	Read	Reset value
16..14	Rombase/size	Treated as ROMBASE or ROMSIZE depending on settings of Pagesize.	yes ne	aw 0	0
13...11	Romsize	ROM size requirements. Reading back a value of “0” indicates memory space requirement of 16k bytes or higher.	ne	0	
10...1	Reserved	Reserved location.	ne	0	
0	ROMEN	Address decode enable. Read/write accessible. If ROMEN = 0, the devices Expansion ROM address space is disabled. If ROMEN = 1 and MEMEN =1 (Command Register), the devices Expansion ROM address space is enabled.	yes	aw	0

Table 35: Expansion ROM Base Address Register

(The Expansion ROM Address Register ‘s reset value may be redefined by a value specified in the FPROM))

4.0.8 Our_Register

The 32-bit register called *Our_Register* is the only location in the device dependent region that the “**SK-NET FDDI-<x>P**” uses.

Most of the switches in Our_Register are not intended for use at run time but during NIC configuration or initialization.

Intention:

This register may be used as a user-configurable register later. To make it user configurable a PCI Configuration File (PCF) according to version 2.1 of the PCI specification must be provided and the UDF (User Defined Feature) bit must be set appropriately. If the UDF bit indicates that there are user configurable options, a PCI configuration utility can prompt the user to step through the offered options and select.

For example the user may then decide whether to make use of Boot code or of the I/O mapping capability.

Bit	Name	Description	Write	Read	Reset value
31...29	Reserved				
28 .. 27	Patch_Dir<1..0>	Defines the type of the pins Ext_Patchs<1..0>. 1 = output 0 = input	yes	aw	0
26 ... 25	Ext_Patchs<1..0>	These bits are routed to the ASIC’s pins for future external configuration options.	yes	aw	0
24	EN_Boot	Boot enable, for software purposes only if EN_BOOT = 0, boot with expansion ROM code if EN_Boot = 1, don’t boot with expansion ROM code.	yes	aw	0
23	EN_IO_Mapping	Controls mapping of the Control Register File to the IO space (manufacturing option). If disabled (EN_IO_Mapping = 0), any address decoding for IO accesses is disabled (see also Base Address Register (2nd)).	ne	0	0
22	EN_FPROM	Controls mapping of the Flash EPROM to the memory space. If disabled (EN_FPROM = 0), any address decoding for memory accesses is disabled (see also ROM Base Address Register).	ne	1	1
21..20	Pagesize	Pagesize/Flash EPROM Defines the size, which is mapped to the system (see Romsize/Rombase). 3 = 128k 2 = 64k 1 = 32k 0 = 16k	ne	3	3

Table 36: Our_Register

Bit	Name	Description	Write	Read	Reset value
19..16	Page_Reg<2..0>	Page Register Selects the page of the Flash EPROM space, which is mapped to the system, if Pagesize is lower than 128k. 0 = page 0, 1 = page 1 ... 7 = page 7 May be undefined on booting without hardware reset.	yes	aw	0
15	Reserved				
14	Force_BE	Assert all BE<3..0>* on Master-Reads	yes	aw	0
13	DIS_MRL	Disable command **Memory Read Line**	yes	aw	0
12	Dis_MRM	Disable command **Memory Read Multiple**	yes	aw	0
11	Dis_MWI	Disable command **Memory Write and Invalidate**	yes	aw	0
10	Disconnect at CLS	Disconnect at Cache Line Size Boundary (command **Memory Write And Invalidate**)	yes	aw	0
9	Burst Disable	Burst disable. Disables bursting on master transfers as a patch for systems not supporting burst transfers.	yes	aw	0
8	Byte_Swap	Defines byte ordering. Byte Swap to big endian is only performed on data transfers from/to the FIFOs. If Byte Swap = 0, NIC is set to little endian. If Byte Swap = 1, NIC is set to big endian.	yes	aw	0
7..4	Skew/DAS<3..0>	Skew Control, DAS Extender	ne	0	0
3..0	Skew/Base<3..0>	Skew Control, Base	ne	0	0

Table 36: Our_Register

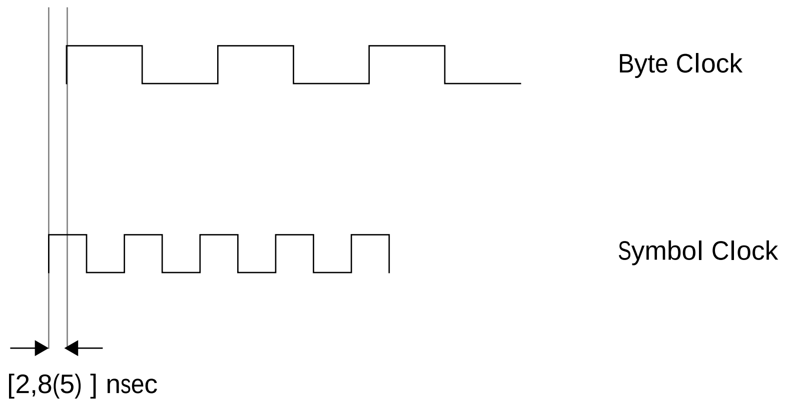
Our_Register ‘s reset value should be redefined by a value specified in the FPROM. Via FPROM initialization EN_Boot and EN_IO_Mapping should be enabled.

The bits <28 .. 25> are provided for future use. You may also look at them as reserved bits.

Skew Bits

The skew control bits for the FDDI ports: for A - and B - ports on DAS or for S-ports on SAS versions are provided

On the card, incoming/outgoing FDDI symbols are transformed into bytes and vice versa. Symbols are clocked at 25 MHz, the bytes (= 2 symbols after encoding) are clocked at 12.5 MHz. AMD specifies that the skew must be in the range [2nsec, 8(5) nsec] for successful interaction of MAC and PLCs.



The skew of the clocks depends on many factors, e.g. the capacitive load attached to the corresponding signal lines. These factors are unpredictable during the card design stage but the effective skew can be measured at the first cards manufactured.
To guarantee that the card meets the specifications, the 12.5 MHz signals[1] can be delayed. The delay is specified by setting the bits <7..4> and <3..0> respectively.

Each skew value is set to 0x00 by the card’s hardware after power-up or card reset from the PCI bus. (Setting to 0x00 means no delay at all.)

Non-predictable effects resulting in shifting the skew can be corrected by setting the bits in the FPROM to an optimum value as measured at running NICs.

1. On the DAS version there are two byte clocks: between FORMAC Plus and PLC and between FORMAC Plus and PLC2 respectively.

5.0 Control Register File - The Entire Register Stack-

The Control Register File comprises all registers that are accessible to the host, independent of their physical location. It also comprises the Configuration Space register treated in the previous chapter.

The Control Register File may be mapped in the 32bit I/O space as well as in the 32 bit Memory space. A base address for each of these spaces must have been assigned by the PCI system software, then any word can be addressed.

5.1 Addressing the Control Register File

The Control Register File is distributed over an address range spanning 2KByte. Thus, the Control Register File is mapped directly into the host memory area. Register in the Memory space can be accessed directly at

<Register Address> = < Memory Base Address> +
<relative address in the Control Register File>.

The I/O space of the "**SK-NET FDDI-<x>P**" is mapped in the host's I/O space spans 256 Byte. To make all registers of the Control Register File available via I/O accesses a Register Address Port (RAP) is provided. The entire 2KByte address space of the Control Register File is divided in 16 × 128 Byte blocks. Giving the block number <15..0> and the address relative to the begin of a block <0x80 .. 0x00> enables to address any register in the 2 KByte Control Register File.

Additionally you must know that block 0 is permanently mapped to the lower half - the lower 128 Bytes out of 256 Bytes I/O addresses mapped in the host's I/O space - and that the block specified by the data in the Register Address Port (RAP) is mapped to the upper half - the upper 128 Bytes.

5.1.1 Register Address Port (RAP)

The RAP is four bits at the relative address 0x00 in block 0 of the Control Register File. Thus writing to the I/O base address specifies the RAP value.

Bit	Name	Description	Write	Read	Reset (SW)
31..4	Reserved		ne	0	
3..0	RAP	Specifies one out of block 0 ...15, which is mapped to the upper half of the 256-Byte I/O range. 0=block 0, ...0xf=block 15	yes	aw	0

Table 37: Register Address Port (RAP)

To address any register in the Control Register File via I/O access, write the block number to the I/O base address and access

<Register Address> = <I/O base address> + 0x80 +
<register address relative to the begin of the block>

(Adding 0x80 means switching to the upper half of the 256 Byte I/O space).

A tabular overview of the control register file's contents is given in chapter *8.0 Appendix A: Control Register File* on page 78. The current chapter will discuss single registers and the meaning of single bits.

5.2 Various Registers

The following deals with various registers in the Control Register File unless discussed in other sections.

5.2.1 Control Register

The Control Register provides Reset control, i.e. it enables bringing the NIC or single state machines in RESET state and it enables clearing the from the RESET state to the normal operation mode.

Bit	Name	Description	Write	Read	Reset
7..6		reserved commands	ne	0	
5 4	HPI state machine Reset Clear HPI state machine Reset Set	Set/Clear HPI state machine Reset. executed, if appropriate bit is set to 1. SW_Reset also sets the HPI state machine Reset (section 3.10 *Reset Behavior* on page 42)	exec	0b10 0b01	0 1 (SW)
3 2	Master Reset Clear Master Reset Set	Set/Clear Master Reset. executed, if appropriate bit is set to 1. SW_Reset also sets the Master Reset (section 3.10 *Reset Behavior* on page 42)	exec	0b10 0b01	0 1 (SW)
1 0	SW_Reset Clear SW_Reset Set	Set/Clear SW_Reset. executed, if appropriate bit is set to 1. If SW_Reset is set, all internal and external devices are in their reset state. To provide an internal minimum pulse width of 1.6us, clearing SW_Reset is suppressed within 1.6us after SW_Reset Set. Write cycles to SW_Reset Clear within the 1.6us recovery time, are terminated with a Target Retry (no impact on software).	exec	0b10 0b01	0 1 (HW)

Table 38: Control/Status Register

The SW_Reset SET brings the entire NIC in reset state including the SUPERNET 3. It is the SUPERNET 3 chipset that requires a minim duration of 1.6 µsec for the reset signal being asserted.

☞ **In order to guarantee the minimum pulse width of SW_Reset, a dummy read access should be inserted between subsequent write accesses setting and clearing SW_Reset. This read access may be read the Control/Status Register.**

5.2.2 Control Register (DAS)

The Control Register (DAS) provides information/control over the additional second FDDI port mounted on DAS NICs. The only control function however is inserting/removing the DAS in/from the ring by switching the external optical bypass.

Bit	Name	Description	Write	Read	Reset (SW)
7..4		reserved			
3	DAS_Available	Single Attachment Station flag. 1= DAS Extension available. 0= no DAS Extension available.	ne	value	value
2	Bypass_Available	Bypass Status. 1= external bypass installed. 0= no external bypass installed. (It is read as 1 on SAS NICs)	ne	value	value
1	Bypass_Insert	Bypass Insert/Remove	exec	0b10	0
0	Bypass_Remove			0b01	1

Table 39: Control Register (DAS)

5.2.3 LED Register

The usage of LED's is software controlled, however during design the intention was that

- ❏ The yellow LED in the middle on indicates that a driver is loaded but there is no connection established yet.
- ❏ The green LEDs indicate that the FDDI ring is operational. (On SAS NICs there's only one green LED on the left side.)

Bit	Name	Description	Write	Read	Reset (SW)
7...6		reserved			
5	LED<2>On	LED (left, green) On/Off	exec	0b10	0
4	LED<2> Off			0b01	1
3	LED<1>On	LED (middle, yellow) On/Off	exec	0b10	0
2	LED<1> Off			0b01	1
1	LED<0>On	LED (right, green) On/ Off	exec	0b10	0
0	LED<0> Off			0b01	1

Table 40: LED Register

5.2.4 Interrupt Source Register

The interrupt source register indicates the source of an interrupt reported bye the "**SK-NET FDDI-<x>P**" NIC. If set to ONE, an interrupt from the corresponding internal source is pending.

Each interrupt is maskable by the Interrupt Mask register. All not masked interrupts are 'or' -ed and popagated to the bus interrupt line INTA#.
Despite masking, an interrupt from a masked source can be read the interrupt source register.

Interrupts are cleared/disabled as stated in the description of the related interrupt source, i.e. for clearing SUPERNET 3 interrupts refer to the SUPERNET 3 documentation.

Bit	Name	Description	Write	Read	Reset (SW)
31...25	Reserved				
24	IRQ Status	IRQ Status Exception Set, if PERR, RMABORT, RTABORT or DATAPERR are set.	ne	value	0
23	IRQ Master Error	IRQ Master Error detected on master accesses Set, if DATAPERR, RTABORT or RMABORT are set.	ne	value	0
22	IRQ_Timer	IRQ_Timer	ne	value	0
21	IRQ_RTM	IRQ_RTM	ne	value	0
20	IRQ_PHY_DAS	IRQ_PHY_DAS fixed value = 0, if no DAS Extension available	ne	value	0
19	IRQ_SUPERNET_3<4>	IRQ_SUPERNET_3<4>	ne	value	0
18	IRQ_SUPERNET_3<3>	IRQ_SUPERNET_3<3>/IRQ_PHY (SUPERNET2)[1]	ne	value	0
17	IRQ_SUPERNET_3<2>	IRQ_SUPERNET_3<2>/IRQ_MAC<2> (SUPERNET2)	ne	value	0
16	IRQ_SUPERNET_3<1>	IRQ_SUPERNET_3<1>/IRQ_MAC<1> (SUPERNET2)	ne	value	0
15	IRQ_PAR_RCV_1	Interrupt Parity Error/Receive Queue 1	ne	value	0

Table 41: Interrupt Source Register

Bit	Name	Description	Write	Read	Reset (SW)
14	IRQ_EOB_RCV_1	Interrupt End Of Buffer/Receive Queue 1	ne	value	0
13	IRQ_EOF_RCV_1	Interrupt End Of Frame/Receive Queue 1	ne	value	0
12	IRQ_CHCK_RCV_1	Interrupt encoding error of descriptor for Receive Queue 1	ne	value	0
11	IRQ_PAR_RCV_2	Interrupt Parity Error/Receive Queue 2	ne	value	0
10	IRQ_EOB_RCV_2	Interrupt End Of Buffer/Receive Queue 2	ne	value	0
9	IRQ_EOF_RCV_2	Interrupt End Of Frame/Receive Queue 2	ne	value	0
8	IRQ_CHCK_RCV_2	Interrupt encoding error of descriptor for Receive Queue 2	ne	value	0
7		reserved			
6	IRQ_EOB_TX_AS	Interrupt End Of Buffer/Async. Queue	ne	value	0
5	IRQ_EOF_TX_AS	Interrupt End Of Frame/Async. Queue	ne	value	0
4	IRQ_CHCK_TX_AS	Interrupt encoding error of descriptor for Async. Queue	ne	value	0
3		reserved			
2	IRQ_EOB_TX_S	Interrupt End Of Buffer/Sync. Queue	ne	value	0
1	IRQ_EOF_TX_S	Interrupt End Of Frame/Sync. Queue	ne	value	0
0	IRQ_CHCK_TX_S	Interrupt encoding error of descriptor for Sync. Queue	ne	value	0

Table 41: Interrupt Source Register

1. During an early design stage, backward compatibility to the SUPERNET 2 chipset was intended.

5.2.5 Interrupt Mask Register

Each bit position defines whether the dedicated interrupt is propagated to the Interrupt Line INTA*.

If set to ONE, interrupt is enabled.

Bit	Name	Description	Write	Read	Reset (SW)
31...25		reserved			
24	EN_IRQ Status	Enable IRQ Status Exception	yes	aw	0
23	EN_IRQ Master Error	Enable IRQ Master Error	yes	aw	0
22	EN_IRQ_Timer	Enable IRQ Timer	yes	aw	0
21	EN_IRQ_RTM	Enable IRQ RTM	yes	aw	0
20	EN_IRQ_PHY_DAS	Enable IRQ_PHY_DAS	yes	aw	0
19	EN_IRQ_SUPERNET_3<4>	Enable IRQ_SUPERNET_3<4>	yes	aw	0
18	EN_IRQ_SUPERNET_3<3>	Enable RQ_SUPERNET 3<3>/IRQ_PHY (SUPERNET2)	yes	aw	0
17	EN_IRQ_SUPERNET_3<2>	Enable SUPERNET 3<2>/IRQ_MAC<2> (SUPERNET2)	yes	aw	0
16	EN_IRQ_SUPERNET_3<1>SUPERNET	Enable IRQ_SUPERNET_3<1>/IRQ_MAC<1> (SUPERNET2)	yes	aw	0
15	EN_IRQ_PAR_RCV_1	Enable Interrupt Parity Error/Receive Queue 1			
14	EN_IRQ_EOB_RCV_1	Enable Interrupt End Of Buffer/Receive Queue 1	yes	aw	0
13	EN_IRQ_EOF_RCV_1	Enable Interrupt End Of Frame/Receive Queue 1	yes	aw	0
12	EN_IRQ_CHCK_RCV_1	Enable Interrupt encoding error of descriptor for Receive Queue 1	yes	aw	0
11	EN_IRQ_PAR_RCV_2	Enable Interrupt Parity Error/Receive Queue 2			

Table 42: Interrupt Mask Register

Bit	Name	Description	Write	Read	Reset (SW)
10	EN_IRQ_EOB_RCV_2	Enable Interrupt End Of Buffer/Receive Queue 2	yes	aw	0
9	EN_IRQ_EOF_RCV_2	Enable Interrupt End Of Frame/Receive Queue 2	yes	aw	0
8	EN_IRQ_CHCK_RCV_2	Enable Interrupt encoding error of descriptor for Receive Queue 2	yes	aw	0
7		reserved			
6	EN_IRQ_EOB_TX_AS	Enable Interrupt End Of Buffer/Async. Queue	yes	aw	0
5	EN_IRQ_EOF_TX_AS	Enable Interrupt End Of Frame/Async. Queue	yes	aw	0
4	EN_IRQ_CHCK_TX_AS	Enable Interrupt encoding error of descriptor for Async. Queue	yes	aw	0
3		reserved			
2	EN_IRQ_EOB_TX_S	Enable Interrupt End Of Buffer/Sync. Queue	yes	aw	0
1	EN_IRQ_EOF_TX_S	Enable Interrupt End Of Frame/Sync. Queue	yes	aw	0
0	EN_IRQ_CHCK_TX_S	Enable Interrupt encoding error of descriptor for Sync. Queue	yes	aw	0

Table 42: Interrupt Mask Register

5.2.6 MAC-Address Registers

These registers are holding the globally unique MAC-Address (physical address) of the NIC. They are loaded at POWER_ON RESET from the Flash EPROM.

Bit	Name	Description	Write	Read	Reset (HW)
		MAC-Address Registers			
31..24	MAC-<7>	Mac-Address, Byte 7	ne	value	0
23..16	MAC<6>	Mac-Address, Byte 6	ne	value	0
15..8	MAC<5>	Mac-Address, Byte 5	ne	value	0

Table 43: MAC Address Registers

Bit	Name	Description	Write	Read	Reset (HW)
7..0	MAC<4>	Mac-Address, Byte 4	ne	value	0
31..24	MAC<3>	Mac-Address, Byte 3	ne	value	0
23..16	MAC<2>	Mac-Address, Byte 2	ne	value	0
15..8	MAC<1>	Mac-Address, Byte 1	ne	value	0
7..0	MAC<0>	Mac-Address, Byte 0	ne	value	0

Table 43: MAC Address Registers

For SysKonnect NICs the MAC- address starts with byte<0> = 0x00 ,byte<1> =0x00, byte<3> = 5A0, ...

5.2.7 Interface Type Register

This register is holding the type of interface. It is loaded from the Flash PROM on power_on reset when the PCI bus RST# line is asserted.

Bit	Name	Description	Write	Read	Reset (HW)
31..16	Reserved				
15..8	PMD	PMD type	ne	value	0
7..0	Connector	Connector type	ne	value	0

Table 44: Interface Type Registers

(In testmode the Interface Type Register may be written.)

The PMD type and connector type information is coded according to the TP-PMD specification.

PMD type	Code letter	ASCII	SMT PMD-Class
Original PMD	P	0x50	multimode
LCF-PMD	L	0x4c	low-cost-fiber
TP-PMD (STP)	S	0x53	twisted pair
TP-PMD (UTP)	U	0x55	twisted pair

Table 45: PMD Type Coding

Connector type	Code letter	ASCCI
MIC	M	0x4d
LCF-MIC SC type	C	0x43
UTP MIC	J	0x4a
STP MIC	D	0x44

Table 46: Connector Type Coding

5.2.8 SUPERNET 3

All 256 SUPERNET 3 registers are mapped in a subsequent range on 32bit word boundaries.

The SUPERNET 3 registers **MUST** be accessed with a width of at least 16 bit at 32bit word boundaries. Other types of accesses are treated like accesses to reserved locations (not forwarded to the SUPERNET 3).

Bit	Name	Description	Write	Read	Reset (SW)
31..16		reserved			
15..0	SUPERNET <x>	SUPERNET3 Register<x>	see SUPERNET 3 data sheet		

Table 47: SUPERNET 3 Example Register

5.2.9 DAS Extension PLC-S

The PLC-S Registers **MUST** be accessed with a width of at least 16 bit at 32bit word boundaries. Other types of accesses are treated like accesses to reserved locations (not forwarded to the PLC-S).

Bit	Name	Description	Write	Read	Reset (SW)
31..16	Reserved	Reserved			
15..0	PLC-S<x>	PLC-S Register<x>	see PLC-S data sheet		
			ne, if no DAS Extension available	0xffff, if no DAS Extension available	

Table 48: DAS Extension PLC-S Example Register

5.2.10 Timer

The Timer is a programmable 32-bit downcounter with a resolution of 80ns (derived from the FDDI clock, Tmax = 344s) for usage as fixed timebase.

Bit	Name	Description	Write	Read	Reset (SW)
31..0	Timer_Init_Value		yes	aw	0
31..0	Timer		yes (for tests only)	value	0
Timer Control/Test					
31..11	Reserved				
10	Timer_Test_On	Testmode ON/OFF	exec	0b10	0
9	Timer_Test_Off	Testmode ON/OFF	exec	0b01	1
8	Timer_Step	Timer decrement (Testmode)	exec	0	

Table 49: Timer Registers

Bit	Name	Description	Write	Read	Reset (SW)
7..3	Reserved				
2	Timer_Start	Start/Stop Timer	exec	0b10	0
1	Timer_Stop			0b01	1
0	Timer_Clear_IRQ	Clear Timer Interrupt	exec	0	

Table 49: Timer Registers

Setting the command Timer_Start loads the timer with the Timer_Init_Value and starts counting. Each time the timer reaches ZERO or if the timer is loaded with ZERO, an interrupt IRQ_Timer is generated, and the timer is reloaded with the Timer Init Value

Reaching ZERO or loaded with ZERO the timer generates an interrupt IRQ_Timer and is reloaded with Timer_Init_Value.

The IRQ_Timer interrupt is cleared by setting the command Timer_Clear_-IRQ. The command Timer_Clear_IRQ overrides a concurrent internal interrupt (guaranteeing IRQ edges).

☞ **In order to prevent IRQ pulses, command Timer_Clear_IRQ should only be set, if IRQ_Timer is pending or if the timer is stopped.**

The timer may be stopped by setting the command Timer_Stop.

While the timer is running, read values may be undefined.

5.2.11 Watchdog

The **Watchdog** is a programmable 32-bit downcounter with a resolution of 80ns (derived from FDDI clock, Tmax = 344s) offering a timeout resetting the Network Interface Card (NIC), if not retriggered by software in time.

Bit	Name	Description	Write	Read	Reset (SW)
31..0		Watchdog_Init_Value	yes	aw	0
31..0		Watchdog	yes (for tests only)	value	0
31..11		reserved			
10	Watchdog_Test_On	Testmode On/Off	exec	0b10	0
9	Watchdog_Test_Off			0b01	1
8	Watchdog_Step	Watchdog decrement	exec	0	
7..4		reserved			

Table 50: Watchdog

Bit	Name	Description	Write	Read	Reset (SW)
3	Watchdog_Alarm	1 = Watchdog has reached ZERO	ne	value	0 Reset (HW)
2	Watchdog_Start	Start/Stop Watchdog	exec	0b10	0
1	Watchdog_Stop			0b01	1
0	Watchdog_Clr_Alarm	Clear Watchdog Alarm	exec	0	

Table 50: Watchdog

Setting the command Watchdog_Start loads the watchdog with Watchdog_Init_Value and starts counting.

Reaching ZERO or loaded with ZERO the watchdog stops, sets Watchdog_Alarm and resets the card by asserting SW_RESET*.

Watchdog_Alarm is cleared by setting the command Watchdog_Clear or on PCI bus signal RST# asserted.

The watchdog may be stopped by setting the command Watchdog_Stop.

While the watchdog is running, read values may be undefined.

5.2.12 Restricted Token Monitor

The Restricted Token Monitor RTM is a programmable 32-bit downcounter with a resolution of 80ns (derived from FDDI clock, Tmax = 344s) offering a timeout for the FDDI ring being in Restricted Token Mode too long.

Restricted Token Mode is monitored by tracking the Receive Status Lines RS<5,4,3,2,0> of the SUPERNET 3 for tokens captured/passed.

RTM_Status is set, if the Receive Status lines are indicating ‘Capture Token - Restricted’ or ‘Pass Token - Restricted’.

RTM_Status is reset, if the Receive Status lines are indicating ‘Capture Token - Nonrestricted’ or ‘Pass Token - nonrestricted’.

Bit	Name	Description	Write	Read	Reset (SW)
31..0		RTM Init Value	yes	aw	0
31..0		RTM	yes (for tests only)	value	0
31..11		reserved			

Table 51: Restricted Token Monitor

Bit	Name	Description	Write	Read	Reset (SW)
10	RTM_Test_On	Testmode On/Off	exec	0b10	0
9	RTM_Test_Off			0b01	1
8	RTM Step	RTM decrement (Testmode)	exec	0	
7..4		reserved	ne	0	
3	RTM Status	1 = Restricted Token Mode 0 = Nonrestricted Token Mode	ne	value	0 by RTM Stop also
2	RTM Start	Start/Stop RTM	exec	0b10	0
1	RTM Stop			0b01	1
0	RTM Clear IRQ	Clear IRQ	exec	0	

Table 51: Restricted Token Monitor

Setting the command RTM_Start loads the RTM with the RTM_Init_Value and enables counting.

If enabled, RTM is counting, while Restricted Token Mode is monitored.

If enabled, RTM is loaded with RTM _Init_Value, Nonrestricted Token Mode is monitored.

When reaching ZERO or loaded with ZERO, the RTM stops counting and generates an interrupt IRQ_RTM. IRQ_RTM is cleared by setting the command RTM_Clear_IRQ. The command RTM_Start re-enables counting)**.**

Monitoring may be stopped by setting the command RTM_Stop

While the RTM is running, read values may be undefined.

6.0 Electrical Interfaces

This chapter describes the pin assignment of the cards electrical interfaces:

- ❏ The bus connector interfacing the PCI bus
- ❏ Mini DIN 6 receptacle for external electrical bypass

6.1 PCI Bus Connector

The “**SK-NET FDDI-<x>P**” card requires either a 3,3V or a 5V PCI bus slot. The following signals are used by the network interface card.

AD <31..0>	C/BE<3..0>#	PAR	FRAME#
TRDY#	IRDY#	STOP#	DEVSEL#
IDSEL	LOCK#	PERR#	SERR#
REQ#	GMT#	INTA#	CLK
RST#	TDI/TDO	PRSNT1#	PRSNT 2#
GND	+ 5V	+3.3 V	

Table 52: PCI Bus Signals

The TDI/TDO pins are hardwired. The NIC does not support boundary scan test.

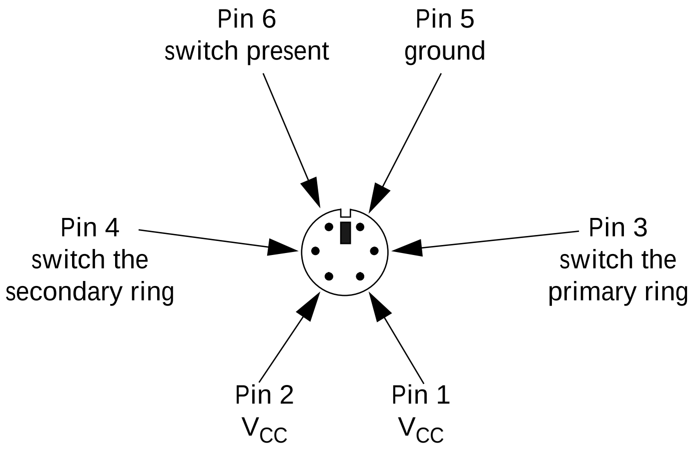
PRSNT1# and PRSNT2# indicate the device’s power requirement category: max. 7.5 W for the SAS version, max 15W for the DAS version.

6.2 Mini DIN 6 Receptacle

Power supply and control signals for the external optical bypass are provided through the Mini DIN 6 receptacle mounted on the piggyback board. The table below shows the pin assignment:

Pin	Signal		
1	V_{CC}		
2	V_{CC}		
3	primary ring	high	Bypass off
		low	Insert on
4	secondary ring	high	Bypass off
		low	Insert on
5	ground		
6	switch present (low)		

Table 53: Pin Assignment MiniDin6



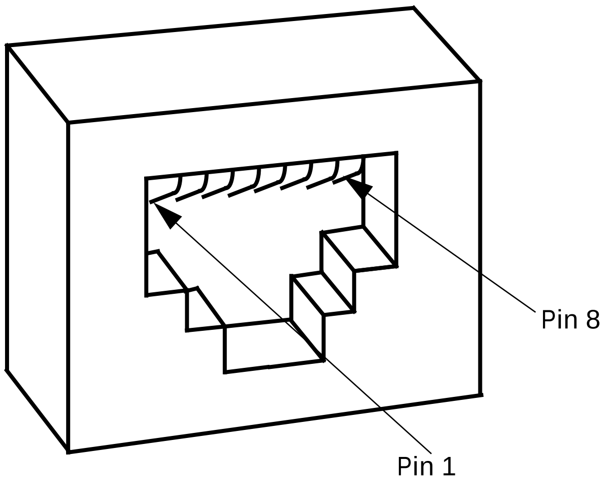
☞ **VCC (Pin 1 and Pin 2) is protected by an 0.5 A SMD fuse. Currents exceeding 0.5 A will damage the bypass controller.**

6.3 Copper Interfaces Pin Assignment

The “**SK-NET FDDI-TP**” is connected to the Category 5 FDDI cabling plant via an RJ45 receptacle.The following table lists the assignment of signals and pins for the RJ45 receptacle mounted on the NIC:

Pin	Signal
1	Transmit +
2	Transmit -
3	—
4	—
5	—
6	—
7	Receive +
8	Receive -
Shell	GND

Table 54: TP-PMD Interface Pin Assignment



RJ45 connector

7.0 Technical Data

The table below lists some important technical data describing the "**SK-NET FDDI-<x>P**" network interface card.

Bus Interface	32bit interface to max. 33MHz PCI Local Bus achieving a max. transfer rate of 132MByte/s
Network Interface	Compatible with the FDDI - ANSI X3T12[1] specifications and FDDI Standards Series
LAN Controller	AMD's SUPERNET 3
RAM	128-KByte static CMOS
FLASH Memory mapped as	128 Kbytes Expansion Boot ROM
Shared Memory	2KByte, including all programmable resources
I/O Addresses	265 Byte Block offers access to a 2KByte I/O space via Register Address Port (RAP)
Interrupts	INTA#
Timer	programmable timer 80nsec resolution
Watchdog	
Power Dissipation	to be determined
Operating Temperature	10° C — 40° C
Storage Temperature	-20° C — 60° C
Relative Humidity (non-condensing)	30% — 80% (operation) 10% — 90% (storage)
Dimensions	Dimensions are those of a PCI short length add-in board. Including bracket dimensions are: 12.6 cm × 18.6 cm 4.98 inch × 7.40 inch

Table 55: Technical Data

1. ANSI X3T12 includes now the former ANSI X3t9.5 committee.

7.1 Standards Compliance

For functionality "**SK-NET FDDI-<x>P**" complies with the international/industry standards — e.g. the ANSI X3T9.5 - FDDI standards - ISO standards, and the PCI Local Bus specifications series. Detailed references are listed within this manual or in the corresponding manuals of e.g. the chipset manufacturers.

Additionally, the "**SK-NET FDDI-<x>P**" network interface card is designed to meet the following standards and specifications for safety, Electro-Magnetic Compatibility (EMC), ...

Safety	
EN60950 (+A1,A2) - IEC 950: 1986 (+A1,A2)	to be verified
UL 1950	to be verified
CSA C22.2	to be verified
CB Certification	to be verified
EMC:	
EN 55022 (1988) - CISPR-22 (1985) Class B (for the fiber version **SK-NET FDDI-FP** and **SK-NET FDDI-LP**)	to be verified
EN 55022 (1988) - CISPR-22 (1985) Class A (for the copper version **SK-NET FDDI-UP)**	to be verified
EN 50082-1 (1992) **IEC 801-2 (1984)** **IEC 801-3 (1984)** **IEC 801-4 (1988)**	to be verified
FCC Class A	to be verified

Table 56: Standards Compliance

8.0 Appendix A: Control Register File

The Control Register File is mapped in the host’s I/O and memory space. To address the register the memory address is given in the “Address” column of the following table and the I/O block in the block column.

For memory address:

<register address> = <Address from “Address” column> +
<Memory base address>

For I/O address:

First specify the block number in the Register Address Port RAP (section 5.1.1 *Register Address Port (RAP)* on page 60)

<register address> = <I/O base address> + 0x80 +
<relative address of the register within the block>

Byte<3>	Byte<2>	Byte<1>	Byte<0>	Address	Block
Reserved	Reserved	Reserved	Register Address Port (RAP)	0x0000	0
Test Control Reg.	LED Register	Control Reg. (DAS)	Control Reg.	0x0004	
Interrupt Source Register				0x0008	
Interrupt Mask Register				0x000c	
Reserved	Reserved	SUPERNET 3 Register <0x0> . . SUPERNET 3 Register <0x3>		0x0010 . . 0x001c	
Reserved	Reserved	SUPERNET 3 Register <0x3c> . . SUPERNET 3 Register <0x3f>		0x0020 . . 0x002c	
Reserved	Reserved	SUPERNET 3 Register <0x60> . . SUPERNET 3 Register <0x67>		0x0030 . . 0x004c	
Reserved	Reserved	SUPERNET 3 Register <0x80> . . SUPERNET 3 Register <0x83>		0x0050 . . 0x5c	
Reserved	Reserved	SUPERNET 3 Register <0x90> SUPERNET 3 Register <0x91>		0x0060 0x0064	
Reserved	Reserved	SUPERNET 3 Register <0x8a>		0x0068	
Reserved	Reserved	SUPERNET 3 Register <0x10>		0x006c	
BMU Control/Status Register (Receive Queue 1)				0x0070	
BMU Control/Status Register (Receive Queue 2)				0x0074	
BMU Control/Status Register (Asynchronous Transmit Queue)				0x0078	
BMU Control/Status Register (Synchronous Transmit Queue)				0x007c	
Block Window Register <31..0> Note: If RAP = 1, read values are ZERO, write cycles have no effect				0x0080	1

Table 57: Control Register File

Byte<3>	**Byte<2>**	**Byte<1>**	**Byte<0>**	**Address**	**Block**
MAC<3>	MAC<2>	MAC<1>	MAC<0>	0x0100	
MAC<7>	MAC<6>	MAC<5>	MAC<4>	0x0104	
Reserved	Reserved	PMD Type	Connector Type	0x0108	
EPROM<3>	EPROM<2>	EPROM<1>	EPROM<0>	0x010c	
EPROM Address Register/Counter				0x0110	
Reserved	Reserved	Reserved	EPROM Data Port	0x0114	
Reserved	Reserved	Loader Test	Loader Control	0x0118	
Reserved				0x011c	
Timer Init Value				0x0120	
Timer				0x0124	
Reserved	Reserved	Timer Test	Timer Control	0x0128	2
Reserved				0x012c	
Watchdog Init Value				0x0130	
Watchdog				0x0134	
Reserved	Reserved	Watchdog Test	Watchdog Con-trol	0x0138	
Reserved				0x013c	
RTM Init Value				0x0140	
RTM				0x0144	
Reserved	Reserved	RTM Test	RTM Control	0x0148	
Reserved				0x014c	
Reserved Block <11...0>					
Reserved Block 2 <11...0>				0x0150	
Configuration Register File (lower half)				0x0180	3

Table 57: Control Register File

Byte<3>	Byte<2>	Byte<1>	Byte<0>	Address	Block
Receive Queue 1					4
Current Receive Descriptor				0x0200	
				0x0204	
				0x0208	
				0x020c	
Current Receive Descriptor Address				0x0210	
Current Address Counter				0x0214	
Current Byte Counter				0x0218	
BMU Control/Status Register				0x021c	
Flag Register				0x0220	
Test Register 1				0x0224	
Test Register 2				0x0228	
Test Register 3				0x022c	
Reserved				0x0230 ... 0x023c	
Receive Queue 2					
Current Receive Descriptor				0x0240	
				0x0244	
				0x0248	
				0x024c	
Current Receive Descriptor Address				0x0250	
Current Address Counter				0x0254	
Current Byte Counter				0x0258	
BMU Control/Status Register				0x025c	
Flag Register				0x0260	
Test Register 1				0x0264	
Test Register 2				0x0268	
Test Register 3				0x026c	
Reserved				0x0270 ... 0x027c	

Byte<3>	Byte<2>	Byte<1>	Byte<0>	Address	Block
Async. Transmit Queue					5
Current Transmit Descriptor				0x0280	
				0x0284	
				0x0288	
				0x028c	
Current Receive Descriptor Address				0x0290	
Current Address Counter				0x0294	
Current Byte Counter				0x0298	
BMU Control/Status Register				0x029c	
Flag Register				0x02a0	
Test Register 1				0x02a4	
Test Register 2				0x02a8	
Test Register 3				0x02ac	
Reserved				0x02b0 ... 0x02bc	
Sync. Transmit Queue					
Current Transmit Descriptor				0x02c0	
				0x02c4	
				0x02c8	
				0x02cc	
Current Transmit Descriptor Address				0x02d0	
Current Address Counter				0x02d4	
Current Byte Counter				0x02d8	
BMU Control/Status Register				0x02dc	
Flag Register				0x02e0	
Test Register 1				0x02e4	
Test Register 2				0x02e8	
Test Register 3				0x02ec	
Reserved				0x02f0 ... 0x02fc	

Byte<3>	Byte<2>	Byte<1>	Byte<0>	Address	Block
Reserved	Reserved	External PLC-S Register <0x00> . . External PLC-S Register <0x1f>		0x0300 . . 0x037c	6
Reserved	Reserved	DAS PLC-S Register <0x00> . . DAS PLC-S Register <0x1f>		0x0380 . . 0x03fc	7
Reserved	Reserved	SUPERNET 3 Register <0x00> . . SUPERNET 3 Register <0x1f>		0x0400 . . 0x047c	8
Reserved	Reserved	SUPERNET 3 Register <0x20> . . SUPERNET 3 Register <0x3f>		0x0480 . . 0x04fc	9
Reserved	Reserved	SUPERNET 3 Register <0x40> . . SUPERNET 3 Register <0x5f>		0x0500 . . 0x057c	10
Reserved	Reserved	SUPERNET 3 Register <0x60> . . SUPERNET 3 Register <0x7f>		0x0580 . . 0x05fc	11
Reserved	Reserved	SUPERNET 3 Register <0x80> . . SUPERNET 3 Register <0x9f>		0x0600 . . 0x067c	12
Reserved	Reserved	SUPERNET 3 Register <0xa0> . . SUPERNET 3 Register <0xbf>		0x0680 . . 0x06fc	13
Reserved	Reserved	SUPERNET 3 Register <0xc0> . . SUPERNET 3 Register <0xdf>		0x0700 . . 0x077c	14
Reserved	Reserved	SUPERNET 3 Register <0xe0> . . SUPERNET 3 Register <0xff>		0x0780 . . 0x07fc	15

Table 57: Control Register File

9.0 Appendix B Testing the NIC

There are several flags and registers provided to test the correct operation of the "**SK-NET FDDI-<x>P**" Network Interface Cards (NIC).

Setting the EN_Config_Write to "1" enables writing registers in the configuration space that are read-only during normal operation of the NIC.

Testing the NIC is not restricted to the flags and registers described in the following, but other combinations of setting and verifying values, forcing states and responses are required too. The following sections describe some of the registers that are provided for test purpose only. Therefore they were omitted in the main part of the manual.

9.0.1 Test Control Register

The Test Control Register enables forcing errors and it enables write accesses to the configuration space registers.

Bit	Name	Description	Write	Read	Reset (SW)
7	FORCE DATAPERR MASTER READ	Simulate data parity error on next master read access	exec	0	0
6	FORCE DATAPERR MASTER WRITE	Generate data parity error on next master write access	exec	0	0
5	FORCE DATAPERR TARGET READ	Generate data parity error on next target read access	exec	0	0
4	FORCE DATAPERR TARGET WRITE	Simulate data parity error on this target write access	exec	0	0
3	FORCE ADDRPERR MASTER	Generate address parity on next master access	exec	0	0
2	FORCE ADDRPERR TARGET	Simulate address parity on next master access	exec	0	0
1	EN Config Write On	Enables write accesses to the Configuration Registers over the Control Register File.	exec	0b10	0
0	EN Config Write Off			0b01	1

Table 58: Test Control Register/Commands

The commands for forcing parity errors are executed once on the same or next access. Status bits are set and interrupts are generated how defined by PCI and/or the related description in this document.

Handle with care!

9.0.2 Test FPROM Loader

The FPROM Loader Control Register provides some bits to step through loading of initialization values from the FPROM. This is done after power_on in the normal operation. However, to test loading the initialization values the Loader_Start command is provided. Set Loader_Test_ON to “1” and write a “1” to Loader_Start. With each setting (writing a “1”) to the Loader_Step the FPROM loader reads the next 32 bit word.

Bit	Name	Description	Write	Read	Reset (HW)
FPROM Address Register/Counter					
31..17	Reserved				
16..0	Address Register/Counter	Defines 17-bit Flash FPROM address	yes	aw	0x1ffff
FPROM Data Register					
31..24	Reserved				
23..16	Reserved				
15..8	Reserved				
7..0	Data Port	Programming FPROM Data Port	exec	value	0
FPROM Loader Control Register					
31..12	Reserved				
11	Loader Test On	Testmode On/Off	exec	0b10	0
10	Loader Test Off			0b01	1
9	Loader Step	Decrement FPROM Address Counter.	exec	0	0
8	Loader Start	Starts loading of Flash EPROM at the location defined by the Address Register.	exec	0	
7..0	Reserved				

Table 59: FPROM Address Register/Counter

9.0.3 Test Downcounters (Timer, Watchdog, RTM)

Testing the timer, the watchdog, and the restricted token monitor is similar. For each of these downcounters a Test_ON/OFF flag is provided and with an <xxx>_Step command clock pulses are generated by the software. For the location of the corresponding bit refer to the

- ❏ Table 49: *Timer Registers* on page 69
- ❏ Table 50: *Watchdog* on page 70
- ❏ Table 51: *Restricted Token Monitor* on page 71

9.0.4 Test BMU Data Tranfer

For each of the queues testing the BMU mechanism can be performed separately. The Transmit /Receive Test Registers enable testing of each individual state machine.
The state machine under test must be switched to test mode. The single steps performed by the supervisor state machine can be done step by step.

This is only valid for the supervisor state machine. Stepping the read descriptor, the write descriptor or the transfer state machine will generate errorneous behavior. The register structure is the same for both Receive and Transmit queue test register.

Bit	Name	Description	Write	Read	Reset (HW)
	Supervisor SM				
31..28	Q<3..0>	States	if Load	value	0
27	Load	Load States Q<3..0> if = 1, Q<3..0> are loaded to SM		0	0
26	Test On	Testmode ON/OFF	exec	0b10	0
25	Test Off			0b01	1
24	SM Step	Step SM	exec	0	
	Read Descriptor SM				
23..20	Q<3..0>	States	if Load	value	0
19	Load	Load States Q<3..0> if = 1, Q<3..0> are loaded to SM		0	0
18	Test On	Testmode ON/OFF	exec	0b10	0
17	Test Off			0b01	1
16	SM Step	Step SM	exec	0	
	Write Descriptor SM				
15 ..12	Q<3..0>	States	if Load	value	0
11	Load	Load States Q<3..0> if = 1, Q<3..0> are loaded to SM		0	0
10	Test On	Testmode ON/OFF	exec	0b10	0
9	Test Off			0b01	1
8	SM Step	Step SM	exec	0	
	Transfer SM				
7..4	Q<3..0>	States	if Load	value	0
3	Load	Load States Q<3..0> if = 1, Q<3..0> are loaded to SM		0	0

Table 60: Receive/Transmit Test Register 1

Bit	Name	Description	Write	Read	Reset (HW)
2	Test On	Testmode ON/OFF	exec	0b10	0
1	Test Off			0b01	1
0	SM Step	Step SM	exec	0	

Table 60: Receive/Transmit Test Register 1

The Load States Q<3..0> are listed in the following tables.

State Receive State Machines	Q<3..0>	Status Transmit State Machines
Supervisor SM		Supervisor SM
Idle	0b0000	Idle
Res_Start	0b0001	Res_Start
Get_Desc	0b0011	Get_Desc
Check	0b0010	Check
Move_Data	0b0110	Move_Data
Put_Desc	0b0111	Put_Desc
Set_Irq	0b0101	Set_Irq
Read Descriptor		Read Descriptor
Idle	0b0000	Idle
Load	0b0001	Load
Wait_TC	0b0011	Wait_TC
Reset_EOF	0b0110	
Wait_Done	0b0010	
	0b0100	Wait_Done
Transfer Data		Transfer Data
Idle	0b0000	Idle
Load	0b0011	Load

Table 61: Descriptor State Machines States

State Receive State Machines	Q<3..0>	Status Transmit State Machines
Wait_TC	0b0010	Wait_TC
Wait_Done	0b0100	Wait_Done
Write Descriptor		Write Descriptor
Idle	0b0000	Idle
Act_Buf_Length	0b0001	Act_Buf_Length
Load_A4	0b0010	Res_OWN
Wait_EOF	0b0011	
Load_N2C	0b0100	
Wait_TC	0b0101	
Wait_TC4	0b0110	Load_A
Load_A	0b0111	Wait_TC
	0b1100	Load_N2C
	0b1001	Wait_DONE
Wait_Done	0b1100	

Table 61: Descriptor State Machines States

In the test register 2 and 3 the address counter given to the transfer state machine can be incremented/decrement. The following table shows only the transmit queue test registers 2 and 3. They are identical with the receive queue test registers, except for the bits 3 and 2 in the test register 3. The loopback enable bits are provided in the transmit queue test register 3 only.

Bit	Name	Description	Write	Read	Reset (HW)
	Transmit Queue Test Register 2				
31..7	Reserved				
6	Add. Counter Test On	Testmode On/Off	exec	0b10	0
5	Add. Counter Test Off			0b01	1
4	Add. Counter Step	Increment **Current Address Counter**	exec	0	0
3	Reserved				

Table 62: Transmit Test Register 2, 3

Bit	Name	Description	Write	Read	Reset (HW)
2	Byte Counter Test On	Testmode On/Off	exec	0b10	0
1	Byte Counter Test Off			0b01	1
0	Byte Counter Step	Decrement **Current Byte Counter**	exec	0	0
	Transmit Queue Test Register 3				
31..6	Reserved				
5	Set Loopback	Loopback asynchronous transmit queue FIFO to receive queue 1 FIFO (without memory buffer)	exec	0b10	0
4	Unset Loopback			0b01	1
3..2	Mux<1..0>	Mux position	no	value	0
1..0	VRam<1..0>	Virtual ram buffer address	yes	value	0

Table 62: Transmit Test Register 2, 3

For testing/controlling purposes, VRam<1..0> may be written, Mux<1..0> is only readable.

If VRam<1..0> is modified, Mux<1..0> should follow depending on the current buffer start address.