Hardware Data Sheet Section III

ET1810 / ET1811 / ET1812

EtherCAT® Slave Controller
IP Core for Altera® FPGAs
Release 2.4.4

Section I – Technology
(Online at http://www.beckhoff.com)

Section II – Register Description
(Online at http://www.beckhoff.com)

Section III – Hardware Description
Installation, Configuration, Resource consumption, Interface specification

Version 1.0
Date: 2015-01-20

BECKHOFF

DOCUMENT ORGANIZATION

The Beckhoff EtherCAT Slave Controller (ESC) documentation covers the following Beckhoff ESCs:

- ET1200
- ET1100
- EtherCAT IP Core for Altera® FPGAs
- EtherCAT IP Core for Xilinx® FPGAs
- ESC20

The documentation is organized in three sections. Section I and section II are common for all Beckhoff ESCs, Section III is specific for each ESC variant.

The latest documentation is available at the Beckhoff homepage (http://www.beckhoff.com).

Section I – Technology (All ESCs)

Section I deals with the basic EtherCAT technology. Starting with the EtherCAT protocol itself, the frame processing inside EtherCAT slaves is described. The features and interfaces of the physical layer with its two alternatives Ethernet and EBUS are explained afterwards. Finally, the details of the functional units of an ESC like FMMU, SyncManager, Distributed Clocks, Slave Information Interface, Interrupts, Watchdogs, and so on, are described.

Since Section I is common for all Beckhoff ESCs, it might describe features which are not available in a specific ESC. Refer to the feature details overview in Section III of a specific ESC to find out which features are available.

Section II – Register Description (All ESCs)

Section II contains detailed information about all ESC registers. This section is also common for all Beckhoff ESCs, thus registers, register bits, or features are described which might not be available in a specific ESC. Refer to the register overview and to the feature details overview in Section III of a specific ESC to find out which registers and features are available.

Section III – Hardware Description (Specific ESC)

Section III is ESC specific and contains detailed information about the ESC features, implemented registers, configuration, interfaces, pinout, usage, electrical and mechanical specification, and so on. Especially the Process Data Interfaces (PDI) supported by the ESC are part of this section.

Additional Documentation

Application notes and utilities can also be found at the Beckhoff homepage. Pinout configuration tools for ET1100/ET1200 are available. Additional information on EtherCAT IP Cores with latest updates regarding design flow compatibility, FPGA device support and known issues are also available.

DOCUMENT HISTORY

Version	Comment
1.0	• Initial release EtherCAT IP Core for Altera FPGAs 2.4.4

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

µC	Microcontroller
ADR	Address
AL	Application Layer
BHE	Bus High Enable
CMD	Command
CS	Chip Select
DC	Distributed Clock
DL	Data Link Layer
ECAT	EtherCAT
ESI	EtherCAT Slave Information
EOF	End of Frame
ESC	EtherCAT Slave Controller
FMMU	Fieldbus Memory Management Unit
FPGA	Field Programmable Gate Array
GPI	General Purpose Input
GPO	General Purpose Output
HDL	Hardware Description Language
IP	Intellectual Property
IRQ	Interrupt Request
LC	Logic Cell
LE	Logic Element
MAC	Media Access Controller
MDIO	Management Data Input / Output
MI	(PHY) Management Interface
MII	Media Independent Interface
MISO	Master In – Slave Out
MOSI	Master Out – Slave In
PDI	Process Data Interface
PLD	Programmable Logic Device
PLL	Phase Locked Loop
RBF	Raw Binary File
RD	Read
RMII	Reduced Media Independent Interface
SM	SyncManager
SoC	System on a Chip
SOF	Start of Frame
SOPC	System on a programmable Chip
SPI	Serial Peripheral Interface
VHDL	Very High Speed Integrated Circuit Hardware Description Language
WR	Write

1 Overview

The EtherCAT IP Core is a configurable EtherCAT Slave Controller (ESC). It takes care of the EtherCAT communication as an interface between the EtherCAT fieldbus and the slave application. The EtherCAT IP Core is delivered as a configurable system so that the feature set fits the requirements perfectly and brings costs down to an optimum.

Table 1: IP Core Main Features

Feature	IP Core configurable features
Ports	1-3 MII ports or 1-2 RMII ports
FMMUs	0-8
SyncManagers	0-8
RAM	1-60 KB
Distributed Clocks	Yes, 32 bit or 64 bit
Process Data Interfaces	• 32 Bit Digital I/O (unidirectional) • SPI Slave • 8/16 bit asynchronous µController Interface • Avalon® on-chip bus
Other features	• Example designs for easy start up included • Slave applications can run on-chip if the appropriate FPGAs with sufficient resources are used

The general functionality of the EtherCAT IP Core is shown in Figure 1:

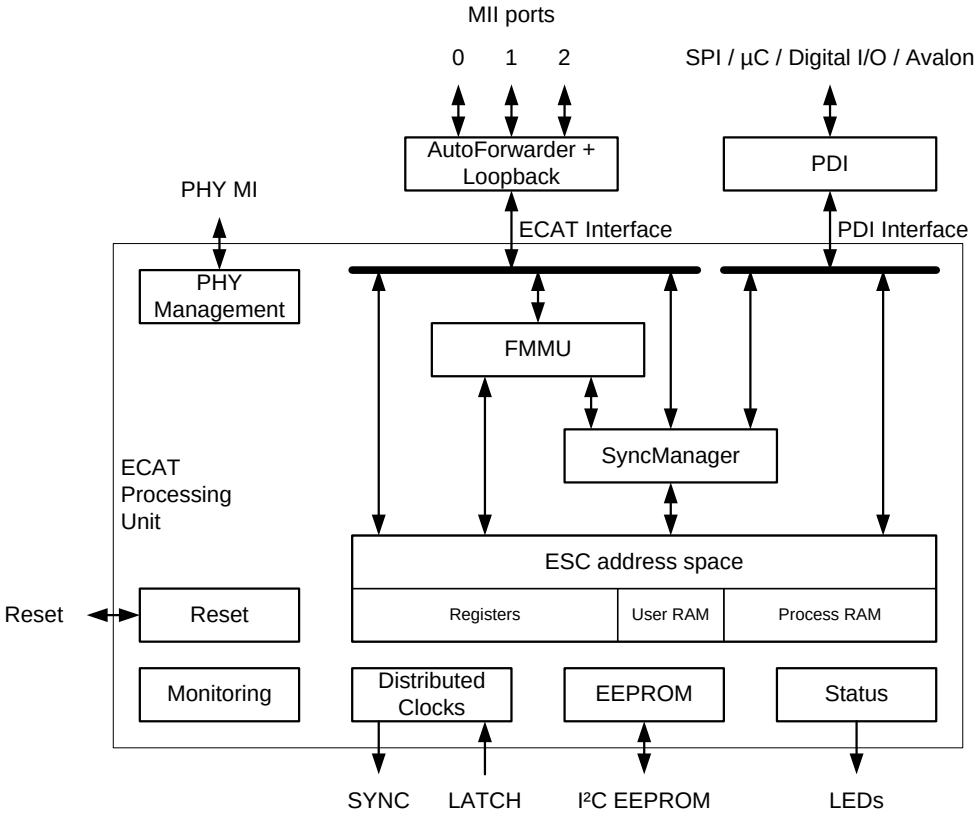

Figure 1: EtherCAT IP Core Block Diagram

1.1 Frame processing order

The frame processing order of the EtherCAT IP Core is as follows (logical port numbers are used):

Table 2: Frame Processing Order

Number of Ports	Frame processing order
1	0→EtherCAT Processing Unit→0
2	0→EtherCAT Processing Unit→1 / 1→0
3	0→EtherCAT Processing Unit→1 / 1→2 / 2→0 (log. ports 0,1, and 2)

Figure 2 shows the frame processing in general:

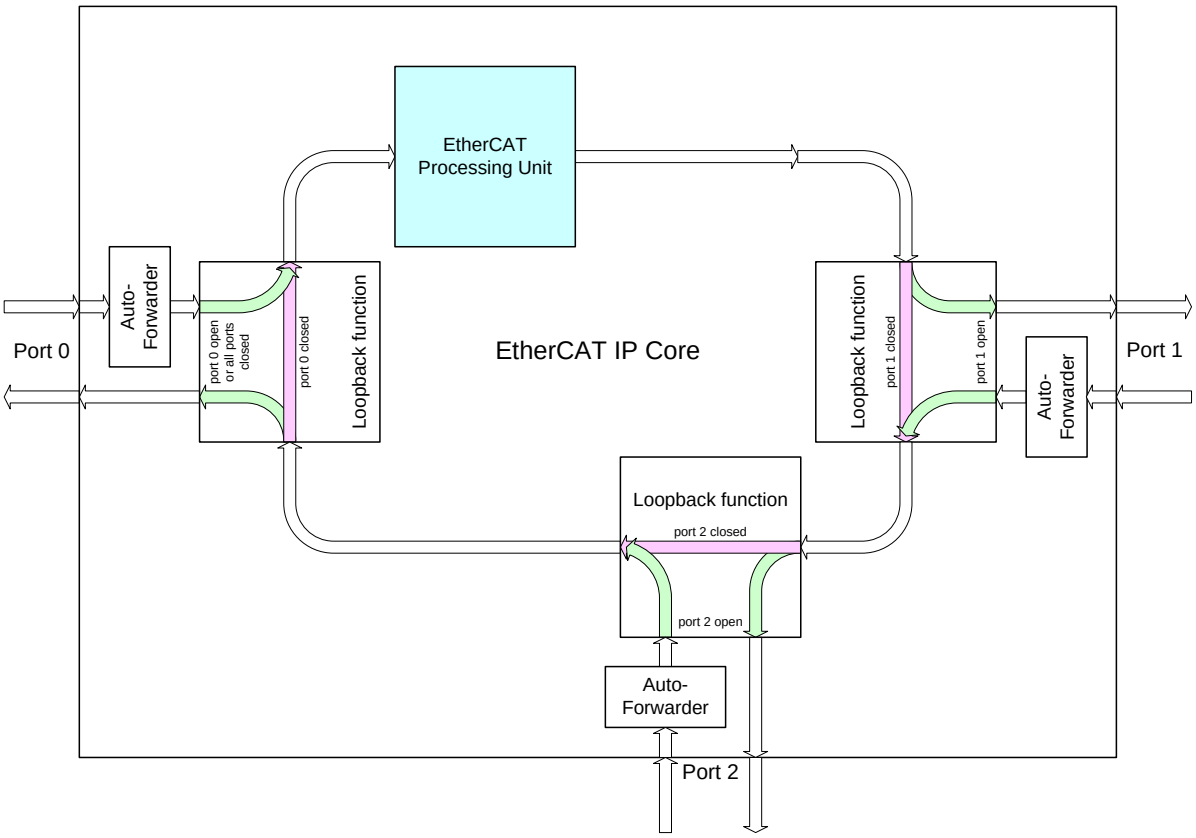

Figure 2: Frame Processing

Frame Processing Example with Ports 0 and 1

A frame received at port 0 goes via the Auto-Forwarder and the Loopback function to the EtherCAT Processing Unit which processes it. Then, the frame is sent to port 1. If port 1 is open, the frame is sent out at port 1. If it is closed, the frame is forwarded by the Loopback function to port 2. Since port 2 is not configured, the Loopback function of port 2 forwards the frame to the Loopback function of port 0, and then it is sent out at port 0 – back to the master.

1.2 Scope of this document

Purpose of this document is to describe the installation and configuration of the EtherCAT IP Core for Altera FPGAs. Furthermore, the signals and registers of the IP Core depending on the chosen configuration are described.

This documentation was made with the assumption that the user is familiar with the handling of the Altera Quartus® Development Environment.

1.3 Scope of Delivery

The EtherCAT IP Core installation file includes:

- EtherCAT IP Core (encrypted VHDL library)
- Example designs

The following files which contain customer specific information are required to synthesize the IP Core. They are delivered independently of the installation file.

- License File to decrypt EtherCAT IP Core: license_<company>_<Dongle/MAC ID>_<date>.dat
- Encrypted Vendor ID package: pk_ECAT_VENDORID_<company>_Altera.vhd

1.4 Target FPGAs

The EtherCAT IP Core for Altera® FPGAs is targeted at these FPGA families:

- Altera Cyclone®, Cyclone II, Cyclone III, Cyclone III LS, Cyclone IV E+GX, Cyclone V
- Altera Cyclone V SoC
- Altera Stratix®, Stratix II, Stratix III, Stratix IV, Stratix V
- Altera Arria® GX, Arria II GX, Arria II GZ, Arria V
- Altera Stratix® GX, Stratix II GX
- Intel® Atom™ Processor E6x5C (formerly Stellarton)
- Altera MAX10

The EtherCAT IP Core is designed to support a wide range of FPGAs without modifications, because it does not instantiate dedicated FPGA resources, or rely on device specific features. Thus, the IP Core is easily portable to new FPGA families.

The complexity of the IP Core is highly configurable, so its demands for logic resources, memory blocks, and FPGA speed cover a wide range. Thus, it is not possible to run any IP Core configuration on any target FPGA with any speed grade. I.e., there are IP Core configurations requiring a faster speed grade, or a larger FPGA, or even a more powerful FPGA family.

It is necessary to run through the whole synthesis process – including timing checks –, to evaluate if the selected FPGA is suitable for a certain IP Core configuration before making the decision for the FPGA. Please consider a security margin for the logic resources to allow for minor enhancements and bug fixes of the IP Core and the user logic.

1.5 Designflow requirements

For synthesis of the EtherCAT IP Core for Altera FPGAs, at least one of the following Altera Quartus II versions is needed (with latest service pack):

- Altera Quartus II version 13.0
- Altera Quartus II version 13.1
- Altera Quartus II version 14.0
- Altera Quartus II version 14.1

Higher Quartus II versions are probably supported. Installation of the latest service pack is recommended. A free version (“Web Edition”) is available from Altera (http://www.altera.com).

Optionally for using the EtherCAT IP Core with a NIOS® based Qsys design, you will need

- Altera Nios II Embedded Design Suite

1.6 Tested FPGA/Designflow combinations

The EtherCAT IP Core has been synthesized successfully with different Quartus II versions and FPGA families. Table 3 lists combinations of FPGA devices and design tools versions which have been synthesized or even tested in real hardware. This list does not claim to be complete, it just illustrates that the EtherCAT IP Core is designed to comply with a broad spectrum of FPGAs.

Table 3: Tested FPGA/Designflow combinations

IP Core	Family	Device	Designflow	Test	Used Example Designs
2.4.4	Cyclone	EP1C12	Quartus II 13.0	Synthesis	
	Cyclone II	EP2C20	Quartus II 13.0	Synthesis	
	Cyclone III	EP3C40	Quartus II 13.1.4	Hardware	DBC3C40 Digital I/O
	Cyclone IV E	EP4CE55	Quartus II 14.1	Hardware	DBC4CE55 Nios
	Cyclone IV E	EP4CE115	Quartus II 14.1	Hardware	DE2-115 Nios
	Cyclone V	5CEBA2	Quartus II 13.1.4	Synthesis	
	Cyclone V SoC	5CSEMA5	Quartus II 13.1.4	Synthesis	
	Stratix	EP1S10	Quartus II 13.0	Synthesis	
	Stratix GX	EP1SGX10	Quartus II 13.0	Synthesis	
	Stratix II	EP2S30	Quartus II 13.0	Synthesis	
	Stratix III	EP3SE50	Quartus II 13.1.4	Synthesis	
	Stratix IV E	EP4SE230	Quartus II 14.1	Synthesis	
	Stratix IV GT	EP4S40G	Quartus II 14.1	Synthesis	
	Stratix IV GX	EP4SGX70	Quartus II 14.1	Synthesis	
	Stratix V	5SGSMD4	Quartus II 13.1.4	Synthesis	
	Arria GX	EP1AGX20	Quartus II 13.0	Synthesis	
	Arria II GX	EP2AGX45	Quartus II 13.1.4	Synthesis	
	Arria V	5AGXMB3	Quartus II 13.1.4	Synthesis	
	Arria V GZ	5AGZME5	Quartus II 13.1.4	Synthesis	
	Intel Atom E6x5C	EP2AGXE6 XXFPGA	Quartus II 13.1.4	Synthesis	
	MAX10	10M40DA	Quartus II 14.1	Synthesis	

NOTE: Synthesis test means analysis, synthesis, fitter, and assembler. Hardware test means the design was operational using real hardware.

NOTE: Turn on Analysis & Synthesis option: Auto RAM Replacement, otherwise the RAM inside the IP Core will be implemented with individual registers.

Refer to the *Hardware Data Sheet Section III Addendum* available at the Beckhoff homepage (http://www.beckhoff.com) for latest updates regarding device support, design flow compatibility, and known issues.

1.7 Release Notes

EtherCAT IP Core updates deliver feature enhancements and removed restrictions.. Feature enhancements are not mandatory regarding conformance to the EtherCAT standard. Restrictions have to be judged whether they are relevant in the user's configuration or not, or if workarounds are possible.

Table 4: Release notes

Version	Release notes
2.4.0 (03/2011)	• Update to Quartus II 10.1 with support for latest MegaWizard/SoPC Builder • Station Alias register (0x0012:0x0013) is now permanently enabled • Extended DL Control register (0x0102:0x0103) is now permanently enabled • ECAT Event Mask register (0x0200:0x0201) is now permanently enabled • AL Control register (0x0120:0x0121) and AL Status register (0x0130:0x0131) are now 16 bit wide • Source code obfuscation introduced Enhancements: • MI link detection and configuration now disables Gigabit Ethernet advertisement • Added example design for Altera DE2-115 Development and Education Board/Industrial Networking Kit (INK) • Preliminary Qsys support • Altera Stratix V, Arria II GZ, and Intel Atom E6x5C (FPGA part) support Restrictions of this version, which are removed in V2.4.3: • Reduced receive time accuracy when Receive Times are enabled while Distributed Clocks are disabled. Restrictions of this version, which are removed in V2.4.4: • The last 4 Kbyte Process Data RAM (0xF000:0xFFFF) cannot be used in the 60 Kbyte RAM configuration.
2.4.3 (07/2013)	• Update to Quartus II 13.0 Enhancements: • Qsys is now supported • Altera Cyclone III LS, Cyclone V, Cyclone V SoC, Arria V support added • RX FIFO size initialized by SII EEPROM • MI link detection: relaxed checking of PHY register 9 (1000Base-T Master-Slave Control register) Restrictions of previous versions which are removed in this version: • Improved receive time accuracy when Receive Times are enabled while Distributed Clocks are disabled (customers using this configuration in V2.4.0 should update to V2.4.3). Restrictions of this version, which are removed in V2.4.4: • The last 4 Kbyte Process Data RAM (0xF000:0xFFFF) cannot be used in the 60 Kbyte RAM configuration.

Version	Release notes
2.4.4 (01/2015)	• Update to Quartus II 13.1 • The EL9800/FB1122 example designs have been removed because these evaluation boards are no longer available. Enhancements • Altera MAX10 is now supported • For EEPROM Emulation, the CRC error bit 0x0502[11] can be written via PDI to indicate CRC errors during a reload command. • The ESI XML device description does not use special data types anymore. Restrictions of previous versions which are removed in this version: • The last 4 Kbyte Process Data RAM (0xF000:0xFFFF) can be used in the 60 Kbyte RAM configuration.

The IP Core version, denoted as X.Y.Z (e.g., 2.4.3), consists of three values X, Y, and Z. These values can be read out in registers 0x0001 and 0x0002.

Table 5: Register Revision (0x0001)

Bit	Description	ECAT	PDI	Reset Value
7:0	IP Core major version X	r/-	r/-	IP Core dep.

Table 6: Register Build (0x0002:0x0003)

Bit	Description	ECAT	PDI	Reset Value
3:0	IP Core maintenance version Z	r/-	r/-	IP Core dep.
7:4	IP Core minor version Y	r/-	r/-	IP Core dep.
15:8	Patch level: 0x00: original release 0x01-0x0F: patch level of original release	r/-	r/-	IP Core dep.

1.8 Design flow

The design flow for creating an EtherCAT Slave Controller based on the EtherCAT IP Core is shown in the following picture:

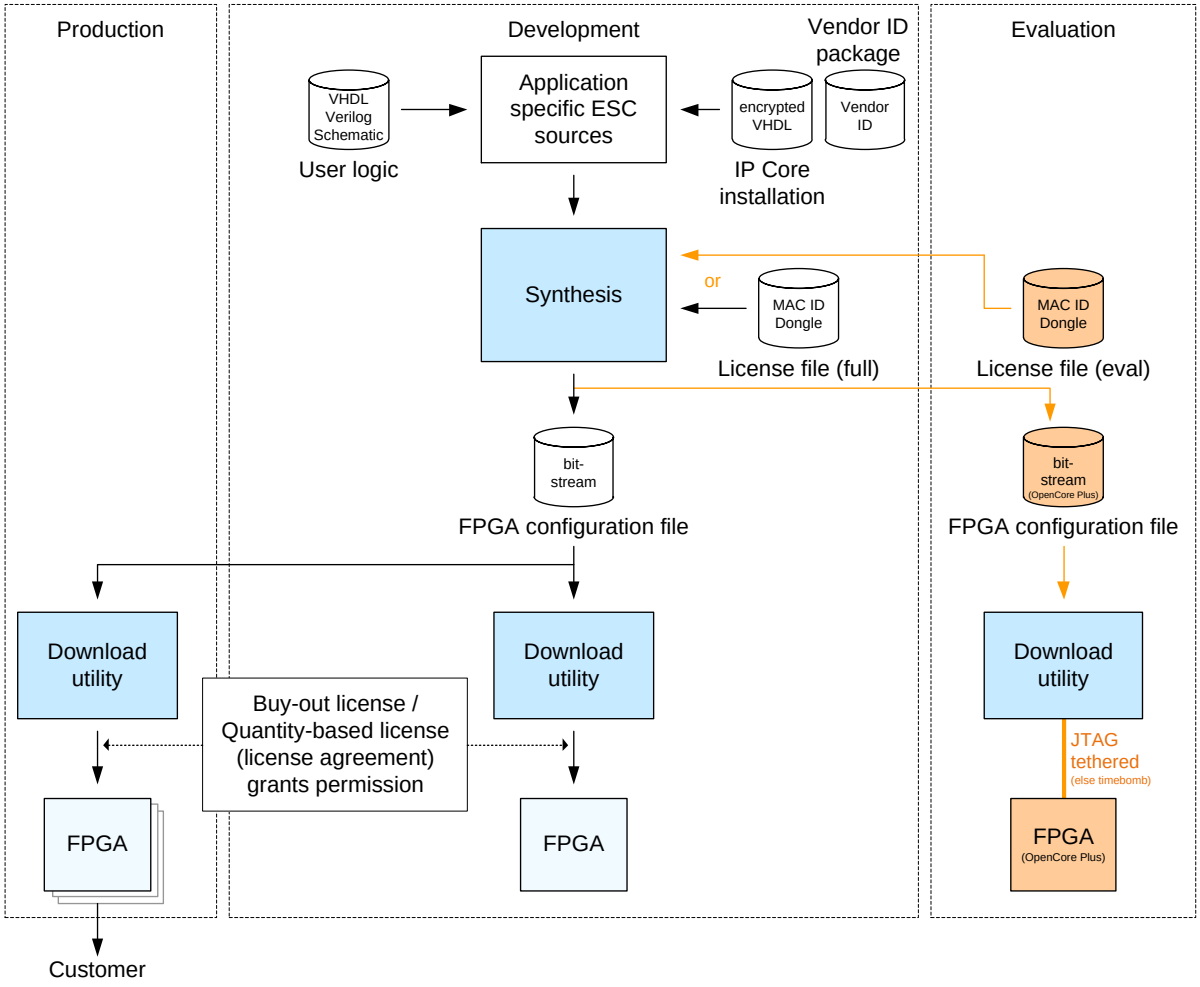

Figure 3: Design flow

1.9 OpenCore Plus Evaluation

The EtherCAT IP Core for Altera FPGAs supports OpenCore Plus evaluation. A special License File with OpenCore Plus support is issued for each user, together with the IP Core Vendor ID package. For further information on OpenCore Plus, refer to the Altera Application Note 320 “OpenCore Plus Evaluation of Megafunctions”, available from Altera (http://www.altera.com).

A design with an OpenCore Plus EtherCAT IP Core is subject to some restrictions:

- Only a time limited programming file (<design_name>_time_limited.sof) for the Altera Quartus II Programmer is generated. Other programming files (e.g., .rbf, .pof) are not generated.
- For hardware testing, the ESC design has to be connected to the PC running the Altera Quartus II Programmer using a programming adapter with a JTAG connection. The EtherCAT IP Core is fully functional while the adapter is connected.
- If the connection is interrupted, the EtherCAT IP Core will discontinue its function after approximately 1 hour.
- The OpenCore Plus version slightly increases the resource consumption of the IP Core.
- The OpenCore Plus programming file must not be distributed/sold.

A vendor ID package is required for both evaluation and full license. It is recommended to use an evaluation vendor ID (package) for evaluation, and the original vendor ID for production. The evaluation vendor ID is beginning with “0xE.....” and ends with the original vendor ID digits. Evaluation vendor IDs cannot pass the EtherCAT conformance tests.

OpenCore Plus Issues

Sometimes additional top-level pins appear in the OpenCore Plus design, these signals should be grounded externally if possible. This is a Quartus OpenCore Plus integration issue, not an EtherCAT IP Core issue. The signals will not appear if a full license is used. Additionally, do not use incremental synthesis together with OpenCore Plus, since this was found to produce defective designs similar to the OpenCore Plus integration issue.

Sometimes timing requirements are not met with OpenCore Plus. Experience shows that timing violations related to the clock altera_reserved_tck can be ignored.

Upgrading to a Full License

A design using an OpenCore Plus EtherCAT IP Core does not have to be changed when upgrading to a full license, only the full License File has to be installed instead of the OpenCore Plus License File. A re-generation of the EtherCAT IP Core (running through the MegaWizard) and a new synthesis run is necessary to generate the unlimited programming files.

1.10 Simulation

A behavioral simulation model of the EtherCAT IP core is not available because of its size and complexity. Thus, simulation of the entire EtherCAT IP Core is not supported, and the EDA Netlist Writer cannot be used for designs which contain the EtherCAT IP Core. In most cases, simulation of the EtherCAT IP Core is not necessary, as the IP Core was thoroughly tested and the interfaces are standardized (Ethernet, Avalon) or simple and well described. Problems at the interface level can often be solved with a scope shot of the interface signals.

Nevertheless, customer designs using the Avalon on-chip bus can easily be simulated using a Bus Functional Model of the on-chip bus slave interface instead of a simulation model of the entire EtherCAT IP Core.

From the processor's view, the EtherCAT IP Core is a memory (or a bunch of registers). For processor bus verification, the EtherCAT IP Core can be substituted by another IP core with Avalon slave interface which behaves like a memory as well. The EtherCAT IP Core can be replaced for simulation by e.g.:

- Altera On-Chip Memory slave
- Avalon slave created with the SOPC Builder or Qsys

2 Features and Registers

2.1 Features

Table 7: IP Core Feature Details

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
EtherCAT Ports	**1-3**	**1-3**
Permanent ports	1-3	1-3
Optional Bridge port 3 (EBUS or MII)	-	-
EBUS ports	-	-
MII ports	0-3	0-3
RMII ports	0-2	0-2
RGMII ports	-	-
Port 0	x	x
Ports 0, 1	x	x
Ports 0, 1, 2	x	x
EtherCAT mode	Direct	Direct
Slave Category	Full Slave	Full Slave
Position addressing	x	x
Node addressing	x	x
Logical addressing	x	x
Broadcast addressing	x	x
Physical Layer General Features		
FIFO Size configurable (0x0100[18:16])	x	x
FIFO Size default from SII EEPROM	x	x
Auto-Forwarder checks CRC and SOF	x	x
Forwarded RX Error indication, detection and Counter (0x0308:0x030B)	x	x
Lost Link Counter (0x0310:0x0313)	c	c
Prevention of circulating frames	x	x
Fallback: Port 0 opens if all ports are closed	x	x
VLAN Tag and IP/UDP support	x	x
Enhanced Link Detection per port configurable	x	x
General Ethernet Features (MII/RMII/RGMII)		
MII Management Interface (0x0510:0x051F)	c	c
Supported PHY Address Offsets	any	any
Individual port PHY addresses	-	-
Port PHY addresses readable	-	-
Link Polarity configurable	User logic	User logic
Enhanced Link Detection supported	x	x
FX PHY support (native)	-	-
PHY reset out signals	-	-
Link detection using PHY signal (LED)	x	x
MI link status and configuration	c	c
MI controllable by PDI (0x0516:0x0517)	x	x
MI read error (0x0510.13)	x	x
MI PHY configuration update status (0x0518.5)	x	x
MI preamble suppression	x	x
Additional MCLK	x	x
Gigabit PHY configuration	x	x
Gigabit PHY register 9 relaxed check	x	x
FX PHY configuration	-	-
Transparent Mode	-	-
MII Features		
CLK25OUT as PHY clock source	User logic	User logic
Bootstrap TX Shift settings	c	c
Automatic TX Shift setting (with TX_CLK)	c	c
TX Shift not necessary (PHY TX_CLK as clock source)	-	-
FIFO size reduction steps	1	1
PDI General Features		
Increased PDI performance	-	-
Extended PDI Configuration (0x0152:0x0153)	x	x
PDI Error Counter (0x030D)	c	c
PDI Error Code (0x030E)	c	c
CPU_CLK output (10, 20, 25 MHz)	User logic	User logic
SOF, EOF, WD_TRIG and WD_STATE independent of PDI	x	x
Available PDIs and PDI features depending on port configuration	-	-
PDI selection at run-time (SII EEPROM)	-	-
PDI active immediately (SII EEPROM settings ignored)	x	x
PDI function acknowledge by write	-	-
PDI Information register 0x014E:0x014F	-	-
Digital I/O PDI	**x**	**x**
Digital I/O width [bits]	8/16/24/32	8/16/24/32
PDI Control register value (0x0140:0x0141)	4	4
Control/Status signals:	7	7
LATCH_IN	x	x
SOF	x	x
OUTVALID	x	x
WD_TRIG	x	x
OE_CONF	-	-
OE_EXT	x	x
EEPROM_ Loaded	-	-
WD_STATE	x	x
EOF	x	x
Granularity of direction configuration [bits]	8	8
Bidirectional mode	- (User logic)	- (User logic)

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
Output high-Z if WD expired	User logic	User logic
Output 0 if WD expired	x	x
Output with EOF	x	x
Output with DC SyncSignals	x	x
Input with SOF	x	x
Input with DC SyncSignals	x	x
SPI Slave PDI	**x**	**x**
Max. SPI clock [MHz]	30	30
SPI modes configurable (0x0150[1:0])	x	x
SPI_IRQ driver configurable (0x0150[3:2])	x	x
SPI_SEL polarity configurable (0x0150.4)	x	x
Data out sample mode configurable (0x0150.5)	x	x
Busy signaling	-	-
Wait State byte(s)	x	x
Number of address extension byte(s)	any	any
2/4 Byte SPI master support	x	x
Extended error detection (read busy violation)	x	x
SPI_IRQ delay	x	x
Status indication	x	x
EEPROM_ Loaded signal	-	-
Asynchronous µController PDI	**8/16 bit**	**8/16 bit**
Extended µC configuration bits 0x0150[7:4], 0x0152:0x0153	x	x
ADR[15:13] available (000_b if not available)	x	x
EEPROM_Loaded signal	-	-
RD polarity configurable (0x0150.7)	-	-
Read BUSY delay (0x0152.0)	x	x
Write after first edge (0x0152.2)	x	x
Synchronous µController PDI	**-**	**-**
EEPROM_ Loaded signal		
On-Chip Bus PDI	**x**	**x**
Avalon®	x	x
OPB®	-	-
PLB v4.6®	-	-
AXI3™	-	-
AXI4™	-	-
AXI4 LITE™	-	-
Bus clock [MHz] (N=1,2,3,…)	N*25	N*25
Data bus width [bits]	8	8
Prefetch cycles	1/2/4	1/2/4
DC SyncSignals available directly and as IRQ	x	x
Bus clock multiplier in register 0x0150[6:0]	x	x
EEPROM_ Loaded signal	-	-
EtherCAT Bridge (port 3, EBUS/MII)	**-**	**-**
General Purpose I/O	**x**	**x**

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
GPO bits	0/8/16/ 32/64	0/8/16/ 32/64
GPI bits	0/8/16/ 32/64	0/8/16/ 32/64
GPIO available independent of PDI or port configuration	x	x
GPIO available without PDI	x	x
Concurrent access to GPO by ECAT and PDI	x	x
ESC Information		
Basic Information (0x0000:0x0006)	x	x
Port Descriptor (0x0007)	x	x
ESC Features supported (0x0008:0x0009)	x	x
Extended ESC Feature Availability in User RAM (0x0F80 ff.)	x	x
Write Protection (0x0020:0x0031)	c	c
Data Link Layer Features		
ECAT Reset (0x0040)	c	c
PDI Reset (0x0041)	c	c
ESC DL Control (0x0100:0x0103) bytes	4	4
EtherCAT only mode (0x0100.0)	x	x
Temporary loop control (0x0100.1)	x	x
FIFO Size configurable (0x0100[18:16])	x	x
Configured Station Address (0x0010:0x0011)	x	x
Configured Station Alias (0x0100.24, 0x0012:0x0013)	x	x
Physical Read/Write Offset (0x0108:0x0109)	c	c
Application Layer Features		
Extended AL Control/Status bits (0x0120[15:5], 0x0130[15:5])	x	x
AL Status Emulation (0x0140.8)	x	x
AL Status Code (0x0134:0x0135)	c	c
Interrupts		
ECAT Event Mask (0x0200:0x0201)	x	x
AL Event Mask (0x0204:0x0207)	c	c
ECAT Event Request (0x0210:0x0211)	x	x
AL Event Request (0x0220:0x0223)	x	x
SyncManager activation changed (0x0220.4)	x	x
SyncManager watchdog expiration (0x0220.6)	x	x
Error Counters		
RX Error Counter (0x0300:0x0307)	x	x
Forwarded RX Error Counter (0x0308:0x030B)	x	x
ECAT Processing Unit Error Counter (0x030C)	c	c
PDI Error Counter (0x030D)	c	c
Lost Link Counter (0x0310:0x0313)	c	c

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
Watchdog		
Watchdog Divider configurable (0x0400:0x0401)	c	c
Watchdog Process Data	x	x
Watchdog PDI	x	x
Watchdog Counter Process Data (0x0442)	x	x
Watchdog Counter PDI (0x0443)	x	x
SII EEPROM Interface (0x0500:0x050F)		
EEPROM sizes supported	1 KB-4 Mbyte	1 KB-4 Mbyte
EEPROM size reflected in 0x0502.7	x	x
EEPROM controllable by PDI	x	x
EEPROM Emulation by PDI	c	c
Read data bytes (0x0502.6)	4	4
Internal Pull-Ups for EEPROM_CLK and EEPROM_DATA	User logic	User logic
FMMUs	0-8	0-8
Bit-oriented operation	x	x
SyncManagers	0-8	0-8
Watchdog trigger generation for 1 Byte Mailbox configuration independent of reading access	x	x
SyncManager Event Times (+0x8[7:6])	c	c
Buffer state (+0x5[7:6])	x	x
Distributed Clocks	**c**	**c**
Width	32/64	32/64
Sync/Latch signals	4 (2 Sync-Signals, 2 Latch-Signals)	4 (2 Sync-Signals, 2 Latch-Signals)
SyncManager Event Times (0x09F0:0x09FF)	c	c
DC Receive Times	c	c
DC Time Loop Control controllable by PDI	c	c
DC activation by EEPROM (0x0140[11:10])	-	-
Propagation delay measurement with traffic (BWR/FPWR 0x900 detected at each port)	x	x
LatchSignal state in Latch Status register (0x09AE:0x09AF)	x	x
SyncSignal Auto-Activation (0x0981.3)	x	x
SyncSignal 32 or 64 bit Start Time (0x0981.4)	x	x
SyncSignal Late Activation (0x0981[6:5])	x	x
SyncSignal debug pulse (0x0981.7)	x	x
SyncSignal Activation State 0x0984)	x	x
Reset filters after writing filter depth	x	x
ESC Specific Registers (0x0E00:0x0EFF)		
Product and Vendor ID	x	x
POR Values	-	-
FPGA Update (online)	-	-

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
Process RAM and User RAM		
Process RAM (0x1000 ff.) [KByte]	1-60	1-60
User RAM (0x0F80:0x0FFF)	x	x
Extended ESC Feature Availability in User RAM	x	x
Additional EEPROMs	1-2	1-2
SII EEPROM (I²C)	c (EEPROM of µC used)	c (EEPROM of µC used)
FPGA configuration EEPROM	x	x
LED Signals		
RUN LED	c	c
RUN LED override	c	c
Link/Activity(x) LED per port	x	x
PERR(x) LED per port	-	-
Device ERR LED	c	c
STATE_RUN LED	c	c
Optional LED states		
RUN LED: Bootstrap	x	x
RUN LED: Booting	c	c
RUN LED: Device identification	c	c
RUN LED: loading SII EEPROM	c	c
Error LED: SII EEPROM loading error	c	c
Error LED: Invalid hardware configuration	-	-
Error LED: Process data watchdog timeout	c	c
Error LED: PDI watchdog timeout	c	c
Link/Activity: port closed	-	-
Link/Activity: local auto-negotiation error	-	-
Link/Activity: remote auto-negotiation error	-	-
Link/Activity: unknown PHY auto-negotiation error	-	-
LED test	-	-
Clock supply		
Crystal	-	-
Crystal oscillator	x	x
TX_CLK from PHY	x	x
25ppm clock source accuracy	x	x
Internal PLL	User logic	User logic
Power Supply Voltages	FPGA dep.	FPGA dep.
I/O Voltage	FPGA dep.	FPGA dep.
Core Voltage	FPGA dep.	FPGA dep.
Internal LDOs	-	-
Package	FPGA dep.	FPGA dep.
Size [mm²]	FPGA dep.	FPGA dep.
Original Release date	1/2015	7/2013
Configuration and Pinout calculator (XLS)	-	-
Register Configuration	individual	individual
Complete IP Core evaluation	x	x

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
Example designs/ pre-synthesized time-limited evaluation core included	3/3	5/4
FB1120 Digital I/O	-	-
FB1120 SPI	-	-
FB1122 Digital I/O	-	x/x
FB1122 SPI	-	x/-
DBC2C20 Digital I/O	-	-
DBC2C20 NIOS®	-	-

Feature	IP Core Altera® V2.4.4	IP Core Altera® V2.4.3
DBC3C40 Digital I/O	x/x	x/x
DBC3C40 NIOS	-	-
DBC4CE55 NIOS	x/x	x/x
DE2-115 NIOS MII	x/x	x/x
DE2-115 NIOS RGMII	-	-

Table 8: Legend

Symbol	Description
x	available
-	not available
c	configurable
User logic	Functionality can be added by user logic inside the FPGA
red	Feature changed in this version

2.2 Registers

An EtherCAT Slave Controller (ESC) has an address space of 64KByte. The first block of 4KByte (0x0000:0x0FFF) is dedicated for registers. The process data RAM starts at address 0x1000, its size is configurable.

Some registers are implemented depending on the configuration.

Table 9 gives an overview of the available registers.

Table 9: Register availability depending on register preset

Address	Length (Byte)	Description	IP Core V2.4.0-V2.4.4/ V2.04a-V2.04e Register set			IP Core V2.3.0-V2.3.2/ V2.03a-V2.03d Register set			IP Core V2.2.1/V2.2.0/ V2.02a Register set		
			S	M	L	S	M	L	S	M	L
0x0000	1	Type	x	x	x	x	x	x	x	x	x
0x0001	1	Revision	x	x	x	x	x	x	x	x	x
0x0002:0x0003	2	Build	x	x	x	x	x	x	x	x	x
0x0004	1	FMMUs supported	x	x	x	x	x	x	x	x	x
0x0005	1	SyncManagers supported	x	x	x	x	x	x	x	x	x
0x0006	1	RAM Size	x	x	x	x	x	x	x	x	x
0x0007	1	Port Descriptor	x	x	x	x	x	x	x	x	x
0x0008:0x0009	2	ESC Features supported	x	x	x	x	x	x	x	x	x
0x0010:0x0011	2	Configured Station Address	x	x	x	x	x	x	x	x	x
0x0012:0x0013	2	Configured Station Alias	x	x	x	c	c	x	c	c	x
0x0020	1	Write Register Enable	c	c	x	c	c	x	c	c	x
0x0021	1	Write Register Protection	c	c	x	c	c	x	c	c	x
0x0030	1	ESC Write Enable	c	c	x	c	c	x	c	c	x
0x0031	1	ESC Write Protection	c	c	x	c	c	x	c	c	x
0x0040	1	ESC Reset ECAT	c	c	c	c	c	c	c	c	c
0x0041	1	ESC Reset PDI	c	c	c	c	c	c	c	c	c
0x0100:0x0101	2	ESC DL Control	x	x	x	x	x	x	x	x	x
0x0102:0x0103	2	Extended ESC DL Control	x	x	x	r/c	r/c	x	r/c	r/c	x
0x0108:0x0109	2	Physical Read/Write Offset	c	c	x	c	c	x	c	c	x
0x0110:0x0111	2	ESC DL Status	x	x	x	x	x	x	x	x	x
0x0120	5 bits [4:0]	AL Control	x	x	x	x	x	x	x	x	x
0x0120:0x0121	2	AL Control	x	x	x	-	-	-	-	-	-
0x0130	5 bits [4:0]	AL Status	x	x	x	x	x	x	x	x	x
0x0130:0x0131	2	AL Status	x	x	x	-	-	-	-	-	-
0x0134:0x0135	2	AL Status Code	c	x	x	c	x	x	c	x	x
0x0138	1	RUN LED Override	c	c	c	c	c	c	-	-	-
0x0139	1	ERR LED Override	c	c	c	c	c	c	-	-	-
0x0140	1	PDI Control	x	x	x	x	x	x	x	x	x

Address	Length (Byte)	Description	IP Core V2.4.0-V2.4.4/ V2.04a-V2.04e Register set			IP Core V2.3.0-V2.3.2/ V2.03a-V2.03d Register set			IP Core V2.2.1/V2.2.0/ V2.02a Register set		
			S	M	L	S	M	L	S	M	L
0x0141	1	ESC Configuration	x	x	x	x	x	x	x	x	x
0x014E:0x014F	2	PDI Information	-	-	-	-	-	-	-	-	-
0x0150	1	PDI Configuration	x	x	x	x	x	x	x	x	x
0x0151	1	DC Sync/Latch Configuration	x	x	x	x	x	x	x	x	x
0x0152:0x0153	2	Extended PDI Configuration	x	x	x	x	x	x	x	x	x
0x0200:0x0201	2	ECAT Event Mask	x	x	x	c	c	x	c	c	x
0x0204:0x0207	4	PDI AL Event Mask	r/c	x	x	r/c	x	x	r/c	x	x
0x0210:0x0211	2	ECAT Event Request	x	x	x	x	x	x	x	x	x
0x0220:0x0223	4	AL Event Request	x	x	x	x	x	x	x	x	x
0x0300:0x0307	4x2	Rx Error Counter[3:0]	x	x	x	x	x	x	x	x	x
0x0308:0x030B	4x1	Forwarded Rx Error counter[3:0]	x	x	x	x	x	x	x	x	x
0x030C	1	ECAT Processing Unit Error Counter	c	c	x	c	c	x	c	c	x
0x030D	1	PDI Error Counter	c	c	x	c	c	x	c	c	x
0x030E	1	PDI Error Code	c	c	x	c	c	x	-	-	-
0x0310:0x0313	4x1	Lost Link Counter[3:0]	c	x	x	c	x	x	c	x	x
0x0400:0x0401	2	Watchdog Divider	r/c	x	x	r/c	x	x	r/c	x	x
0x0410:0x0411	2	Watchdog Time PDI	c	x	x	c	x	x	c	x	x
0x0420:0x0421	2	Watchdog Time Process Data	x	x	x	x	x	x	x	x	x
0x0440:0x0441	2	Watchdog Status Process Data	x	x	x	x	x	x	x	x	x
0x0442	1	Watchdog Counter Process Data	c	c	x	c	c	x	c	c	x
0x0443	1	Watchdog Counter PDI	c	c	x	c	c	x	c	c	x
0x0500:0x050F	16	SII EEPROM Interface	x	x	x	x	x	x	x	x	x
0x0510:0x0515	6	MII Management Interface	c	c	c	c	c	c	c	c	c
0x0516:0x0517	2	MII Management Access State	c	c	c	c	c	c	c	c	c
0x0518:0x051B	4	PHY Port Status[3:0]	c	c	c	c	c	c	c	c	c
0x0600:0x06FC	16x13	FMMU[15:0]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0x0800:0x087F	16x8	SyncManager[15:0]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0x0900:0x090F	4x4	DC – Receive Times[3:0]	rt	rt	rt	rt	rt	rt	rt	rt	rt
0x0910:0x0917	8	DC – System Time	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0918:0x091F	8	DC – Receive Time EPU	dc	dc	dc	dc	dc	dc	dc	dc	dc

Address	Length (Byte)	Description	IP Core V2.4.0-V2.4.4/ V2.04a-V2.04e Register set			IP Core V2.3.0-V2.3.2/ V2.03a-V2.03d Register set			IP Core V2.2.1/V2.2.0/ V2.02a Register set		
			S	M	L	S	M	L	S	M	L
0x0920:0x0935	24	DC – Time Loop Control Unit	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0936	1	DC – Receive Time Latch mode	-	-	-	-	-	-	-	-	-
0x0980	1	DC – Cyclic Unit Control	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0981	1	DC – Activation	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0982:0x0983	2	DC – Pulse length of SyncSignals	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0984	1	DC – Activation Status	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x098E:0x09A7	26	DC – SYNC Out Unit	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09A8	1	DC – Latch0 Control	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09A9	1	DC – Latch1 Control	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09AE	1	DC – Latch0 Status	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09B0:0x09B7	8	DC – Latch0 Positive Edge	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09B8:0x09BF	8	DC – Latch0 Negative Edge	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09C0:0x09C7	8	DC – Latch1 Positive Edge	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09C7:0x09CF	8	DC – Latch1 Negative Edge	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09F0:0x09F3 0x09F8:0x09FF	12	DC – SyncManager Event Times	c	c	c	c	c	c	c	c	c
0x0E00:0x0E03	4	Power-On Values (Bits)	-	-	-	-	-	-	-	-	-
0x0E00:0x0E07	8	Product ID	x	x	x	x	x	x	x	x	x
0x0E08:0x0E0F	8	Vendor ID	x	x	x	x	x	x	x	x	x
0x0F00:0x0F03	4	Digital I/O Output Data	io	io	io	io	io	io	io	io	io
0x0F10:0x0F17	8	General Purpose Outputs [Byte]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0x0F18:0x0F1F	8	General Purpose Inputs [Byte]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0x0F80:0x0FFF	128	User RAM	x	x	x	x	x	x	x	x	x
0x1000:0x1003	4	Digital I/O Input Data	io	io	io	io	io	io	io	io	io
0x1000 ff.		Process Data RAM [Kbyte]	1-60	1-60	1-60	1-60	1-60	1-60	1-60	1-60	1-60

Table 10: Legend

Symbol	Description
x	Available
-	Not available
r	Read only
c	Configurable
dc	Available if Distributed Clocks with all Sync/Latch signals are enabled
rt	Available if Receive Times or Distributed Clocks are enabled (always available for 3-4 ports)
io	Available if Digital I/O PDI is selected
red	Register changed in this version

2.3 Extended ESC Features in User RAM

Table 11: Extended ESC Features (Reset values of User RAM – 0x0F80:0x0FFF)

Bit	Description	small	medium	large
7:0	Number of extended feature bits	142		
	IP Core extended features:	0: Not available 1: Available c: Configurable		
8	Extended DL Control Register (0x0102:0x0103)	1	1	1
9	AL Status Code Register (0x0134:0x0135)	c	1	1
10	ECAT Interrupt Mask (0x0200:0x0201)	c	c	1
11	Configured Station Alias (0x0012:0x0013)	1	1	1
12	General Purpose Inputs (0x0F18:0x0F1F)	c	c	c
13	General Purpose Outputs (0x0F10:0x0F17)	c	c	c
14	AL Event Mask (0x0204:0x0207)	c	1	1
15	Physical Read/Write Offset (0x0108:0x0109)	c	c	1
16	Watchdog divider writeable (0x0400:0x04001) and Watchdog PDI (0x0410:0x0f11)	c	1	1
17	Watchdog counters (0x0442:0x0443)	c	c	1
18	Write Protection (0x0020:0x0031)	c	c	1
19	Reset (0x0040:0x0041)	c	c	c
20	Reserved	0	0	0
21	DC SyncManager Event Times (0x09F0:0x09FF)	c	c	1
22	ECAT Processing Unit/PDI Error Counter (0x030C:0x030D)	c	c	1
23	EEPROM Size configurable (0x0502.7): 0: EEPROM Size fixed to sizes up to 16 Kbit 1: EEPROM Size configurable	1	1	1
24	Reserved	1	1	1
25	Reserved	0	0	0
26	Reserved	0	0	0
27	Lost Link Counter (0x0310:0x0313)	c	1	1
28	MII Management Interface (0x0510:0x0515)	c	c	c
29	Enhanced Link Detection MII	c	c	c
30	Enhanced Link Detection EBUS	0	0	0
31	Run LED (DEV_STATE LED)	c	c	c
32	Link/Activity LED	1	1	1
33	Reserved	0	0	0
34	Reserved	1	1	1
35	Reserved	1	1	1
36	Reserved	0	0	0
37	Reserved	1	1	1
38	DC Time loop control assigned to PDI	c	c	c
39	Link detection and configuration by MI	c	c	c
40	MI control by PDI possible	1	1	1
41	Automatic TX shift	c	c	c
42	EEPROM emulation by µController	c	c	c
43	Reserved	0	0	0

Bit	Description	small	medium	large
44	Reserved	0	0	0
45	Reserved	0	0	0
46	Reserved	0	0	0
47	Reserved	0	0	0
48	Reserved	0	0	0
49	Reserved	0	0	0
50	ERR LED, RUN/ERR LED Override	0	0	0
others	Reserved	0	0	0

3 IP Core Installation

3.1 Installation on Windows PCs

3.1.1 System Requirements

The system requirements of Altera Quartus II are applicable.

3.1.2 Installation

For installation of the EtherCAT IP Core on your system run the setup program

"EtherCAT IP core for Altera FPGAs <version> Setup.exe"

and follow the instructions of the installation wizard.

The EtherCAT IP Core, example designs, and documentation are typically installed in the directory

C:\BECKHOFF\ethercat_altera_v<version>

This folder is further referenced to as *<IPInst_dir>*.

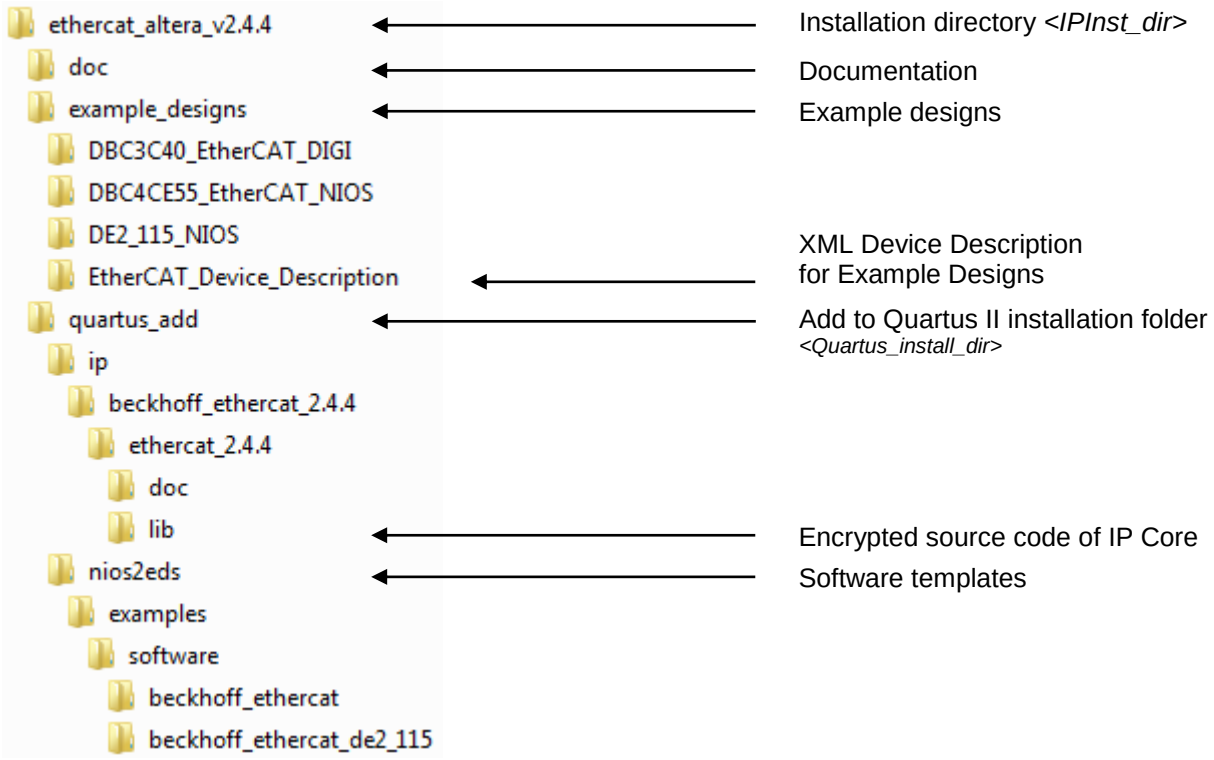

Figure 4: Files installed with EtherCAT IP Core setup

3.2 Installation on Linux PCs

3.2.1 System Requirements

The system requirements of Altera Quartus II are applicable.

3.2.2 Installation

For installation of the EtherCAT IP Core extract the archive to any folder on your Linux PC (same contents as on windows PCs):

1. Create installation directory, , e.g. */opt/beckkhoff/* :
   ```
   # mkdir /opt/beckhoff
   ```
2. Change to installation directory
   ```
   # cd /opt/beckhoff
   ```
3. Copy EtherCAT IP Core archive to installation folder
4. Extract the EtherCAT IP Core:
   ```
   # tar -xf EtherCAT_IP_core_for_Altera_FPGAs_<version>_Linux_
     <region>.tar.gz
   ```
5. Continue with the following installation chapters.

The folder

ethercat_<version>

created inside this directory is further referenced to as <*IPInst_dir*>.

3.3 Files located in the lib folder

Table 12: Contents of lib folder

File name	Description
beckhoff.jpg	BECKHOFF image used in MegaWizard
ethercat_<version>.qprs	Quartus MegaWizard presets
ethercat_<version>_hw.tcl	Quartus MegaWizard and Qsys IP hardware configuration TCL
ethercat_<version>_wizard.lst	Quartus MegaWizard list
ETHERCAT_IPCORE.VHD	Encrypted EtherCAT IP Core source code (core)
ETHERCAT_IPCORE_TOP.VHD	Encrypted EtherCAT IP Core soruce code (top level)
ETHERCAT_IPCORE.ocp	OpenCorePlus description for EtherCAT IP Core
ETHERCAT_VENDORID.VHD	Vendor ID package (added during installation, not part of setup)

3.4 License File

The license file for the EtherCAT IP Core (license_<company>_<Dongle/MAC ID>.dat) has to be linked to the Altera Quartus Development environment. The EtherCAT IP Core can only be used with the delivered license file.

The location of your Altera license file can be found in the Altera Quartus License Setup ("Tools – License Setup…" from the menu.:

There are three options:

a) Add the path of the license file (separated by a semicolon) to the License file input box in the Altera Quartus License Setup.
b) Add the path of the license file to the LM_LICENSE_FILE environment variable (separated by a semicolon) if the LM_LICENSE_FILE variable is used.
c) Add the content of the EtherCAT IP Core license file to an existing license file.

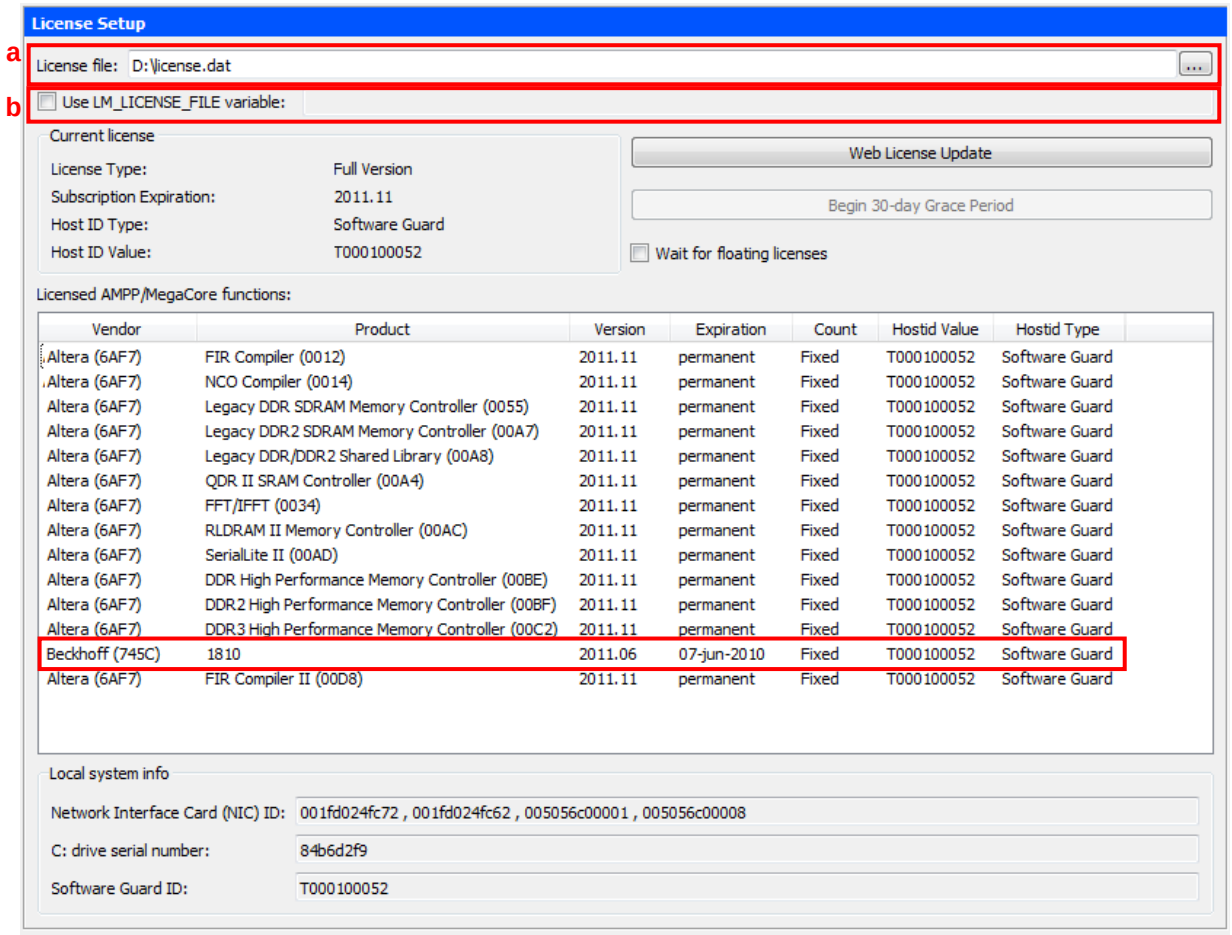

Figure 5: License Setup

The EtherCAT IP Core license is shown in Licensed AMPP/MegaCore® functions list: the Vendor is Beckhoff (745C), and the Product number is 1810.

For further information regarding license setup, refer to Altera Application Note 340 "Altera Software Licensing", found at the Altera homepage http://www.altera.com.

3.5 IP Core Vendor ID package

The Vendor ID Package (VHDL file) is part of the EtherCAT IP Core source code, and it contains your company's unique vendor ID. The vendor ID package is not part of the IP Core setup, it is delivered separately.

Copy the IP Core Vendor ID package (*pk_ECAT_VENDORID_<company>_Altera.vhd)* to the lib folder in the IP Core Directory.

<IPInst_dir>\quartus_add\ip\beckhoff_ethercat_<version>\ethercat_<version>\lib

Rename (or copy) the Vendor ID package to

ETHERCAT_VENDORID.VHD

(exact naming and upper case is necessary).

The steps of adding the IP Core Vendor ID package into the IP Core installation folder can also be performed by the EtherCAT IP Core Setup program (Windows PCs only). Just check the appropriate option and select the path to your *pk_ECAT_VENDORID_<company>_Altera.vhd* file, and the Setup program will perform all necessary steps.

A vendor ID package is required for both evaluation and full license. It is recommended to use an evaluation vendor ID (package) for evaluation, and the original vendor ID for production. The evaluation vendor ID is beginning with "0xE....." and ends with the original vendor ID digits. Evaluation vendor IDs cannot pass the EtherCAT conformance tests.

3.6 Integrating the EtherCAT IP Core into the Altera Designflow

Quartus II expects all IP cores to be installed into

<*Quartus installation folder*>*ip*\

This can be done by the windows setup program automatically if it recognizes the Quartus II installations on the disk. The EtherCAT IP core can also be integrated into Quartus II installations manually by copying the contents of the

<*IPInst_dir*>*quartus_add*\

folder to the Quartus installation folder.

3.6.1 Software Templates for example designs with NIOS processor

Software example templates are available for example designs with NIOS processor. The templates are part of the *quartus_add* folder, they will be copied to your NIOS II installation folder.

Source folder:

<*IPInst_Dir*>*quartus_add**nios2eds**examples**software*

Destination folder:

<*Quartus_Install_Dir*>*nios2eds**examples**software*

The NIOS demo applications are not suitable for production, they cannot be certified. Use the EtherCAT Slave Stack Code (SSC, available from the ETG) for products.

3.7 EtherCAT Slave Information (ESI) / XML device description for example designs

If you want to use the example designs, add the ESI to your EtherCAT master/EtherCAT configuration tool/network configurator.

The ESI is located at

<*IPInst_dir*>*example_designs**EtherCAT_Device_Description**BECKHOFF ET1810.xml*

If you are using TwinCAT, add the ESI to the appropriate folder of your TwinCAT installation before the System Manager is started:

- TwinCAT 2: <*TwinCAT installation folder*>*Io**EtherCAT*
- TwinCAT 3: <*TwinCAT installation folder*>\<*TwinCAT version*>*Config**Io**EtherCAT*

4 IP Core Usage

4.1 IP Catalog

The EtherCAT IP Core is integrated in the Quartus II IP Catalog, you can add it to your Quartus II project like any other IP and configure it with the MegaWizard.

The output of the MegaWizard is a VHDL or Verilog wrapper for the EtherCAT IP Core. The wrapper file makes only those signals and interfaces visible, which are required, and it configures the EtherCAT IP Core using generics as desired.

A synthesizable EtherCAT IP Core consists of the user generated VHDL wrapper, the encrypted EtherCAT IP Core files, and the vendor ID package (*ECAT_VENDORID.vhd*). These files, together with a PLL, represent the minimum source set for a fully functional EtherCAT slave. Typically, additional user logic is added inside the FPGA.

4.2 Qsys

The EtherCAT IP Core can also be integrated into a System on a Programmable Chip (SoPC) with a processor inside the FPGA (e.g., Altera NIOS II processor). The EtherCAT IP Core and the processor can communicate via an Avalon on-chip bus system.

For building an SoPC including the EtherCAT IP Core, Altera Qsys is used (Figure 6). The NIOS processor and the EtherCAT IP Core as well as other resources which might be used for an SOPC are listed under the System Resources (Figure 6). Signal Routing is done automatically. The interrupts used by the EtherCAT IP Core (avalon_ethercat_sync0, avalon_ethercat_sync1, and PDI collector interrupt avalon_ethercat_slave) are listed and signal routing is shown.

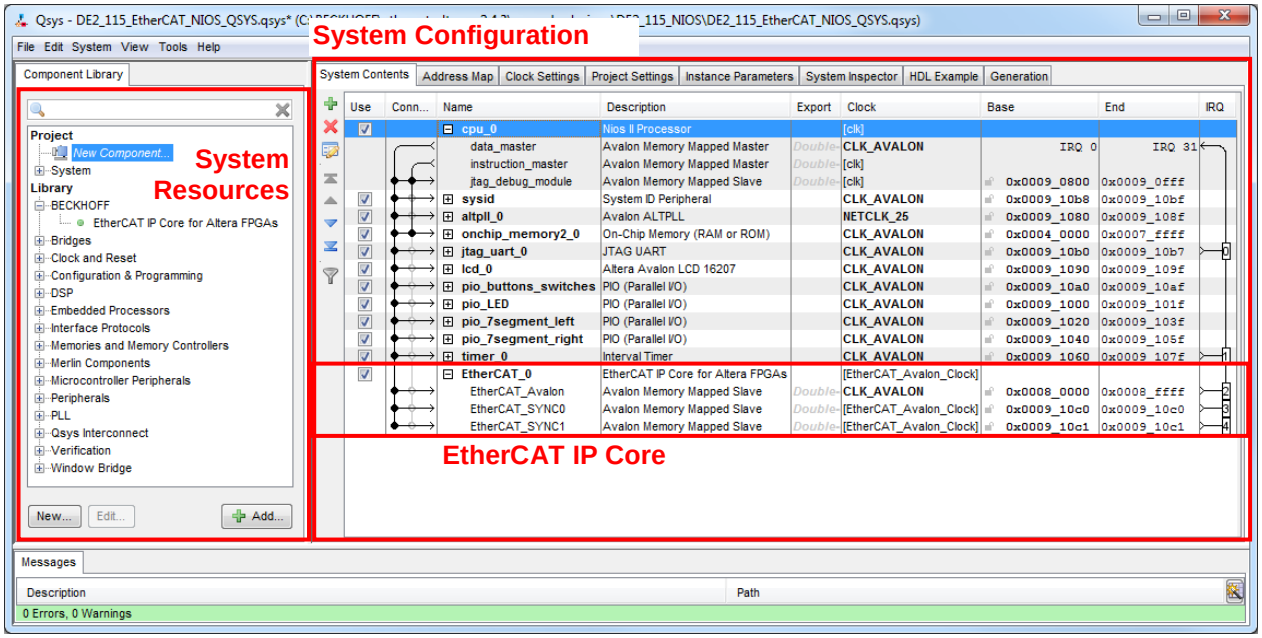

Figure 6: Qsys with EtherCAT IP Core

5 IP Core Configuration

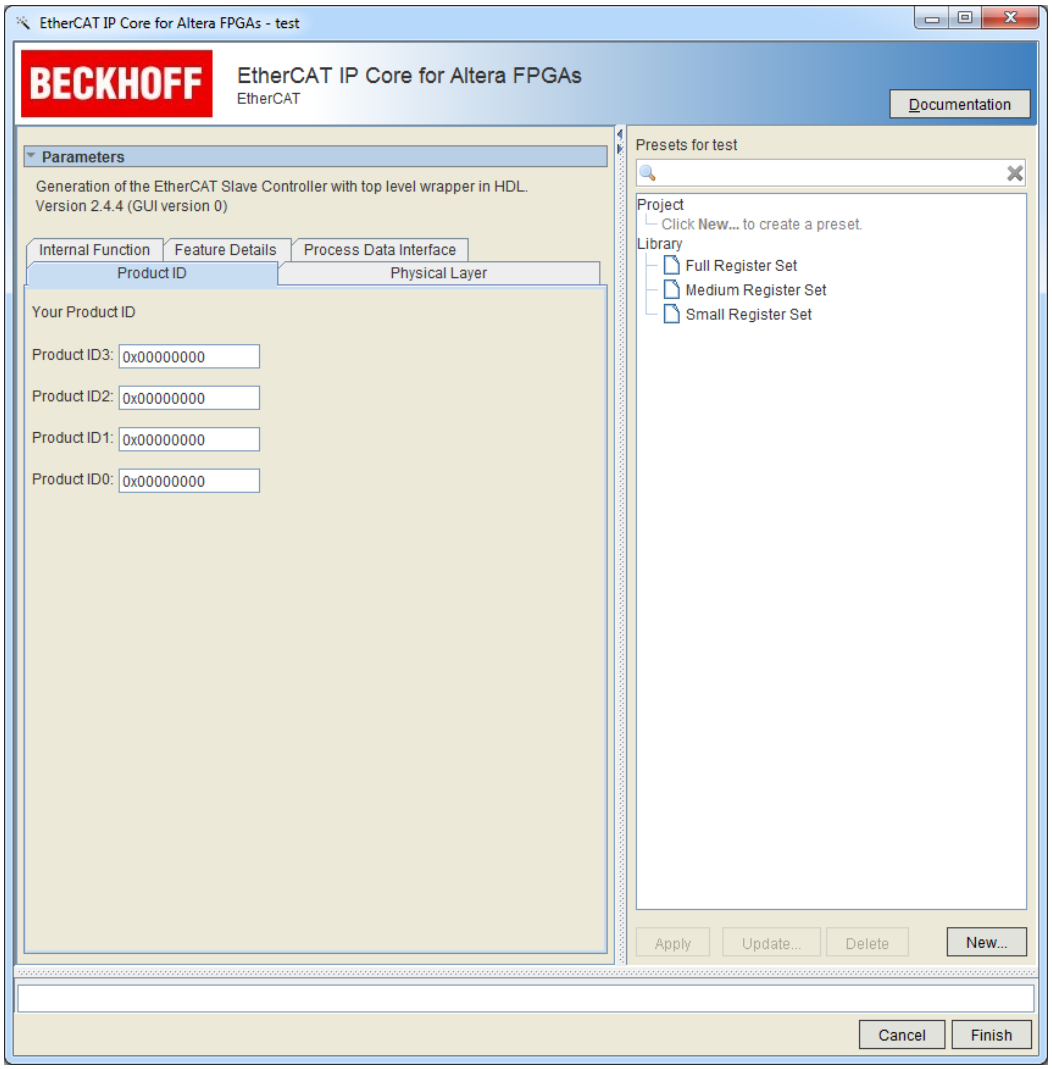

Figure 7: EtherCAT IP Core Configuration Interface

Documentation button

Documentation on the MegaWizard, the EtherCAT IP Core and the configuration options

Parameters pane (left)

The configuration options for the EtherCAT IP Core are available in the IP Core parameters pane on the left side.

Presets pane (right)

Depending on the IP Core functionality that should be implemented and the available resources (LEs) in the FPGA, the internal features can be chosen. Three feature presets are available: small, medium and large. Based upon these presets, additional functions can be enabled in the parameter pane.

Message pane (bottom)

In the lower box additional information like warnings and errors are displayed.

5.1 Documentation

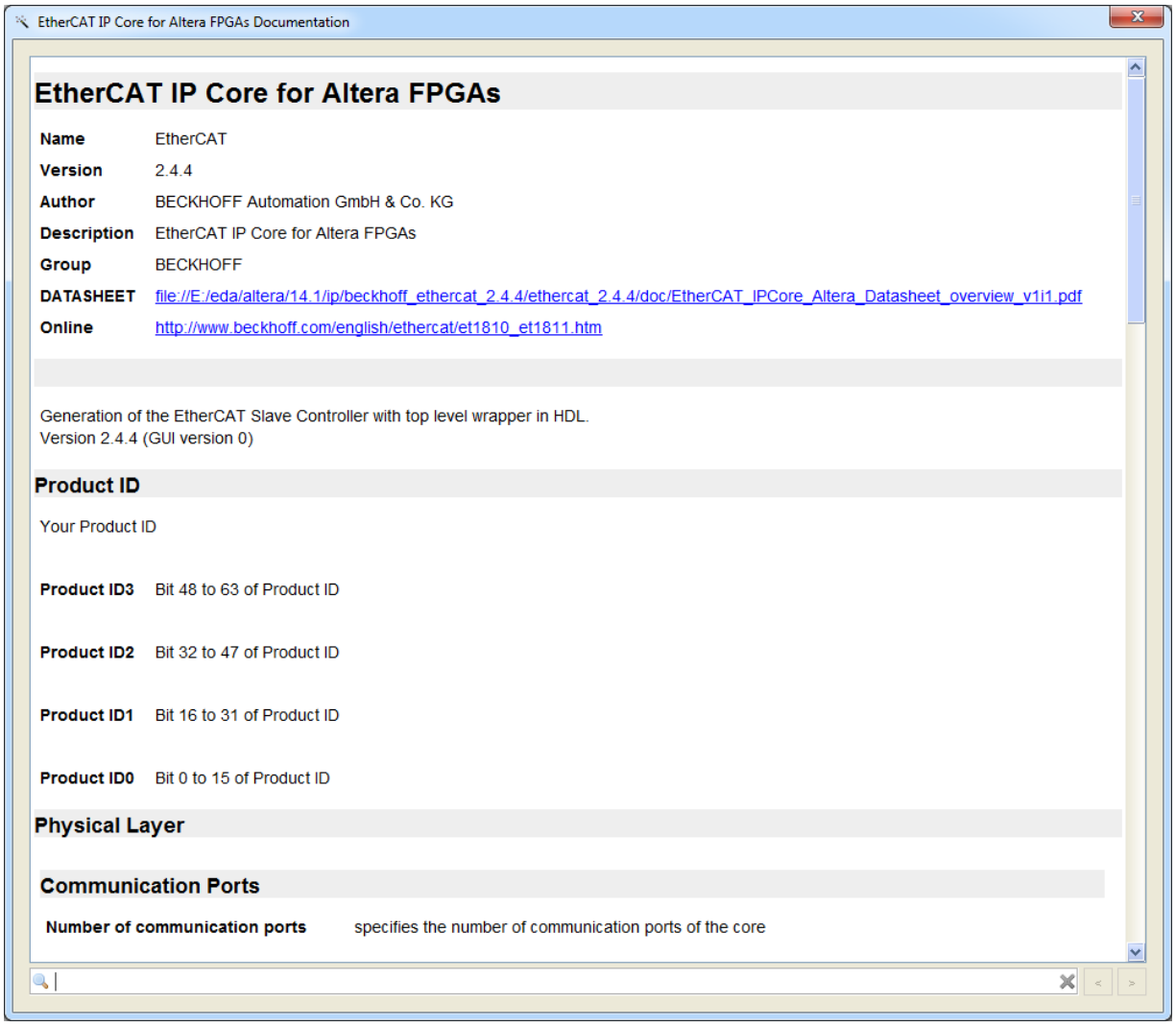

Figure 8: Documentation

General information

Name and version of the IP Core, as well as links to the datasheet and online support are given.

Parameters

Short descriptions on the various parameters of the parameters pane can be found here.

5.2 Parameters

5.2.1 Product ID tab

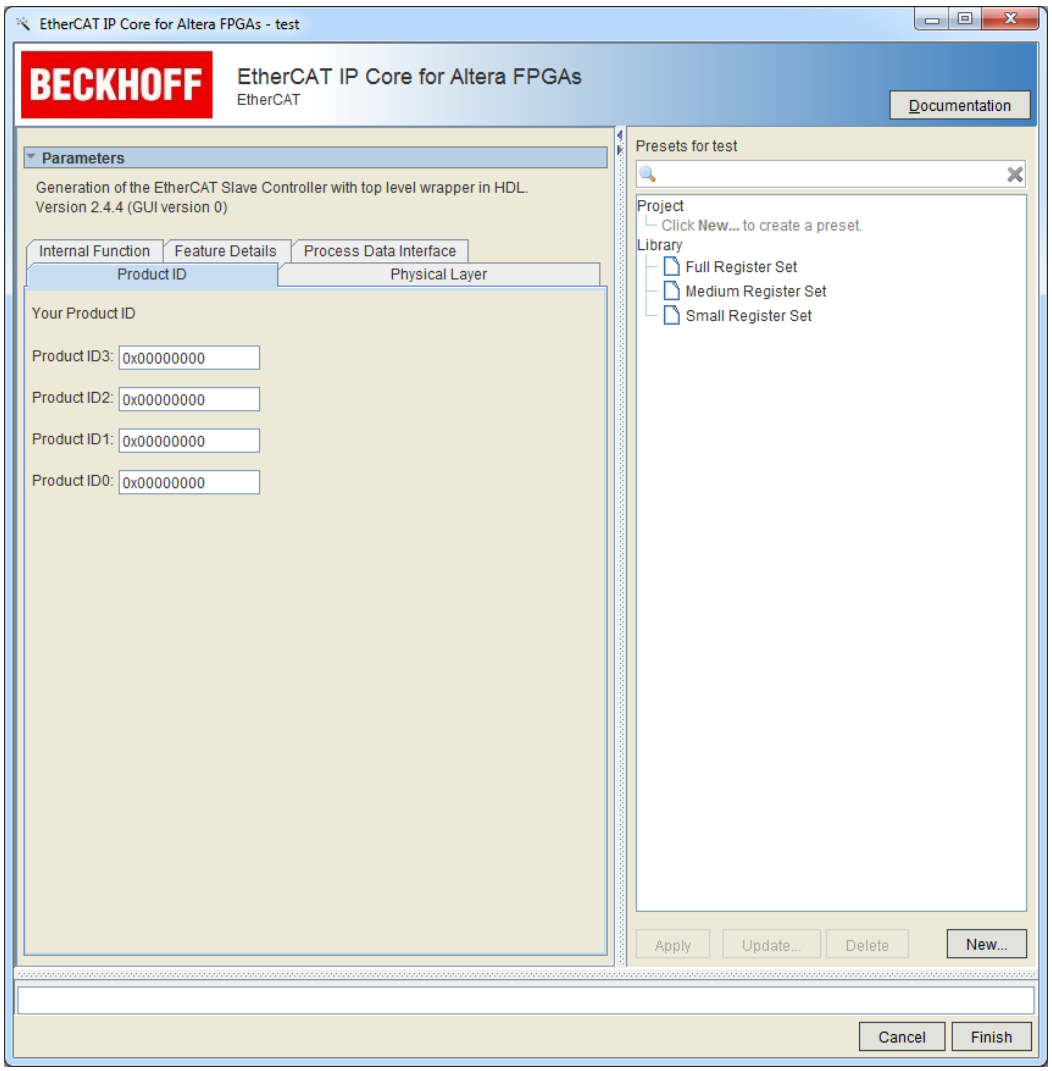

Figure 9: Product ID tab

PRODUCT_ID input in hexadecimal groups
The Product ID can be chosen freely and is for vendor issues. It can be read out in register 0x0E08:0x0E0F.

The PRODUCT_ID has to be entered in hexadecimal format for each of the four 16 bit fields (representing a 16 bit part of the 64 bit Product ID each).

The Product ID is meant to identify special configurations of the IP Core. It does not have to reflect the EtherCAT slave product code, which is part of the EEPROM/XML device description.

NOTE: The current GUI seems to allow 32 bit entries for each of the 4 16 bit fields, but this is not true. This is an Altera MegaWizard configuration restriction.

5.2.2 Physical Layer tab

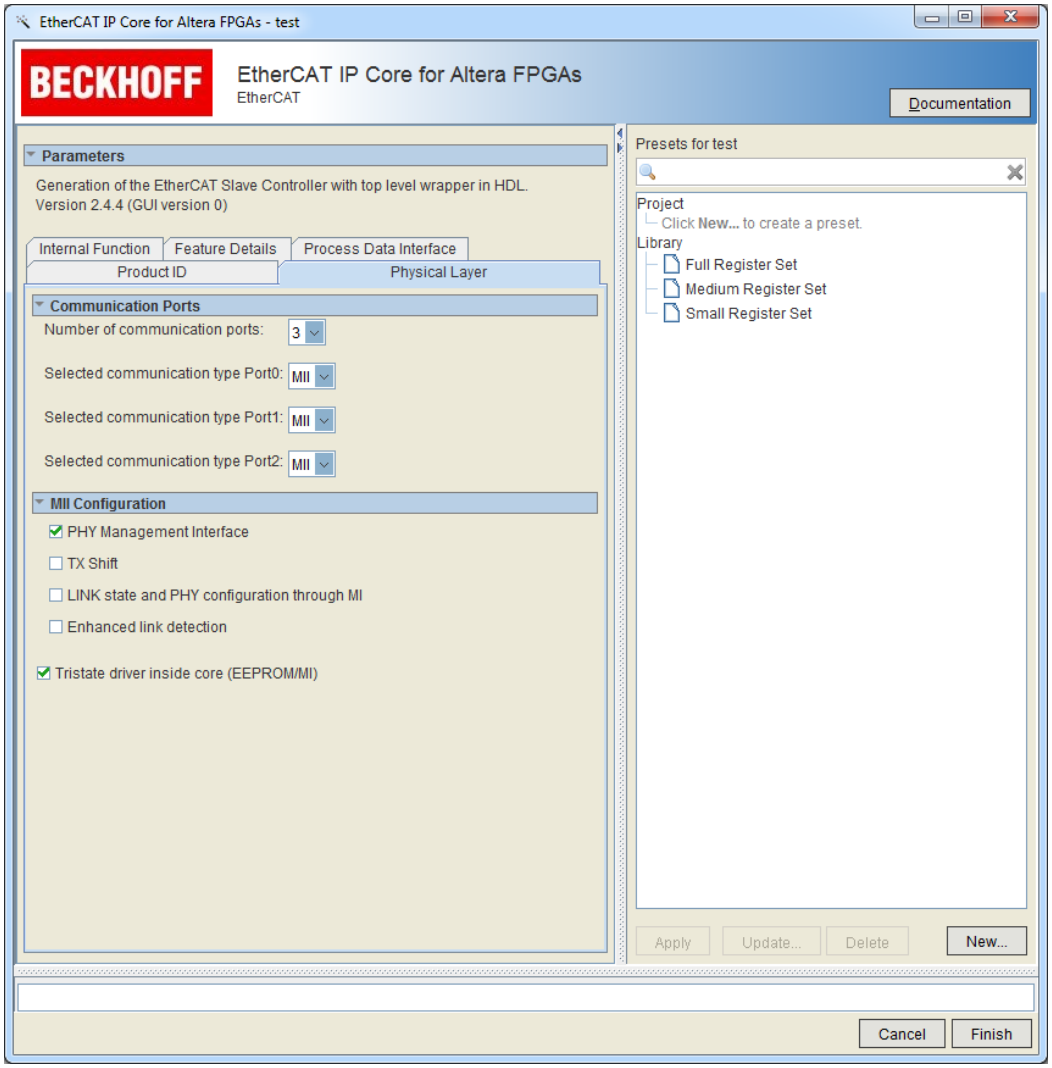

Figure 10: Physical Layer tab

Communication Ports

The number of communication ports by default is two. As PHY interface MII (1, 2, or 3 ports) or RMII (1 or 2 ports only) can be selected. It is recommended to use MII as for accuracy of the distributed clocks is much better with MII.

PHY Management Interface

The PHY Management Interface function can be selected or deselected. If it deselected, the other MII Configuration options are not available.

TX Shift

Automatic or manual TX Shift is available if TX Shift is selected. TX Shift delays MII TX signals to comply to Ethernet PHY setup and hold timing. Automatic TX Shift uses the TX_CLK signals of the PHYs to detect appropriate TX Shift settings automatically. Manual TX Shift configuration allows for delaying the MII TX signals by 0, 10, 20, or 30 ns.

LINK state and PHY configuration through MI

MI link detection and configuration is available if checked. Ethernet PHYs are configured and link status is polled via the MII Management Interface. Enhanced link detection has to be activated if MI link detection and configuration is used and the nMII_LINK0/1/2 signals are not used.

Enhanced link detection

Enhanced MII link detection is a mechanism of informing link partners of receive errors.

Tristate Driver inside core (EEPROM/MI)
If selected tri-state drivers of the core are used for access to EEPROM and PHY Management signals.

5.2.3 Internal Functions tab

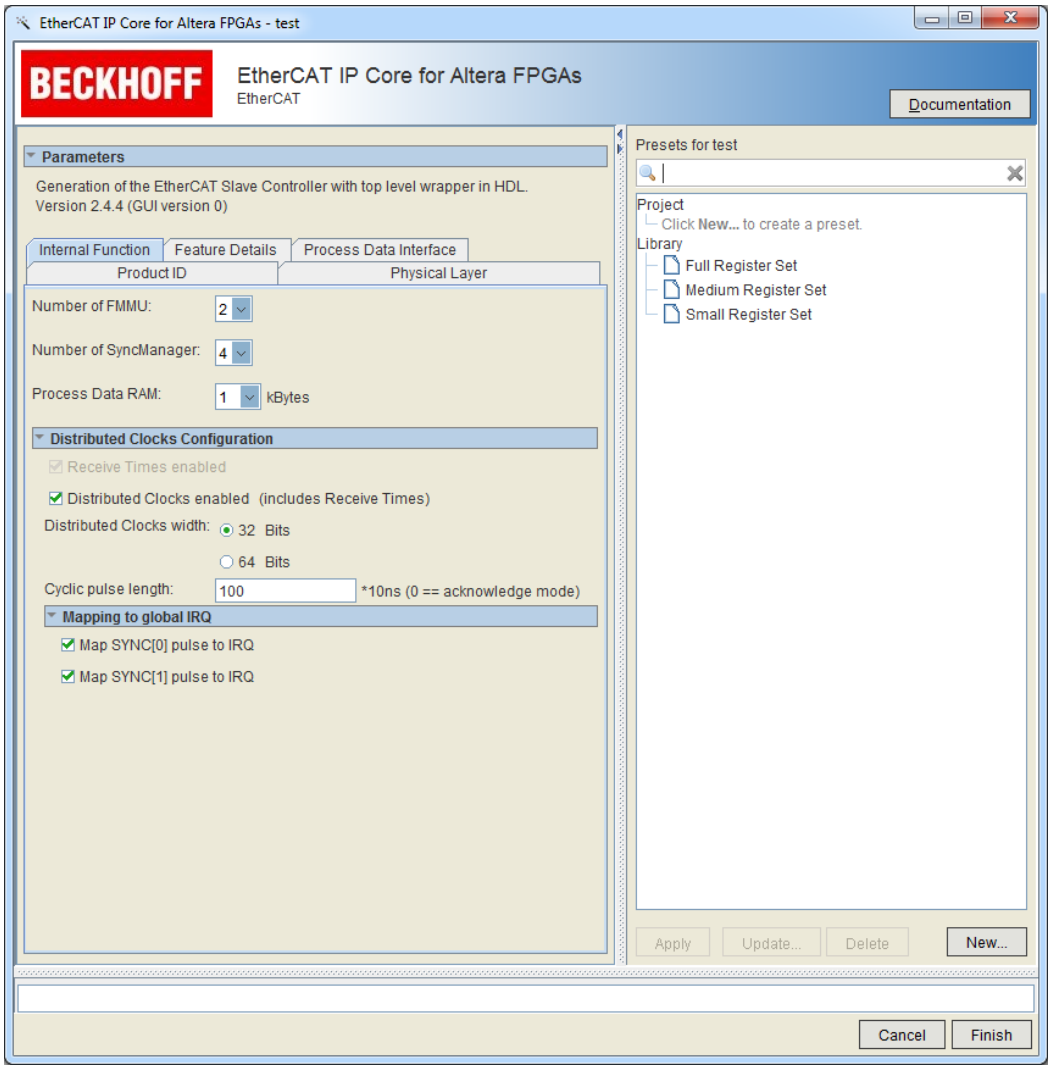

Figure 11: Internal Functions tab

FMMUs
Number of FMMU instances. Between 0 and 8 FMMUs are possible.

SyncManager
Number of SyncManager instances. Between 0 and 8 SyncManagers are possible.

Process Data RAM
The size of the Process data memory can be determined in this dialog. Minimum memory size is 1 KByte, maximum memory size is 60 KByte.

Receive Times enabled
The Distributed Clocks receive time feature for propagation delay calculation can be enabled without using all DC features.

Distributed Clocks enabled
The Distributed Clocks feature comprises synchronized distributed clocks, receive times, SyncSignal generation, and LatchSignal time stamping.

Distributed Clocks Width
The width of the Distributed Clocks can be selected to be either 32 bit or 64 bit. DC with 64 bit require more FPGA resources. DC with 32 bit and DC with 64 bit are interoperable.

Cyclic pulse length
Determines the length of SyncSignal output (register 0x0982:0x0983).

Mapping to global IRQ
Sync0 and Sync1 can additionally be mapped internally to the global IRQ. This might be a good solution if a microcontroller interface is short on IRQs. However, the sync signals will remain available on Sync0 and Sync1 outputs.

5.2.4 Feature Details tab

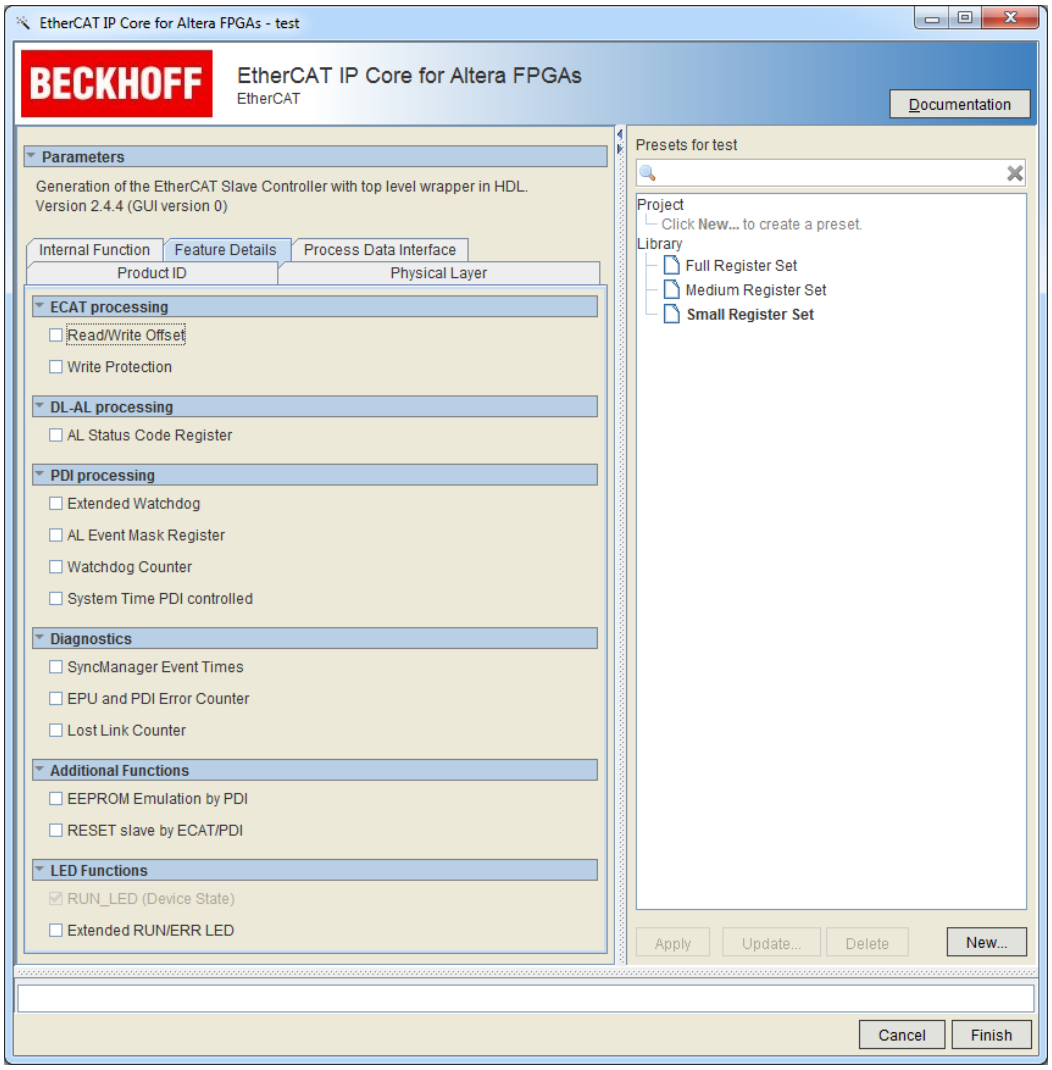

Figure 12: Feature Details tab

Read/Write Offset
Physical Read/Write Offset (0x00108:0x0109) is available if checked.

Write Protection
Register write protection and ESC write protection (0x0020:0x0031) are available if checked.

AL Status Code Register
AL Status Code register (0x0134:0x0135) is available if checked.

Extended Watchdog
Watchdog Divider (0x0400:0x0401) is configurable and PDI Watchdog (0x0410:0x0411, and 0x0100.1) is available if checked.

AL Event Mask Register
AL Event Mask register (0x0204:0x0207) is available if checked.

Watchdog Counter
Watchdog Counters (0x0442:0x0443) are available if checked. Watchdog Counter PDI is only used if Extended Watchdog feature is selected.

System Time PDI controlled
Distributed Clocks Time Loop Control Unit is controlled by PDI (µController) if selected. EtherCAT access is not possible. Used for synchronization of secondary EtherCAT busses.

SyncManager Event Times
Distributed Clocks SyncManager Event Times (0x09F0:0x09FF) are available if checked. Used for debugging SyncManager interactions.

EPU and PDI Error Counter
EtherCAT Processing Unit (EPU) and PDI Error counters (0x030C:0x030D) are available if checked.

Lost Link Counter
Lost Link Counters (0x0310:0x0313) are available if checked.

EEPROM Emulation by PDI
EEPROM is and has to be emulated by a µController with access to a NVRAM. I²C EEPROM is not necessary if EEPROM Emulation is activated, I²C interface is deactivated. Only usable with PDIs for µController connection.

RESET slave by ECAT/PDI
The reset registers (0x0040:0x0041) and the RESET_OUT signal is available if this feature is checked.

RUN LED (Dev_State)
RUN LED output (DEV_STATE) indicates AL Status (0x0130) if activated. Otherwise RUN LED has to be controlled by a µController. Always activated if no PDI is selected or if Digital I/O PDI is selected.

Extended RUN/ERR LED
Support for ERR LED and STATE LED, direct control of RUN/ERR LED via RUN/ERR LED Override register (0x0138:0x0139).

5.2.5 Process Data Interface tab

Several interfaces between ESC and the application are available:

- Digital I/O
- 8 Bit asynchronous µController
- 16 Bit asynchronous µController
- SPI slave
- Avalon MM slave
- General Purpose I/O

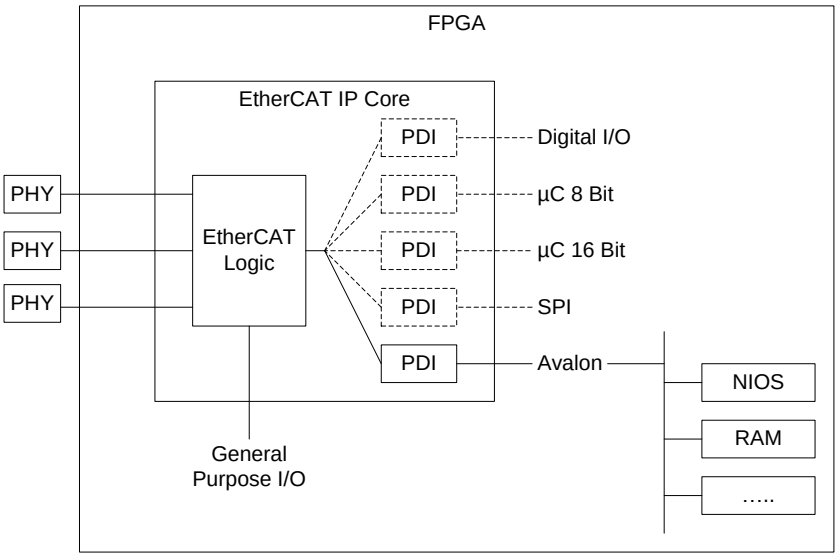

Figure 13: Available PDI Interfaces

The PDI can be selected from the pull down menu. After selection settings for the selected PDI are shown and can be changed. If the EtherCAT IP Core is used in Qsys, only the Avalon bus is selectable.

5.2.5.1 No Interface and General Purpose I/O

If there is no interface selected no communication with the application is possible (except for general purpose I/O).

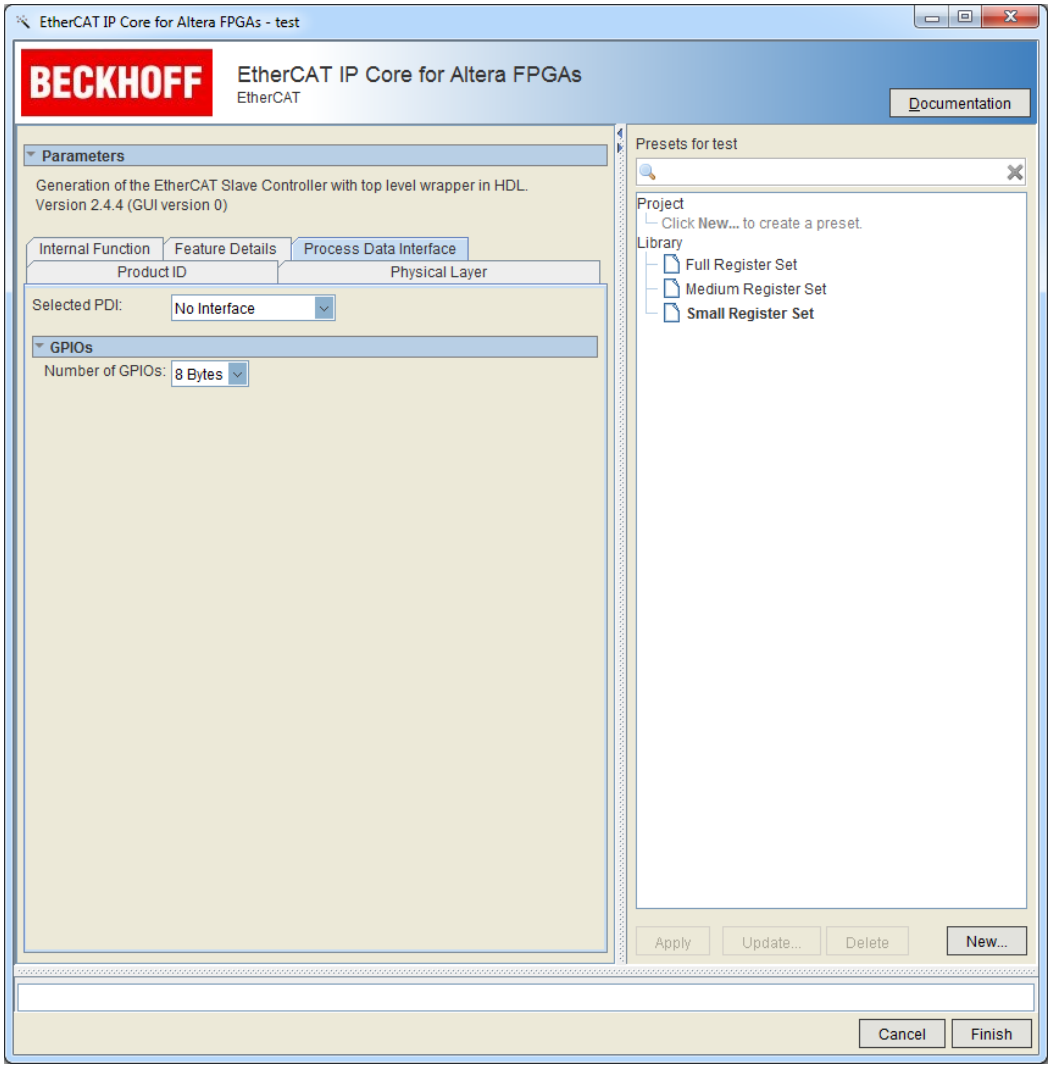

Figure 14: Register Process Data Interface

Number of GPIOs
General purpose I/O signals can be added to any selected PDI. The number of GPIO bytes is configurable to 0, 1, 2, 4, or 8 Bytes. Both general purpose outputs and general purpose inputs of the selected width are available.

5.2.5.2 Digital I/O Configuration

The Digital I/O PDI supports up to 4 Bytes of digital I/O signals. Each byte can be assigned as input or output byte.

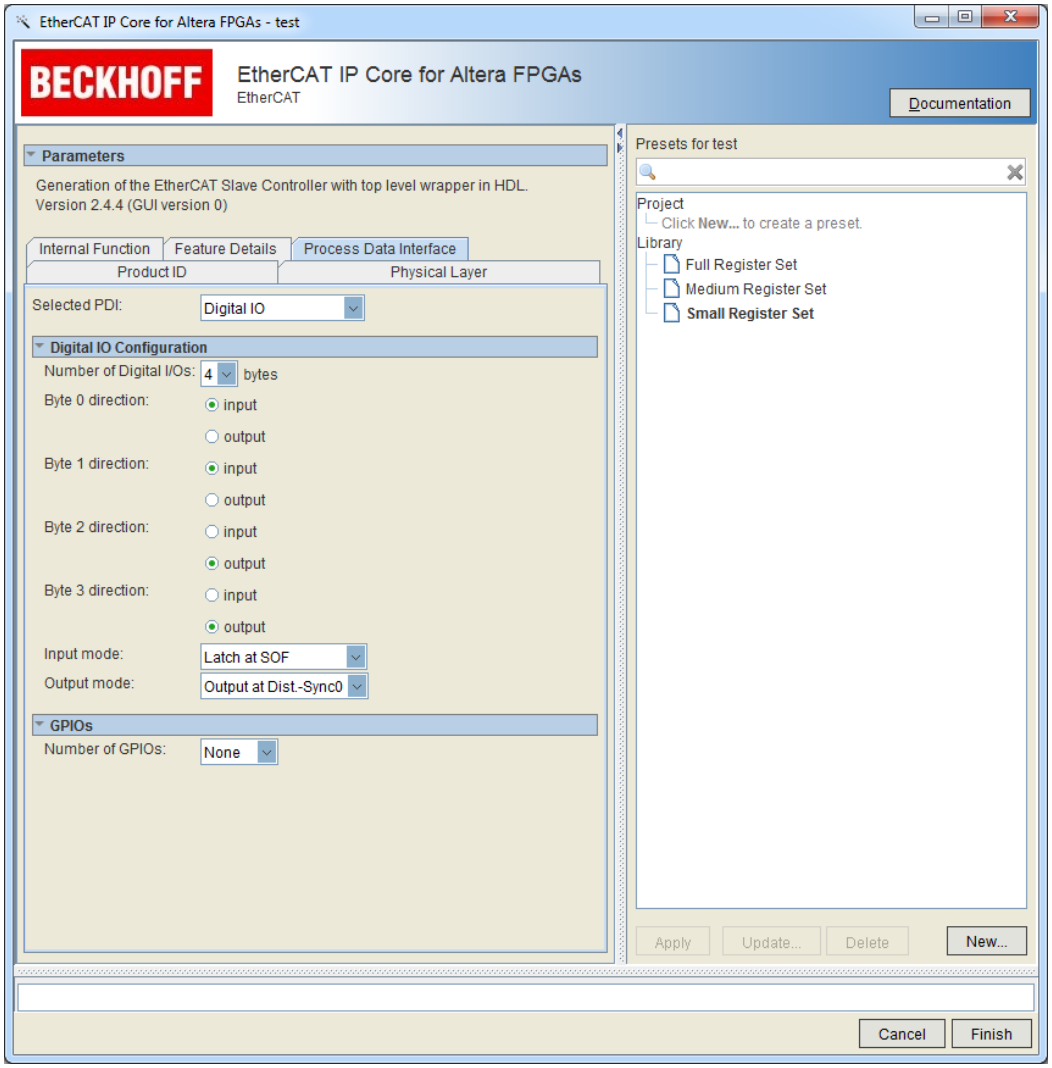

Figure 15: Register PDI – Digital I/O Configuration

Number of digital I/Os
Total number of I/Os. Possible values are 1, 2, 3 or 4 Bytes.

Byte 0-3 direction
Defining byte-wise if digital I/Os are used as input or output byte

Input Mode
Defines the latch signal which is used to take over input data.

- Latch at SOF (Start of Frame)
  The inputs are latched just before the data have to be written in the frame.
- Latch with ext. signal
  Connected to DIGI_LATCH_IN. Application controls latching
- Latch at Dist-Sync0
  Latch input data with distributed clock Sync0 signal
- Latch at Dist-Sync1
  Latch input data with distributed clock Sync1 signal

Output Mode
Defines the trigger signal for data output.

- Output at EOF (End of Frame)
  The outputs will be set if the frame containing the data is received complete and error free.

- Output at Dist-Sync0
  Outputs will be set with Sync0 signal if distributed clocks are enabled.
- Output at Dist-Sync1
  Outputs will be set with Sync1 signal if distributed clocks are enabled.

5.2.5.3 µController Configuration (8/16Bit)

The 8/16 Bit µController interface is an asynchronous parallel interface for µControllers. The difference between 8 and 16 bit interface is the extended data bus and the BHE signal which enables access to the upper byte.

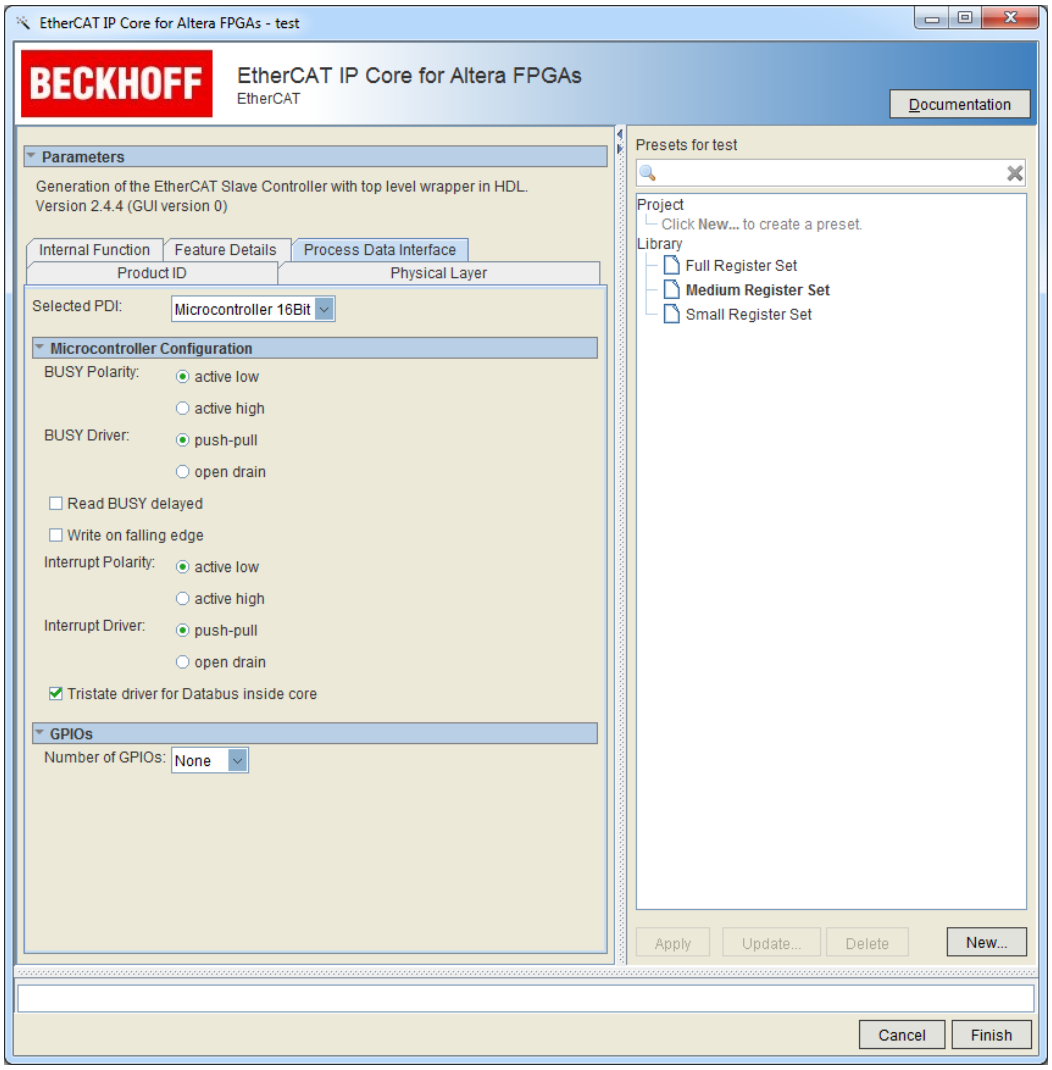

Figure 16: Register PDI – µC-Configuration

BUSY Polarity, BUSY Driver
Electrical definition of the busy signal driver

Read BUSY delayed
Delay the output of the BUSY signal by ~20 ns (refer to register 0x00152.0).

Write on falling edge
Start write access earlier with falling edge of nWR. Single write accesses will become slower, but maximum write access time becomes faster.

Interrupt Polarity, Interrupt Driver
Electrical definition of the interrupt signal driver

Tristate driver for data bus inside core
If Tristate drivers for the data bus should be integrated into the IP Core already activate the check box.

5.2.5.4 SPI Configuration

The SPI interface is a serial slave interface for µControllers.

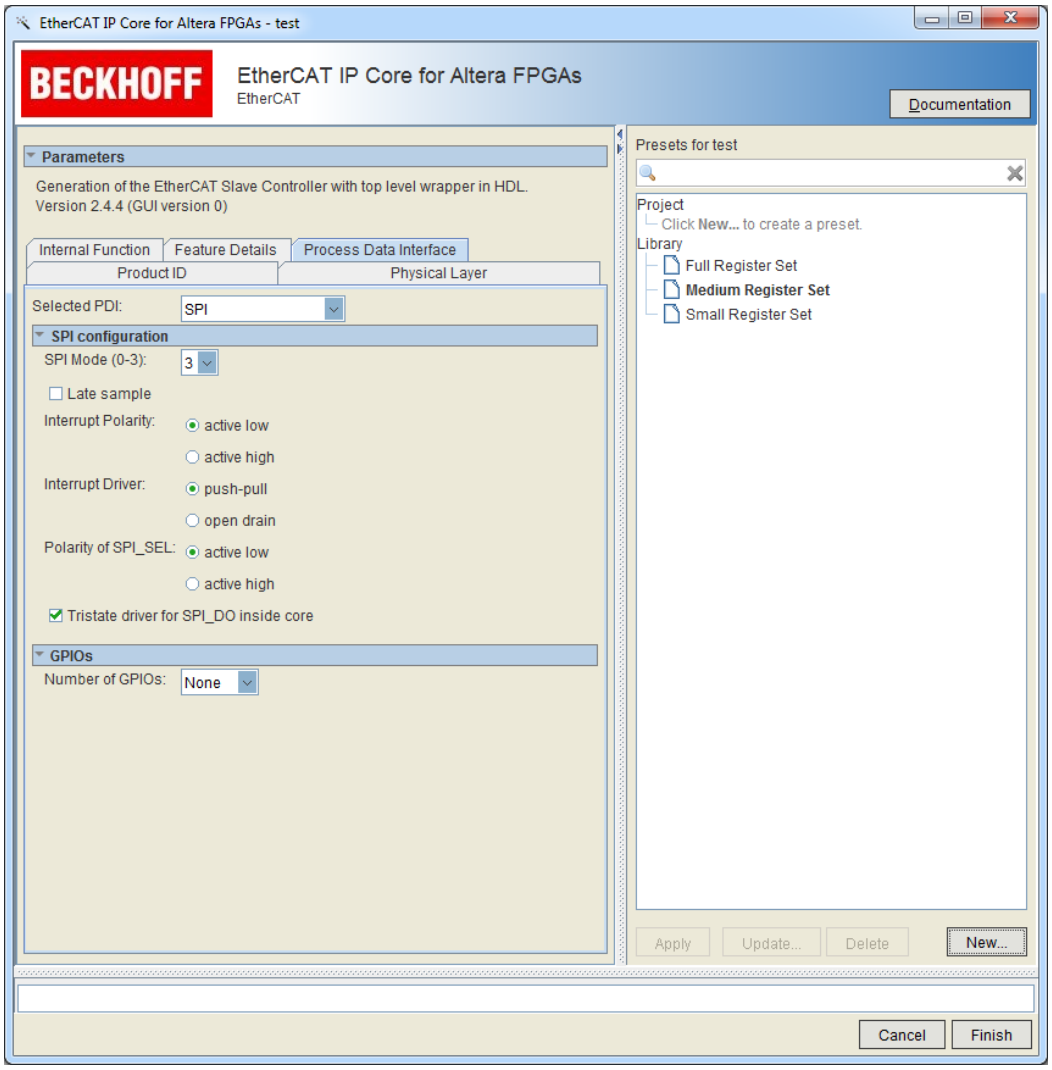

Figure 17: Register PDI – SPI Configuration

SPI Mode
The SPI mode determines the SPI timing. Refer to SPI PDI description for details. Mode 3 is recommended for slave sample code.

Late Sample
The Late Sample configuration determines the SPI timing. Refer to SPI PDI description for details. It is recommended to leave this unchecked for slave sample code.

Interrupt Polarity, Interrupt Driver
SPI_IRQ output driver configuration.

Polarity of SPI_SEL
SPI_SEL signal polarity.

Tristate driver for SPI_DO inside core
Include tri-state driver for SPI Data Out. With tri-state driver, SPI_DO is either driven actively or high impedance output.

5.2.5.5 Avalon Configuration

The Avalon PDI connects the IP Core with an Avalon Master (e.g., Altera NIOS). The Avalon PDI uses memory addressing/dynamic bus sizing. The data bus with is 8 bit, the address bus is 16 bit wide.

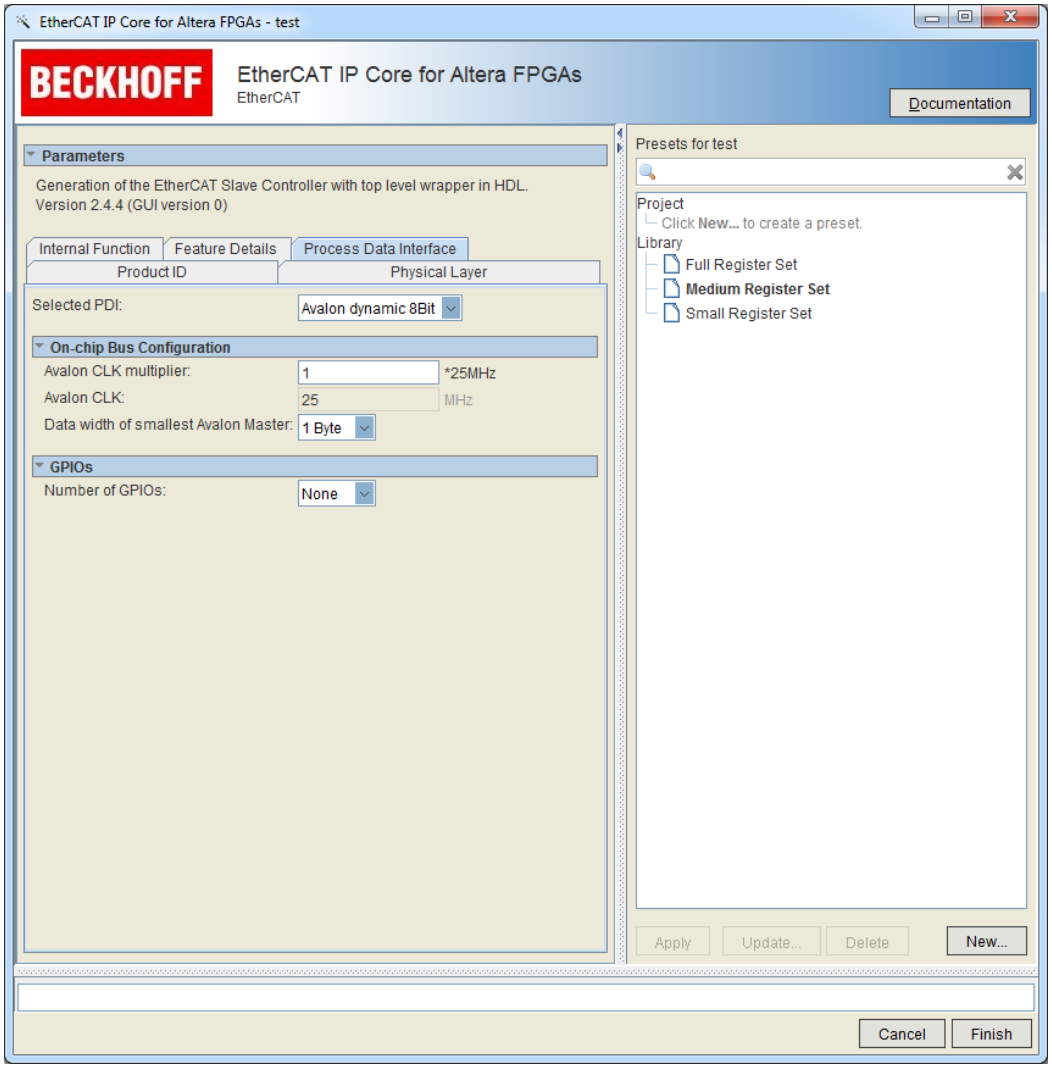

Figure 18: Register PDI – Avalon Interface Configuration

Avalon Clock Multiplier

Avalon Clock Multiplier (n*25MHz) gives the frequency of the Avalon bus clock for communication between ESC and the Avalon master.

Data width of smallest Avalon Master

Data bus width of the Avalon master with the smallest data bus, counted in bytes (1, 2, or 4 Bytes), or the smallest actual access used – also counted in bytes –, whichever is smaller.
The Avalon slave interface prefetches read data internally depending on this setting to improve read performance. Accesses with 2 Bytes have to be 2-Byte-aligned, accesses with 4 Bytes have to be 4-Byte-aligned. The IP Core ignores the lowest or the two lowest address bits for 2 Byte or 4 byte width respectively.
If the IP Core is used in Qsys systems, the Data width has to be set to 1 Byte.

6 Example Designs

Example designs are available for:

- EBV Evaluation Board DBC3C40 with RMII and 16 bit input/16 bit output Digital I/O
- EBV Evaluation Board DBC4CE55 with RMII and NIOS processor
- Altera DE2-115 Development and Education Board/Industrial Networking Kit (INK) with MII and NIOS processor

The EtherCAT master uses an XML file which describes the device and its features. The XML device description file for all example designs and its schema can be found in the installation directory.

<IPInst_dir>\example_designs\EtherCAT_Device_Description

Projects have to be compiled and then can be loaded to the EPCS configuration devices of the evaluation board.

The EtherCAT IP core example design resource consumption figures are based on EtherCAT IP Core for Altera FPGAs Version 2.4.0 and Altera Quartus II 10.1.

6.1 EBV DBC3C40 with Digital I/O

6.1.1 Configuration and resource consumption

Table 13: Resource consumption Digital I/O example design DBC3C40

Configuration		Resources		EP3C40
FMMU	2	LEs	7,177	18 %
SyncManager	4	Registers	3,446	8 %
RAM	1 KB	Pins	287	86 %
Register set	Small	M9K	3	2 %
Distributed Clocks	Disabled	Multipliers	0	0 %
PDI	2 Byte IN, 2 Byte OUT, SOF/EOF	PLLs	1	25 %

NOTE: The board uses two individual PHY management interfaces, with both PHYs having the same PHY addresses. Additionally, some of the PHY address bits have to be configured by extra logic inside the FPGA. Because of the identical PHY addresses, the management interfaces on the board cannot be combined to one, and thus, the EtherCAT IP Core cannot make use of the MII management interfaces of the PHY.

The Ethernet PHYs used on the DBC3C40 require Enhanced link detection for proper link loss reaction times. Due to the hardware restrictions, it cannot be enabled on this board. This is suitable for evaluation purposes, but not for production.

It is probably possible to change the PHY addresses on the board, combine the two management interfaces inside the FPGA and add extra logic for proper configuration of the PHY address bits which are strapped on signals connected to the FPGA. If this can be done, the PHY management interface as well as the Enhanced Link Detection should be enabled.

6.1.2 Functionality

Functionality of the Digital I/O example design:

- Digital input data from the buttons and the joystick is available in the Process Data RAM (0x1000:0x1001).
- Digital output data from Digital Output register (0x0F00:0x0F01) is visualized with IO LEDs.

6.1.3 Implementation

The EtherCAT IP Core MegaFunction needs to be completed before implementing the example design (copy library files to the project folder). Perform the following steps for implementation:

1. Open Altera Quartus II
2. Open example design from *<IPInst_dir>\example_designs\DBC3C40_EtherCAT_DIGI*
3. Open MegaWizard Plug-In Manager, select “Edit and existing custom megafunction variation”
4. Select “ethercat_digitalio.vhd”
5. In the MegaWizard, select Finish. This will complete the EtherCAT IP Core MegaFunction.
6. Start compilation (Menu Processing – Start compilation).
7. Download bitstream into FPGA

6.1.4 SII EEPROM

Use this ESI for the SII EEPROM:

Beckhoff Automation GmbH (Evaluation)/
IP Core example designs ET1810 (Altera)/
ET1810 IP Core 16 Ch. Dig. In-/Output (HW: DBC3C40)

6.1.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

DBC3C40_EtherCAT_DIGI_time_limited.sof

based on this digital I/O example design can be found in the

<IPInst_dir>\example_designs\DBC3C40_EtherCAT_DIGI

folder. After expiration of about 1 hour the design quits its operation unless the JTAG connection to Quartus remains active. This file must only be used for evaluation purposes, any distribution is not allowed.

6.2 EBV DBC4CE55 with NIOS

6.2.1 Configuration and resource consumption

Table 14: Resource consumption NIOS example design DBC4CE55

Configuration		Resources		EP4CE55
FMMU	3	LEs	13,114	23 %
SyncManager	4	Registers	6,642	12 %
RAM	1 KB	Pins	292	90 %
Register set	Medium + EPU/PDI Error Counter + Run LED	M9K	137	53 %
Distributed Clocks	32 bit	Multipliers	0	0 %
PDI	Avalon, 50 MHz	PLLs	1	25 %
NIOS II /e	Debug Level 1			

NOTE: The board uses two individual PHY management interfaces, with both PHYs having the same PHY addresses. Additionally, some of the PHY address bits have to be configured by extra logic inside the FPGA. Because of the identical PHY addresses, the management interfaces on the board cannot be combined to one, and thus, the EtherCAT IP Core cannot make use of the MII management interfaces of the PHY.

The Ethernet PHYs used on the DBC3C40 require Enhanced link detection for proper link loss reaction times. due to the hardware restrictions, it cannot be enabled on this board. This is suitable for evaluation purposes, but not for production.

It is probably possible to change the PHY addresses on the board, combine the two management interfaces inside the FPGA and add extra logic for proper configuration of the PHY address bits which are strapped on signals connected to the FPGA. If this can be done, the PHY management interface as well as the Enhanced Link Detection should be enabled.

6.2.2 Functionality

The NIOS demo application performs the following tasks:

- Accept any EtherCAT Slave State request (copying AL Control to AL Status register)
- Visualize EtherCAT Slave State (7-segment displays and running IO light in Operational mode).

The NIOS demo application is not suitable for production, it cannot be certified. Use the EtherCAT Slave Stack Code (SSC, available from the ETG) for products.

6.2.3 Implementation

The SOPC needs to be generated before implementing the example design. Perform the following steps for implementation:

1. Open Altera Quartus II
2. Open example design from *<IPInst_dir>\example_designs\DBC4CE55_EtherCAT_NIOS*
3. Choose Tools on the menu bar and select Qsys…
4. Open Qsys system "*DBC4CE55_EtherCAT_NIOS_QSYS.qsys*" and view IP configurations
5. Select Generate on the Generation tab to generate system
6. Choose "Tools" on the menu bar and select "NIOS II Software Build Tools for Eclipse"
7. Select workspace, e.g. create *<IPInst_dir>\example_designs\DBC4CE55_EtherCAT_NIOS\workspace*
8. Choose File on the menu bar and select New – "NIOS II Application and BSP from Template"
9. Select SOPC information file "*DBC4CE55_EtherCAT_NIOS_QSYS.sopcinfo*", project template "BECKHOFF EtherCAT" and enter project name the "EtherCAT_Demo"
10. Select Finish
11. Choose Project on the menu bar and select "Build all" to build the software project
    → "EtherCAT_Demo.elf" file is generated in the Debug-Folder of your workspace directory
12. Select "Make Targets – Build…" from the context menu of the "EtherCAT_Demo" project.
13. Select "mem_init_generate" and press "Build" button. This will generate the memory initialization files.
14. Switch over to Quartus II window
15. Select menu "Project – Add/Remove Files in Project…" and add file *"<IPInst_dir>\example_designs\DBC4CE55_EtherCAT_NIOS\EtherCAT_Demo\mem_init\meminit.qip"* to project
16. Start compilation (Menu Processing – Start compilation)
17. Download bitstream into FPGA

6.2.4 SII EEPROM

Use this ESI for the SII EEPROM:

Beckhoff Automation GmbH (Evaluation)/
IP Core example designs ET1810 (Altera)/
ET1810 IP Core NIOSII (HW: DBC4CE55)

6.2.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

DBC4CE55_EtherCAT_NIOS_time_limited.sof

based on this digital I/O example design can be found in the

<IPInst_dir>\example_designs\DBC4CE55

folder. After expiration of about 1 hour the design quits its operation unless the JTAG connection to Quartus remains active. This file must only be used for evaluation purposes, any distribution is not allowed.

6.3 Altera DE2-115 with NIOS

6.3.1 Configuration and resource consumption

Table 15: Resource consumption NIOS example design DE2-115

Configuration		Resources		EP4CE115
FMMU	2	LEs	15,531	14 %
SyncManager	4	Registers	8,283	7 %
RAM	4 KB	Pins	166	31 %
Register set	Full + Run LED + MI Link detection + TX-Shift	M9K	265	61 %
Distributed Clocks	32 bit	Multipliers	0	0 %
PDI	Avalon, 50 MHz	PLLs	1	25 %
NIOS II /e	Debug Level 1			

6.3.2 Functionality

Configure ETHERNET0 and ETHERNET1 for MII mode by setting jumpers JP1 and JP2 to 2-3.

Master is connected to Port ETHERNET0 of DE2-115 (left side, next to VGA). Port ETHERNET1 (right side) can be used to connect other EtherCAT slaves.

The NIOS demo application performs the following tasks:

- Accept any EtherCAT Slave State request (copying AL Control to AL Status register)
- Display EtherCAT IP Core version and slave state on LCD
- RUN LED is LEDG8
- Link/Activity LEDs are LEDG6 and LEDG7
- LEDG0 – LEDG5 are showing a running light if the slave is in OPERATIONAL mode
- Digital input data from the switches SW0-SW17 is available in the Process Data RAM (0x1000:0x1003).
- Digital input data from the push buttons KEY0-KEY3 is available in the Process Data RAM (0x1004).
- Digital output data from Digital Output register (0x1100:0x1103) is visualized with LEDR0-LEDR17
- Digital output data from Digital Output register (0x1104:0x1107) is visualized with the 7-segment LED displays

The NIOS demo application is not suitable for production, it cannot be certified. Use the EtherCAT Slave Stack Code (SSC, available from the ETG) for products.

6.3.3 Implementation

The SOPC needs to be generated before implementing the example design. Perform the following steps for implementation:

1. Open Altera Quartus II
2. Open example design from *<IPInst_dir>\example_designs\DE2_115_NIOS*
3. Choose Tools on the menu bar and select Qsys…
4. Open Qsys system "*DE2_115_EtherCAT_NIOS_QSYS.qsys*" and view IP configurations
5. Select Generate on the Generation tab to generate system
6. Choose "NIOS II" on the menu bar and select "NIOS II Software Build Tools for Eclipse"
7. Select workspace, e.g. create *<IPInst_dir>\example_designs\DE2_115_NIOS\workspace*
8. Choose File on the menu bar and select New – "NIOS II Application and BSP from Template"
9. Select SOPC information file "*DE2_115_EtherCAT_NIOS_QSYS.sopcinfo*", project template "EtherCAT DE2-115" and enter project name "EtherCAT_Demo"
10. Select Finish
11. Choose Project on the menu bar and select "Build all" to build the software project
    → "EtherCAT_Demo.elf" file is generated in the Debug-Folder of your workspace directory
12. Select "Make Targets – Build…" from the context menu of the "EtherCAT_Demo" project.
13. Select "mem_init_ generate" and press "Build" button. This will generate the memory initialization files which will be added to the project later.
14. Switch over to Quartus II window
15. Select menu "Project – Add/Remove Files in Project…" and add file *"<IPInst_dir>\example_designs\DE2_115_NIOS\software\EtherCAT_Demo\mem_init\meminit.qip"* to project
16. Start compilation (Menu Processing – Start compilation)
17. Download bitstream into FPGA

6.3.4 SII EEPROM

Use this ESI for the SII EEPROM:

Beckhoff Automation GmbH (Evaluation)/
IP Core example designs ET1810 (Altera)/
ET1810 IP Core NIOSII (HW: DE2-115)

6.3.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

DE2_115_EtherCAT_NIOS_time_limited.sof

based on this digital I/O example design can be found in the

<IPInst_dir>\example_designs\DE2_115_NIOS

folder. After expiration of about 1 hour the design quits its operation unless the JTAG connection to Quartus remains active. This file must only be used for evaluation purposes, any distribution is not allowed.

7 FPGA Resource Consumption

The resource consumption figures shown in this chapter reflect results of example synthesis runs and can only be used for rough resource estimations. The figures are subject to quite large variations depending on design tools and version, FPGA type, constraints (e.g., area vs. speed), total FPGA utilization (design tools typically stop optimization if the timing goal is reached), etc. No extra effort was undertaken to achieve optimum results, i.e. by sophisticated constraining and design flow setting.

For accurate resource consumption figures, please use the evaluation license of the EtherCAT IP Core and synthesize your individual configuration for the desired FPGA.

The figures of the following table do not imply that the individual features are operational in the selected FPGA (i.e., that the resources are sufficient or that timing closure is achievable). The synthesis runs where performed without timing constraints, without location constraints, and without bitstream generation.

The EtherCAT IP core resource consumption overview figures are based on EtherCAT IP Core for Altera FPGAs Version 2.3.0, Altera Quartus II 9.1, and Altera Cyclone III devices.

Table 16: Typical need of Logic Cells (LE) for main configurable functions

Configurable Function	Approx. LE	Details
Minimum Configuration	3,300	0 x SM, 0 x FMMU, small register preset, no DC, PDI: 32 Bit digital I/O, 1 kByte DPRAM, 1 port MII
Maximum Configuration	25,500	8 x SM, 8 x FMMU, large register preset plus all features except for EEPROM Emulation, DC 64 bit, PDI: SPI, GPIO, 60 kByte DPRAM, 3 ports MII
Additional port	1,000	all port features enabled (without DC Receive time)
SyncManager	550	per SyncManager
FMMU	650	per FMMU
Distributed Clocks	200	Receive time per port
	3,600	32 bit
	6,200	64 bit
	350	SyncManager Event Times
Register preset		
small	-	reference
medium	400	according to small register preset
large	900	according to small register preset
PHY features	900	All MII features: Management Interface, MI link detection and configuration, TX Shift, and enhanced link detection (3 ports)
DPRAM	200	60 KB (M4K/M9K)
PDI		
32 Bit Digital I/O	250	
SPI	350	
8 Bit µController	150	
16 Bit µController	250	
Avalon	200	50 MHz, 32 Bit
GPIO	400	8 Byte

The EtherCAT IP core resource consumption figures for typical EtherCAT devices are based on EtherCAT IP Core for Altera FPGAs Version 2.4.0, Altera Quartus II 10.1, and Altera Cyclone III devices.

Table 17: EtherCAT IP Core configuration for typical EtherCAT Devices

EtherCAT Device	SM	FMMU	DPRAM [kByte]	PDI	DC	Register preset	Logic Elements
IO	2	2	1	32 Bit Digital I/O	-	Small	7,300
Frequency Inverter	4	3	1	SPI	-	Large	10,200
Encoder	4	3	1	SPI	32	Large	14,500
Fieldbus Gateway	4	3	4	16 Bit µC	-	Large	10,100
Servo Drive	4	3	4	16 Bit µC	32	Large	14,400

NOTE: Register preset medium and large including MII Management Interface. All devices have 2 MII ports, DC is 32 bit wide.

8 IP Core Signals

The available signals depend on the IP Core configuration.

8.1 General Signals

Table 18: General Signals

Condition	Name	Direction	Description
	nRESET	INPUT	Resets all registers of the IP Core, active low
Reset slave by ECAT/PDI	RESET_OUT	OUTPUT	Reset by ECAT (reset register 0x0040), active high. RESET_OUT has to trigger nRESET, which clears RESET_OUT.
	CLK25	INPUT	25 MHz clock signal from PLL (rising edge synchronous with rising edge of CLK100)
	CLK100	INPUT	100 MHz clock signal from PLL

8.1.1 Clock source example schematics

The EtherCAT IP Core and the Ethernet PHYs have to share the same clock source. The initial accuracy of the EtherCAT IP clock source has to be 25ppm or better.

Typically, the clock inputs of the EtherCAT IP Core (CLK25, CLK100, and optionally CLK50) are sourced by a PLL inside the FPGA. The PLL has to use a configuration which guarantees a fixed phase relation between clock input and clock outputs, in order to enable TX shift compensation for the MII TX signals.

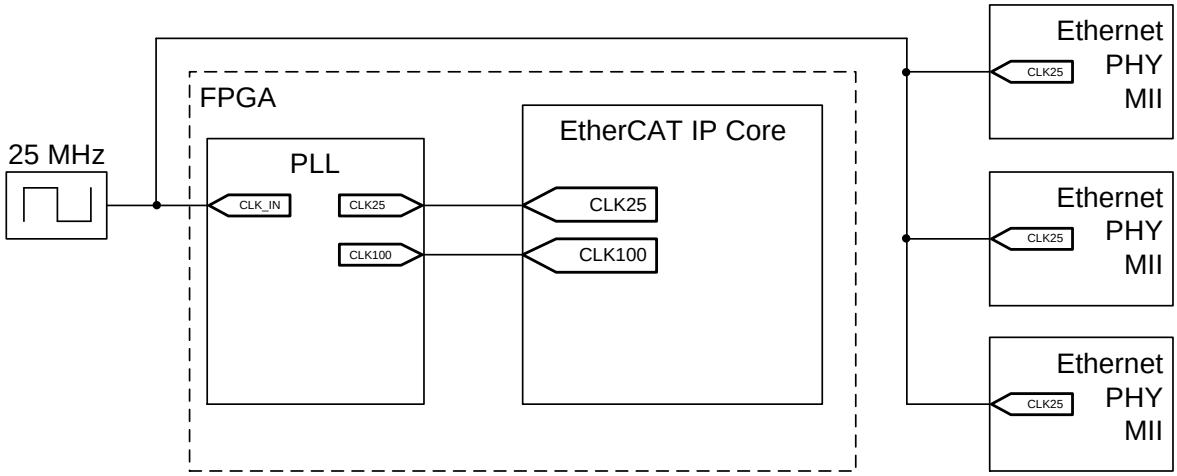

Figure 19: EtherCAT IP Core clock source (MII)

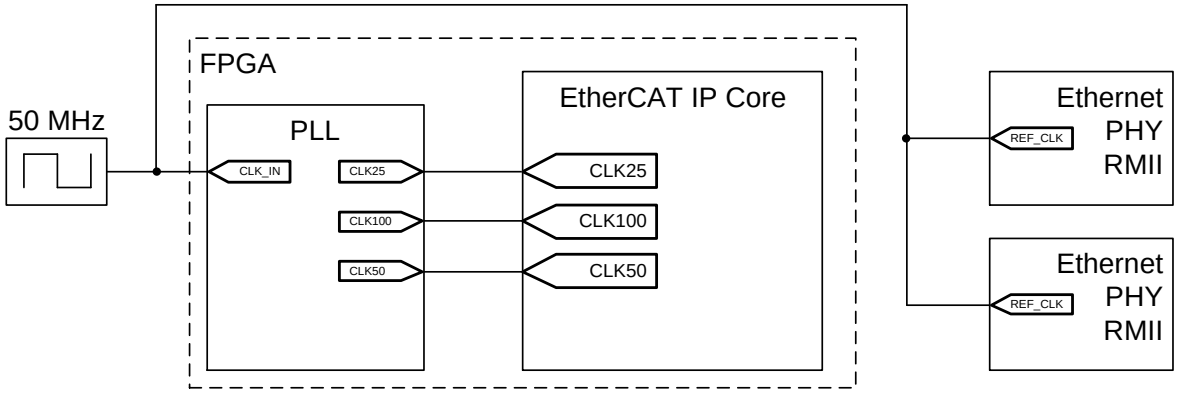

Figure 20: EtherCAT IP Core clock source (RMII)

8.2 SII EEPROM Interface Signals

Table 19: SII EEPROM Signals

Condition	Name	Direction	Description
	PROM_SIZE	INPUT	Sets EEPROM size 0: up to 16 kbit EEPROM 1: 32 kbit-4Mbit EEPROM
Tristate drivers inside core (EEPROM/MI)	PROM_CLK	OUTPUT	EEPROM I²C Clock (output values: 0 or Z)
External tristate drivers for EEPROM/MI	PROM_CLK	OUTPUT	EEPROM I²C Clock (output values: 0 or 1)
Tristate drivers inside core (EEPROM/MI)	PROM_DATA	BIDIR	EEPROM I²C Data
External tristate drivers for EEPROM/MI	PROM_DATA_IN	INPUT	EEPROM I²C Data: EEPROM → IP Core
	PROM_DATA_OUT	OUTPUT	EEPROM I²C Data : IP Core → EEPROM (always 0)
	PROM_DATA_ENA	OUTPUT	0: disable output driver for PROM_DATA_OUT 1: enable output driver for PROM_DATA_OUT

8.3 LED Signals

Table 20 lists the signals used for the LEDs. The LED signals are active high. All LEDs should be green.

Table 20: LED Signals

Condition	Name	Direction	Description
	LINK_ACT[0]	OUTPUT	Link/activity LED for ethernet port 0
2 or 3 communication ports	LINK_ACT[1]	OUTPUT	Link/activity LED for ethernet port 1
3 communication ports	LINK_ACT[2]	OUTPUT	Link/activity LED for Ethernet port 2
RUN_LED enabled	LED_RUN	OUTPUT	RUN LED for device status Always 0 if RUN LED is deactivated.
RUN_LED enabled and Extended RUN/ERR LED enabled	LED_ERR	OUTPUT	ERR LED for device status.
	LED_STATE_RUN	OUTPUT	Connect to RUN pin of dual-color STATE LED, connect LED_ERR to ERR pin of STATE LED

NOTE: The application ERR LED and STATE LED can alternatively be controlled by a µController if required.

8.4 Distributed Clocks SYNC/LATCH Signals

Table 21 lists the signals used with Distributed Clocks.

Table 21: DC SYNC/LATCH signals

Condition	Name	Direction	Description
Distributed Clocks enabled	SYNC_OUT0	OUTPUT	DC sync output 0
	SYNC_OUT1	OUTPUT	DC sync output 1
	LATCH_IN0	INPUT	DC latch input 0
	LATCH_IN1	INPUT	DC latch input 1

NOTE: SYNC_OUT0/1 are active high/push-pull outputs.

8.5 Physical Layer Interface

The IP Core is connected with Ethernet PHYs using MII or RMII interfaces.

Table 22 lists the general PHY interface signals.

Table 22: Physical Layer General

Condition	Name	Direction	Description
PHY Management Interface enabled	PHY_OFFSET_VEC[4:0]	INPUT	PHY address offset
PHY Management Interface enabled	MCLK	OUTPUT	PHY management clock
PHY Management Interface enabled, Tristate drivers inside core (EEPROM/MII)	MDIO	BIDIR	PHY management data
PHY Management Interface enabled, External tristate drivers for EEPROM/MI	MDIO_DATA_IN	INPUT	PHY management data: PHY → IP Core
	MDIO_DATA_OUT	OUTPUT	PHY management data: IP Core → PHY
	MDIO_DATA_ENA	OUTPUT	0: disable output driver for MDIO_DATA_OUT 1: enable output driver for MDIO_DATA_OUT

NOTE: MDIO must have a pull-up resistor (4.7kΩ recommended for ESCs).

8.5.1 MII Interface

Table 23 lists the signals used with MII. The TX_CLK signals of the PHYs are not connected to the IP Core unless TX Shift automatic configuration is enabled.

Table 23: PHY Interface MII

Condition	Name	Direction	Description
Port0 = MII	nMII_LINK0	INPUT	0: 100 Mbit/s (Full Duplex) link at port 0 1: no link at port 0
	MII_RX_CLK0	INPUT	Receive clock port 0
	MII_RX_DV0	INPUT	Receive data valid port 0
	MII_RX_DATA0[3:0]	INPUT	Receive data port 0
	MII_RX_ERR0	INPUT	Receive error port 0
	MII_TX_ENA0	OUTPUT	Transmit enable port 0
	MII_TX_DATA0[3:0]	OUTPUT	Transmit data port 0
Port0 = MII and TX Shift activated	MII_TX_CLK0	INPUT	Transmit clock port 0 for automatic TX Shift configuration. Set to 0 for manual TX Shift configuration.
	MII_TX_SHIFT0[1:0]	INPUT	Manual TX shift configuration port 0. Additional TX signal delay: 00: 0 ns 01: 10 ns 10: 20 ns 11: 30 ns
Port1 = MII	nMII_LINK1	INPUT	0: 100 Mbit/s (Full Duplex) link at port 1 1: no link at port 1
	MII_RX_CLK1	INPUT	Receive clock port 1
	MII_RX_DV1	INPUT	Receive data valid port 1
	MII_RX_DATA1[3:0]	INPUT	Receive data port 1
	MII_RX_ERR1	INPUT	Receive error port 1
	MII_TX_ENA1	OUTPUT	Transmit enable port 1
	MII_TX_DATA1[3:0]	OUTPUT	Transmit data port 1
Port1 = MII and TX Shift activated	MII_TX_CLK1	INPUT	Transmit clock port 1 for automatic TX Shift configuration. Set to 0 for manual TX Shift configuration.
	MII_TX_SHIFT1[1:0]	INPUT	Manual TX shift configuration port 1. Additional TX signal delay: 00: 0 ns 01: 10 ns 10: 20 ns 11: 30 ns

Condition	Name	Direction	Description
Port2 = MII	nMII_LINK2	INPUT	0: 100 Mbit/s (Full Duplex) link at port 2 1: no link at port 2
	MII_RX_CLK2	INPUT	Receive clock port 2
	MII_RX_DV2	INPUT	Receive data valid port 2
	MII_RX_DATA2[3:0]	INPUT	Receive data port 2
	MII_RX_ERR2	INPUT	Receive error port 2
	MII_TX_ENA2	OUTPUT	Transmit enable port 2
	MII_TX_DATA2[3:0]	OUTPUT	Transmit data port 2
Port2 = MII and TX Shift activated	MII_TX_CLK2	INPUT	Transmit clock port 2 for automatic TX Shift configuration. Set to 0 for manual TX Shift configuration.
	MII_TX_SHIFT2[1:0]	INPUT	Manual TX shift configuration port 2. Additional TX signal delay: 00: 0 ns 01: 10 ns 10: 20 ns 11: 30 ns

8.5.2 RMII Interface

Table 24 lists the signals used with RMII.

Table 24: PHY Interface RMII

Condition	Name	Direction	Description
Port0 = RMII	CLK50	INPUT	50 MHz reference clock signal from PLL (rising edge synchronous with rising edge of CLK100), also connected to PHY
	nRMII_LINK0	INPUT	0: 100 Mbit/s (Full Duplex) link at port 0 1: no link at port 0
	RMII_RX_DV0	INPUT	Carrier sense/receive data valid port 0
	RMII_RX_DATA0[1:0]	INPUT	Receive data port 0
	RMII_RX_ERR0	INPUT	Receive error port 0
	RMII_TX_ENA0	OUTPUT	Transmit enable port 0
	RMII_TX_DATA0[1:0]	OUTPUT	Transmit data port 0
Port1 = RMII	nRMII_LINK1	INPUT	0: 100 Mbit/s (Full Duplex) link at port 1 1: no link at port 1
	RMII_RX_DV1	INPUT	Carrier sense/receive data valid port 1
	RMII_RX_DATA1[1:0]	INPUT	Receive data port 1
	RMII_RX_ERR1	INPUT	Receive error port 1
	RMII_TX_ENA1	OUTPUT	Transmit enable port 1
	RMII_TX_DATA1[1:0]	OUTPUT	Transmit data port 1

8.6 PDI Signals

8.6.1 General PDI Signals

Table 26 lists the signals available independent of the PDI configuration.

Table 25: General PDI Signals

Condition	Name	Direction	Description
	PDI_SOF	OUTPUT	Ethernet Start-of-Frame if 1
	PDI_EOF	OUTPUT	Ethernet End-of-Frame if 1
	PDI_WD_TRIGGER	OUTPUT	Process Data Watchdog trigger if 1
	PDI_WD_STATE	OUTPUT	Process Data Watchdog state 0: Expired 1: Not expired
GPIO Bytes > 0	PDI_GPI[8*Bytes-1:0]	INPUT	General purpose inputs (width configurable, 1/2/4/8 Bytes)
GPIO Bytes > 0	PDI_GPO[8*Bytes-1:0]	OUTPUT	General purpose outputs (width N:0 configurable, 1/2/4/8 Bytes)

8.6.2 Digital I/O Interface

Table 26 lists the signals used with the Digital I/O PDI.

Table 26: Digital I/O PDI

Condition	Name	Direction	Description
Byte 0 is Output	PDI_DIGI_DATA_OUT0 [7:0]	OUTPUT	Digital output byte 0
Byte 0 is Input	PDI_DIGI_DATA_IN0 [7:0]	INPUT	Digital input byte 0
Byte 1 is Output	PDI_DIGI_DATA_OUT1[7:0]	OUTPUT	Digital output byte 1
Byte 1 is Input	PDI_DIGI_DATA_IN1[7:0]	INPUT	Digital input byte 1
Byte 2 is Output	PDI_DIGI_DATA_OUT2[7:0]	OUTPUT	Digital output byte 2
Byte 2 is Input	PDI_DIGI_DATA_IN2[7:0]	INPUT	Digital input byte 2
Byte 3 is Output	PDI_DIGI_DATA_OUT3 [7:0]	OUTPUT	Digital output byte 3
Byte 3 is Input	PDI_DIGI_DATA_IN3[7:0]	INPUT	Digital input byte 3
If both, digital input and output selected	PDI_DIGI_DATA_ENA	OUTPUT	Digital output enable
any digital input selected and Input mode=Latch with ext. signal	PDI_DIGI_LATCH_IN	INPUT	Latch digital input at rising edge
any digital output selected	PDI_DIGI_OE_EXT	INPUT	External output enable
	PDI_DIGI_OUTVALID	OUTPUT	Output event: output valid

8.6.3 SPI Slave Interface

Table 27 used with an SPI PDI.

Table 27: SPI PDI

Condition	Name	Direction	Description
SPI PDI	PDI_EMULATION	INPUT	Value for register 0x0140.8: 0: device status register is controlled by µC 1: device status register is identical to device control register
	PDI_SPI_CLK	INPUT	SPI clock
	PDI_SPI_SEL	INPUT	SPI slave select
	PDI_SPI_DI	INPUT	SPI slave data in (MOSI)
	PDI_SPI_IRQ	OUTPUT	SPI interrupt
Tristate drivers inside core (SPI configuration)	PDI_SPI_DO	OUTPUT	SPI slave data out (MISO)
External tristate drivers	PDI_SPI_DO_OUT	OUTPUT	SPI slave data out: IP Core → µC
	PDI_SPI_DO_ENA	OUTPUT	0: disable output driver for PDI_SPI_DO_OUT 1: enable output driver for PDI_SPI_DO_OUT

8.6.4 Asynchronous 8/16 Bit µController Interface

Table 28 lists the signals used with both, 8 Bit and 16 Bit asynchronous µController PDI.

Table 28: 8/16 Bit µC PDI

Condition	Name	Direction	Description
8/16 Bit µC	PDI_EMULATION	INPUT	Value for register 0x0140.8: 0: device status register is controlled by µC 1: device status register is identical to device control register
	PDI_uC_ADR[15:0]	INPUT	µC address bus
	PDI_uC_nBHE	INPUT	µC byte high enable
	PDI_uC_nRD	INPUT	µC read access
	PDI_uC_nWR	INPUT	µC write access
	PDI_uC_nCS	INPUT	µC chip select
	PDI_uC_IRQ	OUTPUT	Interrupt
	PDI_uC_BUSY	OUTPUT	PDI busy
	PDI_uC_DATA_ENA	OUTPUT	0: disable output driver for PDI_uC_DATA_OUT 1: enable output driver for PDI_uC_DATA_OUT

8.6.4.1 8 Bit µController Interface

Table 29 lists the signals used with an 8 Bit µC PDI.

Table 29: 8 Bit µC PDI

Condition	Name	Direction	Description
Tristate drivers inside core (µController configuration)	PDI_uC_DATA[7:0]	BIDIR	µC data bus
External tristate drivers	PDI_uC_DATA_IN[7:0]	INPUT	µC data bus: µC → IP Core
	PDI_uC_DATA_OUT[7:0]	OUTPUT	µC data bus: IP Core → µC

8.6.4.2 16 Bit µController Interface

Table 30 lists the signals used with a 16 Bit µC PDI.

Table 30: 16 Bit µC PDI

Condition	Name	Direction	Description
Tristate drivers inside core (µController configuration)	PDI_uC_DATA[15:0]	BIDIR	µC data bus
External tristate drivers	PDI_uC_DATA_IN[15:0]	INPUT	µC data bus: µC → IP Core
	PDI_uC_DATA_OUT[15:0]	OUTPUT	µC data bus: IP Core → µC

8.6.5 Avalon On-Chip Bus

Table 31 lists the signals used with the Avalon PDI.

Table 31: Avalon PDI

Condition	Name	Direction	Description
Avalon PDI	PDI_EMULATION	INPUT	Value for register 0x0140.8: 0: device status register is controlled by µC 1: device status register is identical to device control register
	PDI_AVALON_CLK	INPUT	N*25 MHz Avalon bus clock from PLL (rising edge of CLK25 synchronous with rising edge of PDI_AVALON_CLK)
	PDI_AVALON_ADR[15:0]	INPUT	Avalon address
	PDI_AVALON_RD_DATA[7:0]	OUTPUT	Avalon slave read data
	PDI_AVALON_WR_DATA[7:0]	INPUT	Avalon write data
	PDI_AVALON_READ	INPUT	Avalon read access
	PDI_AVALON_WRITE	INPUT	Avalon write access
	PDI_AVALON_CS	INPUT	Avalon chip select
	PDI_AVALON_IRQ	OUTPUT	Avalon slave interrupt
	PDI_AVALON_BUSY	OUTPUT	Avalon slave busy
	PDI_AVALON_SYNC0	OUTPUT	DC SYNC0 output. Always 0 if DC is disabled.
	PDI_AVALON_SYNC1	OUTPUT	DC SYNC1 output. Always 0 if DC is disabled.

NOTE: If the EtherCAT IP Core is used inside Qsys or SoPC Builder, PDI_AVALON_SYNC0 and PDI_AVALON_SYNC1 are declared as interrupt signals for the processor (avalon_ethercat_sync0/1). Use SYNC_OUT0/1 signals for external use of the SyncSignals.

9 Ethernet Interface

The IP Core is connected with Ethernet PHYs using MII or RMII interfaces. MII is recommended since the PHY delay (and delay jitter) is smaller in comparison to RMII.

9.1 PHY Management interface

9.1.1 PHY Management Interface Signals

The PHY management interface of the IP Core has the following signals:

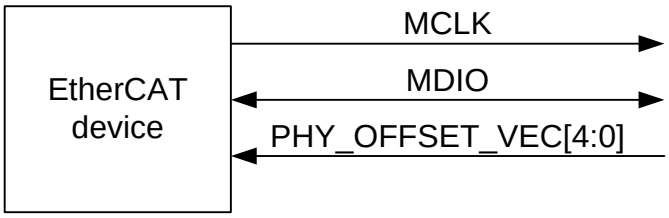

Figure 21: PHY management Interface signals

Table 32: PHY management Interface signals

Signal	Direction	Description
MCLK	OUT	Management Interface clock (alias MCLK)
MDIO	BIDIR	Management Interface data (alias MDIO)
PHY_OFFSET_VEC[4:0]	INPUT	PHY address offset (consecutive PHY addresses, address of port 0)

MDIO must have a pull-up resistor (4.7 kΩ recommended for ESCs), either integrated into the ESC or externally. MCLK is driven rail-to-rail, idle value is High.

9.1.2 PHY Address Configuration

The EtherCAT IP Core addresses Ethernet PHYs typically using logical port number plus PHY address offset. Ideally, the Ethernet PHY addresses should correspond with the logical port number, so PHY addresses 0-2 are used.

A PHY address offset of 0-31 can be applied which moves the PHY addresses to any consecutive address range. The IP Core expects logical port 0 to have PHY address 0 plus PHY address offset (and so on).

9.1.3 Separate external MII management interfaces

If two separate external MII management interfaces are to be connected to the single MII management interface of the EtherCAT IP Core, some glue logic has to be added. Disable internal Tri-State drivers for the MII management bus and combine the signals according to the following figure. Take care of proper PHY address configuration: the PHYs need different PHY addresses.

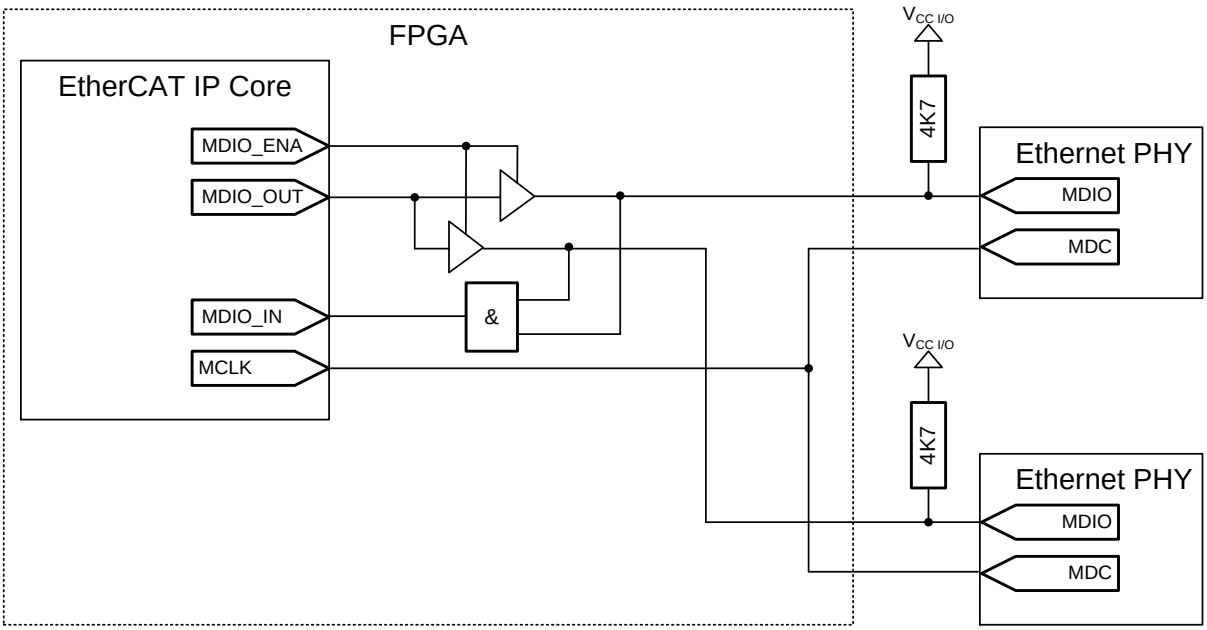

Figure 22: Example schematic with two individual MII management interfaces

9.1.4 MII management timing specifications

For MII Management Interface timing diagrams refer to Section I.

Table 33: MII management timing characteristics

Parameter	Min	Typ	Max	Comment
$t_{MI\_startup}$		1.34 ms		Time between reset end and the first access of via management interface
t_{Clk}		400 ns		MI_CLK period
t_{Write}		~ 25.6 µs		MI Write access time
t_{Read}		~ 25.4 µs		MI Read access time

9.2 MII Interface

The MII interface of the IP Core is optimized for low processing/forwarding delays by omitting a transmit FIFO. To allow this, the IP Core has additional requirements to Ethernet PHYs, which are easily accomplished by several PHY vendors.

Refer to "Section I – Technology" for Ethernet PHY requirements.

Additional information regarding the IP Core:

- The clock source of the PHYs is the same as for the FPGA (25 MHz quartz oscillator)
- The signal polarity of nMII_LINK is not configurable inside the IP Core, nMII_LINK is active low. If necessary, the signal polarity must be swapped by user logic outside the IP Core.
- The IP Core can be configured to use the MII management interface for link detection and link configuration.
- The IP Core supports an arbitrary PHY address offset.

For details about the ESC MII Interface refer to Section I.

9.2.1 MII Interface Signals

The MII interface of the IP Core has the following signals:

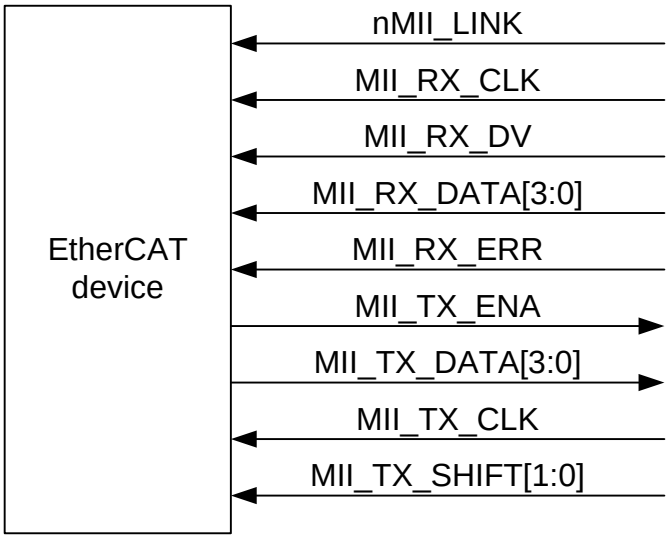

Figure 23: MII Interface signals

Table 34: MII Interface signals

Signal	Direction	Description
nMII_LINK	IN	Input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established (alias LINK_MII)
MII_RX_CLK	IN	Receive Clock
MII_RX_DV	IN	Receive data valid
MII_RX_DATA[3:0]	IN	Receive data (alias RXD)
MII_RX_ERR	IN	Receive error (alias RX_ER)
MII_TX_ENA	OUT	Transmit enable (alias TX_EN)
MII_TX_DATA[3:0]	OUT	Transmit data (alias TXD)
MII_TX_CLK	IN	Transmit Clock for automatic TX Shift compensation
MII_TX_SHIFT[1:0]	IN	Manual TX Shift compensation with additional registers

9.2.2 TX Shift Compensation

Since IP Core and the Ethernet PHYs share the same clock source, TX_CLK from the PHY has a fixed phase relation to MII_TX_ENA/MII_TX_DATA from the IP Core. Thus, TX_CLK is not connected and the delay of a TX FIFO inside the IP Core is saved.

In order to fulfill the setup/hold requirements of the PHY, the phase shift between TX_CLK and MII_TX_ENA/MII_TX_DATA has to be controlled. There are several alternatives:

- TX Shift Compensation by specifying/verifying minimum and maximum clock-to-output times for MII_TX_ENA/MII_TX_DATA with respect to CLK_IN (PHY and PLL clock source).
- TX Shift compensation with additional delays for MII_TX_ENA/MII_TX_DATA of 10, 20, or 30 ns. Such delays can be added using the TX Shift feature and applying MII_TX_SHIFT[1:0]. MII_TX_SHIFT[1:0] determine the delay in multiples of 10 ns for each port. For guaranteed timings, maximum clock-to-output times for MII_TX_ENA/MII_TX_DATA should be applied, too. Set MII_TX_CLK to 0 if manual TX Shift compensation is used.
- Automatic TX Shift compensation if the TX Shift feature is selected: connect MII_TX_CLK and the automatic TX Shift compensation will determine correct shift settings. For guaranteed timings, maximum clock-to-output times for MII_TX_ENA/MII_TX_DATA should be applied, too. Set manual TX Shift compensation to 0 in this case.

MII_TX_ENA and MII_TX_DATA are generated synchronous to CLK25, although the source registers are both CLK25 and CLK100 registers.

The PLL has to use a configuration which guarantees a fixed phase relation between clock input and CLK25/CLK100 output, in order to enable TX shift compensation for the MII TX signals.

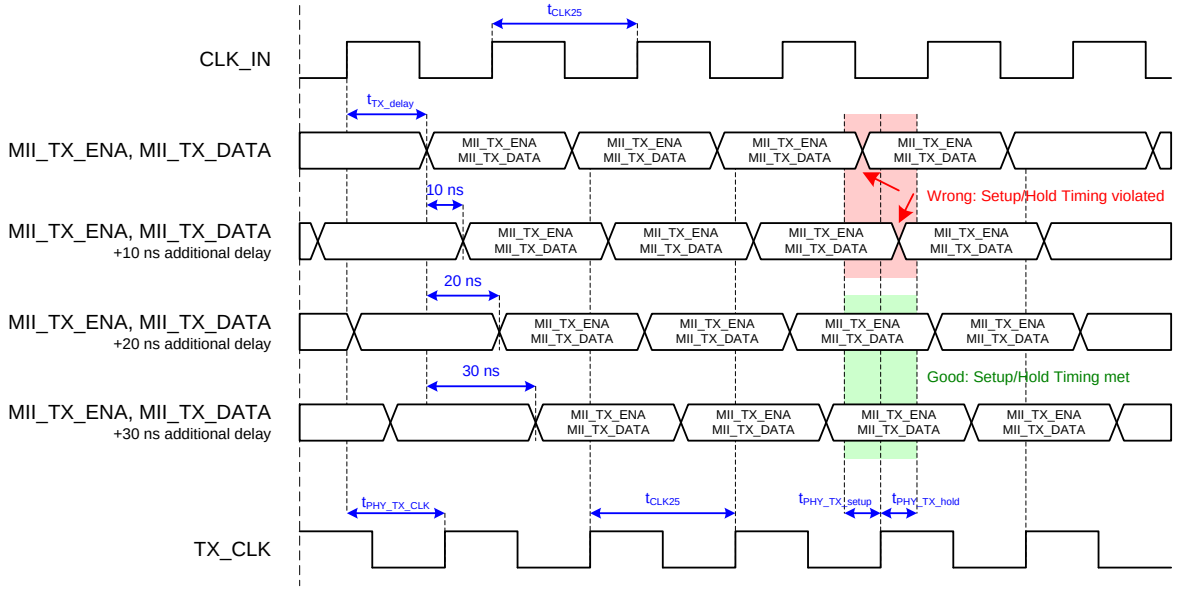

Figure 24: MII TX Timing Diagram

Table 35: MII TX Timing characteristics

Parameter	Comment
t_{CLK25}	25 MHz quartz oscillator (CLK_IN)
t_{TX_delay}	MII_TX_ENA/MII_TX_DATA[3:0] delay after rising edge of CLK_IN, depends on synthesis results
$t_{PHY\_TX\_CLK}$	Delay between PHY clock source and TX_CLK output of the PHY, PHY dependent
$t_{PHY\_TX\_setup}$	PHY setup requirement: TX_ENA/TX_DATA with respect to TX_CLK (PHY dependent, IEEE802.3 limit is 15 ns)
$t_{PHY\_TX\_hold}$	PHY hold requirement: TX_ENA/TX_DATA with respect to TX_CLK (PHY dependent, IEEE802.3 limit is 0 ns)

If the phase shift between CLK25 and TX_CLK should not be constant for a some special PHYs, additional FIFOs for MII_TX_ENA/MII_TX_DATA are necessary. The FIFO input uses CLK25, the FIFO output TX_CLK[0] or TX_CLK[1] respectively.

NOTE: The phase shift can be adjusted by displaying TX_CLK of a PHY and MII_TX_ENA/MII_TX_DATA[3:0] on an oscilloscope. MII_TX_ENA/MII_TX_DATA[3:0] is allowed to change between 0 ns and 25 ns after a rising edge of TX_CLK (according to IEEE802.3 – check your PHY's documentation). Setup phase shift so that MII_TX_ENA/MII_TX_DATA[3:0] change near the middle of this range. MII_TX_ENA/MII_TX_DATA[3:0] signals are generated at the same time.

9.2.3 MII Timing specifications

Table 36: MII timing characteristics

Parameter	Min	Typ	Max	Comment
t_{RX_CLK}	40 ns ± 100 ppm			RX_CLK period (100 ppm with maximum FIFO Size only)
t_{RX_setup}	x[1]			RX_DV/RX_DATA/RX_D[3:0] valid before rising edge of RX_CLK
t_{RX_hold}	x[1]			RX_DV/RX_DATA/RX_D[3:0] valid after rising edge of RX_CLK

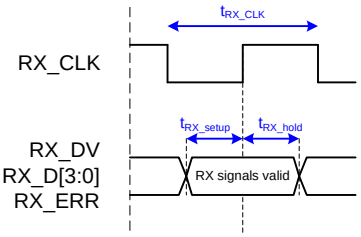

Figure 25: MII timing RX signals

[1] EtherCAT IP Core: time depends on synthesis results

9.2.4 MII example schematic

Refer to chapter 8.5 for more information on special markings (!). Take care of proper compensation of the TX_CLK phase shift.

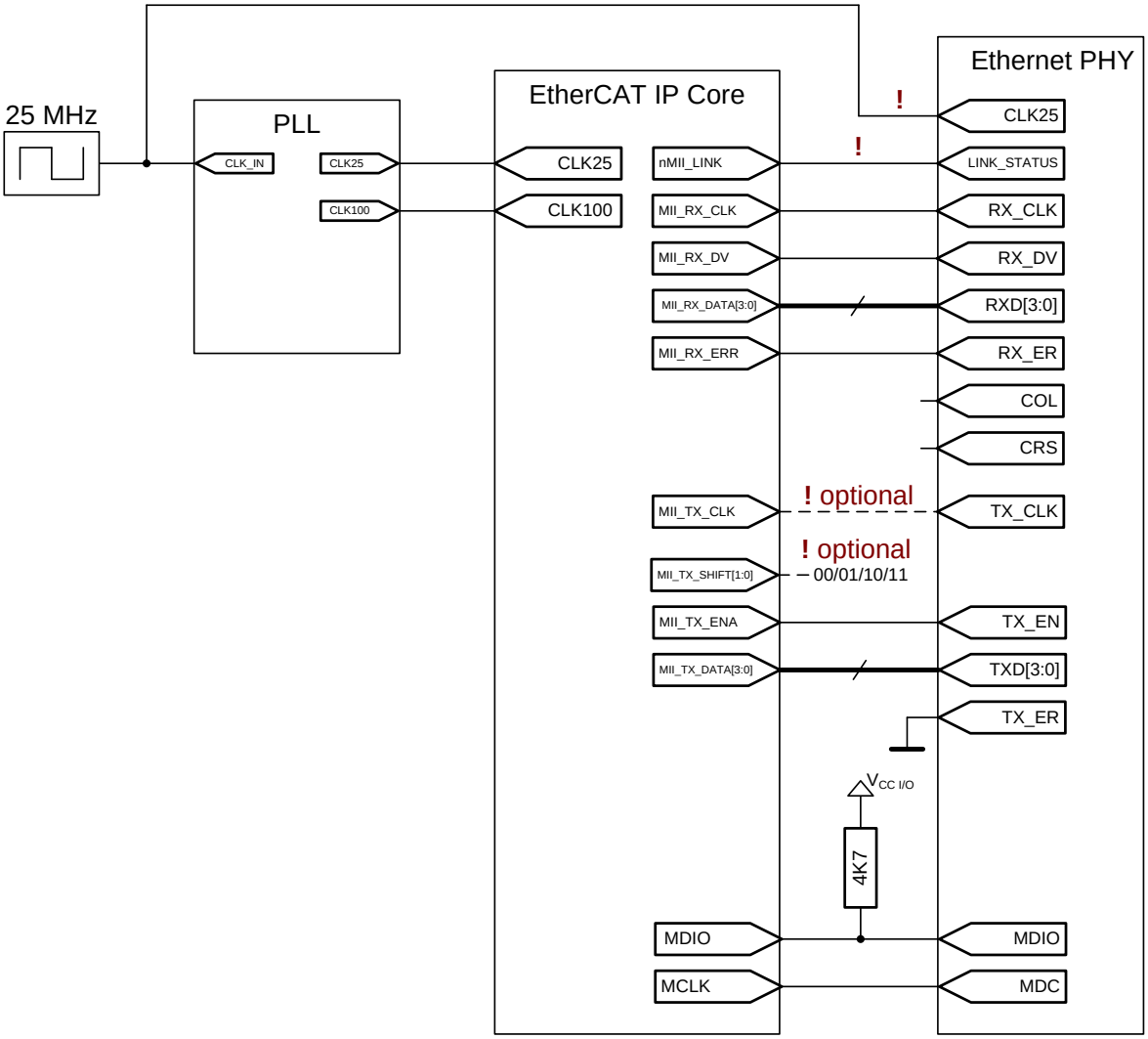

Figure 26: MII example schematic

9.3 RMII Interface

The IP Core supports RMII with 2 communication ports. Nevertheless, MII is recommended since the PHY delay (and delay jitter) is smaller in comparison to RMII.

The Beckhoff ESCs have additional requirements to Ethernet PHYs using RMII, which are easily accomplished by several PHY vendors.

Refer to "Section I – Technology" for Ethernet PHY requirements.

Additional information regarding the IP Core:

- The clock source of the PHYs is the same as for the FPGA (25 MHz quartz oscillator)
- The signal polarity of nRMII_LINK is not configurable inside the IP Core, nRMII_LINK is active low. If necessary, the signal polarity must be swapped outside the IP Core.
- The IP Core can be configured to use the MII management interface for link detection and link configuration.
- The IP Core supports an arbitrary PHY address offset.

For details about the ESC RMII Interface refer to Section I.

9.3.1 RMII Interface Signals

The RMII interface of the IP Core has the following signals:

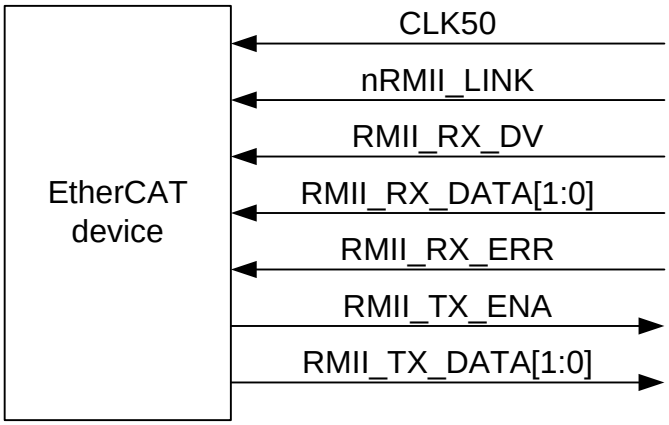

Figure 27: RMII Interface signals

Table 37: RMII Interface signals

Signal	Direction	Description
CLK50	IN	RMII RX/TX reference clock (50 MHz)
nRMII_LINK	IN	Input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established (alias LINK_MII)
RMII_RX_DV	IN	Carrier sense/receive data valid
RMII_RX_DATA[1:0]	IN	Receive data (alias RXD)
RMII_RX_ERR	IN	Receive error (alias RX_ER)
RMII_TX_ENA	OUT	Transmit enable (alias TX_EN)
RMII_TX_DATA[1:0]	OUT	Transmit data (alias TXD)

9.3.2 RMII example schematic

Refer to chapter 8.5 for more information on special markings (!). Take care of proper PHY address configuration.

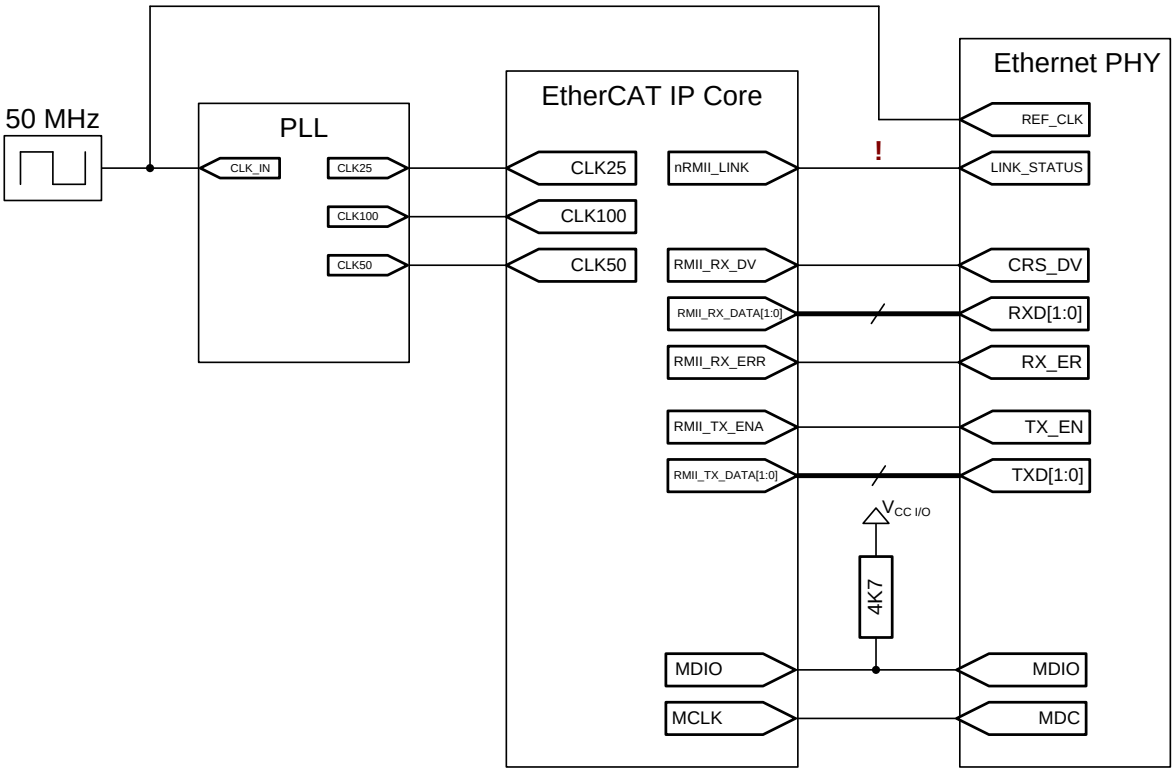

Figure 28: RMII example schematic

10 PDI Description

Table 38: Available PDIs for EtherCAT IP Core

PDI number 0x0140 [7:0]	On-chip bus		PDI name	IP Core
	0x0150 [7:5]	0x0152 [10:8]		
0x00	-	-	Interface deactivated	x
0x01	-	-	4 Digital Input	
0x02	-	-	4 Digital Output	
0x03	-	-	2 Digital Input and 2 Digital Output	
0x04	-	-	Digital I/O	x
0x05	-	-	SPI Slave	x
0x06	-	-	Oversampling I/O	
0x07	-	-	EtherCAT Bridge (port 3)	
0x08	-	-	16 Bit asynchronous Microcontroller interface	x
0x09	-	-	8 Bit asynchronous Microcontroller interface	x
0x0A	-	-	16 Bit synchronous Microcontroller interface	
0x0B	-	-	8 Bit synchronous Microcontroller interface	
0x10	-	-	32 Digital Input/0 Digital Output	
0x11	-	-	24 Digital Input/8 Digital Output	
0x12	-	-	16 Digital Input/16 Digital Output	
0x13	-	-	8 Digital Input/24 Digital Output	
0x14	-	-	0 Digital Input/32 Digital Output	
0x80	000	-	On-chip bus (Avalon)	x
	001	000	On-chip bus (AXI3)	
		001	On-chip bus (AXI4)	
		010	On-chip bus (AXI4LITE)	
	010	-	On-chip bus (PLB v4.6)	
	100	-	On-chip bus (OPB)	
Others			Reserved	

10.1 Digital I/O Interface

10.1.1 Interface

The Digital I/O PDI is selected with PDI type 0x04. The signals of the Digital I/O interface are[2]:

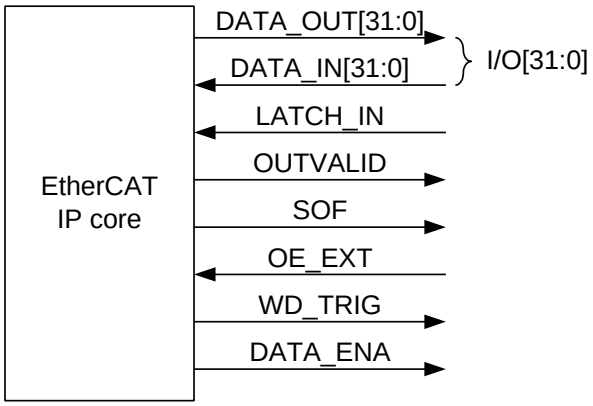

Figure 29: IP core digital I/O signals

Table 39: IP core digital I/O signals

Signal	Direction	Description	Signal polarity
DATA_OUT[31:0]	OUT	Output data	
DATA_IN[31:0]	IN	Input data	
LATCH_IN	IN	External data latch signal	act. high
OUTVALID	OUT	Output data is valid/Output event	act. high
SOF	OUT	Start of Frame	act. high
OE_EXT	IN	Output Enable	act. high
WD_TRIG	OUT	Watchdog Trigger	act. high
DATA_ENA	OUT	Enable external Output data driver	act. high

NOTE: Unsupported Digital I/O control signal OE_CONF is assumed to be low.

The Digital I/O PDI supports 1-4 byte of digital I/O signals, with each byte individually configurable as either input or output. At the IP core interface, the I/O signals are separated in input signals (DATA_IN) and output signals (DATA_OUT). The corresponding I/O bytes and addresses are listed below.

Table 40: Input/Output byte reference

I/O Byte	I/O signal	Output signal	Output address	Input signal	Input address
0	I/O[7:0]	DATA_OUT[7:0]	0x0F00	DATA_IN[7:0]	0x1000
1	I/O[15:8]	DATA_OUT[15:8]	0x0F01	DATA_IN[15:8]	0x1001
2	I/O[23:16]	DATA_OUT[23:16]	0x0F02	DATA_IN[23:16]	0x1002
3	I/O[31:24]	DATA_OUT[31:24]	0x0F03	DATA_IN[31:24]	0x1003

[2] The prefix `PDI_DIGI_` is added to the Digital I/O interface signals if the EtherCAT IP Core is used.

10.1.2 Configuration

The Digital I/O interface is selected with PDI type 0x04 in the PDI control register 0x0140. It supports different configurations, which are located in registers 0x0150 – 0x0153.

10.1.3 Digital Inputs

Digital input values appear in the process memory at address 0x1000:0x1003. EtherCAT devices use Little Endian byte ordering, so I/O[7:0] can be read at 0x1000 etc. Digital inputs are written to the process memory by the Digital I/O PDI using standard PDI write operations.

Digital inputs can be configured to be sampled by the ESC in four ways:

- Digital inputs are sampled at the start of each Ethernet frame, so that EtherCAT read commands to address 0x1000:0x1003 will present digital input values sampled at the start of the same frame. The SOF signal can be used externally to update the input data, because the SOF is signaled before input data is sampled.
- The sample time can be controlled externally by using the LATCH_IN signal. The input data is sampled by the ESC each time a rising edge of LATCH_IN is recognized.
- Digital inputs are sampled at Distributed Clocks SYNC0 events.
- Digital inputs are sampled at Distributed Clocks SYNC1 events.

For Distributed Clock SYNC input, SYNC generation must be activated (register 0x0981). SYNC output is not necessary (register 0x0151). SYNC pulse length (registers 0x0982:0x0983) should not be set to 0, because acknowledging of SYNC events is not possible with Digital I/O PDI. Sample time is the beginning of the SYNC event.

10.1.4 Digital Outputs

Digital Output values have to be written to register 0x0F00:0x0F03 (register 0x0F00 controls I/O[7:0] etc.). Digital Output values are not read by the Digital I/O PDI using standard read commands, instead, there is a direct connection for faster response times.

The process data watchdog (register 0x0440) has to be either active or disabled; otherwise digital outputs will not be updated. Digital outputs can be configured to be updated in four ways:

- Digital Outputs are updated at the end of each EtherCAT frame (EOF mode).
- Digital outputs are updated with Distributed Clocks SYNC0 events (DC SYNC0 mode).
- Digital outputs are updated with Distributed Clocks SYNC1 events (DC SYNC1 mode).
- Digital Outputs are updated at the end of an EtherCAT frame which triggered the Process Data Watchdog (with typical SyncManager configuration: a frame containing a write access to at least one of the registers 0x0F00:0x0F03). Digital Outputs are only updated if the EtherCAT frame was correct (WD_TRIG mode).

For Distributed Clock SYNC output, SYNC generation must be activated (register 0x0981). SYNC output is not necessary (register 0x0151). SYNC pulse length (registers 0x0982:0x0983) should not be set to 0, because acknowledging of SYNC events is not possible with Digital I/O PDI. Output time is the beginning of the SYNC event.

An output event is always signaled by a pulse on OUTVALID even if the digital outputs remain unchanged.

For output data to be visible on the I/O signals, the following conditions have to be met:

- SyncManager watchdog must be either active (triggered) or disabled.
- OE_EXT (Output enable) must be high,.
- Output values have to be written to the registers 0x0F00:0x0F03 within a valid EtherCAT frame.
- The configured output update event must have occurred.

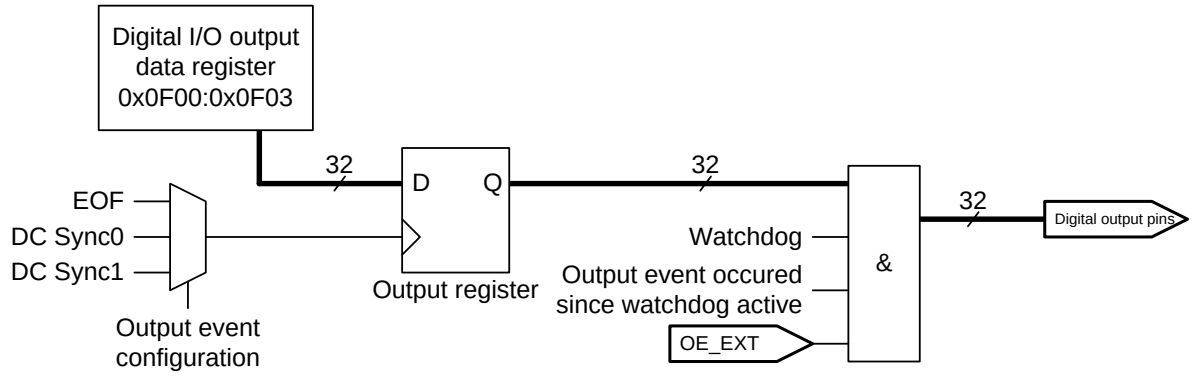

Figure 30: Digital Output Principle Schematic

NOTE: The Digital Outputs are not driven (high impedance) until the EEPROM is loaded. Depending on the FPGA configuration, Digital Outputs (like all other FPGA user pins) might have pull-up resistors until the FPGA has loaded its configuration. This behaviour has to be taken into account when using digital output signals.

10.1.5 Output Enable

The IP Core has an Output Enable signal OE_EXT. With the OE_EXT signal, the I/O signals can be cleared. The I/O signals will be driven low after the output enable signal OE_EXT is set to low or the SyncManager Watchdog is expired (and not disabled).

10.1.6 SyncManager Watchdog

The SyncManager watchdog (registers 0x0440:0x0441) must be either active (triggered) or disabled for output values to appear on the I/O signals. The SyncManager Watchdog is triggered by an EtherCAT write access to the output data registers.

If the output data bytes are written independently, a SyncManager with a length of 1 byte is used for each byte of 0x0F00:0x0F03 containing output bits (SyncManager N configuration: buffered mode, EtherCAT write/PDI read, and Watchdog Trigger enabled: 0x44 in register 0x0804+N*8). Alternatively, if all output data bits are written together in one EtherCAT command, one SyncManager with a length of 1 byte is sufficient (SyncManager N configuration: buffered mode, EtherCAT write/PDI read, and Watchdog Trigger enabled: 0x44 in register 0x0804+N*8). The start address of the SyncManager should be one of the 0x0F00:0x0F03 bytes containing output bits, e.g., the last byte containing output bits.

The SyncManager Watchdog can also be disabled by writing 0 into registers 0x0420:0x0421.

The Watchdog Mode configuration bit is used to configure if the expiration of the SyncManager Watchdog will have an immediate effect on the I/O signals (output reset immediately after watchdog timeout) or if the effect is delayed until the next output event (output reset with next output event). The latter case is especially relevant for Distributed Clock SYNC output events, because any output change will occur at the configured SYNC event.

For external watchdog implementations, the WD_TRIG (watchdog trigger) signal can be used. A WD_TRIG pulse is generated if the SyncManager Watchdog is triggered. In this case, the internal SyncManager Watchdog should be disabled, and the external watchdog may use OE_EXT to reset the I/O signals if the watchdog is expired. For devices without the WD_TRIG signal, OUTVALID can be configured to reflect WD_TRIG.

10.1.7 SOF

SOF indicates the start of an Ethernet/EtherCAT frame. It is asserted shortly after RX_DV=1 or EBUS SOF. Input data is sampled in the time interval between $t_{SOF\_to\_DATA\_setup}$ and $t_{SOF\_to\_DATA\_setup}$ after the SOF signal is asserted.

10.1.8 OUTVALID

A pulse on the OUTVALID signal indicates an output event. If the output event is configured to be the end of a frame, OUTVALID is issued shortly after RX_DV=0 or EBUS EOF, right after the CRC has been checked and the internal registers have taken their new values. OUTVALID is issued independent of actual output data values, i.e., it is issued even if the output data does not change.

10.1.9 Timing specifications

Table 41: Digital I/O timing characteristics IP Core

Parameter	Min	Max	Comment
t_{DATA_setup}	x^3		Input data valid before LATCH_IN
t_{DATA_hold}	x^3		Input data valid after LATCH_IN
t_{LATCH_IN}	x^3		LATCH_IN high time
t_{SOF}	40 ns – x^3	40 ns + x^3	SOF high time
$t_{SOF\_to\_DATA\_setup}$	0 ns	1,2 µs - x^3	Input data valid after SOF, so that Inputs can be read in the same frame
$t_{SOF\_to\_DATA\_hold}$	1,6 µs + x^3		Input data invalid after SOF
$t_{input\_event\_delay}$	440 ns		Time between consecutive input events
$t_{OUTVALID}$	80 ns – x^3	80 ns + x^3	OUTVALID high time
$t_{DATA\_to\_OUTVALID}$	80 ns – x^3		Output data valid before OUTVALID
t_{WD_TRIG}	40 ns – x^3	40 ns + x^3	WD_TRIG high time
$t_{DATA\_to\_WD\_TRIG}$		20 ns + x^3	Output data valid after WD_TRIG
$t_{OE\_EXT\_to\_DATA\_invalid}$	0 ns	x^3	Outputs zero or Outputs high impedance after OE_EXT set to low
$t_{output\_event\_delay}$	320 ns		Time between consecutive output events
$t_{OUT\_ENA\_valid}$	80 ns – x^3		OUT_ENA valid before OUTVALID
$t_{OUT\_ENA\_invalid}$	80 ns – x^3		OUT_ENA invalid after OUTVALID

[3] EtherCAT IP Core: time depends on synthesis results

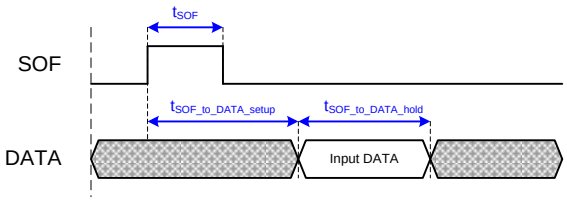

Figure 31: Digital Input: Input data sampled at SOF, I/O can be read in the same frame

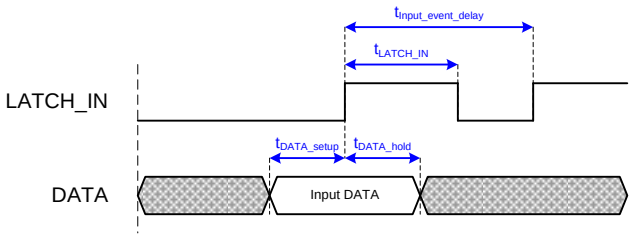

Figure 32: Digital Input: Input data sampled with LATCH_IN

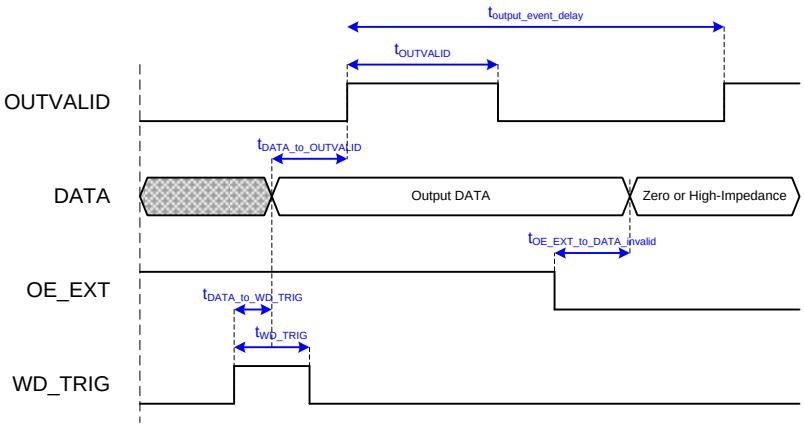

Figure 33: Digital Output timing

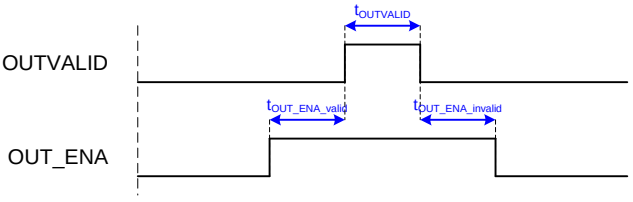

Figure 34: OUT_ENA timing

10.2 SPI Slave Interface

10.2.1 Interface

An EtherCAT device with PDI type 0x05 is an SPI slave. The SPI has 5 signals: SPI_CLK, SPI_DI (MOSI), SPI_DO (MISO), SPI_SEL and SPI_IRQ[4]:

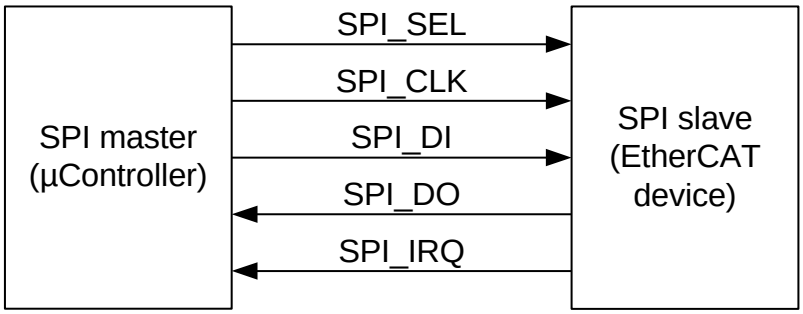

Figure 35: SPI master and slave interconnection

Table 42: SPI signals

Signal	Direction		Description	Signal polarity
SPI_SEL	IN	(master → slave)	SPI chip select	Typical: act. low
SPI_CLK	IN	(master → slave)	SPI clock	
SPI_DI	IN	(master → slave)	SPI data MOSI	act. high
SPI_DO	OUT	(slave → master)	SPI data MISO	act. high
SPI_IRQ	OUT	(slave → master)	SPI interrupt	Typical: act. low

10.2.2 Configuration

The SPI slave interface is selected with PDI type 0x05 in the PDI control register 0x0140. It supports different timing modes and configurable signal polarity for SPI_SEL and SPI_IRQ. The SPI configuration is located in register 0x0150.

[4] The prefix `PDI_` is added to the SPI signals if the EtherCAT IP Core is used.

10.2.3 SPI access

Each SPI access is separated into an address phase and a data phase. In the address phase, the SPI master transmits the first address to be accessed and the command. In the data phase, read data is presented by the SPI slave (read command) or write data is transmitted by the master (write command). The address phase consists of 2 or 3 bytes depending on the address mode. The number of data bytes for each access may range from 0 to N bytes. The slave internally increments the address for the following bytes after reading or writing the start address. The bits of both address/command and data are transmitted in byte groups.

The master starts an SPI access by asserting SPI_SEL and terminates it by taking back SPI_SEL (polarity determined by configuration). While SPI_SEL is asserted, the master has to cycle SPI_CLK eight times for each byte transfer. In each clock cycle, both master and slave transmit one bit to the other side (full duplex). The relevant edges of SPI_CLK for master and slave can be configured by selecting SPI mode and Data Out sample mode.

The most significant bit of a byte is transmitted first, the least significant bit last, the byte order is low byte first. EtherCAT devices use Little Endian byte ordering.

10.2.4 Address modes

The SPI slave interface supports two address modes, 2 byte addressing and 3 byte addressing. With two byte addressing, the lower 13 address bits A[12:0] are selected by the SPI master, while the upper 3 bits A[15:13] are assumed to be 000b inside the SPI slave, thus only the first 8 Kbyte in the EtherCAT slave address space can be accessed. Three byte addressing is used for accessing the whole 64 Kbyte address space of an EtherCAT slave.

For SPI masters which do only support consecutive transfers of more than one byte, additional Address Extension commands can be inserted.

Table 43: Address modes

Byte	2 Byte address mode		3 Byte address mode	
0	A[12:5]	address bits [12:5]	A[12:5]	address bits [12:5]
1	A[4:0] CMD0[2:0]	address bits [4:0] read/write command	A[4:0] CMD0[2:0]	address bits [4:0] 3 byte addressing: 110b
2	D0[7:0]	data byte 0	A[15:13] CMD1[2:0] res[1:0]	address bits [15:13] read/write command two reserved bits, set to 00b
3	D1[7:0]	data byte 1	D0[7:0]	data byte 0
4 ff.	D2[7:0]	data byte 2	D1[7:0]	data byte 1

10.2.5 Commands

The command CMD0 in the second address/command byte may be READ, READ with following Wait State bytes, WRITE, NOP, or Address Extension. The command CMD1 in the third address/command byte may have the same values:

Table 44: SPI commands CMD0 and CMD1

CMD[2]	CMD[1]	CMD[0]	Command
0	0	0	NOP (no operation)
0	0	1	reserved
0	1	0	Read
0	1	1	Read with following Wait State bytes
1	0	0	Write
1	0	1	reserved
1	1	0	Address Extension (3 address/command bytes)
1	1	1	reserved

10.2.6 Interrupt request register (AL Event register)

During the address phase, the SPI slave transmits the PDI interrupt request registers 0x0220-0x0221 (2 byte address mode), and additionally register 0x0222 for 3 byte addressing on SPI_DO (MISO):

Table 45: Interrupt request register transmission

Byte	2 Byte address mode			3 Byte address mode		
	SPI_DI (MOSI)	SPI_DO (MISO)		SPI_DI (MOSI)	SPI_DO (MISO)	
0	A[12:5]	I0[7:0]	interrupt request register 0x0220	A[12:5]	I0[7:0]	interrupt request register 0x0220
1	A[4:0] CMD0[2:0]	I1[7:0]	interrupt request register 0x0221	A[4:0] CMD0[2:0]	I1[7:0]	interrupt request register 0x0221
2	(Data phase)			A[15:13] CMD1[2:0]	I2[7:0]	interrupt request register 0x0222

10.2.7 Write access

In the data phase of a write access, the SPI master sends the write data bytes to the SPI slave (SPI_DI/MOSI). The write access is terminated by taking back SPI_SEL after the last byte. The SPI_DO signal (MISO) is undetermined during the data phase of write accesses.

10.2.8 Read access

In the data phase of a read access, the SPI slave sends the read data bytes to the SPI master (SPI_DO/MISO).

10.2.8.1 Read Wait State

Between the last address phase byte and the first data byte of a read access, the SPI master has to wait for the SPI slave to fetch the read data internally. Subsequent read data bytes are prefetched automatically, so no further wait states are necessary.

The SPI master can choose between these possibilities:

- The SPI master may either wait for the specified worst case internal read time t_{read} after the last address/command byte and before the first clock cycle of the data phase.
- The SPI master inserts one Wait State byte after the last address/command byte. The Wait State byte must have a value of 0xFF transferred on SPI_DI.

10.2.8.2 Read Termination

The SPI_DI signal (MOSI) is used for termination of the read access by the SPI master. For the last data byte, the SPI master has to set SPI_DI to high (Read Termination byte = 0xFF), so the slave will not prefetch the next read data internally. If SPI_DI is low during a data byte transfer, at least one more byte will be read by the master afterwards.

10.2.9 SPI access errors and SPI status flag

The following reasons for SPI access errors are detected by the SPI slave:

- The number of clock cycles recognized while SPI_SEL is asserted is not a multiple of 8 (incomplete bytes were transferred).
- For a read access, a clock cycle occurred while the slave was busy fetching the first data byte.
- For a read access, the data phase was not terminated by setting SPI_DI to high for the last byte.
- For a read access, additional bytes were read after termination of the access.

A wrong SPI access will have these consequences:

- Registers will not accept write data (nevertheless, RAM will be written).
- Special functions are not executed (e.g., SyncManager buffer switching).
- The PDI error counter 0x030D will be incremented.
- A status flag will indicate the error until the next access (not for SPI mode 0/2 with normal data out sample)

A status flag, which indicates if the last access had an error, is available in any mode except for SPI mode 0/2 with normal data out sample. The status flag is presented on SPI_DO (MISO) after the slave is selected (SPI_SEL) and until the first clock cycle occurs. So the status can be read either between two accesses by assertion of SPI_SEL without clocking, or at the beginning of an access just before the first clock cycle. The status flag will be high for a good access, and low for a wrong access.

The reason of the access error can be read in the PDI error code register 0x030E.

10.2.10 2 Byte and 4 Byte SPI Masters

Some SPI masters do not allow an arbitrary number of bytes per access, the number of bytes per access must be a multiple of 2 or 4 (maybe even more). The SPI slave interface supports such masters. The length of the data phase is in control of the master and can be set to the appropriate length, the length of the address phase has to be extended. The address phase of a read access can be set to a multiple of 2/4 by using the 3 byte address mode and a wait state byte. The address phase of a write access can be enhanced to 4 bytes using 3 byte address mode and an additional address extension byte (byte 2) according to Table 46.

Table 46: Write access for 2 and 4 Byte SPI Masters

Byte	2 Byte SPI master		4 Byte SPI master	
0	A[12:5]	address bits [12:5]	A[12:5]	address bits [12:5]
1	A[4:0] CMD0[2:0]	address bits [4:0] write command: 100b	A[4:0] CMD0[2:0]	address bits [4:0] 3 byte addressing: 110b
2	D0[7:0]	data byte 0	A[15:13] CMD1[2:0] res[1:0]	address bits [15:13] 3 byte addressing: 110b two reserved bits, set to 00b
3	D1[7:0]	data byte 1	A[15:13] CMD2[2:0] res[1:0]	address bits [15:13] write command: 100b two reserved bits, set to 00b
4	D2[7:0]	data byte 2	D0[7:0]	data byte 0
5	D3[7:0]	data byte 3	D1[7:0]	data byte 1
6	D4[7:0]	data byte 4	D2[7:0]	data byte 2
7	D5[7:0]	data byte 5	D3[7:0]	data byte 3

NOTE: The address phase of a write access can be further extended by an arbitrary number of address extension bytes containing 110b as the command. The address phase of a read access can also be enhanced with additional address extension bytes (the read wait state has to be maintained anyway). The address portion of the last address extension byte is used for the access.

10.2.11 Timing specifications

Table 47: SPI timing characteristics IP Core

Parameter	Min	Max	Comment
t_{CLK}	33 ns+x[5]		SPI_CLK frequency (f_{CLK} ≤ 30 MHz)
$t_{SEL\_to\_CLK}$	x[5]		First SPI_CLK cycle after SPI_SEL asserted
$t_{CLK\_to\_SEL}$	a) x[5] b) t_{CLK}/2+ x[5]		Deassertion of SPI_SEL after last SPI_CLK cycle a) SPI mode 0/2, SPI mode 1/3 with normal data out sample b) SPI mode 1/3 with late data out sample
t_{read}	240 ns		Only for read access between address/command and first data byte. Can be ignored if BUSY or Wait State Bytes are used.
$t_{C0\_to\_BUSY\_OE}$	t_{CLK}		BUSY OUT Enable assertion after sample time of last command bit C0.
t_{BUSY_valid}		x[5]	BUSY valid after BUSY OUT Enable
$t_{BUSY\_OE\_to\_DO\_valid}$		x[5]	Only for SPI mode 0/2 with normal data out sampling: Data byte 0 bit 7 valid after deassertion of BUSY OUT Enable
$t_{SEL\_to\_DO\_valid}$		x[5]	Status/Interrupt Byte 0 bit 7 valid after SPI_SEL asserted
$t_{SEL\_to\_DO\_invalid}$	0 ns	x[5]	Status/Interrupt Byte 0 bit 7 invalid after SPI_SEL deasserted
t_{STATUS_valid}	x[5]		Time until status of last access is valid. Can be ignored if status is not used.
t_{access_delay}	x[5]		Delay between SPI accesses
t_{DI_setup}	x[5]		SPI_DI valid before SPI_CLK edge
t_{DI_hold}	x[5]		SPI_DI valid after SPI_CLK edge
$t_{CLK\_to\_DO\_valid}$		x[5]	SPI_DO valid after SPI_CLK edge
$t_{CLK\_to\_DO\_invalid}$	0 ns		SPI_DO invalid after SPI_CLK edge
t_{IRQ_delay}	160 ns		Internal delay between AL event and SPI_IRQ output to enable correct reading of the interrupt registers.

[5] EtherCAT IP Core: time depends on synthesis results

Table 48: Read/Write timing diagram symbols

Symbol	Comment
A15..A0	Address bits [15:0]
D0_7..D0_0 D1_7..D1_0	Data bits byte 0 [7:0] Data bits byte 1 [7:0]
I0_7..I0_0 I1_7..I1_0 I2_7..I2_0	Interrupt request register 0x0220 [7:0] Interrupt request register 0x0221 [7:0] Interrupt request register 0x0222 [7:0]
C0_2..C0_0 C1_2..C1_0	Command 0 [2:0] Command 1 [2:0] (3 byte addressing)
Status	0: last SPI access had errors 1: last SPI access was correct
BUSY OUT Enable	0: No Busy output, tread is relevant 1: Busy output on SPI_DO (edge sensitive)
BUSY	0: SPI slave has finished reading first byte 1: SPI slave is busy reading first byte

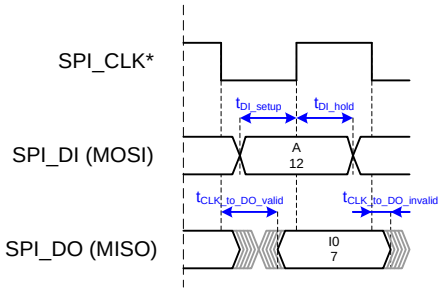

Figure 36: Basic SPI_DI/SPI_DO timing (*refer to timing diagram for relevant edges of SPI_CLK)

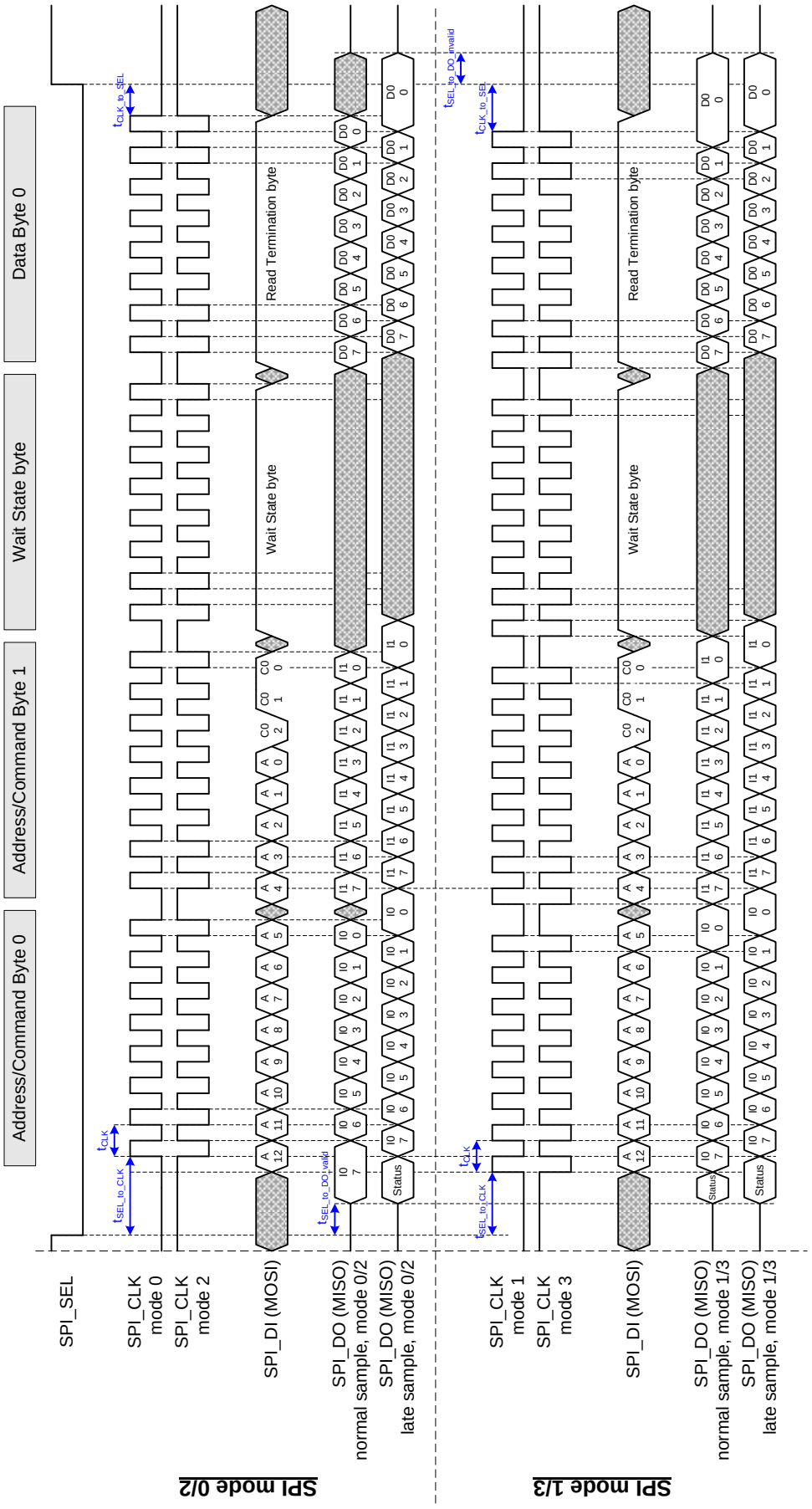

Figure 37: SPI read access (2 byte addressing, 1 byte read data) with Wait State byte

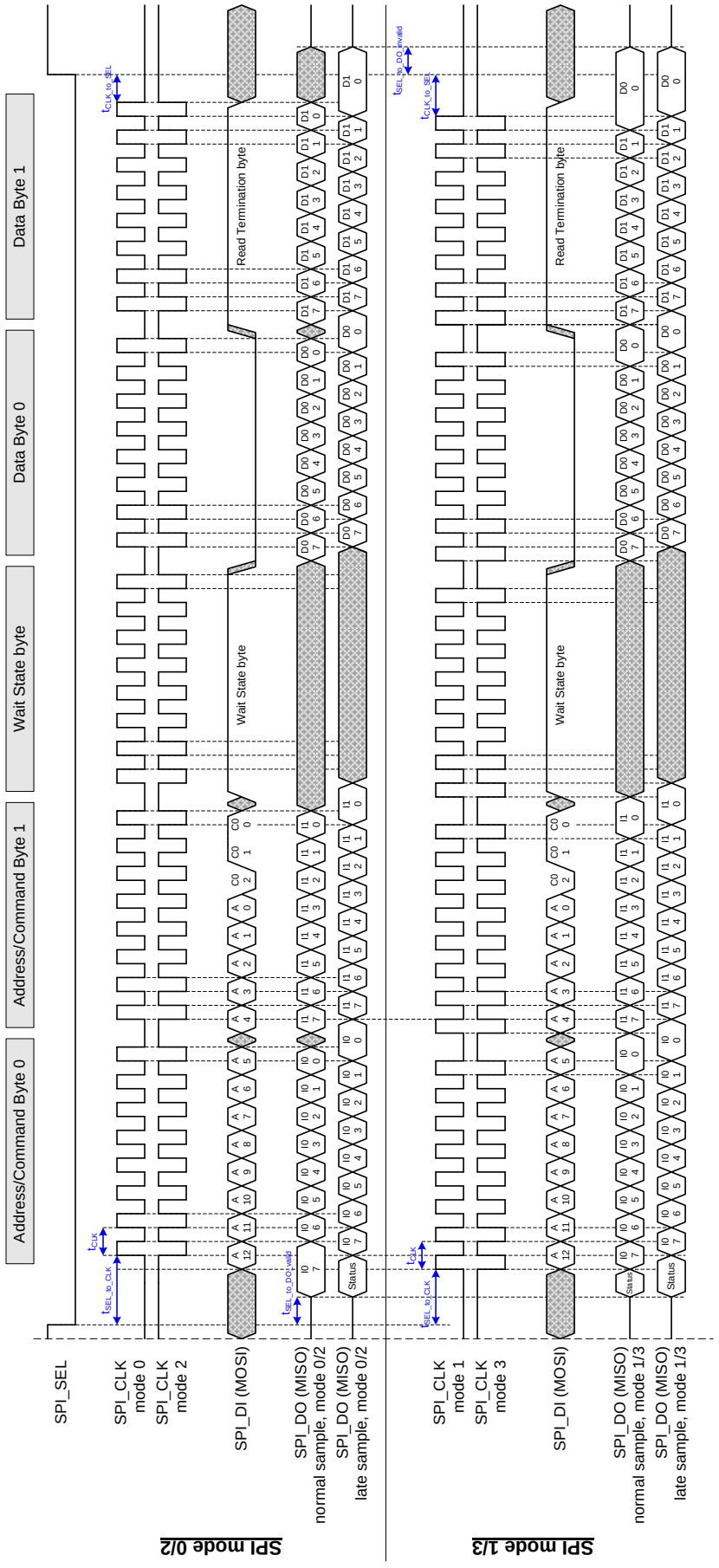

Figure 38: SPI read access (2 byte addressing, 2 byte read data) with Wait State byte

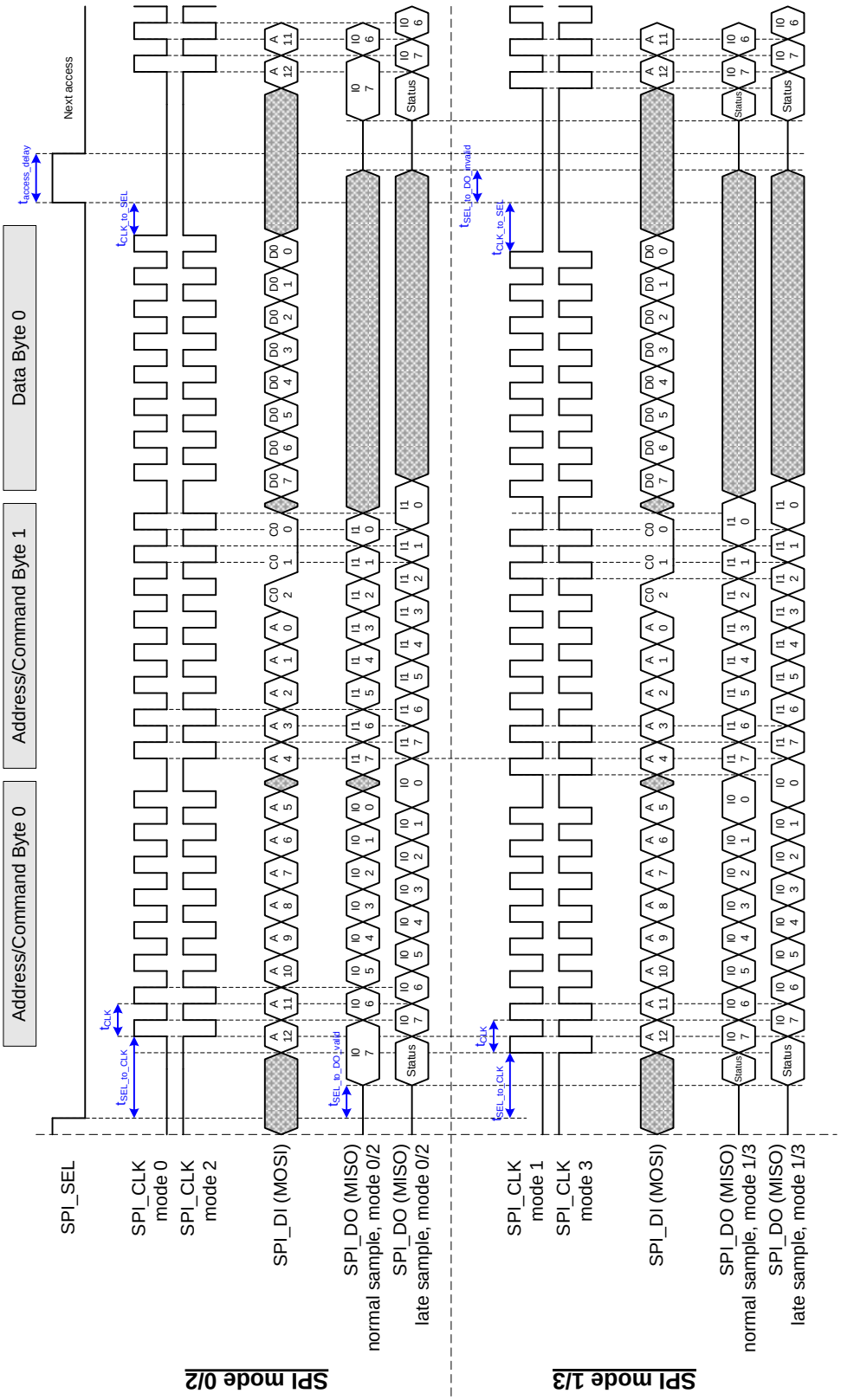

Figure 39: SPI write access (2 byte addressing, 1 byte write data)

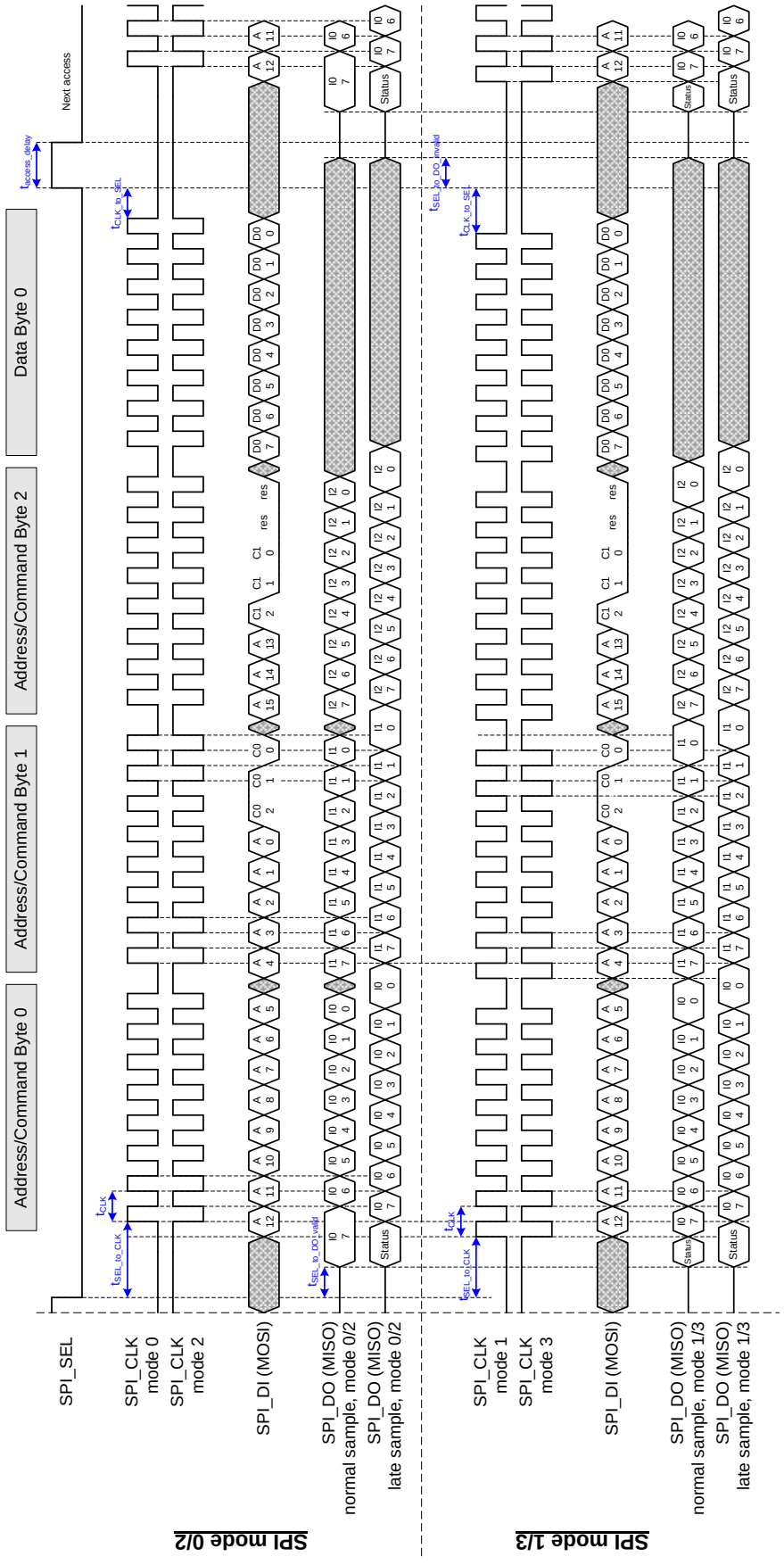

Figure 40: SPI write access (3 byte addressing, 1 byte write data)

10.3 Asynchronous 8/16 bit µController Interface

10.3.1 Interface

The asynchronous µController interface uses demultiplexed address and data busses. The bidirectional data bus can be either 8 bit or 16 bit wide. The signals of the asynchronous µController interface of EtherCAT devices are[6]:

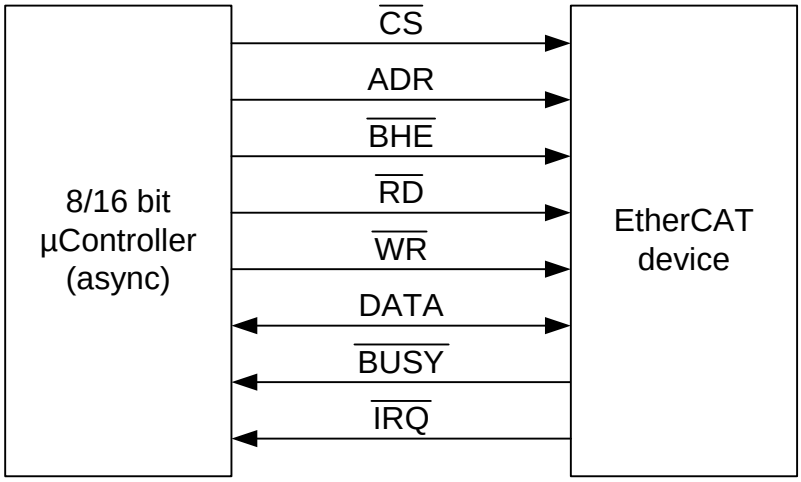

Figure 41: µController interconnection[7]

Table 49: µController signals

Signal async	Direction		Description	Signal polarity
CS	IN	(µC → ESC)	Chip select	Typical: act. low
ADR[15:0]	IN	(µC → ESC)	Address bus	Typical: act. high
BHE	IN	(µC → ESC)	Byte High Enable (16 bit µController interface only)	Typical: act. low
RD	IN	(µC → ESC)	Read command	Typical: act. low
WR	IN	(µC → ESC)	Write command	Typical: act. low
DATA[15:0]	BD	(µC ↔ ESC)	Data bus for 16 bit µController interface	act. high
DATA[7:0]	BD	(µC ↔ ESC)	Data bus for 8 bit µController interface	act. high
BUSY	OUT	(ESC → µC)	EtherCAT device is busy	Typical: act. low
IRQ	OUT	(ESC → µC)	Interrupt	Typical: act. low

Some µControllers have a READY signal, this is the same as the BUSY signal, just with inverted polarity.

10.3.2 Configuration

The 16 bit asynchronous µController interface is selected with PDI type 0x08 in the PDI control register 0x0140, the 8 bit asynchronous µController interface has PDI type 0x09. It supports different configurations, which are located in registers 0x0150 – 0x0153.

[6] The prefix `PDI_uC_` or `PDI_uC_8` is added to the µController signals if the EtherCAT IP Core is used.
[7] All signals are denoted with typical polarity configuration.

10.3.3 µController access

The 8 bit µController interface reads or writes 8 bit per access, the 16 bit µController interface supports both 8 bit and 16 bit read/write accesses. For the 16 bit µController interface, the least significant address bit together with Byte High Enable (BHE) are used to distinguish between 8 bit low byte access, 8 bit high byte access and 16 bit access.

EtherCAT devices use Little Endian byte ordering.

Table 50: 8 bit µController interface access types

ADR[0]	Access	DATA[7:0]
0	8 bit access to ADR[15:0] (low byte, even address)	low byte
1	8 bit access to ADR[15:0] (high byte, odd address)	high byte

Table 51: 16 bit µController interface access types

ADR[0]	BHE (act. low)	Access	DATA [15:8]	DATA [7:0]
0	0	16 bit access to ADR[15:0] and ADR[15:0]+1 (low and high byte)	**high byte**	**low byte**
0	1	8 bit access to ADR[15:0] (low byte, even address)	(RD only: copy of low byte)	**low byte**
1	0	8 bit access to ADR[15:0] (high byte, odd address)	**high byte**	(RD only: copy of high byte)
1	1	invalid access	-	-

10.3.4 Write access

A write access starts with assertion of Chip Select (CS), if it is not permanently asserted. Address, Byte High Enable and Write Data are asserted with the falling edge of WR (active low). Once the µController interface is not BUSY, a rising edge on WR completes the µController access. A write access can be terminated either by deassertion of WR (while CS remains asserted), or by deassertion or CS (while WR remains asserted), or even by deassertion of WR and CS simultaneously. Shortly after the rising edge of WR, the access can be finished by deasserting ADR, BHE and DATA. The µController interface indicates its internal operation with the BUSY signal. Since the BUSY signal is only driven while CS is asserted, the BUSY driver will be released after CS deassertion.

Depending on the configuration, the internal write access is either performed after the falling edge of WR, or after the rising edge of WR. If the falling edge is selected, the internal write operation begins with the falling edge of WR, and BUSY indicates when the write operation is finished. The internal write operation is performed during the external write access.

If the rising edge of WR is selected, the internal operation begins with the rising edge of WR, i.e., after the external write access. Thus, the external write access is very fast, but an access immediately following will be delayed by the preceding write access. The maximum access time is higher in this case.

10.3.5 Read access

A read access starts with assertion of Chip Select (CS), if it is not permanently asserted. Address and BHE have to be valid before the falling edge of RD, which signals the start of the access. The µController interface will show its BUSY state afterwards – if it is not already busy executing a preceding write access – and release BUSY when the read data are valid. The read data will remain valid until either ADR, BHE, RD or CS change. The data bus will be driven while CS and RD are asserted. BUSY will be driven while CS is asserted.

With read busy delay configuration, BUSY deassertion for read accesses can be additionally delayed for 15 ns, so external DATA setup requirements in respect to BUSY can be met.

10.3.6 µController access errors

These reasons for µController access errors are detected by the µController interface:

- Read or Write access to the 16 bit interface with A[0]=1 and BHE(act. low)=1, i.e. an access to an odd address without Byte High Enable.
- Deassertion of WR (or deassertion of CS while WR remains asserted) while the µController interface is BUSY.
- Deassertion of RD (or deassertion of CS while RD remains asserted) while the µController interface is BUSY (read has not finished).

A wrong µController access will have these consequences:

- The PDI error counter 0x030D will be incremented.
- For A[0]=1 and BHE(act. low)=1 accesses, no access will be performed internally.
- Deassertion of WR (or CS) while the µController interface is BUSY might corrupt the current and the preceding transfer (if it is not completed internally). Registers might accept write data and special functions (e.g., SyncManager buffer switching) might be performed.
- If RD (or CS) is deasserted while the µController interface is BUSY (read has not finished), the access will be terminated internally. Although, internal byte transfers might be completed, so special functions (e.g., SyncManager buffer switching) might be performed.

The reason of the access error can be read in the PDI error code register 0x030E.

10.3.7 Connection with 16 bit µControllers without byte addressing

If the ESC is connected to 16 bit µControllers/DSPs which only support 16 bit (word) addressing, ADR[0] and BHE of the EtherCAT device have to be tied to GND, so the ESC will always perform 16 bit accesses. All other signals are connected as usual. Please note that ESC addresses have to be divided by 2 in this case.

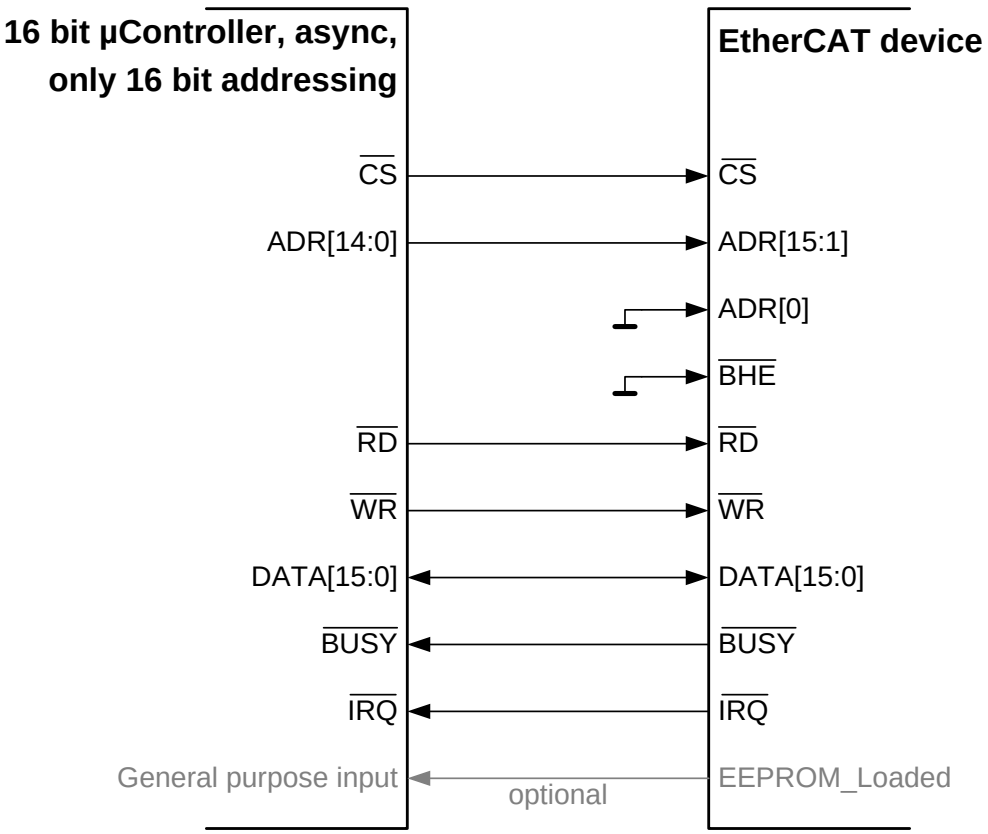

Figure 42: Connection with 16 bit µControllers without byte addressing

10.3.8 Connection with 8 bit µControllers

If the ESC is connected to 8 bit µControllers, the BHE signal as well as the DATA[15:8] signals are not used.

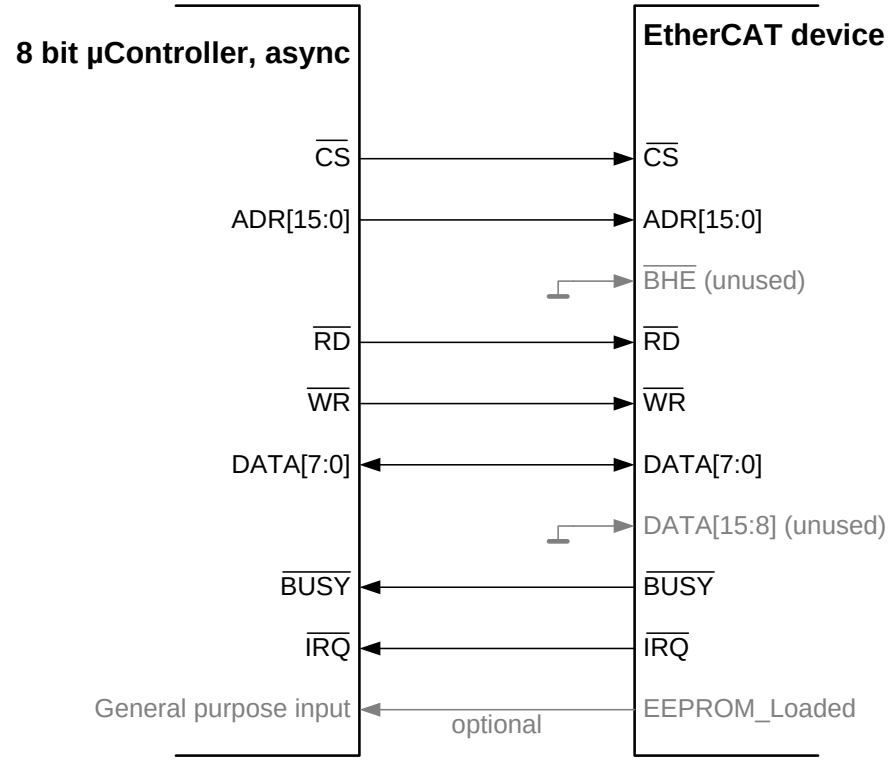

Figure 43: Connection with 8 bit µControllers (BHE and DATA[15:8] should not be left open)

10.3.9 Timing Specification

Table 52: µController timing characteristics IP Core

Parameter	Min	Max	Comment
$t_{CS\_to\_BUSY}$		x[8]	BUSY driven and valid after CS assertion
$t_{ADR\_BHE\_setup}$	x[8]		ADR and BHE valid before RD assertion
$t_{RD\_to\_DATA\_driven}$	0 ns[9]		DATA bus driven after RD assertion
$t_{RD\_to\_BUSY}$	0 ns[9]	x[8]	BUSY asserted after RD assertion
t_{read}			External read time (RD assertion to BUSY deassertion) with normal read busy output (0x0152[0]). Additional 20 ns if delayed read busy output is configured.
		a) t_{read_int}[9]	a) without preceding write access or $t_{WR\_to\_RD} \geq t_{prec\_write} + t_{Coll}$ or configuration: write after falling edge of WR
		b) $t_{read\_int} + t_{prec\_write} + t_{Coll} - t_{WR\_to\_RD}$[9]	b) with preceding write access and $t_{WR\_to\_RD} < t_{prec\_write} + t_{Coll}$
		c) 420 ns[9]	c) 8 bit access, absolute worst case with preceding write access ($t_{WR\_to\_RD}$=min, t_{prec_write} =max, t_{Coll}=max)
		d) 560 ns[9]	d) 16 bit access, absolute worst case with preceding write access ($t_{WR\_to\_RD}$=min, t_{prec_write} =max, t_{Coll}=max)
t_{read_int}		a) 220 ns[9] b) 300 ns[9]	Internal read time a) 8 bit access b) 16 bit access
t_{prec_write}		a) 180 ns b) 260 ns	Time for preceding write access a) 8 bit access b) 16 bit access
$t_{BUSY\_to\_DATA\_valid}$		a) x[8]-5 ns b) x[8]-20 ns	DATA bus valid after device BUSY is deasserted a) normal read busy output b) delayed read busy output
$t_{ADR\_BHE\_to\_DATA\_invalid}$	0 ns[9]		DATA invalid after ADR or BHE change
$t_{CS\_RD\_to\_DATA\_release}$	0 ns[9]	x[8]	DATA bus released after CS deassertion or RD deassertion
$t_{CS\_to\_BUSY\_release}$	0 ns[9]	x[8]	BUSY released after CS deassertion
t_{CS_delay}	0 ns[9]		Delay between CS deassertion an assertion
t_{RD_delay}	x[8]		Delay between RD deassertion and assertion
$t_{ADR\_BHE\_DATA\_setup}$	x[8]		ADR, BHE and Write DATA valid before WR deassertion

[8] EtherCAT IP Core: time depends on synthesis results
[9] EtherCAT IP Core: time depends on synthesis results, specified value has to be met anyway

Parameter	Min	Max	Comment
$t_{ADR\_BHE\_DATA\_hold}$	x[8]		ADR, BHE and Write DATA valid after WR deassertion
t_{WR_active}	x[8]		WR assertion time
$t_{BUSY\_to\_WR\_CS}$	0 ns[9]		WR or CS deassertion after BUSY deassertion
$t_{WR\_to\_BUSY}$		x[8]	BUSY assertion after WR deassertion
t_{write}	0 ns		External write time (WR assertion to BUSY deassertion)
		a) t_{write_int}	a) Configuration: write after falling edge of WR (act. low)
		b) t_{write_int} -t_{WR_delay}[9]	b) with preceding write access and t_{WR_delay} < t_{write_int} (Write after rising edge of WR)
		c) 0 ns[9]	c) without preceding write access or t_{WR_delay} ≥ t_{write_int} (Write after rising edge of WR)
		d) 180 ns[9]	d) 8 bit access, absolute worst case with preceding write access (t_{WR_delay}= min, t_{WR_int}=max, Write after rising edge of WR)
		e) 260 ns[9]	e) 16 bit access, absolute worst case with preceding write access (t_{WR_delay}=min, t_{WR_int}=max, Write after rising edge of WR)
t_{write_int}		a) 180 ns[9] b) 260 ns[9]	Internal write time a) 8 bit access b) 16 bit access
t_{WR_delay}	x[8]		Delay between WR deassertion and assertion
t_{Coll}		a) 20 ns b) 0 ns	Extra read delay a) RD access directly follows WR access with the same address (8 bit accesses or 8 bit WR and 16 bit RD) b) different addresses or 16 bit accesses
$t_{WR\_to\_RD}$	0 ns		Delay between WR deassertion and RD assertion
$t_{CS\_WR\_overlap}$	x[8]		Time both CS and WR have to be deasserted simultaneously (only if CS is deasserted at all)
$t_{CS\_RD\_overlap}$	x[8]		Time both CS and RD have to be deasserted simultaneously (only if CS is deasserted at all)

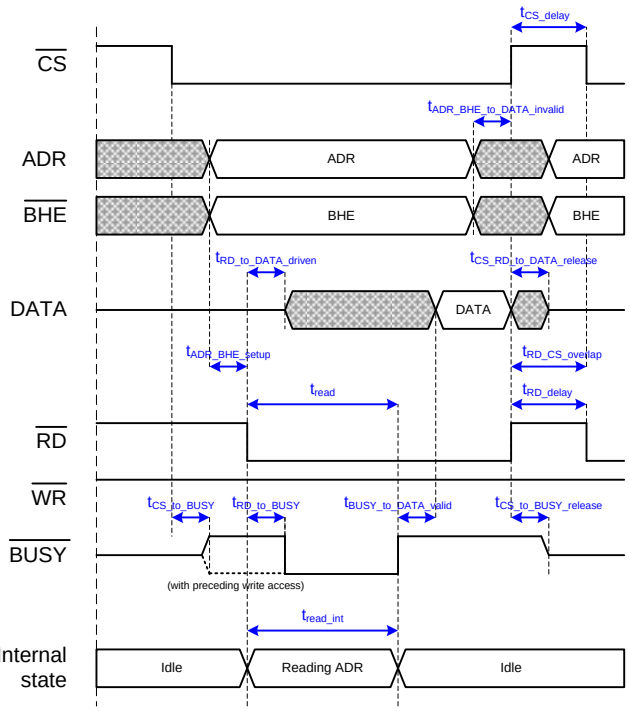

Figure 44: Read access (without preceding write access)

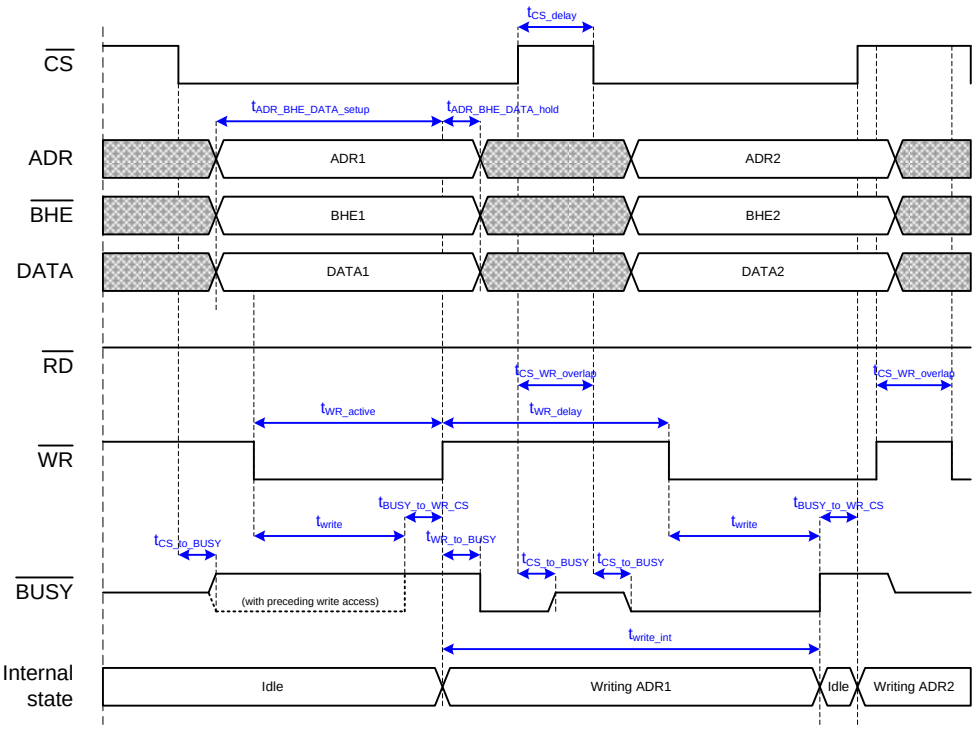

Figure 45: Write access (write after rising edge nWR, without preceding write access)

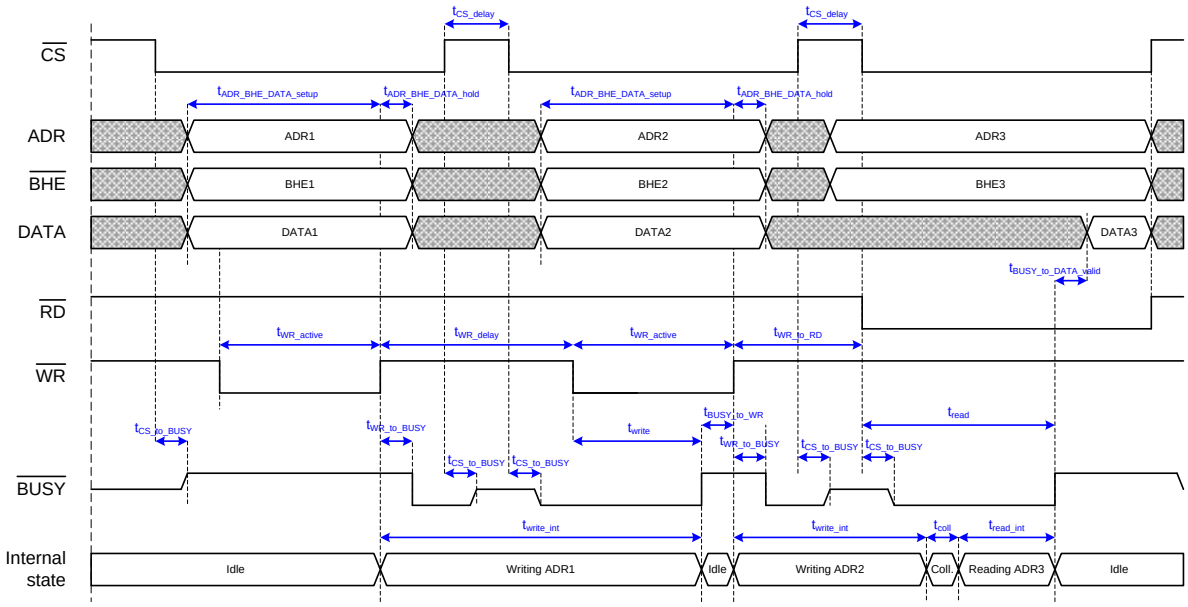

Figure 46: Sequence of two write accesses and a read access

Note: The first write access to ADR1 is performed after the first rising edge of WR. After that, the ESC is internally busy writing to ADR1. After CS is deasserted, BUSY is not driven any more, nevertheless, the ESC is still writing to ADR1.

Hence, the second write access to ADR2 is delayed because the write access to ADR1 has to be completed first. So, the second rising edge of WR must not occur before BUSY is gone. After the second rising edge of WR, the ESC is busy writing to ADR2. This is reflected with the BUSY signal as long as CS is asserted.

The third access in this example is a read access. The ESC is still busy writing to ADR2 while the falling edge of RD occurs. In this case, the write access to ADR2 is finished first, and afterwards, the read access to ADR3 is performed. The ESC signals BUSY during both write and read access.

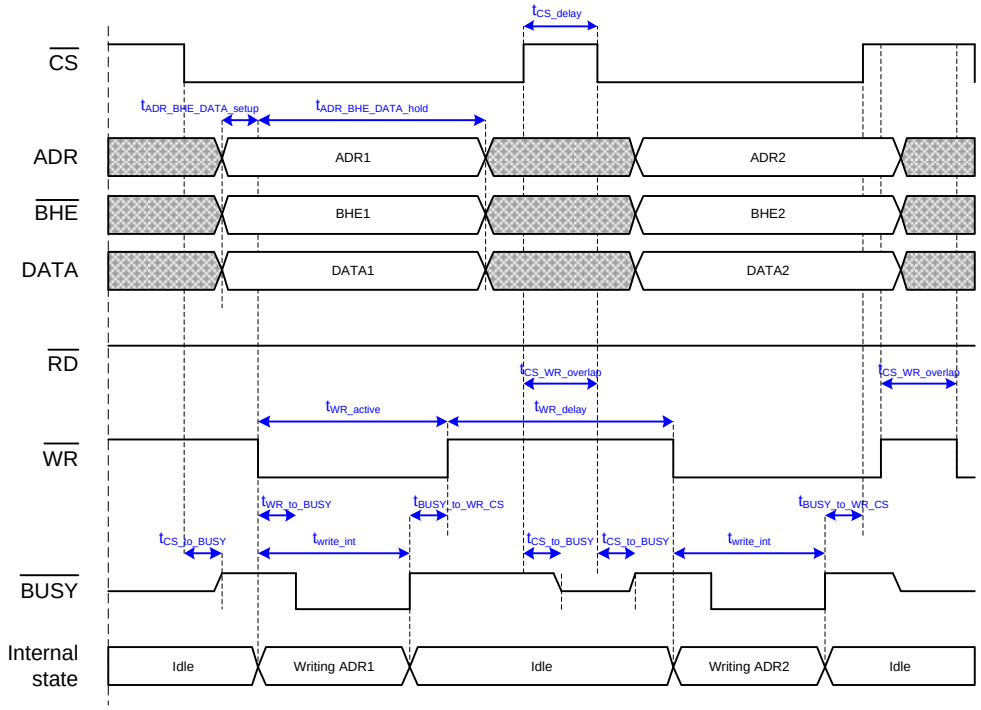

Figure 47: Write access (write after falling edge nWR)

10.4 Avalon Slave Interface

10.4.1 Interface

The Avalon Slave PDI is selected during the IP Core configuration. It uses memory addressing/dynamic bus sizing. The signals of the Avalon interface are:

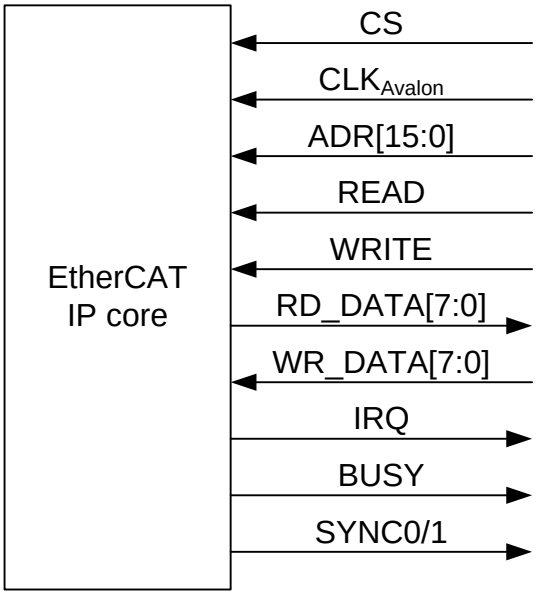

Figure 48: Avalon signals

Table 53: Avalon signals

Signal	Direction	Description	Signal polarity
PDI_AVALON_CS	IN	Chip select	act. high
PDI_AVALON_CLK	IN	Avalon Bus clock (rising edge synchronous with rising edge of CLK25 of the IP Core)	
PDI_AVALON_ADR[15:0]	IN	Address	
PDI_AVALON_READ	IN	Read request	act. high
PDI_AVALON_WRITE	IN	Write request	act. high
PDI_AVALON_RD_DATA[7:0]	OUT	Read data	
PDI_AVALON_WR_DATA[7:0]	IN	Write data	
PDI_AVALON_IRQ	OUT	Interrupt	act. high
PDI_AVALON_SYNC0/1	OUT	DC Sync0/1 used as interrupts	act. high
PDI_AVALON_BUSY	OUT	Busy / Wait request	act. high

Please refer to the Avalon Memory-Mapped Interface Specification from Altera for details about the Avalon bus (http://www.altera.com).

10.4.2 Configuration

The Avalon interface has PDI type 0x80 in the PDI control register 0x0140. The Avalon clock speed and the Avalon master data width are configurable in the Avalon PDI configuration dialog.

Avalon Clock Multiplier

The Avalon clock frequency is a multiple of 25 MHz:

Avalon clock frequency = N * 25 MHz (N=1...31)

The maximum clock speed depends on the FPGA and the synthesis. The rising edge of Avalon clock has to be synchronous with the rising edge of CLK25 of the EtherCAT IP Core.

Data width of smallest Avalon Master

The Avalon Slave interface prefetches read data internally for faster read accesses. The number of read data bytes which will be prefetched is configurable and should be set to the smallest data with of the Avalon masters (or the smallest number of bytes a master of the Avalon bus will read in one access, e.g., if byte enables are used for read accesses).

Valid values are W = 1, 2, or 4 Bytes. Select W = 4 Bytes for the Altera NIOS processor, unless the IP Core is used in Qsys systems. For Qsys systems, the Data width has to be set to W = 1 Byte.

10.4.3 Interrupts

The Avalon Slave interface supports up to 3 interrupts for easy connection in the Altera SOPC Builder:

- the global PDI interrupt (IRQ)
- DC SYNC0 and DC SYNC1. These interrupts are available if DC are selected. The DC SyncSignals are also available as standard DC Sync0/1 signals.

10.4.4 Data Bus With and SyncManager Configuration

Since an Avalon master always performs read accesses with the whole data bus width (e.g. 32 bit for a NIOS II processor) regardless of the actually issued read command (8/16/32 bit) without use of byte enable signals, care has to be taken especially for SyncManager configuration. SyncManagers should be configured with a length and alignment of multiples of this data width. For write accesses, byte enable signals are used to identify bytes which have to be written.

10.4.5 Timing specifications

Table 54: Avalon timing characteristics (IP Core V1.1.1)

Parameter	Min	Max	Comment
N	1	31	Avalon bus clock factor
W	1, 2, or 4		Master data width in Bytes
t_{Clk}	$\frac{1}{N * 25MHz}$ [10]	40 ns	Avalon bus clock (Avalon clock frequency: N*25 MHz)
t_{Read}	440 ns	a) 560 ns b) $560\,ns + \frac{40ns}{N}$	32 Bit read access time (W=4) a) N=1 b) N>1
	280 ns	a) 400 ns b) $400\,ns + \frac{40ns}{N}$	16 Bit read access time (W=2) a) N=1 b) N>1
	200 ns	a) 320 ns b) $320\,ns + \frac{40ns}{N}$	8 Bit read access time (W=1) a) N=1 b) N>1
t_{Write}	240 ns	a) 320 ns b) $320ns + \frac{40ns}{N}$	32 Bit write access time (W=4) a) N=1 b) N>1
	80 ns	a) 160 ns b) $160ns + \frac{40ns}{N}$	16 Bit write access time (W=2) a) N=1 b) N>1
	t_{Clk}		8 Bit write access time (W=1)

[10] EtherCAT IP Core: time depends on synthesis results

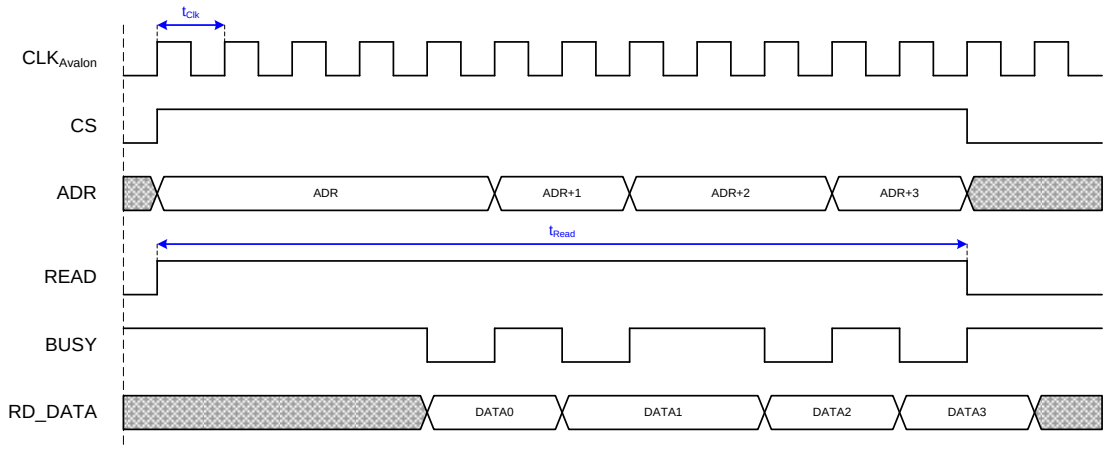

Figure 49: Avalon Read Access (32 Bit, W=4)

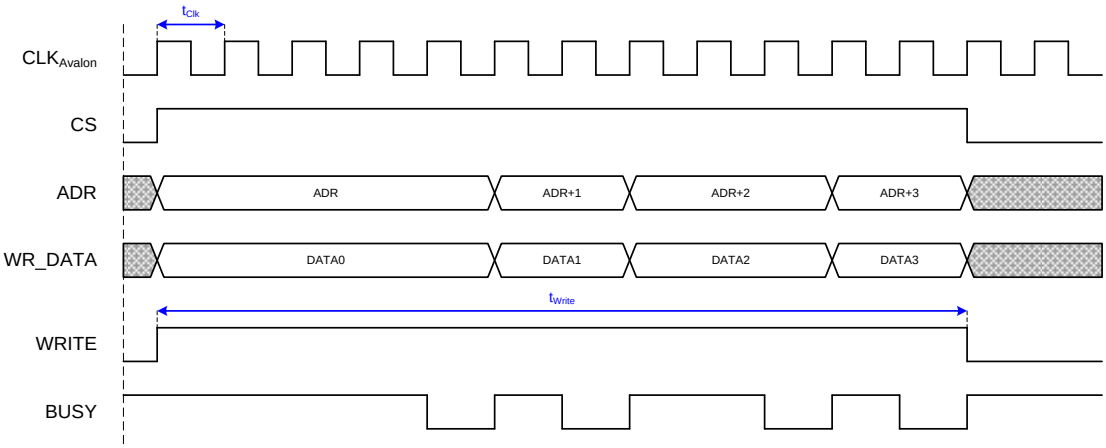

Figure 50: Avalon Write Access (32 Bit, W=4)

11 Distributed Clocks SYNC/LATCH Signals

For details about the Distributed Clocks refer to Section I.

11.1 Signals

The Distributed Clocks unit of the IP Core has the following external signals (depending on the ESC configuration):

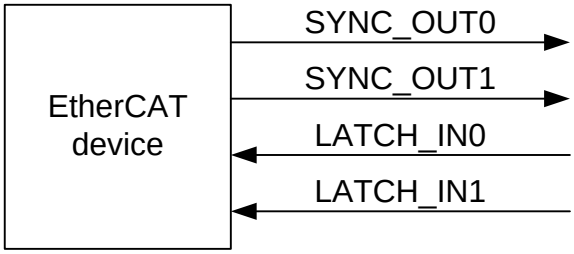

Figure 51: Distributed Clocks signals

Table 55: Distributed Clocks signals

Signal	Direction	Description
SYNC_OUT0/1	OUT	SyncSignals (alias SYNC[1:0])
LATCH_IN0/1	IN	LatchSignals (alias LATCH[1:0])

NOTE: SYNC_OUT0/1 are active high/push-pull outputs.

11.2 Timing specifications

Table 56: DC SYNC/LATCH timing characteristics IP Core

Parameter	Min	Max	Comment
t_{DC_LATCH}	12 ns + x[11]		Time between Latch0/1 events
$t_{DC\_SYNC\_Jitter}$		11 ns + x[1]	SYNC0/1 output jitter

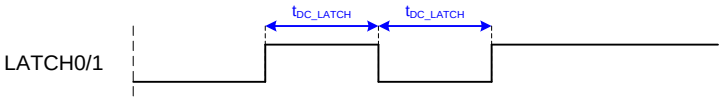

Figure 52: LatchSignal timing

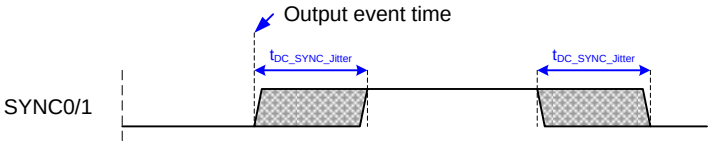

Figure 53: SyncSignal timing

[11] EtherCAT IP Core: time depends on synthesis results

12 SII EEPROM Interface (I²C)

For details about the ESC SII EEPROM Interface refer to Section I. The SII EEPROM Interface is intended to be a point-to-point interface between IP Core and I²C EEPROM. If other I²C masters are required to access the I²C bus, the IP Core must be held in reset state (e.g. for in-circuit-programming of the EEPROM).

12.1 Signals

The EEPROM interface of the IP Core has the following signals:

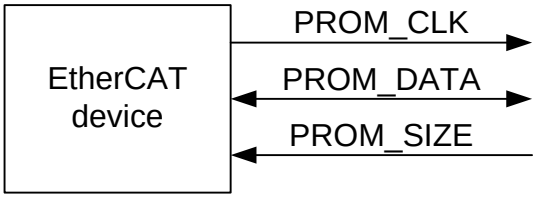

Figure 54: I²C EEPROM signals

Table 57: I²C EEPROM signals

Signal	Direction	Description
PROM_CLK	OUT	I²C clock (alias EEPROM_CLK)
PROM_DATA	BIDIR	I²C data (alias EEPROM_DATA)
PROM_SIZE	IN	EEPROM size configuration (alias EEPROM_SIZE)

Both EEPROM_CLK and EEPROM_DATA must have a pull-up resistor (4.7 kΩ recommended for ESCs), either integrated into the ESC or externally.

12.2 EEPROM Emulation

EEPROM_SIZE has to be 0 for EEPROM emulation (EEPROM emulation with EEPROM_SIZE=1 is for testing only: all commands are acknowledged automatically).

12.3 Timing specifications

Table 58: EEPROM timing characteristics IP Core

Parameter	Typical		Comment
	Up to 16 kBit	32 kBit-4 MBit	
t_{Clk}	~ 6.72 µs		EEPROM clock period (f_{Clk} ≈ 150 kHz)
t_{Write}	~ 250 us	~ 310 µs	Write access time (without errors)
t_{Read}	a) ~ 440 µs b) ~ 1.16 ms	a) ~ 500 µs b) ~ 1.22 ms	Read access time (without errors): a) 2 words b) configuration (8 Words)
t_{Delay}	~ 60 µs		Time until configuration loading begins after Reset is gone

13 Electrical Specifications

Table 59: AC Characteristics

Symbol	Parameter	Min	Typ	Max	Units
f_{CLK25}	Clock source (CLK25) with initial accuracy		25 MHz ± 25 ppm		

Table 60: Forwarding Delays

Symbol	Parameter	Min	Average	Max	Units
t_{Diff}	Average difference processing delay minus forwarding delay (without RX FIFO jitter)		40		ns
t_{MM}	MII port to MII port delay: a) Through ECAT Processing Unit (processing) b) Alongside ECAT Processing Unit (forwarding) Conditions: FIFO size 7, no TX Shift compensation or manual TX Shift configuration with MII_TX_SHIFT = 00	a) 320+x[12] b) 280+x[12]	a) 340+x[12] b) 300+x[12]	a) 360+x[12] b) 320+x[12]	ns

NOTE: Average timings are used for DC calculations.

[12] EtherCAT IP Core: time depends on synthesis results

14 Synthesis Constraints

The EtherCAT IP Core contains true dual-port memory. A simultaneous read and write access to the same memory address is avoided by SyncManagers. To prevent Quartus from adding pass-through logic to the memory which leads to additional resource requirements and timing restrictions, the Analysis & Synthesis option “Add Pass-Through Logic to Inferred RAMs” should be set to “Off”.

Turn on Analysis & Synthesis option: Auto RAM Replacement, otherwise the RAM inside the IP Core will be implemented with individual registers.

The following table contains basic IP Core constraints.

Table 61: EtherCAT IP Core constraints

Signal	Requirement	Value	Clock reference	Description
CLK25	period	40 ns		Reference clock (25 MHz)
CLK50	a) period b) phase shift	a) 20 ns b) 0 ns	CLK25	Derived clock (50 MHz). Phase shift is rising edge to rising edge.
CLK100	a) period b) phase shift	a) 10 ns b) 0 ns	CLK25	Derived clock (100 MHz). Phase shift is rising edge to rising edge.
nRESET	Ignore timing			nRESET is asynchronous to any clock
MCLK	min. period	400 ns		IEEE802.3 requirement (2.5 MHz)
MDIO	a) setup b) hold at PHY input	a) 10 ns b) 10 ns	MCLK (rising edge)	MDIO is changed with falling edge of MCLK, max. output skew of MCLK and MDIO is 190 ns. Constraining is usually not required. IEEE802.3 requirement.
MII_RX_CLK0-2	period	40 ns		MII receive reference clock (25 MHz). IEEE802.3 requirement.
MII_RX_DATA0-2[3:0] MII_RX_DV0-2 MII_RX_ERR0-2	a) setup b) hold	a) 10 ns b) 10 ns	MII_RX_CLK0-2 (rising edge)	IEEE802.3 requirement
MII_TX_CLK0-2	period	40 ns		MII transmit reference clock (25 MHz). Only used for automatic TX Shift compensation. IEEE802.3 requirement.
MII_TX_DATA0-2[3:0] MII_TX_ENA0-2	Clock-to-Pin a) min b) max	a) 0 ns b) 25 ns	TX_CLK0-2 from PHY (rising edge)	IEEE802.3 requirement
	Clock-to-Pin a) min b) max	a) 0 ns b) 10 ns	CLK25 (rising edge)	Incomplete alternative to IEEE802.3 requirement, keeps margin if TX Shift has been determined and compensated. Refer to section III for details.
PROM_CLK	period	App. dep.		I²C clock. Actual ESC output clock is 6.72 µs (≈ 150 kHz). Min. 2.5µs (400 Khz) for example I²C EEPROM chip.
PROM_DATA	a) setup b) hold	a) 250 ns b) 0 ns	PROM_CLK a) rising edge b) falling edge	PROM_DATA is changed in the middle of the low phase of PROM_CLOCK, i.e., max. output skew of PROM_CLK/PROM_DATA is 1.43 µs. Constraining is usually not required. Example I²C EEPROM chip requirement.
RMII_RX_DATA0/1[1:0] RMII_RX_DV0/1 RMII_RX_ERR0/1 RMII_TX_DATA0/1[1:0] RMII_TX_ENA0/1	a) setup b) hold	a) 4 ns b) 2 ns	CLK50 (rising edge)	RMII specification requirement
Other signals, especially PDI signals	application dependent			

Example Design Constraints File (SDC)

```
## Constraints for EL9800_DIGI_EP3C25
#
set_time_format -unit ns -decimal_places 3

# Clocks defintion:
create_clock -period 40 -name MII_RX_CLK [get_ports MII_RX_CLK* ]
create_clock -period 40 -name REF_CLK [get_ports REF_CLK ]
create_clock -period 80 -name DIGI_CLK [get_pins
   {ETHERCAT_INST|EtherCAT_IPCore_inst|\PDI_DIGI_INST:PDI_INST|NRESET_PDI_SELECT_REG|q}]

derive_pll_clocks
derive_clock_uncertainty

# constraining MII ports
set_input_delay -clock { MII_RX_CLK } 20 [get_ports MII_RX_DATA*]
set_input_delay -clock { MII_RX_CLK } 20 [get_ports MII_RX_DV*]
set_input_delay -clock { MII_RX_CLK } 20 [get_ports MII_RX_ERR*]
set_input_delay -clock { PLL_INST|altpll_component|auto_generated|pll1|clk[0] } 5 [get_ports
   nMII_LINK* ]
set_input_delay -clock { PLL_INST|altpll_component|auto_generated|pll1|clk[1] } 5 [get_ports
   MII_TX_CLK*]

set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_ports {MII_TX_ENA* MII_TX_DATA*}] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_ports {MII_TX_ENA* MII_TX_DATA*}] 8
set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {LINK_ACT*}] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {LINK_ACT*}] 15

set_false_path -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_clocks {MII_RX_CLK}]
set_false_path -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_clocks {MII_RX_CLK}]
set_false_path -from [get_clocks {MII_RX_CLK}] -to [get_clocks
   {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}]
set_false_path -from [get_clocks {MII_RX_CLK}] -to [get_clocks
   {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}]

# constraining logical inputs
set_input_delay -clock { PLL_INST|altpll_component|auto_generated|pll1|clk[0] } 5 [get_ports
   {PORT_A* PORT_B* PORT_F*} ]
set_input_delay -clock { PLL_INST|altpll_component|auto_generated|pll1|clk[0] } 5 [get_ports
   {PROM_DATA PROM_SIZE}]
set_input_delay -clock { PLL_INST|altpll_component|auto_generated|pll1|clk[0] } 5 [get_ports
   MDIO* ]

set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_ports {PORT_C* PORT_D* PORT_E*}] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_ports {PORT_C* PORT_D* PORT_E*}] 8
set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {PROM_CLK PROM_DATA}] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {PROM_CLK PROM_DATA}] 15
set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {MDIO MCLK}] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports {MDIO MCLK}] 15
set_min_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports LED_RUN] 0
set_max_delay -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_ports LED_RUN] 15

#false paths
set_false_path -from {nRESET} -to
   {PLL:PLL_INST|altpll:altpll_component|altpll_bbc1:auto_generated|pll_lock_sync}
set_false_path -from
   {PLL:PLL_INST|altpll:altpll_component|altpll_bbc1:auto_generated|pll_lock_sync} -to
   [get_ports {PROM_DATA PROM_CLK}]
set_false_path -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}] -to
   [get_clocks {DIGI_CLK}]
set_false_path -from [get_clocks {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}] -to
   [get_clocks {DIGI_CLK}]
```

```
set_false_path -from [get_clocks {DIGI_CLK}] -to [get_clocks
   {PLL_INST|altpll_component|auto_generated|pll1|clk[0]}]
set_false_path -from [get_clocks {DIGI_CLK}] -to [get_clocks
   {PLL_INST|altpll_component|auto_generated|pll1|clk[1]}]
```

15 Appendix

15.1 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

15.1.1 Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on her internet pages: http://www.beckhoff.com

You will also find further documentation for Beckhoff components there.

15.2 Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG
Huelshorstweg 20
33415 Verl
Germany

phone: + 49 (0) 5246/963-0

fax: + 49 (0) 5246/963-198

e-mail: info@beckhoff.com

web: www.beckhoff.com

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- world-wide support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

hotline: + 49 (0) 5246/963-157

fax: + 49 (0) 5246/963-9157

e-mail: support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

hotline: + 49 (0) 5246/963-460

fax: + 49 (0) 5246/963-479

e-mail: service@beckhoff.com