

Hardware Data Sheet

ET1810 / ET1812

**EtherCAT® Slave Controller IP Core
for Altera® FPGAs**
IP Core Release 2.4.0

**Section I –
EtherCAT Slave Controller Technology**

**Section II –
EtherCAT Slave Controller Register Description**

**Section III –
EtherCAT IP Core Description: Installation, Configuration,
Design flow, Interface specification**

**Version 1.0
Date: 2011-03-15**

BECKHOFF

LEGAL NOTICE

Trademarks

Beckhoff[®], TwinCAT[®], EtherCAT[®], Safety over EtherCAT[®], TwinSAFE[®] and XFC[®] are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following German patent applications and patents: DE10304637, DE102004044764, DE102005009224, DE102007017835 with corresponding applications or registrations in various other countries.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

© Beckhoff Automation GmbH 03/2011.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

DOCUMENT HISTORY

Version	Comment
1.0	Initial release (Section I V2.0, Section II V2.5, Section III V1.0)

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.

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

Additional documentation and utilities like application notes and Excel sheets for ET1100/ET1200 pinout configuration can be found at the Beckhoff homepage (<http://www.beckhoff.com> – Download/Documentation/EtherCAT development products).

Hardware Data Sheet

EtherCAT® Slave Controller

Section I – Technology

**EtherCAT Protocol, Ethernet and EBUS Physical Layer,
EtherCAT Processing Unit, FMMU, SyncManager,
SII EEPROM, Distributed Clocks, etc.**

**Version 2.0
Date: 2011-03-15**

BECKHOFF

Trademarks

Beckhoff[®], TwinCAT[®], EtherCAT[®], Safety over EtherCAT[®], TwinSAFE[®] and XFC[®] are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following German patent applications and patents: DE10304637, DE102004044764, DE102005009224, DE102007017835 with corresponding applications or registrations in various other countries.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

© Beckhoff Automation GmbH 03/2011.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

DOCUMENT HISTORY

Version	Comment
1.0	Initial release
1.1	<ul style="list-style-type: none"> • Chapter Interrupts – AL Event Request: corrected AL Event Mask register address to 0x0204:0x0207 • EtherCAT Datagram: Circulating Frame bit has position 14 (not 13) • PHY addressing configuration changed • Loop control: a port using Auto close mode is automatically opened if a valid Ethernet frame is received at this port • EEPROM read/write/reload example: steps 1 and 2 swapped • EEPROM: Configured Station Alias (0x0012:0x0013) is only taken over at first EEPROM load after power-on or reset • SyncManager: Watchdog trigger and interrupt generation in mailbox mode with single byte buffers requires alternating write and read accesses for some ESCs, thus buffered mode is required for Digital I/O watchdog trigger generation • National Semiconductor DP83849I Ethernet PHY deprecated because of large link loss reaction time and delay • Added distinction between permanent ports and Bridge port (frame processing) • Added PDI chapter • PDI and DC Sync/Latch signals are high impedance until the SII EEPROM is successfully loaded • Editorial changes
1.2	<ul style="list-style-type: none"> • PHY address configuration revised. Refer to Section III for ESC supported configurations • Added Ethernet Link detection chapter • Added MI Link Detection and Configuration, link detection descriptions updated • Added EEPROM Emulation for EtherCAT IP Core • Added General Purpose Input chapter • Corrected minimum datagram sizes in EtherCAT header figure • Editorial changes
1.2.1	<ul style="list-style-type: none"> • Chapter 5.1.1: incompatible PHYs in footnote 1 deleted
1.3	<ul style="list-style-type: none"> • Added advisory for unused MII/RMII/EBUS ports • Ethernet PHY requirements revised: e.g., configuration by strapping options, recommendations enhanced. Footnote about compatible PHYs removed, information has moved to the EtherCAT Slave Controller application note "PHY Selection Guide". • Frame Error detection chapter enhanced • FIFO size reduction chapter enhanced • EBUS enhanced link detection chapter enhanced • Ethernet PHY link loss reaction time must be faster than 15 µs, otherwise use Enhanced link detection • Enhanced link detection description corrected. Enhanced link detection does not remain active if it is disabled by EEPROM and EBUS handshake frames are received • ARMW/FRWM commands increase the working counter by 1 • Editorial changes
1.4	<ul style="list-style-type: none"> • Update to EtherCAT IP Core Release 2.1.0/2.01a • Added restriction to enhanced link configuration: RX_ER has to be asserted outside of frames (IEEE802 optional feature) • ESC power-on sequence for IP Core corrected • Removed footnote on t_{Diff} figures, refer to Section III for actual figures • Editorial changes

Version	Comment
1.5	<ul style="list-style-type: none"> • EEPROM Read/Write/Reload example: corrected register addresses • Updated/clarified PHY requirements, PHY link loss reaction time is mandatory • Enhanced Link Detection can be configured port-wise depending on ESC • Added DC Activation and DC Activation State features for some ESCs • ESC10 removed • Editorial changes
1.6	<ul style="list-style-type: none"> • Fill reserved EEPROM words of the ESC Configuration Area with 0 • Interrupt chapter: example for proper interrupt handling added • Use Position Addressing only for bus scanning at startup and to detect newly attached devices • System Time PDI controlled: detailed description added • Added MII back-to-back connection example • Renamed Err(x) LED to PERR(x) • Editorial changes
1.7	<ul style="list-style-type: none"> • Link status description enhanced • Clarifications for DC System Time and reference between clocks and registers • Chapter on avoiding unconnected Port 0 configurations added • Direct ESC to standard Ethernet MAC MII connection added • MI link detection and configuration must not be used without LINK_MII signals • Added criteria for detecting when DC synchronization is established • SII EEPROM interface is a point-to-point connection • PHY requirements: PHY startup should not rely on MDC clocking, ESD tolerance and baseline wander compensation recommendations added • Editorial changes
1.8	<ul style="list-style-type: none"> • Update to EtherCAT IP Core Release 2.3.0/2.03a • EEPROM acknowledge error (0x0502[13]) can also occur for a read access • ERR and STATE LED updated • Editorial changes
1.9	<ul style="list-style-type: none"> • EtherCAT state machine: additional AL status codes defined • EtherCAT protocol: LRD/LRW read data depends on bit mask • Updated EBUS Enhanced Link Detection • Updated FMMU description • Loop control description updated • EtherCAT frame format (VLAN tag) description enhanced • Update to EtherCAT IP Core Release 2.3.2/2.03c
2.0	<ul style="list-style-type: none"> • Update to EtherCAT IP Core Release 2.4.0/2.04a • SII/ESI denotation now consistent with ETG • Updated AL Status codes • Editorial changes

CONTENTS

1	EtherCAT Slave Controller Overview	1
1.1	EtherCAT Slave Controller Function Blocks	2
1.2	Further Reading on EtherCAT and ESCs	3
1.3	Scope of Section I	3
2	EtherCAT Protocol	4
2.1	EtherCAT Header	4
2.2	EtherCAT Datagram	5
2.3	EtherCAT Addressing Modes	6
2.3.1	Device Addressing	7
2.3.2	Logical Addressing	7
2.4	Working Counter	8
2.5	EtherCAT Command Types	9
3	Frame Processing	11
3.1	Loop Control and Loop State	11
3.2	Frame Processing Order	14
3.3	Permanent Ports and Bridge Port	15
3.4	Shadow Buffer for Register Write Operations	15
3.5	Circulating Frames	15
3.5.1	Unconnected Port 0	16
3.6	Non-EtherCAT Protocols	16
3.7	Special Functions of Port 0	16
4	Physical Layer Common Features	17
4.1	Link Status	17
4.2	Selecting Standard/Enhanced Link Detection	18
4.3	FIFO Size Reduction	19
4.4	Frame Error Detection	19
5	Ethernet Physical Layer	20
5.1	Requirements to Ethernet PHYs	20
5.2	MII Interface Signals	22
5.3	RMII Interface Signals	24
5.4	Link Detection	25
5.4.1	LINK_MII Signal	25
5.4.2	MI Link Detection and Configuration	25
5.5	Standard and Enhanced MII Link Detection	26
5.6	MII Management Interface (MI)	26
5.6.1	PHY Addressing/PHY Address Offset	27
5.6.2	Logical Interface	28
5.6.2.1	MI read/write example	28
5.6.2.2	MI Interface Assignment to ECAT/PDI	28

5.6.3	MI Protocol	29
5.6.4	Timing specifications	29
5.7	Ethernet Termination and Grounding Recommendation	30
5.8	Ethernet Connector (RJ45 / M12)	31
5.9	Back-to-Back MII Connection	32
5.9.1	ESC to ESC Connection	32
5.9.2	ESC to Standard Ethernet MAC	33
6	EBUS/LVDS Physical Layer	34
6.1	Interface	34
6.2	EBUS Protocol	35
6.3	Timing Characteristics	35
6.4	Standard EBUS Link Detection	36
6.5	Enhanced EBUS Link Detection	36
6.6	EBUS RX Errors	37
6.7	EBUS Low Jitter	37
6.8	EBUS Connection	37
7	FMMU	38
8	SyncManager	40
8.1	Buffered Mode	41
8.2	Mailbox Mode	42
8.2.1	Mailbox Communication Protocols	42
8.3	Interrupt and Watchdog Trigger Generation, Latch Event Generation	43
8.4	Single Byte Buffer Length / Watchdog Trigger for Digital Output PDI	43
8.5	Repeating Mailbox Communication	44
8.6	SyncManager Deactivation by the PDI	44
9	Distributed Clocks	45
9.1	Clock Synchronization	45
9.1.1	Clock Synchronization Process	47
9.1.2	Propagation Delay Measurement	48
9.1.2.1	Propagation Delay Measurement Example	48
9.1.3	Offset Compensation	52
9.1.4	Drift Compensation	53
9.1.5	Reference between DC Registers/Functions and Clocks	54
9.1.6	When is Synchronization established?	55
9.1.7	Clock Synchronization Initialization Example	55
9.2	SyncSignals and LatchSignals	56
9.2.1	Interface	56
9.2.2	Configuration	56
9.2.3	SyncSignal Generation	57
9.2.3.1	Cyclic Generation	58
9.2.3.2	Single Shot Mode	58

9.2.3.3	Cyclic Acknowledge Mode	58
9.2.3.4	Single Shot Acknowledge Mode	58
9.2.3.5	SYNC1 Generation	59
9.2.3.6	SyncSignal Initialization Example	60
9.2.4	LatchSignals	60
9.2.4.1	Single Event Mode	61
9.2.4.2	Continuous Mode	61
9.2.4.3	SyncManager Event	61
9.2.5	ECAT or PDI Control	61
9.3	System Time PDI Controlled	62
9.4	Communication Timing	64
10	EtherCAT State Machine	66
10.1	EtherCAT State Machine Registers	67
10.1.1	AL Control and AL Status Register	67
10.1.2	Device Emulation	67
10.1.3	Error Indication and AL Status Code Register	68
10.2	State Machine Services	70
11	SII EEPROM	71
11.1	SII EEPROM Content	72
11.2	SII EEPROM Logical Interface	73
11.2.1	SII EEPROM Errors	74
11.2.1.1	Missing Acknowledge	75
11.2.2	SII EEPROM Interface Assignment to ECAT/PDI	75
11.2.3	Read/Write/Reload Example	76
11.2.4	EEPROM Emulation	76
11.3	SII EEPROM Electrical Interface (I^2C)	77
11.3.1	Word Addressing	77
11.3.2	EEPROM Size	77
11.3.3	I^2C Access Protocol	78
11.3.3.1	Write Access	78
11.3.3.2	Read Access	79
11.3.4	Timing specifications	79
12	Interrupts	81
12.1	AL Event Request (PDI Interrupt)	81
12.2	ECAT Event Request (ECAT Interrupt)	82
12.3	Clearing Interrupts Accidentally	82
13	Watchdogs	83
13.1	Process Data Watchdog	83
13.2	PDI Watchdog	83
14	Error Counters	84
14.1	Frame error detection	85

14.2	Errors and Forwarded Errors	85
15	LED Signals (Indicators)	86
15.1	RUN LED	86
15.2	ERR LED	86
15.3	STATE LED and STATE_RUN LED Signal	86
15.4	LINKACT LED	86
15.5	Port Error LED (PERR)	87
16	Process Data Interface (PDI)	88
16.1	PDI Selection and Configuration	88
16.2	General Purpose I/O	89
16.2.1	General Purpose Inputs	89
16.2.2	General Purpose Output	89
17	Additional Information	90
17.1	ESC Clock Source	90
17.2	Power-on Sequence	90
17.3	Write Protection	91
17.3.1	Register Write Protection	91
17.3.2	ESC Write Protection	91
17.4	ESC Reset	91
18	Appendix	92
18.1	Support and Service	92
18.1.1	Beckhoff's branch offices and representatives	92
18.2	Beckhoff Headquarters	92

TABLES

Table 1: ESC Main Features	1
Table 2: EtherCAT Frame Header.....	5
Table 3: EtherCAT Datagram	6
Table 4: EtherCAT Addressing Modes	6
Table 5: Working Counter Increment	8
Table 6: EtherCAT Command Types	9
Table 7: EtherCAT Command Details	10
Table 8: Registers for Loop Control and Loop/Link Status	13
Table 9: Frame Processing Order	14
Table 10: Link Status Description.....	17
Table 11: Registers for Enhanced Link Detection	18
Table 12: MII Interface signals	22
Table 13: Special/Unused MII Interface signals	23
Table 14: RMII Interface signals.....	24
Table 15: Registers used for Ethernet Link Detection.....	25
Table 16: PHY Address configuration matches PHY address settings.....	27
Table 17: PHY Address configuration does not match actual PHY address settings	27
Table 18: MII Management Interface Register Overview	28
Table 19: MII Management Interface timing characteristics.....	29
Table 20: Signals used for Fast Ethernet.....	31
Table 21: EBUS Interface signals	34
Table 22: EBUS timing characteristics	35
Table 23: Example FMMU Configuration	38
Table 24: SyncManager Register overview.....	40
Table 25: EtherCAT Mailbox Header	43
Table 26: Registers for Propagation Delay Measurement	48
Table 27: Parameters for Propagation Delay Calculation	50
Table 28: Registers for Offset Compensation	52
Table 29: Registers for Drift Compensation	54
Table 30: Reference between DC Registers/Functions and Clocks	54
Table 31: Distributed Clocks signals	56
Table 32: SyncSignal Generation Mode Selection.....	57
Table 33: Registers for SyncSignal Generation	58
Table 34: Registers for Latch Input Events	61
Table 35: Registers for the EtherCAT State Machine	67
Table 36: AL Control and AL Status Register Values	67
Table 37: AL Status Codes (0x0134:0x0135)	68
Table 38: State Machine Services.....	70
Table 39: ESC Configuration Area	72
Table 40: SII EEPROM Content Excerpt.....	73
Table 41: SII EEPROM Interface Register Overview	73
Table 42: SII EEPROM Interface Errors.....	74
Table 43: I ² C EEPROM signals	77
Table 44: EEPROM Size	77
Table 45: I ² C Control Byte	78
Table 46: I ² C Write Access.....	78
Table 47: I ² C Read Access.....	79
Table 48: EEPROM timing characteristics	79
Table 49: Registers for AL Event Request Configuration	81
Table 50: Registers for ECAT Event Request Configuration	82
Table 51: Registers for Watchdogs	83
Table 52: Error Counter Overview.....	84
Table 53: Errors Detected by Physical Layer, Auto-Forwarder, and EtherCAT Processing Unit	85
Table 54: RUN LED States.....	86
Table 55: LINKACT LED States	86
Table 56: Available PDIs depending on ESC	88
Table 57: ESC Power-On Sequence.....	90
Table 58: Registers for Write Protection	91

FIGURES

Figure 1: EtherCAT Slave Controller Block Diagram	1
Figure 2: Ethernet Frame with EtherCAT Data	4
Figure 3: EtherCAT Datagram.....	5
Figure 4: Auto close loop state transitions	12
Figure 5: Frame Processing	14
Figure 6: Circulating Frames	15
Figure 7: All frames are dropped because of Circulating Frame Prevention	16
Figure 8: MII Interface signals	22
Figure 9: RMII Interface signals.....	24
Figure 10: Write access	29
Figure 11: Read access.....	29
Figure 12: Termination and Grounding Recommendation	30
Figure 13: RJ45 Connector	31
Figure 14: M12 D-code Connector	31
Figure 15: Back-to-Back MII Connection (two ESCs)	32
Figure 16: Back-to-Back MII Connection (ESC and standard MAC).....	33
Figure 17: EBUS Interface Signals.....	34
Figure 18: EBUS Protocol	35
Figure 19: Example EtherCAT Network	36
Figure 20: EBUS Connection	37
Figure 21: FMMU Mapping Principle	38
Figure 22: FMMU Mapping Example.....	39
Figure 23: SyncManager Buffer allocation	41
Figure 24: SyncManager Buffered Mode Interaction.....	41
Figure 25: SyncManager Mailbox Interaction	42
Figure 26: EtherCAT Mailbox Header (for all Types)	43
Figure 27: Handling of Write/Read Toggle with Read Mailbox	44
Figure 28: Propagation Delay, Offset, and Drift Compensation	47
Figure 29: Propagation Delay Calculation	49
Figure 30: Distributed Clocks signals	56
Figure 31: SyncSignal Generation Modes.....	57
Figure 32: SYNC0/1 Cycle Time Examples	59
Figure 33: System Time PDI Controlled with three steps	62
Figure 34: System Time PDI Controlled with two steps	63
Figure 35: DC Timing Signals in relation to Communication.....	64
Figure 36: EtherCAT State Machine	66
Figure 37: SII EEPROM Layout.....	71
Figure 38: I ² C EEPROM signals.....	77
Figure 39: Write access (1 address byte, up to 16 kBit EEPROMs)	79
Figure 40: Write access (2 address bytes, 32 kBit - 4 MBit EEPROMs).....	80
Figure 41: Read access (1 address byte, up to 16 kBit EEPROMs).....	80
Figure 42: PDI Interrupt Masking and interrupt signals	81
Figure 43: ECAT Interrupt Masking	82

ABBREVIATIONS

μ C	Microcontroller
ADR	Address
ADS	Automation Device Specification (Beckhoff)
AL	Application Layer
APRD	Auto Increment Physical Read
APWR	Auto Increment Physical Write
APRW	Auto Increment Physical ReadWrite
ARMW	Auto Increment Physical Read Multiple Write
AoE	ADS over EtherCAT
ASIC	Application Specific Integrated Chip
Auto Crossover	Automatic detection of whether or not the send and receive lines are crossed.
Auto Negotiation	Automatic negotiation of transmission speeds between two stations.
Avalon [®]	On-chip bus for Altera [®] FPGAs
Big Endian	Data format (also Motorola format). The more significant byte is transferred first when a word is transferred. However, for EtherCAT the least significant bit is the first on the wire.
BOOT	BOOT state of EtherCAT state machine
Boundary Clock	A station that is synchronized by another station and then passes this information on.
Bridge	A term for switches used in standards. Bridges are devices that pass on messages based on address information.
Broadcast	An unacknowledged transmission to an unspecified number of receivers.
BRD	Broadcast Read
BWR	Broadcast Write
BRW	Broadcast ReadWrite
Cat	Category – classification for cables that is also used in Ethernet. Cat 5 is the minimum required category for EtherCAT. However, Cat 6 and Cat 7 cables are available.
CoE	CANopen over EtherCAT
Communication Stack	A communication software package that is generally divided into successive layers, which is why it is referred to as a stack.
Confirmed	Means that the initiator of a service receives a response.
CRC	Cyclic Redundancy Check, used for FCS
Cut Through	Procedure for cutting directly through an Ethernet frame by a switch before the complete message is received.
Cycle	Cycle in which data is to be exchanged in a system operating on a periodical basis.

DC	Distributed Clocks Mechanism to synchronize EtherCAT slaves and master
Delay	Delays can be caused by run-times during transfer or internal delays of a network component.
Dest Addr	Destination address of a message (the destination can be an individual network station or a group (multicast)).
DHCP	Dynamic Host Configuration Protocol, used to assign IP addresses (and other important startup parameter in the Internet context).
DL	Data Link Layer, also known as Layer 2. EtherCAT uses the Data Link Layer of Ethernet, which is standardized as IEEE 802.3.
DNS	Domain Name Service, a protocol for domain name to IP addresses resolution.
EBUS	Based on LVDS (Low Voltage Differential Signaling) standard specified in ANSI/TIA/EIA-644-1995
ECAT	EtherCAT
EEPROM	Electrically Erasable Programmable Read Only Memory. Non-volatile memory used to store EtherCAT Slave Information (ESI). Connected to the SII.
EMC	Electromagnetic Compatibility, describes the robustness of a device with regard to electrical interference from the environment.
EMI	Electromagnetic Interference
Engineering	Here: All applications required to configure and program a machine.
EoE	Ethernet over EtherCAT
EOF	End of Frame
ERR	Error indicator for AL state
Err(x)	Physical Layer RX Error LED for debugging purposes
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information, stored in SII EEPROM
ESM	EtherCAT State Machine
ETG	EtherCAT Technology Group (http://www.ethercat.org)
EtherCAT	Real-time Standard for Industrial Ethernet Control Automation Technology (Ethernet for Control Automation Technology)
EtherType	Identification of an Ethernet frame with a 16-bit number assigned by IEEE. For example, IP uses EtherType 0x0800 (hexadecimal) and the EtherCAT protocol uses 0x88A4.
EPU	EtherCAT Processing Unit. The logic core of an ESC containing e.g. registers, memory, and processing elements.
Fast Ethernet	Ethernet with a transmission speed of 100 Mbit/s.
FCC	Federal Communications Commission

FCS	Frame Check Sequence
FIFO	First In First Out
Firewall	Routers or other network component that acts as a gateway to the Internet and enables protection from unauthorized access.
FMMU	Fieldbus Memory Management Unit
FoE	File access over EtherCAT
Follow Up	Message that follows Sync and indicates when the Sync frame was sent from the last node (defined in IEEE 1588).
FPGA	Field Programmable Gate Array
FPRD	Configured Address Physical Read
FPWR	Configured Address Physical Write
FPRW	Configured Address Physical ReadWrite
FRMW	Configured Address Physical Read Multiple Write
Frame	See PDU
FTP	File Transfer Protocol
Get	Access method used by a client to read data from a device.
GND	Ground
GPI	General Purpose Input
GPO	General Purpose Output
HW	Hardware
I ² C	Inter-Integrated Circuit, serial bus used for SII EEPROM connection
ICMP	Internet Control Message Protocol: Mechanisms for signaling IP errors.
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
INIT	INIT state of EtherCAT state machine
Interval	Time span
IP	Internet Protocol: Ensures transfer of data on the Internet from end node to end node.
	Intellectual Property
IRQ	Interrupt Request
ISO	International Standard Organization
ISO/OSI Model	ISO Open Systems Interconnection Basic Reference Model (ISO 7498): describes the division of communication into 7 layers.
IT	Information Technology: Devices and methods required for computer-aided information processing.
LatchSignal	Signal for Distributed Clocks time stamping

LED	Light Emitting Diode, used as an indicator
Link/Act	Link/Activity Indicator (LED)
Little Endian	Data format (also Intel format). The less significant byte is transferred first when a word is transferred. With EtherCAT, the least significant bit is the first on the wire.
LLDP	Lower Layer Discovery Protocol – provides the basis for topology discovery and configuration definition (see IEEE802.1ab)
LRD	Logical Read
LWR	Logical Write
LRW	Logical ReadWrite
LVDS	Low Voltage Differential Signaling
M12	Connector used for industrial Ethernet
MAC	Media Access Control: Specifies station access to a communication medium. With full duplex Ethernet, any station can send data at any time; the order of access and the response to overload are defined at the network component level (switches).
MAC Address	Media Access Control Address: Also known as Ethernet address; used to identify an Ethernet node. The Ethernet address is 6 bytes long and is assigned by the IEEE.
Mandatory Services	Mandatory services, parameters, objects, or attributes. These must be implemented by every station.
MBX	Mailbox
MDI	Media Dependant Interface: Use of connector Pins and Signaling (PC side)
MDI-X	Media Dependant Interface (crossed): Use of connector Pins and Signaling with crossed lines (Switch/hub side)
MI	(PHY) Management Interface
MII	Media Independent Interface: Standardized interface between the Ethernet MAC and PHY.
Multicast	Transmission to multiple destination stations with a frame – generally uses a special address.
NOP	No Operation
NVRAM	Non-volatile random access memory, e.g. EEPROM or Flash.
Octet	Term from IEC 61158 – one octet comprises exactly 8 bits.
OP	Operational state of EtherCAT state machine
OPB	On-Chip Peripheral Bus
Optional Service	Optional services can be fulfilled by a PROFINET station in addition to the mandatory services.
OSI	Open System Interconnect

OUI	Organizationally Unique Identifier – are the first 3 Bytes of a Ethernet-Address, that will be assign to companies or organizations and can be used for protocol identifiers as well (e.g. LLDP)
PDI	Process Data Interface or Physical Device Interface: an interface that allows access to ESC from the process side.
PDO	Process Data Object
PDU	Protocol Data Unit: Contains protocol information (Src Addr, Dest Addr, Checksum and service parameter information) transferred from a protocol instance of transparent data to a subordinate level (the lower level contains the information being transferred).
PE	Protection Earth
PHY	Physical layer device that converts data from the Ethernet controller to electric or optical signals.
Ping	Frame that verifies whether the partner device is still available.
PLB	Processor Local Bus
PLL	Phase Locked Loop
PREOP	Pre-Operational state of EtherCAT state machine
Priority Tagging	Priority field inserted in an Ethernet frame.
Protocol	Rules for sequences – here, also the sequences (defined in state machines) and frame structures (described in encoding) of communication processes.
Provider	Device that sends data to other consumers in the form of a broadcast message.
PTP	Precision Time Protocol in accordance with IEEE 1588: Precise time synchronization procedures.
PTP Master	Indicates time in a segment.
PTP Slave	Station synchronized by a PTP master.
Quad Cable	Cable type in which the two cable pairs are twisted together. This strengthens the electromagnetic resistance.
RAM	Random Access Memory. ESC have User RAM and Process Data RAM.
Read	Service enabling read access to an I/O device.
Real-Time	Real-time capability of a system to perform a task within a specific time.
Request	Call of a service in the sender/client.
Response	Response to a service on the client side.
RJ45	FCC Registered Jack, standard Ethernet connector (8P8C)
RMII	Reduced Media Independent Interface
Router	Network component acting as a gateway based on the interpretation of the IP address.

RSTP	Rapid Spanning Tree Protocol: Prevents packet from looping infinitely between switches; RSTP is specified in IEEE 802.1 D (Edition 2004)
RT	Real-time. Name for a real-time protocol that can be run in Ethernet controllers without special support.
RTC	Real-time Clock chip of PCs
RT Frames	EtherCAT Messages with EtherType 0x88A4.
RX	Receive
RXPDO	Receive PDO, i.e. Process Data that will be received by ESC20
RUN	RUN indicator (LED) for application state
SAFEOP	Safe-Operational state of EtherCAT state machine
Safety	Safety function, implemented by an electric, electronic programmable fail-safe system that maintains the equipment in a safe state, even during certain critical external events.
Schedule	Determines what should be transferred and when.
Services	Interaction between two components to fulfill a specific task.
Set	Access method used by a client to write data to a server.
SII	Slave Information Interface
SM	SyncManager
SNMP	Simple Network Management Protocol: SNMP is the standard Internet protocol for management and diagnostics of network components (see also RFC 1157 and RFC 1156 at www.ietf.org).
SoE	Servo Profile over EtherCAT
SOF	Start of Frame Ethernet SOF delimiter at the end of the preamble of Ethernet frames
SPI	Serial Peripheral Interface
Src Addr	Source Address: Source address of a message.
Store and Forward	Currently the common operating mode in switches. Frames are first received in their entirety, the addresses are evaluated, and then they are forwarded. This result in considerable delays, but guarantees that defective frames are not forwarded, causing an unnecessary increase in the bus load.
STP	Shielded Twisted Pair: Shielded cable with at least 2 core pairs to be used as the standard EtherCAT cable.
Subnet Mask	Divides the IP address into two parts: a subnet address (in an area separated from the rest by routers) and a network address.
Switch	Also known as Bridge. Active network component to connect different EtherCAT participants with each other. A switch only forwards the frames to the addressed participants.
SyncManager	ESC unit for coordinated data exchange between master and slave µController

SyncSignal	Signal generated by the Distributed Clocks unit
TCP	Transmission Control Protocol: Higher-level IP protocol that ensures secure data exchange and flow control.
TX	Transmit
TXPDO	Transmit PDO, i.e. Process Data that will be transmitted by ESC20
UDP	User Datagram Protocol: Non-secure multicast/broadcast frame.
UTP	Unshielded Twisted Pair: Unshielded cable with at least 2 core pairs are not recommended for industrial purpose but are commonly used in areas with low electro-magnetic interference.
VLAN	Virtual LAN
VoE	Vendor specific profile over EtherCAT
WD	Watchdog
WKC	Working Counter
XML	Extensible Markup Language: Standardized definition language that can be interpreted by nearly all parsers.
XML Parser	Program for checking XML schemas.

1 EtherCAT Slave Controller Overview

An EtherCAT Slave Controller (ESC) takes care of the EtherCAT communication as an interface between the EtherCAT fieldbus and the slave application. This document covers the following Beckhoff ESCs: ASIC implementations (ET1100, ET1200), functionally fixed binary configurations for FPGAs (ESC20), and configurable IP Cores for FPGAs (ET1810/ET1815).

Table 1: ESC Main Features

Feature	ET1200	ET1100	IP Core	ESC20
Ports	2-3 (each EBUS/MII, max. 1xMII)	2-4 (each EBUS/MII)	1-3 MII or 1-2 RMII	2 MII
FMMUs	3	8	0-8	4
SyncManagers	4	8	0-8	4
RAM [Kbyte]	1	8	1-60	4
Distributed Clocks	64 bit	64 bit	32/64 bit	32 bit
Process Data Interfaces				
Digital I/O	16 bit	32 bit	8-32 bit	32 bit
SPI Slave	Yes	Yes	Yes	Yes
8/16 bit µController	-	Async/Sync	Async	Async
On-chip bus	-	-	Avalon or PLB/OPB	-

The general functionality of an ESC is shown in Figure 1:

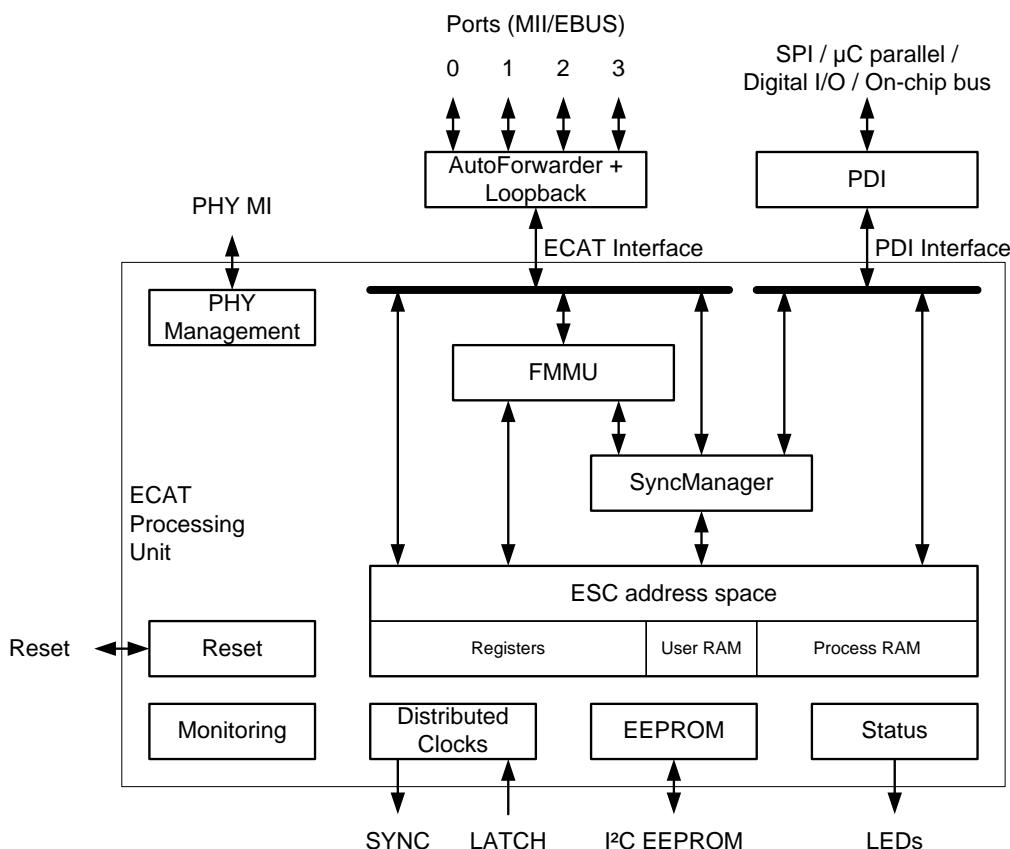


Figure 1: EtherCAT Slave Controller Block Diagram

1.1 EtherCAT Slave Controller Function Blocks

EtherCAT Interfaces (Ethernet/EBUS)

The EtherCAT interfaces or ports connect the ESC to other EtherCAT slaves and the master. The MAC layer is integral part of the ESC. The physical layer may be Ethernet or EBUS. The physical layer for EBUS is fully integrated into the ASICs. For Ethernet ports, external Ethernet PHYs connect to the MII/RMII ports of the ESC. Transmission speed for EtherCAT is fixed to 100 Mbit/s with Full Duplex communication. Link state and communication status are reported to the Monitoring device. EtherCAT slaves support 2-4 ports, the logical ports are numbered 0-1-2-3, formerly they were denoted by A-B-C-D.

EtherCAT Processing Unit

The EtherCAT Processing Unit (EPU) receives, analyses and processes the EtherCAT data stream. It is logically located between port 0 and port 3. The main purpose of the EtherCAT Processing unit is to enable and coordinate access to the internal registers and the memory space of the ESC, which can be addressed both from the EtherCAT master and from the local application via the PDI. Data exchange between master and slave application is comparable to a dual-ported memory (process memory), enhanced by special functions e.g. for consistency checking (SyncManager) and data mapping (FMMU). The EtherCAT Processing Units contains the main function blocks of EtherCAT slaves besides Auto-Forwarding, Loop-back function, and PDI.

Auto-Forwarder

The Auto-Forwarder receives the Ethernet frames, performs frame checking and forwards it to the Loop-back function. Time stamps of received frames are generated by the Auto-Forwarder.

Loop-back function

The Loop-back function forwards Ethernet frames to the next logical port if there is either no link at a port, or if the port is not available, or if the loop is closed for that port. The Loop-back function of port 0 forwards the frames to the EtherCAT Processing Unit. The loop settings can be controlled by the EtherCAT master.

FMMU

Fieldbus Memory Management Units are used for bitwise mapping of logical addresses to physical addresses of the ESC.

SyncManager

SyncManagers are responsible for consistent data exchange and mailbox communication between EtherCAT master and slaves. The communication direction can be configured for each SyncManager. Read or write transactions may generate events for the EtherCAT master and an attached µController respectively. The SyncManagers are responsible for the main difference between an ESC and a dual-ported memory, because they map addresses to different buffers and block accesses depending on the SyncManager state. This is also a fundamental reason for bandwidth restrictions of the PDI.

Monitoring

The Monitoring unit contains error counters and watchdogs. The watchdogs are used for observing communication and returning to a safe state in case of an error. Error counters are used for error detection and analysis.

Reset

The integrated reset controller observes the supply voltage and controls external and internal resets (ET1100 and ET1200 ASICs only).

PHY Management

The PHY Management unit communicates with Ethernet PHYs via the MII management interface. This is either used by the master or by the slave. The MII management interface is used by the ESC itself for restarting autonegotiation after receive errors with the enhanced link detection mechanism, and for the optional MI link detection and configuration feature.

Distributed Clock

Distributed Clocks (DC) allow for precisely synchronized generation of output signals and input sampling, as well as time stamp generation of events. The synchronization may span the entire EtherCAT network.

Memory

An EtherCAT slave can have an address space of up to 64Kbyte. The first block of 4 Kbyte (0x0000-0x0fff) is used for registers and user memory. The memory space from address 0x1000 onwards is used as the process memory (up to 60 Kbyte). The size of process memory depends on the device. The ESC address range is directly addressable by the EtherCAT master and an attached µController.

Process Data Interface (PDI) or Application Interface

There are several types of PDIs available, depending on the ESC:

- Digital I/O (8-32 bit, unidirectional/bidirectional, with DC support)
- SPI slave
- 8/16 bit µController (asynchronous or synchronous)
- On-chip bus (e.g., Avalon® for Altera® FPGAs or PLB/OPB for Xilinx® FPGAs)
- General purpose I/O

The PDIs are described in Section III of the particular ESC, since the PDI functions are highly depending on the ESC type.

SII EEPROM

One non-volatile memory is needed for EtherCAT Slave Information (ESI) storage, typically an I²C EEPROM. If the ESC is implemented as an FPGA, a second non-volatile memory is necessary for the FPGA configuration code.

Status / LEDs

The Status block provides ESC and application status information. It controls external LEDs like the application RUN LED/ERR LED and port Link/Activity LEDs.

1.2 Further Reading on EtherCAT and ESCs

For further information on EtherCAT, refer to the EtherCAT specification ETG.1000, available from the EtherCAT Technology Group (ETG, <http://www.ethercat.org>), and the IEC standard “Digital data communications for measurement and control – Fieldbus for use in industrial control systems”, IEC 61158 Type 12: EtherCAT, available from the IEC (<http://www.iec.ch>).

Additional documents on EtherCAT can be found on the EtherCAT Technology Group website (<http://www.ethercat.org>).

Documentation on Beckhoff Automation EtherCAT Slave Controllers are available at the Beckhoff website (<http://www.beckhoff.com>), e.g., data sheets, application notes, and ASIC pinout configuration tools.

1.3 Scope of Section I

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.

The following Beckhoff ESCs are covered by Section I:

- ET1200
- ET1100
- EtherCAT IP Core for Altera® FPGAs (V2.4.0)
- EtherCAT IP Core for Xilinx® FPGAs (V2.04a)
- ESC20 (Build 22)

2 EtherCAT Protocol

EtherCAT uses standard IEEE 802.3 Ethernet frames, thus a standard network controller can be used and no special hardware is required on master side.

EtherCAT has a reserved EtherType of 0x88A4 that distinguishes it from other Ethernet frames. Thus, EtherCAT can run in parallel to other Ethernet protocols¹.

EtherCAT does not need the IP protocol, however it can be encapsulated in IP/UDP. The EtherCAT Slave Controller processes the frame in hardware. Thus, communication performance is independent from processor power.

An EtherCAT frame is subdivided into the EtherCAT frame header followed by one or more EtherCAT datagrams. At least one EtherCAT datagram has to be in the frame. Only EtherCAT frames with Type 1 in the EtherCAT Header are currently processed by the ESCs. The ESCs also support IEEE802.1Q VLAN Tags, although the VLAN Tag contents are not evaluated by the ESC.

If the minimum Ethernet frame size requirement is not fulfilled, padding bytes have to be added. Otherwise the EtherCAT frame is exactly as large as the sum of all EtherCAT datagrams plus EtherCAT frame header.

2.1 EtherCAT Header

Figure 2 shows how an Ethernet frame containing EtherCAT data is assembled.

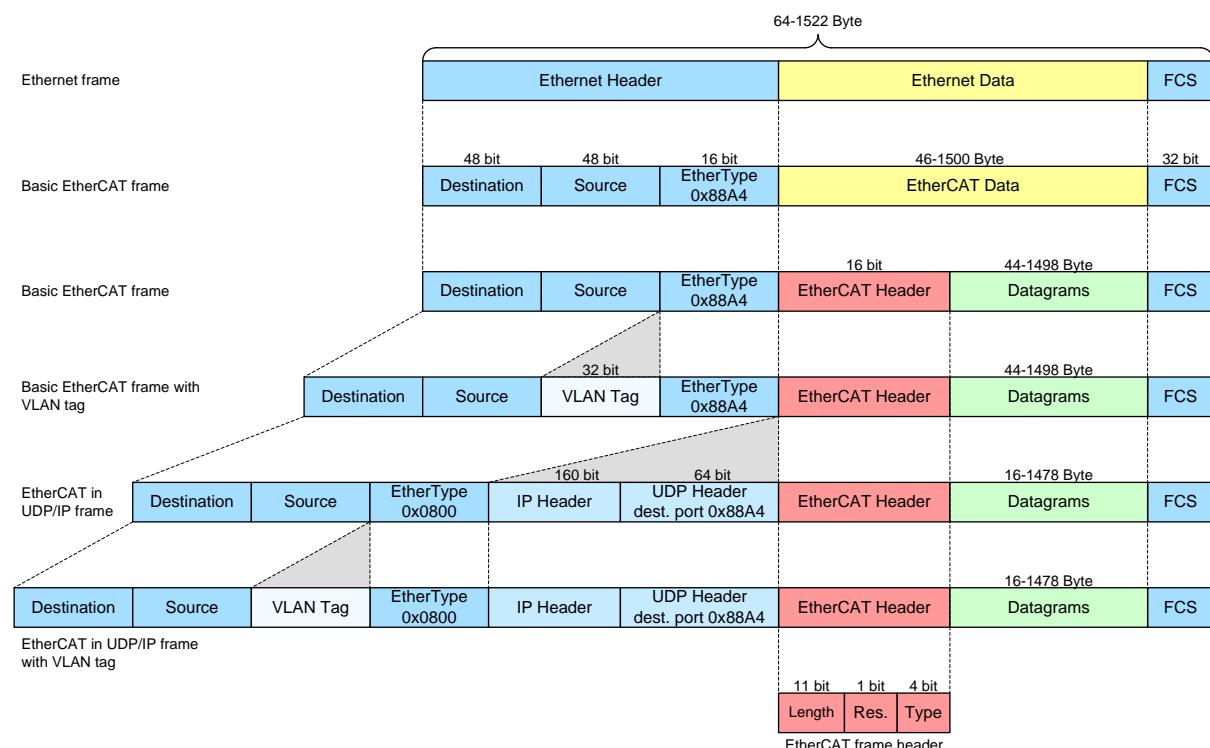


Figure 2: Ethernet Frame with EtherCAT Data

¹ ESCs have to be configured to forward non-EtherCAT frames via DL Control register 0x0100.0.

Table 2: EtherCAT Frame Header

Field	Data Type	Value/Description
Length	11 bit	Length of the EtherCAT datagrams (excl. FCS)
Reserved	1 bit	Reserved, 0
Type	4 bit	Protocol type. Only EtherCAT commands (Type = 0x1) are supported by ESCs.

NOTE: The EtherCAT header length field is ignored by ESCs, they rely on the datagram length fields.

2.2 EtherCAT Datagram

Figure 3 shows the structure of an EtherCAT frame.

* add 1-32 padding bytes if Ethernet frame is shorter than 64 Bytes (Ethernet Header+Ethernet Data+FCS)

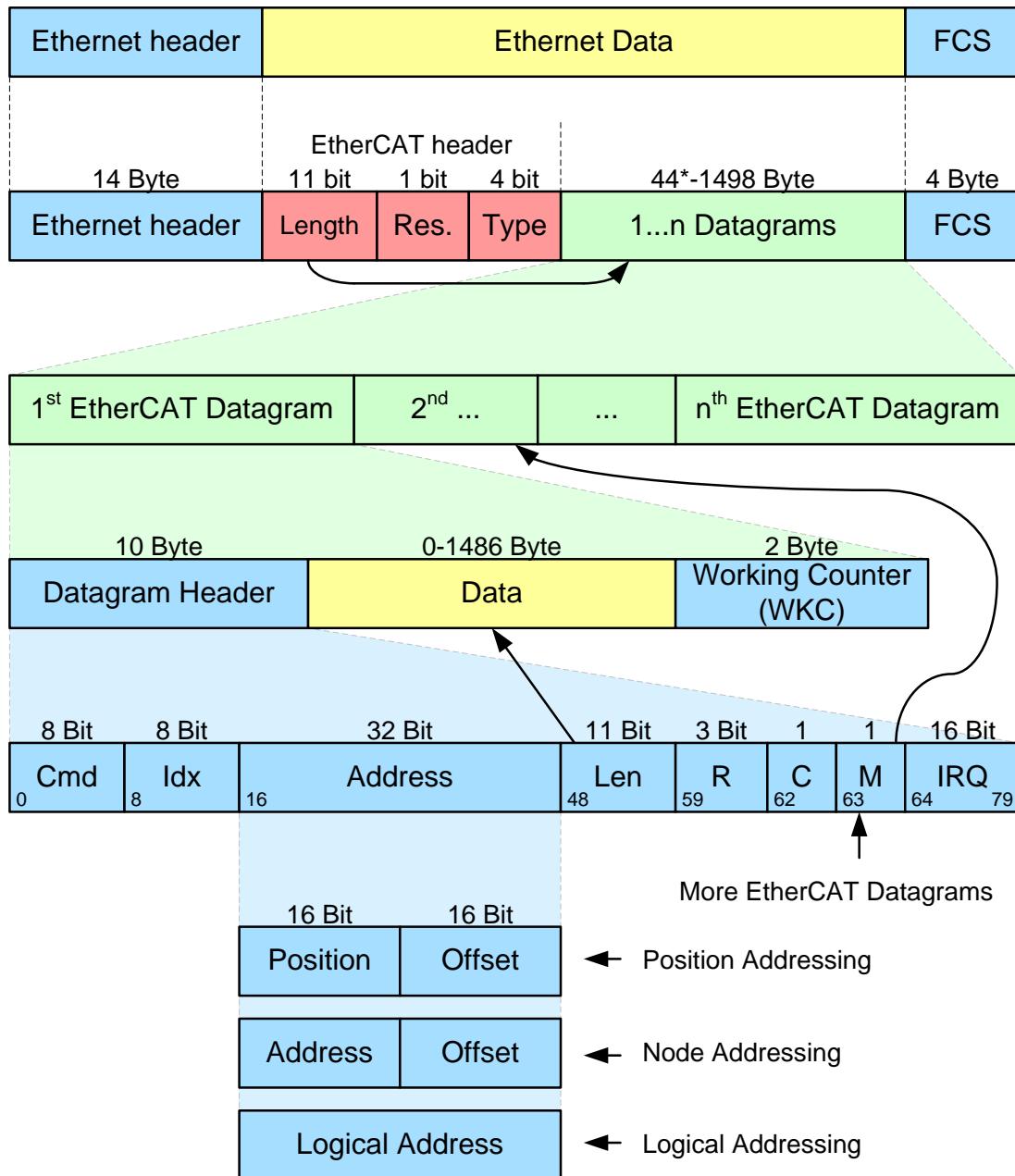


Figure 3: EtherCAT Datagram

Table 3: EtherCAT Datagram

Field	Data Type	Value/Description
Cmd	BYTE	EtherCAT Command Type (see 2.5)
Idx	BYTE	The index is a numeric identifier used by the master for identification of duplicates/lost datagrams, that shall not be changed by EtherCAT slaves
Address	BYTE[4]	Address (Auto Increment, Configured Station Address, or Logical Address, see 2.3)
Len	11 bit	Length of the following data within this datagram
R	3 bit	Reserved, 0
C	1 bit	Circulating frame (see 3.3): 0: Frame is not circulating 1: Frame has circulated once
M	1 bit	More EtherCAT datograms 0: Last EtherCAT datagram 1: More EtherCAT datograms will follow
IRQ	WORD	EtherCAT Event Request registers of all slaves combined with a logical OR
Data	BYTE[n]	Read/Write Data
WKC	WORD	Working Counter (see 2.4)

2.3 EtherCAT Addressing Modes

Two addressing modes of EtherCAT devices are supported within one segment: device addressing and logical addressing. Three device addressing modes are available: auto increment addressing, configured station address, and broadcast. EtherCAT devices can have up to two configured station addresses, one is assigned by the master (Configured Station Address), the other one is stored in the SII EEPROM and can be changed by the slave application (Configured Station Alias address). The EEPROM setting for the Configured Station Alias address is only taken over at the first EEPROM loading after power-on or reset.

Table 4: EtherCAT Addressing Modes

Mode	Field	Data Type	Value/Description
Auto Increment Address	Position	WORD	Each slave increments Position. Slave is addressed if Position = 0.
	Offset	WORD	Local register or memory address of the ESC
Configured Station Address	Address	WORD	Slave is addressed if Address matches Configured Station Address or Configured Station Alias (if enabled).
	Offset	WORD	Local register or memory address of the ESC
Broadcast	Position	WORD	Each slave increments Position (not used for addressing)
	Offset	WORD	Local register or memory address of the ESC
Logical Address	Address	DWORD	Logical Address (configured by FMMUs) Slave is addressed if FMMU configuration matches Address.

2.3.1 Device Addressing

The device can be addressed via Device Position Address (Auto Increment address), by Node Address (Configured Station Address/Configured Station Alias), or by a Broadcast.

- Position Address / Auto Increment Address:
The datagram holds the position address of the addressed slave as a negative value. Each slave increments the address. The slave which reads the address equal zero is addressed and will execute the appropriate command at receive.
Position Addressing should only be used during start up of the EtherCAT system to scan the fieldbus and later only occasionally to detect newly attached slaves. Using Position addressing is problematic if loops are closed temporarily due to link problems. Position addresses are shifted in this case and e.g. a mapping of error register values to devices becomes impossible, thus the faulty link can not be localized.
- Node Address / Configured Station Address and Configured Station Alias:
The configured Station Address is assigned by the master during start up and can not be changed by the EtherCAT slave. The Configured Station Alias address is stored in the SII EEPROM and can be changed by the EtherCAT slave. The Configured Station Alias has to be enabled by the master. The appropriate command action will be executed if Node Address matches with either Configured Station Address or Configured Station Alias.
Node addressing is typically used for register access to individual and already identified devices.
- Broadcast:
Each EtherCAT slave is addressed.
Broadcast addressing is used e.g. for initialization of all slaves and for checking the status of all slaves if they are expected to be identical.

Each slave device has a 16 bit local address space (address range 0x0000:0x0FFF is dedicated for EtherCAT registers, address range 0x1000:0xFFFF is used as process memory) which is addressed via the Offset field of the EtherCAT datagram. The process memory address space is used for application communication (e.g. mailbox access).

2.3.2 Logical Addressing

All devices read from and write to the same logical 4 Gbyte address space (32 bit address field within the EtherCAT datagram). A slave uses a mapping unit (FMMU, Fieldbus Memory Management Unit) to map data from the logical process data image to its local address space. During start up the master configures the FMMUs of each slave. The slave knows which parts of the logical process data image have to be mapped to which local address space using the configuration information of the FMMUs.

Logical Addressing supports bit wise mapping. Logical Addressing is a powerful mechanism to reduce the overhead of process data communication, thus it is typically used for accessing process data.

2.4 Working Counter

Every EtherCAT datagram ends with a 16 Bit Working Counter (WKC). The Working Counter counts the number of devices that were successfully addressed by this EtherCAT datagram. Successfully means that the ESC is addressed and the addressed memory is accessible (e.g., protected SyncManager buffer). EtherCAT Slave Controllers increments the Working Counter in hardware. Each datagram should have an expected Working Counter value calculated by the master. The master can check the valid processing of EtherCAT datagrams by comparing the Working Counter with the expected value.

The Working Counter is increased if at least one byte/one bit of the access was successfully read and/or written. The Read-Multiple-Write commands ARMW and FRWM are either treated like a read command or like a write command, depending on the address match.

Table 5: Working Counter Increment

Command	Data Type	Increment
Read command	No success	no change
	Successful read	+1
Write command	No success	no change
	Successful write	+1
ReadWrite command	No success	no change
	Successful read	+1
	Successful write	+2
	Successful read and write	+3

2.5 EtherCAT Command Types

All supported EtherCAT Command types are listed in Table 6. For ReadWrite operations, the Read operation is performed before the Write operation.

Table 6: EtherCAT Command Types

CMD	Abbr.	Name	Description
0	NOP	No Operation	Slave ignores command
1	APRD	Auto Increment Read	Slave increments address. Slave puts read data into the EtherCAT datagram if received address is zero.
2	APWR	Auto Increment Write	Slave increments address. Slave writes data into memory location if received address is zero.
3	APRW	Auto Increment Read Write	Slave increments address. Slave puts read data into the EtherCAT datagram and writes the data into the same memory location if received address is zero.
4	FPRD	Configured Address Read	Slave puts read data into the EtherCAT datagram if address matches with one of its configured addresses
5	FPWR	Configured Address Write	Slave writes data into memory location if address matches with one of its configured addresses
6	FPRW	Configured Address Read Write	Slave puts read data into the EtherCAT datagram and writes data into the same memory location if address matches with one of its configured addresses
7	BRD	Broadcast Read	All slaves put logical OR of data of the memory area and data of the EtherCAT datagram into the EtherCAT datagram. All slaves increment position field.
8	BWR	Broadcast Write	All slaves write data into memory location. All slaves increment position field.
9	BRW	Broadcast Read Write	All slaves put logical OR of data of the memory area and data of the EtherCAT datagram into the EtherCAT datagram, and write data into memory location. BRW is typically not used. All slaves increment position field.
10	LRD	Logical Memory Read	Slave puts read data into the EtherCAT datagram if received address matches with one of the configured FMMU areas for reading.
11	LWR	Logical Memory Write	Slaves writes data to into memory location if received address matches with one of the configured FMMU areas for writing.
12	LRW	Logical Memory Read Write	Slave puts read data into the EtherCAT datagram if received address matches with one of the configured FMMU areas for reading. Slaves writes data to into memory location if received address matches with one of the configured FMMU areas for writing.
13	ARMW	Auto Increment Read Multiple Write	Slave increments address. Slave puts read data into the EtherCAT datagram if received address is zero, otherwise slave writes the data into memory location.

CMD	Abbr.	Name	Description			
14	FRMW	Configured Read Multiple Write	Slave puts read data into the EtherCAT datagram if address matches with one of its configured addresses, otherwise slave writes the data into memory location.			
15-255		reserved				

Table 7: EtherCAT Command Details

CMD	High Adr. In	High Adr. Out	Low Adr.	Address Match	Data In	Data Out	WKC	
NOP	untouched			none	untouched			
APRD	Position	Pos.+1	Offset	ADP=0	-	Read	+0/1	
APWR	Position	Pos.+1	Offset	ADP=0		Write	+0/1	
APRW	Position	Pos.+1	Offset	ADP=0	Write	Read	+0/1/2/3	
FPRD	Address		Offset	ADP=conf. station addr.	-	Read	+0/1	
FPWR	Address		Offset	ADP=conf. station addr.	Write		+0/1	
FPRW	Address		Offset	ADP=conf. station addr.	Write	Read	+0/1/2/3	
BRD	High Add. In+1	Offset	all			Data In OR Read	+0/1	
BWR		Offset	all			Write	+0/1	
BRW	High Add. In+1	Offset	all		Write	Data In OR Read	+0/1/2/3	
LRD	Logical address		FMMU		-	(Read AND bit_mask ¹) or (Data In and not bit_mask ¹)	+0/1	
LWR	Logical address		FMMU			Write	+0/1	
LRW	Logical address		FMMU		Write	(Read AND bit_mask ¹) or (Data In and not bit_mask ¹)	+0/1/2/3	
ARMW	Position	Pos.+1	Offset	Read: ADP=0	-	Read	+0/1	
				Write: ADP/=0		Write	+0/1	
FRMW	Address		Offset	Read: ADP= conf. station address./alias	-	Read	+0/1	
				Write: ADP/= conf. station address./alias	Write		+0/1	

NOTE: Working Counter (WKC) increment depends on address match

¹ bit_mask depends on FMMU configuration if bit-wise mapping is used: only masked bits are actually addressed by the logical read/write command.

3 Frame Processing

The ET1100, ET1200, IP Core, and ESC20 slave controllers only support Direct Mode addressing: neither a MAC address nor an IP address is assigned to the ESC, they process EtherCAT frames with any MAC or IP address.

It is not possible to use unmanaged switches between these ESCs or between master and the first slave, because source and destination MAC addresses are not evaluated or exchanged by the ESCs. Only the source MAC address is modified when using the default settings, so outgoing and incoming frames can be distinguished by the master.

NOTE: Attaching an ESC directly to an office network will result in network flooding, since the ESC will reflect any frame – especially broadcast frames – back into the network (broadcast storm).

The frames are processed by the ESC on the fly, i.e., they are not stored inside the ESC. Data is read and written as the bits are passing the ESC. The forwarding delay is minimized to achieve fast cycle times. The forwarding delay is defined by the receive FIFO size and the EtherCAT Processing Unit delay. A transmit FIFO is omitted to reduce delay times.

The ESCs support EtherCAT, UDP/IP, and VLAN tags. EtherCAT frames and UDP/IP frames containing EtherCAT datagrams are processed. Frames with VLAN tags are processed by the ESCs, the VLAN settings are ignored and the VLAN tag is not modified.

The source MAC address is changed for every frame passing the EtherCAT Processing Unit (SOURCE_MAC[1] is set to 1 – locally administered address). This helps to distinguish between frames transmitted by the master and frames received by the master.

3.1 Loop Control and Loop State

Each port of an ESC can be in one of two states: open or closed. If a port is open, frames are transmitted to other ESCs at this port, and frames from other ESCs are received. A port which is closed will not exchange frames with other ESCs, instead, the frames are forwarded internally to the next logical port, until an open port is reached.

The loop state of each port can be controlled by the master (ESC DL Control register 0x0100). The ESCs supports four loop control settings, two manual configurations, and two automatic modes:

Manual open

The port is open regardless of the link state. If there is no link, outgoing frames will be lost.

Manual close

The port is closed regardless of the link state. No frames will be sent out or received at this port, even if there is a link with incoming frames.

Auto

The loop state of each port is determined by the link state of the port. The loop is open if there is a link, and it is closed without a link.

Auto close (manual open)

The port is closed depending on the link state, i.e., if the link is lost, the loop will be closed (auto close). If the link is established, the loop will not be automatically opened, instead, it will remain closed (closed wait state). Typically, the port has to be opened by the master explicitly by writing the loop configuration again to the ESC DL Control register 0x0100. This write access has to enter the ESC via a different open port. There is an additional fall-back option for opening the port: if a valid Ethernet frame is received at the closed port in Auto close mode, it will also be opened after the CRC is received correctly, without explicit master interaction.

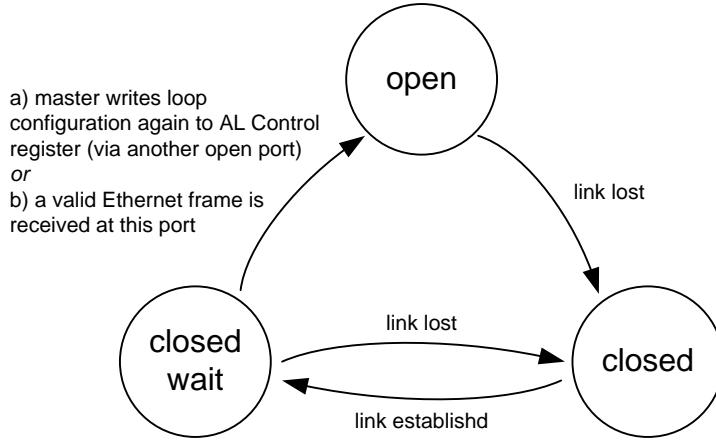


Figure 4: Auto close loop state transitions

A port is considered open if the port is available, i.e., it is enabled in the configuration, and one of the following conditions is met:

- The loop setting in the DL Control register is Auto and there is an active link at the port.
- The loop setting in the DL Control register is Auto close and there is an active link at the port and the DL Control register was written again after the link was established.
- The loop setting in the DL Control register is Auto close and there is an active link at the port and a valid frame was received at this port after the link was established.
- The loop setting in the DL control register is Always open

A port is considered closed if one of the following conditions is met:

- The port is not available or not enabled in the configuration.
- The loop setting in the DL Control register is Auto and there is no active link at the port.
- The loop setting in the DL Control register is Auto close and there is no active link at the port or the DL Control register was not written again after the link was established
- The loop setting in the DL Control register is Always closed

NOTE: If all ports are closed (either manually or automatically), port 0 will be opened as the recovery port. Reading and writing via this port is possible, although the DL status register reflects the correct status. This can be used to correct DL control register settings.

Registers used for loop control and loop/link status are listed in Table 8.

Table 8: Registers for Loop Control and Loop/Link Status

Register Address	Name	Description
0x0100[15:8]	ESC DL Control	Loop control/loop setting
0x0110[15:4]	ESC DL Status	Loop and link status

3.2 Frame Processing Order

The frame processing order of EtherCAT Slave Controllers depends on the number of ports (logical port numbers are used):

Table 9: 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) or 0→EtherCAT Processing Unit→3 / 3→1 / 1→0 (log. ports 0,1, and 3)
4	0→EtherCAT Processing Unit→3 / 3→1 / 1→2 / 2→0

The direction through an ESC including the EtherCAT Processing Unit is called “processing” direction, other directions without passing the EtherCAT Processing Unit are called “forwarding” direction.

Ports which are not implemented behave similar to closed ports, the frame is forwarded to the next port.

Figure 5 shows the frame processing in general:

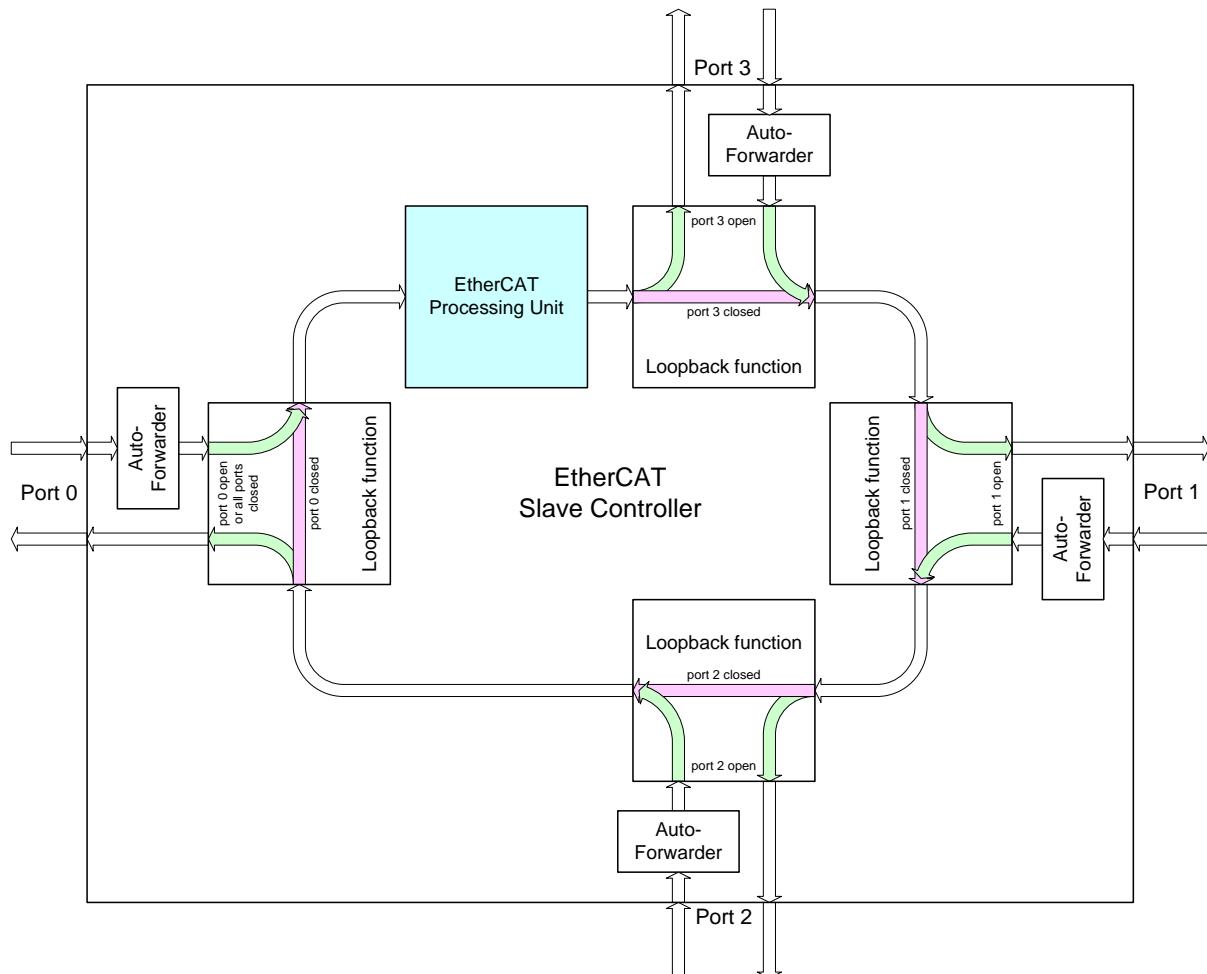


Figure 5: Frame Processing

Example Port Configuration with Ports 0, 1, and 2

If there are only ports 0, 1, and 2, 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 logical port 3 which is not configured, so the Loopback function of port 3 forwards it to port 1. If port 1 is closed, the frame is forwarded by the Loopback function to port 2. If port 1 is open, the frame is sent out at port 1. When the frame comes back into port 1, it is handled by the Auto-Forwarder and sent to port 2. Again, if port 2 is closed, the frame is forwarded to port 0, otherwise, it is sent out at port 2. When the frame comes back into port 2, it is handled by the Auto-Forwarder and then sent to the Loopback function of port 0. Then it is handled by the Loopback function and sent out at port 0 – back to the master.

3.3 Permanent Ports and Bridge Port

The EtherCAT ports of an ESC are typically permanent ports, which are directly available after Power-On. Permanent ports are initially configured for Auto mode, i.e., they are opened after the link is established. Additionally, some ESCs support an EtherCAT Bridge port (port 3), which is configured in the SII EEPROM like PDI interfaces. This Bridge port becomes available if the EEPROM is loaded successfully, and it is closed initially, i.e., it has to be opened (or set to Auto mode) explicitly by the EtherCAT master.

3.4 Shadow Buffer for Register Write Operations

The ESCs have shadow buffers for write operations to registers (0x0000 to 0x0F7F). During a frame, write data is stored in the shadow buffers. If the frame is received correctly, the values of the shadow buffers are transferred into the effective registers. Otherwise, the values of the shadow buffers are not taken over. As a consequence of this behavior, registers take their new value shortly after the FCS of an EtherCAT frame is received. SyncManagers also change the buffers after the frame was received correctly.

User and Process Memory do not have shadow buffers. Accesses to these areas are taking effect directly. If a SyncManager is configured to User Memory or Process Memory, write data will be placed in the memory, but the buffer will not change in case of an error.

3.5 Circulating Frames

The ESCs incorporate a mechanism for prevention of circulating frames. This mechanism is very important for proper watchdog functionality.

This is an example network with a link failure between slave 1 and slave 2:

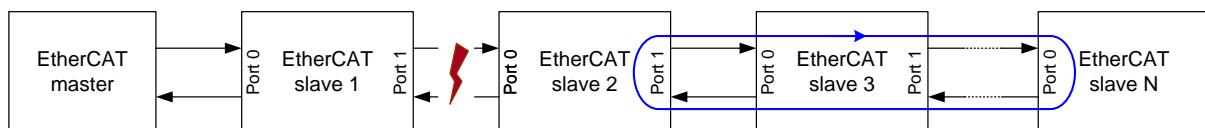


Figure 6: Circulating Frames

Both slave 1 and slave 2 detect the link failure and close their ports (port 1 at slave 1 and port 0 at slave 2). A frame currently traveling through the ring at the right side of slave 2 might start circulating. If such a frame contains output data, it might trigger the built-in watchdog of the ESCs, so the watchdog never expires, although the EtherCAT master can not update the outputs anymore.

To prevent this, a slave with no link at port 0 and loop control for port 0 set to Auto or Auto close (ESC DL Control register 0x0100) will do the following inside the EtherCAT Processing Unit:

- If the Circulating bit of the EtherCAT datagram is 0, set the Circulating bit to 1
- If the Circulating bit is 1, do not process the frame and destroy it

The result is that circulating frames are detected and destroyed. Since the ESCs do not store the frames for processing, a fragment of the frame will still circulate triggering the Link/Activity LEDs. Nevertheless, the fragment is not processed.

3.5.1 Unconnected Port 0

Port 0 must not be left intentionally unconnected (slave hardware or topology) because of the circulating frame prevention. All frames will be dropped after they have passed an automatically closed Port 0 for the second time, and this can prohibit any EtherCAT communication.

Example: Port 0 of slave 1 and 3 are automatically closed because nothing is connected. Slave 1 detects this and destroys the frames.

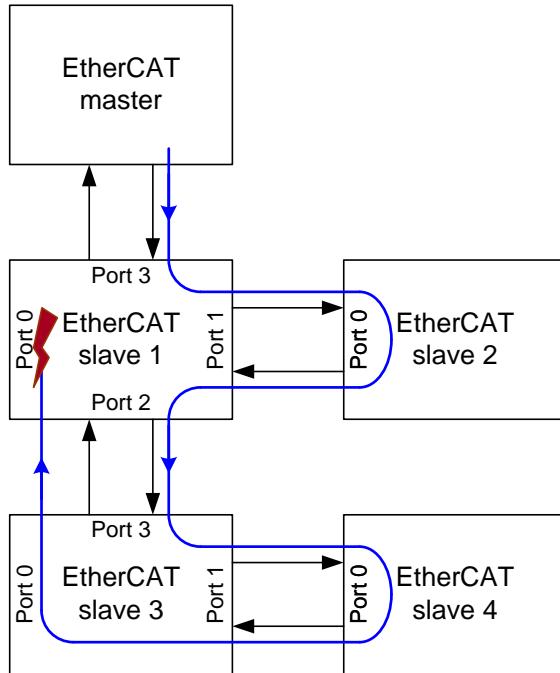


Figure 7: All frames are dropped because of Circulating Frame Prevention

In redundancy operation, only one Port 0 is automatically closed, so the communication remains active.

3.6 Non-EtherCAT Protocols

If non-EtherCAT protocols are used, the forwarding rule in the ESC DL Control register (0x0100.0) has to be set to forward non-EtherCAT protocols. Otherwise they are destroyed by the ESC.

3.7 Special Functions of Port 0

Port 0 of each EtherCAT is characterized by some special functions in contrast to ports 1, 2, and 3:

- Port 0 is intended to lead to the master, i.e., port 0 is the *upstream* port, all other ports (1-3) are considered to be *downstream* ports.
- The link state of Port 0 influences the Circulating Frame bit, and frames are dropped at port 0 if the bit is set and the link is automatically closed.
- Port 0 loop state is open if all ports are closed (either automatically or manually).
- Port 0 has a special behavior when using standard EBUS link detection.

4 Physical Layer Common Features

EtherCAT supports two types of Physical Layers, Ethernet and EBUS. The Ethernet interface of ESCs is MII (or RMII), connecting to an external Ethernet PHY according to IEEE 802.3 100BaseTX or FX. For EBUS, the physical layer is integrated into the EtherCAT ASICs. EtherCAT requires 100 Mbit/s links with full duplex communication.

The MII/RMII interfaces of Beckhoff ESCs are optimized e.g. for low processing/forwarding delays. The resulting additional requirements to Ethernet PHYs are described in the corresponding chapters.

4.1 Link Status

The link status of each port is available in the ESC DL Status register (0x0110:0x0111), most important are the “Communication established” bits 15,13,11, and 9. Additional link information is available in the PHY Port status register (0x0518:0x051B) if MI link detection and configuration is used. All other status bits are mainly for debugging purposes. The link status bits are described in the following table.

Table 10: Link Status Description

Status register	MII				EBUS	
	LINK_MII signal		Management Interface			
	Standard link detection	Enhanced link detection	Standard link detection	Enhanced link detection	Standard link detection	Enhanced link detection
ESC DL Status: Physical link 0x0110[7:4]	LINK_MII signal state		LINK_MII signal combined with MI Link Detection and Configuration result		Result of the standard link detection mechanism	
ESC DL Status: Communication established 0x0110[15,13,11,9]	LINK_MII signal state	LINK_MII signal state combined with RX_ERR threshold state	LINK_MII signal combined with MI Link Detection and Configuration result	LINK_MII signal combined with RX_ERR threshold state and MI Link Detection and Configuration result	Result of the standard link detection mechanism.	Result of the enhanced link detection mechanism.
PHY port status: physical link status 0x0518.0, 0x0519.0, 0x051A.0, 0x051B.0	n.a.		PHY has detected link (PHY Status register 1.2)		n.a.	
PHY port status: Link status 0x0518.1, 0x0519.1, 0x051A.1, 0x051B.1			PHY has detected link, link is suitable for ECAT			

If all ports are closed (either manually or automatically, e.g., because no port has a communication link), port 0 is automatically opened as the recovery port. Reading and writing via this port is possible, although the DL status register reflects the correct status. This can be used to correct erroneous DL control register settings or to fix LINK_MII polarity configuration.

4.2 Selecting Standard/Enhanced Link Detection

Some ESCs distinguish between standard and enhanced link detection. Enhanced link detection provides additional security mechanisms regarding link establishment and surveillance. The Enhanced link detection setting affects both Ethernet and EBUS physical layer. Using enhanced link detection is recommended for MII ports (refer to chapter 6.5 for compatibility issues with EBUS enhanced link detection). Some ESCs only support global Enhanced Link Detection configuration for all ports (issue if EBUS ports are also used), some support port-wise configuration.

After power-on, enhanced link detection is enabled by default. It is disabled or remains enabled after the SII EEPROM is loaded according to the EEPROM setting (register 0x0140). An invalid EEPROM content will also disable enhanced link detection.

The EEPROM setting for enhanced Link detection is only taken over at the first EEPROM loading after power-on or reset. Changing the EEPROM and manually reloading it will not affect the enhanced link detection enable status (register 0x0110.2), even if the EEPROM could not be read initially.

Registers used for Enhanced link detection are listed in Table 11.

Table 11: Registers for Enhanced Link Detection

Register Address	Name	Description
0x0140.9	PDI Control	Enable/disable Enhanced link detection for all ports
0x0140[15:12]	PDI Control	Enable/disable Enhanced link detection port-wise
0x0110.2	ESC DL Status	Enhanced link detection status

NOTE: Some of these register bits are set via SII EEPROM/IP Core configuration, or they are not available in specific ESCs. Refer to Section II and III for details.

4.3 FIFO Size Reduction

The ESCs incorporate a receive FIFO (RX FIFO) for decoupling receive clock and processing clock. The FIFO size is programmable by the EtherCAT master (ESC DL Control register 0x0100). The FIFO size values determine a reduction of the FIFO size, the FIFO can not be disabled completely. The FIFO size can be reduced considering these three factors:

- Accuracy of the receiver's clock source
- Accuracy of the sender's clock source
- Maximum frame size

The default FIFO size is sufficient for maximum Ethernet Frames and default Ethernet clock source accuracy (100 ppm). If the clock accuracy is 25 ppm or better, the FIFO size can be reduced to the minimum. The minimum FIFO size is also sufficient for minimum Ethernet Frames (64 Byte) and 100 ppm clock sources, larger frames are not forwarded reliably. If the FIFO size was accidentally reduced, a short 64 Byte frame can be sent for resetting the FIFO size to the default value.

The FIFO size can be reduced to minimum if both sender and receiver have 25 ppm accuracy of their clock sources, even with maximum frame size.

Since 25 ppm clock accuracy can typically not be guaranteed for the entire life-time of a clock source, the actual clock deviation has to be measured on a regular basis for FIFO size reduction. If a slave does not support Distributed Clocks or the actual deviation is larger than 25 ppm, the FIFO size of all neighbors and the slave itself can not be reduced. The actual deviation can be measured using Distributed Clocks:

- Compare DC Receive Times over a period of time for slaves which only support DC Receive Times. Do not use this method if both slaves which are compared support DC Time Loop, since the measured deviation will approximate zero if the DC control loop has settled, but the actual deviation determining the FIFO size might be larger than 25 ppm.
- Compare calculated deviation from register Speed Counter Diff (0x0932:0x0933) for adjacent slaves with DC Time Loop support after the DC control loop has settled (i.e., System Time Difference 0x092C:0x092F is at its minimum).

NOTE: Be careful with FIFO size reduction at the first slave, if off-the-shelf network interface cards without 25 ppm accuracy are used by master.

4.4 Frame Error Detection

Refer to chapter 14 (Error Counters) for details on frame error detection.

5 Ethernet Physical Layer

ESCs with Ethernet Physical Layer support use MII interfaces, some do also support the RMII interface. Since RMII PHYs include TX FIFOs, they increase the forwarding delay of an EtherCAT slave device as well as the jitter. RMII is not recommended due to these reasons.

5.1 Requirements to Ethernet PHYs

EtherCAT and Beckhoff ESCs have some general requirements to Ethernet PHYs, which are typically fulfilled by state-of-the-art Ethernet PHYs. Refer to the EtherCAT Slave Controller application note "PHY Selection Guide" for example Ethernet PHYs.

 The MII interfaces of Beckhoff ESCs are optimized for low processing/forwarding delays by **omitting a transmit FIFO**. To allow this, the Beckhoff ESCs have additional requirements to Ethernet PHYs, which are easily accomplished by several PHY vendors.

Refer to Section III for ESC specific information about supported features.

Requirements to Ethernet PHYs used for EtherCAT:

- The PHYs have to comply with **IEEE 802.3 100BaseTX or 100BaseFX**.
- The PHYs have to support 100 Mbit/s Full Duplex links.
- The PHYs have to provide an **MII** (or RMII¹) interface.
- The PHYs have to use autonegotiation.
- The PHYs have to support the MII management interface.
- The PHYs have to support **MDI/MDI-X auto-crossover**.
- **PHY link loss reaction time** (link loss to link signal/LED output change) has to be faster than 15 µs to enable redundancy operation².
- The PHYs must not modify the preamble length.

Additional requirements to Ethernet PHYs used with Beckhoff ESCs:

- The PHYs have to provide a **signal indicating a 100 Mbit/s (Full Duplex) link**³, typically a configurable LED output. The signal polarity is active low or configurable for some ESCs.
- The **PHY addresses** should be equivalent to the **logical port number (0-3)**. Some ESCs also support a fixed offset (e.g. offset 16, PHY addresses are logical port number plus 16: 16-19), or even an arbitrary offset. If none of these possibilities can be used, the PHY address should be configured to logical port number plus 1 (1-4), although some features (e.g., Enhanced Link Detection) can not be used in this case, because apart from the optional configurable PHY address offset, the PHY addresses are hard-coded inside the ESCs.
- PHY configuration must not rely on configuration via the MII management interface, i.e., required features have to be enabled after power-on, e.g., by default or by **strapping options**. PHY startup should not rely on MII management interaction, i.e., MDC clocking, since many ESCs do not communicate with the PHY via management interface unless the EtherCAT master requests this (only the EtherCAT IP Core with MI Link detection and configuration will communicate without master interaction).

¹ RMII is only supported by the EtherCAT IP Core

² This can either be achieved by a PHY with such a link loss reaction time or by activating Enhanced link detection if the PHY asserts RX_ER both inside and outside of frames for each invalid symbol. Enhanced link detection requires proper PHY address configuration. Devices with one or more EBUS ports which do not support port-wise configuration can not be configured to use Enhanced link detection without sacrificing compatibility to older ESCs.

³ If a combined signal (100 MBit/s link with Full Duplex) is not available, a signal indicating 100 Mbit/s speed might be used. Take care that the speed signal is inactive (10 Mbit/s) in case of no link. If only a Link signal is available, this might be used. Never use (combined) activity signals.

Additional requirements to Ethernet PHYs used with Beckhoff ESCs using the MII Interface:

- All PHYs connected to one ESC and the ESC itself must share the **same clock source**. This can be achieved by sourcing the PHYs from an ESC clock output or by sourcing the PHYs and the ESC from the same quartz oscillator. The ESC20 uses TX_CLK as a clock source, both PHYs have to share the same quartz oscillator.
- The **TX_CLK** signals of the PHYs must have a **fixed phase relation to the clock input** of the PHYs with a tolerance of ± 5 ns, because a TX FIFO is omitted. During operation the phase relation can not change since the PHYs and the ESC have to share the same clock source. The phase offset is compensated inside the ESC either manually by configuration or automatic:
Manual TX Shift compensation: ET1100, ET1200, and IP Core provide a TX Shift configuration option (configurable TX_EN/TXD signal delay by 0/10/20/30 ns) which is used for all MII ports. Thus, all PHYs connected to one ESC must have the same fixed phase relation between TX_CLK and their clock input. This is typically true if the same PHY model is used for all ports. The phase relation has to be the same each time the PHYs are powered on. As the ESC20 use TX_CLK as device clock source, configuration is not necessary, but the requirements for manual TX Shift compensation have to be fulfilled anyway.
Automatic TX Shift compensation: The IP Core supports automatic TX Shift compensation individually for each port. With automatic TX Shift compensation, the PHYs are not required to have the same fixed phase relation each time they are powered on.

Recommendations to Ethernet PHYs used for EtherCAT:

- Receive and transmit delays should be deterministic.
- Maximum cable length should be ≥ 120 m to maintain a safety margin if the standard maximum cable length of 100 m is used.
- ESD tolerance should be as high as possible (4kV or better)
- Baseline wander should be compensated (even at maximum cable length)
- MDC should not incorporate pull-up/pull-down resistors, as this signal is used as a configuration input signal by some ESCs.
- Restriction of Autonegotiation advertisement to 100 Mbit/s / Full Duplex is desirable (configured by hardware strapping options).
- Power consumption should be as low as possible.
- I/O voltage: 3.3V should be supported for current ASIC and FPGA ESCs, additional 2.5V I/O support is recommended for recent FPGA ESCs.
- Single power supply according to I/O voltage (3.3V or 2.5V).
- The PHY should use a 25 MHz clock source (quartz oscillator or ESC output).
- Industrial temperature range should be supported.

NOTE: The following requirements defined by IEEE802.3 have to be observed: a) The preamble length should be maintained. Accumulating preamble reduction below 2 bytes including Start-of-Frame-Delimiter/SFD (0x55 5D) must not occur for single or cascaded ESCs. ESCs can not regenerate preambles to 8 bytes including SFD because of the on-the-fly processing, received and transmitted preamble length is identical. b) Receive and transmit delays should comply with the standard (RX delay should be below ~320 ns, TX delay below ~140 ns).

5.2 MII Interface Signals

The MII interface has the signals¹ shown in Figure 8:

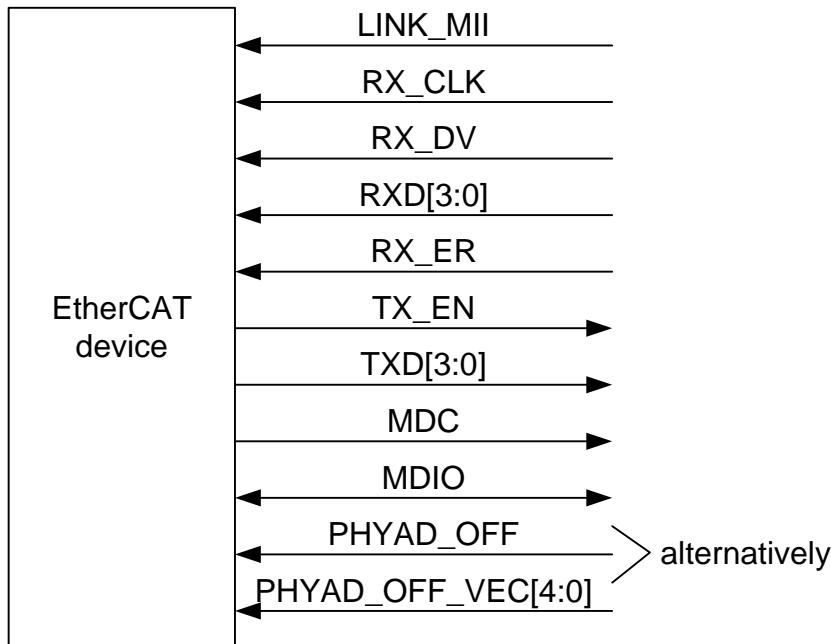


Figure 8: MII Interface signals

Table 12: MII Interface signals

Signal	Direction	Description
LINK_MII	IN	Input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established
RX_CLK	IN	Receive Clock
RX_DV	IN	Receive data valid
RXD[3:0]	IN	Receive data (alias RX_D)
RX_ER	IN	Receive error (alias RX_ERR)
TX_EN	OUT	Transmit enable (alias TX_ENA)
TXD[3:0]	OUT	Transmit data (alias TX_D)
MDC	OUT	Management Interface clock (alias MI_CLK)
MDIO	BIDIR	Management Interface data (alias MI_DATA)
PHYAD_OFF or PHYAD_OFF_VEC[4:0]	IN	PHY address offset configuration (0 or 16) or PHY address offset configuration (0-31)

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

If an ESC MII interface is not used, LINK_MII has to be tied to a logic value which indicates no link, and RX_CLK, RXD, RX_ER, and especially RX_DV have to be tied to GND. The TX outputs can be left unconnected, unless they are used for ESC configuration.

¹ The availability of the MII signals as well as their exact names depend on the specific ESC. Refer to Section III.

Table 13: Special/Unused MII Interface signals

Signal	Direction at PHY	Description
TX_CLK	OUT	<p>ESC20: TX_CLK of one PHY is used as clock source, TX_CLK of other PHY is unused, leave open.</p> <p>IP Core: TX_CLK is optionally used for automatic TX Shift compensation.</p> <p>Other Beckhoff ESCs: Unused, leave unconnected.</p>
COL	OUT	<p>Collision detected.</p> <p>ESC20: Connected, but not used.</p> <p>Other Beckhoff ESCs: Unused. Leave unconnected.</p>
CRS	OUT	<p>Carrier sense.</p> <p>ESC20: Connected, but not used.</p> <p>Other Beckhoff ESCs: Unused. Leave unconnected</p>
TX_ER	IN	<p>Transmit error.</p> <p>ESC20: Connected, always driven low.</p> <p>Other Beckhoff ESCs: Connect to GND.</p>

For more details about the MII interface, refer to IEEE Standard 802.3 (Clause 22), available from the IEEE (<http://standards.ieee.org/getieee802>).

5.3 RMII Interface Signals

The RMII interface has the signals¹ shown in Figure 9:

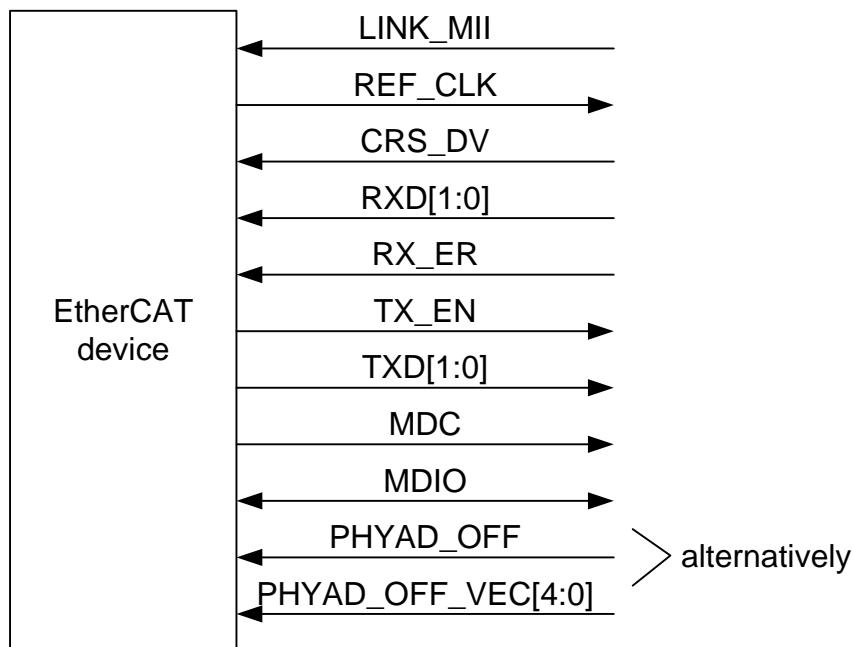


Figure 9: RMII Interface signals

Table 14: RMII Interface signals

Signal	Direction	Description
LINK_MII	IN	Input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established
REF_CLK	OUT	50 MHz Reference clock
CRS_DV	IN	Carrier Sense/Receive data valid
RXD[1:0]	IN	Receive data (alias RX_D)
RX_ER	IN	Receive error (alias RX_ERR)
TX_EN	OUT	Transmit enable (alias TX_ENA)
TXD[1:0]	OUT	Transmit data (alias TX_D)
MCLK	OUT	Management Interface clock (alias MI_CLK)
MDIO	BIDIR	Management Interface data (alias MI_DATA)
PHYAD_OFF or PHYAD_OFF_VEC[4:0]	IN	PHY address offset configuration (0 or 16) or PHY address offset configuration (0-31)

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.

If an ESC RMII interface is not used, LINK_MII has to be tied to a logic value which indicates no link, and RXD, RX_ER, and especially CRS_DV have to be tied to GND. The TX signals can be left unconnected, unless they are used for ESC configuration.

For more details about the RMII interface, refer to the RMII Specification, available from the RMII consortium (e.g., <http://www.national.com/appinfo/networks>).

¹ The availability of the RMII signals as well as their exact names depend on the specific ESC. Refer to Section III.

5.4 Link Detection

All ESCs support a LINK_MII signal for link detection at each Ethernet MII port. Some ESCs (e.g., EtherCAT IP Core) additionally support link detection and configuration via the MII management interface. Both the LINK_MII signals and the MI Link Detection and Configuration results (if available) are combined to determine the link state of each port, which is reflected in the ESC DL Status register (0x0110[15,13,11,9] – Communication established). Using the LINK_MII signal is recommended since it is the only way to achieve fast link loss reaction times.

Table 15: Registers used for Ethernet Link Detection

Register Address	Name	Description
0x0110:0x0111	ESC DL Status	Link Status (Link MII, Communication established)
0x0518:0x051B	PHY Port Status	MI Link Detection results if available

5.4.1 LINK_MII Signal

The LINK_MII signal used for link detection is typically an LED output signal of the Ethernet PHY. If available, LINK_MII should be connected to a combined signal indicating a 100 Mbit/s Full Duplex link. If such a signal is not available, a signal indicating a 100 Mbit/s link (speed LED) might be used. If only a Link signal is available (link LED), this might be used. Never use (combined) activity signals, e.g., Link/Act LED outputs, because the link state will toggle upon activity.

The main advantage of using a dedicated link signal instead of reading out MII management interface registers is the fast reaction time in case of a link loss. This is crucial for redundancy operation, since only one lost frame is tolerated. The EtherCAT port of an ESC which loses a link has to be closed as fast as possible to maintain EtherCAT communication at the other ports and to reduce the number of lost frames.

The LINK_MII signal state is reflected in the ESC DL Status register (0x0110[7:4]).

5.4.2 MI Link Detection and Configuration

The EtherCAT IP Core supports link detection and PHY configuration by using the MII management interface. Initially, the PHY configuration is checked and updated if necessary. Afterwards, the link status of each Ethernet port is cyclically polled. PHY accesses of the EtherCAT master are inferred upon request.

The MI Link Configuration mechanism configures the Ethernet PHYs to use Autonegotiation and advertise only 100BASE-TX Full-Duplex connections.

In spite of the configured Autonegotiation advertisement, Ethernet PHYs will establish links to link partners without or with disabled Autonegotiation. Such a link can not be used by EtherCAT, so the MI Link Detection will check if the link characteristics fulfill EtherCAT requirements. It checks if a link is established, if Autonegotiation has finished successfully and if the link partner also used Autonegotiation. If all conditions are met, an MI Link is detected.

Since the MI Link Detection does not solely rely on the PHY link status bit (register 1.2), the local PHY and the remote PHY may indicate a link, but the ESC refuses it because it does not fulfill EtherCAT requirements. The current MI Link Detection state is reflected in the MI Interface registers (PHY Port Status 0x0518:0x051B).

MI Link Detection and Configuration must not be used without link detection via LINK_MII signals, because link loss reaction time would otherwise be too slow for redundancy operation. Enhanced Link Detection might not be a suitable solution in this case if too few RX_ERR are issued to the ESC before the PHY takes down the link.

The MI Link Detection and Configuration checks the management communication with Ethernet PHYs. If communication is not possible – e.g. because no PHY is configured for the expected PHY address – the results are ignored. Take care of proper PHY address configuration to prevent erroneous behavior.

NOTE: Proper PHY address settings and PHY address offset configuration is crucial for MI Link Detection and Configuration.

5.5 Standard and Enhanced MII Link Detection

For Ethernet, the standard or enhanced MII link detection feature is a feature of link error detection and reaction. This has to be distinguished from the actual link detection, which tells the ESC if a physical link is available (i.e., the LINK_MII signal or the MI link detection and configuration mechanism).

Enhanced MII link detection, in contrast to standard MII link detection, will additionally disconnect a link if at least 32 RX errors (RX_ER) occur in a fixed interval of time (~10 µs). The local loop is closed and the link partner is informed by restarting the Auto-Negotiation mechanism via the MII Management Interface. This informs the link partner of the error condition, and the link partner will close the loop.

The ESC keeps the port closed until the link goes down during Auto-Negotiation and comes up again (the port remains closed if the link does not go down).

The availability of Enhanced MII Link Detection depends on a supported PHY address / PHY address offset configuration, otherwise it has to be disabled.

5.6 MII Management Interface (MI)

Most EtherCAT slave controllers with MII/RMII ports use the MII management interface for communication with the Ethernet PHYs. Most ESCs do not use the management interface for link detection or configuration of link modes. For link detection, the ESC is recommended to use a separate signal (LINK_MII). The MII management interface can be used by the EtherCAT master – or the local µController if supported by the ESC. Enhanced MII link detection uses the management interface for restarting autonegotiation after communication errors occurred. Some ESCs (EtherCAT IP Core) can make use of the MII management interface for link detection and PHY configuration.

Refer to chapter 5.4 for details about link detection with Ethernet PHYs. For more details about the MII management interface, refer to IEEE Standard 802.3 (Clause 22), available from the IEEE (<http://standards.ieee.org/getieee802>).

5.6.1 PHY Addressing/PHY Address Offset

Proper PHY address configuration is crucial for Enhanced Link Detection and MI Link Detection and configuration, because the ESC itself needs to relate logical ports to the corresponding PHY addresses. The EtherCAT master can access the Ethernet PHYs using any address by means of the PHY address register 0x0513.

Basically, each ESC addresses a PHY by using the logical port number plus an optional PHY address offset. The available PHY address offset values are ESC dependent and configured by using the PHY address configuration signal (PHYAD_OFFSET). The PHY address offset is also added to the PHY address register value if the EtherCAT master accesses a PHY.

Typically, the PHY address offset should be 0, and the logical port numbers match with the PHY addresses. Some Ethernet PHYs associate a special function with PHY address 0, e.g., address 0 is a broadcast PHY address. In these cases, PHY address 0 cannot be used. Instead, a PHY address offset different from 0 should be selected, preferably an offset which is supported by the ESC. If PHY addresses are chosen which are not supported by the ESC, Enhanced Link Detection and MI Link Detection and Configuration cannot be used and have to be disabled (the PHY address offset should be 0 in these cases). Nevertheless, the EtherCAT master can communicate with the PHYs using the actual PHY addresses, and EtherCAT communication is possible anyway – using the LINK_MII signal. It is recommended that the PHY addresses are selected to be equal to the logical port number plus 1 in this case. If port 0 is EBUS, ports 1-3 should have PHY addresses 1-3, i.e., PHY address offset is 0.

If the PHY address offset configuration of an ESC reflects the actual PHY address settings, the EtherCAT master can use addresses 0-3 in PHY address register 0x0513 for accessing the PHYs of logical ports 0-3, regardless of the PHY address offset.

Table 16: PHY Address configuration matches PHY address settings

Logical Port	Configured address of the PHY		PHY address register value used by EtherCAT master
	PHY address offset = 0	PHY address offset = 16	
0	0	16	0
1	1	17	1
2	2	18	2
3	3	19	3
none	4-15	20-31	4-15
none	16-31	0-15	16-31

If the actual PHY address settings differ from the PHY address configuration of the ESC, the EtherCAT master has to use the actual PHY address mapping, i.e., PHY addresses 1-4 for accessing the PHYs of logical ports 0-3. The PHY address mapping is intended to become part of the ESI.

Table 17: PHY Address configuration does not match actual PHY address settings

Logical Port	Configured address of the PHY	PHY address register value used by EtherCAT master
none	0	0
0	1	1
1	2	2
2	3	3
3	4	4
none	5-31	5-13

NOTE: PHY address offset is 0 in this case (recommended).

5.6.2 Logical Interface

The MI of the ESC is typically controlled by EtherCAT via the MI registers¹.

Table 18: MII Management Interface Register Overview

Register Address	Description
0x0510:0x0511	MII Management Control/Status
0x0512	PHY Address
0x0513	PHY Register Address
0x0514:0x0515	PHY Data

The MI supports two commands: write to one PHY register or read one PHY register.

5.6.2.1 MI read/write example

The following steps have to be performed for a PHY register access:

1. Check if the Busy bit of the MI Status register is cleared and the MI is not busy.
2. Write PHY address to PHY Address register.
3. Write PHY register number to be accessed into PHY Register Address register (0-31).
4. Write command only: put write data into PHY Data register (1 word/2 byte).
5. Issue command by writing to Control register.
For read commands, write 1 into Command Register Read 0x0510.8.
For write commands, write 1 into Write Enable bit 0x0510.0 and also 1 into Command Register Write 0x0510.9. Both bits have to be written in one frame. The Write enable bit realizes a write protection mechanism. It is valid for subsequent MI commands issued in the same frame and self-clearing afterwards.
6. The command is executed after the EOF, if the EtherCAT frame had no errors.
7. Wait until the Busy bit of the MI Status register is cleared.
8. Check the Error bits of the MI Status register. The command error bit is cleared with a valid command or by clearing the command register. The read error bit indicates a read error, e.g., a wrong PHY address. It is cleared by writing to the register.
9. Read command only: Read data is available in PHY Data register.

NOTE: The Command register bits are self-clearing. Manually clearing the command register will also clear the status information.

5.6.2.2 MI Interface Assignment to ECAT/PDI

The EtherCAT master controls the MI Interface (default) if the MII Management PDI Access State register 0x0517.0 is not set. The EtherCAT master can prevent PDI control over the MI Interface, and it can force the PDI to release the MI Interface control. After power-on, the PDI can take over MI Interface control without any master transactions.

¹ ET1100 only: MI Control is transferred to PDI if the Transparent Mode is enabled. IP Core: MI Control by PDI is possible.

5.6.3 MI Protocol

Each MI access begins with a Preamble of 32 “Ones”, followed by a Start-of-Frame (01) and the Operation Code (01 for write and 10 for read operations). Then the PHY address (5 bits) and the PHY register address (5 bits) are transmitted to the PHY. After a Turnaround (10 for write and Z0 for read operations – Z means MDIO is high impedance), two bytes of data follow. The transfer finishes after the second data byte.

5.6.4 Timing specifications

Table 19: MII Management Interface timing characteristics

Parameter	Comment
t_{Clk}	MDC period
t_{Write}	Write access time
t_{Read}	Read access time

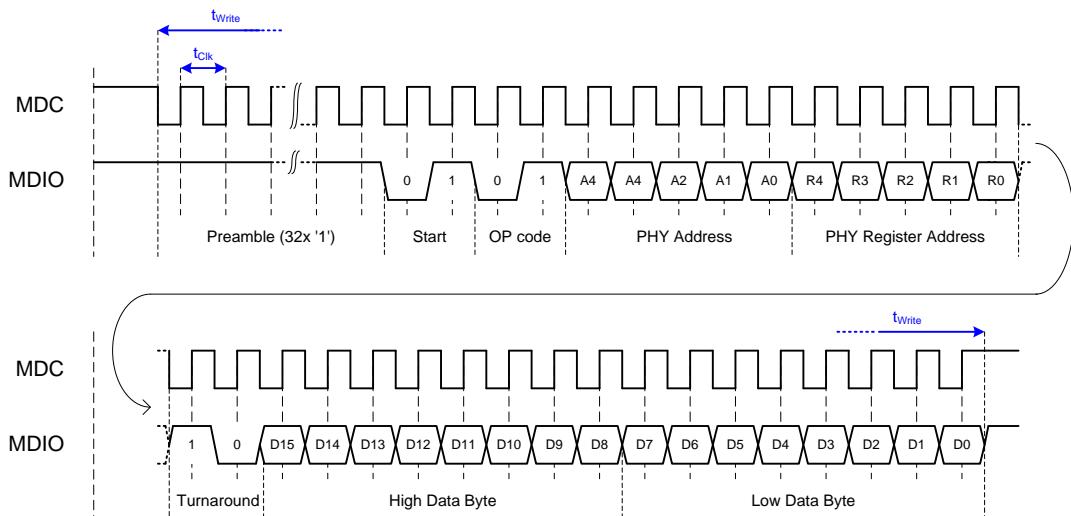


Figure 10: Write access

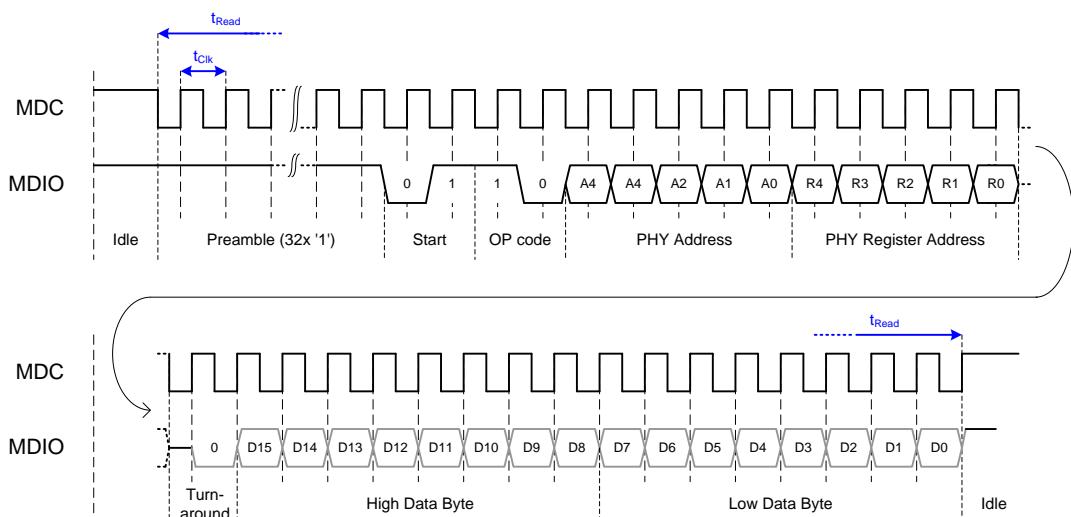


Figure 11: Read access

5.7 Ethernet Termination and Grounding Recommendation

This termination and grounding design recommendation may help to meet the overall requirements for industrial communication. Nevertheless, implementation may vary depending on other requirements like board layout, other capacities, common ground interconnection, and shield grounding.

Unused RJ-45 pins are terminated by 75Ω resistors which will be connected to virtual ground. Virtual GND is connected to Protection Earth (PE) by a $10\text{nF}/500\text{V}$ capacitor in parallel to a $1\text{M}\Omega$ resistor. Shield is also connected to PE by a $10\text{nF}/500\text{V}$ capacitor in parallel to a $1\text{M}\Omega$ resistor. Especially the values for the connection between Virtual GND and PE are subject to change to meet industrial requirements (EMC). Shield is not directly connected with PE to avoid ground loops across large industrial plants with different PE potentials.

This design recommendation of termination and grounding is shown in Figure 12.

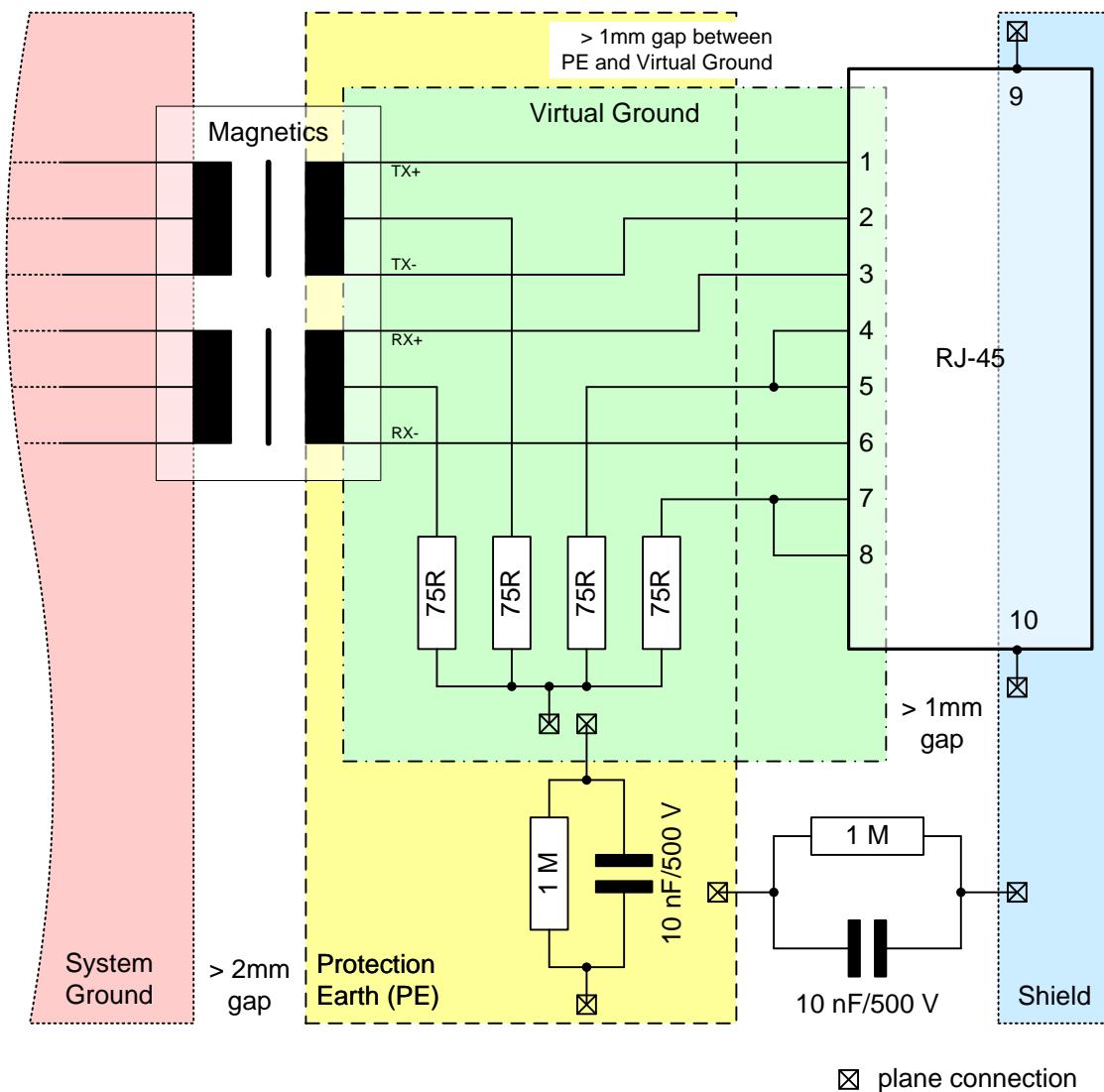


Figure 12: Termination and Grounding Recommendation

5.8 Ethernet Connector (RJ45 / M12)

Fast Ethernet (100BASE-TX) uses two pairs/four pins. An RJ45 connector (8P8C) or an M12 D-code connector can be used. The RJ45 connector is recommended to use MDI pinout (PC side) for all ports for uniformity reasons. Standard Ethernet cables are used, not crossover cables. PHYs have to support MDI/MDI-X auto-crossover.

Table 20: Signals used for Fast Ethernet

Signal	Name	Core Color	Contact Assignment	
	MDI	(may change depending on cable)	RJ45	M12 D-code
TX+	Transmission Data +	Yellow (light orange)	1	1
TX-	Transmission Data -	Orange	2	3
RX+	Receive Data +	Light green	3	2
RX-	Receive Data -	Green	6	4

NOTE: MDI-X (Switches/Hubs) uses same outline as MDI but TX+ is RX+ and TX- is RX- and vice versa.

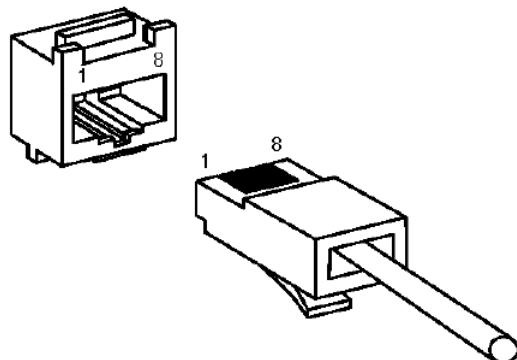


Figure 13: RJ45 Connector

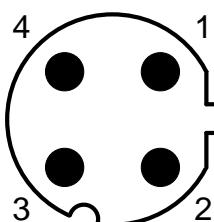


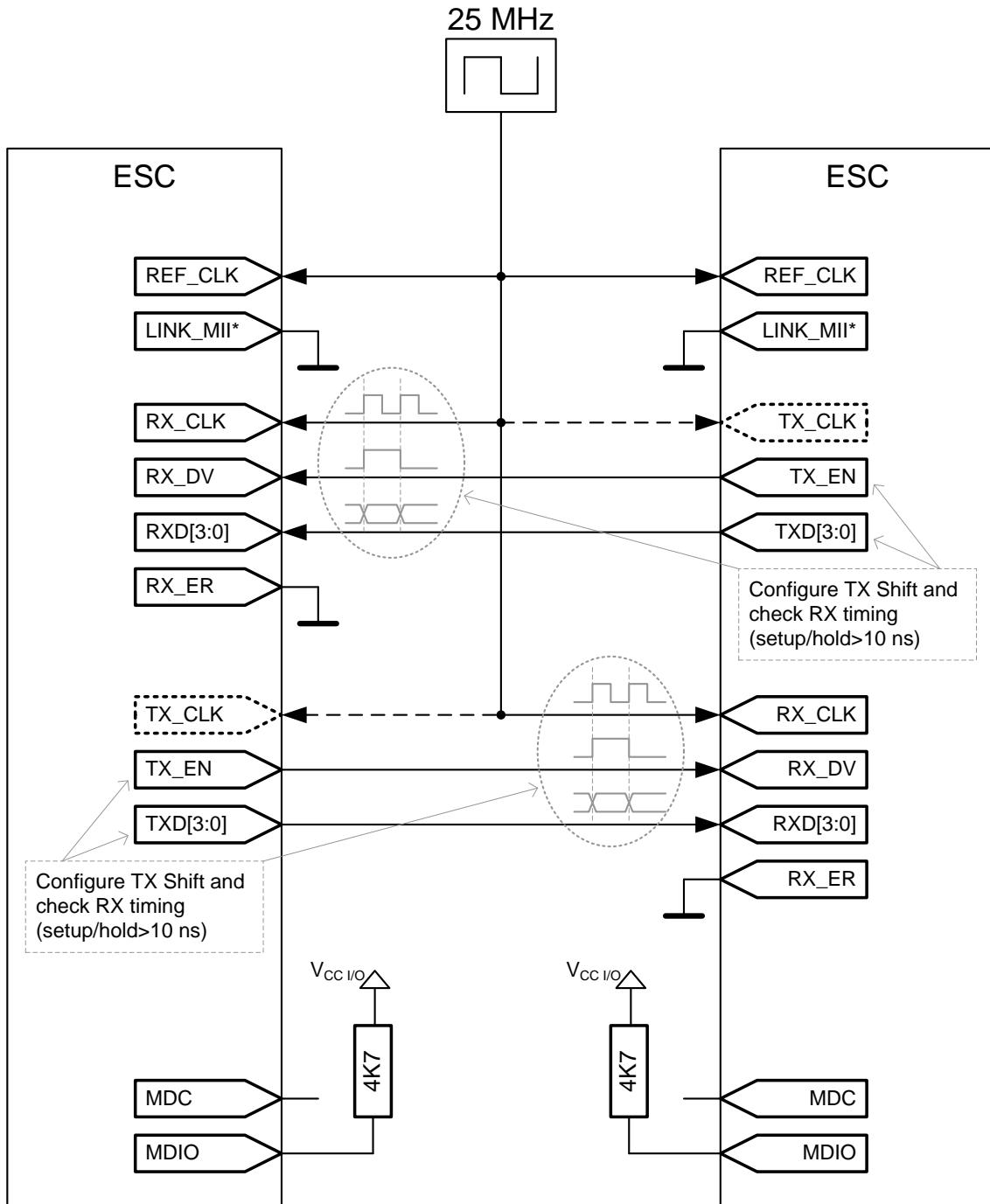
Figure 14: M12 D-code Connector

5.9 Back-to-Back MII Connection

5.9.1 ESC to ESC Connection

Two EtherCAT slave controllers can be connected back-to-back using MII as shown in the figure below. The timing of RX_DV and RXD with respect to RX_CLK has to be checked at both ESCs to be compliant with the IEEE 802.3 requirements of min. 10 ns setup time and min. 10 ns hold time. The timing can be adjusted by configuring the TX Shift settings of each ESC.

Other clocking options besides using a quartz oscillator are also possible.



* if LINK_MII is act. low

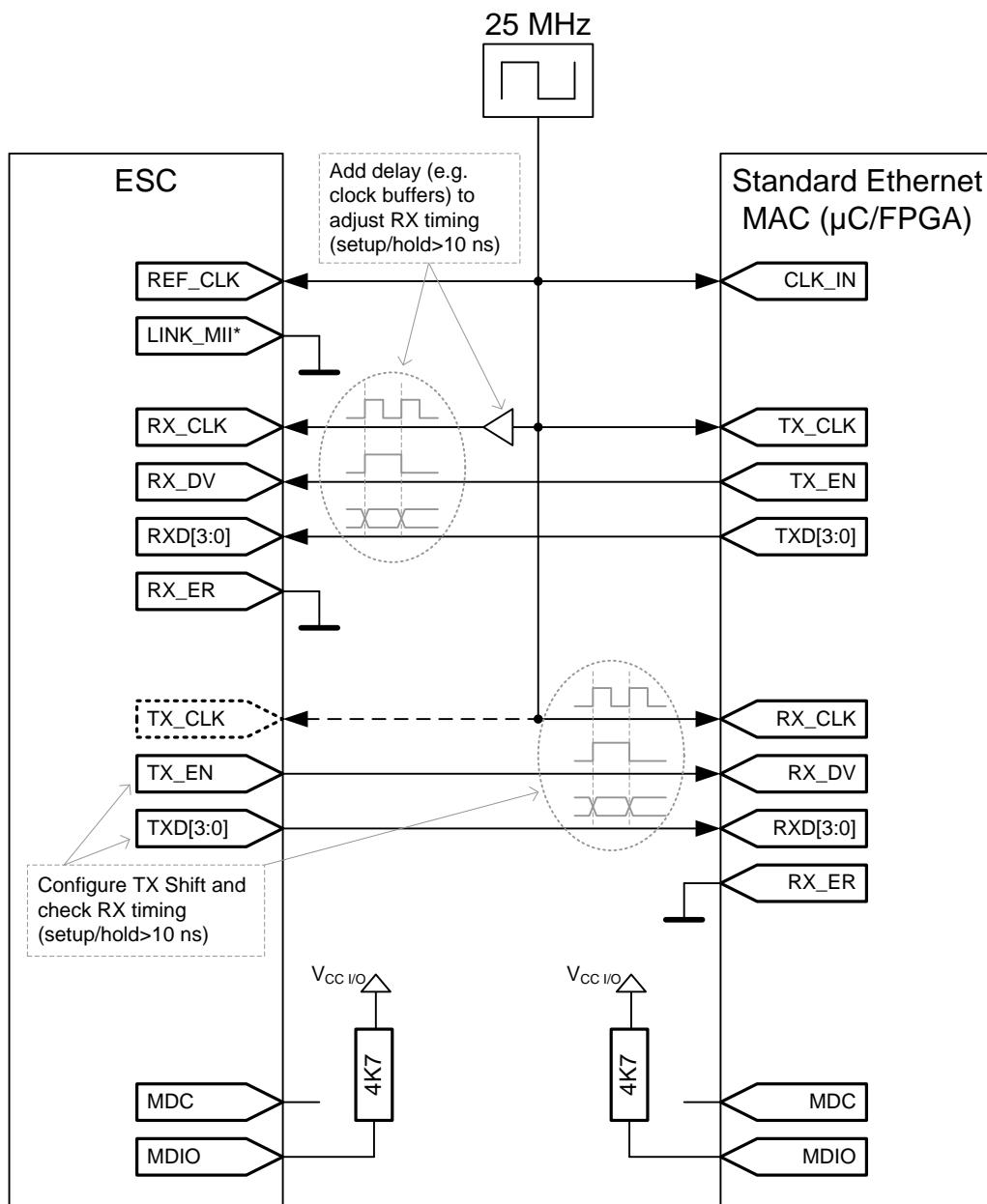
Figure 15: Back-to-Back MII Connection (two ESCs)

5.9.2 ESC to Standard Ethernet MAC

If an ESC is to be connected directly to a standard Ethernet MAC (e.g. µController or FPGA), RX timing at the ESC and the MAC has to be checked. Since TX Shift configuration is not possible in the MAC, RX_CLK for the ESC has to be adjusted (delayed) to achieve proper RX timing at the ESC.

An EtherCAT slave controller can be directly connected to a standard Ethernet MAX using MII as shown in the figure below. The timing of RX_DV and RXD with respect to RX_CLK has to be checked at both ESC and MAC to be compliant with the IEEE 802.3 requirements of min. 10 ns setup time and min. 10 ns hold time. The timing can be adjusted by configuring the TX Shift setting of the ESC and the clock buffer (e.g. using one ore more buffers).

If the standard Ethernet MAC completely uses the IEEE 802.3 TX signal timing window of 0..25 ns, standard compliant RX timing (-10..10 ns) is impossible. Since most Beckhoff ESCs do not require the entire IEEE802.3 RX timing window (check in Section III), a valid configuration is possible even in this case.



* if LINK_MII is act. low

Figure 16: Back-to-Back MII Connection (ESC and standard MAC)

6 EBUS/LVDS Physical Layer

EBUS is an EtherCAT Physical Layer designed to reduce components and costs. It also reduces delay inside the ESC. The EBUS physical layer uses Low Voltage Differential Signaling (LVDS) according to the ANSI/TIA/EIA-644 „Electrical Characteristics of Low Voltage Differential Signaling (LVDS) Interface Circuits” standard.

EBUS has a data rate of 100 Mbit/s to accomplish the Fast Ethernet data rate. The EBUS protocol simply encapsulates Ethernet Frames, thus EBUS can carry any Ethernet frame – not only EtherCAT frames.

EBUS is intended to be used as a backplane bus, it is not qualified for wire connections.

6.1 Interface

Two LVDS signal pairs per EBUS link are used, one for reception and one for transmission of Ethernet/EtherCAT frames.

The EBUS interface has the following signals:

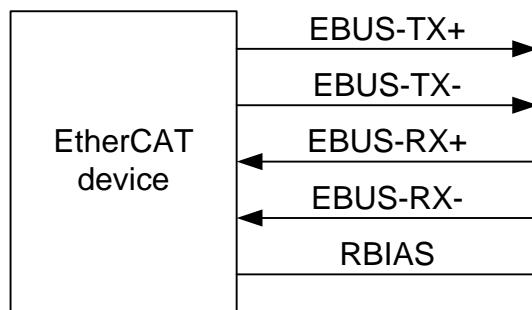


Figure 17: EBUS Interface Signals

Table 21: EBUS Interface signals

Signal	Direction	Description
EBUS-TX+	OUT	EBUS/LVDS transmit signals
EBUS-TX-		
EBUS-RX+	IN	EBUS/LVDS receive signals
EBUS-RX-		
RBIAS		BIAS resistor for EBUS-TX current adjustment

Unused EBUS ports can be left unconnected, only termination (BIAS resistor) is mandatory.

6.2 EBUS Protocol

Ethernet/EtherCAT frames are Manchester encoded (Biphase L) and encapsulated in Start-of-Frame (SOF) and End-of-Frame (EOF) identifiers. A beginning of a frame is detected if a Manchester violation with positive level (N+) followed by a '1' bit occurs. The falling edge in the middle of the '1' indicates the SOF to the ESC. This '1' bit is also the first bit of the Ethernet preamble. Then the whole Ethernet frame is transmitted (including Ethernet SOF at the end of the preamble, up to the CRC). The frame finishes with a Manchester violation with negative level (N-), followed by a '0' bit. This '0' bit is also the first bit of the IDLE phase, which consists of '0' bits.

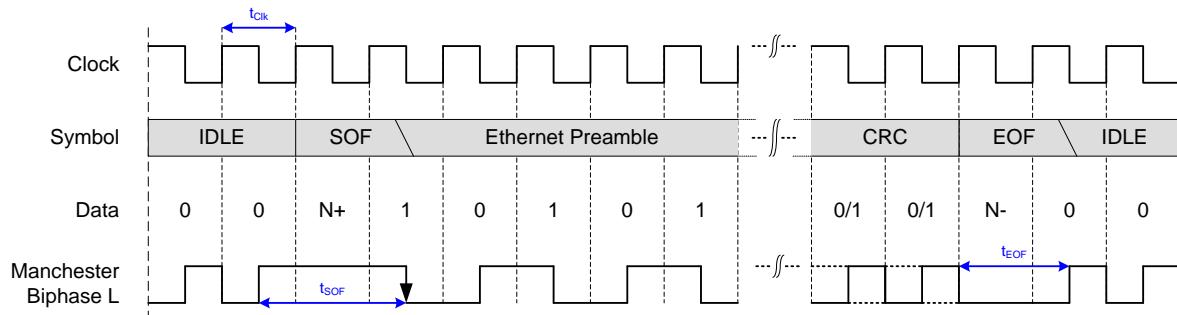


Figure 18: EBUS Protocol

6.3 Timing Characteristics

Table 22: EBUS timing characteristics

Parameter	Min	Typ	Max	Comment
t_{Clk}		10 ns		EBUS Clock (100 Mbit/s)
t_{SOF}	15 ns			Positive level of SOF before falling edge
t_{EOF}	15 ns			Negative level of EOF after last edge

NOTE: After power-on, a receiver which receives IDLE symbols can not distinguish incoming '0' bits from '1' bits, because it is not synchronized to the transmitters clock. Synchronization is established at the falling edge at the end of the EBUS SOF, which indicates the center of the first preamble bit. After synchronization, idle errors can be detected by the ESC.

NOTE: SOF is detected at a falling edge following a period of at least 15 ns (nominal) of positive level, EOF is detected after a period of at least 15 ns (nominal) of negative level. I.e., the length of SOF and EOF can be even longer.

6.4 Standard EBUS Link Detection

Standard EBUS link detection is realized by counting the number of good signal events (no RX error) in a defined interval of time and comparing it to a given value. The link is established if enough events occurred, and disconnected if too few events occurred. IDLE symbols as well as any kind of EtherCAT traffic produce enough good events.

In order to handle partial link failures correctly, the following mechanism is used:

- An ESC transmits at port 0 only if a link is detected (e.g., IDLE symbols are received), otherwise it will transmit N- symbols.
- An ESC transmits at ports 1, 2, and 3 regardless of the link state (typically IDLE symbols if no frames are pending).

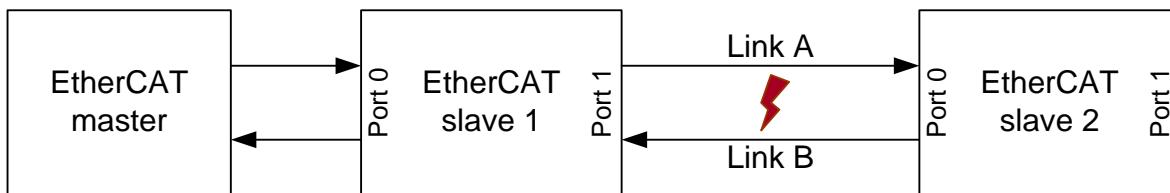


Figure 19: Example EtherCAT Network

This method addresses these two cases of partial link failure (see Figure 19):

- A failure on Link A will be detected by Slave 2, which will stop transmitting anything on Link B (and close the loop at port 0). This is detected by Slave 1, which will close the loop at port 1. The master can still communicate with slave 1.
- A failure on Link B will be detected by Slave 1, which will close the loop at port 1. The master can still communicate with slave 1. This failure can not be detected by slave 2, which will leave port 0 open.

Do not connect EBUS port 0 of one EtherCAT slave to EBUS port 0 of another EtherCAT slave using standard link detection, because standard link detection will not establish a link after it was down.

Do not connect EBUS port 1-3 of one EtherCAT slave to EBUS port 1-3 of another EtherCAT slave using standard link detection, because partial link failures will not result in closed ports.

NOTE: Standard link detection can not cope with a specific partial link fault (Link B failure), which affects redundancy operation (e.g., port 1 of slave 2 is connected to the master), because the master can not communicate with slave 2 which leaves its port 0 open.

NOTE: Another advantage of this mechanism is that in case slave 2 is added to the network, at first port 0 of slave 2 is opened because there is activity on Link A, then transmission on Link B is started, and finally slave 1 opens Port 1. This assures that no frames get lost during link establishment.

6.5 Enhanced EBUS Link Detection

Enhanced EBUS link detection uses the standard link detection mechanism and adds a simple link detection handshake protocol before the link is established.

With enhanced link detection, the ESC transmits on all ports regardless of the link state (unless frames are transmitted; typically IDLE symbols are transmitted if no frames are pending).

The handshake protocol consists of three phases:

1. Each device starts transmitting a 4 nibble link request frame with 0x55C4 regularly at each port. This frame has no SOF/CRC and would be discarded if it was accidentally received by standard Ethernet devices (e.g., masters).
2. If a link request frame (0x55C4) is received at a port, the ESC will transmit link acknowledge frames (0x55CC) instead of link request frames at this port.
3. If a link acknowledge frame (0x55CC) is received at a port, the link at the port is established. Both link partners know that the link is good and establish the link. No further link detection frames are transmitted (until the link is interrupted and the link detection handshake phases are starting from the beginning).

Link disconnection is signaled to the link partner by stopping transmission for a certain time. This will be detected by the default link detection mechanism. The link gets disconnected at both sides, and both sides close their loops. After that, the first phase of the handshake protocol starts again.

EBUS enhanced link detection is not compatible with older devices which forward enhanced link detection handshake frames depending on the direction (e.g. ESC20 and bus terminals without ASICs): the handshake frames are not forwarded through the EtherCAT Processing Unit, but they are forwarded without modification alongside the EtherCAT Processing Unit.

A device using enhanced link detection will stop generating handshake frames after the link is established or the enhanced link detection is disabled by SII EEPROM setting. It will restart generating handshake frames shortly after a link is lost (unless enhanced link detection is disabled).

6.6 EBUS RX Errors

The EBUS receiver detects the following RX errors:

- Missing edge in the middle of one bit (but not EBUS SOF/EOF)
- EBUS SOF inside a frame (two SOFs without EOF in between)
- EBUS EOF outside a frame (two EOFs without SOF in between)
- IDLE violation: '1' outside a frame
- Too short pulses (smaller than ~3.5 ns)

A frame with an RX error is discarded. Too many RX errors in a defined interval of time will result in link disconnection.

Errors outside of a frame are counted only once, and errors inside a frame are also counted only once, because the Manchester decoding might have lost synchronization.

6.7 EBUS Low Jitter

In Low Jitter mode, the jitter of the forwarding delay (EBUS port to EBUS port) is reduced.

6.8 EBUS Connection

The connection of two EBUS ports is shown in Figure 20. LVDS termination with 100 Ohm impedance is necessary between each pair of receive signals – located adjacent to the receive inputs.

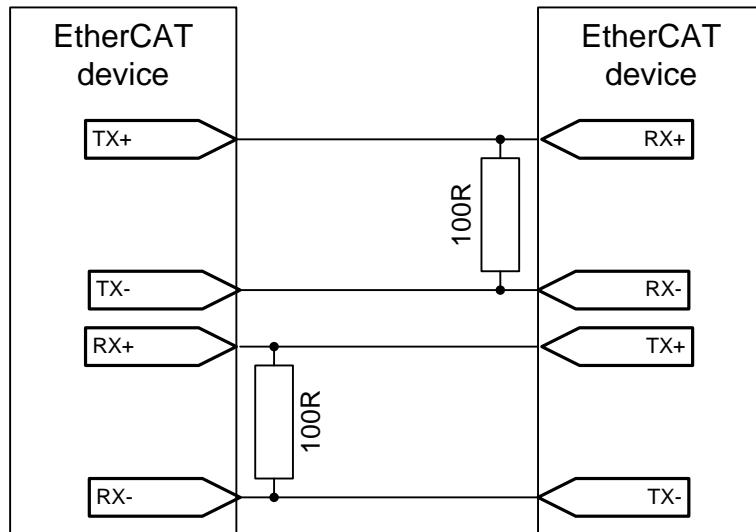


Figure 20: EBUS Connection

7 FMMU

Fieldbus Memory Management Units (FMMU) convert logical addresses into physical addresses by the means of internal address mapping. Thus, FMMUs allow to use logical addressing for data segments that span several slave devices: one datagram addresses data within several arbitrarily distributed ESCs. Each FMMU channel maps one continuous logical address space to one continuous physical address space of the slave. The FMMUs of Beckhoff ESCs support bit wise mapping, the number of supported FMMUs depends on the ESC. The access type supported by an FMMU is configurable to be either read, write, or read/write.

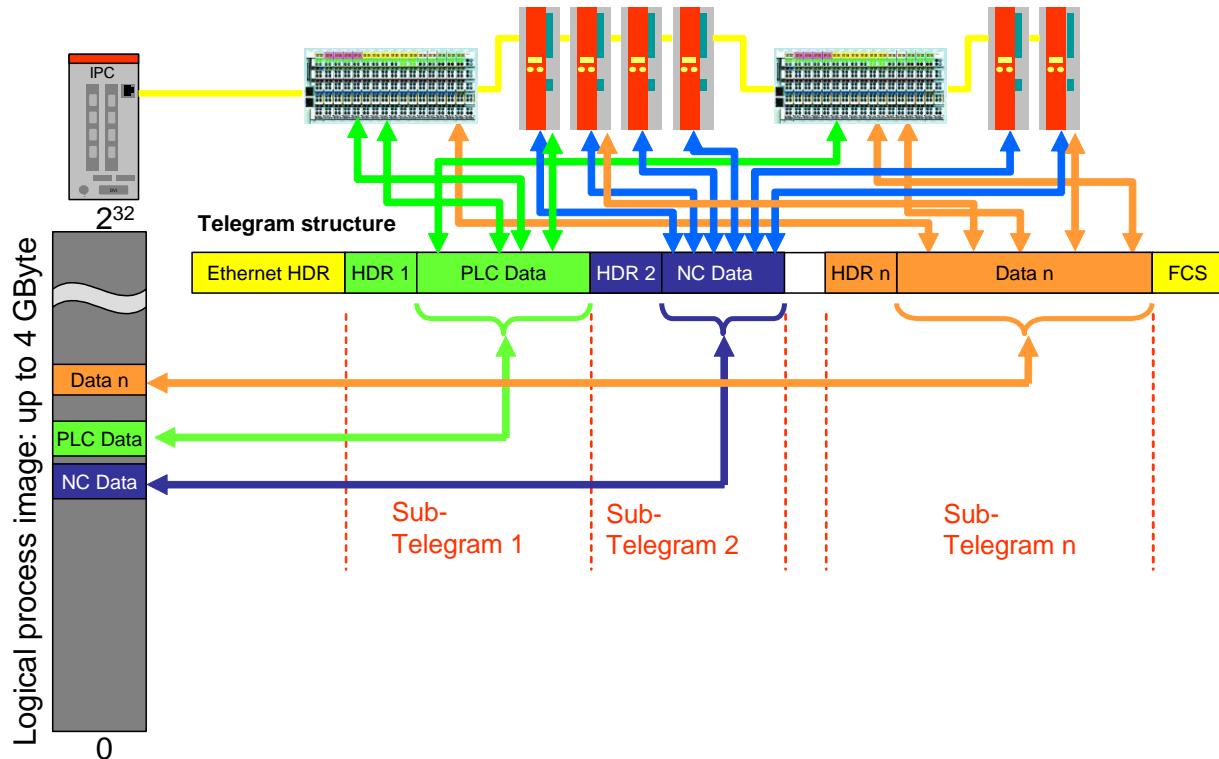


Figure 21: FMMU Mapping Principle

The following example illustrates the functions of an FMMU configured to map 14 bits from logical address 0x00010011.3 to 0x00010013.0 to the physical register bits 0x0F01.1 to 0x0F02.6. The FMMU length is 3 Byte, since the mapped bits span 3 Bytes of the logical address space. Length calculation begins with the first logical byte which contains mapped bits, and ends with the last logical byte which contains mapped bits.

Table 23: Example FMMU Configuration

FMMU configuration register	FMMU reg. offset	Value
Logical Start Address	0x0:0x3	0x00010011
Length (Bytes)	0x4:0x5	3
Logical Start bit	0x6	3
Logical Stop bit	0x7	0
Physical Start Address	0x8:0x9	0x0F01
Physical Start bit	0xA	1
Type	0xB	read and/or write
Activate	0xC	1 (enabled)

NOTE: FMMU configuration registers start at address 0x0600.

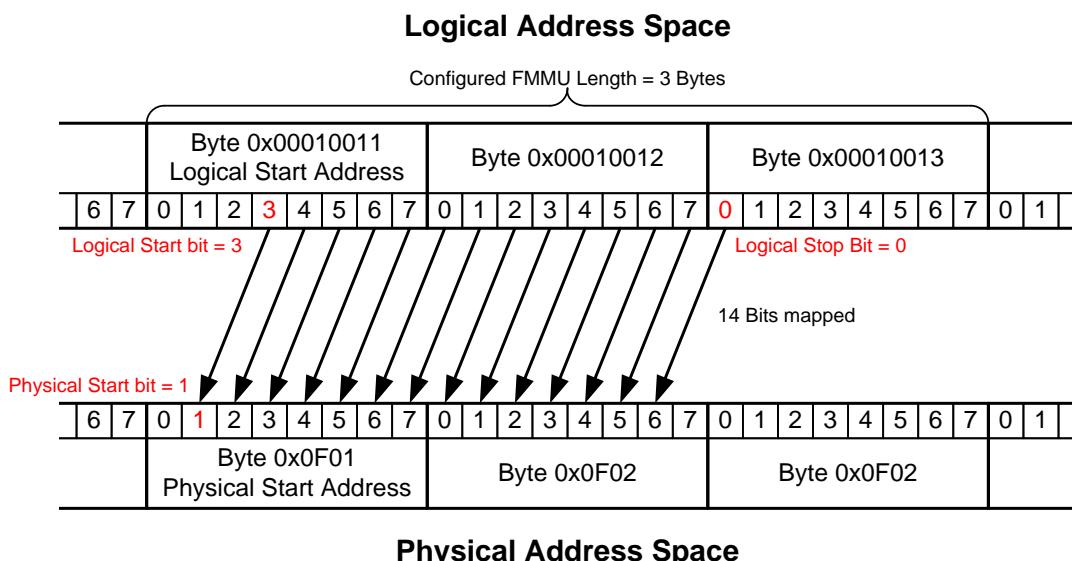


Figure 22: FMMU Mapping Example

Attention: This drawing of the bit string shows the least significant bit first, in a hexadecimal representation of the octets the least significant value is at the right place and the most significant on the left place (00110011 is represented as octet by 0xCC).

Restrictions on FMMU Settings

The FMMUs of Beckhoff ESCs are subject to restrictions. The logical address ranges of two FMMUs of the same direction (read or write) in one ESC must be separated by at least 3 logical bytes not configured by any FMMU of the same type, if one of the FMMUs or both use bit-wise mapping (logical start bit ≠ 0, logical stop bit ≠ 7, or physical start bit ≠ 0). In the above example, the first logical address area after the one shown must have a logical start address of 0x00010017 or higher (the last byte of the example FMMU is 0x00010013, three bytes free 0x00010014-0x00010016).

If only byte-wise mapping is used (logical start bit = 0, logical stop bit = 7, or physical start bit = 0), the logical address ranges can be adjacent.

Bit-wise writing is only supported by the Digital Output register (0x0F00:0x0F03). All other registers and memories are always written byte-wise. If bit-wise mapping is used for writing into these areas, bits without mapping to logical addresses are written with undefined values (e.g., if only physical address bit 0x1000.0 is mapped by a write FMMU, the bits 1-7 are written with undefined values).

Additional FMMU Characteristics

- Each logical address byte can at most be mapped either by one FMMU(read) plus one FMMU(write), or by one FMMU(read/write). If two or more FMMUs (with the same direction – read or write) are configured for the same logical byte, the FMMU with the lower number (lower configuration address space) is used, the other ones are ignored.
- One or more FMMUs may point to the same physical memory, all of them are used. Collisions can not occur.
- It is the same to use one read/write FMMU or two FMMUs – one read, the other one write – for the same logical address.
- A read/write FMMU can not be used together with SyncManagers, since independent read and write SyncManagers can not be configured to use the same (or overlapping) physical address range.
- Bit-wise reading is supported at any address. Bits which are not mapped to logical addresses are not changed in the EtherCAT datagram. E.g., this allows for mapping bits from several ESCs into the same logical byte.
- Reading an unconfigured logical address space will not change the data.

8 SyncManager

The memory of an ESC can be used for exchanging data between the EtherCAT master and a local application (on a µController attached to the PDI) without any restrictions. Using the memory for communication like this has some draw-backs which are addressed by the SyncManagers inside the ESCs:

- Data consistency is not guaranteed. Semaphores have to be implemented in software for exchanging data in a coordinated way.
- Data security is not guaranteed. Security mechanisms have to be implemented in software.
- Both EtherCAT master and application have to poll the memory in order to get to know when the access of the other side has finished.

SyncManagers enable consistent and secure data exchange between the EtherCAT master and the local application, and they generate interrupts to inform both sides of changes.

SyncManagers are configured by the EtherCAT master. The communication direction is configurable, as well as the communication mode (Buffered Mode and Mailbox Mode). SyncManagers use a buffer located in the memory area for exchanging data. Access to this buffer is controlled by the hardware of the SyncManagers.

A buffer has to be accessed beginning with the start address, otherwise the access is denied. After accessing the start address, the whole buffer can be accessed, even the start address again, either as a whole or in several strokes. A buffer access finishes by accessing the end address, the buffer state changes afterwards and an interrupt or a watchdog trigger pulse are generated (if configured). The end address can not be accessed twice inside a frame.

Two communication modes are supported by SyncManagers:

- Buffered Mode
 - The buffered mode allows both sides, EtherCAT master and local application, to access the communication buffer at any time. The consumer gets always the latest consistent buffer which was written by the producer, and the producer can always update the content of the buffer. If the buffer is written faster than it is read out, old data will be dropped.
 - The buffered mode is typically used for cyclic process data.
- Mailbox Mode
 - The mailbox mode implements a handshake mechanism for data exchange, so that no data will be lost. Each side, EtherCAT master or local application, will get access to the buffer only after the other side has finished its access. At first, the producer writes to the buffer. Then, the buffer is locked for writing until the consumer has read it out. Afterwards, the producer has write access again, while the buffer is locked for the consumer.
 - The mailbox mode is typically used for application layer protocols.

The SyncManagers accept buffer changes caused by the master only if the FCS of the frame is correct, thus, buffer changes take effect shortly after the end of the frame.

The configuration registers for SyncManagers are located beginning at register address 0x0800.

Table 24: SyncManager Register overview

Description	Register Address Offset
Physical Start Address	0x0:0x1
Length	0x2:0x3
Control Register	0x4
Status Register	0x5
Activate	0x6
PDI Control	0x7

8.1 Buffered Mode

The buffered mode allows writing and reading data simultaneously without interference. If the buffer is written faster than it is read out, old data will be dropped. The buffered mode is also known as 3-buffer-mode.

Physically, 3 buffers of identical size are used for buffered mode. The start address and size of the first buffer is configured in the SyncManager configuration. The addresses of this buffer have to be used by the master and the local application for reading/writing the data. Depending on the SyncManager state, accesses to the first buffer's (0) address range are redirected to one of the 3 buffers. The memory used for buffers 1 and 2 can not be used and should be taken into account for configuring other SyncManagers.

One buffer of the three buffers is allocated to the producer (for writing), one buffer to the consumer (for reading), and the third buffer keeps the last consistently written data of the producer.

As an example, Figure 23 demonstrates a configuration with start address 0x1000 and Length 0x100. The other buffers shall not be read or written. Access to the buffer is always directed to addresses in the range of buffer 0.

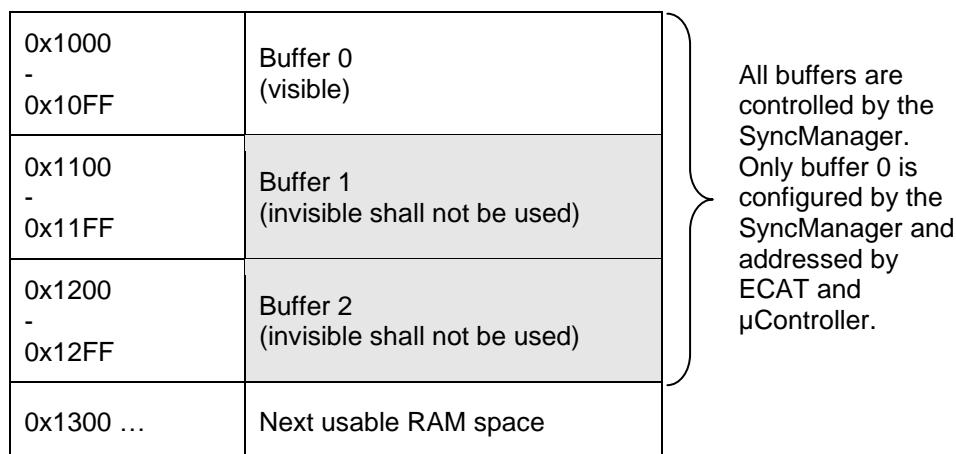


Figure 23: SyncManager Buffer allocation

The buffer interaction is shown in Figure 24:

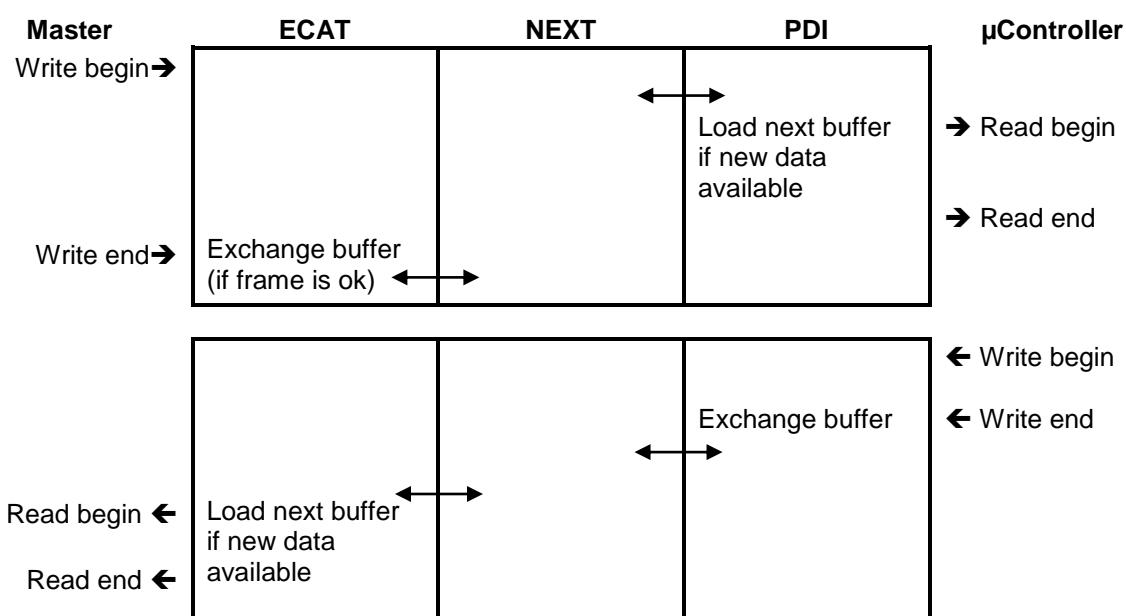


Figure 24: SyncManager Buffered Mode Interaction

The Status register of the SyncManager reflects the current state. The last written buffer is indicated (informative only, access redirection is performed by the ESC), as well as the interrupt states. If the SyncManager buffer was not written before, the last written buffer is indicated to be 3 (start/empty).

8.2 Mailbox Mode

The mailbox mode only allows alternating reading and writing. This assures all data from the producer reaches the consumer. The mailbox mode uses just one buffer of the configured size.

At first, after initialization/activation, the buffer (mailbox, MBX) is writeable. Once it is written completely, write access is blocked, and the buffer can be read out by the other side. After it was completely read out, it can be written again.

The time it takes to read or write the mailbox does not matter.

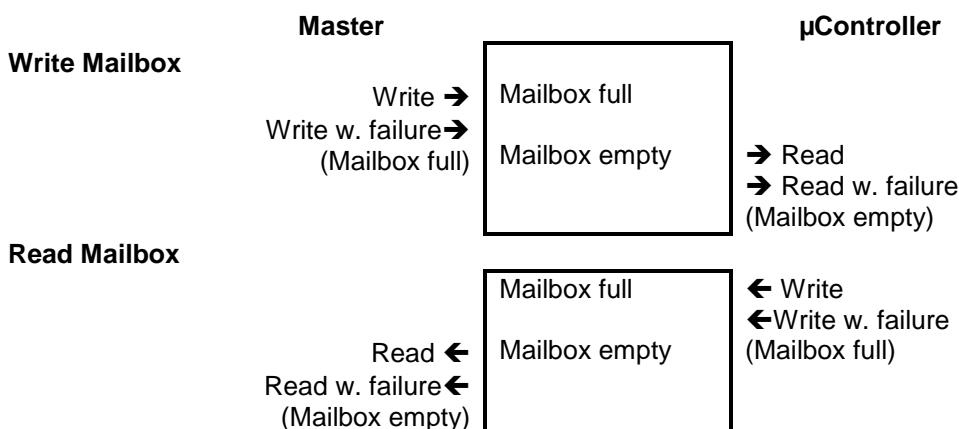


Figure 25: SyncManager Mailbox Interaction

8.2.1 Mailbox Communication Protocols

This is only a brief overview of the mailbox communication protocols. For details on the mailbox protocols refer to the EtherCAT specification ETG.1000.6: "Application layer protocol specification" available from the EtherCAT Technology Group (<http://www.ethercat.org>) or to the IEC specification "Digital data communications for measurement and control – Fieldbus for use in industrial control systems", IEC 61158 Type 12: EtherCAT, available from the IEC (<http://www.iec.ch>).

Ethernet over EtherCAT (EoE)

Defines a standard way to exchange or tunnel standard Ethernet Frames over EtherCAT.

CANopen over EtherCAT (CoE)

Defines a standard way to access a CANopen object dictionary and to exchange CANopen Emergency and PDO messages on an event driven path.

File Access over EtherCAT (FoE)

Defines a standard way to download and upload firmware and other 'files'.

Servo Profile over EtherCAT (SoE)

Defines a standard way to access the IEC 61800-7 identifier.

Vendor specific Profile over EtherCAT (VoE)

A vendor specific protocol follows after a VoE header, that identifies the vendor and a vendor specific type.

ADS over EtherCAT (AoE)

Defines a standard way to exchange Automation Device Specification (ADS) messages over EtherCAT.

The content of an EtherCAT mailbox header is shown in Figure 26.

16 Bit	16 Bit	6 Bit	2 Bit	4 Bit	3 Bit	1 Bit
0 Length	₁₆	Address	₃₂ Channel	₃₈ Prio	₄₀ Type	₄₄ Ctr. ₄₇ 0

Figure 26: EtherCAT Mailbox Header (for all Types)

Table 25: EtherCAT Mailbox Header

Field	Data Type	Value/Description
Length	WORD	Number of Bytes of this mailbox command excluding the mailbox header
Address	WORD	Station Address of originator
Channel	6 bit	0 – reserved for future use
Priority	2 bit	Priority between 0 (lowest) and 3 (highest)
Type	4 bit	Mailbox protocol types: 0x0: Error 0x1: Vendor specific (Beckhoff: AoE – ADS over EtherCAT) 0x2: EoE (Ethernet over EtherCAT) 0x3: CoE (CAN application layer over EtherCAT) 0x4: FoE (File access over EtherCAT) 0x5: SoE (Servo profile over EtherCAT) 0xF: Vendor specific (VoE)
Ctr.	3 bit	Sequence number that is used for detection of duplicated frames
Reserved	1 bit	0

8.3 Interrupt and Watchdog Trigger Generation, Latch Event Generation

Interrupts can be generated when a buffer was completely and successfully written or read. A watchdog trigger signal can be generated to rewind (trigger) the Process Data watchdog used for Digital I/O after a buffer was completely and successfully written. Interrupt and watchdog trigger generation are configurable. The SyncManager Status register reflects the current buffer state.

For debugging purposes it is also possible to trigger Distributed Clock Latch events upon successful buffer accesses (for some ESCs).

8.4 Single Byte Buffer Length / Watchdog Trigger for Digital Output PDI

If a SyncManager is configured for a length of 1 byte (or even 0), the buffer mechanism is disabled, i.e., read/write accesses to the configured address will pass the SyncManager without interference. The additional buffers 1 and 2 are not used in buffered mode, and alternation of write/read accesses is not necessary for mailbox mode. Consistency is not an issue for a single byte buffer.

Nevertheless, watchdog generation is still possible if the buffer length is 1 byte (interrupt generation as well).

NOTE: For some ESCs in Mailbox mode, watchdog and interrupt generation are depending on the alternation of write and read accesses, although the write/read accesses itself are executed without interference. I.e., Buffered mode should be used for single byte buffers for watchdog generation.

Watchdog trigger generation with single byte SyncManagers is used for Digital Outputs, because the outputs are only driven with output data if the Process Data watchdog is triggered. One SyncManager has to be configured for each byte of the Digital Output register (0x0F00:0x0F03) which is used for outputs. The SyncManagers have to be configured like this:

- Buffered Mode (otherwise the Watchdog will not be generated with some ESCs upon the second and following writes, because the Digital I/O PDI does not read the addresses)
- Length of 1 byte
- EtherCAT write / PDI read
- Watchdog Trigger enabled

For more details refer to the Digital I/O PDI description of Section III and the chapters about Watchdog and Digital Output in this document.

NOTE: A SyncManager with length 0 behaves like a disabled SyncManager. It does not interfere with accesses nor generate interrupt or watchdog trigger signals.

8.5 Repeating Mailbox Communication

A lost datagram with mailbox data is handled by the application layer. The Repeat Request/Repeat Acknowledge bits in the SyncManager Activation register (offset 0x06.1) and the PDI Control register (offset 0x07.1) are used in mailbox mode for retransmissions of buffers from a slave to the master. If a mailbox read frame gets lost/broken on the way back to the master, the master can toggle the Repeat Request bit. The slave polls this bit or receives an interrupt (SyncManager activation register changed, register 0x0220.4) and writes the last buffer again to the SyncManager. Then the PDI toggles the Repeat Acknowledge bit in the PDI Control register. The master will read out this bit and read the buffer content. Communication resumes afterwards.

This mechanism is shown in Figure 27 for a mailbox write service. The Mailbox confirmation is lost on its way from the slave to the master and has to be repeated again.

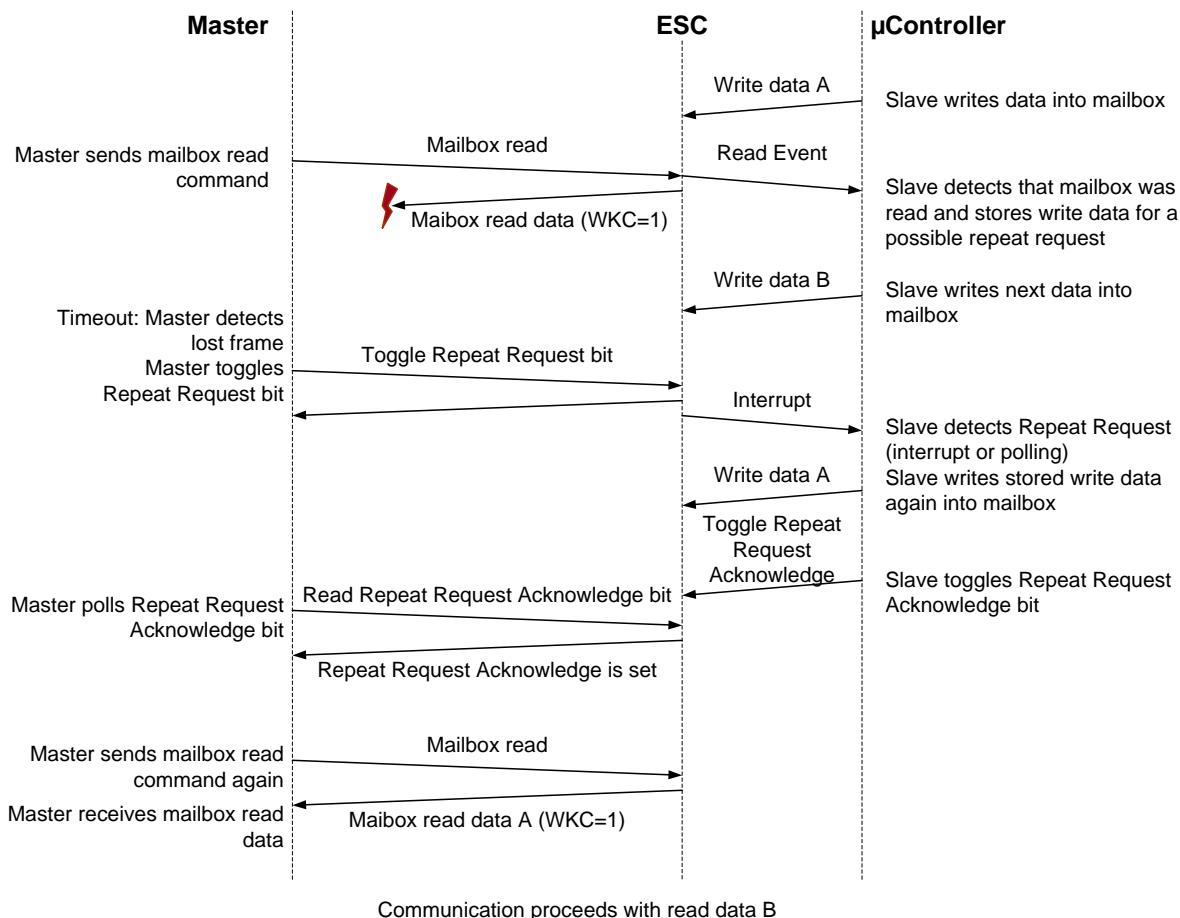


Figure 27: Handling of Write/Read Toggle with Read Mailbox

8.6 SyncManager Deactivation by the PDI

A SyncManager can be deactivated by the PDI to inform the master of local problems (typically used in buffered mode only). The master can detect SyncManager deactivation by checking the Working Counter, which is not incremented if a deactivated SyncManager buffer is accessed. If a SyncManager is deactivated by the PDI (PDI Control register 0x7.0=1), the state of the SyncManager is reset, interrupts are cleared and the SyncManager has to be written first after re-activation. The entire SyncManager buffer area is read/write protected while the SyncManager is deactivated by the PDI.

9 Distributed Clocks

The Distributed Clocks (DC) unit of EtherCAT slave controllers supports the following features:

- Clock synchronization between the slaves (and the master)
- Generation of synchronous output signals (SyncSignals)
- Precise time stamping of input events (LatchSignals)
- Generation of synchronous interrupts
- Synchronous Digital Output updates
- Synchronous Digital Input sampling

9.1 Clock Synchronization

DC clock synchronization enables all EtherCAT devices (master and slaves) to share the same EtherCAT System Time. The EtherCAT devices can be synchronized to each other, and consequently, the local applications are synchronized as well.

For system synchronization all slaves are synchronized to one Reference Clock. Typically, the first ESC with Distributed Clocks capability after the master within one segment holds the reference time (System Time). This System Time is used as the reference clock to synchronize the DC slave clocks of other devices and of the master. The propagation delays, local clock sources drift, and local clock offsets are taken into account for the clock synchronization.

The ESCs can generate SyncSignals for local applications to be synchronized to the EtherCAT System Time. SyncSignals can be used directly (e.g., as interrupts) or for Digital Output updating/Digital Input sampling. Additionally, LatchSignals can be time stamped with respect to the EtherCAT System Time.

Definition of the System Time

- Beginning on January, 1st 2000 at 0:00h
- Base unit is 1 ns
- 64 bit value (enough for more than 500 years)
- Lower 32 bits span over 4.2 seconds (typically enough for communication and time stamping). Some ESCs only have 32 bit DCs, which are compatible with 64 bit DCs.

Definition of the Reference Clock

One EtherCAT device will be used as a Reference Clock. Typically, the Reference Clock is the first ESC with DC capability between master and all the slaves to be synchronized (DC slaves). The Reference Clock might be adjusted to a “global” reference clock, e.g. to an IEEE 1588 grandmaster clock. The reference clock provides the System Time.

Definition of the Local Clock

Each DC slave has a local clock, initially running independent of the Reference Clock. The difference between local clock and Reference Clock (offset) can be compensated, as well as clock drifts. The offset is compensated by adding it to the local clock value. The drift is compensated by measuring and adjusting the local clock speed.

Each DC slave holds a copy of the Reference Clock, which is calculated from the local clock and the local offset. The Reference Clock has a local clock, too.

Definition of the Master Clock

The Reference Clock is typically initialized by the EtherCAT master using the master clock to deliver the System Time according to the System Time definition. The EtherCAT master clock is typically bound to a global clock reference (RTC or the master PC, IEEE1588, GPS, etc.), which is either directly available to the master or indirect by an EtherCAT slave providing access to the reference.

Propagation Delay

The propagation delay between Reference Clock and slave clock has to be taken into account when the System Time is distributed to the slaves.

Offset

The offset between local clock and Reference Clock has two reasons: the propagation delay from the ESC holding the Reference Clock to the device with the slave clock, and initial differences of the local times resulting from different times at which the ESCs have been powered up. This offset is compensated locally in each slave.

The ESC holding the Reference Clock derives the System Time from its local time by adding a local offset. This offset represents the difference between local time (started at power-up) and master time (starting at January, 1st 2000 at 0:00h).

Drift

Since Reference Clock and DC slaves are typically not sourced by the same clock source (e.g. a quartz), their clock sources are subject to small deviations of the clock periods. The result is that one clock is running slightly faster than the other one, their Local Clocks are drifting apart.

ESC Classification regarding DC Support

Three classes of ESCs are distinguished regarding Distributed Clocks support:

1. Slaves supporting System Time/Time Loop Control Unit:
Receive time stamps and System Time/Time Loop Control Unit available; SyncSignal generation, LatchSignal time stamping, and SyncManager Event Times are optionally supported depending on application.
2. Slaves supporting only propagation delay measurement:
Mandatory for ESCs with 3 or more ports (topology devices like EK1100 and ET1100). Local clock and receive time stamps are supported.
3. Slaves without Distributed Clocks support:
Slaves with max. 2 ports do not have to support DC features. Processing/forwarding delay of such slaves is treated like a "wire delay" by the surrounding DC capable slaves.

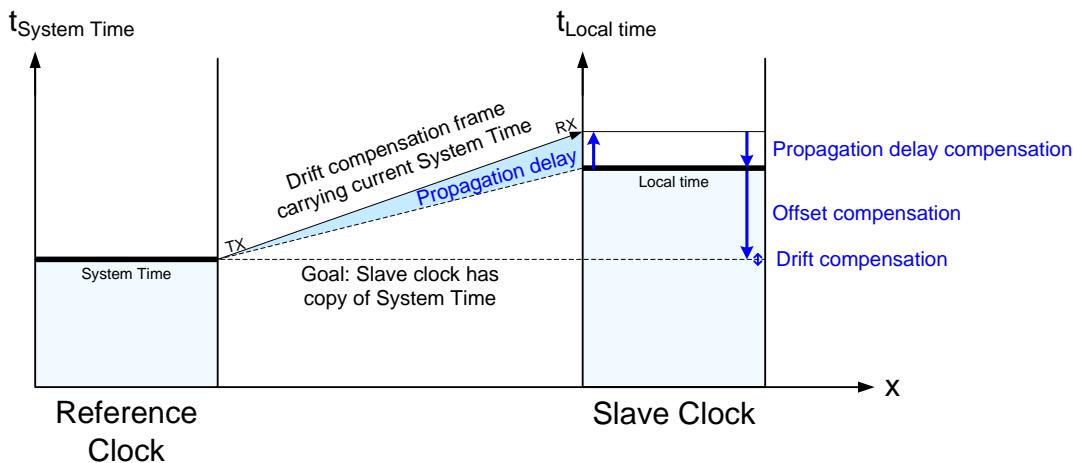
9.1.1 Clock Synchronization Process

The clock synchronization process consists of three steps:

1. Propagation Delay Measurement:
The master initiates propagation delay measurement between all slaves in all directions. Each EtherCAT slave controller measures the receive time of the measurement frame. The master collects the time stamps afterwards and calculates the propagation delays between all slaves.
2. Offset compensation to Reference Clock (System Time):
The local time of each slave clock is compared to the System Time. The difference is compensated individually by writing it to each slave. All devices get the same absolute System Time.
3. Drift compensation to Reference Clock:
The drift between Reference Clock and local clock has to be compensated by regularly measuring the differences and readjusting the local clocks.

The following figure illustrates the compensation calculations for two cases, in the first case the System Time is smaller than the slave's local time, in the second case, it is the other way around.

System Time < Local Time



System Time > Local Time

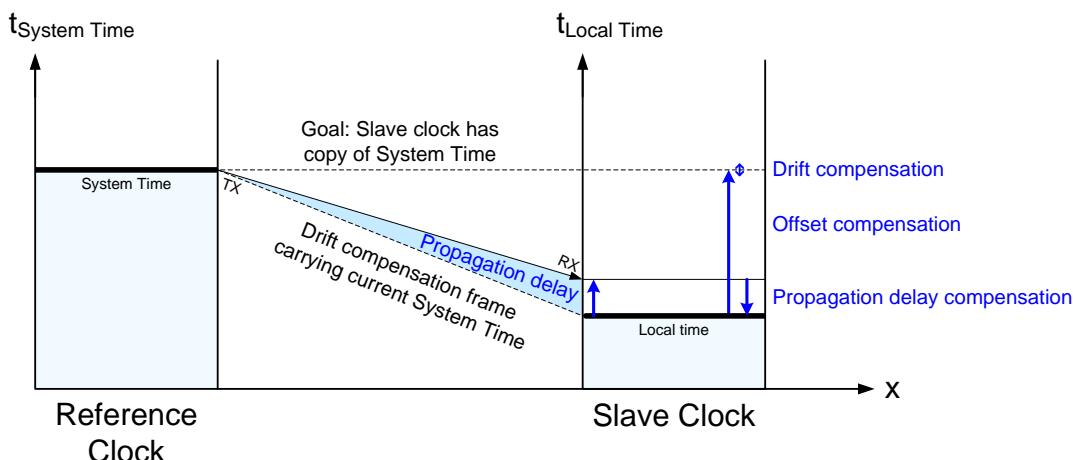


Figure 28: Propagation Delay, Offset, and Drift Compensation

9.1.2 Propagation Delay Measurement

Since each slave introduces a small processing/forwarding delay in each direction (within the device and also in the physical layer), as well as the cable between the ESCs has a delay, the propagation delay between Reference Clock and the respective slave clock has to be considered for the synchronization of the slave clocks.

1. For measuring the propagation delay, the master sends a broadcast write to register DC Receive Time Port 0 (at least first byte).
2. Each slave device stores the time of its local clock when the first bit of the Ethernet preamble of the frame was received, separately for each port (Receive Time Port 0-3 registers).
3. The master reads all time stamps and calculates the delay times with respect to the topology. The delay time between Reference Clock and the individual slave is written to slave's System Time Delay register (0x0928:0x092B).

The receive time registers are used to sample the receive time of a specific frame (a broadcast write to Receive Time Port 0 register).

The clocks must not be synchronized for the delay measurement, only local clock values are used. Since the local clocks of the slaves are not synchronized, there is no relation between the Receive Times of different slaves. So the propagation delay calculation has to be based on receive time differences between the ports of a slave.

Devices with two ports do not need to support Distributed Clocks at all, their delay is treated to be an additional "wire delay" between the surrounding DC capable slaves. Devices with more than 2 ports have to support at least propagation delay measurements (DC Receive Times).

NOTE: Some ESCs use the broadcast write to Receive Time Port 0 register as an indicator to latch the receive times of the next frame at all ports other than port 0 (if port 0 is open). Thus, another frame which is still traveling around the ring might trigger the measurement, and the receive times do not correlate. For these ESCs, the ring has to be empty before the broadcast write is issued. Refer to Section II Receive Time Port x registers for further information.

Registers used for Propagation Delay Measurement are listed in Table 26.

Table 26: Registers for Propagation Delay Measurement

Register Address	Name	Description
0x0900:0x0903	Receive Time Port 0	Local time when receiving frame on Port 0
0x0904:0x0907	Receive Time Port 1	Local time when receiving frame on Port 1
0x0908:0x09B	Receive Time Port 2	Local time when receiving frame on Port 2
0x090C:0x090F	Receive Time Port 3	Local time when receiving frame on Port 3
0x0918:0x091F	Receive Time ECAT Processing Unit	Local time when receiving frame at the ECAT Processing Unit

9.1.2.1 Propagation Delay Measurement Example

The propagation delay between the local device and the Reference Clock device is calculated for the network example shown in Figure 29. The example assumes that slave A is the Reference Clock. The loops of slave D and F are closed internally. The wire delays are assumed to be symmetrical, and the processing and forwarding delays are assumed to be identical for all ESCs.

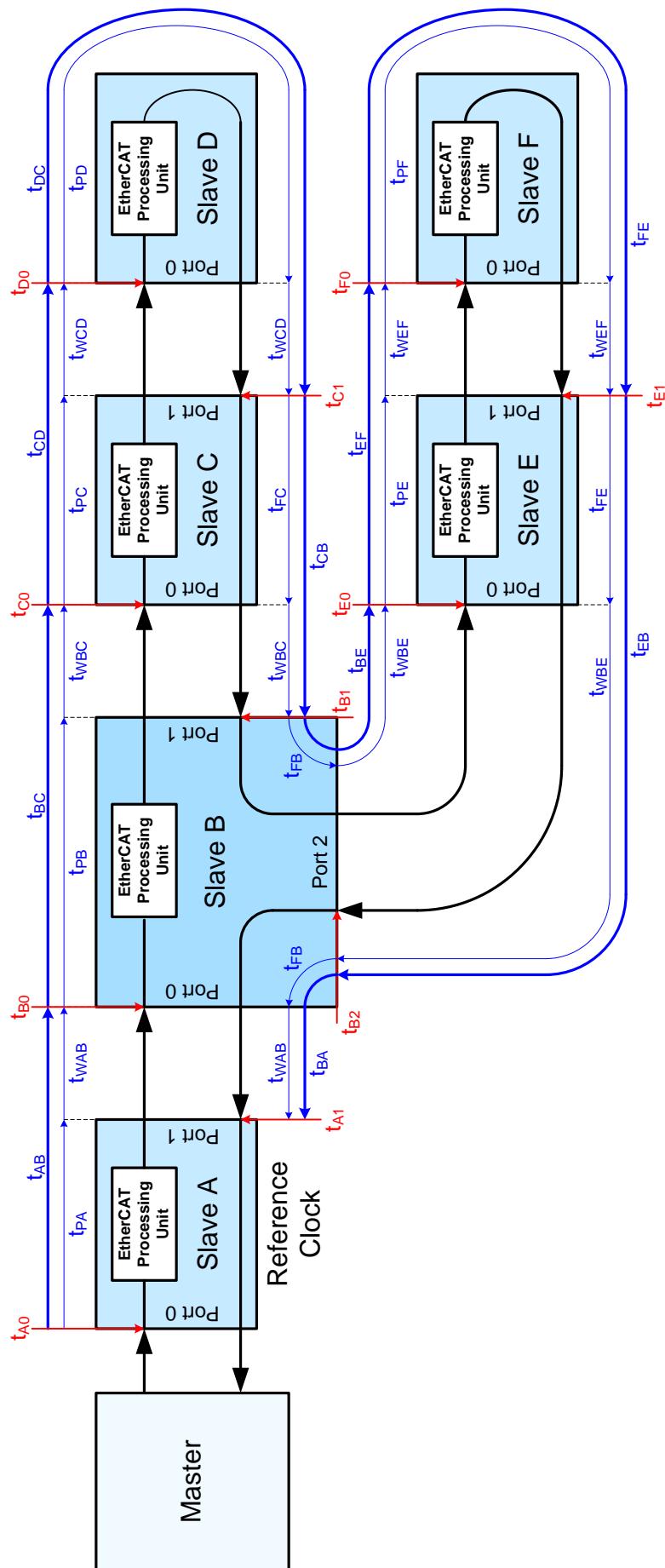


Figure 29: Propagation Delay Calculation

Parameters used for propagation delay calculation are listed in Table 27:

Table 27: Parameters for Propagation Delay Calculation

Parameter	Description
t_{Px}	Processing delay of slave x (through EtherCAT Processing Unit, x=A-F)
t_{Fx}	Forwarding delay of slave x (alongside EtherCAT Processing Unit, x=A-F)
t_{xy}	Propagation delay from slave x to slave y (x/y=A-F)
t_{Wxy}	Wire propagation delay between slaves x and y (assumed to be symmetrical in both directions, x/y=A-F)
t_{x0}, t_{x1}, t_{x2}	Receive Time Port 0/1/2 values of slave x (time when first preamble bit is detected, x=A-F), measured with a write access to DC Receive Time 0 register.
t_p	Processing delay (through EtherCAT Processing Unit) if all slaves are identical
t_f	Forwarding delay (alongside EtherCAT Processing Unit) if all slaves are identical
t_{Diff}	Difference between Processing delay and forwarding delay $t_{Diff} = t_p - t_f$ if all slaves are identical. ESC specific information, part of the ESI. Refer to Section III for actual figures.
t_{Ref_x}	Propagation delay from Reference Clock (slave A) to slave x

Propagation delay between Slave C and D

The propagation delays between slave C and D (t_{CD} and t_{DC}) consist of a processing delay and the wire delay:

$$t_{CD} = t_{PC} + t_{WCD}$$

$$t_{DC} = t_{PD} + t_{WCD}$$

Assuming that the processing delays of slave C and D are identical ($t_p = t_{PC} = t_{PD}$):

$$t_{CD} = t_{DC} = t_p + t_{WCD}$$

The two Receive Times of slave C have the following relation:

$$t_{C1} = t_{C0} + t_{CD} + t_{DC}$$

So the propagation delays between slave C and D are

$$t_{CD} = t_{DC} = (t_{C1} - t_{C0}) / 2$$

Propagation delay between Slave B and C

The propagation delays between slave B and C (t_{BC} and t_{CB}) are calculated as follows:

$$t_{BC} = t_{PB} + t_{WBC}$$

$$t_{CB} = t_{FC} + t_{WBC}$$

Assuming that the processing delays of slaves B, C and D are identical ($t_p = t_{PB} = t_{PC} = t_{PD}$), and the difference between forwarding and processing delay of slave C is $t_{Diff} = t_{PC} - t_{FC}$:

$$t_{BC} = t_p + t_{WBC}$$

$$t_{CB} = t_{BC} - t_{Diff}$$

The Receive Times (port 0 and 1) of slave B have the following relation:

$$t_{B1} = t_{B0} + t_{BC} + t_{CD} + t_{DC} + t_{CB}$$

So the propagation delay between slave B and C is

$$2*t_{BC} - t_{Diff} = (t_{B1} - t_{B0}) - (t_{C1} - t_{C0})$$

$$t_{BC} = ((t_{B1} - t_{B0}) - (t_{C1} - t_{C0}) + t_{Diff}) / 2$$

And for the other direction:

$$t_{CB} = ((t_{B1} - t_{B0}) - (t_{C1} - t_{C0}) - t_{Diff}) / 2$$

Propagation delay between Slave E and F

The propagation delays between slave E and F are calculated like the delays between slave C and D:

$$t_{EF} = t_{PE} + t_{WEF}$$

$$t_{FE} = t_{PF} + t_{WEF}$$

Assuming that the processing delays of slave E and F are identical ($t_P = t_{PE} = t_{PF}$):

$$t_{EF} = t_{FE} = (t_{E1} - t_{E0}) / 2$$

Propagation delay between Slave B and E

The propagation delays between slave B and E (t_{BE} and t_{EB}) are calculated as follows:

$$t_{BE} = t_{FB} + t_{WBE}$$

$$t_{EB} = t_{FE} + t_{WBE}$$

Assuming that the processing delays of slaves B to F are identical ($t_P = t_{Px}$), and the difference between forwarding and processing delay of these slaves is $t_{Diff} = t_{Px} - t_{Fx}$:

$$t_{BE} = t_{EB} = t_P - t_{Diff} + t_{WBE}$$

The Receive Times Port 1 and 2 of slave B have the following relation:

$$t_{B2} = t_{B1} + t_{BE} + t_{EF} + t_{FE} + t_{EB}$$

So the propagation delay between slave B and E is

$$2*t_{BE} = (t_{B2} - t_{B1}) - t_{EF} - t_{FE}$$

$$t_{BE} = t_{EB} = ((t_{B2} - t_{B1}) - (t_{E1} - t_{E0})) / 2$$

Propagation delay between Slave A and B

The propagation delays between slave A and B are calculated as follows:

$$t_{AB} = t_{PA} + t_{WAB}$$

$$t_{BA} = t_{FB} + t_{WAB}$$

Assuming that the processing delays of all slaves are identical ($t_P = t_{Px}$), and the difference between forwarding and processing delay of these slaves is $t_{Diff} = t_{Px} - t_{Fx}$:

$$t_{AB} = t_P + t_{WAB}$$

$$t_{BA} = t_{AB} - t_{Diff}$$

The Receive Times of slave A have the following relation:

$$t_{A1} = t_{A0} + t_{AB} + (t_{B1} - t_{B0}) + (t_{B2} - t_{B1}) + t_{BA}$$

So the propagation delay between slave A and B is

$$t_{AB} = ((t_{A1} - t_{A0}) - (t_{B2} - t_{B0}) + t_{Diff}) / 2$$

And for the other direction:

$$t_{BA} = ((t_{A1} - t_{A0}) - (t_{B2} - t_{B0}) - t_{Diff}) / 2$$

Summary of Propagation Delay Calculation between Slaves

$$t_{AB} = ((t_{A1} - t_{A0}) - (t_{B2} - t_{B0}) + t_{Diff}) / 2$$

$$t_{BA} = ((t_{A1} - t_{A0}) - (t_{B2} - t_{B0}) - t_{Diff}) / 2$$

$$t_{BC} = ((t_{B1} - t_{B0}) - (t_{C1} - t_{C0}) + t_{Diff}) / 2$$

$$t_{CB} = ((t_{B1} - t_{B0}) - (t_{C1} - t_{C0}) - t_{Diff}) / 2$$

$$t_{CD} = t_{DC} = (t_{C1} - t_{C0}) / 2$$

$$t_{EF} = t_{FE} = (t_{E1} - t_{E0}) / 2$$

$$t_{BE} = t_{EB} = ((t_{B2} - t_{B1}) - (t_{E1} - t_{E0})) / 2$$

Propagation Delays between Reference Clock and Slave Clocks

The System Time Delay register of each slave clock takes the propagation delay from the Reference Clock to the slave. This delay is calculated like this:

$$t_{Ref_B} = t_{AB}$$

$$t_{Ref_C} = t_{AB} + t_{BC}$$

$$t_{Ref_D} = t_{AB} + t_{BC} + t_{CD}$$

$$t_{Ref_E} = t_{AB} + t_{BC} + t_{CD} + t_{DC} + t_{CB} + t_{BE}$$

$$t_{Ref_F} = t_{AB} + t_{BC} + t_{CD} + t_{DC} + t_{CB} + t_{BE} + t_{EF}$$

9.1.3 Offset Compensation

The local time of each device is a free running clock which typically will not have the same time as the Reference Clock. To achieve the same absolute System Time in all devices, the offset between the Reference Clock and every slave device's clock is calculated by the master. The offset time is written to register System Time Offset to adjust the local time for every individual device. Small offset errors are eliminated by the drift compensation after some time, but this time might become extremely high for large offset errors – especially for 64 bit DCs.

Each DC slave calculates its local copy of the System time using its local time and the local offset value:

$$t_{Local\ copy\ of\ System\ Time} = t_{Local\ time} + t_{Offset}$$

This time is used for SyncSignal generation and time stamping of LatchSignals. It is also provided to the PDI for use by µControllers.

The System Time of the Reference Clock is bound to the master clock by calculating the difference and compensating it using the System Time Offset of the Reference Clock.

Registers used for Offset Compensation are listed in Table 28.

Table 28: Registers for Offset Compensation

Register Address	Name	Description
0x0910:0x0917	System Time	Local copy of System Time (read from PDI)
0x0918:0x091F	Receive Time ECAT Processing Unit	Local Time ($t_{Local\ time}$)
0x0920:0x0927	System Time Offset	Difference between local time and System Time

9.1.4 Drift Compensation

After the delay time between the Reference Clock and the slave clocks has been measured, and the offset between both clocks has been compensated, the natural drift of every local clock (emerging from quartz variations between Reference Clock's quartz and local quartz) is compensated by the time control loop which is integrated within each ESC.

For drift compensation, the master distributes the System Time from the Reference Clock to all slave clocks periodically. The ARMW or FRMW commands can be used for this purpose. The time control loop of each slave takes the lower 32 bit of the System Time received from the Reference Clock and compares it to its local copy of the System Time. For this difference, the propagation delay has to be taken into account:

$$\Delta t = (t_{\text{Local time}} + t_{\text{Offset}} - t_{\text{Propagation delay}}) - t_{\text{Received System Time}}$$

If Δt is positive, the local time is running faster than the System time, and has to be slowed down. If Δt is negative, the local time is running slower than the System time, and has to be sped up. The time control loop adjusts the speed of the local clock.

For a fast compensation of the static deviations of the clock speeds, the master should initially send many ARMW/FRMW commands (e.g. 15,000) for drift compensation in separate frames after initialization of the propagation delays and offsets. The control loops compensate the static deviations and the distributed clocks are synchronized. Afterwards, the drift compensation frames are sent periodically for compensation of dynamic clock drifts.

NOTE: The System Time Offset allows fast compensation of differences between local copy of the system time and the System Time, the drift compensation is very slow. Thus, shortly before drift compensation is started, the offset should be roughly compensated using the System Time Offset register. Otherwise settling time might become very high.

Time Control Loop Configuration and Status

The time control loop has some configuration and status registers (System Time Difference, Speed Counter Start, Speed Counter Difference, System Time Difference Filter Depth, and Speed Counter Filter Depth). The default settings of these registers are sufficient for proper operation of the drift compensation. Setting the Speed Counter Filter Depth (0x0935) to 0 improves control loop behavior.

The System Time Difference register (0x092C:0x092F) contains the mean value of the difference between local copy of the System Time and the System Time (Δt). This value converges to zero when both times are identical.

The Speed Counter Start register (0x0930:0x0931) represents the bandwidth of the drift compensation. The value of the Speed Counter Difference register (0x0932:0x0933) represents the deviation between the clock periods of the Reference Clock and the local ESC.

The System Time Difference Filter Depth register (0x0934) and the Speed Counter Filter Depth register (0x0935) set filter depths for mean value calculation of the received System Times and of the calculated clock period deviations.

Registers used for Control Loop/Drift Compensation are listed in Table 29.

Table 29: Registers for Drift Compensation

Register Address	Name	Description
0x0900:0x090F	Receive Time Port n	Local time when receiving frame on Port n
0x0918:0x091F	Receive Time ECAT Processing Unit	Local time when receiving frame for ECAT Processing Unit
0x0910:0x0917	System Time	Local copy of System Time (read from PDI) (local time if System Time Offset=0)
0x0920:0x0927	System Time Offset	Time difference between System Time and local time
0x0928:0x092B	System Time Delay	Delay between Reference Clock and the ESC
0x092C:0x092F	System Time Difference	Mean difference between local copy of System Time and received System Time values
0x0930:0x0931	Speed Counter Start	Bandwidth for adjustment of local copy of System Time
0x0932:0x0933	Speed Counter Diff	Deviation between local clock period and Reference Clock's clock period
0x0934	System Time Difference Filter Depth	Filter depth for averaging the received System Time deviation
0x0935	Speed Counter Filter Depth	Filter depth for averaging the clock period deviation

9.1.5 Reference between DC Registers/Functions and Clocks

Table 30: Reference between DC Registers/Functions and Clocks

Register Address	Name	Referring to clock
0x0900:0x090F	Receive Time Port n	Local Time ($t_{\text{Local time}}$)
0x0918:0x091F	Receive Time ECAT Processing Unit	Local Time ($t_{\text{Local time}}$)
0x0910:0x0917	System Time	ECAT: Local copy of System Time when frame passed Reference Clock ($t_{\text{Local time}} + t_{\text{Offset}} - t_{\text{Propagation delay}}$) PDI: Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x0990:0y0997	SYNC0 Start Time	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x0998:0x099F	NEXT SYNC1 Pulse	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x09B0:0x09B7	Latch0 Time Positive Edge	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x09B8:0x09BF	Latch0 Time Negative Edge	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x09C0:0x09C7	Latch1 Time Positive Edge	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x09C8:0x09CF	Latch1 Time Negative Edge	Local copy of System Time ($t_{\text{Local time}} + t_{\text{Offset}}$)
0x09F0:0x09F3	EtherCAT Buffer Change Event Time	Local Time ($t_{\text{Local time}}$)
0x09F8:0x09FB	PDI Buffer Start Event Time	Local Time ($t_{\text{Local time}}$)
0x09FC:0x09FF	PDI Buffer Change Event Time	Local Time ($t_{\text{Local time}}$)

9.1.6 When is Synchronization established?

There are two possibilities to detect if DC synchronization of a slave is established:

- Read System Time Difference (0x92C:0x092F):
If the difference is below an application specific threshold, DC has locked.
Advantage: Can be read using a single BRD command for the entire network: if the upper N bits are zero, synchronization is established.
Recommended if an EtherCAT slave is the reference clock. If the master is the reference clock, the threshold has to be increased to accomplish for the master jitter, which could make this solution unusable.
- Read Speed Counter Difference (0x0932:0x0933):
If the value is stable (within an application specific range), DC has locked.
Disadvantage: Loss of lock is recognized late.

9.1.7 Clock Synchronization Initialization Example

The initialization procedure of clock synchronization including propagation delay measurement, offset compensation and drift compensation is shown in the following. After initialization, all DC slaves are synchronized with the Reference Clock.

1. Master reads the DL Status register of all slaves and calculates the network topology.
2. Master sends a broadcast write to Receive Time Port 0 register (at least first byte). All slaves latch the local time of the first preamble bit of this frame at all ports and at the ECAT Processing Unit. Some ESCs need the EtherCAT network to be free of frames before the broadcast write is sent.
3. Master waits until the broadcast write frame has returned.
4. Master reads all Receive Time Port 0-3 registers (depending on the topology) and the Receive Time ECAT Processing Unit register (0x0918:0x091F) which contains the upper 32 bits of the receive times.
5. Master calculates individual propagation delays and writes them to System Time Delay registers of the slaves. Possible overruns of the 32 bit Receive Times have to be checked and taken into account.
6. Master sets System Time Offset register of the Reference Clock so that the Reference Clock is bound to the master time. The offset for the Reference Clock is master time minus Receive Time ECAT Processing Unit (local time) of the Reference Clock.
7. Master calculates System Time offsets for all DC slaves and writes them to the System Time Offset registers. The offset of each slave is Receive Time ECAT Processing Unit from Reference Clock minus Receive Time ECAT Processing Unit from each DC slave.
8. For static drift compensation, the master sends many separate ARMW or FRMW drift compensation frames (e.g., 15,000 frames) to distribute the System Time of the Reference Clock to all DC slaves.
9. For dynamic drift compensation, the master sends ARMW or FRMW commands periodically to distribute the System Time of the Reference Clock to all DC slaves. The rate of the drift compensation commands depends on the acceptable maximum deviation.

9.2 SyncSignals and LatchSignals

ESCs with Distributed Clocks support generation of SyncSignals and time stamping of LatchSignals. The SyncSignals can be used internally for

- Interrupt generation (mapping to AL Event Request register 0x0220:0x0223 and PDI IRQ)
- PDI Digital Output Update events
- PDI Digital Input Latch events

The SyncSignals can also be directly mapped to output signals (SYNC[1:0]) for use by external devices, e.g., as interrupt signals (less jitter than PDI IRQ, no interrupt source decoding).

The Latch Event unit supports time stamping of up to two LatchSignals (LATCH[1:0], rising and falling edge separately), and time stamping of SyncManager events for debugging purposes.

9.2.1 Interface

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

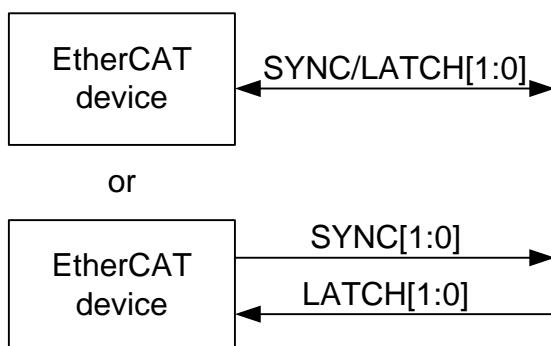


Figure 30: Distributed Clocks signals

Table 31: Distributed Clocks signals

Signal	Direction	Description
SYNC/LATCH[1:0]	IN/OUT	Combined SyncSignals / LatchSignals
or (ESC dependent)		
SYNC[1:0]	OUT	SyncSignals (also named SYNC0/SYNC1)
LATCH[1:0]	IN	LatchSignals (also named LATCH0/LATCH1)

Not all of these signals might be available depending on the ESC and its hardware configuration.

9.2.2 Configuration

The mapping of Distributed Clocks SyncSignals and LatchSignals to the external SYNC/LATCH[1:0] signals is controlled by the setting of the Sync/Latch PDI Configuration register 0x0151. The SYNC[1:0] driver characteristics are also selected in this register. The SyncSignals are internally available for interrupt generation and Digital I/O synchronization regardless of the Sync/Latch PDI Configuration. The mapping of SyncSignals to the AL Event Request register is also controlled by the Sync/Latch PDI Configuration register 0x0151.

The length of a SyncSignal pulse is defined in the DC Pulse Length of SYNC Signals register (0x0982:0x0983). A value of 0 selects acknowledged modes.

Some ESCs support power saving options (partly disabling DC units) controlled by two bits of the PDI Control register (0x0140[11:10]).

The Sync/Latch signals are not driven (high-impedance) by some ESCs until the SII EEPROM is successfully loaded. Refer to Section III for details. Take care of proper SyncSignal usage while the EEPROM is not loaded (e.g. pull-down/pull-up resistors).

9.2.3 SyncSignal Generation

The DC Cyclic Unit / Sync Unit supports the generation of two SyncSignals, SYNC0 and SYNC1. The SyncSignals can both be used internally and externally of the ESC. SyncSignals can be generated at a specific System Time. Four operation modes are supported: cyclic generation, single shot, cyclic acknowledge, and single shot acknowledge mode. The acknowledged modes are typically used for interrupt generation. The interrupts have to be acknowledged by a µController.

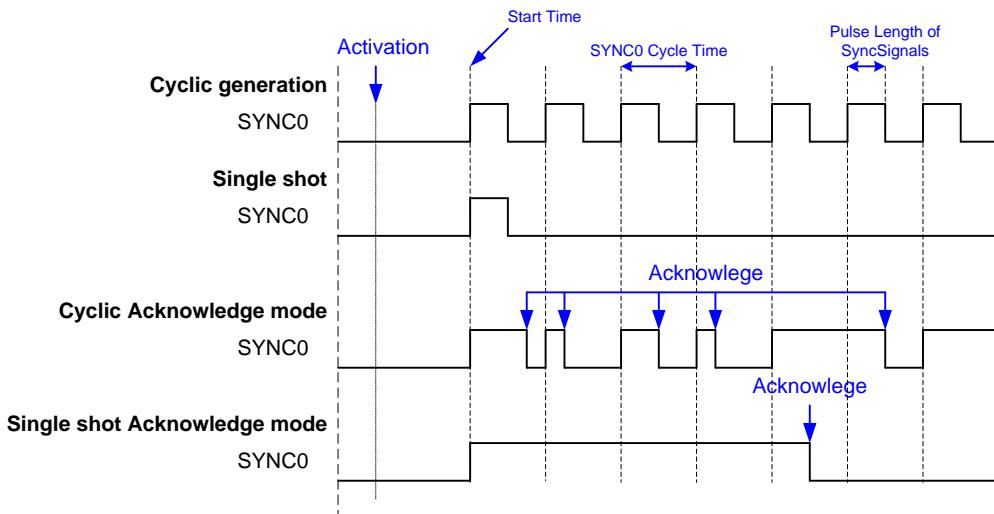


Figure 31: SyncSignal Generation Modes

The SyncSignal operation mode is selected by the configuration of the Pulse Length and the SYNC0 Cycle Time, according to the following table:

Table 32: SyncSignal Generation Mode Selection

Pulse Length of SYNC Signals (0x0982:0x0983)	SYNC0 Cycle Time (0x09A0:0x09A3)	
	> 0	= 0
> 0	Cyclic Generation	Single Shot
= 0	Cyclic Acknowledge	Single Shot Acknowledge

The cycle time of the SYNC0 signal is configured in the SYNC0 Cycle Time register (0x09A0:0x09A3), the start time is set in the Start Time Cyclic Operation register (0x0990:0x0997). After the Sync Unit is activated and the output of the SYNC0/1 signals is enabled (DC Activation register 0x0981), the Sync Unit waits until the start time is reached and generates the first SYNC0 pulse.

Some ESCs support additional activation options like auto-activation when the Start Time is written, or 64 bit extension if only 32 bit of the Start Time is written. Other options are to detect invalid Start Times and provide debug output of SyncSignals.

Internally, the SyncSignals are generated with an update rate of 100 MHz (10 ns update cycle). The jitter of the internal SyncSignal generation in comparison to the System Time is 12 ns.

The registers used for SyncSignal Generation are shown in Table 33.

Table 33: Registers for SyncSignal Generation

Register Address	Name	Description
0x0140[11:10]	PDI Control	Enable/Disable DC Units (power saving)
0x0151	Sync/Latch PDI Configuration	Configuration of SYNC/LATCH[1:0] pins
0x0980.0	Cyclic Unit Control	Assignment of cyclic function to EtherCAT or PDI
0x0981	Activation	Activation of cyclic function and SYNC pins
0x0982:0x0983	Pulse Length of SYNC signals	Length of SYNC impulse length
0x0984	Activation Status	Activation status of SYNC0/SYNC1
0x098E	SYNC0 Status	Status of SYNC0 signal
0x098F	SYNC1 Status	Status of SYNC1 signal
0x0990:0y0997	SYNC0 Start Time	Start System time of cyclic operation
0x0998:0x099F	NEXT SYNC1 Pulse	System Time of next Sync1 Pulse
0x09A0:0x09A3	SYNC0 Cycle Time	Cycle Time of SYNC0
0x09A4:0x09A7	SYNC1 Cycle Time	Cycle Time of SYNC1

NOTE: Some of these registers are set via SII EEPROM/IP Core configuration, or they are not available in specific ESCs. Refer to Section II for details.

9.2.3.1 Cyclic Generation

In Cyclic Generation mode, the Sync unit generates isochronous SyncSignals after the Start Time. The generation ends if the Cyclic Unit is deactivated or SYNC0/1 generation is deactivated. The Cycle times are determined by the SYNC0/1 Cycle Time registers. The Pulse Length of the SYNC signals has to be greater than 0. If the Pulse Length is greater than the Cycle Time, the SyncSignal will always be activated after the Start Time.

9.2.3.2 Single Shot Mode

In Single Shot mode (SYNC0 Cycle Time set to 0), only one SyncSignal pulse is generated after the Start Time is reached. Another pulse can only be generated by deactivating the Cyclic Unit (0x0981.0=0), reprogramming the Start Time, and reactivation of the Cyclic Unit.

9.2.3.3 Cyclic Acknowledge Mode

The Cyclic Acknowledge mode is typically used for generation of isochronous interrupts. The acknowledged modes are selected by setting the Pulse Length of SYNC Signals to 0 (0x0982:0x0983). Each SyncSignal pulse remains active until it is acknowledged – typically by a µController – by reading the appropriate SYNC0 or SYNC1 Status register (0x098E, 0x098F). The first pulse is generated after the Start Time is reached, following pulses are generated when the next regular SYNC0/1 event would occur.

9.2.3.4 Single Shot Acknowledge Mode

In Single Shot Acknowledge mode (both Pulse Length of SYNC Signals and SYNC0 Cycle Time are 0), only one pulse is generated when the Start Time is reached. The pulse remains active until it is acknowledged by reading the appropriate SYNC0/1 Status registers. Another pulse can only be generated by deactivating the Cyclic Unit (0x0981.0=0), reprogramming the Start Time, and reactivation of the Cyclic Unit.

9.2.3.5 SYNC1 Generation

The second SyncSignal (SYNC1) depends on SYNC0, it can be generated with a predefined delay after SYNC0 pulses. The delay is configured in the SYNC1 Cycle Time register (0x09A4:0x09A7).

If the SYNC1 Cycle Time is larger than the SYNC0 Cycle Time, it will be generated as follows: when the Start Time Cyclic Operation is reached, a SYNC0 pulse is generated. The SYNC1 pulse is generated after the SYNC0 pulse with a delay of SYNC1 Cycle Time. The next SYNC1 pulse is generated when the next SYNC0 pulse was generated, plus the SYNC1 Cycle Time.

Some example configurations are shown in the following figure:

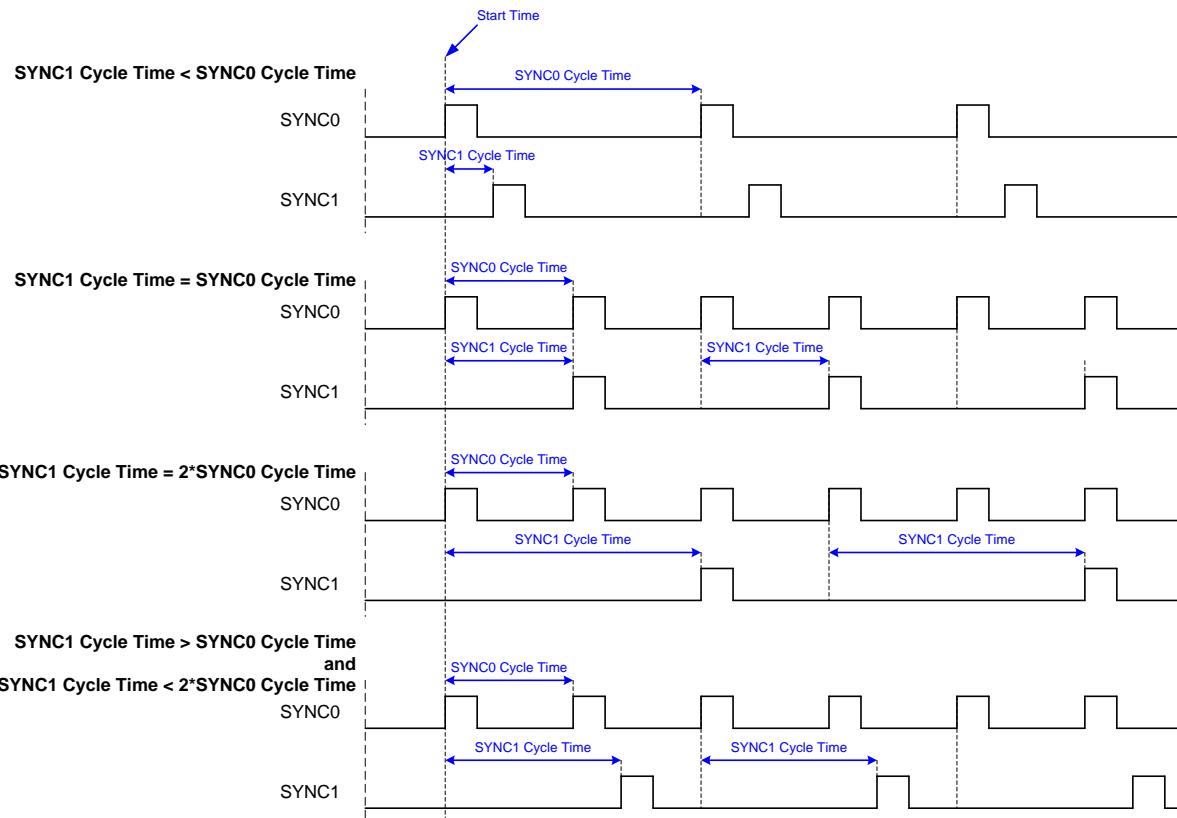


Figure 32: SYNC0/1 Cycle Time Examples

NOTE: If The SYNC1 Cycle Time is 0, SYNC1 reflects SYNC0.

9.2.3.6 SyncSignal Initialization Example

The SyncSignal generation is initialized with the following procedure:

1. Enable DC SYNC Out Unit in PDI Control register (0x0140.10=1; specific ESCs only)
2. Set SYNC/Latch PDI Configuration register (0x0151, initialized by SII EEPROM) to SYNC0/1 output with appropriate driver settings.
3. Set Pulse Length register (0x0982:0x0983, initialized by EEPROM) to pulse length of SYNC signals. Select a value > 0 ns for cyclic repetition of the SyncSignals
4. Assign Sync Unit to ECAT or PDI (0x0980, part of ESI)
5. Set cycle time of SYNC0 signal (0x09A0:0x09A3) and for SYNC1 signal (0x09A4:0x09A7)
6. Set Start Time of Cyclic Operation (0x0990:0x0997) to a time later than the time the cyclic generation will be activated (end of activation frame; e.g., read the System Time and add the time for writing Start Time and Activation). For 32 bit DCs, the SyncSignal generation will start at worst after a turn-over of the System Time (~ 4 s), but with 64 bit DCs, SyncSignal generation may start in hundreds of years.
7. Activate Cyclic Operation (0x0981.0=1) to start cyclic generation of SyncSignals and activate SYNC0/1 generation (0x0981[2:1]=0x3). The Sync Unit waits until the Start Time of Cyclic Operation is reached for the generation of the first SYNC0 pulse.

Register Start Time of Cyclic Operation and register Next SYNC1 pulse can be read to get the time of the next output event. In the acknowledged modes, the Sync0/1 Status registers (0x098E:0x098F) give the status of the SyncSignals. The SyncSignals are acknowledged by reading the SYNC0/1 Status registers.

9.2.4 LatchSignals

The DC Latch Unit enables time stamping of LatchSignal events for two external signals, LATCH0 and LATCH1. Both rising edge and falling edge time stamps are recorded. Additionally, time stamping of SyncManager events is possible with some ESCs.

LatchSignals are sampled with a sample rate of 100 MHz, the corresponding time stamp has an internal jitter of 11 ns.

The state of the LatchSignals can be read from the Latch Status registers (0x09AE:0x09AF) – if supported by the ESC.

The DC Latch Unit support two modes: single event or continuous mode, configured in the Latch0/1 Control registers (0x09A8:0x09A8).

The registers used for LatchSignal event time stamping are shown in Table 34:

Table 34: Registers for Latch Input Events

Register Address	Name	Description
0x0140[11:10]	PDI Control	Enable/Disable DC Units (power saving)
0x0151	Sync/Latch PDI Configuration	Configuration of SYNC/LATCH[1:0] pins
0x0980[5:4]	Cyclic Unit Control	Assignment of cyclic function to EtherCAT or PDI
0x09A8	Latch0 Control	Latch unit configuration for Latch0
0x09A9	Latch1 Control	Latch unit configuration for Latch1
0x09AE	Latch0 Status	Latch status of Latch0
0x09AF	Latch1 Status	Latch status Latch1
0x09B0:0x09B7	Latch0 Time Positive Edge	System time at positive edge Latch0
0x09B8:0x09BF	Latch0 Time Negative Edge	System time at negative edge Latch0
0x09C0:0x09C7	Latch1 Time Positive Edge	System time at positive edge Latch1
0x09C8:0x09CF	Latch1 Time Negative Edge	System time at negative edge Latch1
0x09F0:0x09F3	EtherCAT Buffer Change Event Time	Local time at beginning of frame causing ECAT SyncManager buffer change event
0x09F8:0x09FB	PDI Buffer Start Event Time	Local time at PDI SyncManager buffer start event
0x09FC:0x09FF	PDI Buffer Change Event Time	Local time at PDI SyncManager buffer change event

NOTE: Some of these registers are set via SII EEPROM/IP Core configuration, or they are not available in specific ESCs. Refer to Section II for details.

9.2.4.1 Single Event Mode

In single event mode, only the timestamps of the first rising and the first falling edge of the LatchSignals are recorded. The Latch Status registers (0x09AE:0x09AF) contain information about the events which already have occurred. The Latch Time registers (0x09B0 to 0x09CF) contain the time stamps.

Each event is acknowledged by reading the corresponding Latch Time register. After reading the Latch Time register, the Latch unit is waiting for the next event. Latch events are mapped to the AL Event Request register in single event mode.

9.2.4.2 Continuous Mode

In continuous mode, each event is stored in the Latch Time registers. At reading, the time stamp of the last event is read. The Latch Status registers (0x09AE:0x09AF) do not reflect the latch event states in continuous mode.

9.2.4.3 SyncManager Event

Some ESCs support debugging of SyncManager interactions with time stamps for buffer events. The last event can be read out at the SyncManager Event Time registers (0x09F0:0x09FF), if the SyncManager is configured appropriately.

9.2.5 ECAT or PDI Control

The SyncSignal unit and the two LatchSignal units of the Distributed Clocks entity can be assigned independently by the master to be controlled either by ECAT or a local µController (PDI) using the Cyclic Unit Control register 0x0980. With PDI control, a µController can e.g. set up cyclic interrupts for itself.

9.3 System Time PDI Controlled

Sometimes Distributed Clocks of different EtherCAT networks have to be synchronized. One solution is master-master communication, the other one is based on a physical device which is present in both EtherCAT networks. One of the networks contains the DC Reference Clock (DC source), the other one – DC destination – is synchronized to the Reference Clock in the DC source network.

Some ESCs support such a synchronization by a different functionality of the System Time register (0x0910:0x0913). In normal operation mode, a write access initiated by the EtherCAT master to the System Time register triggers the synchronization: the written value is compared to the local copy of the System Time, and the difference is fed into the control loop. If the System Time is PDI controlled, the PDI writes the System Time register, and the written value is compared to the DC Latch0 Time Positive Edge register (0x09B0:0x09B3). This feature makes the accuracy of the synchronization independent of the μ Controller/PDI response times.

The following figures illustrate how the System Time is transferred from the DC source to the DC destination. ESC 1 and ESC 2 are located in different EtherCAT networks. The EtherCAT network of ESC 1 contains the Reference Clock, the network of ESC 2 will become synchronized to this Reference Clock. ESC 2 is the “reference clock” of its EtherCAT network. There are two options for synchronization, which has to be performed on a regular basis.

The first option is to let the μ Controller generate a trigger pulse for ESC 1 and 2. The time of the rising edge is stored in the Latch0 Time Positive Edge register both in ESC 1 and 2. Afterwards, the μ Controller reads this time from ESC 1 and writes it into the System Time register of ESC 2. The difference of the Latch0 times is used to feed the control loop.

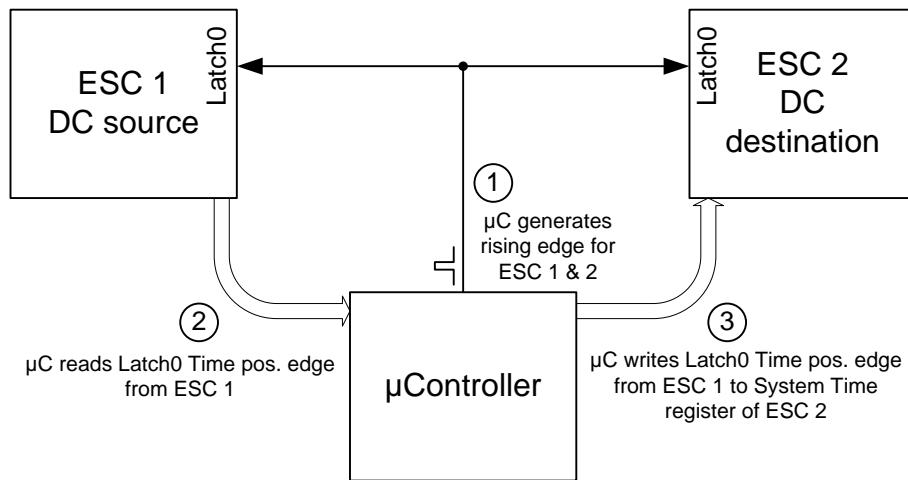


Figure 33: System Time PDI Controlled with three steps

The second option uses a SyncSignal output of ESC 1 to trigger Latch0 at ESC 2 and an interrupt at the µController. Upon receiving an interrupt, the µController writes the time of the SyncSignal pulse to the System Time register of ESC 2. The µController has to calculate the time of the SyncSignal based upon Start Time Cyclic Operation and SYNC Cycle Time configuration of ESC 1 from interrupt to interrupt. The advantage of the second solution is less communication, the disadvantages are more calculation overhead and error detection/troubleshooting.

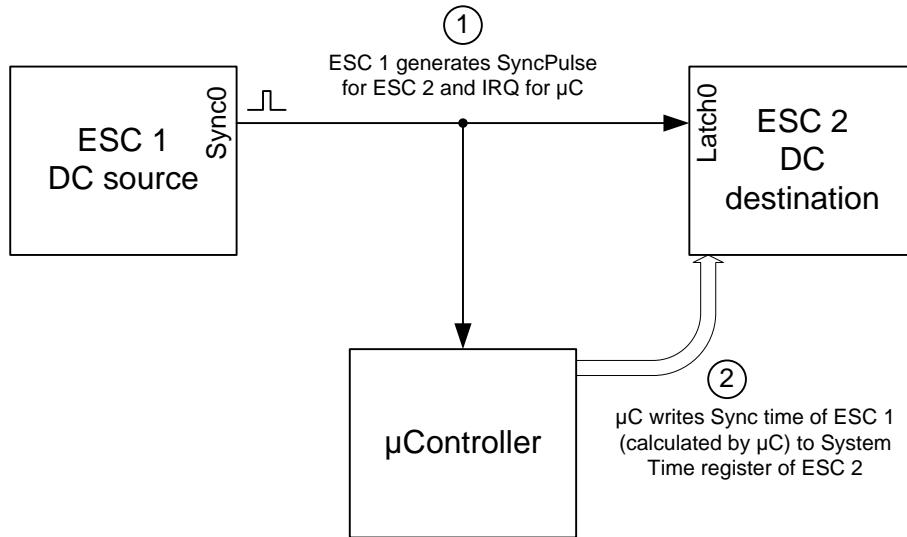


Figure 34: System Time PDI Controlled with two steps

9.4 Communication Timing

Three communication modes are possible:

1. Free Run
EtherCAT Communication and application are running independently from each other.
2. Synchronized to Output Event
The slave application is synchronized to an Output event. If no Outputs are used the Input event is used for synchronization.
3. Synchronized to SyncSignal
Application is synchronized to the SyncSignal.

For further information please refer to the corresponding section within the EtherCAT Information System.

The Communication Timing with use of Distributed Clocks is explained in Figure 35.

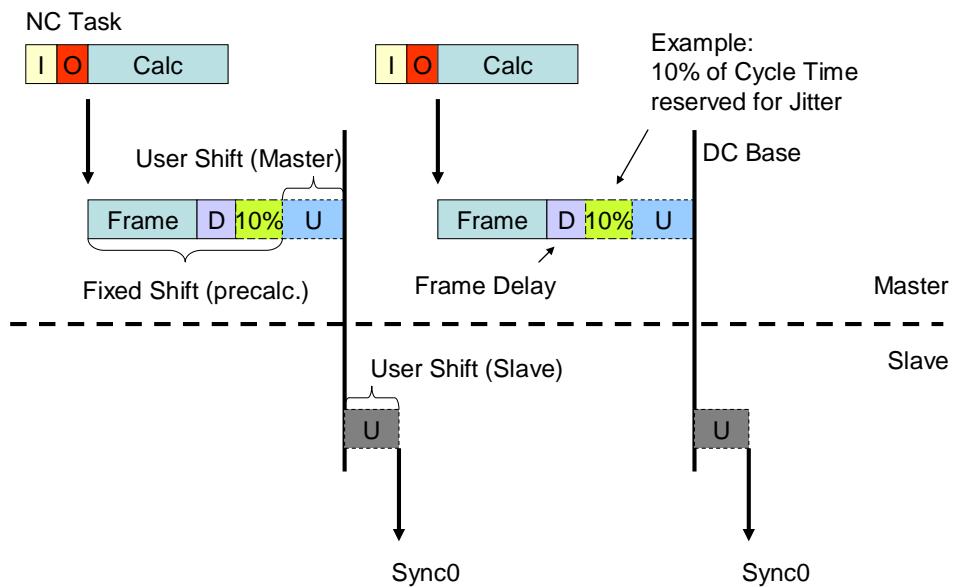


Figure 35: DC Timing Signals in relation to Communication

I/O(Master)

Time to load IO Data to communication buffer and vice versa.

Calc(Master)

Processing time of the master.

Frame(Communication)

Time to transmit the IO-Data-Frame (about 5µs overhead plus 80ns per Byte of Data).

D(Communication)

Delay time of the EtherCAT-Slaves to transfer data (approx. 1 µs with 100BASE-TX, plus line delay of approx. 5ns per m).

Jitter(Communication)

Depends mostly on Master timing quality.

U(Communication-Master)

Shift time that is adjusted internally by the master to deal with delays needed by the master and adjust the cycle time.

U(Slave)

Delay time of the EtherCAT-Slaves. This can be set by each slave individually and is usually 0. There is a need to set this parameter in case of timing inaccuracies of the slave or to deal with slaves that have a slow output method compared to others with high speed output.

Cycle Time Jitter

Cycle Time Jitter is application specific and depends on the jitter of the master system, the used infrastructure components and the slaves. This example assumes a time of 10% of the cycle time for jitter compensation.

10 EtherCAT State Machine

The EtherCAT State machine (ESM) is responsible for the coordination of master and slave applications at start up and during operation. State changes are typically initiated by requests of the master. They are acknowledged by the local application after the associated operations have been executed. Unsolicited state changes of the local application are also possible.

Simple devices without a µController can be configured to use EtherCAT State Machine emulation. These devices simply accept and acknowledge any state change automatically.

There are four states an EtherCAT slave shall support, plus one optional state:

- Init
- Pre-Operational
- Safe-Operational
- Operational
- Bootstrap (optional)

The states and the allowed state changes are shown in Figure 36:

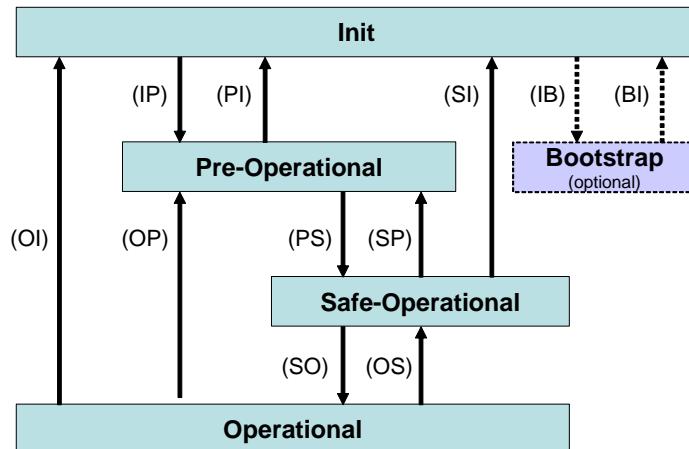


Figure 36: EtherCAT State Machine

NOTE: Not all state changes are possible, e.g., the transition from 'Init' to 'Operational' requires the following sequence: Init → Pre-Operational → Save-Operational → Operational.

Each state defines required services. Before a state change is confirmed by the slave all services required for the requested state have to be provided or stopped respectively.

10.1 EtherCAT State Machine Registers

The state machine is controlled and monitored via registers within the ESC. The master requests state changes by writing to the AL Control register. The slave indicates its state in the AL Status register and puts error codes into the AL Status Code register.

Table 35: Registers for the EtherCAT State Machine

Register Address	Name	Description
0x0120:0x0121	AL Control	Requested state by the master
0x0130:0x0131	AL Status	AL Status of the slave application
0x0134:0x0135	AL Status Code	Error codes from the slave application
0x0140.8	PDI Control	Device emulation configuration

NOTE: The PDI Control register is set via SII EEPROM/IP Core configuration, others are not available in specific ESCs. Refer to Section II and Section III for details.

10.1.1 AL Control and AL Status Register

Writing the AL Control register (0x0120:0x0121) initiates a state transition of the device state machine. The AL Status register (0x0130:0x0131) reflects the current state of the slave.

Table 36: AL Control and AL Status Register Values

Register [3:0]	AL Control Register 0x0120	AL Status Register 0x0130
1	Request Init state	Init state
3	Request Bootstrap state (optional)	Bootstrap state (optional)
2	Request Pre-Operational state	Pre-Operational state
4	Request SAFE-Operational state	SAFE-Operational state
8	Request Operational state	Operational state

10.1.2 Device Emulation

Simple devices (without µController) have the device emulation enabled (0x0140.8=1). The AL Control register is directly copied into the AL Status register by the ESC. The master should not set the Error Indication Acknowledge bit for such slaves at all, because setting this bit would result in setting the Error Indication bit – although no error occurred.

10.1.3 Error Indication and AL Status Code Register

The slave indicates errors during a state transition by setting the Error Indication flag (0x0130.4=1) and writing an error description into the AL Status Code register (0x0134:0x0135). The master acknowledges the Error Indication flag of the slave by setting the Error Ind Ack flag (0x0120.4).

An excerpt of defined AL Status Codes is shown in Table 37. For more information, refer to the EtherCAT Knowledge Base available at the EtherCAT Technology Group website (<http://www.ethernetcat.org>, Guidelines and Protocol Enhancements).

Table 37: AL Status Codes (0x0134:0x0135)

Code	Description	Current state (or state change)	Resulting state	ERR LED code
0x0000	No error	Any	Current state	
0x0001	Unspecified error	Any	Any + E	
0x0002	No Memory	Any	Any + E	
0x0011	Invalid requested state change	I → S, I → O, P → O, O → B, S → B, P → B	Current state + E	
0x0012	Unknown requested state	Any	Current state + E	
0x0013	Bootstrap not supported	I → B	I + E	
0x0014	No valid firmware	I → P	I + E	
0x0015	Invalid mailbox configuration	I → B	I + E	
0x0016	Invalid mailbox configuration	I → P	I + E	
0x0017	Invalid sync manager configuration	P → S, S → O	Current state + E	
0x0018	No valid inputs available	O, S → O	S + E	
0x0019	No valid outputs	O, S → O	S + E	
0x001A	Synchronization error	O, S → O	S + E	Single flash
0x001B	Sync manager watchdog	O, S → O	S + E	Double flash
0x001C	Invalid Sync Manager Types	O, S, P → S	S + E	
0x001D	Invalid Output Configuration	O, S, P → S	S + E	
0x001E	Invalid Input Configuration	O, S, P → S	P + E	
0x001F	Invalid Watchdog Configuration	O, S, P → S	P + E	
0x0020	Slave needs cold start	Any	Current state + E	
0x0021	Slave needs INIT	B, P, S, O	Current state + E	
0x0022	Slave needs PREOP	S, O	S + E, O + E	
0x0023	Slave needs SAFEOP	O	O + E	
0x0024	Invalid Input Mapping	P → S	P + E	
0x0025	Invalid Output Mapping	P → S	P + E	
0x0026	Inconsistent Settings	P → S	P + E	
0x0027	Freerun not supported	P → S	P + E	
0x0028	Synchronization not supported	P → S	P + E	
0x0029	Freerun needs 3 Buffer Mode	P → S	P + E	
0x002A	Background Watchdog	S, O	P + E	
0x002B	No Valid Inputs and Outputs	O, S → O	S + E	
0x002C	Fatal Sync Error	O	S + E	
0x002D	No Sync Error	S → O	S + E	Single flash
0x0030	Invalid DC SYNC Configuration	O, S → O, P → S	P + E, S + E	
0x0031	Invalid DC Latch Configuration	O, S → O, P → S	P + E, S + E	

Code	Description	Current state (or state change)	Resulting state	ERR LED code
0x0032	PLL Error	O, S → O	S + E	Single flash
0x0033	DC Sync IO Error	O, S → O	S + E	
0x0034	DC Sync Timeout Error	O, S → O	S + E	
0x0035	DC Invalid Sync Cycle Time	P → S	P + E	
0x0036	DC Sync0 Cycle Time	P → S	P + E	
0x0037	DC Sync1 Cycle Time	P → S	P + E	
0x0041	MBX_AOE	B, P, S, O	Current state + E	
0x0042	MBX_EOE	B, P, S, O	Current state + E	
0x0043	MBX_COE	B, P, S, O	Current state + E	
0x0044	MBX_FOE	B, P, S, O	Current state + E	
0x0045	MBX_SOE	B, P, S, O	Current state + E	
0x004F	MBX_VOE	B, P, S, O	Current state + E	
0x0050	EEPROM No Access	Any	Any + E	
0x0051	EEPROM Error	Any	Any + E	
0x0060	Slave Restarted Locally	Any	I	
other codes <0x8000	Reserved			
0x8000- 0xFFFF	Vendor specific			

NOTE: “+ E” in the resulting state column indicates setting of the Error Indication flag.

10.2 State Machine Services

The active services of each state are shown in Table 38.

Table 38: State Machine Services

State/ State Change	Services
INIT	<ul style="list-style-type: none"> • No communication on Application Layer • Master has access to the DL-Information registers
INIT TO PREOP	<ul style="list-style-type: none"> • Master configures registers, at least: <ul style="list-style-type: none"> - DL Address register - SyncManager channels for Mailbox communication • Master initializes DC clock synchronization • Master requests ‘Pre-Operational’ state <ul style="list-style-type: none"> - Master sets AL Control register • wait for AL Status register confirmation
PREOP	<ul style="list-style-type: none"> • Mailbox communication on the Application Layer • No Process Data communication
PREOP TO SAFEOP	<ul style="list-style-type: none"> • Master configures parameters using the Mailbox: <ul style="list-style-type: none"> - e.g., Process Data Mapping • Master configures DL Register: <ul style="list-style-type: none"> - SyncManager channels for Process Data communication - FMMU channels • Master requests ‘Safe-Operational’ state • wait for AL Status register confirmation
SAFEOP	<ul style="list-style-type: none"> • Mailbox communication on the Application Layer • Process Data communication, but only Inputs are evaluated – Outputs remain in ‘Safe’ state
SAFEOP TO OP	<ul style="list-style-type: none"> • Master sends valid Outputs • Master requests ‘Operational’ state (AL Control/Status) • wait for AL Status register confirmation
OP	<ul style="list-style-type: none"> • Inputs and Outputs are valid
BOOT	<p>Optional, but recommended if firmware updates are necessary.</p> <ul style="list-style-type: none"> • State changes only from and to INIT • No Process Data communication • Mailbox communication on Application Layer, only FoE protocol available (possibly limited “file” range) • Special mailbox configuration possible

11 SII EEPROM

EtherCAT slave controllers use a mandatory NVRAM (typically a serial EEPROM with I²C interface) to store EtherCAT Slave Information (ESI). EEPROM sizes from 1 kBit up to 4 Mbit are supported, depending on the ESC.

The EtherCAT IP Core supports omitting the serial I²C EEPROM if a µController with read/write access to an NVRAM (e.g. the one which contains the µController's program and data, or the FPGA configuration EPPROM) is used to emulate the EEPROM transactions. Since the logical interface is the same in this case, the EEPROM emulation is treated to be equivalent to the typical I²C EEPROM solution throughout this chapter. Refer to chapter 11.2.4 for more details about EEPROM emulation.

The EEPROM structure is shown in Figure 37. The ESI uses word addressing.

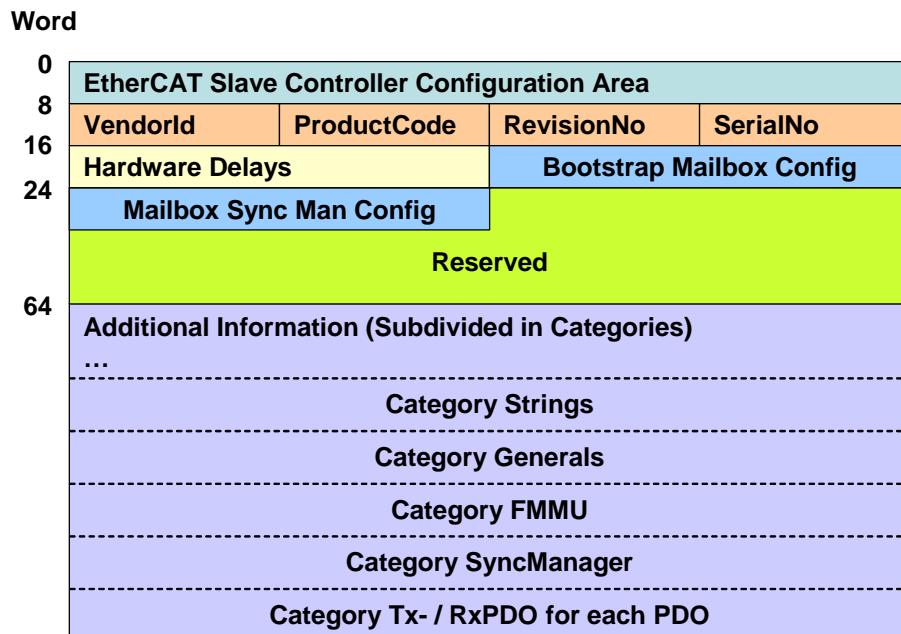


Figure 37: SII EEPROM Layout

At least the information stored in the address range from word 0 to 63 (0x00 to 0x3F) is mandatory, as well as the general category (→ absolute minimum SII EEPROM size is 2kbit, complex devices with many categories should be equipped with 32 kbit EEPROMs or larger). The ESC Configuration area is used by the ESC for configuration. All other parts are used by the master or the local application.

For a more detailed description of the ESI and other mandatory parts refer to the ETG.2000 EtherCAT Slave Information (ESI) Specification, available from the download section of the EtherCAT Technology Group website (<http://www.ethercat.org>).

11.1 SII EEPROM Content

The ESC Configuration Area (EEPROM word addresses 0 to 7) is automatically read by the ESC after power-on or reset. It contains the PDI configuration, DC settings, and the Configured Station Alias. The consistency of the ESC Configuration data is secured with a checksum.

For SII coding refer to ETG.1000 EtherCAT Specification, Part 6, Clause 5.4, available in the download area of the EtherCAT Technology Group website (<http://www.ethercat.org>).

The EtherCAT master can invoke reloading the EEPROM content. In this case the Configured Station Alias 0x0012:0x0013 and PDI Control Bit 0x0140.9 (enhanced link detection) are not taken over, they are only taken over at the initial EEPROM loading after power-on or reset.

The ESC Configuration Area is shown in Table 39.

Table 39: ESC Configuration Area

Word Address	Parameter	Description	Register Address
0x0	PDI Control	Initialization value for PDI Control register (EEPROM ADR 0x0000.9 is also mapped to register 0x0110.2)	0x0140:0x0141
0x1	PDI Configuration	Initialization value for PDI Configuration register	0x0150:0x0151
0x2	Pulse Length of SYNC Signals	Initialization value for Pulse Length of SYNC Signals register	0x0982:0x0983
0x3	Extended PDI Configuration	Initialization value for extended PDI Configuration register	0x0152:0x0153
0x4	Configured Station Alias	Initialization value for Configured Station Alias Address register	0x0012:0x0013
0x5	Reserved	Reserved, shall be zero	-
0x6	Reserved	Reserved, shall be zero	-
0x7	Checksum	Low byte contains remainder of division of word 0 to word 6 as unsigned number divided by the polynomial x^8+x^2+x+1 (initial value 0xFF). NOTE: For debugging purposes it is possible to disable the checksum validation with a checksum value of 0x88A4. Never use this for production!	-

NOTE: Reserved words or reserved bits of the ESC Configuration Area should be filled with 0.

An excerpt of the SII EEPROM content following the ESC Configuration area is shown in Table 40. For more information, refer to the ETG.2000 EtherCAT Slave Information (ESI) Specification, available from the download section of the EtherCAT Technology Group website (<http://www.ethercat.org>).

Table 40: SII EEPROM Content Excerpt

Word Address	Parameter	Word Address	Parameter
0x0	PDI Control	0x14	Bootstrap Receive Mailbox Offset
0x1	PDI Configuration	0x15	Bootstrap Receive Mailbox Size
0x2	Pulse Length of SYNC Signals	0x16	Bootstrap Send Mailbox Offset
0x3	Extended PDI Configuration	0x17	Bootstrap Send Mailbox Size
0x4	Configured Station Alias	0x18	Standard Receive Mailbox Offset
0x5:0x6	Reserved	0x19	Standard Receive Mailbox Size
0x7	Checksum	0x1A	Standard Send Mailbox Offset
0x8:0x9	Vendor ID	0x1B	Standard Send Mailbox Size
0xA:0xB	Product Code	0x1C	Mailbox Protocol
0xC:0xD	Revision Number	0x1D:0x3D	Reserved
0xE:0xF	Serial Number	0x3E	Size
0x10	Execution Delay	0x3F	Version
0x11	Port0 Delay	0x40	First Category Type/Vendor Specific
0x12	Port1 Delay	0x41	Following Category Word Size
0x13	Reserved	0x42	Category Data
		...	Second Category ...

11.2 SII EEPROM Logical Interface

The SII EEPROM interface of the ESC is either controlled by EtherCAT or by the PDI. Initially, EtherCAT has EEPROM interface access, but it can transfer access to the PDI.

Table 41: SII EEPROM Interface Register Overview

Register Address	Description
0x0500	EEPROM Configuration
0x0501	EEPROM PDI Access State
0x0502:0x0503	EEPROM Control/Status
0x0504:0x0507	EEPROM Address
0x0508:0x050F	EEPROM Data

The EEPROM interface supports three commands: write to one EEPROM address (1 Word), read from EEPROM (2 or 4 Words, depending on ESC), or reload ESC configuration from EEPROM.

11.2.1 SII EEPROM Errors

The ESC retries reading the EEPROM after power-on or reset once if an error has occurred (missing acknowledge, wrong checksum). If reading the ESC Configuration Area fails twice, the Error Device Information bit is set, and the PDI Operational bit in the ESC DL Status register (0x0110.0) remains clear and the EEPROM_Loaded signal (if available) remains inactive. The process memory is not accessible until the ESC Configuration Area is loaded successfully.

All registers initialized by the ESC Configuration Area keep their values in case of an error. This is also true for the Error Device Information bit as well as the PDI Operational bit. Only if the EEPROM was loaded/reloaded successfully, the registers take over the new values (except for Configured Station Alias 0x0012:0x0013 and PDI Control Bit 0x140.9 – enhanced link detection).

The SII EEPROM interface has these error status bits:

Table 42: SII EEPROM Interface Errors

Bit	Name	Description
11	Checksum Error	<p>ESC Configuration Area checksum is wrong (after device initialisation or EEPROM reload). Registers initialized with EEPROM values keep their value.</p> <p>Reason: CRC error</p> <p>Solution: Check CRC</p> <p>EEPROM Emulation only: Checksum error indicates a non-temporary reload failure</p>
12	Error Device Information	<p>ESC Configuration not loaded</p> <p>Reasons: Checksum error, acknowledge error, EEPROM missing</p> <p>Solution: Check other error bits</p>
13	Error Acknowledge/Command	<p>Missing Acknowledge or invalid command</p> <p>Reason: a) Missing acknowledge from EEPROM chip (see below). EEPROM chip is busy performing operations internally or EEPROM chip is not available. b) Invalid command issued</p> <p>Solution: a) Retry access. EEPROM device does not acknowledge if it is internally busy. b) Use valid commands</p> <p>EEPROM Emulation only: Missing Acknowledge error indicates a temporary failure. Invalid command error is automatically supported by the EEPROM interface.</p>
14	Error Write Enable	<p>Write without Write Enable (ECAT control only):</p> <p>Reason: ECAT issued a write command without Write Enable bit set (0x0502.0)</p> <p>Solution: Set Write Enable bit in the same frame as the write command</p>

11.2.1.1 Missing Acknowledge

Missing acknowledges from the EEPROM chip are a common issue, especially if a fast PDI uses the EEPROM interface. E.g., a write access to the EEPROM with missing acknowledge may look like this:

1. ECAT/PDI issue write command (first command)
2. ESC is busy transferring the write data to the EEPROM chip.
3. ESC is not busy anymore. EEPROM chip is internally busy transferring data from input buffer to storage area.
4. ECAT/PDI issue a second command.
5. ESC is busy transferring the write data to the EEPROM chip. EEPROM chip does not acknowledge any access until internal transfer has finished (may take up to several ms).
6. ESC is not busy anymore. Error Acknowledge/Command bit is set. (ESC has to re-issue the second command after EEPROM chip is finished and the command is acknowledged).
7. EEPROM chip finishes internal transfer.
8. ESC re-issues the second command, the command is acknowledged and executed successful.

This is also possible for a read access, because some EEPROM chips require an idle period between any two accesses. During this idle period, they do not acknowledge any access.

11.2.2 SII EEPROM Interface Assignment to ECAT/PDI

The EtherCAT master controls the EEPROM interface (default) if EEPROM configuration register 0x0500.0=0 and EEPROM PDI Access register 0x0501.0=0, otherwise PDI controls EEPROM interface. These access rights should be checked before using the EEPROM Interface by both sides.

A typical EEPROM interface control hand-over is as follows:

1. ECAT assigns EEPROM interface to PDI by writing 0x0500.0=1
2. If PDI wishes to access EEPROM, it takes over EEPROM control by writing 0x0501.0=1.
3. PDI issues EEPROM commands.
4. After PDI has finished EEPROM commands, PDI releases EEPROM control by writing 0x0501.0=0.
5. ECAT may take back the EEPROM interface by writing 0x0500.0=0
6. ECAT checks EEPROM control by reading 0x0501
7. ECAT issues EEPROM commands.

If the PDI does not release EEPROM control (e.g. because of a software failure), ECAT can force releasing the access:

1. ECAT writes 0x02 to register 0x0500 (as the result, 0x0501.0 is cleared)
2. ECAT writes 0x00 to register 0x0500
3. ECAT has control over EEPROM interface

11.2.3 Read/Write/Reload Example

The following steps have to be performed for a SII EEPROM read or write access:

1. Check if the Busy bit of the EEPROM Status register is cleared (0x0502.15==0) and the EEPROM interface is not busy, otherwise wait until the EEPROM interface is not busy anymore.
2. Check if the Error bits of the EEPROM Status register are cleared. If not, write "000" to the command register (register 0x0502 bits [10:8]).
3. Write EEPROM word address to EEPROM Address register.
4. Write command only: put write data into EEPROM Data register (1 word/2 byte only).
5. Issue command by writing to Control register.
 - a) For a read command, write 001 into Command Register 0x0502[10:8].
 - b) For a write command, write 1 into Write Enable bit 0x0502.0 and 010 into Command Register 0x0502[10:8]. Both bits have to be written in one frame. The Write enable bit realizes a write protection mechanism. It is valid for subsequent EEPROM commands issued in the same frame and self-clearing afterwards. The Write enable bit needs not to be written from PDI if it controls the EEPROM interface.
 - c) For a reload command, write 100 into Command Register 0x0502[10:8].
6. The command is executed after the EOF if the EtherCAT frame had no errors. With PDI control, the command is executed immediately.
7. Wait until the Busy bit of the EEPROM Status register is cleared.
8. Check the Error bits of the EEPROM Status register. The Error bits are cleared by clearing the command register. Retry command (back to step 5) if EEPROM acknowledge was missing. Eventually wait some time before retrying to allow slow EEPROMs to store the data internally.
9. a) For a Read command: Read data is available in EEPROM Data registers (2 or 4 Words, depending on ESC – check register 0x0502.6).
b) For a Reload command: ESC configuration is reloaded into appropriate registers.

NOTE: The Command register bits are self-clearing. Manually clearing the command register will also clear the status information.

11.2.4 EEPROM Emulation

The EEPROM emulation mode is used in IP Core based ESCs with a non-volatile memory (NVRAM) attached to a µController. The ESC configuration and the device description can be stored in the NVRAM of the µController, e.g., together with the program or FPGA configuration code. An additional I²C EEPROM chip for the ESC is not needed any more if EEPROM emulation is used.

The µController emulates the EEPROM interface actions of the ESC and executes all EEPROM reload, read, and write requests. EEPROM write data is stored in the NVRAM of the µController, and EEPROM read data is read from the NVRAM and presented to the EEPROM interface of the ESC.

From the EtherCAT master's point of view, EEPROM emulation mode is equivalent to an I²C EEPROM. The master issues EEPROM commands and waits until the EEPROM interface is not busy anymore.

In EEPROM emulation mode, the EEPROM interface of the ESC issues an interrupt to the µController if an EEPROM command is pending and sets the busy bit. While the busy bit is set, the µController can read out the command and the EEPROM address. For a write access, write data is present in the data register. For a read command, read data has to be stored in the data register by the µController. A reload command requires the µController to place the Configured Station Alias and Enhanced Link detection settings in the data register.

Once the µController has finished reading/writing the EEPROM data register, it acknowledges the command by writing to the EEPROM command register bits. The µController has to write the command value it has executed into the EEPROM command register. Errors can be indicated using two of the error bits. After acknowledging the command, the EEPROM state machine is not busy anymore and the interrupt is released.

The read data for a reload command (or the initial EEPROM loading) is reduced to the Configured Station Alias (0x0012:0x0013) and the Enhanced Link Detection Enable bits (0x0140[9], 0x0140[15:12]).

NOTE: The EEPROM can be assigned to the PDI even if EEPROM Emulation is used. EEPROM_SIZE has to be 0 for EEPROM emulation (EEPROM emulation with EEPROM_SIZE=1 is for testing only: all commands are acknowledged automatically).

11.3 SII EEPROM Electrical Interface (I²C)

The SII EEPROM Interface is intended to be a point-to-point interface between ESC and I²C EEPROM. If other I²C masters are required to access the I²C bus, the ESC must be held in reset state (e.g. for in-circuit-programming of the EEPROM), otherwise access collisions will occur.

The SII EEPROM interface has the following signals¹:

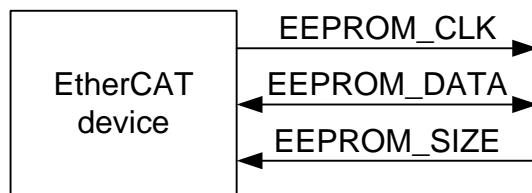


Figure 38: I²C EEPROM signals

Table 43: I²C EEPROM signals

Signal	Direction	Description
EEPROM_CLK	OUT	I ² C clock
EEPROM_DATA	BIDIR	I ² C data
EEPROM_SIZE	IN	EEPROM size configuration

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.

11.3.1 Word Addressing

EtherCAT and ESCs use word addressing when accessing the EEPROM, although the I²C interface actually uses byte addressing. The lowest address bit A[0] is added internally by the EEPROM interface controller of the ESCs. I.e., the EEPROM address register (0x0504:0x0507) reflects the physical EEPROM address bits A[18:1] (higher address bits are reserved).

11.3.2 EEPROM Size

Depending on the EEPROM size, one out of two EEPROM algorithms has to be selected with the EEPROM_SIZE configuration signal. Smaller EEPROMs need only one address byte, larger ones need two address bytes:

Table 44: EEPROM Size

EEPROM Size	Address Bytes	Max. I ² C Address Bits	EEPROM_SIZE signal
Up to 16 KBit	1	11	0
32 KBit – 4 MBit	2	19	1

¹ The availability of the EEPROM signals as well as their names depend on the specific ESC.

11.3.3 I²C Access Protocol

Each EEPROM access begins with a Start condition and ends with an Stop condition. Data is transferred byte-wise, and each byte is acknowledged by the recipient.

The Start condition is a falling edge on EEPROM_DATA while EEPROM_CLK is high, the Stop condition is a rising edge on EEPROM_DATA while EEPROM_CLK is high. In all other cases, EEPROM_DATA has to remain stable while EEPROM_CLK is high, as this indicates valid data. A byte transfer is acknowledged in an additional bit, which is driven low by the recipient of the byte transfer if he acknowledges the byte.

NOTE: If the EEPROM does not acknowledge an access (Ack bit=high), it might be busy internally. Especially if the EEPROM interface is handled by a µController via the PDI, this situation may come up, because many µControllers can write to the EEPROM interface much faster than many EEPROMs can transfer the data from its input registers into its NVRAM.

The first byte of an I²C access is the Control Byte (Bit 7/MSB is transferred first):

Table 45: I²C Control Byte

Bit	Description
0	Read/Write access: 0: Write 1: Read
[3:1]	Chip Select Bits/Highest Address Bits
[7:4]	Control Code: 1010

Depending on the access, either read data will follow or additional address bytes and write data. This is described in the following chapters.

The EEPROM has an internal byte pointer, which is incremented automatically after each data byte transfer.

For more details about the I²C protocol, refer to “The I²C-Bus Specification”, available from NXP (<http://www.nxp.com>, document number 39340011) and <http://www.i2c-bus.org>.

11.3.3.1 Write Access

An EEPROM write access always writes one word (2 bytes) to the EEPROM. In this case, page boundaries are not relevant, because they will not be violated.

The ESC will perform the following steps for a write access to the EEPROM:

Table 46: I²C Write Access

Step	Description	Up to 16 kBit	32 kBit – 4 MBit
1	Start condition		
2	Control Byte (Write)	A[10:8]	A[18:16]
3*	High Address Byte	Not preset	A[15:8]
4	Low Address Byte	A[7:0]	A[7:0]
5	Low Data Byte		D[7:0]
7	High Data Byte		D[15:8]
8	Stop condition		

* This step is only for EEPROMs larger than 16 KBit.

11.3.3.2 Read Access

An EEPROM read Access reads 2 or 4 words (4 or 8 bytes, depending on device capabilities) from the EEPROM, the load or reload EEPROM access typically reads 8 words (16 bytes). The address wrap-around at the end of the EEPROM address space has to be taken into account by the application, the ESC has no knowledge about it.

The ESC will perform the following steps for a read access to the EEPROM. At first, the address is written to the EEPROM, then the data is read (N=3 or N=7):

Table 47: I^C Read Access

Step	Description	Up to 16 KBit	32 KBit – 4 MBit
1	Start condition		
2	Control Byte (Write)	A[10:8]	A[18:16]
3*	High Address Byte	Not present	A[15:8]
4	Low Address Byte	A[7:0]	A[7:0]
5	Start condition		
6	Control Byte (Read)	A[10:8]	A[18:16]
7	Data Byte 0	D0 [7:0]	
8	Data Byte 1	D1 [7:0]	
...	
N+7	Data Byte N	DN [7:0]	
N+8	Stop condition		

* This step is only for EEPROMs larger than 16 KBit.

11.3.4 Timing specifications

Table 48: EEPROM timing characteristics

Parameter	Comment
t _{Clk}	EEPROM clock period
t _{Write}	Write access time (without errors)
t _{Read}	Read access time (without errors)
t _{Delay}	Time until configuration loading begins after Reset is gone

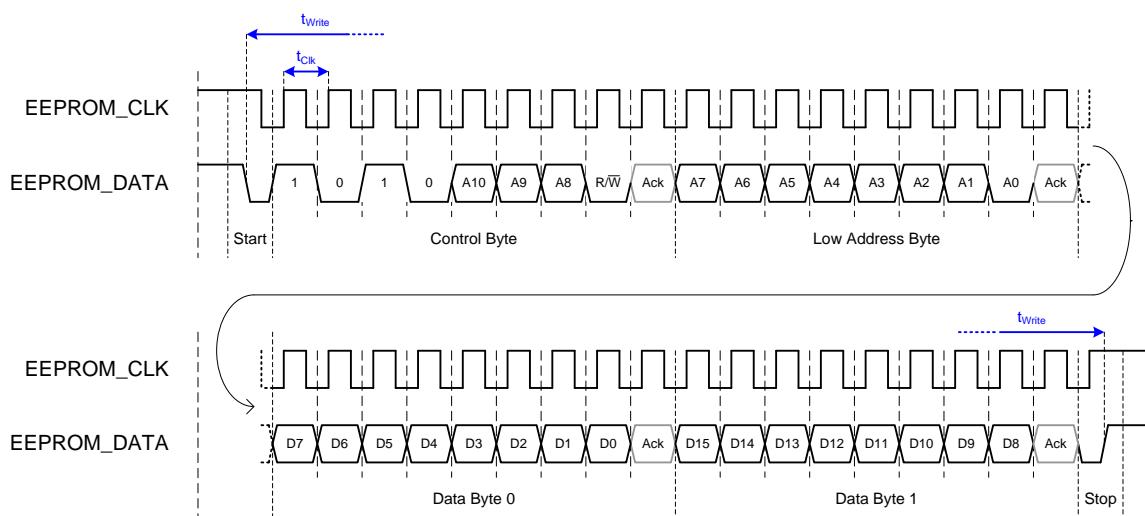


Figure 39: Write access (1 address byte, up to 16 kBit EEPROMs)

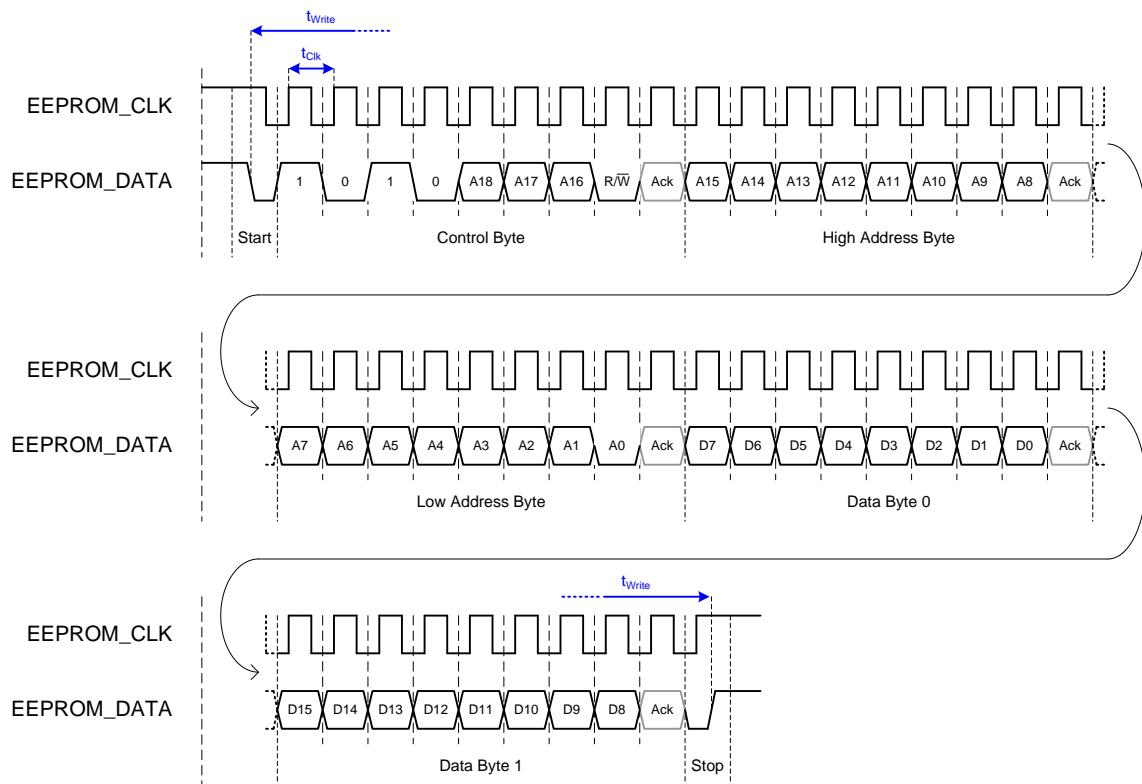


Figure 40: Write access (2 address bytes, 32 kBit - 4 MBit EEPROMs)

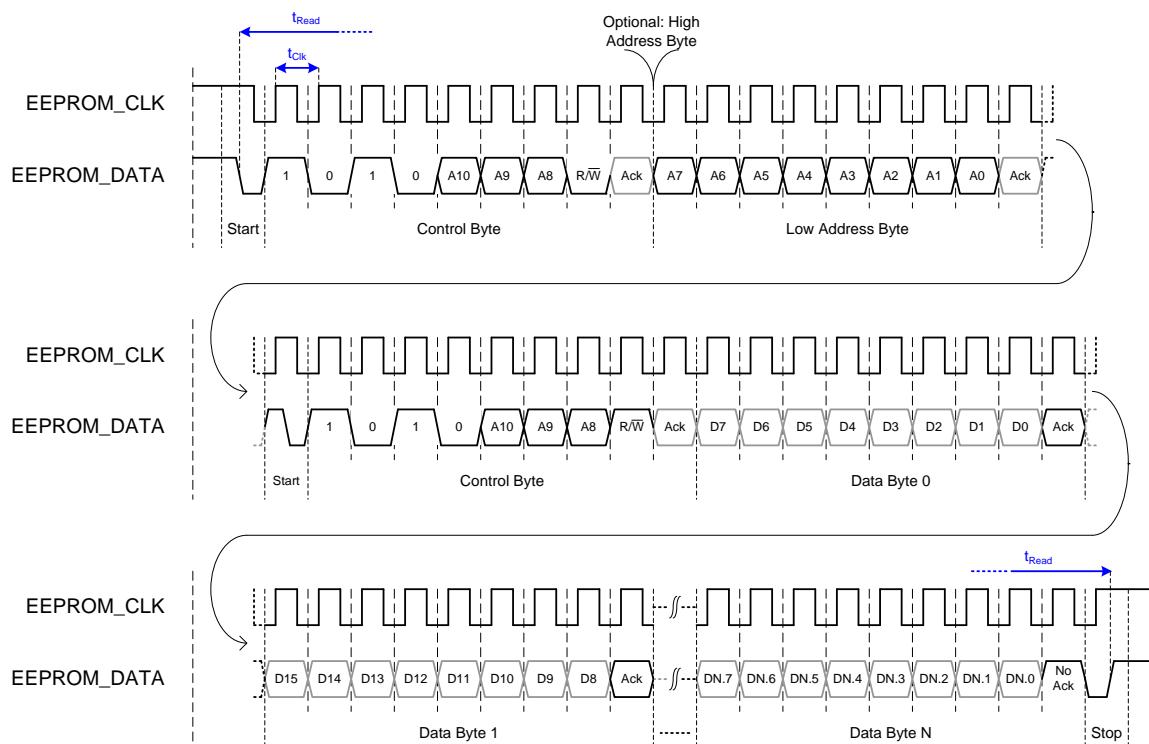


Figure 41: Read access (1 address byte, up to 16 kBit EEPROMs)

12 Interrupts

ESCs support two types of interrupts: AL Event Requests dedicated for a µController, and ECAT event requests dedicated for the EtherCAT master. Additionally, the Distributed Clocks SyncSignals can be used as interrupts for a µController as well.

12.1 AL Event Request (PDI Interrupt)

AL Event Requests can be signaled to a µController using the PDI Interrupt Request signal (IRQ/SPI_IRQ, etc.). For IRQ generation, the AL Event Request register (0x0220:0x0223) is combined with the AL Event Mask register (0x0204:0x0207) using a logical AND operation, then all resulting bits are combined (logical OR) into one interrupt signal. The output driver characteristics of the IRQ signal are configurable using the SYNC/LATCH PDI configuration register (0x0151). The AL Event Mask register allows for selecting the interrupts which are relevant for the µController and handled by the application.

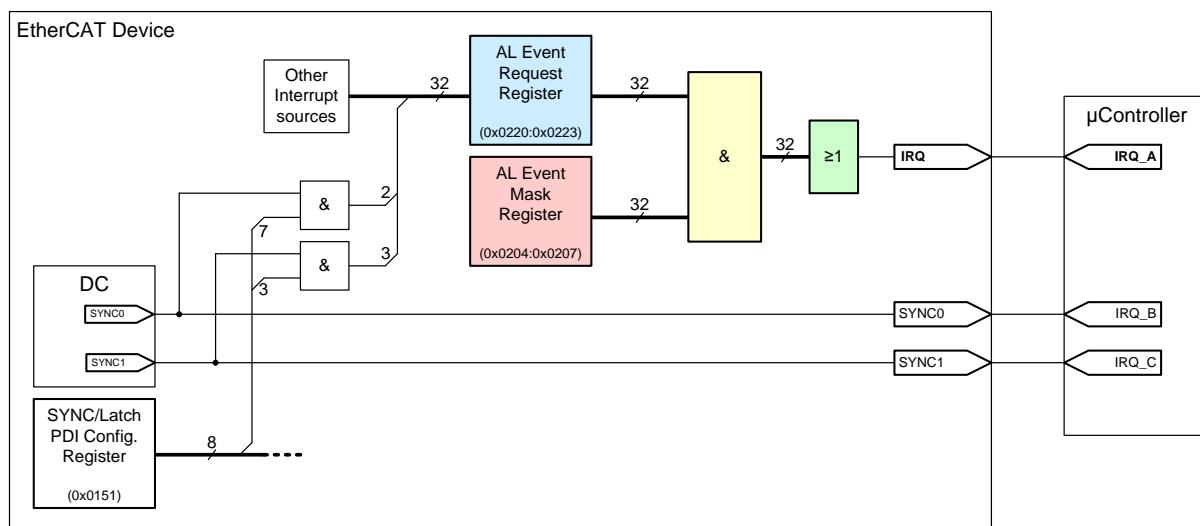


Figure 42: PDI Interrupt Masking and interrupt signals

The DC SyncSignals can be used for interrupt generation in two ways:

- The DC SYNC signals are mapped into the AL Event Request Register (configured with SYNC/LATCH PDI Configuration register 0x0151.3/7). In this case, all interrupts from the ESC to the µController are combined into one IRQ signal, and the Distributed Clocks LATCH0/1 inputs can still be used. The IRQ signal has a jitter of ~40 ns.
- The DC SyncSignals are directly connected to µController interrupt inputs. The µController can react on DC SyncSignal interrupts faster (without reading AL Request register), but it needs more interrupt inputs. The jitter of the SyncSignals is ~12 ns. The DC Latch functions are only available for one Latch input or not at all (if both DC SYNC outputs are used).

Registers used for AL event requests are described in Table 49:

Table 49: Registers for AL Event Request Configuration

Register Address	Name	Description
0x0150	PDI Configuration	IRQ driver characteristics, depending on PDI
0x0151	SYNC/LATCH PDI Configuration	Mapping DC SyncSignals to Interrupts
0x0204:0x0207	AL Event Mask	Mask register
0x0220:0x0223	AL Event Request	Pending Interrupts
0x0804 + N*8	SyncManager Control	Mapping SyncManager Interrupts

NOTE: Some of these registers are set via SII EEPROM/IP Core configuration, or they are not available in specific ESCs. Refer to Section II for details.

12.2 ECAT Event Request (ECAT Interrupt)

ECAT event requests are used to inform the EtherCAT master of slave events. ECAT events make use of the IRQ field inside EtherCAT datagrams. The ECAT Event Request register (0x0210:0x0211) is combined with the ECAT Event Mask register (0x0200:0x0201) using a logical AND operation. The resulting interrupt bits are combined with the incoming ECAT IRQ field using a logical OR operation, and written into the outgoing ECAT IRQ field. The ECAT Event Mask register allows for selecting the interrupts which are relevant for the EtherCAT master and handled by the master application.

NOTE: The master can not distinguish which slave (or even more than one) was the origin of an interrupt.

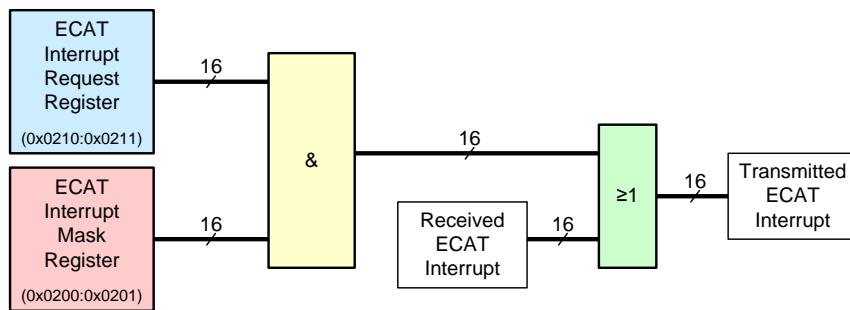


Figure 43: ECAT Interrupt Masking

Registers used for ECAT Interrupts are described in Table 50:

Table 50: Registers for ECAT Event Request Configuration

Register Address	Name	Description
0x0200:0x0201	ECAT Event Mask	Mask register
0x0210:0x0211	ECAT Event Request	Pending Interrupts
0x0804 + N*8	SyncManager Control	Mapping SyncManager Interrupts

NOTE: Some of these registers are not available in specific ESCs. Refer to Section II for details.

12.3 Clearing Interrupts Accidentally

Event request registers and register actions which clear interrupts are intended to be accessed independently, i.e., with separate EtherCAT frames or separate PDI accesses. Otherwise it may happen that interrupts and/or data are missed.

Examples:

- Using SPI to read a SyncManager buffer: polling SyncManager buffers and interrupts delivered at the beginning of each SPI access in the same access can lead to missed interrupts/data. Fault scenario: the interrupt is not pending while the interrupts delivered at the beginning of the access are sampled. The µController gets the information “no interrupt”, but it continues reading the SyncManager buffer because the read command can not be stopped without causing a PDI error. If the SyncManager Interrupt occurs in the time windows between interrupt sampling and buffer reading, new buffer data will be delivered and the interrupt is acknowledged. As a consequence, the µController application will ignore the new data because no interrupt was set.
Solution: Read the SyncManager buffer only if the IRQ signal indicates a pending interrupt or if a **preceding** access indicates pending interrupts.
- Using a single ECAT frame to read DC Latch0/1 status and Latch Time registers: the status registers may indicate no event, but if the event occurs in the time window between reading status and time registers, the new latch time will be delivered and the corresponding interrupt is cleared directly. The master gets the information “no interrupt”, but new latch times, so it will ignore the time values and the interrupt/data is missed.
Solution: Read DC Latch time registers only if an ECAT event was indicated in a previous frame or if the DC Latch status registers were polled in a **previous** frame.

13 Watchdogs

The ESCs support up to two internal watchdogs (WD), a Process Data watchdog used for monitoring process data accesses, and a PDI watchdog monitoring PDI activity.

The timeout for both watchdogs can be configured individually, but they share a single Watchdog Divider (WD_DIV, register 0x0400:0x0401). The watchdog timeout is calculated from the Watchdog Divider settings multiplied with the Watchdog Time settings for PDI (WD_PDI, register 0x0410:0x0411) or Process Data (WD_PD, register 0x0420:0x0421). Base time unit is 40 ns. The Watchdog timeout jitters, the jitter depends on the Watchdog Divider settings. I.e., selecting smaller Watchdog Divider settings results in smaller jitter.

The following equations are used for a quick estimation of the watchdog timeout (they are not exact in terms of nanoseconds):

$$\begin{aligned} t_{WD_Div} &= (WD_DIV + 2) * 40\text{ns} \\ t_{WD_PDI} &= [t_{WD_Div} * WD_PDI ; t_{WD_Div} * WD_PDI + t_{WD_Div}] \\ t_{WD_PD} &= [t_{WD_Div} * WD_PD ; t_{WD_Div} * WD_PD + t_{WD_Div}] \end{aligned}$$

Registers used for Watchdogs are described in Table 51:

Table 51: Registers for Watchdogs

Register Address	Name	Description
0x0110.1	ESC DL Status	Status PDI Watchdog
0x0400:0x0401	Watchdog Divider	Watchdog Divider (WD_DIV)
0x0410:0x0411	Watchdog Time PDI	Watchdog Time PDI (WD_PDI)
0x0420:0x0421	Watchdog Time Process Data	Watchdog Time Process Data (WD_PD)
0x0440:0x0441	Watchdog Status Process Data	Status Process Data Watchdog
0x0442	Watchdog Counter Process Data	Watchdog expiration counter Process Data
0x0443	Watchdog Counter PDI	Watchdog expiration counter PDI
0x0804 +N*8	SyncManager Control	Watchdog trigger enable

NOTE: Some of these registers are not available in specific ESCs. Refer to Section II for details.

13.1 Process Data Watchdog

The Process Data watchdog is rewound (triggered) by a write access to a SyncManager buffer area, if the SyncManager is configured to generate a watchdog trigger signal (SyncManager Control register 0x0804.6 for SyncManager 0, etc.). The watchdog trigger signal is generated after the buffer was completely and successfully written (similar to the Interrupt Write of a SyncManager).

The Process Data watchdog can be disabled by setting the Process Data Watchdog Time to 0.

A timeout of the Process Data watchdog has these consequences:

- Watchdog Status register for Process Data (0x0440.0) reflects the watchdog status.
- The Digital I/O PDI takes back digital output data, either by not driving the signals anymore or by driving them low (ESC and configuration dependent).
- The Watchdog Counter Process Data (0x0442) is incremented.

13.2 PDI Watchdog

The PDI watchdog is rewound (triggered) by any correct read or write access by the PDI. The PDI watchdog can be disabled by setting the PDI Watchdog Time to 0.

A timeout of the PDI watchdog has these consequences:

- ESC DL Status register (0x0110.1) reflects the watchdog status. This can be mapped to the ECAT Interrupt to inform the master.
- The Watchdog Counter PDI (0x0443) is incremented.

NOTE: The Digital I/O PDI only triggers the PDI watchdog upon input events.

14 Error Counters

The ESCs have numerous error counters which help in detecting and locating errors. All error counters are saturated at 0xFF (no wrap-around) and they are cleared individually or group-wise by writing any value to them.

Table 52: Error Counter Overview

Error Counter	Register	Description
Port Error Counters	0x0300:0x0307	Errors counted at the Auto-Forwarder (per port):
Invalid Frame Counter	0x0300/2/4/6	Invalid frame initially detected (includes RX Errors)
RX Error Counter	0x0301/3/5/7	Physical layer RX Errors (inside/outside frame): MII: RX_ER EBUS: Manchester-Violations
Forwarded RX Error Counter	0x0308:0x030B	Invalid frame with marking from previous ESC detected (per port)
ECAT Processing Unit Error Counter	0x030C	Invalid frame passing the EtherCAT Processing Unit (additional checks by processing unit)
PDI Error Counter	0x030D	Physical Errors detected by the PDI. Refer to PDI description in Section III for details.
Lost Link Counter	0x0310:0x0313	Link lost events (per port), counts only if port is in Auto or Auto close mode
Watchdog Counter Process Data	0x0442	Watchdog timeout events
Watchdog Counter PDI	0x0443	Watchdog timeout events

NOTE: Some errors will be counted in multiple registers. E.g., a physical layer RX Error received at port 0 is counted in registers 0x0300, 0x0301, and 0x030C. A forwarded error received at port 0 is counted in registers 0x0308 and 0x030C.

Some of these registers are not available in specific ESCs. Refer to Section II for details.

14.1 Frame error detection

EtherCAT frame error detection takes place at three functional blocks, at the physical layer (device), inside the Auto-Forwarder and inside the EtherCAT Processing Unit. The following errors are detected by these units:

Table 53: Errors Detected by Physical Layer, Auto-Forwarder, and EtherCAT Processing Unit

Physical Layer	Auto-Forwarder	EtherCAT Processing Unit
Registers 0x301/3/5/7	Registers 0x300/2/4/8 (original error) or registers 0x308/9/A/B (forwarded error)	Registers 0x30C
RX Errors: • MII/RMII: RX_ER event • EBUS: EBUS/Manchester code violations (refer to chapter 6.6)	<ul style="list-style-type: none"> • Physical Layer Errors (RX Errors) • Too long frames ($> \sim 2000$ Byte) • CRC errors • Frames without Ethernet SOF 	<ul style="list-style-type: none"> • Physical Layer Errors (RX Errors) • Auto-Forwarder Errors • EtherCAT frame length errors (e.g., frame ends although more header/data bytes are expected) • Too short frames (< 64 Byte) • Non-EtherCAT frames if register 0x0100.0=1 • Circulating bit=1 and port 0 automatically closed

Any of the above errors will have these consequences:

- The frame transmission is aborted (a frame with an RX Error at the beginning is truncated). The CRC of the transmitted data is modified (or appended) so that it becomes bad. A special marking for Forwarded Errors is added.
- The EtherCAT Processing Unit discards register operations, e.g., write operations to registers, SyncManager buffer changes, etc.. RAM areas will be written, because they do not have a shadow buffer for write data like registers.
- Error counters are increased.

14.2 Errors and Forwarded Errors

The ESCs distinguish errors initially detected by an ESC and forwarded errors detected by a previous ESC. This is useful for error location when interpreting the RX Error/Forwarded RX Error counters.

The first device detecting an error (e.g., a CRC error or an RX Error of the physical layer), will discard register operations and count a port error (0x0300-0x0307). The outgoing frame gets a special marking, consisting of one extra nibble added after the (invalid) CRC.

A device receiving a frame with a CRC error and an additional nibble will also discard register operations, but it will count one Forwarded RX Error instead of a normal port error.

NOTE: A forwarded error is sometimes called “green error”, the initial error is sometimes called “red error”. A physical layer RX Error is always a “red error”, because it could not have been forwarded.

15 LED Signals (Indicators)

EtherCAT slave controllers support different LEDs regarding link state and AL status. For details about EtherCAT indicators refer to the ETG.1300 EtherCAT Indicator and Labeling Specification, available from the download section of the EtherCAT Technology Group website (<http://www.ethercat.org>).

15.1 RUN LED

The AL status is displayed with the RUN LED (green). The RUN output of an ESC is controlled by the AL status register (0x0130) and supports the following states:

Table 54: RUN LED States

RUN LED	Description
Off	The device is in state INIT
Blinking (slow)	The device is in state PRE-OPERATIONAL
Single Flash	The device is in state SAFE-OPERATIONAL
On	The device is in state OPERATIONAL
Flickering (fast)	The device is in state BOOTSTRAP

Some ESCs support optional RUN LED outputs by overriding the state indication of the RUN LED. The output can be set by master or local application. This can be used e.g. for locating a specific slave by forcing the RUN LED to indicate a triple flash (Device Identification).

15.2 ERR LED

The ERR LED indicates application errors. It is either sourced by the application controller, or by the ESC (if supported). If the ESC supports the ERR LED, the local application (and the master) is able to control the ERR LED. Some errors which can be detected by the ESC are directly indicated.

NOTE: Do not confuse the application ERR LED with the port receive error LEDs (PERR(x)) supported by some ESCs.

15.3 STATE LED and STATE_RUN LED Signal

The STATE LED is a bicolor-LED combining RUN and ERR LED. Since the RUN LED part of the STATE LED must be turned off while the ERR LED part is active, the RUN and ERR LED signals cannot be simply combined to drive the bicolor LED. Some ESCs support a STATE_RUN signal, which is turned off while ERR LED is on, so STATE_RUN and ERR signals can be used to drive the bicolor STATE LED. Otherwise the logical combination “RUN and not(ERR)” has to be used to control the RUN LED part of the STATE LED.

15.4 LINKACT LED

The Link/Activity state of each port is displayed with the LINKACT LED (green).

Table 55: LINKACT LED States

LINKACT LED	Description
Off	No link
Blinking	Link and activity
On	Link without activity

It is recommended to use the LINKACT LED signals of the ESCs instead of the Link/Activity LED signals of the PHY, because the ESC signals reflect the actual link/activity state of the device – not only the state of the PHYs –, and the ESC signals adhere to the ETG.1300 EtherCAT Indicator and Labeling Specification.

15.5 Port Error LED (PERR)

Some ESCs support port receive error indicators PERR(x), which display physical layer RX errors. The PERR(x) LEDs are not part of the ETG.1300 EtherCAT Indicator and Labeling Specification. They are only intended for testing and debugging. The PERR(x) LEDs flash once if a physical layer RX error occurs.

16 Process Data Interface (PDI)

The Process Data Interface (PDI) realizes the connection between slave application and ESC. Several types of PDIs are defined, e.g., serial and parallel µController interfaces and Digital I/O interfaces. Table 56 gives an overview of the available PDI types for each ESC.

Due to the high dependency between EtherCAT and PDI accesses to memory, registers, and especially SyncManagers, the internal PDI interface can achieve a maximum throughput of approx. 12.5 MByte/s.

Details on individual PDI functionality can be found in Section III of each ESC.

Table 56: Available PDIs depending on ESC

PDI number (PDI Control register 0x0140[7:0])	PDI name	ESC20	IP Core	ET1100	ET1200
0	Interface deactivated	x	x	x	x
4	Digital I/O		x	x	x
5	SPI Slave	x	x	x	x
7	EtherCAT Bridge (port 3)				x
8	16 Bit async. µC	x	x	x	
9	8 Bit async. µC	x	x	x	
10	16 Bit sync. µC			x	
11	8 Bit sync. µC			x	
16	32 Digital Input/0 Digital Output	x			
17	24 Digital Input/8 Digital Output	x			
18	16 Digital Input/16 Digital Output	x			
19	8 Digital Input/24 Digital Output	x			
20	0 Digital Input/32 Digital Output	x			
128	On-chip bus (Avalon or PLB/OPB)		x		
Others	Reserved				

NOTE: On-Chip bus: the EtherCAT IP Core for Altera FPGAs supports Avalon bus, the EtherCAT IP Core for Xilinx FPGAs supports PLB/OPB.

16.1 PDI Selection and Configuration

Typically, the PDI selection and configuration is part of the ESC Configuration Area of the SII EEPROM. Some ESCs (IP Core) have the PDI selected and configured at generation time, the ESC Configuration Area should reflect the actual settings, although they are not evaluated by the ESC itself.

For most ESCs, the PDI becomes active after the SII EEPROM is successfully loaded. All PDI pins are inactive (high impedance) until then (as well as the DC Sync/Latch signals). Some ESCs and PDIs provide an EEPROM_Loaded signal, which indicates that the EEPROM is successfully loaded and the PDI can be used. Attach a pull-down resistor to the EEPROM_Loaded pin, because it is also not driven (high impedance) until the EEPROM is successfully loaded. The PDI of an IP Core is active after reset is released, which enables e.g. EEPROM emulation by a µController.

Take care of Digital Output signals and DC SyncSignals while the EEPROM is not loaded to achieve proper output behavior.

16.2 General Purpose I/O

Some ESCs support general purpose inputs, outputs, or both, depending on the selected PDI or even independent of the PDI (IP Core).

General Purpose I/O is much different from Digital I/O, because of missing consistency and security features, and no bit-write support.

16.2.1 General Purpose Inputs

The general purpose inputs are directly mapped into the General Purpose Input registers. Consistency of the general purpose inputs is not provided.

16.2.2 General Purpose Output

The general purpose output signals reflect the values of the General Purpose Output register without watchdog protection. The General Purpose Output register can be written both by ECAT and PDI. The general purpose outputs are intended e.g. for application specific LED outputs. General purpose outputs are updated at the end of an EtherCAT frame or at the end of the PDI access.

Consistency of the general purpose outputs is not provided, and they are also not watchdog-secured. Additionally, they do not support bit-wise modification with FMMUs.

17 Additional Information

17.1 ESC Clock Source

The initial accuracy of the ESC clock sources has to be 25ppm or better. This enables FIFO size reduction, i.e., forwarding delay reduction. Existing designs do not need to be changed.

17.2 Power-on Sequence

The power-on sequence of ESCs looks like this:

Table 57: ESC Power-On Sequence

No.	Step	Result
1	Power-on	Voltages reach proper levels ASICs only: Power-on values are sampled
2 (FPGA only)	Loading FPGA configuration	FPGA loads its hardware configuration
3	PLL locks	Clocks are generated properly
4	Release RESET	ESC operation begins. Process memory is not accessible until the SII EEPROM is loaded, as well as any function depending on ESC Configuration data. IP Core : PDI is operational; others: PDI is not operational until EEPROM is loaded.
5*	Links are established	EtherCAT communication begins, master can access ESC registers
6*	Loading ESC EEPROM	Only upon successful EEPROM loading: <ul style="list-style-type: none"> • ESC Configuration registers initialized • PDI is activated (not IP Core: active after RESET) • PDI operation begins • Register 0x0110.0 turns to 1 • Process Data RAM becomes accessible • Some PDIs: EEPROM_Loaded signal is driven high • ESC is in Init state
7	Example: Master proceeds to Operational state	ESC proceeds to Operational state

* Steps 5 and 6 are executed in parallel.

NOTE: The PDI signals are not driven until the ESC EEPROM is loaded successfully, especially the EEPROM_Loaded signal is not driven and needs a pull-down resistor if it is used.

17.3 Write Protection

Some ESCs are capable of register write protection or entire ESC write protection.

Registers used for write protection are described in Table 58:

Table 58: Registers for Write Protection

Register Address	Name	Description
0x0020	Write Register Enable	Temporarily release register write protection
0x0021	Write Register Protection	Activate register write protection
0x0030	ESC Write Enable	Temporarily release ESC write protection
0x0031	ESC Write Protection	Activate ESC write protection

NOTE: Some of these registers are not available in specific ESCs. Refer to Section II for details.

17.3.1 Register Write Protection

With register write protection, only the register area (0x0000 to 0x0FFF) is write protected (except for registers 0x0020 and 0x0030).

If register write protection is enabled (register 0x0021.0=1), the Register Write Enable bit (0x0020.0) has to be set in the same frame before any register write operations. This is also true for disabling the register write protection. Otherwise, write operation to registers are discarded.

17.3.2 ESC Write Protection

ESC write protection disables write operations to any memory location (except for registers 0x0020 and 0x0030).

If ESC write protection is enabled (register 0x0031.0=1), the ESC Write Enable bit (0x0030.0) has to be set in the same frame before any write operations. This is also true for disabling the ESC write protection as well as the register write protection. Otherwise, write operations are discarded.

NOTE: If both register write protection and ESC write protection are enabled (not recommended), both enable bits have to be set before the write operations are allowed.

17.4 ESC Reset

Some ESCs (e.g., ET1100, ET1200, and IP Core) are capable of issuing a hardware reset by the EtherCAT master or even via the PDI. A special sequence of three independent and consecutive frames/commands has to be sent to the slave (Reset register ECAT 0x0040 or PDI 0x0041). Afterwards, the slave is reset.

NOTE: It is likely that the last frame of the sequence will not return to the master (depending on the topology), because the links to and from the slave which is reset will go down.

18 Appendix

18.1 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.

18.1.1 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.

18.2 Beckhoff Headquarters

Beckhoff Automation GmbH
Eiserstr. 5
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

Hardware Data Sheet

EtherCAT® Slave Controller

Section II – Register Description

Register overview and detailed description

Version 2.5
Date: 2011-03-15

BECKHOFF

Trademarks

Beckhoff[®], TwinCAT[®], EtherCAT[®], Safety over EtherCAT[®], TwinSAFE[®] and XFC[®] are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following German patent applications and patents: DE10304637, DE102004044764, DE102005009224, DE102007017835 with corresponding applications or registrations in various other countries.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

© Beckhoff Automation GmbH 03/2011.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

DOCUMENT HISTORY

Version	Comment
1.0	Initial release
1.1	<ul style="list-style-type: none"> • Latch0/1 state register bit 0x09AE.2 and 0x09AF.2 added (ET1100 and IP Core) • On-chip Bus configuration for Avalon®: Extended PDI configuration register 0x0152[1:0] added
1.2	<ul style="list-style-type: none"> • On-chip Bus configuration: Extended PDI configuration register 0x0152[1:0] now valid for both Avalon and OPB • ESC DL Status: PDI Watchdog Status constantly 1 for ESC10 • EEPROM Control/Status: Selected EEPROM Algorithm not readable for ESC10/20
1.3	<ul style="list-style-type: none"> • EEPROM/MII Management Interface: Added self-clearing feature of command register • SPI extended configuration (0x0152:0x0153): Reset Value is EEPROM ADR 0x0003, not 0x0001 • ESC DL Control (0x0100.0): Added details about Source MAC address change • Power-On Values ET1100 (0x0E000): P_CONF does not correspond with physical ports
1.4	<ul style="list-style-type: none"> • Sync/Latch PDI configuration register: Latch configuration clarified • AL Control register: mailbox behavior described • Editorial changes
1.5	<ul style="list-style-type: none"> • ESC DL Control (0x0100:0x0103): FIFO Size description enhanced • IP Core: Extended Features (reset value of User RAM 0x0F80:0xFFFF) added • MII Management Interface: Write access by PDI is only possible for ET1100 if Transparent Mode is enabled. Corrected register read/write descriptions. • MII Management Control/Status register (0x0510:0x0511): Error bit description clarified. Write Enable bit is self-clearing. • ESC DL Control (0x0100:0x0103): Temporary setting DL not available for ESC10/20 • EEPROM PDI Access State register (0x0501): write access depends on EEPROM configuration • EEPROM Control/Status register (0x0502:0x0503): Error bit description clarified. Write Enable bit is self-clearing. • Registers initialized from EEPROM have Reset value 0, and EEPROM value after EEPROM was loaded successful • AL Event Request (0x0220:0x0223) description clarified: SyncManager configuration changed interrupt indicates activation register changes. • DC Latch0/1 Status (0x09AE:0x09AF): Event flags are only available in Single event mode • DC SYNC0 Cycle Time (0x09A0:0x09A3): Value of 0 selects single pulse generation • 64 Bit Receive Time ECAT Processing Unit (0x0918:0x091F) is also available for 32 Bit DCs. Renamed register to Receive Time ECAT Processing Unit • RAM Size (0x0006) ET1200: 1 Kbyte • Editorial changes

Version	Comment
1.6	<ul style="list-style-type: none"> • EEPROM Control/Status register (0x0502:0x0503): Error bit description clarified • EEPROM Interface and MII Management Interface: access to special registers is blocked while interface is busy • EEPROM Interface: EEPROM emulation by PDI added • Extended IP Core features (0x0F80:0x0FFF): reset values moved to Section III • Reset values of DC Receive Time registers are undefined • MI Control/Status register bit 0x510.7 is read only • FMMUs supported (0x0004): ET1200 has 3 FMMUs, not 4 • AL Event Request register: SyncManager changed flag (0x220.4) is not available in IP Core versions before and including 1.1.1/1.01b • Configured Station Alias (0x0012:0x0013) is only taken over at first EEPROM load after power-on or reset • Moved available PDIs depending on ESC to Section I • SyncManager PDI Control (0x807 etc.): difference between read and write access described • General Purpose I/O registers (0x0F10:0x0F1F) width variable (1/2/4/8 Byte) • MII Management Interface enhancement: link detection and assignment to PDI added • Write access to DC Time Loop Control unit by PDI configurable for IP Core (V2.0.0/2.00a) • Editorial changes
1.7	<ul style="list-style-type: none"> • MII Management Control/Status (0x0510) updated: PHY address offset is 5 bits, feature bits have moved • System time register (0x0910:0x0917): clarified functionality • Process Data RAM (0x1000 ff.): accessible only if EEPROM is loaded • Digital I/O extended configuration (0x0152:0x0153): Set to 0 in bidirectional mode • Editorial changes
1.8	<ul style="list-style-type: none"> • DC register accessibility depends on DC power saving settings in PDI Control register (0x0140[11:10]) • AL Event Request register (0x0220): AL Control Event (Bit 0) is cleared by reading AL Control register (0x0120), not AL Event Request register • EEPROM Control/Status register bit 0x0502.12 renamed to EEPROM loading status • Description of Push-Pull/Open-Drain output drivers for SPI, µController, and SYNC0/1 enhanced • Speed Counter Start register (0x0930:0x0931): Write access resets calculated Time Loop Control values • Speed Counter Diff register (0x0932:0x0933): Deviation calculation added • DC Start Time Cyclic operation (0x0990:0x0997) and Next Sync1 Pulse (0x0998:0x099F) relate to the System time • Reset DC Control loop (write 0x0930:0x0931) after changing filter depths (0x0934 or 0x0935) • Editorial changes
1.9	<ul style="list-style-type: none"> • Update to EtherCAT IP Core Release 2.2.0/2.02a • Register availability added • Writing to DC Filter Depth registers 0x0934:0x0935 resets filters • DC Activation register (0x0981) enhanced • DC Activation state register (0x0984) added • Reserved registers or register bits: write 0, ignore read values • Enhanced link detection 0x0140.9 has compatibility issues with EBUS ports, not MII ports • Port dependent Enhanced link detection (0x0140[15:12] added) • PHY Port y Status bit 5 added (port configuration updated) • ESC10 removed • Editorial changes

Version	Comment
2.0	<ul style="list-style-type: none"> DC SYNC Activation register (0x0981.6): bit polarity corrected Deviation calculation formula for Speed Counter Diff register (0x0932:0x0933) corrected AL Event Mask register (0x0204:0x0207): corresponding to AL Event Request register bits, not to ECAT Event Request register bits Register availability noted in ESC availability tabs Register Digital I/O configuration (0x0150): corrected OUTVALID mode = 1 description Power-on values ET1200 (0x0E00.6): CLK25OUT on PDI[6], not PDI[31] Editorial changes
2.1	<ul style="list-style-type: none"> Register bit 0x0220.4 is not available for ESC20 DC System Time (0x0910:0x0917): read value differs between ECAT and PDI DC Latch Times and DC Event Times are internally latched when lowest byte is read DC Speed Counter Start (0x0930:0x0931): minimum value is 0x80 Editorial changes
2.2	<ul style="list-style-type: none"> ESC20: Register Configured Station Alias (0x0012:0x0013) is taken over after each EEPROM reload command MII Management Control register 0x0510[0]: Updated to ET1100-0002 Registers 0x0020 and 0x0030 are readable for ET1100 and ET1200 Editorial changes
2.3	<ul style="list-style-type: none"> Update to EtherCAT IP Core Release 2.3.0/2.03a (registers 0x0138/0x0139, 0x0150 On-chip Bus, 0x0220, 0x030E, 0x0805 affected) Separated registers 0x0140 (PDI Control) and 0x0141 (now: ESC Configuration) Editorial changes
2.4	<ul style="list-style-type: none"> ESC DL Control register (0x0100.0): Source MAC address bit is set regardless of forwarding rule. Added ESC Feature Bits 0x0008[11:9] Update to EtherCAT IP Core Release 2.3.2/2.03c ESC Features 0x0008 and ESC Configuration 0x0141[1]: Enhanced Link Detection must not be activated for ET1100/ET1200 if EBUS ports are used. Editorial changes
2.5	<ul style="list-style-type: none"> Update to EtherCAT IP Core Release 2.4.0/2.04a ESC20: 0x0140[1:0] and [5:4] are available for SPI PDI Range for DC Speed Counter Start (0x0930:0x0931) and Speed Counter Diff (0x0932:0x0933) corrected, representation of Speed Counter Diff mentioned.

CONTENTS

Section II – Register Description	1
1 Address Space Overview	1
1.1 Scope of Section II	4
1.2 Reserved Registers/Reserved Register Bits	4
1.3 ESC Availability Tab Legend	5
2 ESC Register Availability	6
3 Register description	9
3.1 Type (0x0000)	9
3.2 Revision (0x0001)	9
3.3 Build (0x0002:0x0003)	9
3.4 FMMUs supported (0x0004)	9
3.5 SyncManagers supported (0x0005)	10
3.6 RAM Size (0x0006)	10
3.7 Port Descriptor (0x0007)	10
3.8 ESC Features supported (0x0008:0x0009)	11
3.9 Configured Station Address (0x0010:0x0011)	12
3.10 Configured Station Alias (0x0012:0x0013)	12
3.11 Write Register Enable (0x0020)	12
3.12 Write Register Protection (0x0021)	13
3.13 ESC Write Enable (0x0030)	13
3.14 ESC Write Protection (0x0031)	13
3.15 ESC Reset ECAT (0x0040)	14
3.16 ESC Reset PDI (0x0041)	14
3.17 ESC DL Control (0x0100:0x0103)	15
3.18 Physical Read/Write Offset (0x0108:0x0109)	16
3.19 ESC DL Status (0x0110:0x0111)	17
3.20 AL Control (0x0120:0x0121)	19
3.21 AL Status (0x0130:0x0131)	19
3.22 AL Status Code (0x0134:0x0135)	20
3.23 RUN LED Override (0x0138)	20
3.24 ERR LED Override (0x0139)	20
3.25 PDI Control (0x0140)	21
3.26 ESC Configuration (0x0141)	22
3.27 PDI Configuration (0x0150:0x0153)	23
3.27.1 Digital I/O configuration	24
3.27.2 SPI Slave Configuration	26
3.27.3 8/16Bit asynchronous Microcontroller configuration	27
3.27.4 8/16Bit synchronous Microcontroller configuration	28
3.27.5 EtherCAT Bridge (port 3)	30

3.27.6	On-chip bus configuration	31
3.27.7	Sync/Latch PDI Configuration	32
3.28	ECAT Event Mask (0x0200:0x0201)	33
3.29	AL Event Mask (0x0204:0x0207)	33
3.30	ECAT Event Request (0x0210:0x0211)	34
3.31	AL Event Request (0x0220:0x0223)	35
3.32	RX Error Counter (0x0300:0x0307)	37
3.33	Forwarded RX Error Counter (0x0308:0x030B)	37
3.34	ECAT Processing Unit Error Counter (0x030C)	37
3.35	PDI Error Counter (0x030D)	38
3.36	PDI Error Code (0x030E)	38
3.36.1	SPI PDI Error Code	38
3.36.2	Asynchronous/Synchronous Microcontroller PDI Error Code	38
3.37	Lost Link Counter (0x0310:0x0313)	39
3.38	Watchdog Divider (0x0400:0x0401)	40
3.39	Watchdog Time PDI (0x0410:0x0411)	40
3.40	Watchdog Time Process Data (0x0420:0x0421)	40
3.41	Watchdog Status Process Data (0x0440:0x0441)	41
3.42	Watchdog Counter Process Data (0x0442)	41
3.43	Watchdog Counter PDI (0x0443)	41
3.44	SII EEPROM Interface (0x0500:0x050F)	42
3.44.1	EEPROM emulation with IP Core	45
3.45	MII Management Interface (0x0510:0x0515)	46
3.46	FMMU (0x0600:0x06FF)	51
3.47	SyncManager (0x0800:0x087F)	53
3.48	Distributed Clocks (0x0900:0x09FF)	57
3.48.1	Receive Times	58
3.48.2	Time Loop Control Unit	60
3.48.3	Cyclic Unit Control	63
3.48.4	SYNC Out Unit	64
3.48.5	Latch In unit	67
3.48.6	SyncManager Event Times	70
3.49	ESC specific registers (0x0E00:0x0EFF)	71
3.49.1	Power-On Values ET1200	71
3.49.2	Power-On Values ET1100	72
3.49.3	IP Core	73
3.49.4	ESC20	73
3.50	Digital I/O Output Data (0x0F00:0x0F03)	74
3.51	General Purpose Outputs (0x0F10:0x0F17)	74
3.52	General Purpose Inputs (0x0F18:0x0F1F)	74
3.53	User RAM (0x0F80:0x0FFF)	75

4	Process Data RAM (0x1000:0xFFFF)	77
4.1	Digital I/O Input Data (0x1000:0x1003)	77
4.2	Process Data RAM (0x1000:0xFFFF)	77
5	Appendix	78
5.1	Support and Service	78
5.1.1	Beckhoff's branch offices and representatives	78
5.2	Beckhoff Headquarters	78

TABLES

Table 1: ESC address space	1
Table 2: ESC Register Availability	6
Table 3: ESC Register Availability Legend	8
Table 4: Register Type (0x0000)	9
Table 5: Register Revision (0x0001)	9
Table 6: Register Build (0x0002:0x0003)	9
Table 7: Register FMMUs supported (0x0004)	9
Table 8: Register SyncManagers supported (0x0005)	10
Table 9: Register RAM Size (0x0006)	10
Table 10: Register Port Descriptor (0x0007)	10
Table 11: Register ESC Features supported (0x0008:0x0009)	11
Table 12: Register Configured Station Address (0x0010:0x0011)	12
Table 13: Register Configured Station Alias (0x0012:0x0013)	12
Table 14: Register Write Register Enable (0x0020)	12
Table 15: Register Write Register Protection (0x0021)	13
Table 16: Register ESC Write Enable (0x0030)	13
Table 17: Register ESC Write Protection (0x0031)	13
Table 18: Register ESC Reset ECAT (0x0040)	14
Table 19: Register ESC Reset PDI (0x0041)	14
Table 20: Register ESC DL Control (0x0100:0x0103)	15
Table 21: Register Physical Read/Write Offset (0x0108:0x0109)	16
Table 22: Register ESC DL Status (0x0110:0x0111)	17
Table 23: Decoding port state in ESC DL Status register 0x0111 (typical modes only)	18
Table 24: Register AL Control (0x0120:0x0121)	19
Table 25: Register AL Status (0x0130:0x0131)	19
Table 26: Register AL Status Code (0x0134:0x0135)	20
Table 27: Register RUN LED Override (0x0138)	20
Table 28: Register ERR LED Override (0x0139)	20
Table 29: Register PDI Control (0x0140)	21
Table 30: Register ESC Configuration (0x0141)	22
Table 31: PDI Configuration Register overview	23
Table 32: Register Digital I/O configuration (0x0150)	24
Table 33: Register Digital I/O extended configuration (0x0152:0x0153)	25
Table 34: Register SPI Configuration (0x0150)	26
Table 35: Register SPI extended configuration (0x0152:0x0153)	26
Table 36: Register asynchronous Microcontroller Configuration (0x0150)	27
Table 37: Register Asynchronous Microcontroller extended Configuration (0x0152:0x0153)	27
Table 38: Register Synchronous Microcontroller Configuration (0x0150)	28
Table 39: Register Synchronous Microcontroller extended Configuration (0x0152:0x0153)	29
Table 40: Register EtherCAT Bridge configuration (0x0150)	30
Table 41: Register EtherCAT Bridge extended configuration (0x0152:0x0153)	30
Table 42: Register On-chip bus configuration (0x0150)	31
Table 43: Register On-chip bus extended configuration (0x0152:0x0153)	31
Table 44: Register Sync/Latch PDI Configuration (0x0151)	32
Table 45: Register ECAT Event Mask (0x0200:0x0201)	33
Table 46: Register AL Event Mask (0x0204:0x0207)	33
Table 47: Register ECAT Event Request (0x0210:0x0211)	34
Table 48: Register AL Event Request (0x0220:0x0223)	35
Table 49: Register RX Error Counter Port y (0x0300+y*2:0x0301+y*2)	37
Table 50: Register Forwarded RX Error Counter Port y (0x0308+y)	37
Table 51: Register ECAT Processing Unit Error Counter (0x030C)	37
Table 52: Register PDI Error Counter (0x030D)	38
Table 53: Register SPI PDI Error Code (0x030E)	38
Table 54: Register Microcontroller PDI Error Code (0x030E)	38
Table 55: Register Lost Link Counter Port y (0x0310+y)	39
Table 56: Register Watchdog Divider (0x0400:0x0401)	40
Table 57: Register Watchdog Time PDI (0x0410:0x0411)	40
Table 58: Register Watchdog Time Process Data (0x0420:0x0421)	40
Table 59: Register Watchdog Status Process Data (0x0440:0x0441)	41
Table 60: Register Watchdog Counter Process Data (0x0442)	41

Table 61: Register Watchdog Counter PDI (0x0443).....	41
Table 62: SII EEPROM Interface Register overview.....	42
Table 63: Register EEPROM Configuration (0x0500).....	42
Table 64: Register EEPROM PDI Access State (0x0501)	42
Table 65: Register EEPROM Control/Status (0x0502:0x0503)	43
Table 66: Register EEPROM Address (0x0504:0x0507)	44
Table 67: Register EEPROM Data (0x0508:0x050F [0x0508:0x050B])	44
Table 68: Register EEPROM Data for EEPROM Emulation Reload IP Core (0x0508:0x050F)	45
Table 69: MII Management Interface Register Overview	46
Table 70: Register MII Management Control/Status (0x0510:0x0511).....	47
Table 71: Register PHY Address (0x0512)	48
Table 72: Register PHY Register Address (0x0513).....	48
Table 73: Register PHY Data (0x0514:0x0515)	48
Table 74: Register MII Management ECAT Access State (0x0516)	49
Table 75: Register MII Management PDI Access State (0x0517)	49
Table 76: Register PHY Port y (port number y=0 to 3) Status (0x0518+y)	50
Table 77: FMMU Register overview	51
Table 78: Register Logical Start address FMMU y (0x06y0:0x06y3).....	51
Table 79: Register Length FMMU y (0x06y4:0x06y5).....	51
Table 80: Register Start bit FMMU y in logical address space (0x06y6)	51
Table 81: Register Stop bit FMMU y in logical address space (0x06y7)	52
Table 82: Register Physical Start address FMMU y (0x06y8-0x06y9).....	52
Table 83: Register Physical Start bit FMMU y (0x06yA)	52
Table 84: Register Type FMMU y (0x06yB)	52
Table 85: Register Activate FMMU y (0x06yC)	52
Table 86: Register Reserved FMMU y (0x06yD:0x06yF)	52
Table 87: SyncManager Register overview.....	53
Table 88: Register physical Start Address SyncManager y (0x0800+y*8:0x0801+y*8)	53
Table 89: Register Length SyncManager y (0x0802+y*8:0x0803+y*8)	53
Table 90: Register Control Register SyncManager y (0x0804+y*8)	54
Table 91: Register Status Register SyncManager y (0x0805+y*8)	55
Table 92: Register Activate SyncManager y (0x0806+y*8)	56
Table 93: Register PDI Control SyncManager y (0x0807+y*8)	56
Table 94: Distributed Clocks Register overview.....	57
Table 95: Register Receive Time Port 0 (0x0900:0x0903)	58
Table 96: Register Receive Time Port 1 (0x0904:0x0907)	58
Table 97: Register Receive Time Port 2 (0x0908:0x090B)	58
Table 98: Register Receive Time Port 3 (0x090C:0x090F)	59
Table 99: Register Receive Time ECAT Processing Unit (0x0918:0x091F).....	59
Table 100: Register System Time (0x0910:0x0913 [0x0910:0x0917])	60
Table 101: Register System Time Offset (0x0920:0x0923 [0x0920:0x0927])	60
Table 102: Register System Time Delay (0x0928:0x092B)	61
Table 103: Register System Time Difference (0x092C:0x092F).....	61
Table 104: Register Speed Counter Start (0x0930:0x931)	61
Table 105: Register Speed Counter Diff (0x0932:0x933)	61
Table 106: Register System Time Difference Filter Depth (0x0934).....	62
Table 107: Register Speed Counter Filter Depth (0x0935).....	62
Table 108: Register Cyclic Unit Control (0x0980)	63
Table 109: Register Activation register (0x0981)	64
Table 110: Register Pulse Length of SyncSignals (0x0982:0x983)	65
Table 111: Register Activation Status (0x0984)	65
Table 112: Register SYNC0 Status (0x098E)	65
Table 113: Register SYNC1 Status (0x098F)	65
Table 114: Register Start Time Cyclic Operation (0x0990:0x0993 [0x0990:0x0997])	66
Table 115: Register Next SYNC1 Pulse (0x0998:0x099B [0x0998:0x099F])	66
Table 116: Register SYNC0 Cycle Time (0x09A0:0x09A3)	66
Table 117: Register SYNC1 Cycle Time (0x09A4:0x09A7)	66
Table 118: Register Latch0 Control (0x09A8)	67
Table 119: Register Latch1 Control (0x09A9)	67
Table 120: Register Latch0 Status (0x09AE)	68
Table 121: Register Latch1 Status (0x09AF)	68
Table 122: Register Latch0 Time Positive Edge (0x09B0:0x09B3 [0x09B0:0x09B7]).....	69

Table 123: Register Latch0 Time Negative Edge (0x09B8:0x09BB [0x09B8:0x09BF])	69
Table 124: Register Latch1 Time Positive Edge (0x09C0:0x09C3 [0x09C0:0x09C7])	69
Table 125: Register Latch1 Time Negative Edge (0x09C8:0x09CB [0x09C8:0x09CF])	69
Table 126: Register EtherCAT Buffer Change Event Time (0x09F0:0x09F3)	70
Table 127: Register PDI Buffer Start Event Time (0x09F8:0x09FB).....	70
Table 128: Register PDI Buffer Change Event Time (0x09FC:0x09FF)	70
Table 129: Register Power-On Values ET1200 (0x0E00)	71
Table 130: Register Power-On Values ET1100 (0x0E00:0xE01)	72
Table 131: Register Product ID (0x0E00:0xE07)	73
Table 132: Register Vendor ID (0x0E08:0xE0F).....	73
Table 133: Register FPGA Update (0x0E00:0xEFF)	73
Table 134: Register Digital I/O Output Data (0xF00:0xF03)	74
Table 135: Register General Purpose Outputs (0xF10:0xF17).....	74
Table 136: Register General Purpose Inputs (0xF18:0xF1F)	74
Table 137: User RAM (0xF80:0xFFFF)	75
Table 138: Extended ESC Features (Reset values of User RAM).....	75
Table 139: Digital I/O Input Data (0x1000:0x1003)	77
Table 140: Process Data RAM (0x1000:0xFFFF)	77

ABBREVIATIONS

ADR	Address
AL	Application Layer
APRW	Auto Increment Physical ReadWrite
BHE	Bus High Enable
BWR	Broadcast Write
DC	Distributed Clock
DL	Data Link Layer
ECAT	EtherCAT
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information
FCS	Frame Check Sequence
FMMU	Fieldbus Memory Management Unit
FPRD	Configured Address Physical Read
FPRW	Configured Address Physical ReadWrite
FPWR	Configured Address Physical Write
GPI	General Purpose Input
GPO	General Purpose Output
IP	Intellectual Property
μC	Microcontroller
MI	(PHY) Management Interface
MII	Media Independent Interface
OPB	On-Chip Peripheral Bus
PDI	Process Data Interface
RMII	Reduced Media Independent Interface
SII	Slave Information Interface
SM	SyncManager
SOC	System on a Chip
SOF	Start of Frame
SoPC	System on a Programmable Chip
SPI	Serial Peripheral Interface
WD	Watchdog

1 Address Space Overview

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 depends on the ESC. The availability of the registers depends on the ESC.

Table 1: ESC address space

Address ¹	Length (Byte)	Description
ESC Information		
0x0000	1	Type
0x0001	1	Revision
0x0002:0x0003	2	Build
0x0004	1	FMMUs supported
0x0005	1	SyncManagers supported
0x0006	1	RAM Size
0x0007	1	Port Descriptor
0x0008:0x0009	2	ESC Features supported
Station Address		
0x0010:0x0011	2	Configured Station Address
0x0012:0x0013	2	Configured Station Alias
Write Protection		
0x0020	1	Write Register Enable
0x0021	1	Write Register Protection
0x0030	1	ESC Write Enable
0x0031	1	ESC Write Protection
Data Link Layer		
0x0040	1	ESC Reset ECAT
0x0041	1	ESC Reset PDI
0x0100:0x0103	4	ESC DL Control
0x0108:0x0109	2	Physical Read/Write Offset
0x0110:0x0111	2	ESC DL Status
Application Layer		
0x0120:0x0121	2	AL Control
0x0130:0x0131	2	AL Status
0x0134:0x0135	2	AL Status Code
0x0138	1	RUN LED Override
0x0139	1	ERR LED Override
PDI		
0x0140	1	PDI Control
0x0141	1	ESC Configuration
0x0150	4	PDI Configuration
0x0151	4	SYNC/LATCH PDI Configuration
0x0152:0x0153	4	Extended PDI Configuration

¹ Address areas not listed here are reserved. They are not writable. A read access to reserved addresses will typically return 0.

Address¹	Length (Byte)	Description
Interrupts		
0x0200:0x0201	2	ECAT Event Mask
0x0204:0x0207	4	AL Event Mask
0x0210:0x0211	2	ECAT Event Request
0x0220:0x0223	4	AL Event Request
Error Counters		
0x0300:0x0307	4x2	Rx Error Counter[3:0]
0x0308:0x030B	4x1	Forwarded Rx Error counter[3:0]
0x030C	1	ECAT Processing Unit Error Counter
0x030D	1	PDI Error Counter
0x030E	1	PDI Error Code
0x0310:0x0313	4x1	Lost Link Counter[3:0]
Watchdogs		
0x0400:0x0401	2	Watchdog Divider
0x0410:0x0411	2	Watchdog Time PDI
0x0420:0x0421	2	Watchdog Time Process Data
0x0440:0x0441	2	Watchdog Status Process Data
0x0442	1	Watchdog Counter Process Data
0x0443	1	Watchdog Counter PDI
SII EEPROM Interface		
0x0500	1	EEPROM Configuration
0x0501	1	EEPROM PDI Access State
0x0502:0x0503	2	EEPROM Control/Status
0x0504:0x0507	4	EEPROM Address
0x0508:0x050F	4/8	EEPROM Data
MII Management Interface		
0x0510:0x0511	2	MII Management Control/Status
0x0512	1	PHY Address
0x0513	1	PHY Register Address
0x0514:0x0515	2	PHY Data
0x0516	1	MII Management ECAT Access State
0x0517	1	MII Management PDI Access State
0x0518:0x051B	4	PHY Port Status
0x0600:0x06FF	16x16	FMMU[15:0]
+0x0:0x3	4	Logical Start Address
+0x4:0x5	2	Length
+0x6	1	Logical Start bit
+0x7	1	Logical Stop bit
+0x8:0x9	2	Physical Start Address
+0xA	1	Physical Start bit
+0xB	1	Type
+0xC	1	Activate
+0xD:0xF	3	Reserved

Address ¹	Length (Byte)	Description
0x0800:0x087F	16x8	SyncManager[15:0]
+0x0:0x1	2	Physical Start Address
+0x2:0x3	2	Length
+0x4	1	Control Register
+0x5	1	Status Register
+0x6	1	Activate
+0x7	1	PDI Control
0x0900:0x09FF		Distributed Clocks (DC)
		DC – Receive Times
0x0900:0x0903	4	Receive Time Port 0
0x0904:0x0907	4	Receive Time Port 1
0x0908:0x090B	4	Receive Time Port 2
0x090C:0x090F	4	Receive Time Port 3
		DC – Time Loop Control Unit
0x0910:0x0917	4/8	System Time
0x0918:0x091F	4/8	Receive Time ECAT Processing Unit
0x0920:0x0927	4/8	System Time Offset
0x0928:0x092B	4	System Time Delay
0x092C:0x092F	4	System Time Difference
0x0930:0x0931	2	Speed Counter Start
0x0932:0x0933	2	Speed Counter Diff
0x0934	1	System Time Difference Filter Depth
0x0935	1	Speed Counter Filter Depth
		DC – Cyclic Unit Control
0x0980	1	Cyclic Unit Control
		DC – SYNC Out Unit
0x0981	1	Activation
0x0982:0x0983	2	Pulse Length of SyncSignals
0x0984	1	Activation Status
0x098E	1	SYNC0 Status
0x098F	1	SYNC1 Status
0x0990:0x0997	4/8	Start Time Cyclic Operation/Next SYNC0 Pulse
0x0998:0x099F	4/8	Next SYNC1 Pulse
0x09A0:0x09A3	4	SYNC0 Cycle Time
0x09A4:0x09A7	4	SYNC1 Cycle Time
		DC – Latch In Unit
0x09A8	1	Latch0 Control
0x09A9	1	Latch1 Control
0x09AE	1	Latch0 Status
0x09AF	1	Latch1 Status
0x09B0:0x09B7	4/8	Latch0 Time Positive Edge
0x09B8:0x09BF	4/8	Latch0 Time Negative Edge
0x09C0:0x09C7	4/8	Latch1 Time Positive Edge
0x09C8:0x09CF	4/8	Latch1 Time Negative Edge

Address ¹	Length (Byte)	Description
DC – SyncManager Event Times		
0x09F0:0x09F3	4	EtherCAT Buffer Change Event Time
0x09F8:0x09FB	4	PDI Buffer Start Event Time
0x09FC:0x09FF	4	PDI Buffer Change Event Time
ESC specific		
0x0E00:0x0EFF	256	ESC specific registers (e.g., Power-On Values / Product and Vendor ID)
Digital Input/Output		
0xF00:0xF03	4	Digital I/O Output Data
0xF10:0xF17	1-8	General Purpose Outputs
0xF18:0xF1F	1-8	General Purpose Inputs
User RAM/Extended ESC features		
0xF80:0xFFFF	128	User RAM/Extended ESC features
Process Data RAM		
0x1000:0x1003	4	Digital I/O Input Data
0x1000:0xFFFF	0-60 KB	Process Data RAM
[0x1000:0x13FF]	1 KB	ET1200
[0x1000:0x1FFF]	4 KB	ESC20
[0x1000:0x2FFF]	8 KB	ET1100
[0x1000:0xFFFF]	1-60 KB	IP-Core

For Registers longer than one byte, the LSB has the lowest and MSB the highest address.

1.1 Scope of Section II

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 in Section III of a specific ESC to find out which registers are available. Additionally, refer to the feature details overview in Section III of a specific ESC to find out which features are available.

The following Beckhoff ESCs are covered by Section II:

- ET1200-000x
- ET1100-000x
- EtherCAT IP Core for Altera® FPGAs (V2.4.0)
- EtherCAT IP Core for Xilinx® FPGAs (V2.04a)
- ESC20 (Build 22)

1.2 Reserved Registers/Reserved Register Bits

Reserved registers must not be written, reserved register bits have to be written as 0. Read values of reserved registers or register bits have to be ignored (nevertheless, the typical value is 0). Reserved registers or register bits initialized by EEPROM values have to be initialized with 0.

Reserved EEPROM words of the ESC configuration area have to be 0.

1.3 ESC Availability Tab Legend

The availability of registers and exceptions for individual register bits or IP Core versions are indicated in a small area at the top right edge of each register table.

Example 1:

ESC20	ET1100	ET1200	IP Core
		[5]	V2.0.0/ V2.00a

- Register is not available for ESC20 (reserved)
- Register is available for ET1100 (all bits mentioned below)
- Register is available for ET1200, except for bit 5 which is reserved
- Register is available for IP Core since V2.0.0/V2.00a, reserved for previous versions

Example 2:

ESC20	ET1100	ET1200	IP Core
	write config.		[5] V2.0.0/ V2.00a

- Register is available for ET1100 (read), write access is optionally available (e.g. SII EEPROM or IP Core configuration)
- Register is available for IP Core, bit 5 is available since V2.0.0/V2.00a, bit 5 is not available for previous versions (and reserved)

Example 3:

ESC20	ET1100	ET1200	IP Core
	[63:16] config.		V2.0.0/ V2.00a

- Register is available for ET1100, bits [63:16] are optionally available (e.g. SII EEPROM or IP Core configuration)
- Register is optionally available/configurable for IP Core since V2.0.0/V2.00a (“IP Core” is not **bold**)

2 ESC Register Availability

Table 2: ESC Register Availability

Address	Length (Byte)	Description	ET1200	ET1100	IP Core V2.4.0/V2.04a Register set			ESC20
			S	M	L			
0x0000	1	Type	X	X	X	X	X	X
0x0001	1	Revision	X	X	X	X	X	X
0x0002:0x0003	2	Build	X	X	X	X	X	X
0x0004	1	FMMUs supported	X	X	X	X	X	X
0x0005	1	SyncManagers supported	X	X	X	X	X	X
0x0006	1	RAM Size	X	X	X	X	X	X
0x0007	1	Port Descriptor	X	X	X	X	X	-
0x0008:0x0009	2	ESC Features supported	X	X	X	X	X	X
0x0010:0x0011	2	Configured Station Address	X	X	X	X	X	X
0x0012:0x0013	2	Configured Station Alias	X	X	X	X	X	X
0x0020	1	Write Register Enable	X	X	C	C	X	X
0x0021	1	Write Register Protection	X	X	C	C	X	X
0x0030	1	ESC Write Enable	X	X	C	C	X	X
0x0031	1	ESC Write Protection	X	X	C	C	X	X
0x0040	1	ESC Reset ECAT	X	X	C	C	C	-
0x0041	1	ESC Reset PDI	-	-	C	C	C	-
0x0100:0x0101	2	ESC DL Control	X	X	X	X	X	X
0x0102:0x0103	2	Extended ESC DL Control	X	X	X	X	X	X
0x0108:0x0109	2	Physical Read/Write Offset	X	X	C	C	X	X
0x0110:0x0111	2	ESC DL Status	X	X	X	X	X	X
0x0120	5 bits [4:0]	AL Control	X	X	X	X	X	X
0x0120:0x0121	2	AL Control	X	X	X	X	X	-
0x0130	5 bits [4:0]	AL Status	X	X	X	X	X	X
0x0130:0x0131	2	AL Status	X	X	X	X	X	-
0x0134:0x0135	2	AL Status Code	X	X	C	X	X	X
0x0138	1	RUN LED Override	-	-	C	C	C	-
0x0139	1	ERR LED Override	-	-	C	C	C	-
0x0140	1	PDI Control	X	X	X	X	X	X
0x0141	1	ESC Configuration	X	X	X	X	X	X
0x0150:0x0153	4	PDI Configuration	X	X	X	X	X	X
0x0152:0x0153	2	Extended PDI Configuration	X	X	X	X	X	X
0x0200:0x0201	2	ECAT Event Mask	X	X	X	X	X	X
0x0204:0x0207	4	AL Event Mask	X	X	r/c	X	X	X
0x0210:0x0211	2	ECAT Event Request	X	X	X	X	X	X
0x0220:0x0223	4	AL Event Request	X	X	X	X	X	X
0x0300:0x0307	4x2	Rx Error Counter[3:0]	X	X	X	X	X	X
0x0308:0x030B	4x1	Forwarded Rx Error counter[3:0]	X	X	X	X	X	-
0x030C	1	ECAT Processing Unit Error Counter	-	X	C	C	X	-

Address	Length (Byte)	Description	ET1200	ET1100	IP Core V2.4.0/V2.04a Register set			ESC20
					S	M	L	
0x030D	1	PDI Error Counter	-	x	c	c	x	-
0x030E	1	PDI Error Code	-	-	c	c	x	-
0x0310:0x0313	4x1	Lost Link Counter[3:0]	x	x	c	x	x	x
0x0400:0x0401	2	Watchdog Divider	x	x	r/c	x	x	x
0x0410:0x0411	2	Watchdog Time PDI	x	x	c	x	x	x
0x0420:0x0421	2	Watchdog Time Process Data	x	x	x	x	x	x
0x0440:0x0441	2	Watchdog Status Process Data	x	x	x	x	x	x
0x0442	1	Watchdog Counter Process Data	x	x	c	c	x	-
0x0443	1	Watchdog Counter PDI	x	x	c	c	x	-
0x0500:0x050F	16	SII EEPROM Interface	x	x	x	x	x	x
0x0510:0x0515	6	MII Management Interface	x	x	c	c	c	x
0x0516:0x0517	2	MII Management Access State	-	-	c	c	c	-
0x0518:0x051B	4	PHY Port Status[3:0]	-	-	c	c	c	-
0x0600:0x06FC	16x13	FMMU[15:0]	3	8	0-8	0-8	0-8	4
0x0800:0x087F	16x8	SyncManager[15:0]	4	8	0-8	0-8	0-8	4
0x0900:0x090F	4x4	DC – Receive Times[3:0]	x	x	rt	rt	rt	x
0x0910:0x0917	8	DC – System Time	x	s/l	dc	dc	dc	x
0x0918:0x091F	8	DC – Receive Time EPU	x	s/l	dc	dc	dc	x
0x0920:0x0935	24	DC – Time Loop Control Unit	x	s/l	dc	dc	dc	x
0x0980	1	DC – Cyclic Unit Control	x	s	dc	dc	dc	x
0x0981:0x0983	3	DC – SYNC Out Unit	x	s	dc	dc	dc	x
0x0984	1	DC – Activation Status	-	-	dc	dc	dc	-
0x098E:0x09A7	26	DC – SYNC Out Unit	x	s	dc	dc	dc	x
0x09A8:0x09A9	36	DC – Latch In Unit	x	l	dc	dc	dc	x
0x09AE:0x09CF								
0x09F0:0x09F3	12	DC – SyncManager Event Times	-	s/l	c	c	dc	-
0x09F8:0x09FF								
0xE00:0x0EFF	256	ESC specific registers (e.g., Power-On Values / Product and Vendor ID)	x	x	x	x	x	x
0xF00:0xF03	4	Digital I/O Output Data	x	x	io	io	io	x
0xF10:0xF17	8	General Purpose Outputs [Byte]	2	2	0-8	0-8	0-8	-
0xF18:0xF1F	8	General Purpose Inputs [Byte]	-	2	0-8	0-8	0-8	-
0xF80:0xFFFF	128	User RAM	x	x	x	x	x	x
0x1000:0x1003	4	Digital I/O Input Data	io	io	io	io	io	io
0x1000 ff.		Process Data RAM [Kbyte]	1	8	1-60	1-60	1-60	4

Table 3: ESC Register Availability Legend

Symbol	Description
x	Available
-	Not available
r	Read only
c	Configurable
dc	Available if Distributed Clocks are enabled
rt	Available if Receive Times or Distributed Clocks are enabled (always available for 3-4 ports)
s	Available if DC SYNC Out Unit enabled (Register 0x0140.10=1)
I	Available if DC Latch In Unit enabled (Register 0x0140.11=1)
s/I	Available if DC SYNC Out Unit enabled and/or DC Latch In Unit enabled (Register 0x0140.10=1 and/or 0x0140.11=1)
io	Available if Digital I/O PDI is selected

3 Register description

3.1 Type (0x0000)

Table 4: Register Type (0x0000)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
7:0	Type of EtherCAT controller	r/-	r/-	ESC10, ESC20: 0x02 IP Core: 0x04 ET1100: 0x11 ET1200: 0x12 Other FPGA-based Beckhoff ESCs: First terminals : 0x01 First EK1100: 0x03 Current terminals: 0x05

3.2 Revision (0x0001)

Table 5: Register Revision (0x0001)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
7:0	Revision of EtherCAT controller. IP Core: major version X	r/-	r/-	ESC dep.

3.3 Build (0x0002:0x0003)

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

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
15:0	Actual build of EtherCAT controller. IP Core: [7:4] = minor version Y, [3:0] = maintenance version Z	r/-	r/-	ESC dep.

3.4 FMMUs supported (0x0004)

Table 7: Register FMMUs supported (0x0004)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
7:0	Number of supported FMMU channels (or entities) of the EtherCAT Slave Controller.	r/-	r/-	ESC20: 4 IP Core: depends on configuration ET1100: 8 ET1200: 3

3.5 SyncManagers supported (0x0005)

Table 8: Register SyncManagers supported (0x0005)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
7:0	Number of supported SyncManager channels (or entities) of the EtherCAT Slave Controller	r/-	r/-	ESC20: 4 IP Core: depends on configuration ET1100: 8 ET1200: 4

3.6 RAM Size (0x0006)

Table 9: Register RAM Size (0x0006)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
7:0	Process Data RAM size supported by the EtherCAT Slave Controller in KByte	r/-	r/-	ESC20: 4 IP Core: depends on configuration ET1100: 8 ET1200: 1

3.7 Port Descriptor (0x0007)

Table 10: Register Port Descriptor (0x0007)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
	Port configuration: 00: Not implemented 01: Not configured (SII EEPROM) 10: EBUS 11: MII / RMII			
1:0	Port 0	r/-	r/-	ESC and ESC configuration dep.
3:2	Port 1	r/-	r/-	
5:4	Port 2	r/-	r/-	
7:6	Port 3	r/-	r/-	

3.8 ESC Features supported (0x0008:0x0009)

Table 11: Register ESC Features supported (0x0008:0x0009)

Bit	Description	ECAT	PDI	Reset Value	
				ESC20	ET1100
0	FMMU Operation: 0: Bit oriented 1: Byte oriented	r/-	r/-	[8] V2.2.0/ V2.02a	0
1	Reserved	r/-	r/-		0
2	Distributed Clocks: 0: Not available 1: Available	r/-	r/-	ESC20: 1 IP Core: depends on configuration ET1100: 1 ET1200: 1	
3	Distributed Clocks (width): 0: 32 bit 1: 64 bit	r/-	r/-	ET1100: 1 ET1200: 1 IP Core: depends on configuration Others : 0	
4	Low Jitter EBUS: 0: Not available, standard jitter 1: Available, jitter minimized	r/-	r/-	ET1100: 1 ET1200: 1 Others : 0	
5	Enhanced Link Detection EBUS: 0: Not available 1: Available ET1100/ET1200: Enhanced Link Detection EBUS must not be activated.	r/-	r/-	ET1100: 1 ET1200: 1 Others : 0	
6	Enhanced Link Detection MII: 0: Not available 1: Available	r/-	r/-	ET1100: 1 ET1200: 1 Others : 0	
7	Separate Handling of FCS Errors: 0: Not supported 1: Supported, frames with wrong FCS and additional nibble will be counted separately in Forwarded RX Error Counter	r/-	r/-	IP Core: 1 ET1100: 1 ET1200: 1 Others : 0	
8	Enhanced DC SYNC Activation 0: Not available 1: Available NOTE: This feature refers to registers 0x981[7:3], 0x0984	r/-	r/-	IP Core: depends on version Others : 0	
9	EtherCAT LRW command support: 0: Supported 1: Not supported	r/-	r/-	0	
10	EtherCAT read/write command support (BRW, APRW, FPRW): 0: Supported 1: Not supported	r/-	r/-	0	
11	Fixed FMMU/SyncManager configuration 0: Variable configuration 1: Fixed configuration (refer to documentation of supporting ESCs)	r/-	r/-	0	

Bit	Description	ECAT	PDI	Reset Value
15:12	Reserved	r/-	r/-	0

3.9 Configured Station Address (0x0010:0x0011)

Table 12: Register Configured Station Address (0x0010:0x0011)

Bit	Description	ECAT	PDI	Reset Value
15:0	Address used for node addressing (FPxx commands)	r/w	r/-	0

3.10 Configured Station Alias (0x0012:0x0013)

Table 13: Register Configured Station Alias (0x0012:0x0013)

Bit	Description	ECAT	PDI	Reset Value
15:0	Alias Address used for node addressing (FPxx commands). The use of this alias is activated by Register DL Control Bit 24 (0x0100.24/0x0103.0) NOTE: EEPROM value is only taken over at first EEPROM load after power-on or reset. ESC20 exception: EEPROM value is taken over after each EEPROM reload command.	r/-	r/w	0 until first EEPROM load, then EEPROM ADR 0x0004

3.11 Write Register Enable (0x0020)

Table 14: Register Write Register Enable (0x0020)

Bit	Description	ECAT	PDI		Reset Value	
			ESC20	ET1100	ET1200	IP Core
0	If write register protection is enabled, this register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed. Write protection is still active after this frame (if Write Register Protection register is not changed).	r/w	r/-		0	read: V2.4.0/ V2.04a
7:1	Reserved, write 0	r/-	r/-		0	

3.12 Write Register Protection (0x0021)

Table 15: Register Write Register Protection (0x0021)

Bit	Description	ECAT	PDI	Reset Value
		ESC20	ET1100	ET1200
0	Write register protection: 0: Protection disabled 1: Protection enabled Registers 0x0000-0xF0F are write protected, except for 0x0030.	r/w	r/-	0
7:1	Reserved, write 0	r/-	r/-	0

3.13 ESC Write Enable (0x0030)

Table 16: Register ESC Write Enable (0x0030)

Bit	Description	ECAT	PDI	Reset Value
		ESC20	ET1100	ET1200
0	If ESC write protection is enabled, this register has to be written in the same Ethernet frame (value does not care) before other writes to this station are allowed. ESC write protection is still active after this frame (if ESC Write Protection register is not changed).	r/w	r/-	0
7:1	Reserved, write 0	r/-	r/-	0

3.14 ESC Write Protection (0x0031)

Table 17: Register ESC Write Protection (0x0031)

Bit	Description	ECAT	PDI	Reset Value
		ESC20	ET1100	ET1200
0	Write protect: 0: Protection disabled 1: Protection enabled All areas are write protected, except for 0x0030.	r/w	r/-	0
7:1	Reserved, write 0	r/-	r/-	0

3.15 ESC Reset ECAT (0x0040)

Table 18: Register ESC Reset ECAT (0x0040)

Bit	Description	ECAT	PDI	Reset Value
Write				
7:0	A reset is asserted after writing 0x52 ('R'), 0x45 ('E') and 0x53 ('S') in this register with 3 consecutive frames.	r/w	r/-	0
Read				
1:0	Progress of the reset procedure: 01: after writing 0x52 10: after writing 0x45 (if 0x52 was written before) 00: else	r/w	r/-	00
7:2	Reserved, write 0	r/-	r/-	0

3.16 ESC Reset PDI (0x0041)

Table 19: Register ESC Reset PDI (0x0041)

Bit	Description	ECAT	PDI	Reset Value
Write				
7:0	A reset is asserted after writing 0x52 ('R'), 0x45 ('E') and 0x53 ('S') in this register with 3 consecutive commands.	r/-	r/w	0
Read				
1:0	Progress of the reset procedure: 01: after writing 0x52 10: after writing 0x45 (if 0x52 was written before) 00: else	r/-	r/w	00
7:2	Reserved, write 0	r/-	r/-	0

3.17 ESC DL Control (0x0100:0x0103)

Table 20: Register ESC DL Control (0x0100:0x0103)

Bit	Description	ECAT	PDI		Reset Value
			ESC20	ET1100	ET1200
0	<p>Forwarding rule: 0: EtherCAT frames are processed, Non-EtherCAT frames are forwarded without processing 1: EtherCAT frames are processed, Non- EtherCAT frames are destroyed The source MAC address is changed for every frame (SOURCE_MAC[1] is set to 1 – locally administered address) regardless of the forwarding rule.</p>	r/w	r/-		1
1	<p>Temporary use of settings in Register 0x101: 0: permanent use 1: use for about 1 second, then revert to previous settings</p>	r/w	r/-	0	
7:2	Reserved, write 0	r/-	r/-	0	
9:8	<p>Loop Port 0: 00: Auto 01: Auto Close 10: Open 11: Closed</p> <p>NOTE: Loop open means sending/receiving over this port is enabled, loop closed means sending/receiving is disabled and frames are forwarded to the next open port internally. Auto: loop closed at link down, opened at link up Auto Close: loop closed at link down, opened with writing 01 again after link up (or receiving a valid Ethernet frame at the closed port) Open: loop open regardless of link state Closed: loop closed regardless of link state</p>	r/w*	r/-	00	
11:10	<p>Loop Port 1: 00: Auto 01: Auto Close 10: Open 11: Closed</p>	r/w*	r/-	00	
13:12	<p>Loop Port 2: 00: Auto 01: Auto Close 10: Open 11: Closed</p>	r/w*	r/-	00	
15:14	<p>Loop Port 3: 00: Auto 01: Auto Close 10: Open 11: Closed</p>	r/w*	r/-	ET1200: 11 others: 00	

Bit	Description	ECAT	PDI	Reset Value
18:16	RX FIFO Size (ESC delays start of forwarding until FIFO is at least half full). RX FIFO Size/RX delay reduction** : Value: EBUS: MII: 0: -50 ns -40 ns 1: -40 ns -40 ns 2: -30 ns -40 ns 3: -20 ns -40 ns 4: -10 ns no change 5: no change no change 6: no change no change 7: default default	r/w	r/-	7
19	EBUS Low Jitter: 0: Normal jitter 1: Reduced jitter	r/w	r/-	0
23:20	Reserved, write 0	r/-	r/-	0
24	Station alias: 0: Ignore Station Alias 1: Alias can be used for all configured address command types (FPRD, FPWR, ...)	r/w	r/-	0
31:25	Reserved, write 0	r/-	r/-	0

* Loop configuration changes are delayed until the end of a currently received or transmitted frame at the port.

** The possibility of RX FIFO Size reduction depends on the clock source accuracy of the ESC and of every connected EtherCAT/Ethernet devices (master, slave, etc.). RX FIFO Size of 7 is sufficient for 100ppm accuracy, FIFO Size 0 is possible with 25ppm accuracy (frame size of 1518/1522 Byte).

3.18 Physical Read/Write Offset (0x0108:0x0109)

Table 21: Register Physical Read/Write Offset (0x0108:0x0109)

Bit	Description	ECAT	ESC20	ET1100	ET1200	IP Core
			PDI	Reset Value		
15:0	Offset of R/W Commands (FPRW, APRW) between Read address and Write address. RD_ADR = ADR and WR_ADR = ADR + R/W-Offset	r/w	r/-	0		

3.19 ESC DL Status (0x0110:0x0111)

Table 22: Register ESC DL Status (0x0110:0x0111)

Bit	Description	ECAT	PDI	Reset Value
0	PDI operational/EEPROM loaded correctly: 0: EEPROM not loaded, PDI not operational (no access to Process Data RAM) 1: EEPROM loaded correctly, PDI operational (access to Process Data RAM)	r(ack)/-	r/-	0
1	PDI Watchdog Status: 0: Watchdog expired 1: Watchdog reloaded	r(ack)/-	r/-	0
2	Enhanced Link detection: 0: Deactivated for all ports 1: Activated for at least one port NOTE: EEPROM value is only taken over at first EEPROM load after power-on or reset	r(ack)/-	r/-	ET1100/ET1200: 1 until first EEPROM load, then EEPROM ADR 0x0000.9 IP Core with feature: 1 until first EEPROM load, then EEPROM ADR 0x0000.9 or 0x0000[15:12] Others: 0
3	Reserved	r(ack)/-	r/-	0
4	Physical link on Port 0: 0: No link 1: Link detected	r(ack)/-	r/-	0
5	Physical link on Port 1: 0: No link 1: Link detected	r(ack)/-	r/-	0
6	Physical link on Port 2: 0: No link 1: Link detected	r(ack)/-	r/-	0
7	Physical link on Port 3: 0: No link 1: Link detected	r(ack)/-	r/-	0
8	Loop Port 0: 0: Open 1: Closed	r(ack)/-	r/-	0
9	Communication on Port 0: 0: No stable communication 1: Communication established	r(ack)/-	r/-	0
10	Loop Port 1: 0: Open 1: Closed	r(ack)/-	r/-	0
11	Communication on Port 1: 0: No stable communication 1: Communication established	r(ack)/-	r/-	0

Bit	Description	ECAT	PDI	Reset Value
12	Loop Port 2: 0: Open 1: Closed	r(ack)/-	r/-	0
13	Communication on Port 2: 0: No stable communication 1: Communication established	r(ack)/-	r/-	0
14	Loop Port 3: 0: Open 1: Closed	r(ack)/-	r/-	0
15	Communication on Port 3: 0: No stable communication 1: Communication established	r(ack)/-	r/-	0

NOTE: Reading DL Status register from ECAT clears ECAT Event Request 0x0210[2].

Table 23: Decoding port state in ESC DL Status register 0x0111 (typical modes only)

Register 0x0111	Port 3	Port 2	Port 1	Port 0
0x55	No link, closed	No link, closed	No link, closed	No link, closed
0x56	No link, closed	No link, closed	No link, closed	Link, open
0x59	No link, closed	No link, closed	Link, open	No link, closed
0x5A	No link, closed	No link, closed	Link, open	Link, open
0x65	No link, closed	Link, open	No link, closed	No link, closed
0x66	No link, closed	Link, open	No link, closed	Link, open
0x69	No link, closed	Link, open	Link, open	No link, closed
0x6A	No link, closed	Link, open	Link, open	Link, open
0x95	Link, open	No link, closed	No link, closed	No link, closed
0x96	Link, open	No link, closed	No link, closed	Link, open
0x99	Link, open	No link, closed	Link, open	No link, closed
0x9A	Link, open	No link, closed	Link, open	Link, open
0xA5	Link, open	Link, open	No link, closed	No link, closed
0xA6	Link, open	Link, open	No link, closed	Link, open
0xA9	Link, open	Link, open	Link, open	No link, closed
0xAA	Link, open	Link, open	Link, open	Link, open
0xD5	Link, closed	No link, closed	No link, closed	No link, closed
0xD6	Link, closed	No link, closed	No link, closed	Link, open
0xD9	Link, closed	No link, closed	Link, open	No link, closed
0xDA	Link, closed	No link, closed	Link, open	Link, open

3.20 AL Control (0x0120:0x0121)

Table 24: Register AL Control (0x0120:0x0121)

Bit	Description	ECAT	PDI	Reset Value			
				ESC20 [15:5]	ET1100	ET1200	IP Core [15:5] V2.4.0/ V2.04a
3:0	Initiate State Transition of the Device State Machine: 1: Request Init State 3: Request Bootstrap State 2: Request Pre-Operational State 4: Request Safe-Operational State 8: Request Operational State	r/(w)	r/(clear)-	1			
4	Error Ind Ack: 0: No Ack of Error Ind in AL status register 1: Ack of Error Ind in AL status register	r/(w)	r/(clear)-	0			
15:5	Reserved, write 0	r(w)	r/(clear)-	0			

NOTE: AL Control register behaves like a mailbox if Device Emulation is off (0x0140.8=0): The PDI has to read the AL Control register after ECAT has written it. Otherwise ECAT can not write again to the AL Control register. After Reset, AL Control register can be written by ECAT. (Regarding mailbox functionality, both registers 0x0120 and 0x0121 are equivalent, e.g. reading 0x0121 is sufficient to make this register writeable again.) If Device Emulation is on, the AL Control register can always be written, its content is copied to the AL Status register. Reading AL Control from PDI clears AL Event Request 0x0220[0].

3.21 AL Status (0x0130:0x0131)

Table 25: Register AL Status (0x0130:0x0131)

Bit	Description	ECAT	PDI	Reset Value			
				ESC20 [15:5]	ET1100	ET1200	IP Core [15:5] V2.4.0/ V2.04a
3:0	Actual State of the Device State Machine: 1: Init State 3: Request Bootstrap State 2: Pre-Operational State 4: Safe-Operational State 8: Operational State	r(ack)/-	r/(w)	1			
4	Error Ind: 0: Device is in State as requested or Flag cleared by command 1: Device has not entered requested State or changed State as result of a local action	r(ack)/-	r/(w)	0			
15:5	Reserved, write 0	r(ack)/-	r/(w)	0			

NOTE: AL Status register is only writable if Device Emulation is off (0x0140.8=0), otherwise AL Status register will reflect AL Control register values. Reading AL Status from ECAT clears ECAT Event Request 0x0210[3].

3.22 AL Status Code (0x0134:0x0135)

Table 26: Register AL Status Code (0x0134:0x0135)

Bit	Description	ECAT	PDI	Reset Value
15:0	AL Status Code	r/-	r/w	0

3.23 RUN LED Override (0x0138)

Table 27: Register RUN LED Override (0x0138)

Bit	Description	ECAT	PDI	Reset Value	
3:0	LED code: 0x0: Off 0x1-0xC: Flash 1x – 12x 0xD: Blinking 0xE: Flickering 0xF: On	(FSM State: (1-Init) (4-SafeOp 1x) (2-PreOp) (3-Bootstrap) (8-Op))	r/w	r/w	0
4	Enable Override: 0: Override disabled 1: Override enabled		r/w	0	
7:5	Reserved, write 0	r/w	r/w	0	

NOTE: Changes to AL Status register (0x0130) with valid values will disable RUN LED Override (0x0138[4]=0). The value read in this register always reflects current LED output.

3.24 ERR LED Override (0x0139)

Table 28: Register ERR LED Override (0x0139)

Bit	Description	ECAT	PDI	Reset Value
3:0	LED code: 0x0: Off 0x1-0xC: Flash 1x – 12x 0xD: Blinking 0xE: Flickering 0xF: On	r/w	r/w	0
4	Enable Override: 0: Override disabled 1: Override enabled	r/w	r/w	0
7:5	Reserved, write 0	r/w	r/w	0

NOTE: New error conditions will disable ERR LED Override (0x0139[4]=0). The value read in this register always reflects current LED output.

3.25 PDI Control (0x0140)

Table 29: Register PDI Control (0x0140)

Bit	Description	ECAT	PDI	Reset Value
7:0	Process data interface: 0x00: Interface deactivated (no PDI) 0x01: 4 Digital Input 0x02: 4 Digital Output 0x03: 2 Digital Input and 2 Digital Output 0x04: Digital I/O 0x05: SPI Slave 0x06: Oversampling I/O 0x07: EtherCAT Bridge (port 3) 0x08: 16 Bit asynchronous Microcontroller interface 0x09: 8 Bit asynchronous Microcontroller interface 0x0A: 16 Bit synchronous Microcontroller interface 0x0B: 8 Bit synchronous Microcontroller interface 0x0C: 16 Bit multiplexed asynchronous Microcontroller interface 0x0D: 8 Bit multiplexed asynchronous Microcontroller interface 0x10: 32 Digital Input and 0 Digital Output 0x11: 24 Digital Input and 8 Digital Output 0x12: 16 Digital Input and 16 Digital Output 0x13: 8 Digital Input and 24 Digital Output 0x14: 0 Digital Input and 32 Digital Output 0x80: On-chip bus Others: Reserved	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0000

3.26 ESC Configuration (0x0141)

Table 30: Register ESC Configuration (0x0141)

Bit	Description	ECAT	PDI	Reset Value
		ESC20 [7:1]	ET1100 [7:4]	ET1200 [7:2]
0	Device emulation (control of AL status): 0: AL status register has to be set by PDI 1: AL status register will be set to value written to AL control register	r/-	r/-	IP Core: 1 with Digital I/O PDI, PDI_EMULATION pin with µC/On-chip bus Others: 0, later EEPROM ADR 0x0000
1	Enhanced Link detection all ports: 0: disabled (if bits [15:12]=0) 1: enabled at all ports ET1100/ET1200: Enhanced Link Detection EBUS must not be activated if EBUS ports are used.	r/-	r/-	0, later EEPROM ADR 0x0000
2	Distributed Clocks SYNC Out Unit: 0: disabled (power saving) 1: enabled	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0000
3	Distributed Clocks Latch In Unit: 0: disabled (power saving) 1: enabled	r/-	r/-	
4	Enhanced Link port 0: 0: disabled (if bit 9=0) 1: enabled	r/-	r/-	0, later EEPROM ADR 0x0000
5	Enhanced Link port 1: 0: disabled (if bit 9=0) 1: enabled	r/-	r/-	
6	Enhanced Link port 2: 0: disabled (if bit 9=0) 1: enabled	r/-	r/-	
7	Enhanced Link port 3: 0: disabled (if bit 9=0) 1: enabled	r/-	r/-	

3.27 PDI Configuration (0x0150:0x0153)

The PDI configuration register 0x0150 and the extended PDI configuration registers 0x0152:0x0153 depend on the selected PDI. The Sync/Latch PDI configuration register 0x0151 is independent of the selected PDI.

Table 31: PDI Configuration Register overview

PDI number	PDI name	Configuration registers	
0x04	Digital I/O	0x0150	0x0152:0x0153
0x05	SPI Slave	0x0150	0x0152:0x0153
0x08/0x09	8/16Bit asynchronous Microcontroller	0x0150	0x0152:0x0153
0x0A/0x0B	8/16Bit synchronous Microcontroller	0x0150	0x0152:0x0153
0x07	EtherCAT Bridge (port 3)	0x0150	0x0152:0x0153
0x80	On-chip bus	0x0150	0x0152:0x0153
Sync/Latch PDI Configuration			
-	Sync/Latch PDI Configuration	0x0151	

3.27.1 Digital I/O configuration

Table 32: Register Digital I/O configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value
0	OUTVALID polarity: 0: Active high 1: Active low	r/-	r/-	IP Core: 0 Others: 0, later EEPROM ADR 0x0001
1	OUTVALID mode: 0: Output event signaling 1: Process Data Watchdog trigger (WD_TRIG) signaling on OUTVALID pin (see SyncManager). Output data is updated if watchdog is triggered. Overrides 0x0150[7:6]	r/-	r/-	
2	Unidirectional/Bidirectional mode*: 0: Unidirectional mode: input/output direction of pins configured individually 1: Bidirectional mode: all I/O pins are bidirectional, direction configuration is ignored	r/-	r/-	IP Core: 1 Others: 0, later EEPROM ADR 0x0001
3	Watchdog behavior: 0: Outputs are reset immediately after watchdog expires 1: Outputs are reset with next output event that follows watchdog expiration	r/-	r/-	IP Core: 0 Others: 0, later EEPROM ADR 0x0001
5:4	Input DATA is sampled at 00: Start of Frame ² 01: Rising edge of LATCH_IN 10: DC SYNC0 event ² 11: DC SYNC1 event ²	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001
7:6	Output DATA is updated at 00: End of Frame 01: Reserved 10: DC SYNC0 event 11: DC SYNC1 event If 0x0150[1]=1, output DATA is updated at Process Data Watchdog trigger event (0x0150[7:6] are ignored)	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001

* IP Core: I/O direction depends on configuration, bidirectional mode is not supported.

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

² ET1200: LATCH_IN/SOF reflects Start of Frame (SOF) if input data is sampled with SOF or DC SYNC events.

Table 33: Register Digital I/O extended configuration (0x0152:0x0153)

		ESC20	ET1100	ET1200	IP Core [15:8]
Bit	Description	ECAT	PDI	Reset Value	
	Digital I/Os are configured in pairs as inputs or outputs: 0: Input 1: Output NOTE: Reserved in bidirectional mode, set to 0. Configuration bits for unavailable I/Os are reserved, set to 0.				
0	Direction of I/O[1:0]	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0003	
1	Direction of I/O[3:2]				
2	Direction of I/O[5:4]				
3	Direction of I/O[7:6]				
4	Direction of I/O[9:8]				
5	Direction of I/O[11:10]				
6	Direction of I/O[13:12]				
7	Direction of I/O[15:14]				
8	Direction of I/O[17:16]				
9	Direction of I/O[19:18]				
10	Direction of I/O[21:20]				
11	Direction of I/O[23:22]				
12	Direction of I/O[25:24]				
13	Direction of I/O[27:26]				
14	Direction of I/O[29:26]				
15	Direction of I/O[31:30]				

3.27.2 SPI Slave Configuration

Table 34: Register SPI Configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value
		ESC20 3:2, 7:6	ET1100	ET1200
1:0	SPI mode: 00: SPI mode 0 01: SPI mode 1 10: SPI mode 2 11: SPI mode 3 NOTE: SPI mode 3 is recommended for Slave Sample Code NOTE: SPI status flag is not available in SPI modes 0 and 2 with normal data out sample.	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001
3:2	SPI_IRQ output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high)	r/-	r/-	
4	SPI_SEL polarity: 0: Active low 1: Active high	r/-	r/-	
5	Data Out sample mode: 0: Normal sample (SPI_DO and SPI_DI are sampled at the same SPI_CLK edge) 1: Late sample (SPI_DO and SPI_DI are sampled at different SPI_CLK edges) NOTE: Normal Data Out sample mode is recommended for Slave Sample Code	r/-	r/-	
7:6	Reserved, set EEPROM value 0	r/-	r/-	

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

Table 35: Register SPI extended configuration (0x0152:0x0153)

Bit	Description	ECAT	PDI	Reset Value
		ESC20	ET1100	ET1200
15:0	Reserved, set EEPROM value 0	r/-	r/-	IP Core: 0 Others: 0, later EEPROM ADR 0x0003

3.27.3 8/16Bit asynchronous Microcontroller configuration

Table 36: Register asynchronous Microcontroller Configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value			
				ESC20	ET1100	ET1200	IP Core
1:0	BUSY output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high) NOTE: Push-Pull: no CS → not BUSY (driven) Open Drain/Source: no CS → BUSY open	r/-	r/-				IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001
3:2	IRQ output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high)	r/-	r/-				
4	BHE polarity: 0: Active low 1: Active high	r/-	r/-				IP Core: 0 Others: 0, later EEPROM ADR 0x0001
5	Reserved, set EEPROM value 0	r/-	r/-				
6	Reserved, set EEPROM value 0	r/-	r/-				
7	RD Polarity: 0: Active low 1: Active high	r/-	r/-				

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

Table 37: Register Asynchronous Microcontroller extended Configuration (0x0152:0x0153)

Bit	Description	ECAT	PDI	Reset Value			
				ESC20	ET1100	ET1200	IP Core
0	Read BUSY delay: 0: Normal read BUSY output 1: Delayed read BUSY output	r/-	r/-				V2.2.0/ V2.02a [1] V2.3.0/ V2.03a
1	Perform internal write at: 0: End of write access 1: Beginning of write access	r/-	r/-				
15:2	Reserved, set EEPROM value 0	r/-	r/-				

3.27.4 8/16Bit synchronous Microcontroller configuration

Table 38: Register Synchronous Microcontroller Configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value
1:0	TA output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high) NOTE: Push-Pull: no CS → no TA (driven) Open Drain/Source: no CS → TA open	r/-	r/-	0, later EEPROM ADR 0x0001
3:2	IRQ output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high)	r/-	r/-	
4	BHE polarity: 0: Active low 1: Active high	r/-	r/-	
5	ADR(0) polarity: 0: Active high 1: Active low	r/-	r/-	
6	Byte access mode: 0: BHE or Byte Select mode 1: Transfer Size mode	r/-	r/-	
7	TS Polarity: 0: Active low 1: Active high	r/-	r/-	

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

Table 39: Register Synchronous Microcontroller extended Configuration (0x0152:0x0153)

Bit	Description	ECAT	PDI	Reset Value
7:0	Reserved, set EEPROM value 0	r/-	r/-	0, later EEPROM ADR 0x0003
8	Write data valid: 0: Write data valid one clock cycle after CS 1: Write data valid together with CS	r/-	r/-	
9	Read mode: 0: Use Byte Selects for read accesses 1: Ignore Byte Selects for read accesses, always read 16 bit	r/-	r/-	
10	CS mode: 0: Sample CS with rising edge of CPU_CLK 1: Sample CS with falling edge of CPU_CLK	r/-	r/-	
11	TA/IRQ mode: 0: Update TA/IRQ with rising edge of CPU_CLK 1: Update TA/IRQ with falling edge of CPU_CLK	r/-	r/-	
15:12	Reserved, set EEPROM value 0	r/-	r/-	

3.27.5 EtherCAT Bridge (port 3)

Table 40: Register EtherCAT Bridge configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value
0	Bridge port physical layer: 0: EBUS 1: MII	r/-	r/-	0, later EEPROM ADR 0x0001
7:1	Reserved, set EEPROM value 0	r/-	r/-	

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

Table 41: Register EtherCAT Bridge extended configuration (0x0152:0x0153)

Bit	Description	ECAT	PDI	Reset Value
15:0	Reserved, set EEPROM value 0	r/-	r/-	0, later EEPROM ADR 0x0003

3.27.6 On-chip bus configuration

Table 42: Register On-chip bus configuration (0x0150)

Bit	Description	ECAT	PDI	Reset Value
4:0	On-chip bus clock: 0: asynchronous 1-31: synchronous multiplication factor (N * 25 MHz)	r/-	r/-	IP Core: Depends on configuration
7:5	On-chip bus: 000: Altera® Avalon® 010: Xilinx® PLB v4.6 100: Xilinx OPB others: reserved	r/-	r/-	

Table Register Sync/Latch PDI Configuration (0x0151) moved to chapter 3.27.7

Table 43: Register On-chip bus extended configuration (0x0152:0x0153)

Bit	Description	ECAT	PDI	Reset Value
1:0	Data Bus Width W 0: 4 Byte 1: 1 Byte 2: 2 Byte 3: Reserved	r/-	r/-	IP Core: Depends on configuration
15:2	Reserved	r/-	r/-	0

3.27.7 Sync/Latch PDI Configuration

Table 44: Register Sync/Latch PDI Configuration (0x0151)

Bit	Description	ECAT	PDI	Reset Value
1:0	SYNC0 output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high)	r/-	r/-	IP Core: 10 Others: 00, later EEPROM ADR 0x0001
2	SYNC0/LATCH0 configuration*: 0: LATCH0 Input 1: SYNC0 Output	r/-	r/-	IP Core: 1 Others: 0, later EEPROM ADR 0x0001
3	SYNC0 mapped to AL Event Request register 0x0220.2: 0: Disabled 1: Enabled	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001
5:4	SYNC1 output driver/polarity: 00: Push-Pull active low 01: Open Drain (active low) 10: Push-Pull active high 11: Open Source (active high)	r/-	r/-	IP Core: 10 Others: 00, later EEPROM ADR 0x0001
6	SYNC1/LATCH1 configuration*: 0: LATCH1 input 1: SYNC1 output	r/-	r/-	IP Core: 1 Others: 0, later EEPROM ADR 0x0001
7	SYNC1 mapped to AL Event Request register 0x0220.3: 0: Disabled 1: Enabled	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0001

* The IP Core has concurrent SYNC0/SYNC1 outputs and LATCH0/LATCH1 inputs, independent of this configuration.

3.28 ECAT Event Mask (0x0200:0x0201)

Table 45: Register ECAT Event Mask (0x0200:0x0201)

Bit	Description	ECAT	PDI	Reset Value	
		ESC20	ET1100	ET1200	IP Core write config.
15:0	ECAT Event masking of the ECAT Event Request Events for mapping into ECAT event field of EtherCAT frames: 0: Corresponding ECAT Event Request register bit is not mapped 1: Corresponding ECAT Event Request register bit is mapped	r/w	r/-	0	

3.29 AL Event Mask (0x0204:0x0207)

Table 46: Register AL Event Mask (0x0204:0x0207)

Bit	Description	ECAT	PDI	Reset Value	
		ESC20	ET1100	ET1200	IP Core write config.
31:0	AL Event masking of the AL Event Request register Events for mapping to PDI IRQ signal: 0: Corresponding AL Event Request register bit is not mapped 1: Corresponding AL Event Request register bit is mapped	r/-	r/w	0x00FF:0xFF0F	

3.30 ECAT Event Request (0x0210:0x0211)

Table 47: Register ECAT Event Request (0x0210:0x0211)

Bit	Description	ECAT	PDI	Reset Value
0	DC Latch event: 0: No change on DC Latch Inputs 1: At least one change on DC Latch Inputs (Bit is cleared by reading DC Latch event times from ECAT for ECAT controlled Latch Units, so that Latch 0/1 Status 0x09AE:0x09AF indicates no event)	r/-	r/-	0
1	Reserved	r/-	r/-	0
2	DL Status event: 0: No change in DL Status 1: DL Status change (Bit is cleared by reading out DL Status 0x0110:0x0111 from ECAT)	r/-	r/-	0
3	AL Status event: 0: No change in AL Status 1: AL Status change (Bit is cleared by reading out AL Status 0x0130:0x0131 from ECAT)	r/-	r/-	0
4	Mirrors values of each SyncManager Status: 0: No Sync Channel 0 event 1: Sync Channel 0 event pending	r/-	r/-	0
5	0: No Sync Channel 1 event 1: Sync Channel 1 event pending			
...	...			
11	0: No Sync Channel 7 event 1: Sync Channel 7 event pending			
15:12	Reserved	r/-	r/-	0

3.31 AL Event Request (0x0220:0x0223)

Table 48: Register AL Event Request (0x0220:0x0223)

Bit	Description	ECAT	PDI	Reset Value
		ESC20 [6:4]	ET1100 [6:5]	IP Core [5:4] V2.0.0/ V2.00a; [6] V2.3.0/ V2.03a
0	AL Control event: 0: No AL Control Register change 1: AL Control Register has been written ³ (Bit is cleared by reading AL Control register 0x0120:0x0121 from PDI)	r/-	r/-	0
1	DC Latch event: 0: No change on DC Latch Inputs 1: At least one change on DC Latch Inputs (Bit is cleared by reading DC Latch event times from PDI for PDI controlled Latch Units, so that Latch 0/1 Status 0x09AE:0x09AF indicates no event)	r/-	r/-	0
2	State of DC SYNC0 (if register 0x0151.3=1): (Bit is cleared by reading SYNC0 status 0x098E from PDI)	r/-	r/-	0
3	State of DC SYNC1 (if register 0x0151.7=1): (Bit is cleared by reading of SYNC1 status 0x098F from PDI)	r/-	r/-	0
4	SyncManager activation register (SyncManager register offset 0x6) changed: 0: No change in any SyncManager 1: At least one SyncManager changed (Bit is cleared by reading SyncManager Activation registers 0x0806 etc. from PDI)	r/-	r/-	0
5	EEPROM Emulation: 0: No command pending 1: EEPROM command pending (Bit is cleared by acknowledging the command in EEPROM command register 0x0502 from PDI)	r/-	r/-	0
6	Watchdog Process Data: 0: Has not expired 1: Has expired (Bit is cleared by reading Watchdog Status Process Data 0x0440 from PDI)	r/-	r/-	0
7	Reserved	r/-	r/-	0

³ AL control event is only generated if PDI emulation is turned off (PDI Control register 0x0140.8=0)

Bit	Description	ECAT	PDI	Reset Value
8	SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]): 0: No SyncManager 0 interrupt 1: SyncManager 0 interrupt pending	r/-	r/-	0
9	0: No SyncManager 1 interrupt 1: SyncManager 1 interrupt pending			
....			
23	0: No SyncManager 15 interrupt 1: SyncManager 15 interrupt pending			
31:24	Reserved	r/-	r/-	0

3.32 RX Error Counter (0x0300:0x0307)

Errors are only counted if the corresponding port is enabled.

Table 49: Register RX Error Counter Port y (0x0300+y*2:0x0301+y*2)

Bit	Description	ECAT	PDI	Reset Value
7:0	Invalid frame counter of Port y (counting is stopped when 0xFF is reached). Cleared if one of the RX Error counters 0x0300-0x030B is written.	r/ w(clr)	r/-	0
15:8	RX Error counter of Port y (counting is stopped when 0xFF is reached). This is coupled directly to RX ERR of MII interface/EBUS interface. Cleared if one of the RX Error counters 0x0300-0x030B is written.	r/ w(clr)	r/-	0

The invalid frame counters are incremented if there is an error in the frame format (Preamble, SFD – Start of Frame Delimiter, FCS – Checksum, invalid length). If the FCS is invalid and an additional nibble is appended, the FCS error is not counted. This is why EtherCAT forwards frames with errors with an invalid FCS and an additional nibble.

RX Errors may appear either inside or outside frames. RX Errors inside frames will lead to invalid frames.

3.33 Forwarded RX Error Counter (0x0308:0x030B)

Table 50: Register Forwarded RX Error Counter Port y (0x0308+y)

Bit	Description	ECAT	PDI	Reset Value
7:0	Forwarded error counter of Port y (counting is stopped when 0xFF is reached). Cleared if one of the RX Error counters 0x0300-0x030B is written.	r/ w(clr)	r/-	0

3.34 ECAT Processing Unit Error Counter (0x030C)

Table 51: Register ECAT Processing Unit Error Counter (0x030C)

Bit	Description	ECAT	PDI	Reset Value
7:0	ECAT Processing Unit error counter (counting is stopped when 0xFF is reached). Counts errors of frames passing the Processing Unit (e.g., FCS is wrong or datagram structure is wrong). Cleared if register is written.	r/ w(clr)	r/-	0

3.35 PDI Error Counter (0x030D)

Table 52: Register PDI Error Counter (0x030D)

Bit	Description	ECAT	PDI	Reset Value
7:0	PDI Error counter (counting is stopped when 0xFF is reached). Counts if a PDI access has an interface error. Cleared if register is written.	r/ w(clr)	r/-	0

3.36 PDI Error Code (0x030E)

3.36.1 SPI PDI Error Code

Table 53: Register SPI PDI Error Code (0x030E)

Bit	Description	ECAT	PDI	Reset Value
	SPI access which caused last PDI Error. Cleared if register 0x030D is written.	r/-	r/-	0
2:0	Number of SPI clock cycles of whole access (modulo 8)			
3	Busy violation for first read data byte			
4	Read termination missing			
5	Access continued after read termination byte			
7:6	SPI command CMD[2:1]			

3.36.2 Asynchronous/Synchronous Microcontroller PDI Error Code

Table 54: Register Microcontroller PDI Error Code (0x030E)

Bit	Description	ECAT	PDI	Reset Value
	μ C access which caused last PDI Error. Cleared if register 0x030D is written.	r/-	r/-	0
0	Busy violation during read access			
1	Busy violation during write access			
2	Addressing error for a read access (A[0]=1 and BHE(act. low)=0)			
3	Addressing error for a write access (A[0]=1 and BHE(act. low)=0)			
7:4	reserved			

3.37 Lost Link Counter (0x0310:0x0313)

Table 55: Register Lost Link Counter Port y (0x0310+y)

Bit	Description	ECAT	PDI	Reset Value
7:0	Lost Link counter of Port y (counting is stopped when 0xff is reached). Counts only if port loop is Auto or Auto-Close. Cleared if one of the Lost Link counter registers is written.	r/ w(clr)	r/-	0

NOTE: Only lost links at open ports are counted.

3.38 Watchdog Divider (0x0400:0x0401)

Table 56: Register Watchdog Divider (0x0400:0x0401)

Bit	Description	ECAT	PDI	Reset Value	
				ESC20	ET1100
15:0	Watchdog divider: Number of 25 MHz tics (minus 2) that represents the basic watchdog increment. (Default value is 100µs = 2498)	r/w	r/-	0x09C2	write config.

3.39 Watchdog Time PDI (0x0410:0x0411)

Table 57: Register Watchdog Time PDI (0x0410:0x0411)

Bit	Description	ECAT	PDI	Reset Value	
				ESC20	ET1100
15:0	Watchdog Time PDI: number of basic watchdog increments (Default value with Watchdog divider 100µs means 100ms Watchdog)	r/w	r/-	0x03E8	0x03E8

Watchdog is disabled if Watchdog time is set to 0x0000. Watchdog is restarted with every PDI access.

3.40 Watchdog Time Process Data (0x0420:0x0421)

Table 58: Register Watchdog Time Process Data (0x0420:0x0421)

Bit	Description	ECAT	PDI	Reset Value	
				ESC20	ET1100
15:0	Watchdog Time Process Data: number of basic watchdog increments (Default value with Watchdog divider 100µs means 100ms Watchdog)	r/w	r/-	0x03E8	0x03E8

There is one Watchdog for all SyncManagers. Watchdog is disabled if Watchdog time is set to 0x0000. Watchdog is restarted with every write access to SyncManagers with Watchdog Trigger Enable Bit set.

Watchdog Status PDI

The Watchdog Status for the PDI can be read in the DL Status register 0x0110.1.

3.41 Watchdog Status Process Data (0x0440:0x0441)

Table 59: Register Watchdog Status Process Data (0x0440:0x0441)

Bit	Description	ECAT	PDI	Reset Value
0	Watchdog Status of Process Data (triggered by SyncManagers) 0: Watchdog Process Data expired 1: Watchdog Process Data is active or disabled	r/-	r(ack)/-	0
15:1	Reserved	r/-	r/-	0

NOTE: Reading this register clears AL Event Request 0x0220[6].

3.42 Watchdog Counter Process Data (0x0442)

Table 60: Register Watchdog Counter Process Data (0x0442)

Bit	Description	ECAT	PDI	Reset Value
7:0	Watchdog Counter Process Data (counting is stopped when 0xFF is reached). Counts if Process Data Watchdog expires. Cleared if one of the Watchdog counters 0x0442:0x0443 is written.	r/ w(clr)	r/-	0

3.43 Watchdog Counter PDI (0x0443)

Table 61: Register Watchdog Counter PDI (0x0443)

Bit	Description	ECAT	PDI	Reset Value
7:0	Watchdog PDI counter (counting is stopped when 0xFF is reached). Counts if PDI Watchdog expires. Cleared if one of the Watchdog counters 0x0442:0x0443 is written.	r/ w(clr)	r/-	0

3.44 SII EEPROM Interface (0x0500:0x050F)

Table 62: SII EEPROM Interface Register overview

Register Address	Length (Byte)	Description
0x0500	1	EEPROM Configuration
0x0501	1	EEPROM PDI Access State
0x0502:0x0503	2	EEPROM Control/Status
0x0504:0x0507	4	EEPROM Address
0x0508:0x050F	4/8	EEPROM Data

EtherCAT controls the SII EEPROM interface if EEPROM configuration register 0x0500.0=0 and EEPROM PDI Access register 0x0501.0=0, otherwise PDI controls the EEPROM interface.

In EEPROM emulation mode (IP Core with selected feature only), the PDI executes outstanding EEPROM commands. The PDI has access to some registers while the EEPROM Interface is busy.

Table 63: Register EEPROM Configuration (0x0500)

Bit	Description	ECAT	PDI	Reset Value
0	EEPROM control is offered to PDI: 0: no 1: yes (PDI has EEPROM control)	r/w	r/-	0
1	Force ECAT access: 0: Do not change Bit 501.0 1: Reset Bit 501.0 to 0	r/w	r/-	0
7:2	Reserved, write 0	r/-	r/-	0

Table 64: Register EEPROM PDI Access State (0x0501)

Bit	Description	ECAT	PDI	Reset Value
0	Access to EEPROM: 0: PDI releases EEPROM access 1: PDI takes EEPROM access (PDI has EEPROM control)	r/-	r/(w)	0
7:1	Reserved, write 0	r/-	r/-	0

NOTE: r/(w): write access is only possible if 0x0500.0=1 and 0x0500.1=0.

Table 65: Register EEPROM Control/Status (0x0502:0x0503)

Bit	Description				
		ESC20 [7]	ET1100	ET1200	IP Core
		ECAT	PDI	Reset Value	
0	ECAT write enable* ² : 0: Write requests are disabled 1: Write requests are enabled This bit is always 1 if PDI has EEPROM control.	r/(w)	r/-	0	
4:1	Reserved, write 0	r/-	r/-	0	
5	EEPROM emulation: 0: Normal operation (I ² C interface used) 1: PDI emulates EEPROM (I ² C not used)				
6	Supported number of EEPROM read bytes: 0: 4 Bytes 1: 8 Bytes	r/-	r/-	ET1100: 1 ET1200: 1 Others: 0	
7	Selected EEPROM Algorithm: 0: 1 address byte (1KBit – 16KBit EEPROMs) 1: 2 address bytes (32KBit – 4 MBit EEPROMs)	r/-	r/-	ESC20: 0* ¹ IP Core: depending on PROM_SIZE and features Others: PIN EEPROM size	
10:8	Command register* ² : Write: Initiate command. Read: Currently executed command Commands: 000: No command/EEPROM idle (clear error bits) 001: Read 010: Write 100: Reload Others: Reserved/invalid commands (do not issue) EEPROM emulation only: after execution, PDI writes command value to indicate operation is ready.	r/(w)	r/(w) r/[w]	0	
11	Checksum Error at in ESC Configuration Area: 0: Checksum ok 1: Checksum error EEPROM emulation for IP Core only: PDI writes 1 if reload failure has occurred.	r/-	r/- r/[w]	0	
12	EEPROM loading status: 0: EEPROM loaded, device information ok 1: EEPROM not loaded, device information not available (EEPROM loading in progress or finished with a failure)	r/-	r/-	0	
13	Error Acknowledge/Command* ³ : 0: No error 1: Missing EEPROM acknowledge or invalid command EEPROM emulation only: PDI writes 1 if a temporary failure has occurred.	r/-	r/- r/[w]	0	
14	Error Write Enable* ³ : 0: No error 1: Write Command without Write enable	r/-	r/-	0	

Bit	Description	ECAT	PDI	Reset Value
15	Busy: 0: EEPROM Interface is idle 1: EEPROM Interface is busy	r/-	r/-	0

NOTE: r/(w): write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (0x0502.15=1).

NOTE: r/[w]: EEPROM emulation/IP Core only: write access is possible if EEPROM interface is busy (0x0502.15=1). PDI acknowledges pending commands by writing a 1 into the corresponding command register bits (0x0502[10:8]). Errors can be indicated by writing a 1 into the error bits (0x0502.11 and 0x0502.13). Acknowledging clears AL Event Request 0x0220[5].

*¹ ESC20: configurable with pin EEPROM SIZE, but not readable in this register.

*² Write Enable bit 0 is self-clearing at the SOF of the next frame, Command bits [10:8] are self-clearing after the command is executed (EEPROM Busy ends). Writing "000" to the command register will also clear the error bits [14:13]. Command bits [10:8] are ignored if Error Acknowledge/Command is pending (bit 13).

*³ Error bits are cleared by writing "000" (or any valid command) to Command Register Bits [10:8].

Table 66: Register EEPROM Address (0x0504:0x0507)

Bit	Description	ECAT	PDI	Reset Value
31:0	EEPROM Address 0: First word (= 16 bit) 1: Second word ... Actually used EEPROM Address bits: [9:0]: EEPROM size up to 16 kBit [17:0]: EEPROM size 32 kBit – 4 Mbit [32:0]: EEPROM Emulation	r/(w)	r/(w)	0

NOTE: r/(w): write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (0x0502.15=1).

Table 67: Register EEPROM Data (0x0508:0x050F [0x0508:0x050B])

Bit	Description	ECAT	PDI	Reset Value
15:0	EEPROM Write data (data to be written to EEPROM) or EEPROM Read data (data read from EEPROM, lower bytes)	r/(w)	r/(w) r/[w]	0
63:16	EEPROM Read data (data read from EEPROM, higher bytes)	r/-	r/- r/[w]	

NOTE: r/(w): write access depends upon the assignment of the EEPROM interface (ECAT/PDI). Write access is generally blocked if EEPROM interface is busy (0x0502.15=1).

NOTE: r/[w]: write access for EEPROM emulation (IP Core only) if read or reload command is pending. See the following information for further details:

3.44.1 EEPROM emulation with IP Core

Write access to EEPROM Data register 0x0508:0x050F is possible if EEPROM interface is busy (0x0502.15=1). PDI places EEPROM read data in this register before the pending EEPROM Read command is acknowledged (writing to 0x0502[10:8]). For Reload command: place the following information in the EEPROM Data register before acknowledging the command. This data is automatically transferred to the designated registers when the Reload command is acknowledged:

Table 68: Register EEPROM Data for EEPROM Emulation Reload IP Core (0x0508:0x050F)

Bit	Description	ECAT	PDI	Reset Value
		ESC20	ET1100	ET1200
15:0	Configured Station Alias (reloaded into 0x0012[15:0])	r/-	r/[w]	0
16	Enhanced Link Detection for all ports (reloaded into 0x0140[9])	r/-	r/[w]	0
20:17	Enhanced Link Detection for individual ports (reloaded into 0x0140[15:12])	r/-	r/[w]	0
63:21	Reserved, write 0	r/-	r/[w]	0

3.45 MII Management Interface (0x0510:0x0515)

Table 69: MII Management Interface Register Overview

Register Address	Length (Byte)	Description
0x0510:0x0511	2	MII Management Control/Status
0x0512	1	PHY Address
0x0513	1	PHY Register Address
0x0514:0x0515	2	PHY Data
0x0516	1	MII Management ECAT Access State
0x0517	1	MII Management PDI Access State
0x0518:0x051B	4	PHY Port Status

IP Core only: PDI controls the MII management interface if MII Management PDI Access register 0x0517.0=1, otherwise EtherCAT controls the MII management interface.

ET1100 only: PDI controls the MII management interface only if Transparent Mode is enabled.

Table 70: Register MII Management Control/Status (0x0510:0x0511)

Bit	Description	ECAT	PDI	Reset Value			
				ESC20	ET1100	ET1200	IP Core
		[4:3]	[4:3]	[1:3]	[13]	V2.0.0/ V2.00a	
0	Write enable*: 0: Write disabled 1: Write enabled This bit is always 1 if PDI has MI control. ET1100-0000/-0001 exception: Bit is not always 1 if PDI has MI control, and bit is writable by PDI.	r/(w)	r/-		0		
1	Management Interface can be controlled by PDI (registers 0x0516-0 x0517): 0: Only ECAT control 1: PDI control possible	r/-	r/-		IP Core: Depends on configuration Others: 0		
2	MI link detection (link configuration, link detection, registers 0x0518-0x051B): 0: Not available 1: MI link detection active	r/-	r/-		IP Core: Depends on configuration Others: 0		
7:3	PHY address offset NOTE: ET1100, ET1200, and IP Core up to V1.1.1/V1.01b support only PHY address offsets 0 or 16. ESC20 supports no PHY address offset.	r/-	r/-		ET1100, ET1200: PHYAD_OFF for bit 7 IP Core: PHYAD_OFF_VEC or PHYAD_OFF for bit 7 Others: 0		
9:8	Command register*: Write: Initiate command. Read: Currently executed command Commands: 00: No command/MI idle (clear error bits) 01: Read 10: Write Others: Reserved/invalid commands (do not issue)	r/(w)	r/(w)		0		
12:10	Reserved, write 0	r/-	r/-		0		
13	Read error: 0: No read error 1: Read error occurred (PHY or register not available) Cleared by writing to this register.	r/(w)	r/(w)		0		
14	Command error: 0: Last Command was successful 1: Invalid command or write command without Write Enable Cleared with a valid command or by writing “00” to Command register bits [9:8].	r/-	r/-		0		
15	Busy: 0: MI control state machine is idle 1: MI control state machine is active	r/-	r/-		0		

NOTE: r/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (0x0510.15=1).

* Write enable bit 0 is self-clearing at the SOF of the next frame (or at the end of the PDI access), Command bits [9:8] are self-clearing after the command is executed (Busy ends). Writing “00” to the command register will also clear the error bits [14:13]. The Command bits are cleared after the command is executed.

Table 71: Register PHY Address (0x0512)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
4:0	PHY Address	r/(w)	r/(w)	0
7:5	Reserved, write 0	r/-	r/-	0

NOTE: r/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (0x0510.15=1).

Table 72: Register PHY Register Address (0x0513)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
4:0	Address of PHY Register that shall be read/written	r/(w)	r/(w)	0
7:5	Reserved, write 0	r/-	r/-	0

NOTE: r/ (w): write access depends on assignment of MI (ECAT/PDI). Write access is generally blocked if Management interface is busy (0x0510.15=1).

Table 73: Register PHY Data (0x0514:0x0515)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
15:0	PHY Read/Write Data	r/(w)	r/(w)	0

NOTE: r/ (w): write access depends on assignment of MI (ECAT/PDI). Access is generally blocked if Management interface is busy (0x0510.15=1).

Table 74: Register MII Management ECAT Access State (0x0516)

Bit	Description	ECAT	PDI	Reset Value
0	Access to MII management: 0: ECAT enables PDI takeover of MII management control 1: ECAT claims exclusive access to MII management	r/(w)	r/-	0
7:1	Reserved, write 0	r/-	r/-	0

NOTE: r/ (w): write access is only possible if 0x0517.0=0.

Table 75: Register MII Management PDI Access State (0x0517)

Bit	Description	ECAT	PDI	Reset Value
0	Access to MII management: 0: ECAT has access to MII management 1: PDI has access to MII management	r/-	r/(w)	0
1	Force PDI Access State: 0: Do not change Bit 517.0 1: Reset Bit 517.0 to 0	r/w	r/-	0
7:2	Reserved, write 0	r/-	r/-	0

NOTE: r/ (w): write access to bit 0 is only possible if 0x0516.0=0 and 0x0517.1=0.

Table 76: Register PHY Port y (port number y=0 to 3) Status (0x0518+y)

Bit	Description	ECAT	PDI	Reset Value
0	Physical link status (PHY status register 1.2): 0: No physical link 1: Physical link detected	r/-	r/-	0
1	Link status (100 Mbit/s, Full Duplex, Autonegotiation): 0: No link 1: Link detected	r/-	r/-	0
2	Link status error: 0: No error 1: Link error, link inhibited	r/-	r/-	0
3	Read error: 0: No read error occurred 1: A read error has occurred Cleared by writing any value to at least one of the PHY Status Port y registers.	r/(w/clr)	r/(w/clr)	0
4	Link partner error: 0: No error detected 1: Link partner error	r/-	r/-	0
5	PHY configuration updated: 0: No update 1: PHY configuration was updated Cleared by writing any value to at least one of the PHY Status Port y registers.	r/(w/clr)	r/(w/clr)	0
7:6	Reserved	r/-	r/-	0

NOTE: r/ (w): write access depends on assignment of MI (ECAT/PDI).

3.46 FMMU (0x0600:0x06FF)

Each FMMU entry is described in 16 Bytes from 0x0600:0x060F to 0x06F0:0x06FF. y is the FMMU index (y=0 to 15).

Table 77: FMMU Register overview

Register Address Offset	Length (Byte)	Description
+0x0:0x3	4	Logical Start Address
+0x4:0x5	2	Length
+0x6	1	Logical Start bit
+0x7	1	Logical Stop bit
+0x8:0x9	2	Physical Start Address
+0xA	1	Physical Start bit
+0xB	1	Type
+0xC	1	Activate
+0xD:0xF	3	Reserved

Table 78: Register Logical Start address FMMU y (0x06y0:0x06y3)

Bit	Description				
		ESC20	ET1100	ET1200	IP Core
31:0	Logical start address within the EtherCAT Address Space.	r/w	r/-	0	

Table 79: Register Length FMMU y (0x06y4:0x06y5)

Bit	Description				
		ESC20	ET1100	ET1200	IP Core
15:0	Offset from the first logical FMMU Byte to the last FMMU Byte + 1 (e.g., if two bytes are used then this parameter shall contain 2)	r/w	r/-	0	

Table 80: Register Start bit FMMU y in logical address space (0x06y6)

Bit	Description				
		ESC20	ET1100	ET1200	IP Core
2:0	Logical starting bit that shall be mapped (bits are counted from least significant bit (=0) to most significant bit(=7))	r/w	r/-	0	
7:3	Reserved, write 0	r/-	r/-	0	

Table 81: Register Stop bit FMMU y in logical address space (0x06y7)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
2:0	Last logical bit that shall be mapped (bits are counted from least significant bit (=0) to most significant bit(=7))	r/w	r/-	0
7:3	Reserved, write 0	r/-	r/-	0

Table 82: Register Physical Start address FMMU y (0x06y8-0x06y9)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
15:0	Physical Start Address (mapped to logical Start address)	r/w	r/-	0

Table 83: Register Physical Start bit FMMU y (0x06yA)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
2:0	Physical starting bit as target of logical start bit mapping (bits are counted from least significant bit (=0) to most significant bit(=7))	r/w	r/-	0
7:3	Reserved, write 0	r/-	r/-	0

Table 84: Register Type FMMU y (0x06yB)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
0	0: Ignore mapping for read accesses 1: Use mapping for read accesses	r/w	r/-	0
1	0: Ignore mapping for write accesses 1: Use mapping for write accesses	r/w	r/-	0
7:2	Reserved, write 0	r/-	r/-	0

Table 85: Register Activate FMMU y (0x06yC)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
0	0: FMMU deactivated 1: FMMU activated. FMMU checks logical addressed blocks to be mapped according to mapping configured	r/w	r/-	0
7:1	Reserved, write 0	r/-	r/-	0

Table 86: Register Reserved FMMU y (0x06yD:0x06yF)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
23:0	Reserved, write 0	r/-	r/-	0

3.47 SyncManager (0x0800:0x087F)

SyncManager registers are mapped from 0x0800:0x0807 to 0x0818:0x087F. y specifies SyncManager (y=0 to 15).

Table 87: SyncManager Register overview

Register Address Offset	Length (Byte)	Description
+0x0:0x1	2	Physical Start Address
+0x2:0x3	2	Length
+0x4	1	Control Register
+0x5	1	Status Register
+0x6	1	Activate
+0x7	1	PDI Control

Table 88: Register physical Start Address SyncManager y (0x0800+y*8:0x0801+y*8)

Bit	Description	ESC20	ET1100	ET1200	IP Core
		ECAT	PDI	Reset Value	
15:0	Specifies first byte that will be handled by SyncManager	r/(w)	r/-	0	

NOTE r/(w): Register can only be written if SyncManager is disabled (+0x6.0 = 0).

Table 89: Register Length SyncManager y (0x0802+y*8:0x0803+y*8)

Bit	Description	ESC20	ET1100	ET1200	IP Core
		ECAT	PDI	Reset Value	
15:0	Number of bytes assigned to SyncManager (shall be greater 1, otherwise SyncManager is not activated. If set to 1, only Watchdog Trigger is generated if configured)	r/(w)	r/-	0	

NOTE r/(w): Register can only be written if SyncManager is disabled (+0x6.0 = 0).

Table 90: Register Control Register SyncManager y (0x0804+y*8)

Bit	Description	ECAT	PDI	Reset Value
1:0	Operation Mode: 00: Buffered (3 buffer mode) 01: Reserved 10: Mailbox (Single buffer mode) 11: Reserved	r/(w)	r/-	00
3:2	Direction: 00: Read: ECAT read access, PDI write access. 01: Write: ECAT write access, PDI read access. 10: Reserved 11: Reserved	r/(w)	r/-	00
4	Interrupt in ECAT Event Request Register: 0: Disabled 1: Enabled	r/(w)	r/-	0
5	Interrupt in PDI Event Request Register: 0: Disabled 1: Enabled	r/(w)	r/-	0
6	Watchdog Trigger Enable: 0: Disabled 1: Enabled	r/(w)	r/-	0
7	Reserved, write 0	r/-	r/-	0

NOTE r/(w): Register can only be written if SyncManager is disabled (+0x6.0 = 0).

Table 91: Register Status Register SyncManager y (0x0805+y*8)

Bit	Description	ECAT	PDI		Reset Value
			ESC20 [7:6]	ET1100 [7:6]	ET1200 [7:6]
0	Interrupt Write: 1: Interrupt after buffer was completely and successfully written 0: Interrupt cleared after first byte of buffer was read	r/-	r/-	r/-	0
1	Interrupt Read: 1: Interrupt after buffer was completely and successful read 0: Interrupt cleared after first byte of buffer was written	r/-	r/-	r/-	0
2	Reserved	r/-	r/-	r/-	0
3	Mailbox mode: mailbox status: 0: Mailbox empty 1: Mailbox full Buffered mode: reserved	r/-	r/-	r/-	0
5:4	Buffered mode: buffer status (last written buffer): 00: 1. buffer 01: 2. buffer 10: 3. buffer 11: (no buffer written) Mailbox mode: reserved	r/-	r/-	r/-	11
6	Read buffer in use (opened)	r/-	r/-	r/-	0
7	Write buffer in use (opened)	r/-	r/-	r/-	0

Table 92: Register Activate SyncManager y (0x0806+y*8)

Bit	Description	ECAT	ESC20	ET1100	ET1200	IP Core
			[7:6]	[7:6]	[7:6]	[7:6]
0	SyncManager Enable/Disable: 0: Disable: Access to Memory without SyncManager control 1: Enable: SyncManager is active and controls Memory area set in configuration	r/w	r(ack)/-	0		
1	Repeat Request: A toggle of Repeat Request means that a mailbox retry is needed (primarily used in conjunction with ECAT Read Mailbox)	r/w	r(ack)/-	0		
5:2	Reserved, write 0	r/-	r(ack)/-	0		
6	Latch Event ECAT: 0: No 1: Generate Latch event if EtherCAT master issues a buffer exchange	r/w	r(ack)/-	0		
7	Latch Event PDI: 0: No 1: Generate Latch events if PDI issues a buffer exchange or if PDI accesses buffer start address	r/w	r(ack)/-	0		

NOTE: Reading this register from PDI in all SyncManagers which have changed activation clears AL Event Request 0x0220[4].

Table 93: Register PDI Control SyncManager y (0x0807+y*8)

Bit	Description	ECAT	ESC20	ET1100	ET1200	IP Core
			[7:6]	[7:6]	[7:6]	[7:6]
0	Deactivate SyncManager: Read: 0: Normal operation, SyncManager activated. 1: SyncManager deactivated and reset SyncManager locks access to Memory area. Write: 0: Activate SyncManager 1: Request SyncManager deactivation NOTE: Writing 1 is delayed until the end of a frame which is currently processed.	r/-	r/w	0		
1	Repeat Ack: If this is set to the same value as set by Repeat Request, the PDI acknowledges the execution of a previous set Repeat request.	r/-	r/w	0		
7:2	Reserved, write 0	r/-	r/-	0		

3.48 Distributed Clocks (0x0900:0x09FF)

Table 94: Distributed Clocks Register overview

Register Address	Length (Byte)	Description
		DC – Receive Times
0x0900:0x0903	4	Receive Time Port 0
0x0904:0x0907	4	Receive Time Port 1
0x0908:0x090B	4	Receive Time Port 2
0x090C:0x090F	4	Receive Time Port 3
		DC – Time Loop Control Unit
0x0910:0x0917	4/8	System Time
0x0918:0x091F	4/8	Receive Time ECAT Processing Unit
0x0920:0x0927	4/8	System Time Offset
0x0928:0x092B	4	System Time Delay
0x092C:0x092F	4	System Time Difference
0x0930:0x0931	2	Speed Counter Start
0x0932:0x0933	2	Speed Counter Diff
0x0934	1	System Time Difference Filter Depth
0x0935	1	Speed Counter Filter Depth
		DC – Cyclic Unit Control
0x0980	1	Cyclic Unit Control
		DC – SYNC Out Unit
0x0981	1	Activation
0x0982:0x0983	2	Pulse Length of SyncSignals
0x098E	1	SYNC0 Status
0x098F	1	SYNC1 Status
0x0990:0x0997	4/8	Start Time Cyclic Operation/Next SYNC0 Pulse
0x0998:0x099F	4/8	Next SYNC1 Pulse
0x09A0:0x09A3	4	SYNC0 Cycle Time
0x09A4:0x09A7	4	SYNC1 Cycle Time
		DC – Latch In Unit
0x09A8	1	Latch0 Control
0x09A9	1	Latch1 Control
0x09AE	1	Latch0 Status
0x09AF	1	Latch1 Status
0x09B0:0x09B7	4/8	Latch0 Time Positive Edge
0x09B8:0x09BF	4/8	Latch0 Time Negative Edge
0x09C0:0x09C7	4/8	Latch1 Time Positive Edge
0x09C8:0x09CF	4/8	Latch1 Time Negative Edge
		DC – SyncManager Event Times
0x09F0:0x09F3	4	EtherCAT Buffer Change Event Time
0x09F8:0x09FB	4	PDI Buffer Start Event Time
0x09FC:0x09FF	4	PDI Buffer Change Event Time

3.48.1 Receive Times

Table 95: Register Receive Time Port 0 (0x0900:0x0903)

Bit	Description	ECAT	PDI	Reset Value
31:0	<p>Write:</p> <p>A write access to register 0x0900 with BWR, APWR (any address) or FPWR (configured address) latches the local time of the beginning of the receive frame (start first bit of preamble) at each port.</p> <p>Write (ESC20, ET1200 exception):</p> <p>A write access latches the local time of the beginning of the receive frame at port 0. It enables the time stamping at the other ports.</p> <p>Read:</p> <p>Local time of the beginning of the last receive frame containing a write access to this register.</p> <p>NOTE: The time stamps cannot be read in the same frame in which this register was written.</p>	r/w (special function)	r/-	Undefined

Table 96: Register Receive Time Port 1 (0x0904:0x0907)

Bit	Description	ECAT	PDI	Reset Value
31:0	<p>Local time of the beginning of a frame (start first bit of preamble) received at port 1 containing a BWR/APWR or FPWR to Register 0x0900.</p> <p>ESC20, ET1200 exception:</p> <p>Local time of the beginning of the first frame received at port 1 after time stamping was enabled. Time stamping is disabled for this port afterwards.</p>	r/-	r/-	Undefined

Table 97: Register Receive Time Port 2 (0x0908:0x090B)

Bit	Description	ECAT	PDI	Reset Value
31:0	Local time of the beginning of a frame (start first bit of preamble) received at port 2 containing a BWR/APWR or FPWR to Register 0x0900.	r/-	r/-	Undefined

Table 98: Register Receive Time Port 3 (0x090C:0x090F)

Bit	Description	ECAT	PDI	Reset Value
31:0	Local time of the beginning of a frame (start first bit of preamble) received at port 3 containing a BWR/APWR or FPWR to Register 0x0900. ET1200 exception: Local time of the beginning of the first frame received at port 3 after time stamping was enabled. Time stamping is disabled for this port afterwards.	r/-	r/-	Undefined

NOTE: Register 0x0910:0x0913[0x910:0x917] is described in the next chapter.

Table 99: Register Receive Time ECAT Processing Unit (0x0918:0x091F)

Bit	Description	ECAT	PDI	Reset Value
63:0	Local time of the beginning of a frame (start first bit of preamble) received at the ECAT Processing Unit containing a write access to Register 0x0900 ESC20, ET1200 exception: Local time of the beginning of the frame received at the ECAT Processing Unit containing a write access to register 0x0900:0x0903. NOTE: E.g., if port 0 is open, this register reflects the Receive Time Port 0 as a 64 Bit value.	r/-	r/-	Undefined

3.48.2 Time Loop Control Unit

Time Loop Control unit is usually assigned to ECAT. Write access to Time Loop Control registers by PDI (and not ECAT) is only possible with explicit IP Core configuration.

Table 100: Register System Time (0x0910:0x0913 [0x0910:0x0917])

Bit	Description	ECAT	PDI	Reset Value
		[63:32]		[63:32] config.
63:0	ECAT read access: Local copy of the System Time when the frame passed the reference clock (i.e., including System Time Delay). Time latched at beginning of the frame (Ethernet SOF delimiter)	r	-	0
63:0	PDI read access: Local copy of the System Time. Time latched when reading first byte (0x0910)	-	r	
31:0	Write access: Written value will be compared with the local copy of the System time. The result is an input to the time control loop. NOTE: written value will be compared at the end of the frame with the latched (SOF) local copy of the System time if at least the first byte (0x0910) was written.	(w) (special function)	-	
31:0	Write access: Written value will be compared with Latch0 Time Positive Edge time. The result is an input to the time control loop. NOTE: written value will be compared at the end of the access with Latch0 Time Positive Edge (0x09B0:0x09B3) if at least the last byte (0x0913) was written.	-	(w) (special function)	

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

NOTE: Register 0x0918:0x091F is described in the previous chapter.

Table 101: Register System Time Offset (0x0920:0x0923 [0x0920:0x0927])

Bit	Description	ECAT	PDI	Reset Value
		[63:32]		[63:32] config.
63:0	Difference between local time and System Time. Offset is added to the local time.	r/(w)	r/(w)	0

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

Table 102: Register System Time Delay (0x0928:0x092B)

Bit	Description	ECAT	PDI	Reset Value
31:0	Delay between Reference Clock and the ESC	r/(w)	r/(w)	0

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

Table 103: Register System Time Difference (0x092C:0x092F)

Bit	Description	ECAT	PDI	Reset Value
30:0	Mean difference between local copy of System Time and received System Time values	r/-	r/-	0
31	0: Local copy of System Time greater than or equal received System Time 1: Local copy of System Time smaller than received System Time	r/-	r/-	0

Table 104: Register Speed Counter Start (0x0930:0x931)

Bit	Description	ECAT	PDI	Reset Value
14:0	Bandwidth for adjustment of local copy of System Time (larger values → smaller bandwidth and smoother adjustment) A write access resets System Time Difference (0x092C:0x092F) and Speed Counter Diff (0x0932:0x0933). Valid range: 0x0080 to 0xFFFF	r/(w)	r/(w)	0x1000
15	Reserved, write 0	r/-	r/-	0

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

Table 105: Register Speed Counter Diff (0x0932:0x933)

Bit	Description	ECAT	PDI	Reset Value
15:0	Representation of the deviation between local clock period and Reference Clock's clock period (representation: two's complement) Range: ±(Speed Counter Start – 0x7F)	r/-	r/-	0x0000

NOTE: Calculate the clock deviation after System Time Difference has settled at a low value as follows:

$$\text{Deviation} = \frac{\text{Speed Counter Diff}}{5(\text{Speed Counter Start} + \text{Speed Counter Diff} + 2)(\text{Speed Counter Start} - \text{Speed Counter Diff} + 2)}$$

Table 106: Register System Time Difference Filter Depth (0x0934)

Bit	Description	ECAT	PDI	Reset Value
3:0	Filter depth for averaging the received System Time deviation IP Core since V2.2.0/V2.02a: A write access resets System Time Difference (0x092C:0x092F)	r/(w)	r/(w)	4
7:4	Reserved, write 0	r/-	r/-	0

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

ET1100, ET1200, ESC20, IP Core before V2.2.0/V2.02a: Reset System Time Difference by writing Speed Counter Start (0x0930:0x0931) after changing this value.

Table 107: Register Speed Counter Filter Depth (0x0935)

Bit	Description	ECAT	PDI	Reset Value
3:0	Filter depth for averaging the clock period deviation IP Core since V2.2.0/V2.02a: A write access resets the internal speed counter filter.	r/(w)	r/(w)	12
7:4	Reserved, write 0	r/-	r/-	0

NOTE: Write access to this register depends upon ESC configuration (typically ECAT, PDI only with explicit ESC configuration: System Time PDI controlled).

ET1100, ET1200, ESC20, IP Core before V2.2.0/V2.02a: Reset internal speed counter filter by writing Speed Counter Start (0x0930:0x0931) after changing this value.

3.48.3 Cyclic Unit Control

Table 108: Register Cyclic Unit Control (0x0980)

Bit	Description	ECAT	PDI	Reset Value
0	SYNC out unit control: 0: ECAT controlled 1: PDI controlled	r/w	r/-	0
3:1	Reserved, write 0	r/-	r/-	0
4	Latch In unit 0: 0: ECAT controlled 1: PDI controlled NOTE: Always 1 (PDI controlled) if System Time is PDI controlled. Latch interrupt is routed to ECAT/PDI depending on this setting	r/w	r/-	0
5	Latch In unit 1: 0: ECAT controlled 1: PDI controlled NOTE: Latch interrupt is routed to ECAT/PDI depending on this setting	r/w	r/-	0
7:6	Reserved, write 0	r/-	r/-	0

3.48.4 SYNC Out Unit

Table 109: Register Activation register (0x0981)

Bit	Description	ECAT	PDI	Reset Value
		[7:3]	[7:3]	[7:3] V2.2.0/ V2.02a
0	Sync Out Unit activation: 0: Deactivated 1: Activated NOTE: Write 1 after Start Time was written.	r/(w)	r/(w)	0
1	SYNC0 generation: 0: Deactivated 1: SYNC0 pulse is generated	r/(w)	r/(w)	0
2	SYNC1 generation: 0: Deactivated 1: SYNC1 pulse is generated	r/(w)	r/(w)	0
3	Auto-activation by writing Start Time Cyclic Operation (0x0990:0x0997): 0: Disabled 1: Auto-activation enabled. 0x0981.0 is set automatically after Start Time is written.	r/(w)	r/(w)	0
4	Extension of Start Time Cyclic Operation (0x0990:0x0993): 0: No extension 1: Extend 32 bit written Start Time to 64 bit	r/(w)	r/(w)	0
5	Start Time plausibility check: 0: Disabled. SyncSignal generation if Start Time is reached. 1: Immediate SyncSignal generation if Start Time is outside near future (see 0x0981.6)	r/(w)	r/(w)	0
6	Near future configuration (approx.): 0: $\frac{1}{2}$ DC width future (2^{31} ns or 2^{63} ns) 1: 2.1 sec. future (2^{31} ns)	r/(w)	r/(w)	0
7	SyncSignal debug pulse (Vasili bit): 0: Deactivated 1: Immediately generate a single debug ping on SYNC0 and SYNC1 according to 0x0981[2:1] This bit is self-clearing, always read 0.	r/(w)	r/(w)	0

NOTE: Write to this register depends upon setting of 0x0980.0.

Table 110: Register Pulse Length of SyncSignals (0x0982:0x983)

Bit	Description	ECAT	PDI	Reset Value
15:0	Pulse length of SyncSignals (in Units of 10ns) 0: Acknowledge mode: SyncSignal will be cleared by reading SYNC0/SYNC1 Status register	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM ADR 0x0002

Table 111: Register Activation Status (0x0984)

Bit	Description	ECAT	PDI	Reset Value
0	SYNC0 activation state: 0: First SYNC0 pulse is not pending 1: First SYNC0 pulse is pending	r/-	r/-	0
1	SYNC1 activation state: 0: First SYNC1 pulse is not pending 1: First SYNC1 pulse is pending	r/-	r/-	0
2	Start Time Cyclic Operation (0x0990:0x0997) plausibility check result when Sync Out Unit was activated: 0: Start Time was within near future 1: Start Time was out of near future (0x0981.6)	r/-	r/-	0
7:3	Reserved	r/-	r/-	0

Table 112: Register SYNC0 Status (0x098E)

Bit	Description	ECAT	PDI	Reset Value
0	SYNC0 state for Acknowledge mode. SYNC0 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode	r/-	r(ack)/-	0
7:1	Reserved	r/-	r/-	0

Table 113: Register SYNC1 Status (0x098F)

Bit	Description	ECAT	PDI	Reset Value
0	SYNC1 state for Acknowledge mode. SYNC1 in Acknowledge mode is cleared by reading this register from PDI, use only in Acknowledge mode	r/-	r(ack)/-	0
7:1	Reserved	r/-	r/-	0

Table 114: Register Start Time Cyclic Operation (0x0990:0x0993 [0x0990:0x0997])

		ESC20 [63:32]	ET1100	ET1200	IP Core [63:32] config.
Bit	Description	ECAT	PDI	Reset Value	
63:0	Write: Start time (System time) of cyclic operation in ns Read: System time of next SYNC0 pulse in ns	r/(w)	r/(w)	0	

NOTE: Write to this register depends upon setting of 0x0980.0. Only writable if 0x0981.0=0.

Auto-activation (0x0981.3=1): upper 32 bits are automatically extended if only lower 32 bits are written within one frame.

Table 115: Register Next SYNC1 Pulse (0x0998:0x099B [0x0998:0x099F])

		ESC20 [63:32]	ET1100	ET1200	IP Core [63:32] config.
Bit	Description	ECAT	PDI	Reset Value	
63:0	System time of next SYNC1 pulse in ns	r/-	r/-	0	

Table 116: Register SYNC0 Cycle Time (0x09A0:0x09A3)

		ESC20	ET1100	ET1200	IP Core
Bit	Description	ECAT	PDI	Reset Value	
31:0	Time between two consecutive SYNC0 pulses in ns. 0: Single shot mode, generate only one SYNC0 pulse.	r/(w)	r/(w)	0	

NOTE: Write to this register depends upon setting of 0x0980.0.

Table 117: Register SYNC1 Cycle Time (0x09A4:0x09A7)

		ESC20	ET1100	ET1200	IP Core
Bit	Description	ECAT	PDI	Reset Value	
31:0	Time between SYNC1 pulses and SYNC0 pulse in ns	r/(w)	r/(w)	0	

NOTE: Write to this register depends upon setting of 0x0980.0.

3.48.5 Latch In unit

Table 118: Register Latch0 Control (0x09A8)

Bit	Description	ECAT	PDI	Reset Value
0	Latch0 positive edge: 0: Continuous Latch active 1: Single event (only first event active)	r/(w)	r/(w)	0
1	Latch0 negative edge: 0: Continuous Latch active 1: Single event (only first event active)	r/(w)	r/(w)	0
7:2	Reserved, write 0	r/-	r/-	0

NOTE: Write access depends upon setting of 0x0980.4.

Table 119: Register Latch1 Control (0x09A9)

Bit	Description	ECAT	PDI	Reset Value
0	Latch1 positive edge: 0: Continuous Latch active 1: Single event (only first event active)	r/(w)	r/(w)	0
1	Latch1 negative edge: 0: Continuous Latch active 1: Single event (only first event active)	r/(w)	r/(w)	0
7:2	Reserved, write 0	r/-	r/-	0

NOTE: Write access depends upon setting of 0x0980.5.

Table 120: Register Latch0 Status (0x09AE)

Bit	Description	ECAT	ESC20	ET1100	ET1200	IP Core
			[2]	[2]	[2]	
0	Event Latch0 positive edge. 0: Positive edge not detected or continuous mode 1: Positive edge detected in single event mode only. Flag cleared by reading out Latch0 Time Positive Edge.	r/-	r/-		0	
1	Event Latch0 negative edge. 0: Negative edge not detected or continuous mode 1: Negative edge detected in single event mode only. Flag cleared by reading out Latch0 Time Negative Edge.	r/-	r/-		0	
2	Latch0 pin state	r/-	r/-		0	
7:3	Reserved	r/-	r/-		0	

Table 121: Register Latch1 Status (0x09AF)

Bit	Description	ECAT	ESC20	ET1100	ET1200	IP Core
			[2]	[2]	[2]	
0	Event Latch1 positive edge. 0: Positive edge not detected or continuous mode 1: Positive edge detected in single event mode only. Flag cleared by reading out Latch1 Time Positive Edge.	r/-	r/-		0	
1	Event Latch1 negative edge. 0: Negative edge not detected or continuous mode 1: Negative edge detected in single event mode only. Flag cleared by reading out Latch1 Time Negative Edge.	r/-	r/-		0	
2	Latch1 pin state	r/-	r/-		0	
7:3	Reserved	r/-	r/-		0	

Table 122: Register Latch0 Time Positive Edge (0x09B0:0x09B3 [0x09B0:0x09B7])

Bit	Description	ECAT	PDI	Reset Value
63:0	Register captures System time at the positive edge of the Latch0 signal. Reading clears Latch0 Status 0x09AE[0]	r(ack)/-	r(ack)/-	0

NOTE: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Clearing Latch0 Status flag function depends upon setting of 0x0980.4.

Table 123: Register Latch0 Time Negative Edge (0x09B8:0x09BB [0x09B8:0x09BF])

Bit	Description	ECAT	PDI	Reset Value
63:0	Register captures System time at the negative edge of the Latch0 signal. Reading clears Latch0 Status 0x09AE[1]	r(ack)/-	r(ack)/-	0

NOTE: ET1100: register is only available if 0x0140.11=1. Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Clearing Latch0 Status flag function depends upon setting of 0x0980.4.

Table 124: Register Latch1 Time Positive Edge (0x09C0:0x09C3 [0x09C0:0x09C7])

Bit	Description	ECAT	PDI	Reset Value
63:0	Register captures System time at the positive edge of the Latch1 signal. Reading clears Latch1 Status 0x09AF[0]	r(ack)/-	r(ack)/-	0

NOTE: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Clearing Latch1 Status flag function depends upon setting of 0x0980.5.

Table 125: Register Latch1 Time Negative Edge (0x09C8:0x09CB [0x09C8:0x09CF])

Bit	Description	ECAT	PDI	Reset Value
63:0	Register captures System time at the negative edge of the Latch1 signal. Reading clears Latch1 Status 0x09AF[1]	r(ack)/-	r(ack)/-	0

NOTE: Register bits [63:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value. Clearing Latch1 Status flag function depends upon setting of 0x0980.5.

3.48.6 SyncManager Event Times

Table 126: Register EtherCAT Buffer Change Event Time (0x09F0:0x09F3)

Bit	Description	ECAT	PDI	Reset Value
31:0	Register captures local time of the beginning of the frame which causes at least one SyncManager to assert an ECAT event	r/-	r/-	0

NOTE: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

Table 127: Register PDI Buffer Start Event Time (0x09F8:0x09FB)

Bit	Description	ECAT	PDI	Reset Value
31:0	Register captures local time when at least one SyncManager asserts an PDI buffer start event	r/-	r/-	0

NOTE: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

Table 128: Register PDI Buffer Change Event Time (0x09FC:0x09FF)

Bit	Description	ECAT	PDI	Reset Value
31:0	Register captures local time when at least one SyncManager asserts an PDI buffer change event	r/-	r/-	0

NOTE: Register bits [31:8] are internally latched (ECAT/PDI independently) when bits [7:0] are read, which guarantees reading a consistent value.

3.49 ESC specific registers (0x0E00:0x0EFF)

3.49.1 Power-On Values ET1200

Table 129: Register Power-On Values ET1200 (0x0E00)

Bit	Description	ECAT	PDI	Reset Value
1:0	Chip mode (MODE): 00: Port 0: EBUS, Port 1: EBUS, 18 bit PDI 01: Reserved 10: Port 0: MII, Port 1: EBUS, 8 bit PDI 11: Port 0: EBUS, Port 1: MII, 8 bit PDI	r/-	r/-	Depends on Hardware configuration
3:2	CPU clock output (CLK_MODE): 00: Off – PDI[7] available as PDI port 01: PDI[7] = 25MHz 10: PDI[7] = 20MHz 11: PDI[7] = 10MHz	r/-	r/-	
5:4	TX signal shift (C25_SHI): 00: MII TX signals shifted by 0° 01: MII TX signals shifted by 90° 10: MII TX signals shifted by 180° 11: MII TX signals shifted by 270°	r/-	r/-	
6	CLK25 Output Enable (C25_ENA): 0: Disabled – PDI[6] available as PDI port 1: Enabled – PDI[6] = 25MHz (OSC) NOTE: Only used in Chip mode 10 and 11	r/-	r/-	
7	PHY Address Offset (PHYAD_OFF): 0: No PHY address offset 1: PHY address offset is 16	r/-	r/-	

3.49.2 Power-On Values ET1100

Table 130: Register Power-On Values ET1100 (0x0E00:0x0E01)

Bit	Description	ECAT	PDI	Reset Value
1:0	Port mode (P_MODE): 00: Logical ports 0 and 1 available 01: Logical ports 0, 1 and 2 available 10: Logical ports 0, 1 and 3 available 11: Logical ports 0, 1, 2 and 3 available	r/-	r/-	Depends on Hardware configuration
5:2	Physical layer of available ports (P_CONF). Bit 2 → logical port 0, Bit 3 → logical port 1, Bit 4 → third logical port (2/3), Bit 5 → logical port 3. 0: EBUS 1: MII	r/-	r/-	
7:6	CPU clock output (CLK_MODE): 00: Off – PDI[7] available as PDI port 01: PDI[7] = 25MHz 10: PDI[7] = 20MHz 11: PDI[7] = 10MHz	r/-	r/-	
9:8	TX signal shift (C25_SHI): 00: MII TX signals shifted by 0° 01: MII TX signals shifted by 90° 10: MII TX signals shifted by 180° 11: MII TX signals shifted by 270°	r/-	r/-	
10	CLK25 Output Enable (C25_ENA): 0: Disabled – PDI[31] available as PDI port 1: Enabled – PDI[31] = 25MHz (OSC)	r/-	r/-	
11	Transparent Mode MII (Trans_Mode_Ena): 0: Disabled 1: Enabled – ERR is input (0: TX signals are tristated, 1: ESC is driving TX signals)	r/-	r/-	
12	Digital Control/State Move (Ctrl_Status_Move): 0: Control/Status signals are mapped to PDI[39:32] – if available 1: Control/Status signals are remapped to the highest available PDI Byte.	r/-	r/-	
13	PHY Address Offset (PHYAD_OFF): 0: No PHY address offset 1: PHY address offset is 16	r/-	r/-	
14	PHY Link Polarity (LINKPOL): 0: LINK_MII is active low 1: LINK_MII is active high	r/-	r/-	
15	Reserved configuration bit	r/-	r/-	

3.49.3 IP Core

Table 131: Register Product ID (0x0E00:0x0E07)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
63:0	Product ID	r/-	r/-	Depends on configuration				

Table 132: Register Vendor ID (0x0E08:0x0E0F)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
31:0	Vendor ID	r/-	r/-	Depends on License file				
63:32	Reserved	r/-	r/-					

3.49.4 ESC20

Table 133: Register FPGA Update (0x0E00:0x0EFF)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
	FPGA Update (ESC20 and TwinCAT only)							

3.50 Digital I/O Output Data (0x0F00:0x0F03)

Table 134: Register Digital I/O Output Data (0x0F00:0x0F03)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
31:0	Output Data	r/w	r/-	0				

NOTE: Register size depends on PDI setting and/or device configuration. This register is bit-writable (using Logical addressing).

3.51 General Purpose Outputs (0x0F10:0x0F17)

Table 135: Register General Purpose Outputs (0x0F10:0x0F17)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
[7:0]	General Purpose Output Data	r/w	r/w	0		{63:16} config.	{63:12}	
[15:0]								
[31:0]								
[63:0]								

NOTE: Register size depends on PDI setting and/or device configuration

3.52 General Purpose Inputs (0x0F18:0x0F1F)

Table 136: Register General Purpose Inputs (0x0F18:0x0F1F)

Bit	Description	ECAT	PDI	Reset Value	ESC20	ET1100	ET1200	IP Core
[7:0]	General Purpose Input Data	r/-	r/-	0		{63:16} config.		
[15:0]								
[31:0]								
[63:0]								

NOTE: Register size depends on PDI setting and/or device configuration

3.53 User RAM (0x0F80:0xFFFF)

Table 137: User RAM (0x0F80:0xFFFF)

Bit	Description	ECAT	PDI	Reset Value
ESC20 ET1100 ET1200 IP Core				
----	Application specific information	r/w	r/w	IP Core: Extended ESC features Others: Random/undefined

Table 138: Extended ESC Features (Reset values of User RAM)

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

Bit	Description	Reset Value
36	Reserved	
37	Reserved	
38	DC Time loop control assigned to PDI	
39	Link detection and configuration by MI	
40	MI control by PDI possible	
41	Automatic TX shift	
42	EEPROM emulation by µController	
43	Reserved	
44	Reserved	
45	Reserved	
46	Reserved	
47	Reserved	
48	Reserved	
49	Reserved	
50	ERR LED, RUN/ERR LED Override	
others	Reserved	Reserved

NOTE: Extended ESC features are only available with the IP Core.

4 Process Data RAM (0x1000:0xFFFF)

4.1 Digital I/O Input Data (0x1000:0x1003)

Digital I/O Input Data is written into the Process Data RAM by the Digital I/O PDI.

Table 139: Digital I/O Input Data (0x1000:0x1003)

Bit	Description	ECAT	PDI	Reset Value
31:0	Input Data	(r/w)	(r/w)	Random/undefined

NOTE (r/w): Process Data RAM is only accessible if EEPROM was correctly loaded (register 0x0110.0 = 1).

NOTE: Input Data size depends on PDI setting and/or device configuration. Digital I/O Input Data is written into the Process Data RAM at these addresses if a Digital I/O PDI with inputs is configured.

4.2 Process Data RAM (0x1000:0xFFFF)

The Process Data RAM starts at address 0x1000, its size depends on the ESC.

Table 140: Process Data RAM (0x1000:0xFFFF)

Bit	Description	ECAT	PDI	Reset Value
	Process Data RAM	(r/w)	(r/w)	Random/undefined

NOTE (r/w): Process Data RAM is only accessible if EEPROM was correctly loaded (register 0x0110.0 = 1).

5 Appendix

5.1 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.

5.1.1 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.

5.2 Beckhoff Headquarters

Beckhoff Automation GmbH
Eiserstr. 5
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

Hardware Data Sheet

ET1810 / ET1812

**EtherCAT® Slave Controller IP Core
for Altera® FPGAs**
IP Core Release 2.4.0

Section III – IP Core Description Installation, Configuration, Design flow, Interface specification

Version 1.0
Date: 2011-03-15

BECKHOFF

Trademarks

Beckhoff®, TwinCAT®, EtherCAT®, Safety over EtherCAT®, TwinSAFE® and XFC® are registered trademarks of and licensed by Beckhoff Automation GmbH. Other designations used in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owners.

Patent Pending

The EtherCAT Technology is covered, including but not limited to the following German patent applications and patents: DE10304637, DE102004044764, DE102005009224, DE102007017835 with corresponding applications or registrations in various other countries.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development. For that reason the documentation is not in every case checked for consistency with performance data, standards or other characteristics. In the event that it contains technical or editorial errors, we retain the right to make alterations at any time and without warning. No claims for the modification of products that have already been supplied may be made on the basis of the data, diagrams and descriptions in this documentation.

Copyright

© Beckhoff Automation GmbH 03/2011.

The reproduction, distribution and utilization of this document as well as the communication of its contents to others without express authorization are prohibited. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

DOCUMENT HISTORY

Version	Comment
1.0	<ul style="list-style-type: none">Initial release EtherCAT IP Core for Altera FPGAs 2.4.0

CONTENTS

Section III – IP Core Description	1
1 Overview	1
1.1 Frame processing order	2
1.2 Scope of this document	3
1.3 Scope of Delivery	3
1.4 Target FPGAs	4
1.5 Tested FPGA/Designflow combinations	5
1.6 Release Notes	6
1.7 Design flow	10
1.8 OpenCore Plus Evaluation	11
1.9 Simulation	12
2 Features and Registers	13
2.1 Features	13
2.2 Registers	18
2.3 Extended ESC Features in User RAM	21
3 IP Core Installation	23
3.1 Requirements	23
3.2 Installation on Windows PCs	24
3.3 Installation on Linux PCs	25
3.4 License File	26
3.5 IP Core Vendor ID package	27
3.6 Software Templates for reference designs with NIOS processor	28
3.7 EtherCAT Slave Information (ESI) / XML device description for reference designs	28
3.8 Quartus II libraries	28
4 IP Core Usage	29
4.1 MegaWizard Plug-In Manager	29
4.1.1 Implementing the EtherCAT IP Core	29
4.1.2 EtherCAT IP Core Configuration Interface	30
4.2 SOPC Builder	31
5 IP Core Configuration	33
5.1 Documentation	34
5.2 Parameters	35
5.2.1 Product ID tab	35
5.2.2 Physical Layer tab	36
5.2.3 Internal Functions tab	37
5.2.4 Feature Details tab	38
5.2.5 Process Data Interface tab	40
5.2.5.1 No Interface and General Purpose I/O	41
5.2.5.2 Digital I/O Configuration	42

5.2.5.3	µController Configuration (8/16Bit)	43
5.2.5.4	SPI Configuration	44
5.2.5.5	Avalon Configuration	45
6	Reference Designs	46
6.1	Beckhoff EL9800/FB1122 with Digital I/O	47
6.1.1	Configuration and resource consumption	47
6.1.2	Functionality	47
6.1.3	Implementation	47
6.1.4	SII EEPROM	47
6.1.5	Downloadable configuration file	47
6.2	Beckhoff EL9800/FB1122 with SPI	48
6.2.1	Configuration and resource consumption	48
6.2.2	Functionality	48
6.2.3	Implementation	48
6.2.4	SII EEPROM	48
6.3	EBV DBC3C40 with Digital I/O	49
6.3.1	Configuration and resource consumption	49
6.3.2	Functionality	49
6.3.3	Implementation	49
6.3.4	SII EEPROM	49
6.3.5	Downloadable configuration file	49
6.4	EBV DBC4CE55 with NIOS	51
6.4.1	Configuration and resource consumption	51
6.4.2	Functionality	51
6.4.3	Implementation	51
6.4.4	SII EEPROM	52
6.4.5	Downloadable configuration file	52
6.5	Altera DE2-115 with NIOS	53
6.5.1	Configuration and resource consumption	53
6.5.2	Functionality	53
6.5.3	Implementation	53
6.5.4	SII EEPROM	54
6.5.5	Downloadable configuration file	54
6.6	FPGA Download using Altera Quartus Programmer	55
6.7	Update SII EEPROM with EtherCAT Slave Information (ESI) using TwinCAT System Manager	56
7	FPGA Resource Consumption	57
8	IP Core Signals	59
8.1	Overview	59
8.2	General Signals	61
8.2.1	Clock source example schematics	61

8.3	SII EEPROM Interface Signals	62
8.4	LED Signals	63
8.5	Distributed Clocks SYNC/LATCH Signals	63
8.6	Physical Layer Interface	64
8.6.1	MII Interface	65
8.6.2	RMII Interface	67
8.7	PDI Signals	68
8.7.1	General PDI Signals	68
8.7.2	Digital I/O Interface	68
8.7.3	SPI Slave Interface	69
8.7.4	Asynchronous 8/16 Bit µController Interface	69
8.7.4.1	8 Bit µController Interface	70
8.7.4.2	16 Bit µController Interface	70
8.7.5	Avalon On-Chip Bus	71
9	Ethernet Interface	72
9.1	PHY Address Configuration	72
9.2	MII Interface	72
9.2.1	MII Interface Signals	73
9.2.2	TX Shift Compensation	74
9.2.3	MII Timing specifications	75
9.2.4	MII example schematic	76
9.3	RMII Interface	77
9.3.1	RMII Interface Signals	77
9.3.2	RMII example schematic	78
9.4	General Ethernet Timing specifications	79
10	PDI Description	80
10.1	Digital I/O Interface	81
10.1.1	Interface	81
10.1.2	Configuration	82
10.1.3	Digital Inputs	82
10.1.4	Digital Outputs	82
10.1.5	Output Enable	83
10.1.6	SyncManager Watchdog	83
10.1.7	SOF	84
10.1.8	OUTVALID	84
10.1.9	Timing specifications	84
10.2	SPI Slave Interface	86
10.2.1	Interface	86
10.2.2	Configuration	86
10.2.3	SPI access	87
10.2.4	Address modes	87

10.2.5	Commands	88
10.2.6	Interrupt request register (AL Event register)	88
10.2.7	Write access	88
10.2.8	Read access	88
10.2.8.1	Read Wait State	89
10.2.8.2	Read Termination	89
10.2.9	SPI access errors and SPI status flag	89
10.2.10	2 Byte and 4 Byte SPI Masters	90
10.2.11	Timing specifications	91
10.3	Asynchronous 8/16 bit µController Interface	97
10.3.1	Interface	97
10.3.2	Configuration	97
10.3.3	µController access	98
10.3.4	Write access	98
10.3.5	Read access	98
10.3.6	µController access errors	99
10.3.7	Connection with 16 bit µControllers without byte addressing	99
10.3.8	Connection with 8 bit µControllers	100
10.3.9	Timing Specification	101
10.4	Avalon Slave Interface	105
10.4.1	Interface	105
10.4.2	Configuration	106
10.4.3	Interrupts	106
10.4.4	Data Bus With and SyncManager Configuration	106
10.4.5	Timing specifications	107
11	Distributed Clocks SYNC/LATCH Signals	109
11.1	Signals	109
11.2	Timing specifications	109
12	SII EEPROM Interface (I ² C)	110
12.1	Signals	110
12.2	Timing specifications	110
13	Electrical Specifications	111
14	Synthesis Constraints	112
15	Appendix	115
15.1	Support and Service	115
15.1.1	Beckhoff's branch offices and representatives	115
15.2	Beckhoff Headquarters	115

TABLES

Table 1: IP Core Main Features	1
Table 2: Frame Processing Order	2
Table 3: Tested FPGA/Designflow combinations	5
Table 4: Release History	6
Table 5: Register Revision (0x0001)	9
Table 6: Register Build (0x0002:0x0003)	9
Table 7: IP Core Feature Details	13
Table 8: Legend	17
Table 9: Register availability depending on register preset	18
Table 10: Legend	20
Table 11: Extended ESC Features (Reset values of User RAM – 0x0F80:0xFFFF)	21
Table 12: Generated files description	30
Table 13: Resource consumption Digital I/O reference design EL9800/FB1122	47
Table 14: Resource consumption SPI reference design EL9800/FB1122	48
Table 15: Resource consumption Digital I/O reference design DBC3C40	49
Table 16: Resource consumption NIOS reference design DBC4CE55	51
Table 17: Resource consumption NIOS reference design DE2-115	53
Table 18: Typical need of Logic Cells (LE) for main configurable functions	57
Table 19: EtherCAT IP Core configuration for typical EtherCAT Devices	58
Table 20: Signal Overview	59
Table 21: PDI signal overview	60
Table 22: General Signals	61
Table 23: SII EEPROM Signals	62
Table 24: LED Signals	63
Table 25: DC SYNC/LATCH signals	63
Table 26: Physical Layer General	64
Table 27: PHY Interface MII	65
Table 28: PHY Interface RMII	67
Table 29: General PDI Signals	68
Table 30: Digital I/O PDI	68
Table 31: SPI PDI	69
Table 32: 8/16 Bit µC PDI	69
Table 33: 8 Bit µC PDI	70
Table 34: 16 Bit µC PDI	70
Table 35: Avalon PDI	71
Table 36: MII Interface signals	73
Table 37: MII TX Timing characteristics	75
Table 38: MII timing characteristics	75
Table 39: RMII Interface signals	78
Table 40: General Ethernet timing characteristics	79
Table 41: Available PDIs for EtherCAT IP Core	80
Table 42: IP core digital I/O signals	81
Table 43: Input/Output byte reference	81
Table 44: Digital I/O timing characteristics IP Core	84
Table 45: SPI signals	86
Table 46: Address modes	87
Table 47: SPI commands CMD0 and CMD1	88
Table 48: Interrupt request register transmission	88
Table 49: Write access for 2 and 4 Byte SPI Masters	90
Table 50: SPI timing characteristics IP Core	91
Table 51: Read/Write timing diagram symbols	92
Table 52: µController signals	97
Table 53: 8 bit µController interface access types	98
Table 54: 16 bit µController interface access types	98
Table 55: µController timing characteristics IP Core	101
Table 56: Avalon signals	105
Table 57: Avalon timing characteristics (IP Core V1.1.1)	107
Table 58: Distributed Clocks signals	109
Table 59: DC SYNC/LATCH timing characteristics IP Core	109
Table 60: I²C EEPROM signals	110

Table 61: EEPROM timing characteristics IP Core	110
Table 62: AC Characteristics.....	111
Table 63: Forwarding Delays.....	111
Table 64: EtherCAT IP Core constraints	112

FIGURES

Figure 1: EtherCAT IP Core Block Diagram	1
Figure 2: Frame Processing	2
Figure 3: Design flow	10
Figure 4: Files installed with EtherCAT IP Core setup	24
Figure 5: Files installed with ethercat-vX.Y.Z	25
Figure 6: License Setup.....	26
Figure 7: <IPInst_dir>\lib folder with vendor ID package (ETHERCAT_VENDORID.VHD)	27
Figure 8: MegaWizard Plug-In Manager	29
Figure 9: SOPC Builder	31
Figure 10: EtherCAT IP Core Configuration Interface.....	33
Figure 11: Documentation	34
Figure 12: Product ID tab	35
Figure 13: Physical Layer tab	36
Figure 14: Internal Functions tab.....	37
Figure 15: Feature Details tab	38
Figure 16: Available PDI Interfaces	40
Figure 17: Register Process Data Interface	41
Figure 18: Register PDI – Digital I/O Configuration.....	42
Figure 19: Register PDI – µC-Configuration.....	43
Figure 20: Register PDI – SPI Configuration.....	44
Figure 21: Register PDI – Avalon Interface Configuration	45
Figure 22: Quartus Programmer.....	55
Figure 23: EtherCAT IP Core clock source (MII).....	61
Figure 24: EtherCAT IP Core clock source (RMII)	62
Figure 25: MII Interface signals	73
Figure 26: MII TX Timing Diagram	74
Figure 27: MII timing RX signals.....	75
Figure 28: MII example schematic.....	76
Figure 29: RMII Interface signals.....	77
Figure 30: RMII example schematic	78
Figure 31: IP core digital I/O signals	81
Figure 32: Digital Output Principle Schematic.....	83
Figure 33: Digital Input: Input data sampled at SOF, I/O can be read in the same frame	85
Figure 34: Digital Input: Input data sampled with LATCH_IN	85
Figure 35: Digital Output timing	85
Figure 36: OUT_ENA timing.....	85
Figure 37: SPI master and slave interconnection.....	86
Figure 38: Basic SPI_DI/SPI_DO timing (*refer to timing diagram for relevant edges of SPI_CLK)	92
Figure 39: SPI read access (2 byte addressing, 1 byte read data) with Wait State byte	93
Figure 40: SPI read access (2 byte addressing, 2 byte read data) with Wait State byte	94
Figure 41: SPI write access (2 byte addressing, 1 byte write data)	95
Figure 42: SPI write access (3 byte addressing, 1 byte write data)	96
Figure 43: µController interconnection	97
Figure 44: Connection with 16 bit µControllers without byte addressing	99
Figure 45: Connection with 8 bit µControllers (BHE and DATA[15:8] should not be left open)	100
Figure 46: Read access (without preceding write access).....	103
Figure 47: Write access (write after rising edge nWR, without preceding write access)	103
Figure 48: Sequence of two write accesses and a read access	104
Figure 49: Write access (write after falling edge nWR)	104
Figure 50: Avalon signals	105
Figure 51: Avalon Read Access (32 Bit, W=4)	108
Figure 52: Avalon Write Access (32 Bit, W=4)	108
Figure 53: Distributed Clocks signals	109
Figure 54: LatchSignal timing	109
Figure 55: SyncSignal timing.....	109
Figure 56: I ² C EEPROM signals.....	110

ABBREVIATIONS

μ C	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	<ul style="list-style-type: none"> • 32 Bit Digital I/O (unidirectional) • SPI Slave • 8/16 bit asynchronous µController Interface • Avalon® on-chip bus
Other features	<ul style="list-style-type: none"> • Reference 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:

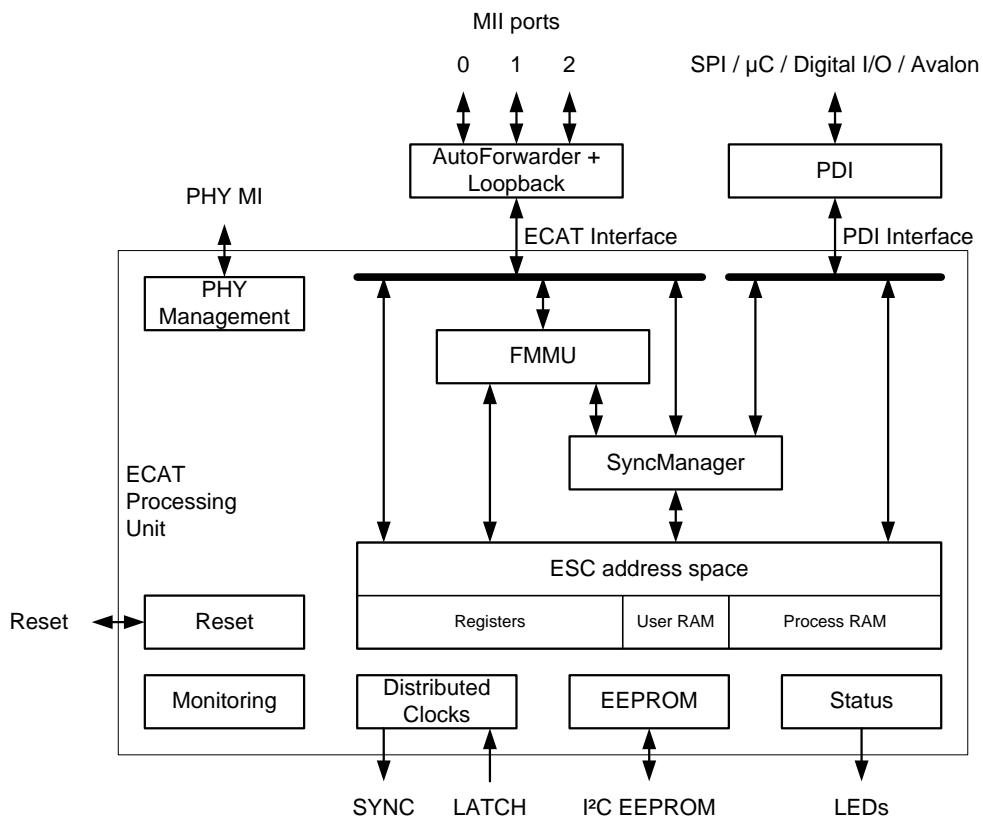


Figure 1: EtherCAT IP Core Block Diagram

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:

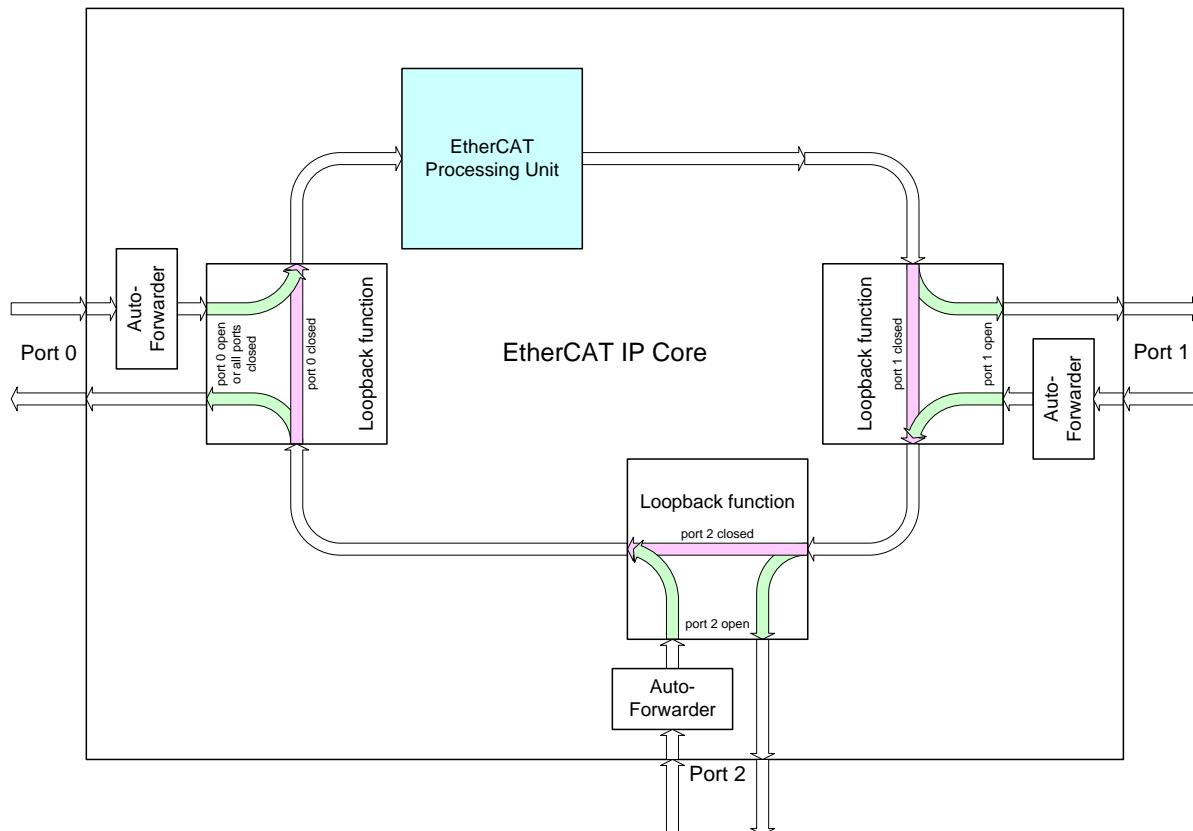


Figure 2: Frame Processing

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.

This documentation refers to the EtherCAT IP Core for Altera FPGAs version 2.4.0.

1.3 Scope of Delivery

The EtherCAT IP Core installation file includes:

- EtherCAT IP Core (encrypted VHDL library)
- Documentation
- Reference 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 IV E+GX
- Altera Stratix®, Stratix II, Stratix III, Stratix IV, Stratix V
- Altera Arria® GX, Arria II GX, Arria II GZ
- Altera Stratix® GX, Stratix II GX
- Intel® Atom™ Processor E6x5C (formerly Stellarton)

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 (e.g., Cyclone V).

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 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 Reference Designs
2.4.0	Cyclone	EP1C12	Quartus II 10.1	Synthesis	
	Cyclone II	EP2C20	Quartus II 10.1	Synthesis	
	Cyclone III	EP3C25	Quartus II 10.1	Hardware	FB1122 Digital I/O & SPI
	Cyclone III	EP3C40	Quartus II 10.1	Hardware	DBC3C40 Digital I/O
	Cyclone III LS	EP3CLS200	Quartus II 10.1	Synthesis	
	Cyclone IV E	EP4CE55	Quartus II 10.1	Hardware	DBC4CE55 Nios
	Cyclone IV E	EP4CE115	Quartus II 10.1	Hardware	DE2-115 Nios
	Stratix	EP1S10	Quartus II 10.1	Synthesis	
	Stratix GX	EP1SGX10	Quartus II 10.1	Synthesis	
	Stratix II	EP2S30	Quartus II 10.1	Synthesis	
	Stratix II GX	EP2SGX30	Quartus II 10.1	Synthesis	
	Stratix III	EP3SE50	Quartus II 10.1	Synthesis	
	Stratix IV E	EP4SE230	Quartus II 10.1	Synthesis	
	Stratix IV GT	EP4S40G	Quartus II 10.1	Synthesis	
	Stratix IV GX	EP4SGX70	Quartus II 10.1	Synthesis	
	Stratix V	5SGSMB7I2	Quartus II 10.1	Synthesis	
	Arria GX	EP1AGX20	Quartus II 10.1	Synthesis	
	Arria II GX	EP2AGX45	Quartus II 10.1	Synthesis	
	Intel Atom E6x5C	EP2AGXE6XXFPGA	Quartus II 10.1	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.

1.6 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 History

Version	Release notes
1.0.0 (7/2006)	<p>Initial release</p> <p>Restrictions removed in version 1.1.0:</p> <ul style="list-style-type: none"> • Synthesis problems with Avalon Clock >25 MHz. Workaround: set Avalon Clock to 25 MHz • SyncManager: Interrupt/Process Data Watchdog trigger generation depends on read interaction if SM is configured for 1 byte length and Mailbox mode. Workaround: use Buffered mode. • AL Error Code register not available in small register set. EtherCAT conformance test issue for complex slaves with small register set. Workaround: use medium or full register set. <p>Restrictions removed in version 1.1.1:</p> <ul style="list-style-type: none"> • EtherCAT IP Core is incompatible with Microchip I²C EEPROM 24xx16 Workaround: use other EEPROM size or other vendor
1.1.0 (11/2006)	<p>Enhancements:</p> <ul style="list-style-type: none"> • DL Control Register enhanced to 32 Bit • SyncManager: Event Times added, buffer protection enhanced • SPI speed enhanced • Distributed Clocks SyncManager Event Times added • OpenCore Plus support added • Avalon Clock Multiplier available in register 0x0150 • Initialization of User RAM with detailed feature information • GUI shows approx. Les • Added DBC2C20 reference designs <p>Restrictions removed in version 1.1.1:</p> <ul style="list-style-type: none"> • Asynchronous µController cannot access EEPROM Workaround: None. Use different PDI, or do not access EEPROM from async. µC PDI • EtherCAT IP Core is incompatible with Microchip I²C EEPROM 24xx16 Workaround: use other EEPROM size or other vendor
1.1.1 (1/2007)	<p>Enhancements:</p> <ul style="list-style-type: none"> • Support for Avalon Masters with 8/16 Bit • Support for I²C EEPROMS < 16 kBit added <p>Restrictions removed in version 2.0.0:</p> <ul style="list-style-type: none"> • Watchdog counters (0x0442:0x0443) are not available, even in the full register set Workaround: None • SyncManager Changed Flag in AL Event Request register missing (0x0220.4) Workaround: Poll all SyncManager Activation registers

Version	Release notes
2.0.0 (8/2007)	<p>Enhancements:</p> <ul style="list-style-type: none"> • New devices: Cyclone III, Stratix III, Stratix GX, Stratix II GX, Arria GX • Watchdog counters (0x0442:0x0443) are available • SyncManager Changed Flag in AL Event Request register added (0x0220.4) • User can configure detailed IP Core features • MI Link Detection and Configuration • MI controllable by PDI • MI clock speed increased to 2.5 MHz • Enhanced Link Detection • Automatic/manual TX Shift compensation • General Purpose I/O up to 64 bits • Signals EOF, SOF, WD_TRIGGER, and WD_STATE added • Avalon DC Sync interrupts independent of DC SYNC0/1 outputs • SyncManager with 1 Byte length and mailbox mode generates watchdog trigger signal independent of SyncManager reading • PHY address offset 0-31 <p>Restrictions removed in version 2.2.0:</p> <ul style="list-style-type: none"> • MII Management Interface stuck after invalid commands were issued and MI link detection and configuration is not selected Workaround: Make sure to use valid commands, or select MI link detection and configuration
2.2.0 (6/2008)	<p>Enhancements</p> <ul style="list-style-type: none"> • New devices: Stratix IV • Up to 3 MII ports configurable • 32 or 64 Bit Distributed clocks selectable • Reset register (0x0040:0x0041) with RESET_OUT signal selectable • Time limited binary configuration files for evaluation purposes added • Writing DC filter depth registers 0x0934:0x0935 resets filter • DC Activation register 0x0981 and DC Activation Status 0x0984 added • Enhanced Link Detection can be configured port-wise (EEPROM) • PHY Port Status register 0x0518 ff.): new status bit "Port configuration updated" • FB1122 and DBC3C40 reference designs added • Pre-synthesized time-limited configuration files included <p>Restrictions removed in version 2.2.1:</p> <ul style="list-style-type: none"> • ESC Feature register 0x0008.8 is 0 although Enhanced DC SYNC Activation is available Workaround: Ignore 0x0008.8
2.2.1 (6/2009)	<p>Enhancements</p> <ul style="list-style-type: none"> • New devices: Arria II GX • Support for Quartus II 9.0 including SOPC Builder. Previous Quartus II versions are no longer tested, thus, they are not officially supported.

Version	Release notes
2.3.0 (12/2009)	<p>Enhancements</p> <ul style="list-style-type: none"> Support for Quartus II 9.0/9.1 (removed support for previous versions) New devices: Cyclone IV Support for configurations with only one EtherCAT port DEV_STATE LED signal renamed to LED_RUN, to have all LED signals named similar. The IP Core still supports DEV_STATE, but new wrappers will only use LED_RUN. Device ERR LED and STATE(-RUN) LED configurable Direct control of RUN and ERR LED configurable SPI PDI timing improved μC PDI: active write edge configurable PDI Error Code register (0x030E) added, activated with ECAT Processing Unit/PDI Error counter SyncManager buffer status (0x0805[7:6] etc) enhanced: state of read and write buffer available PHY Management interface: preamble suppression supported Forwarding delay has changed <p>Restrictions removed in version 2.3.2:</p> <ul style="list-style-type: none"> DC Timeloop does not react appropriately to System time register write access
2.3.1 (02/2010)	<p>Enhancements</p> <ul style="list-style-type: none"> MII Management Interface now supports PHYs which require an additional clock cycle on MCLK after some operations, and which do not fully support preamble suppression (2 clock cycles of the preamble are now remaining) <p>Restrictions removed in version 2.3.2:</p> <ul style="list-style-type: none"> DC Timeloop does not react appropriately to System time register write access
2.3.2 (03/2010)	<p>Enhancements:</p> <ul style="list-style-type: none"> Distributed Clock Timeloop reacts appropriately to System time register write access
2.04a (03/2011)	<ul style="list-style-type: none"> 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 <p>Enhancements:</p> <ul style="list-style-type: none"> MI link detection and configuration now disables Gigabit Ethernet advertisement Added reference 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

The IP Core version, denoted as X.Y.Z (e.g., 2.4.0), 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	Reserved	r/-	r/-	0

1.7 Design flow

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

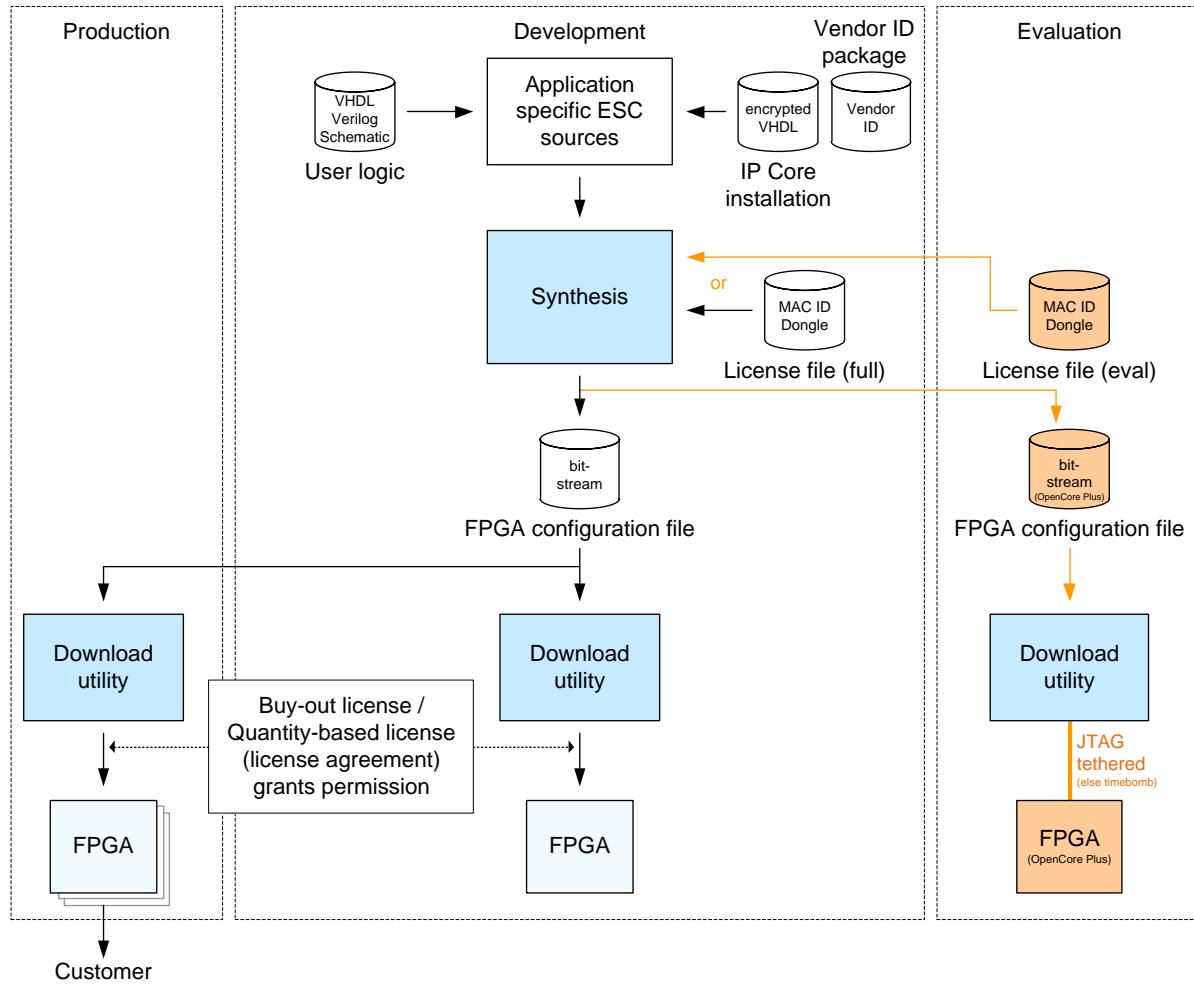


Figure 3: Design flow

1.8 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.

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.

1.9 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. 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 Avalon 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

2 Features and Registers

2.1 Features

Table 7: IP Core Feature Details

Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/V2.3.2	IP Core Altera V2.3.0	Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/V2.3.2	IP Core Altera V2.3.0
EtherCAT Ports	1-3	1-3	1-3	General Ethernet Features (MII/RMII)			
Permanent ports	1-3	1-3	1-3	MII Management Interface (0x0510:0x051F)	c	c	c
Optional Bridge port 3 (EBUS or MII)	-	-	-	Supported PHY Address Offsets	any	any	any
EBUS ports	-	-	-	Link Polarity configurable	User logic	User logic	User logic
MII ports	0/1/2/3	0/1/2/3	0/1/2/3	Enhanced Link Detection	x	x	x
RMII ports	0/1/2	0/1/2	0/1/2	Link detection using PHY signal (LED)	x	x	x
Port 0	x	x	x	MII link status and configuration	c	c	c
Ports 0, 1	x	x	x	MI controllable by PDI (0x0516:0x0517)	x	x	x
Ports 0, 1, 2	x	x	x	MI read error (0x0510.13)	x	x	x
Ports 0, 1, 3	-	-	-	MI PHY configuration update status (0x0518.5)	x	x	x
Ports 0, 1, 2, 3	-	-	-	MI preamble suppression	x	x	x
EtherCAT mode	Direct	Direct	Direct	Additional MCLK	x	x	-
Slave Category	Full Slave	Full Slave	Full Slave	Gigabit PHY configuration	x	-	-
Position addressing	x	x	x	Transparent Mode	-	-	-
Node addressing	x	x	x	MII Features			
Logical addressing	x	x	x	CLK25OUT as PHY clock source	User logic	User logic	User logic
Broadcast addressing	x	x	x	Bootstrap TX Shift settings	c	c	c
Physical Layer General Features				Automatic TX Shift setting (with TX_CLK)	c	c	c
FIFO Size configurable (0x0100[18:16])	x	c	c	TX Shift not necessary (PHY TX_CLK as clock source)	-	-	-
Auto-Forwarder checks CRC and SOF	x	x	x	PDI General Features			
Forwarded RX Error indication, detection and Counter (0x0308:0x030B)	x	x	x	Extended PDI Configuration (0x0152:0x0153)	x	x	x
Lost Link Counter (0x0310:0x0313)	c	c	c	PDI Error Counter (0x030D)	c	c	c
Prevention of circulating frames	x	x	x	PDI Error Code (0x030E)	c	c	x
Fallback: Port 0 opens if all ports are closed	x	x	x	CPU_CLK output (10, 20, 25 MHz)	User logic	User logic	User logic
VLAN Tag and IP/UDP support	x	x	x				
Enhanced Link Detection per port configurable	x	x	x				

Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0	Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0
SOF, EOF, WD_TRIG and WD_STATE independent of PDI	x	x	x	Data out sample mode configurable (0x0150.5)	x	x	x
Available PDIs and PDI features depending on port configuration	-	-	-	Busy signaling	-	-	-
PDI selection at run-time (SII EEPROM)	-	-	-	Wait State byte(s)	x	x	x
PDI active immediately (SII EEPROM settings ignored)	x	x	x	Number of address extension byte(s)	any	any	any
Digital I/O PDI	x	x	x	2/4 Byte SPI master support	x	x	x
Digital I/O width [bits]	8/16/24/ 32	8/16/24/ 32	8/16/24/ 32	Extended error detection (read busy violation)	x	x	x
PDI Control register value (0x0140:0x0141)	4	4	4	SPI_IRQ delay	x	x	x
Control/Status signals:	7	7	7	Status indication	x	x	x
LATCH_IN	x	x	x	EEPROM_Loaded signal	-	-	-
SOF	x	x	x	8/16 bit asynchronous μController PDI	x	x	x
OUTVALID	x	x	x	Extended μC configuration bits 0x0150[7:4], 0x0152:0x0153	x	x	x
WD_TRIG	x	x	x	ADR[15:13] available (000 _b if not available)	x	x	x
OE_CONF	-	-	-	EEPROM_Loaded signal	-	-	-
OE_EXT	x	x	x	RD polarity configurable (0x0150.7)	-	-	-
EEPROM_Loaded	-	-	-	Read BUSY delay (0x0152.0)	x	x	x
WD_STATE	x	x	x	Write after first edge (0x0152.2)	x	x	x
EOF	x	x	x	8/16 bit synchronous μController PDI	-	-	-
Granularity of direction configuration [bits]	8	8	8	On-Chip Bus PDI	x	x	x
Bidirectional mode	- (User logic)	- (User logic)	- (User logic)	Avalon®	x	x	x
Output high-Z if WD expired	User logic	User logic	User logic	OPB	-	-	-
Output 0 if WD expired	x	x	x	PLB v4.6	-	-	-
Output with EOF	x	x	x	Bus clock [MHz] (N=1,2,3,...)	N*25	N*25	N*25
Output with DC SyncSignals	x	x	x	Master bus width (prefetched bytes) [Bytes]	1/2/4	1/2/4	1/2/4
Input with SOF	x	x	x	DC SyncSignals available directly and as IRQ	x	x	x
Input with DC SyncSignals	x	x	x	Bus clock multiplier in register 0x0150[6:0]	x	x	x
SPI Slave PDI	x	x	x	EtherCAT Bridge (port 3, EBUS/MII)	-	-	-
Max. SPI clock [MHz]	30	30	30				
SPI modes configurable (0x0150[1:0])	x	x	x				
SPI_SEL polarity configurable (0x0150.4)	x	x	x				

Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0	Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0
General Purpose I/O	x	x	x	AL Status Emulation (0x0140.8)	x	x	x
GPO bits	0/8/16/ 32/64	0/8/16/ 32/64	0/8/16/ 32/64	AL Status Code (0x0134:0x0135)	c	c	c
GPI bits	0/8/16/ 32/64	0/8/16/ 32/64	0/8/16/ 32/64	Interrupts			
GPIO available independent of PDI or port configuration	x	x	x	ECAT Event Mask (0x0200:0x0201)	x	c	c
Concurrent access to GPO by ECAT and PDI	x	x	x	AL Event Mask (0x0204:0x0207)	c	c	c
ESC Information				ECAT Event Request (0x0210:0x0211)	x	x	x
Basic Information (0x0000:0x0006)	x	x	x	AL Event Request (0x0220:0x0223)	x	x	x
Port Descriptor (0x0007)	x	x	x	SyncManager activation changed (0x0220.4)	x	x	x
ESC Features supported (0x0008:0x0009)	x	x	x	SyncManager watchdog expiration (0x0220.6)	x	x	x
Extended IP Core Configuration in User RAM (0x0F80 ff.)	x	x	x	Error Counters			
Write Protection (0x0020:0x0031)	c	c	c	RX Error Counter (0x0300:0x0307)	x	x	x
Data Link Layer Features				Forwarded RX Error Counter (0x0308:0x030B)	x	x	x
ECAT Reset (0x0040)	c	c	c	ECAT Processing Unit Error Counter (0x030C)	c	c	c
PDI Reset (0x0041)	c	c	c	PDI Error Counter (0x030D)	c	c	c
ESC DL Control (0x0100:0x0103) bytes	4	2/4	2/4	Lost Link Counter (0x0310:0x0313)	c	c	c
EtherCAT only mode (0x0100.0)	x	x	x	Watchdog			
Temporary loop control (0x0100.1)	x	x	x	Watchdog Divider configurable (0x0400:0x0401)	c	c	c
FIFO Size configurable (0x0100[18:16])	x	c	c	Watchdog Process Data	x	x	x
Configured Station Address (0x0010:0x0011)	x	x	x	Watchdog PDI	x	x	x
Configured Station Alias (0x0100.24, 0x0012:0x0013)	x	c	c	Watchdog Counter Process Data (0x0442)	x	x	x
Physical Read/Write Offset (0x0108:0x0109)	c	c	c	Watchdog Counter PDI (0x0443)	x	x	x
Application Layer Features				SII EEPROM Interface (0x0500:0x050F)			
Extended AL Control/Status bits (0x0120[15:5], 0x0130[15:5])	x	-	-	EEPROM sizes supported	1 KB-4 Mbyte	1 KB-4 Mbyte	1 KB-4 Mbyte
				EEPROM size reflected in 0x0502.7	x	x	x

Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0	Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0
EEPROM controllable by PDI	x	x	x	SyncSignal Late Activation (0x0981[6:5])	x	x	x
EEPROM Emulation by PDI	c	c	c	SyncSignal debug pulse (0x0981.7)	x	x	x
Read data bytes (0x0502.6)	4	4	4	SyncSignal Activation State 0x0984)	x	x	x
Internal Pull-Ups for EEPROM_CLK and EEPROM_DATA	User logic	User logic	User logic	Reset filters after writing filter depth	x	x	x
FMMUs	0-8	0-8	0-8	ESC Specific Registers (0x0E00:0x0EFF)			
Bit-oriented operation	x	x	x	Product and Vendor ID	x	x	x
SyncManagers	0-8	0-8	0-8	POR Values	-	-	-
Watchdog trigger generation for 1 Byte Mailbox configuration independent of reading access	x	x	x	FPGA Update (online)	-	-	-
SyncManager Event Times (+0x8[7:6])	c	c	c	Process RAM and User RAM			
Buffer state (+0x5[7:6])	x	x	x	Process RAM (0x1000 ff.) [KByte]	1-60	1-60	1-60
Distributed Clocks	c	c	c	User RAM (0x0F80:0x0FFF)	x	x	x
Width	32/64	32/64	32/64	Extended IP Core Configuration in User RAM	x	x	x
Sync/Latch signals	4 (2 Sync-Signals, 2 Latch-Signals)	4 (2 Sync-Signals, 2 Latch-Signals)	4 (2 Sync-Signals, 2 Latch-Signals)	Additional EEPROMs	1-2	1-2	1-2
SyncManager Event Times (0x09F0:0x09FF)	c	c	c	SII EEPROM (I ² C)	c (EEPRO M of μC used)	c (EEPRO M of μC used)	c (EEPRO M of μC used)
DC Receive Times	c	c	c	FPGA configuration EEPROM	x	x	x
DC Time Loop Control controllable by PDI	c	c	c	LED Signals			
DC activation by EEPROM (0x0140[11:10])	-	-	-	RUN LED	c	c	c
Propagation delay measurement with traffic (BWR/FPWR 0x900 detected at each port)	x	x	x	RUN LED override	c	c	c
LatchSignal state in Latch Status register (0x09AE:0x09AF)	x	x	x	Link/Activity(x) LED per port	x	x	x
SyncSignal Auto-Activation (0x0981.3)	x	x	x	PERR(x) LED per port	-	-	-
SyncSignal 32 or 64 bit Start Time (0x0981.4)	x	x	x	Device ERR LED	c	c	c
				STATE_RUN LED	c	c	c
				Clock supply			
				Crystal	-	-	-
				Crystal oscillator	x	x	x
				TX_CLK from PHY	x	x	x
				25ppm clock source accuracy	x	x	x
				Internal PLL	User logic	User logic	User logic

Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0	Feature	IP Core Altera® V2.4.0	IP Core Altera V2.3.1/ V2.3.2	IP Core Altera V2.3.0
Power Supply Voltages	FPGA dep.	FPGA dep.	FPGA dep.	Reference designs/ pre-synthesized time- limited evaluation core included	5/3	7/3	7/3
I/O Voltage	FPGA dep.	FPGA dep.	FPGA dep.	FB1120 Digital I/O	-	x/-	x/-
3.3 V	FPGA dep.	FPGA dep.	FPGA dep.	FB1120 SPI	-	x/-	x/-
3.3V / 5V tolerant	FPGA dep.	FPGA dep.	FPGA dep.	FB1122 Digital I/O	x/x	x/x	x/x
5 V	FPGA dep.	FPGA dep.	FPGA dep.	FB1122 SPI	x/-	-	-
Core Voltage	FPGA dep.	FPGA dep.	FPGA dep.	DBC2C20 Digital I/O	-	x/x	x/x
Internal LDOs	-	-	-	DBC2C20 NIOS®	-	x/-	x/-
Package	FPGA dep.	FPGA dep.	FPGA dep.	DBC3C40 Digital I/O	x/x	x/x	x/x
Size [mm²]	FPGA dep.	FPGA dep.	FPGA dep.	DBC3C40 NIOS	-	x/-	x/-
Release date	3/2011	2/2010	12/2009	DBC4CE55 NIOS	x/x	-	-
Configuration and Pinout calculator (XLS)	-	-	-	DE2-115 NIOS	x/x	-	-
Register Configuration	individual	individual	individual				
Complete IP Core evaluation	x	x	x				

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 (0x0000:0xFFFF) 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 ¹	Length (Byte)	Description	IP Core V2.4.0			IP Core V2.3.0-V2.3.2			IP Core V2.2.1/V2.2.0		
			Register set			Register set			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

¹ Registers not listed here are reserved. They are not writable. A read access to reserved registers receives 0 as return value.

Address ¹	Length (Byte)	Description	IP Core V2.4.0			IP Core V2.3.0-V2.3.2			IP Core V2.2.1/V2.2.0		
			Register set			Register set			Register set		
S	M	L	S	M	L	S	M	L	S	M	L
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
0x0141	1	ESC Configuration	X	X	X	X	X	X	X	X	X
0x0142:0x0143	2	Extended ESC Configuration	-	-	-	-	-	-	-	-	-
0x0150:0x0153	4	PDI 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	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 ¹	Length (Byte)	Description	IP Core V2.4.0			IP Core V2.3.0-V2.3.2			IP Core V2.2.1/V2.2.0		
			Register set			Register set			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
0x0980	1	DC – Cyclic Unit Control	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x0981:0x0983	3	DC – SYNC Out Unit	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
0x09A8:0x09A9 0x09AE:0x09CF	36	DC – Latch In Unit	dc	dc	dc	dc	dc	dc	dc	dc	dc
0x09F0:0x09F3 0x09F8:0x09FF	12	DC – SyncManager Event Times	c	c	dc	c	c	dc	c	c	dc
0xE00:0xEFF	256	ESC specific registers (e.g., Power-On Values / Product and Vendor ID)	x	x	x	x	x	x	x	x	x
0xF00:0xF03	4	Digital I/O Output Data	io	io	io	io	io	io	io	io	io
0xF10:0xF17	8	General Purpose Outputs [Byte]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0xF18:0xF1F	8	General Purpose Inputs [Byte]	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8	0-8
0xF80:0xFFFF	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

Table 10: Legend

Symbol	Description
x	Available
-	Not available
r	Read only
c	Configurable
dc	Available if Distributed Clocks 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 – 0x0F80:0xFFFF)

Bit	Description	small	medium	large
7:0	Number of extended feature bits		51	
	IP Core extended features:	0: Not available 1: Available c: Configurable		
8	Extended DL Control Register (0x0102:0x0103)	c	c	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)	c	c	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 Requirements

System Requirements

The system requirements of Altera Quartus II 10.1 are applicable²:

- Microsoft Windows XP/Vista/Windows7 (32 and 64 bit)
- Red Hat Enterprise Linux 4.0 and 5.0 (32 and 64 bit)
- SUSE Linux Enterprise 9 and 10 (32 and 64 bit)
- CentOS 4.0 and 5.0 (32 and 64 bit)

Program Requirements

For synthesis of the EtherCAT IP Core for Altera FPGAs the following Software is needed:

- Altera Quartus II Version 10.1 or higher. A free version ("Web Edition") is available from Altera (<http://www.altera.com>).
- EtherCAT IP Core for Altera FPGAs

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

- Altera Nios II Embedded Design Suite 10.1 or higher

² Not all of these variants have been tested with the EtherCAT IP core.

3.2 Installation on Windows PCs

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 and documentation are typically installed in the directory
`<Quartus installation folder>\ip\beckhoff\ethercat_<version>`
otherwise Quartus II is not able to locate the EtherCAT IP (the folder `ethercat-<version>`, which is
automatically created inside this directory, is further referenced to as `<IPInst_dir>`).

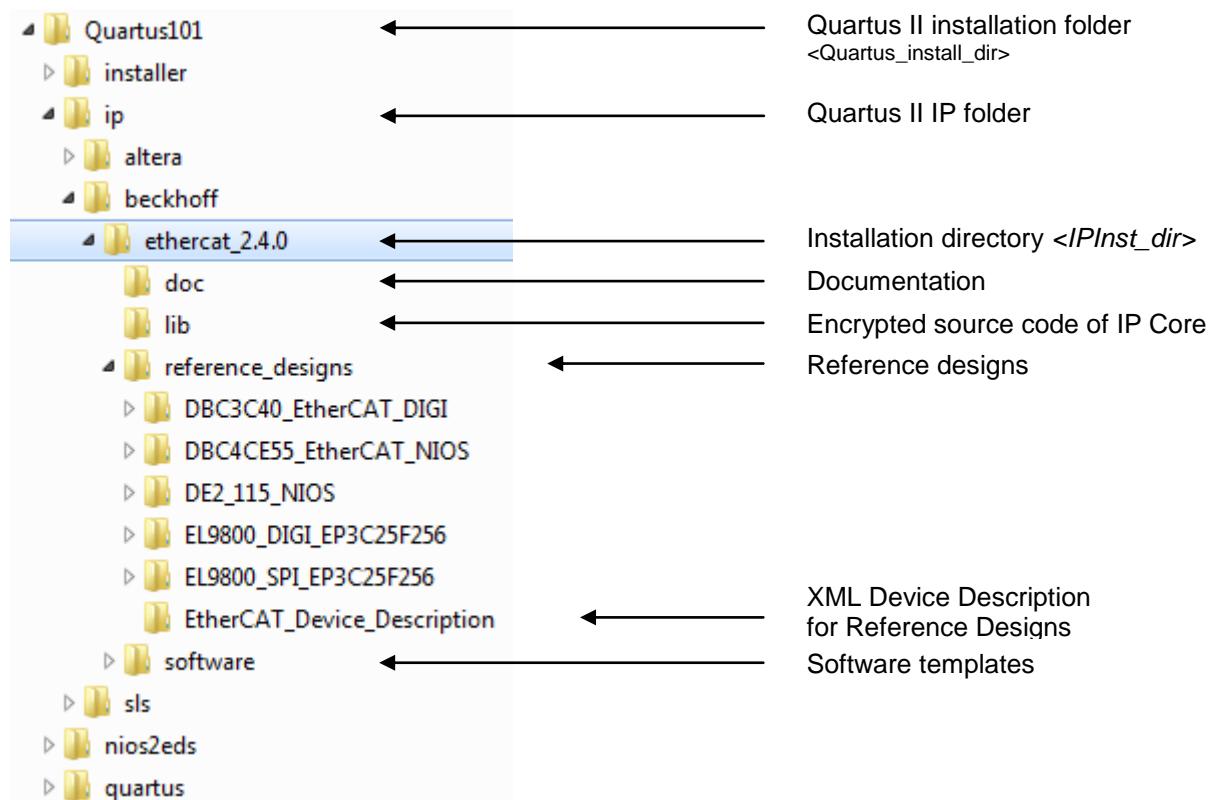


Figure 4: Files installed with EtherCAT IP Core setup

3.3 Installation on Linux PCs

For installation of the EtherCAT IP Core extract the archive to the *ip* folder of your Altera installation on your Linux PC:

1. Change to Altera Quartus II installation directory (e.g. if /opt/altera/10.1 is your Altera installation folder):

```
# cd /opt/altera/10.1/ip
```
2. Extract the EtherCAT IP Core:

```
# tar -xf ethercat-<version>.linux.tar.gz
```

The folder *ethercat_<version>* created inside this directory is further referenced to as *<IPInst_dir>*.

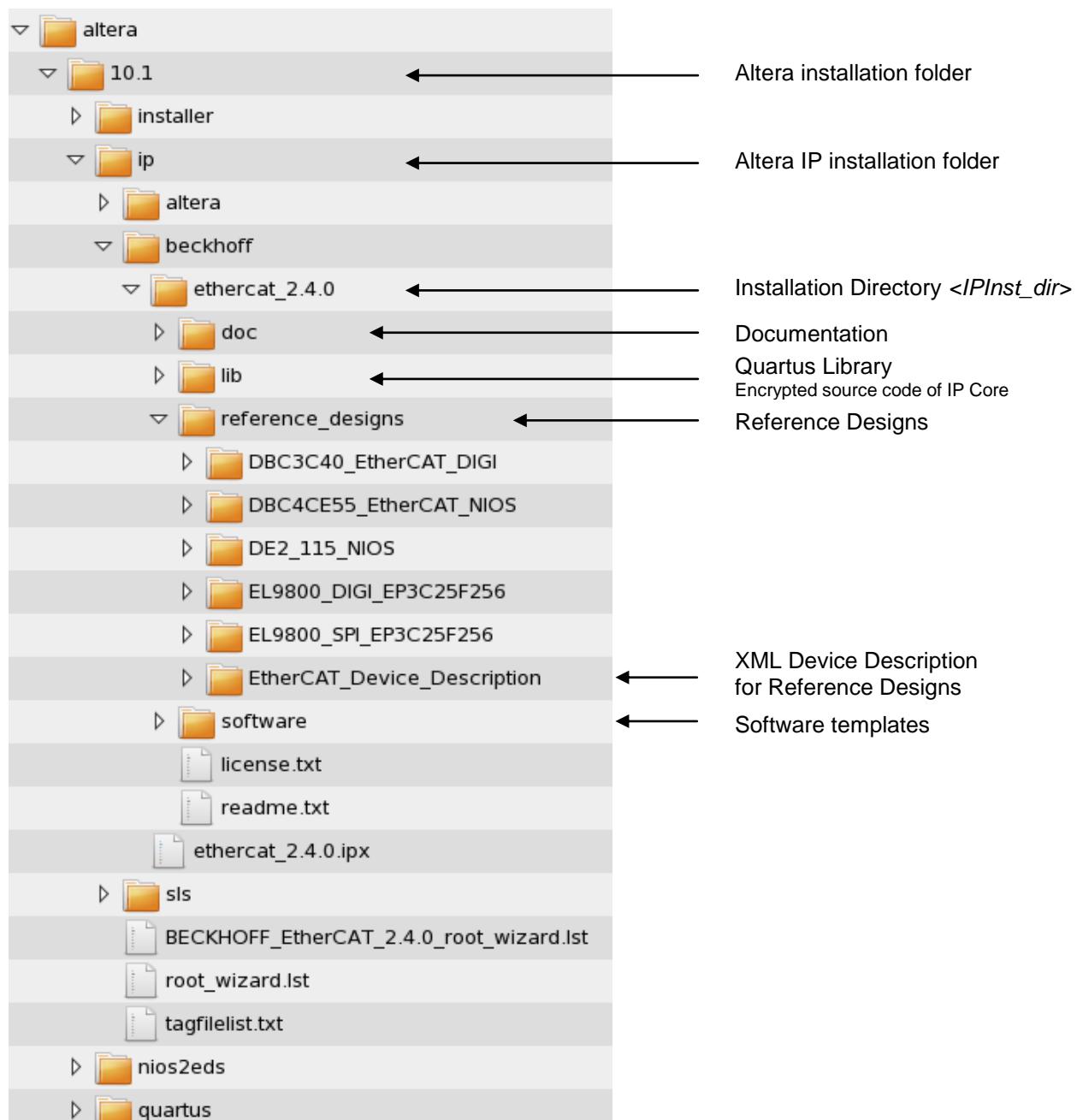


Figure 5: Files installed with ethercat-vX.Y.Z

The EtherCAT IP Core Plug-In is now available in the Megawizard Plugin Manager and in the SOPC Builder System Resources.

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:

1. Choose Tools on the menu bar → License Setup.
2. Choose General → License Setup on the left hand tree structure

There are two options:

- a) If you have selected a license file in the Quartus License Setup dialog, you have to add the content of the EtherCAT IP Core license file to this file.
- b) The LM_LICENSE_FILE variable is used. You can either add the content of the EtherCAT IP Core license file to the Altera license file specified by the LM_LICENSE_FILE variable, or you can add the path to the EtherCAT IP Core license file to this variable.

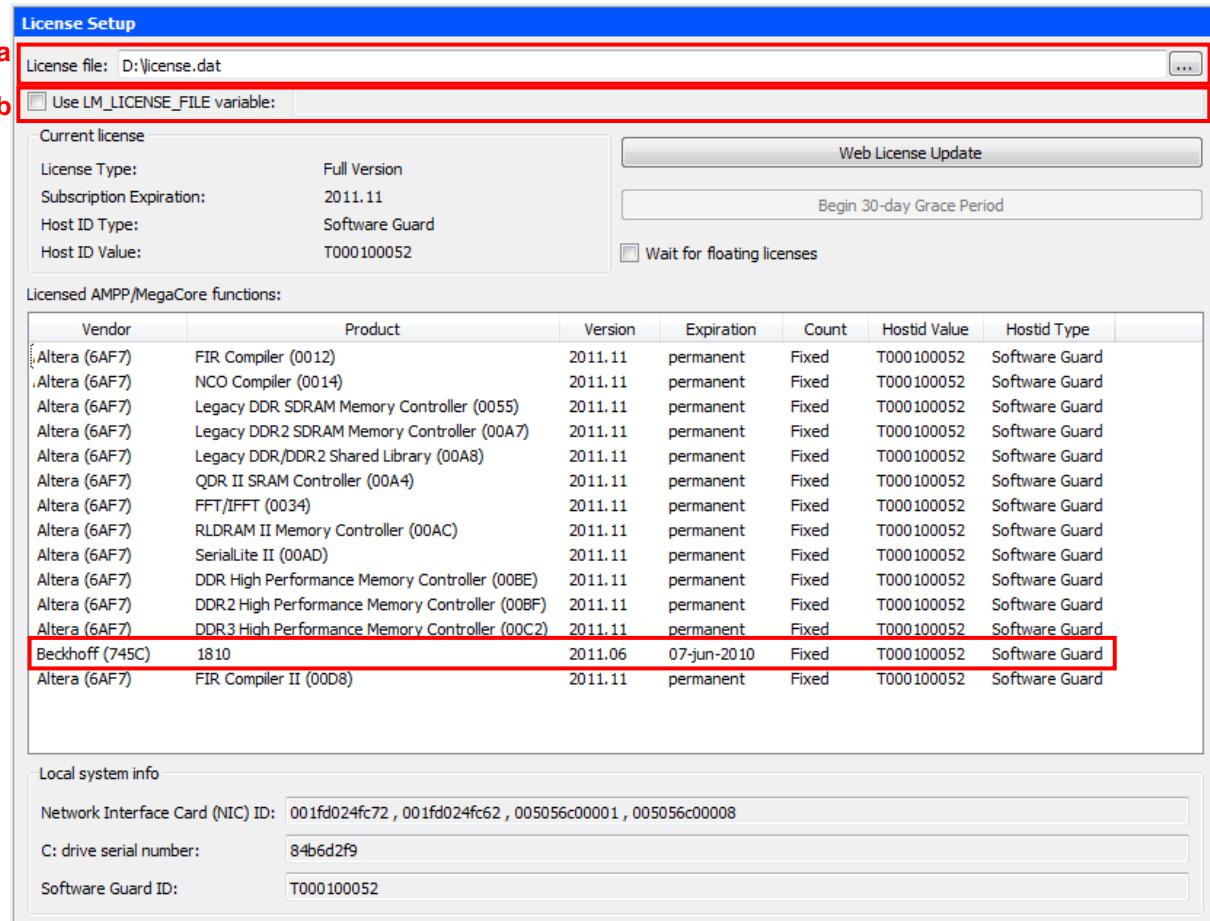


Figure 6: License Setup

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>\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.

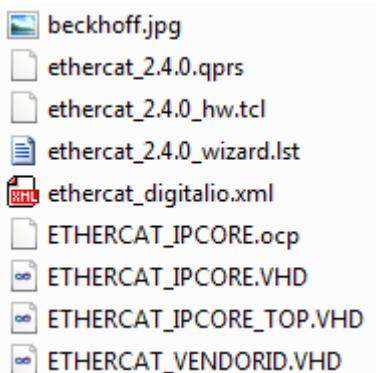


Figure 7: <IPInst_dir>\lib folder with vendor ID package (ETHERCAT_VENDORID.VHD)

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 Software Templates for reference designs with NIOS processor

Software example templates are available for reference designs with NIOS processor. The templates have to be copied to your NIOS II installation folder.

Copy everything inside the templates folder

`<IPInst_Dir>\software`

to your NIOS II installation folder

`<Quartus_Install_Dir>\nios2eds\examples\software`

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

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

The ESI is located at

`<IPInst_dir>\reference_designs\EtherCAT_Device_Description\BECKHOFF ET1810.xml`

If you are using TwinCAT, add the ESI to the appropriate folder of your TwinCAT installation (`<TwinCAT installation folder>\lo\EtherCAT`) before the System Manager is started.

3.8 Quartus II libraries

The project libraries or global libraries of Altera Quartus II are no longer used by Quartus II to locate MegaWizard IP sources.

4 IP Core Usage

4.1 MegaWizard Plug-In Manager

This chapter explains how to configure your own EtherCAT IP Core using the Quartus II MegaWizard Plug-In Manager. The EtherCAT MegaWizard Plug-In is used for configuration of the EtherCAT IP Core. The output of the Plug-In is a VHDL or Verilog wrapper for the EtherCAT IP Core library file. The wrapper file makes only those interfaces visible which were selected by the user, and it configures the EtherCAT IP Core using generics as desired. The EtherCAT IP Core library file contains the encrypted source code with the EtherCAT functionality.

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

4.1.1 Implementing the EtherCAT IP Core

1. Open Altera Quartus II
 2. Choose Tools on the menu bar and select *MegaWizard Plug-In Manager...* On the dialog choose "Create a new custom megafunction variation" and press Next.
- a) Choose the device family from the pull down menu on the upper right corner.
 - b) Select the BECKHOFF/ EtherCAT megafunction from megafunction tree.
 - c) Type of output file:
Depending on the hardware description language which might be used to implement extra vendor specific logic, the output type VHDL or Verilog HDL can be chosen.
 - d) Choose output directory and name of the output file.
 - e) Press Next

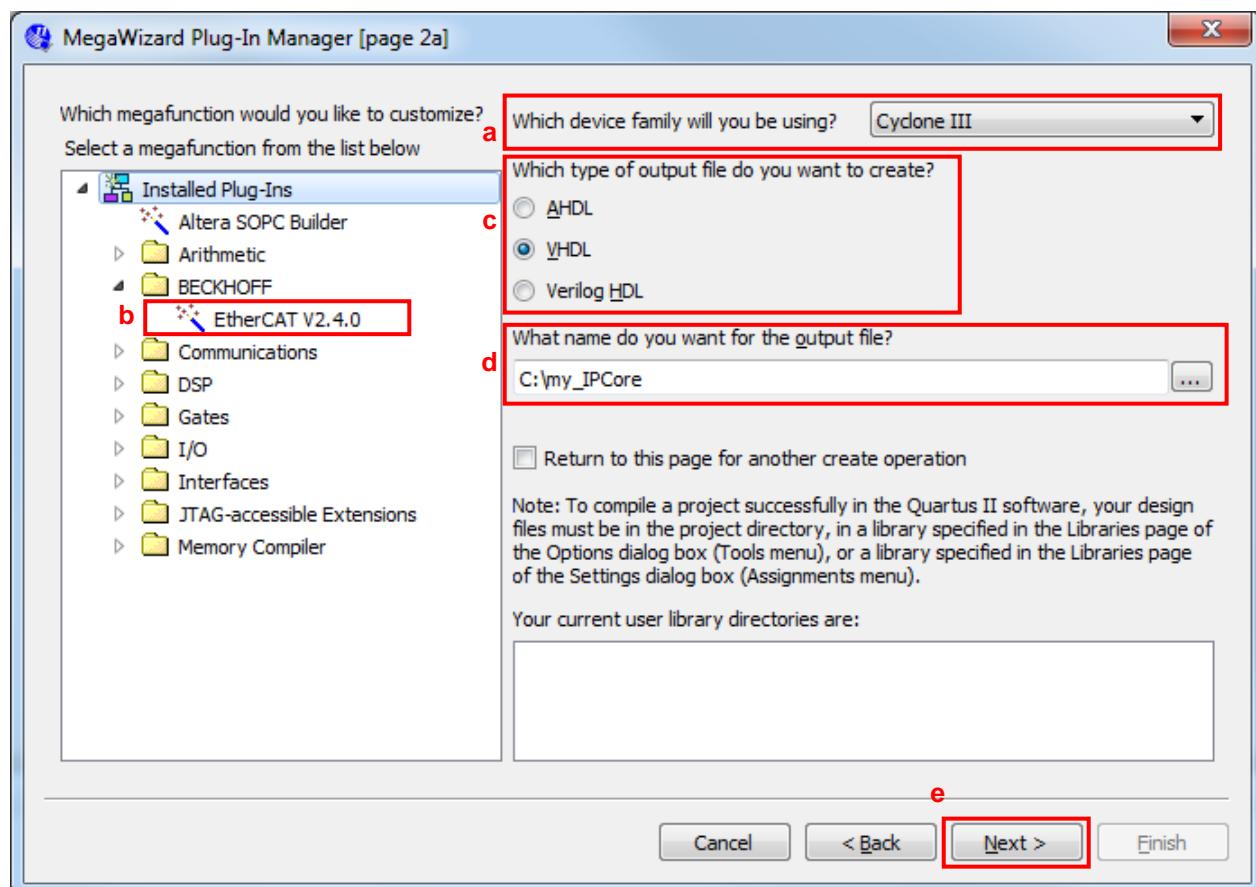


Figure 8: MegaWizard Plug-In Manager

4.1.2 EtherCAT IP Core Configuration Interface

Step 1: Parameterize EtherCAT Core

See chapter 5 for Parameterization options.

Step 2: Finish

After finishing the EtherCAT Core configuration the core can be built by pressing the “Finish” button.
The following files are created:

Table 12: Generated files description

File	Description
<name>.vhd	Defines VHDL top level description
<name>.v	Defines Verilog top level description
<name>.bsf	Quartus II symbol file for the Megafunction variation. You can use this file in the Quartus II schematic editor.
<name>.xml	Megafunction configuration options.
<name>.qip	Quartus II IP file
<name>\	Source code folder, IP Core files from installation folder are copied into this folder

NOTE: The MegaWizard adds default values for MegaFunction inputs. Take care that all signals of the EtherCAT IP Core are connected, unconnected inputs will not be reported as errors.

4.2 SOPC Builder

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, the Altera Quartus II SOPC Builder is used (Figure 9). 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 9). 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.

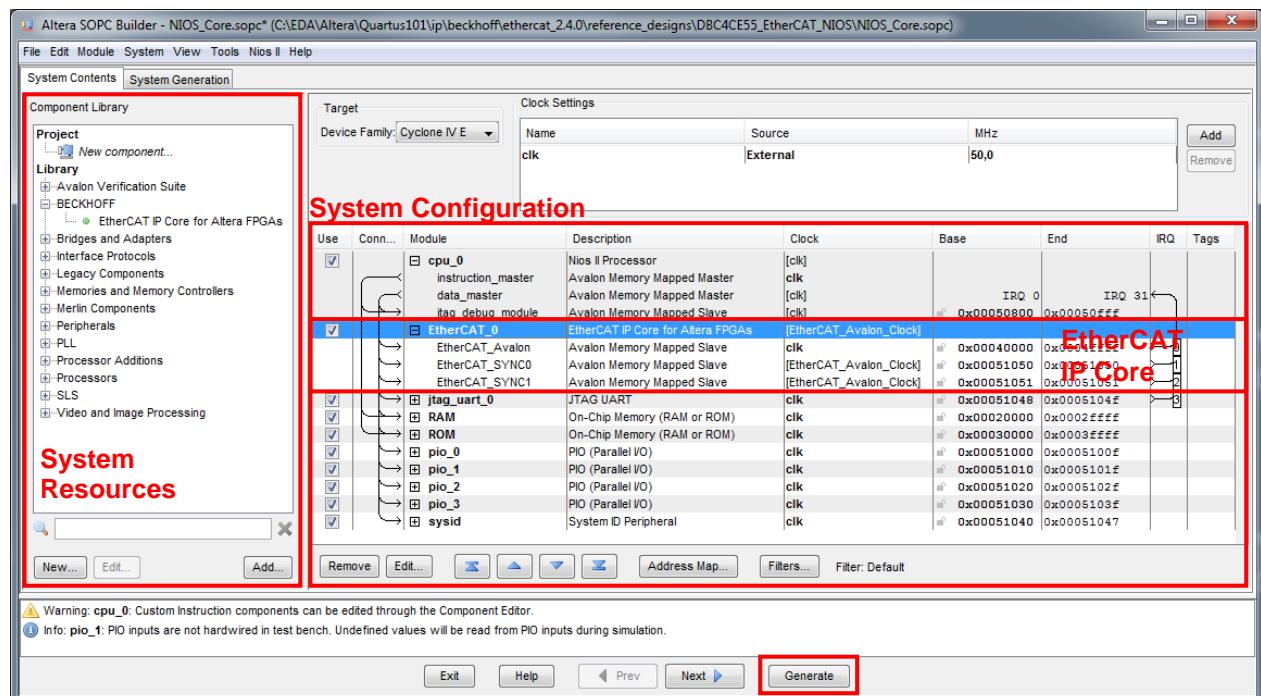


Figure 9: SOPC Builder

NOTE: Push “Filter...” button and check “Interrupt” to view all interrupt connections (column IRQ).

Follow these steps for adding the EtherCAT IP Core to a new SOPC Builder project:

1. Open Altera Quartus II
2. Create a new project
3. Choose Tools on the menu bar and select SOPC Builder...
4. Enter New System Name
5. Double-click on the "Library / BECKHOFF / EtherCAT IP Core for Altera FPGAs" entry in the Component Library, this will add the EtherCAT IP to the system and open the configuration dialog.
6. Configure the EtherCAT IP core
7. Press "Finish" to update the new configuration.

5 IP Core Configuration

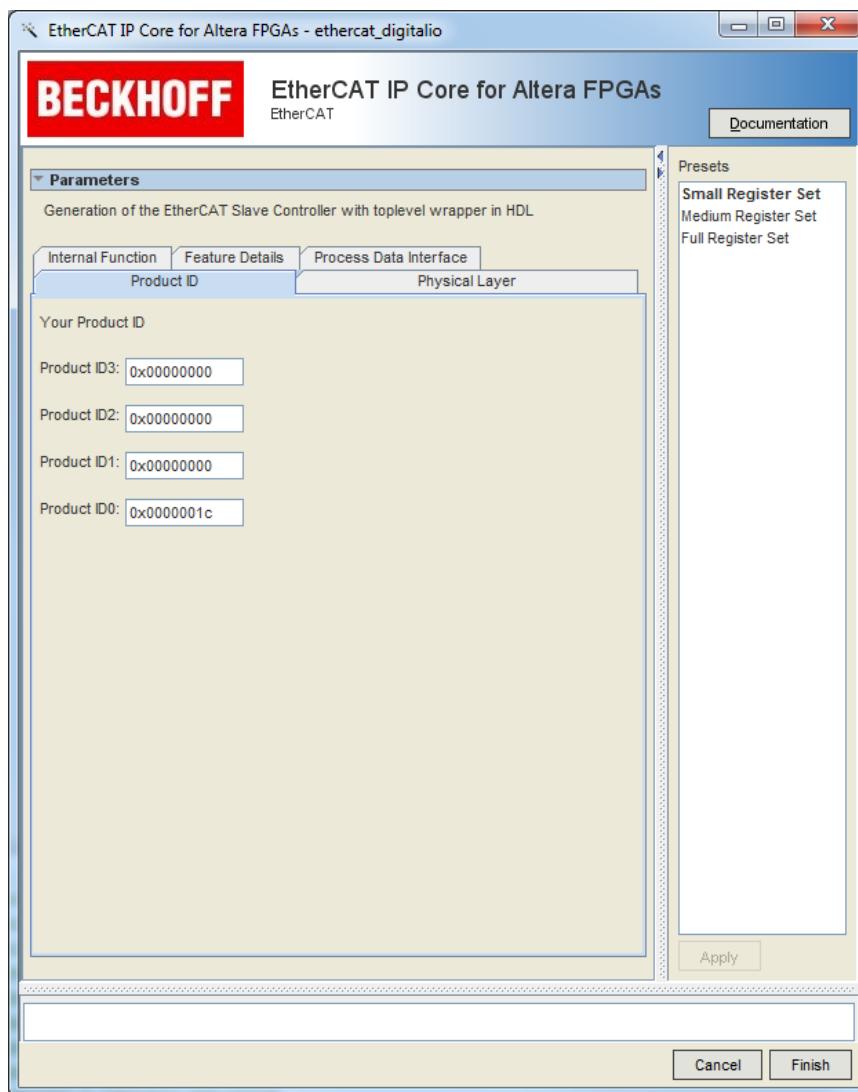


Figure 10: EtherCAT IP Core Configuration Interface

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

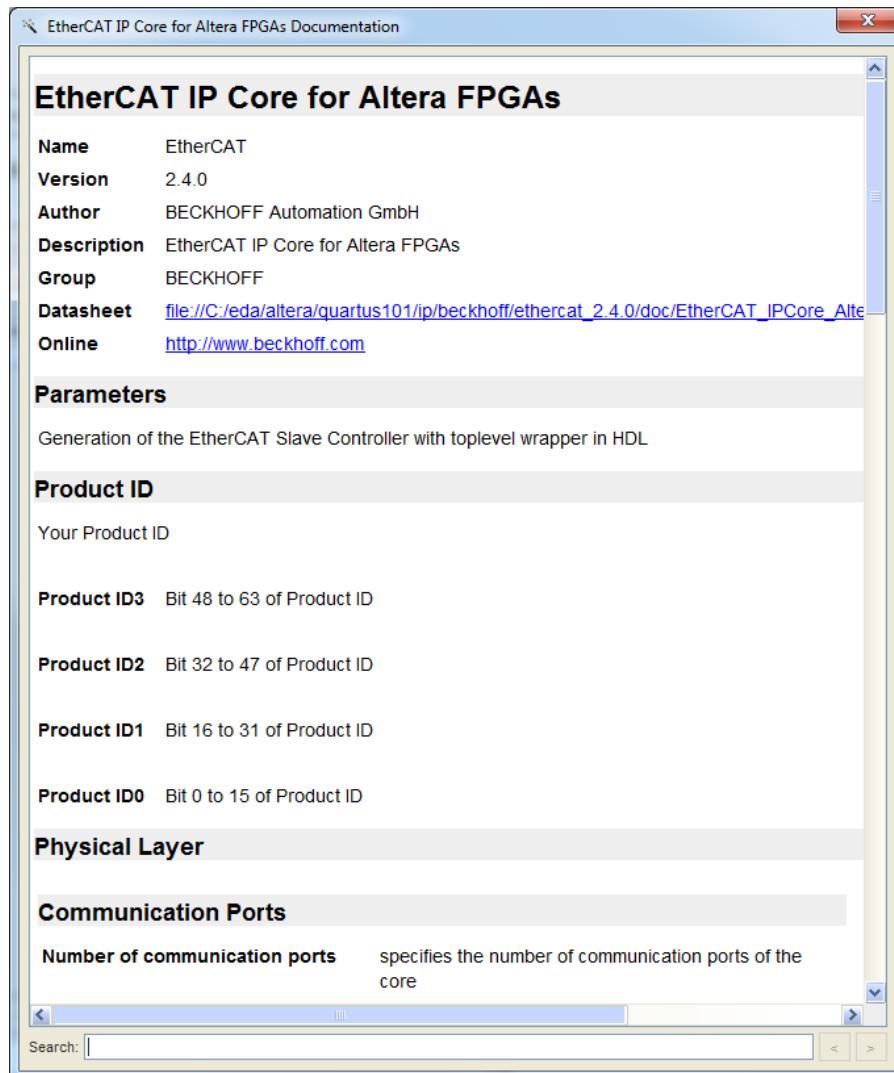


Figure 11: Documentation

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

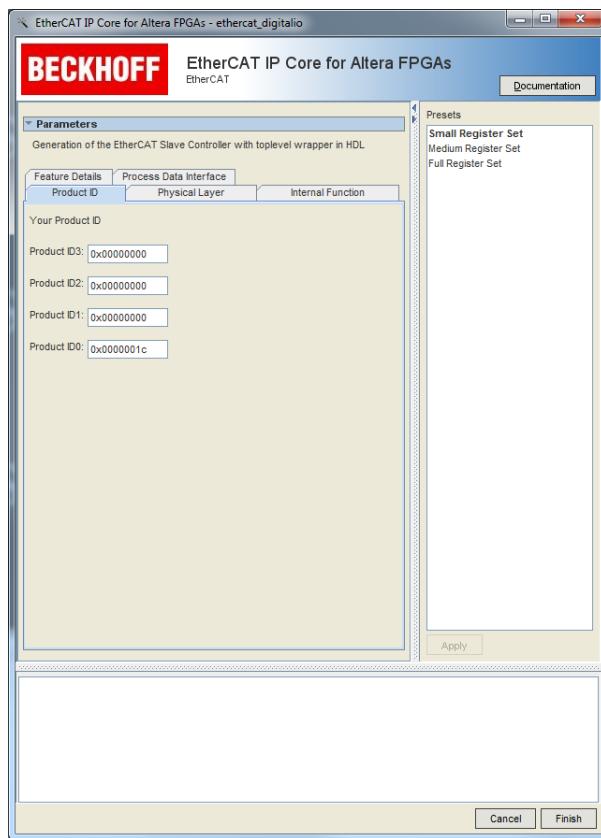


Figure 12: Product ID tab

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 0xE08:0xE0F.

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

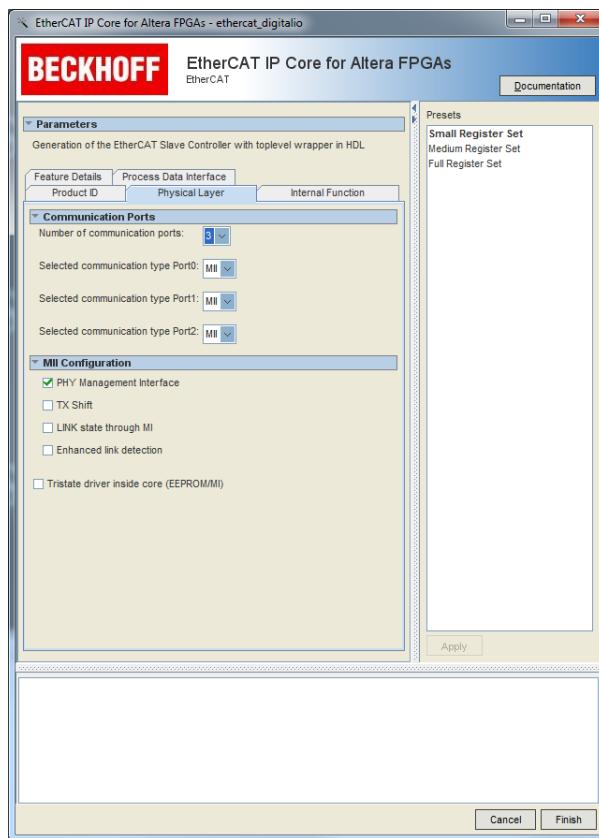


Figure 13: Physical Layer tab

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 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

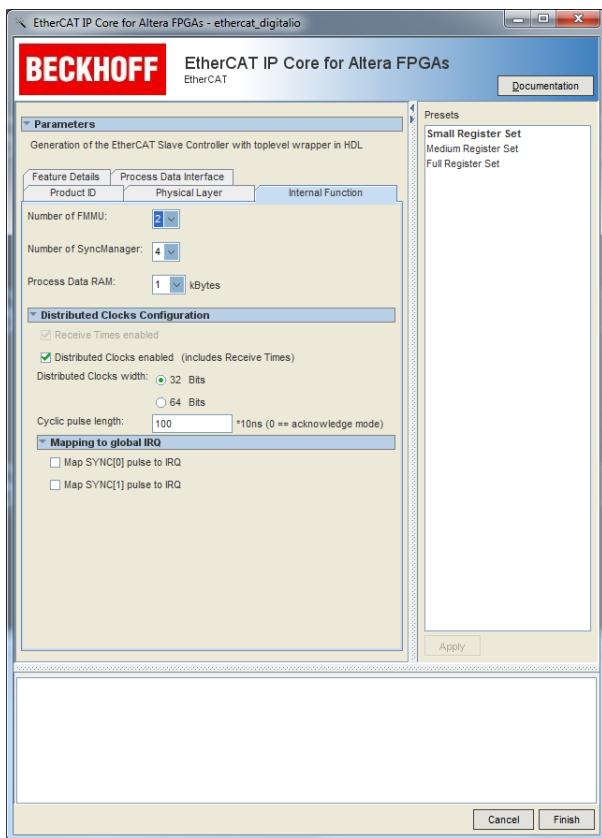


Figure 14: Internal Functions tab

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

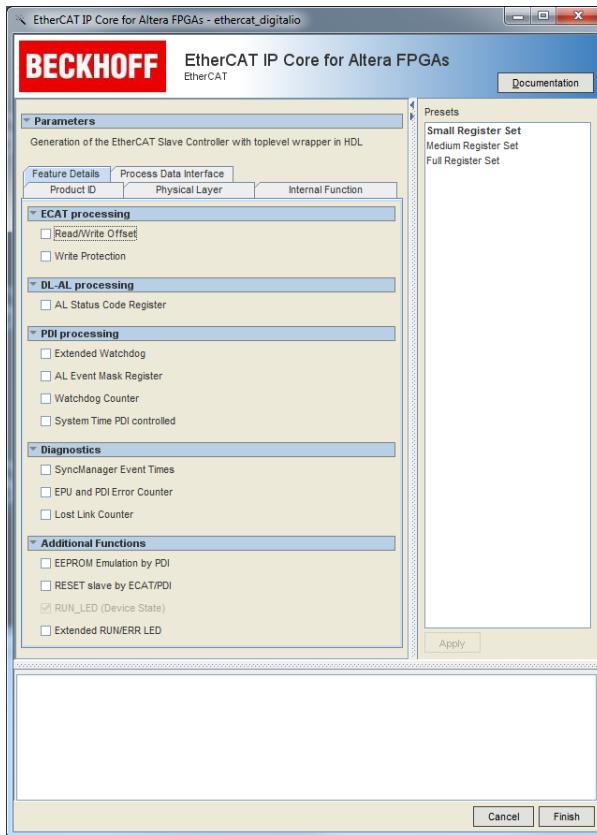


Figure 15: Feature Details tab

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 (μ Controller) 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 (Device State)

RUN LED output 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
- General Purpose I/O

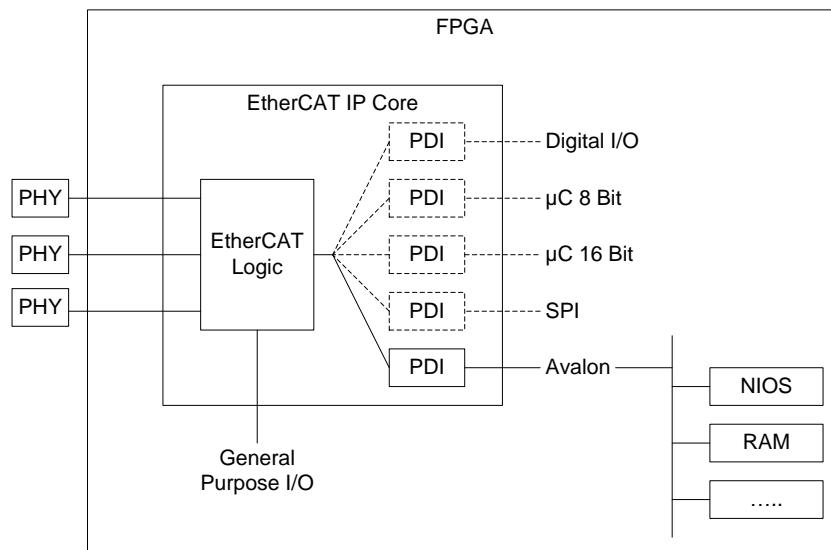


Figure 16: Available PDI Interfaces

The PDI can be selected from the pull down menu. After selection settings for the selected PDI are shown and can be changed.

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).

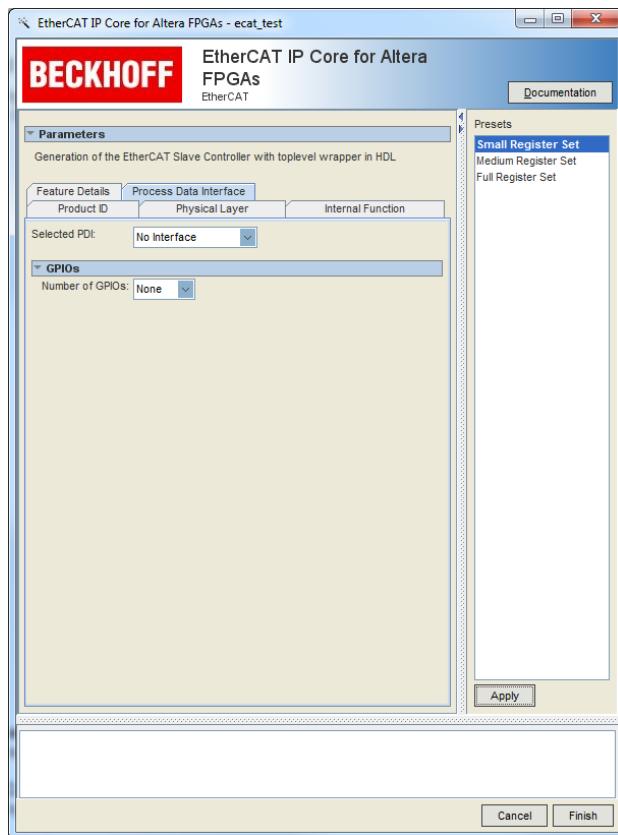


Figure 17: Register Process Data Interface

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.

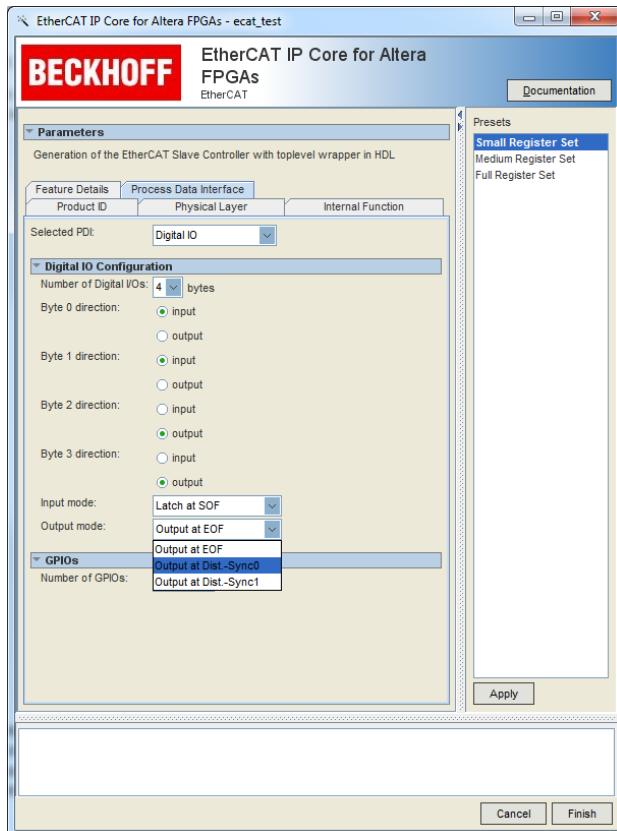


Figure 18: Register PDI – Digital I/O Configuration

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.

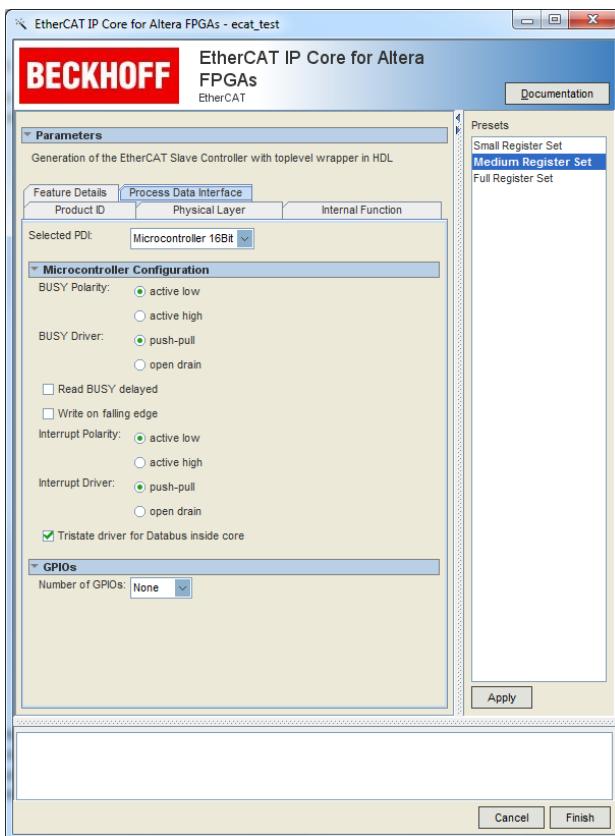


Figure 19: Register PDI – µC-Configuration

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.

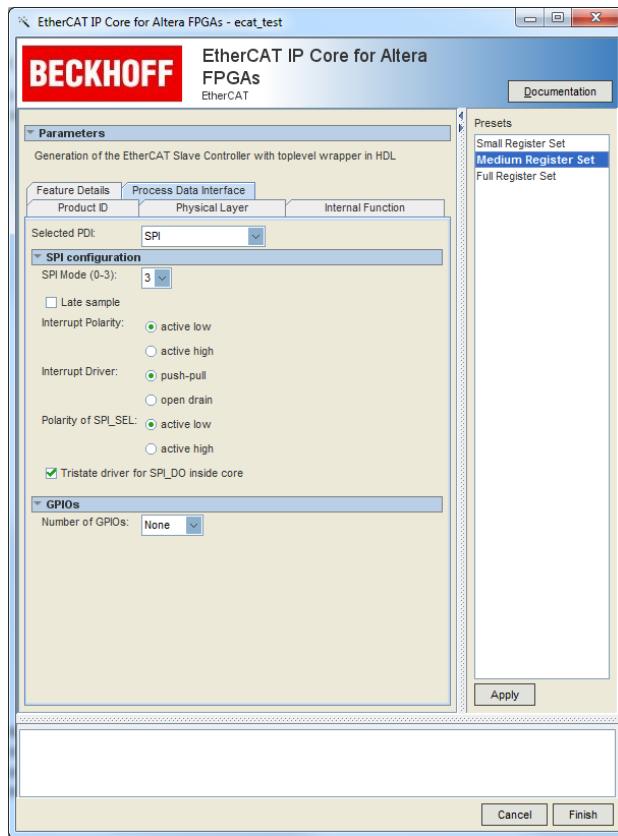


Figure 20: Register PDI – SPI Configuration

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 width is 8 bit, the address bus is 16 bit wide.

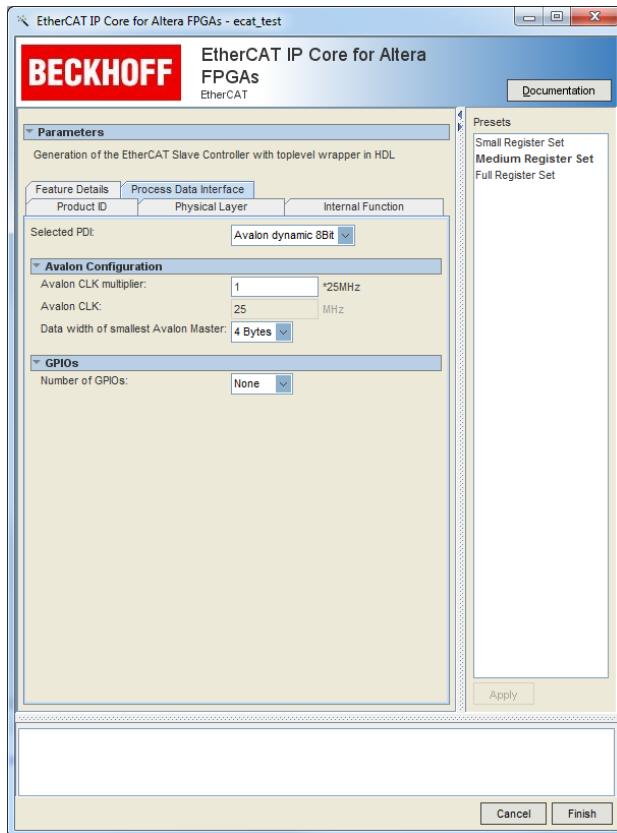


Figure 21: Register PDI – Avalon Interface Configuration

Avalon Clock Multiplier

Avalon Clock Multiplier ($n \times 25\text{MHz}$) 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.

6 Reference Designs

Reference designs are available for:

- Beckhoff Evaluation Board EL9800/FB1122 with MII and 16 bit input/16 bit output Digital I/O
- Beckhoff Evaluation Board EL9800/FB1122 with MII and SPI
- 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 reference designs and its schema can be found in the installation directory.

`<IPInst_dir>\reference_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 reference 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 Beckhoff EL9800/FB1122 with Digital I/O

6.1.1 Configuration and resource consumption

Table 13: Resource consumption Digital I/O reference design EL9800/FB1122

Configuration		Resources		EP3C25
FMMU	2	LEs	11,790	48 %
SyncManager	4	Registers	6,265	25 %
RAM	1 KB	Pins	90	57 %
Register set	Small + MI + TX Shift	M9K	3	5 %
Distributed Clocks	32 bit	PLLs	1	50 %
PDI	2 Byte IN, 2 Byte OUT, SOF/EOF	Multiplier elements	0	0 %

6.1.2 Functionality

Configure PDI Selector S200 of the EL9800 Evaluation base board to PDI4: 16IN/16OUT.

Functionality of the Digital I/O reference design:

- Digital input data from PORT A and PORT B switches is available in the Process Data RAM (0x1000:0x1001).
- Digital output data from Digital Output register (0x0F00:0x0F01) is visualized with PORT C and PORT D LEDs.

6.1.3 Implementation

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

1. Open Altera Quartus II
2. Open reference design from <IPInst_dir>\reference_designs\EL9800_DIGI_EP3C25F256
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 reference designs ET1810 (Altera)/
ET1810 IP Core 16 Ch. Dig. In-/Output (HW: FB1122)*

6.1.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

EL9800_DIGI_EP3C25F256_time_limited.sof

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

<IPInst_dir>\reference_designs\EL9800_DIGI_EP3C25F256

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 Beckhoff EL9800/FB1122 with SPI

6.2.1 Configuration and resource consumption

Table 14: Resource consumption SPI reference design EL9800/FB1122

Configuration		Resources		EP3C25
FMMU	2	LEs	12,405	50 %
SyncManager	4	Registers	6,519	26 %
RAM	4 KB	Pins	90	57 %
Register set	Medium + MI + TX Shift	M4K	6	9 %
Distributed Clocks	32 bit	PLLs	1	20 %
PDI	SPI mode 3, normal sample	Multiplier elements	0	0 %

6.2.2 Functionality

Configure PDI Selector S200 of the EL9800 Evaluation base board to PDI7: PIC/SPI.

Slave Sample Code Version 4.20 or later for EL9800 can be run with the SPI reference design provided.

6.2.3 Implementation

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

1. Open Altera Quartus II
2. Open reference design from <IPInst_dir>\reference_designs\EL9800_SPI_EP3C25F256
3. Open MegaWizard Plug-In Manager, select “Edit and existing custom megafunction variation”
4. Select “EtherCAT_SPI.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

6.2.4 SII EEPROM

Use this ESI for the SII EEPROM (PIC18 for green EL9800_2, PIC24 for black EL9800_4):

*Beckhoff Automation GmbH (Evaluation)/
IP Core reference designs ET1810 (Altera)/
ET1810 IP Core SPI V4.20 (HW: FB1122, PIC xx)*

6.3 EBV DBC3C40 with Digital I/O

6.3.1 Configuration and resource consumption

Table 15: Resource consumption Digital I/O reference 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.

6.3.2 Functionality

Functionality of the Digital I/O reference 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.

6.3.3 Implementation

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

1. Open Altera Quartus II
2. Open reference design from <IPInst_dir>\reference_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

6.3.4 SII EEPROM

Use this ESI for the SII EEPROM:

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

6.3.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

DBC3C40_EtherCAT_DIGI_time_limited.sof

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

<IPInst_dir>\reference_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.

6.4 EBV DBC4CE55 with NIOS

6.4.1 Configuration and resource consumption

Table 16: Resource consumption NIOS reference 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.

6.4.2 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).

6.4.3 Implementation

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

1. Open Altera Quartus II
2. Open reference design from <IPInst_dir>\reference_designs\DBC4CE55_EtherCAT_NIOS
3. Choose Tools on the menu bar and select SOPC Builder...
4. View SOPC and IP configurations
5. Select Generate at the bottom of the windows 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. <IPInst_dir>\reference_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, 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
→ “beckhoff_ethercat_0.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_install and press Build button. This will generate the memory initialization files.
14. Switch over to Quartus II window, start compilation (Menu Processing – Start compilation)
15. Download bitstream into FPGA

6.4.4 SII EEPROM

Use this ESI for the SII EEPROM:

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

6.4.5 Downloadable configuration file

An already synthesized time limited OpenCore Plus configuration file

DBC4CE55_EtherCAT_NIOS_time_limited.sof

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

<IPInst_dir>\reference_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.

6.5 Altera DE2-115 with NIOS

6.5.1 Configuration and resource consumption

Table 17: Resource consumption NIOS reference 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			

6.5.2 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-KKEY3 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

6.5.3 Implementation

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

1. Open Altera Quartus II
2. Open reference design from <IPInst_dir>\reference_designs\DE2_115_NIOS
3. Choose Tools on the menu bar and select SOPC Builder...
4. View SOPC and IP configurations
5. Select Generate at the bottom of the windows 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. <IPInst_dir>\reference_designs\DE2_115_NIOSworkspace
8. Choose File on the menu bar and select New – “NIOS II Application and BSP from Template”
9. Select SOPC information file, 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
→ “beckhoff_ethercat_0.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_install and press Build button. This will generate the memory initialization files.

14. Switch over to Quartus II window, start compilation (Menu Processing – Start compilation)
15. Download bitstream into FPGA

6.5.4 SII EEPROM

Use this ESI for the SII EEPROM:

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

6.5.5 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 reference design can be found in the

<IP\Inst_dir>\reference_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.

6.6 FPGA Download using Altera Quartus Programmer

1. Connect your configuration device (e.g. USB-Blaster™) to the PC and to the evaluation board
2. Beckhoff EL9800: Set PDI selector SW200 to PDI0:OFF
3. Beckhoff EL9800: Set SW150 to ON
4. Supply evaluation board with power
5. Open Quartus menu bar Tools → Programmer

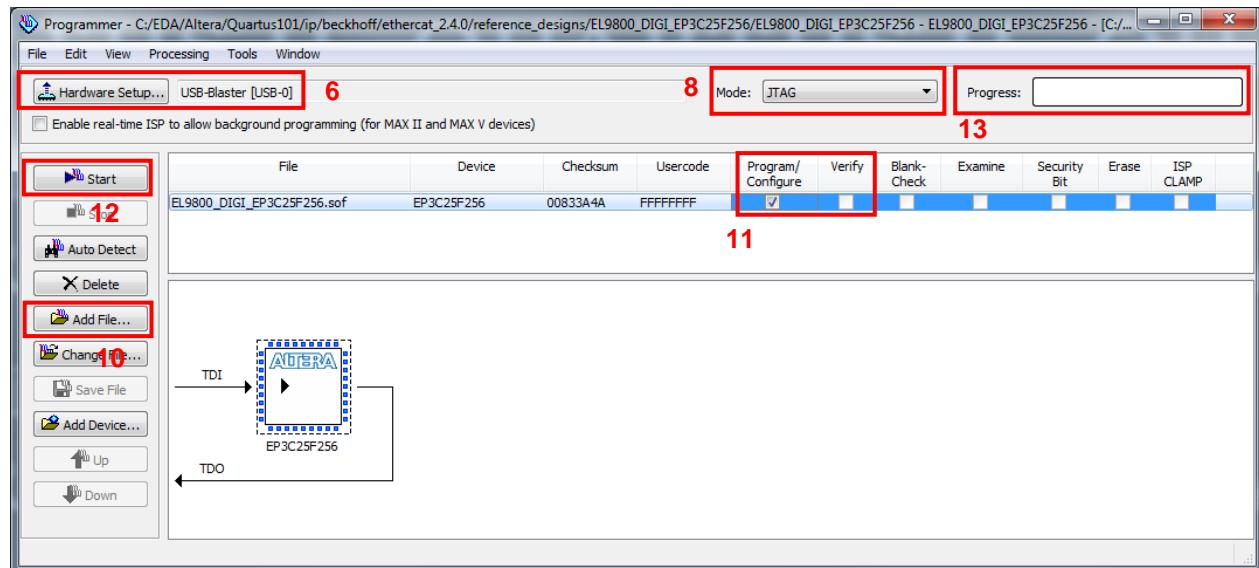


Figure 22: Quartus Programmer

6. Select Hardware Setup... → Configure your Download Hardware
7. Close Dialog with Close button
8. Choose JTAG
9. Select Auto Detect to read the JTAG chain
10. Add FPGA EtherCAT IP Core Reference file
Use *.sof file for temporarily configuration, use *.jic file for permanent configuration
11. Check Program/Configure and Verify options
12. Press Start button
13. Wait until FPGA code has been loaded to the FPGA
14. Beckhoff EL9800: Switch power off
15. Beckhoff EL9800: Set SW150 to OFF
16. Beckhoff EL9800: Set PDI selector SW200 to appropriate PDI (according to your reference design)
17. Beckhoff EL9800: Switch power on
18. Continue with updating the SII EEPROM.

6.7 Update SII EEPROM with EtherCAT Slave Information (ESI) using TwinCAT System Manager

1. Connect EtherCAT master with your slave, the link LEDs should indicate a link
2. Start TwinCAT
3. Select I/O – Configuration / I/O Devices
4. Hit F5 to scan Sub-Devices
5. Press OK for the hint that not all types of devices can be found automatically
6. Select EtherCAT device and press OK
7. Press OK to scan for boxes
8. Select No to not activate Free run
9. SII EEPROM has to be initialized with the proper device description:
Mark the EtherCAT slave (e.g. Box 1 in the Device (EtherCAT)-Tree) → Register EtherCAT → button Advanced Settings
10. Choose ESC Access → E²PROM → Smart View
11. Button Write E²PROM
12. Choose appropriate IP Core reference design and select file
13. Confirm with OK
14. Wait until EEPROM is written
15. Select Device 1 (EtherCAT) and hit F5 to scan Sub-Devices again
16. Select Copy All if differences were found, then press OK
17. You might want to check your Product ID in register 0x0E00:0x0E07, and your Vendor ID in register 0x0E08:0x0E0F (Advanced settings/ESC Access/Memory)

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.

was undertaken to achieve optimum results, i.e. by sophisticated constraining and design flow setting.

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 were 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 18: 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
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
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 19: 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 Overview

Table 20: Signal Overview

Signal	Type	Dir.	Description
CLK100	Clock	I	Clock input 100 MHz
CLK25	Clock	I	Clock input 25 MHz
CLK50	RMII	I	RMII reference clock
LED_RUN	LED	O	RUN LED
LED_ERR	LED	O	Error LED
LED_STATE_RUN	LED	O	RUN pin of dual-color STATE LEDs
LATCH_IN0/1	DC	I	Distributed Clocks LatchSignal input
LINK_ACT[2:0]	LED	O	Link/Activity LED
MCLK	MII/RMII	O	PHY signal indicating a link
MDIO	MII/RMII	BD	PHY Management Interface data
MDIO_DATA_ENA	MII/RMII	O	PHY Management Interface data output enable
MDIO_DATA_IN	MII/RMII	I	PHY Management Interface data input
MDIO_DATA_OUT	MII/RMII	O	PHY Management Interface data output
MII_RX_CLK0/1/2	MII	I	MII receive clock
MII_RX_DATA0/1/2[3:0]	MII	I	MII receive data
MII_RX_DV0/1/2	MII	I	MII receive data valid
MII_RX_ERR0/1/2	MII	I	MII receive error
MII_TX_CLK0/1/2	MII	I	MII transmit clock for automatic TX Shift configuration
MII_TX_DATA0/1/2[3:0]	MII	O	MII transmit data
MII_TX_ENA0/1/2	MII	O	MII transmit enable
MII_TX_SHIFT0/1/2[1:0]	MII	I	MII transmit signal shift for manual TX Shift configuration
nMII_LINK0/1/2	MII	I	MII PHY signal indicating a link
nRESET	General	I	Reset Input
nRMII_LINK0/1	RMII	I	RMII PHY signal indicating a link
PHY_OFFSET_VEC[4:0]	MII/RMII	I	Ethernet PHY Address Offset
PROM_CLK	EEPROM	O	EEPROM I ² C Clock
PROM_DATA	EEPROM	BD	EEPROM I ² C Data
PROM_DATA_ENA	EEPROM	O	EEPROM I ² C Data enable
PROM_DATA_IN	EEPROM	I	EEPROM I ² C Data input
PROM_DATA_OUT	EEPROM	O	EEPROM I ² C Data output
PROM_SIZE	EEPROM	I	EEPROM size configuration
RESET_OUT	General	O	Reset output (reset register)
RMII_RX_DATA0/1[1:0]	RMII	I	RMII receive data
RMII_RX_DV0/1	RMII	I	RMII carrier sense/receive data valid
RMII_RX_ERR0/1	RMII	I	RMII receive error
RMII_TX_DATA0/1[1:0]	RMII	O	RMII transmit data
RMII_TX_ENA0/1	RMII	O	RMII transmit data enable
SYNC_OUT0/1	DC	O	Distributed Clocks SyncSignal output

Table 21: PDI signal overview

PDI	Signal	Dir.	Description
General	PDI_GPI[63:0]	I	General purpose inputs (width configurable)
	PDI_GPO[63:0]	O	General purpose outputs (width configurable)
	PDI_SOF	O	Ethernet Start-of-Frame
	PDI_EOF	O	Ethernet End-of-Frame
	PDI_WD_TRIGGER	O	Watchdog trigger
	PDI_WD_STATE	O	Watchdog state
Digital I/O	PDI_DIGI_DATA_OUT0-3[7:0]	O	Output data
	PDI_DIGI_DATA_IN0-3[7:0]	I	Input data
	PDI_DIGI_DATA_ENA	O	Output data enable
	PDI_DIGI_LATCH_IN	I	External data latch signal
	PDI_DIGI_OE_EXT	I	External output Enable
	PDI_DIGI_OUTVALID	O	Output data valid
SPI	PDI_EMULATION	I	PDI emulation enable
	PDI_SPI_CLK	I	SPI clock
	PDI_SPI_SEL	I	SPI chip select
	PDI_SPI_DI	I	SPI data MOSI
	PDI_SPI_IRQ	O	SPI interrupt
	PDI_SPI_DO	BD	SPI data MISO
	PDI_SPI_DO_OUT	O	SPI data MISO
	PDI_SPI_DO_ENA	O	SPI data MISO enable
μC async.	PDI_EMULATION	I	PDI emulation enable
	PDI_uC_ADR[15:0]	I	Address bus
	PDI_uC_nBHE	I	Byte High Enable (16 bit μController interface only)
	PDI_uC_nRD	I	Read command
	PDI_uC_nWR	I	Write command
	PDI_uC_nCS	I	Chip select
	PDI_uC_IRQ	O	Interrupt
	PDI_uC_BUSY	O	EtherCAT IP Core is busy
	PDI_uC_DATA[7:0]	BD	Data bus 8 bit
	PDI_uC_DATA[15:0]	BD	Data bus 16 bit
	PDI_uC_DATA_IN[7:0]	I	Data bus 8 bit
	PDI_uC_DATA_IN[15:0]	I	Data bus 16 bit
	PDI_uC_DATA_OUT[7:0]	O	Data bus 8 bit
	PDI_uC_DATA_OUT[15:0]	O	Data bus 16 bit
	PDI_uC_DATA_ENA	O	Data bus enable
Avalon	PDI_EMULATION	I	PDI emulation enable
	PDI_AVALON_CLK	I	Avalon bus clock
	PDI_AVALON_ADR[15:0]	I	Address bus
	PDI_AVALON_RD_DATA[7:0]	O	Read data bus
	PDI_AVALON_WR_DATA[7:0]	I	Write data bus
	PDI_AVALON_READ	I	Read command
	PDI_AVALON_WRITE	I	Write command
	PDI_AVALON_CS	I	Chip select
	PDI_AVALON_IRQ	O	Interrupt
	PDI_AVALON_BUSY	O	EtherCAT IP Core is busy
	PDI_AVALON_SYNC0/1	O	Distributed Clocks SyncSignal output as interrupt

8.2 General Signals

Table 22: 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

8.2.1 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.

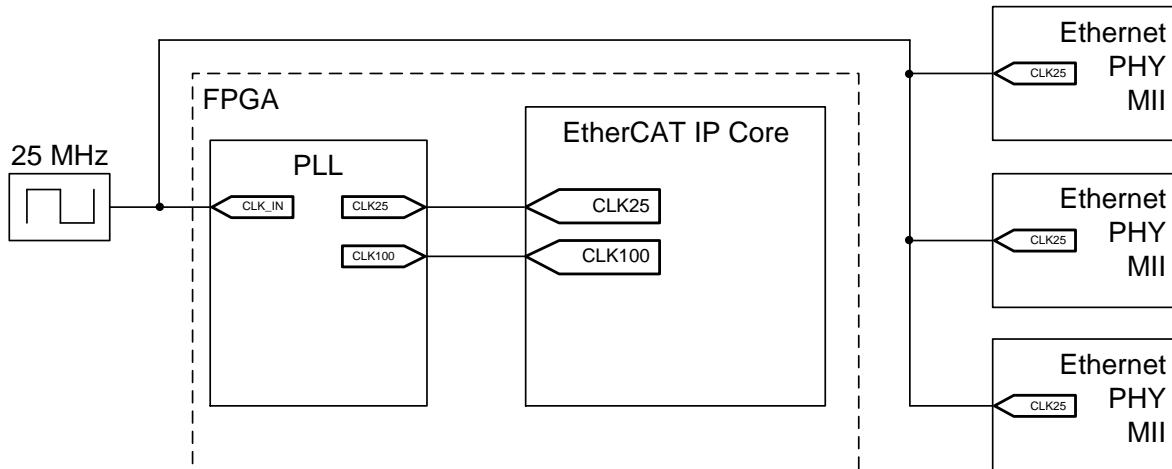


Figure 23: EtherCAT IP Core clock source (MII)

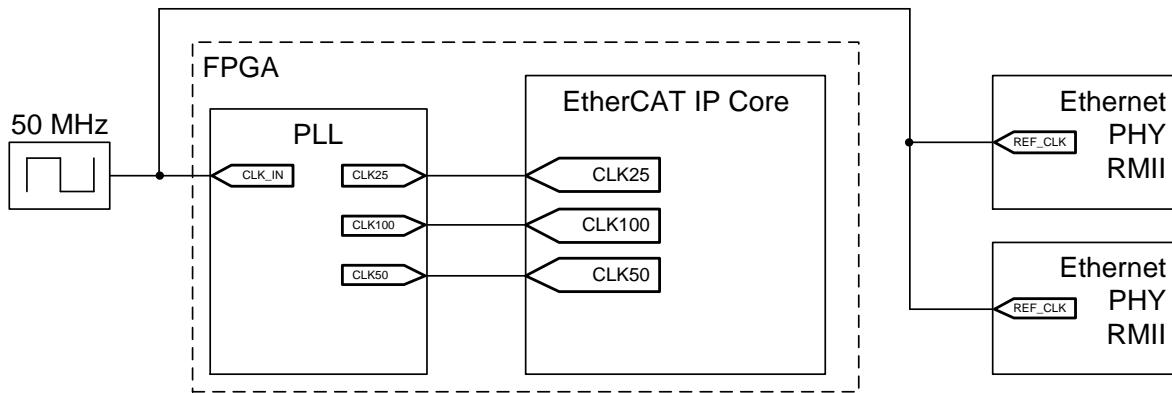


Figure 24: EtherCAT IP Core clock source (RMII)

8.3 SII EEPROM Interface Signals

Table 23: 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 I ² C Clock (output values: 0 or Z)
External tristate drivers for EEPROM/MI	PROM_CLK	OUTPUT	EEPROM I ² C Clock (output values: 0 or 1)
Tristate drivers inside core (EEPROM/MI)	PROM_DATA	BIDIR	EEPROM I ² C Data
External tristate drivers for EEPROM/MI	PROM_DATA_IN	INPUT	EEPROM I ² C Data: EEPROM → IP Core
	PROM_DATA_OUT	OUTPUT	EEPROM I ² C 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

8.4 LED Signals

Table 24 lists the signals used for the LEDs. The LED signals are active high. All LEDs should be green.

Table 24: 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.

8.5 Distributed Clocks SYNC/LATCH Signals

Table 25 lists the signals used with Distributed Clocks.

Table 25: 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.

8.6 Physical Layer Interface

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

Table 26 lists the general PHY interface signals.

Table 26: 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).

8.6.1 MII Interface

Table 27 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

Table 27: PHY Interface MII

Condition	Name	Direction	Description
Selected communication interface 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
Selected communication interface 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
	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
2 communication ports and selected communication interface Port1 = MII	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
	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
3 communication ports	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
3 communication ports 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

8.6.2 RMII Interface

Table 28 lists the signals used with RMII.

Table 28: PHY Interface RMII

Condition	Name	Direction	Description
Selected communication interface Port0/Port1 = 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
2 communication ports and selected communication interface 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

8.7 PDI Signals

8.7.1 General PDI Signals

Table 30 lists the signals available independent of the PDI configuration.

Table 29: 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	Watchdog trigger if 1
	PDI_WD_STATE	OUTPUT	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)

8.7.2 Digital I/O Interface

Table 30 lists the signals used with the Digital I/O PDI.

Table 30: 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

8.7.3 SPI Slave Interface

Table 31 used with an SPI PDI.

Table 31: 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

8.7.4 Asynchronous 8/16 Bit µController Interface

Table 32 lists the signals used with both, 8 Bit and 16 Bit asynchronous µController PDI.

Table 32: 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

8.7.4.1 8 Bit µController Interface

Table 33 lists the signals used with an 8 Bit µC PDI.

Table 33: 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

8.7.4.2 16 Bit µController Interface

Table 34 lists the signals used with a 16 Bit µC PDI.

Table 34: 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

8.7.5 Avalon On-Chip Bus

Table 35 lists the signals used with the Avalon PDI.

Table 35: 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 the 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 Address Configuration

The EtherCAT IP Core addresses Ethernet PHYs using logical port number (or PHY address register value) plus PHY address offset. Typically, 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).

9.2 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 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:

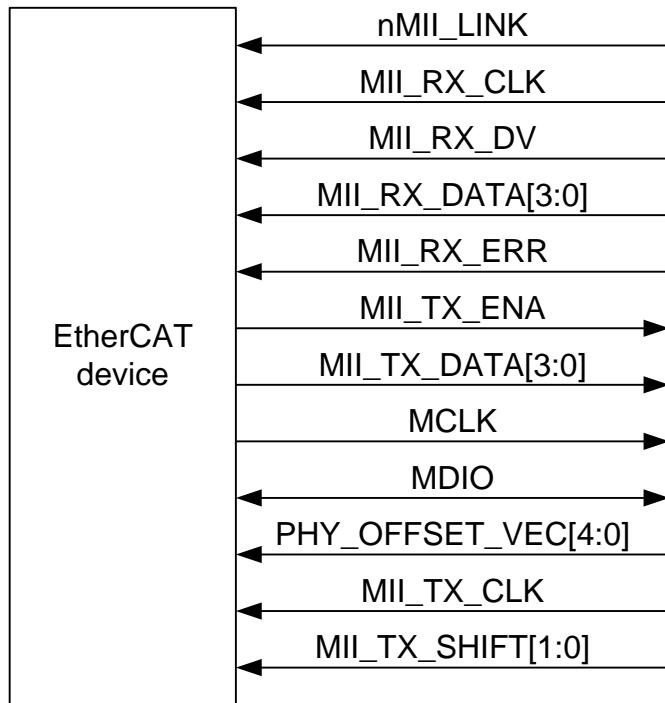


Figure 25: MII Interface signals

Table 36: 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)
MCLK	OUT	Management Interface clock (alias MCLK)
MDIO	BIDIR	Management Interface data (alias MDIO)
PHY_OFFSET_VEC[4:0]	IN	Configuration: PHY address offset
MII_TX_CLK	IN	Transmit Clock for automatic TX Shift compensation
MII_TX_SHIFT[1:0]	IN	Manual TX Shift compensation with additional registers

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.

9.2.2 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_TXH_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.

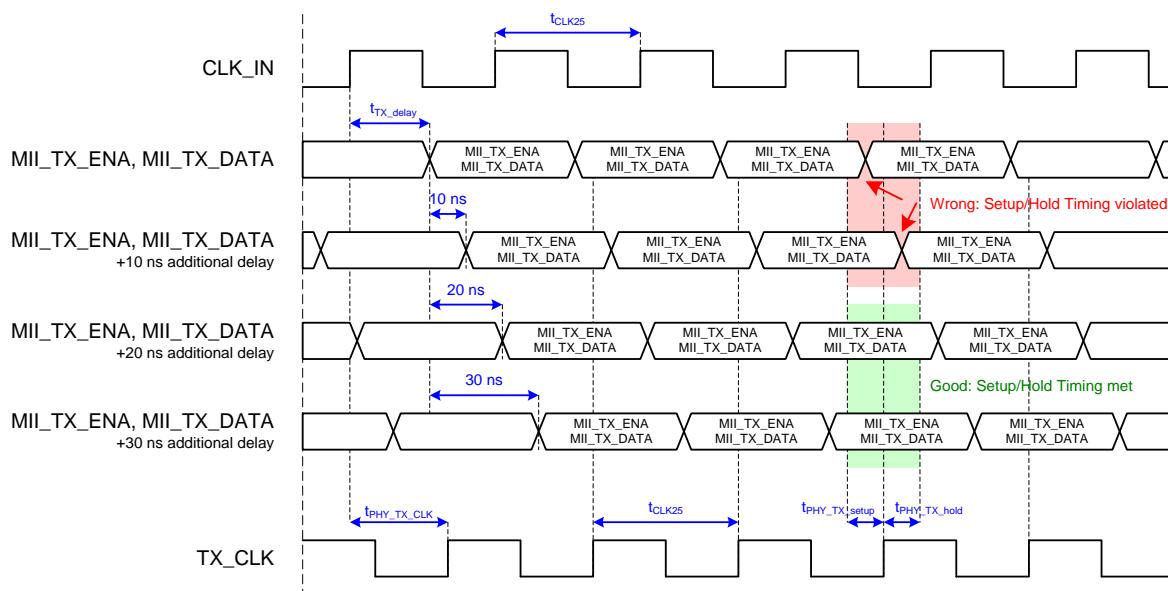


Figure 26: MII TX Timing Diagram

Table 37: 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 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.

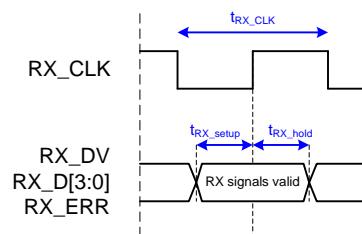
NOTE: The phase shift can be adjusted by displaying TX_CLK of a PHY and MII_TX_ENA/MII_TX_DATA[3:0] on an oscilloscope. MII_TX_ENA/MII_TX_DATA[3:0] is allowed to change between 0 ns and 25 ns after a rising edge of TX_CLK (according to IEEE802.3 – check your PHY's documentation). Setup phase shift so 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 38: MII timing characteristics

Parameter	Min	Typ	Max	Comment
t_{RX_CLK}		40 ns \pm 100 ppm		RX_CLK period (100 ppm with maximum FIFO Size only)
t_{RX_setup}	x^3			RX_DV/RX_DATA/RX_D[3:0] valid before rising edge of RX_CLK
t_{RX_hold}	x^3			RX_DV/RX_DATA/RX_D[3:0] valid after rising edge of RX_CLK
$t_{MI_startup}$		1.34 ms		Time between reset end and the first access of MI Link detection and configuration

NOTE: For MI timing diagrams refer to Section I.

**Figure 27: MII timing RX signals**

³ EtherCAT IP Core: time depends on synthesis results

9.2.4 MII example schematic

Refer to chapter 8.6 for more information on special markings (!). Take care of proper PHY address configuration. Take care of proper compensation of the TX_CLK phase shift.

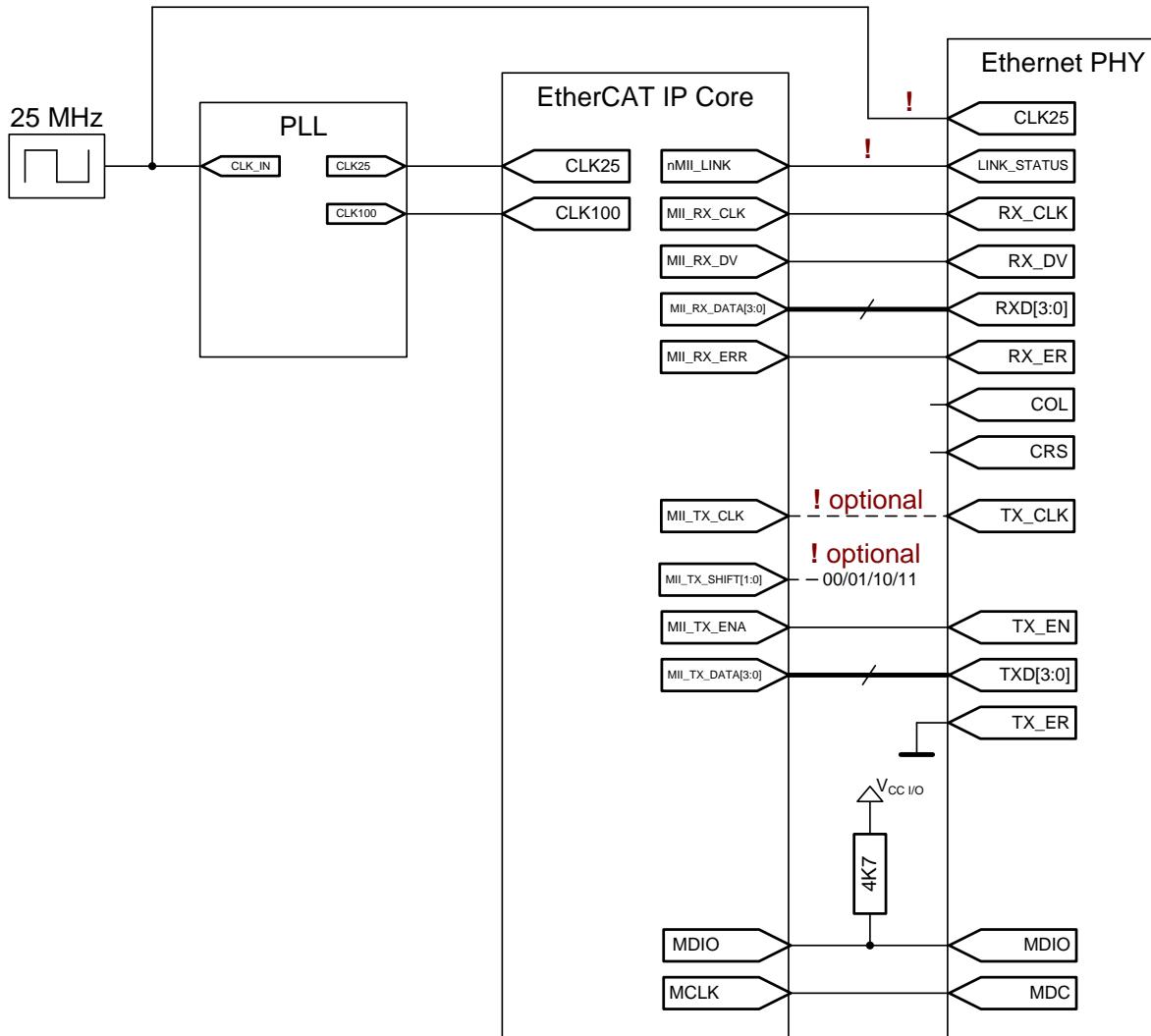


Figure 28: MII example schematic

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 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:

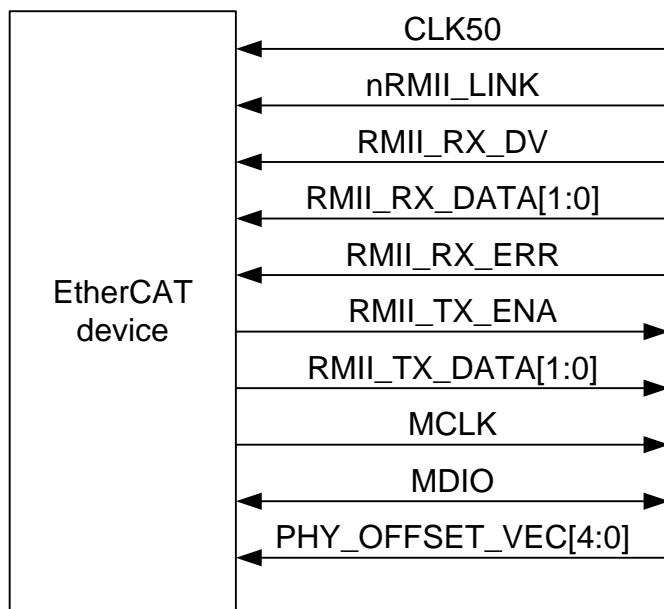


Figure 29: RMII Interface signals

Table 39: 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)
MCLK	OUT	Management Interface clock (alias MCLK)
MDIO	BIDIR	Management Interface data (alias MDIO)
PHY_OFFSET_VEC[4:0]	IN	Configuration: PHY address offset

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.

9.3.2 RMII example schematic

Refer to chapter 8.6 for more information on special markings (!). Take care of proper PHY address configuration.

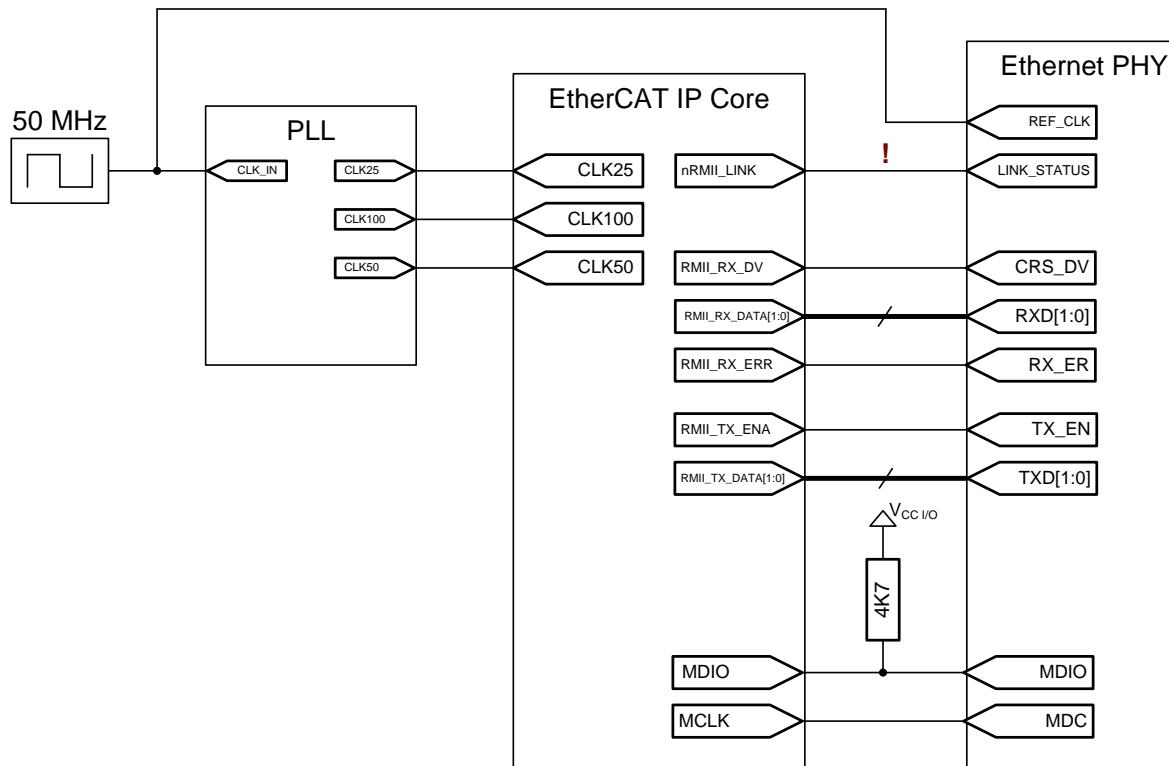


Figure 30: RMII example schematic

9.4 General Ethernet Timing specifications

For MII Management Interface timing diagrams refer to Section I.

Table 40: General Ethernet timing characteristics

Parameter	Min	Typ	Max	Comment
t_{Clk}		a) 400 ns b) ~ 1.44 μ s		MI_CLK period a) IP Core Version 2.0.0 and later ($f_{Clk} \approx 2.5$ MHz) b) IP Core Versions before 2.0.0 ($f_{Clk} \approx 700$ kHz)
t_{Write}		a) ~ 25.6 μ s b) ~ 92.16 μ s		MI Write access time a) IP Core Version 2.0.0 and later b) IP Core Versions before 2.0.0
t_{Read}		a) ~ 25.4 μ s b) ~ 91.44 μ s		MI Read access time a) IP Core Version 2.0.0 and later b) IP Core Versions before 2.0.0
t_{Diff}		40 ns		Processing delay (through ECAT Processing Unit) minus forwarding delay (alongside ECAT Processing Unit), for propagation delay calculation

10 PDI Description

Table 41: Available PDIs for EtherCAT IP Core

PDI number (PDI Control register 0x0140[7:0])	PDI name	IP Core
0	Interface deactivated	x
4	Digital I/O	x
5	SPI Slave	x
7	EtherCAT Bridge (port 3)	
8	16 Bit async. µC	x
9	8 Bit async. µC	x
10	16 Bit sync. µC	
11	8 Bit sync. µC	
16	32 Digital Input/0 Digital Output	
17	24 Digital Input/8 Digital Output	
18	16 Digital Input/16 Digital Output	
19	8 Digital Input/24 Digital Output	
20	0 Digital Input/32 Digital Output	
128	On-chip bus (Avalon)	x
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 are⁴:

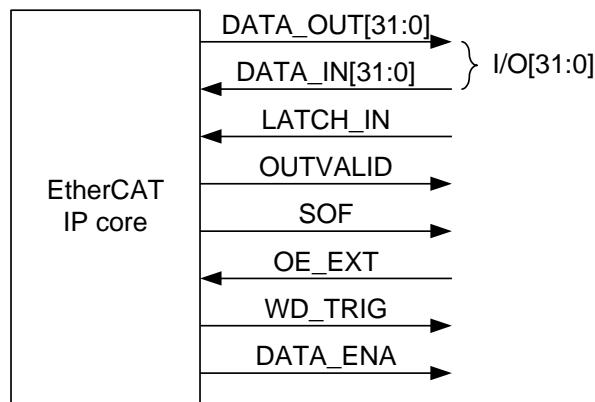


Figure 31: IP core digital I/O signals

Table 42: 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 43: 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

⁴ 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_TRIGGER 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.

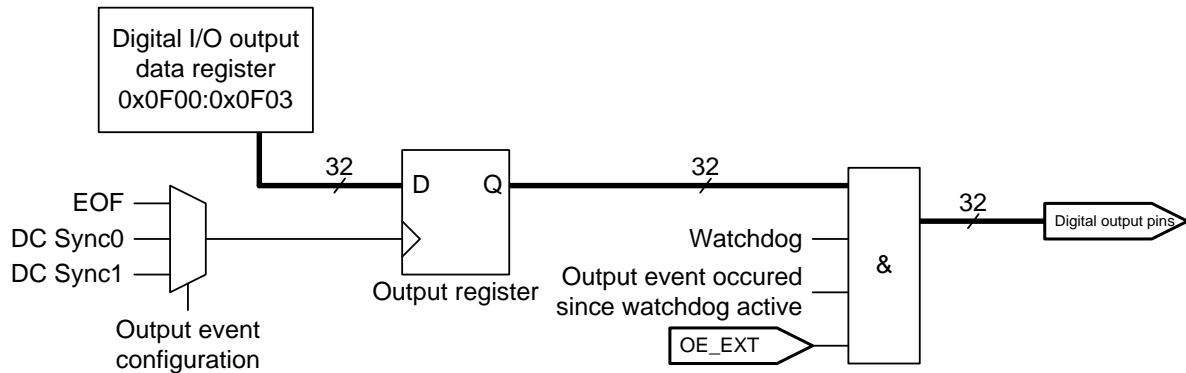


Figure 32: Digital Output Principle Schematic

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 0x0440:0x0441.

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_TRIGGER (watchdog trigger) signal can be used. A WD_TRIGGER 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_TRIGGER signal, OUTVALID can be configured to reflect WD_TRIGGER.

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 44: Digital I/O timing characteristics IP Core

Parameter	Min	Max	Comment
t_{DATA_setup}	x^5		Input data valid before LATCH_IN
t_{DATA_hold}	x^5		Input data valid after LATCH_IN
t_{LATCH_IN}	x^5		LATCH_IN high time
t_{SOF}	$40\text{ ns} - x^5$	$40\text{ ns} + x^5$	SOF high time
$t_{SOF_to_DATA_setup}$	0 ns	$1,2\text{ }\mu\text{s} - x^5$	Input data valid after SOF, so that Inputs can be read in the same frame
$t_{SOF_to_DATA_hold}$	$1,6\text{ }\mu\text{s} + x^5$		Input data invalid after SOF
$t_{input_event_delay}$	440 ns		Time between consecutive input events
$t_{OUTVALID}$	$80\text{ ns} - x^5$	$80\text{ ns} + x^5$	OUTVALID high time
$t_{DATA_to_OUTVALID}$	$80\text{ ns} - x^5$		Output data valid before OUTVALID
t_{WD_TRIG}	$40\text{ ns} - x^5$	$40\text{ ns} + x^5$	WD_TRIG high time
$t_{DATA_to_WD_TRIG}$		$20\text{ ns} + x^5$	Output data valid after WD_TRIG
$t_{OE_EXT_to_DATA_invalid}$	0 ns	x^5	Outputs zero or Outputs high impedance after OE_EXT set to low
$t_{output_event_delay}$	320 ns		Time between consecutive output events
$t_{OUT_ENA_valid}$	$80\text{ ns} - x^5$		OUT_ENA valid before OUTVALID
$t_{OUT_ENA_invalid}$	$80\text{ ns} - x^5$		OUT_ENA invalid after OUTVALID

⁵ EtherCAT IP Core: time depends on synthesis results

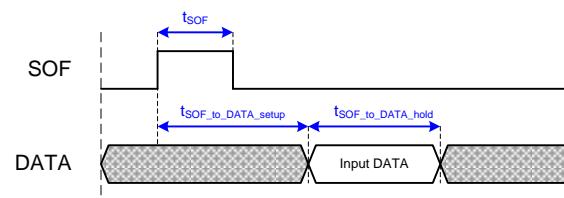


Figure 33: Digital Input: Input data sampled at SOF, I/O can be read in the same frame

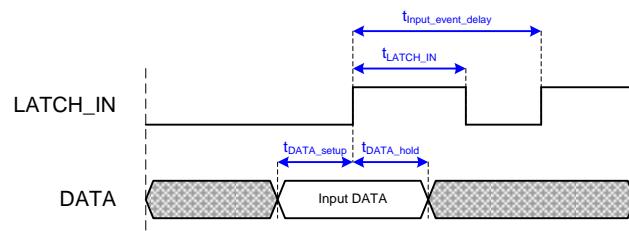


Figure 34: Digital Input: Input data sampled with LATCH_IN

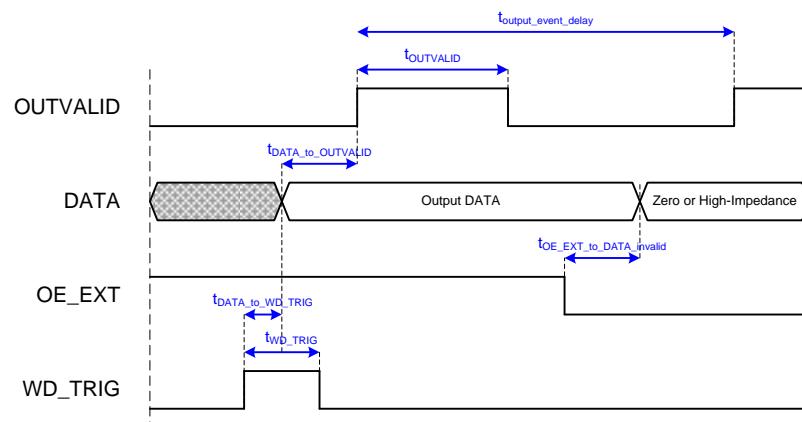


Figure 35: Digital Output timing

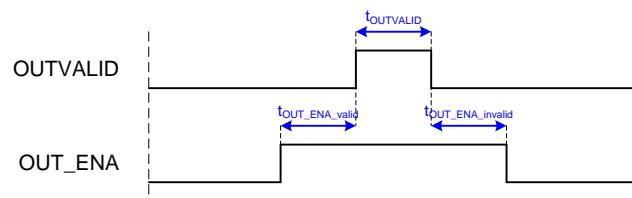


Figure 36: OUT_ENA timing

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_IRQ⁶:

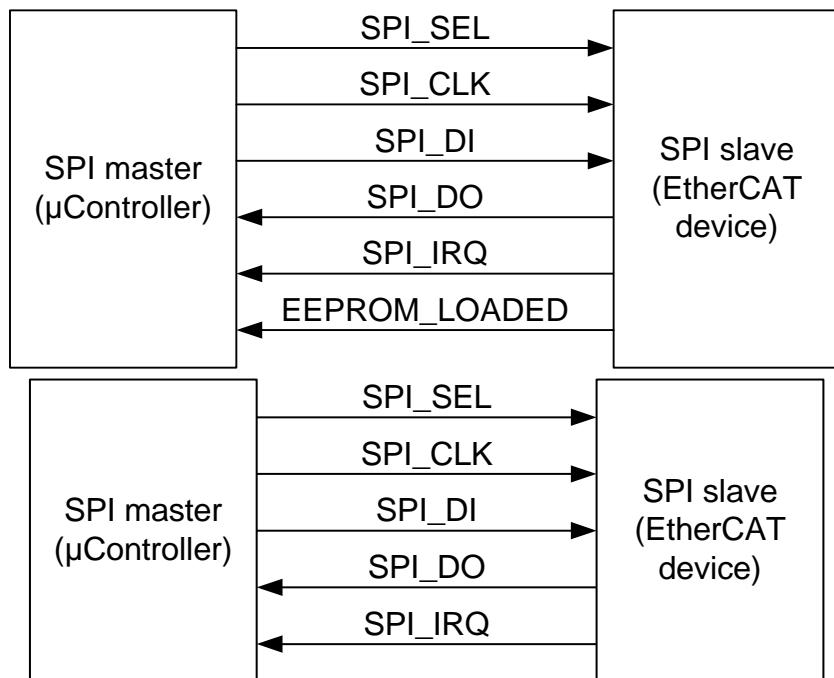


Figure 37: SPI master and slave interconnection

Table 45: 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.

⁶ 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 46: 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]	address bits [4:0]	A[4:0]	address bits [4:0]
	CMD0[2:0]	read/write command	CMD0[2:0]	3 byte addressing: 110b
2	D0[7:0]	data byte 0	A[15:13]	address bits [15:13]
			CMD1[2:0]	read/write command
			res[1:0]	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 47: 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 48: 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 49.

Table 49: 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]	address bits [4:0]	A[4:0]	address bits [4:0]
	CMD0[2:0]	write command: 100b	CMD0[2:0]	3 byte addressing: 110b
2	D0[7:0]	data byte 0	A[15:13]	address bits [15:13]
			CMD1[2:0]	3 byte addressing: 110b
			res[1:0]	two reserved bits, set to 00b
3	D1[7:0]	data byte 1	A[15:13]	address bits [15:13]
			CMD2[2:0]	write command: 100b
			res[1:0]	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 50: SPI timing characteristics IP Core

Parameter	Min	Max	Comment
t_{CLK}	$33\text{ ns} + x^7$		SPI_CLK frequency ($f_{CLK} \leq 30\text{ MHz}$)
$t_{SEL_to_CLK}$	x^7		First SPI_CLK cycle after SPI_SEL asserted
$t_{CLK_to_SEL}$	a) x^7 b) $t_{CLK}/2 + x^7$		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^7	BUSY valid after BUSY OUT Enable
$t_{BUSY_OE_to_DO_valid}$		x^7	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^7	Status/Interrupt Byte 0 bit 7 valid after SPI_SEL asserted
$t_{SEL_to_DO_invalid}$	0 ns	x^7	Status/Interrupt Byte 0 bit 7 invalid after SPI_SEL deasserted
t_{STATUS_valid}	x^7		Time until status of last access is valid. Can be ignored if status is not used.
t_{access_delay}	x^7		Delay between SPI accesses
t_{DI_setup}	x^7		SPI_DI valid before SPI_CLK edge
t_{DI_hold}	x^7		SPI_DI valid after SPI_CLK edge
$t_{CLK_to_DO_valid}$		x^7	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.

⁷ EtherCAT IP Core: time depends on synthesis results

Table 51: Read/Write timing diagram symbols

Symbol	Comment
A15..A0	Address bits [15:0]
D0_7..D0_0	Data bits byte 0 [7:0]
D1_7..D1_0	Data bits byte 1 [7:0]
I0_7..I0_0	Interrupt request register 0x0220 [7:0]
I1_7..I1_0	Interrupt request register 0x0221 [7:0]
I2_7..I2_0	Interrupt request register 0x0222 [7:0]
C0_2..C0_0	Command 0 [2:0]
C1_2..C1_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

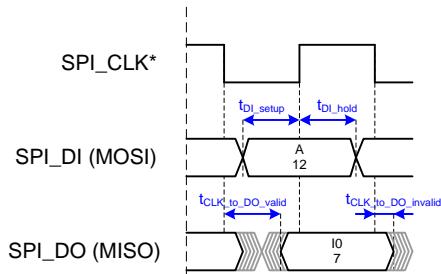


Figure 38: Basic SPI_DI/SPI_DO timing (*refer to timing diagram for relevant edges of SPI_CLK)

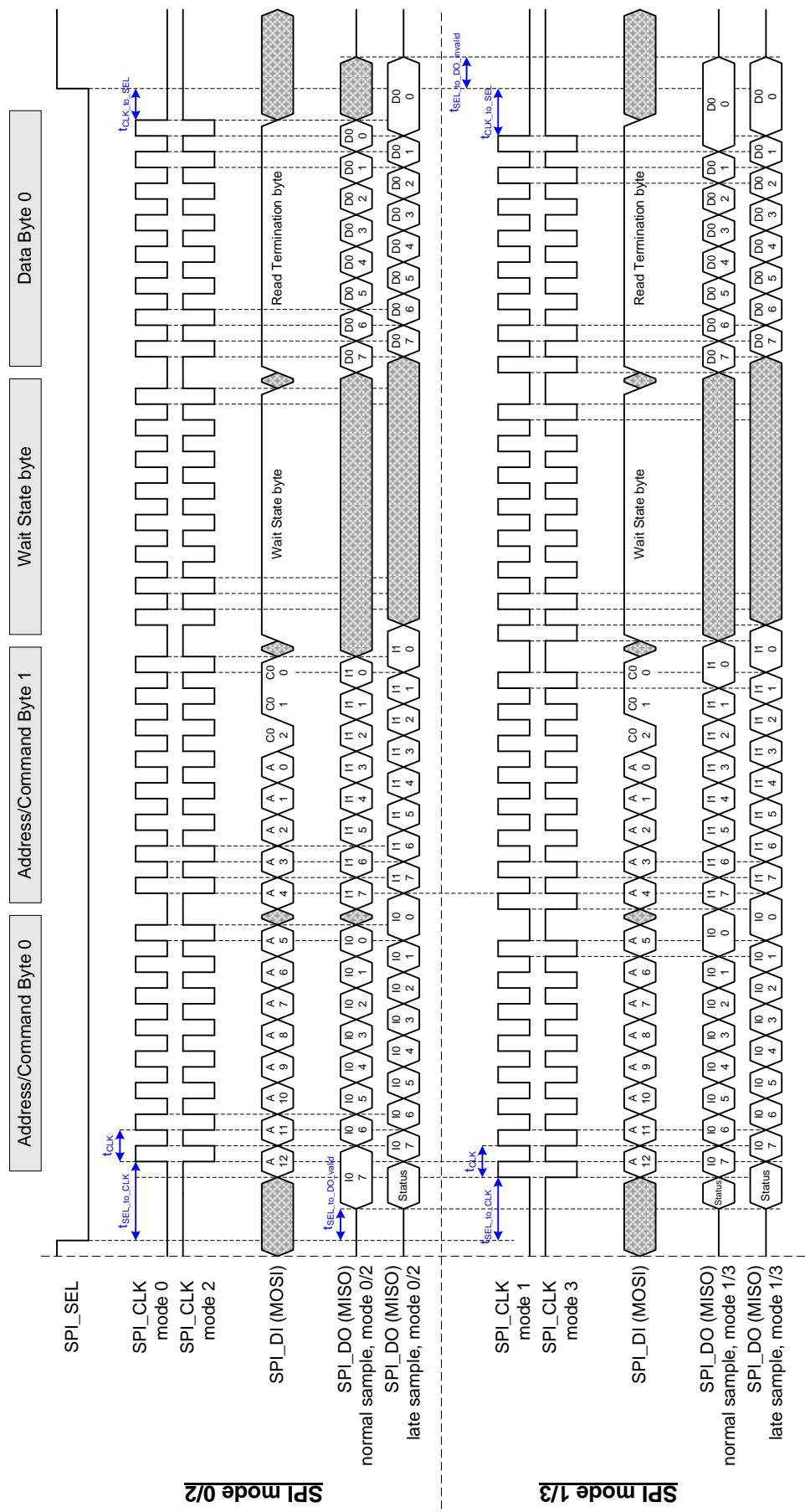


Figure 39: SPI read access (2 byte addressing, 1 byte read data) with Wait State byte

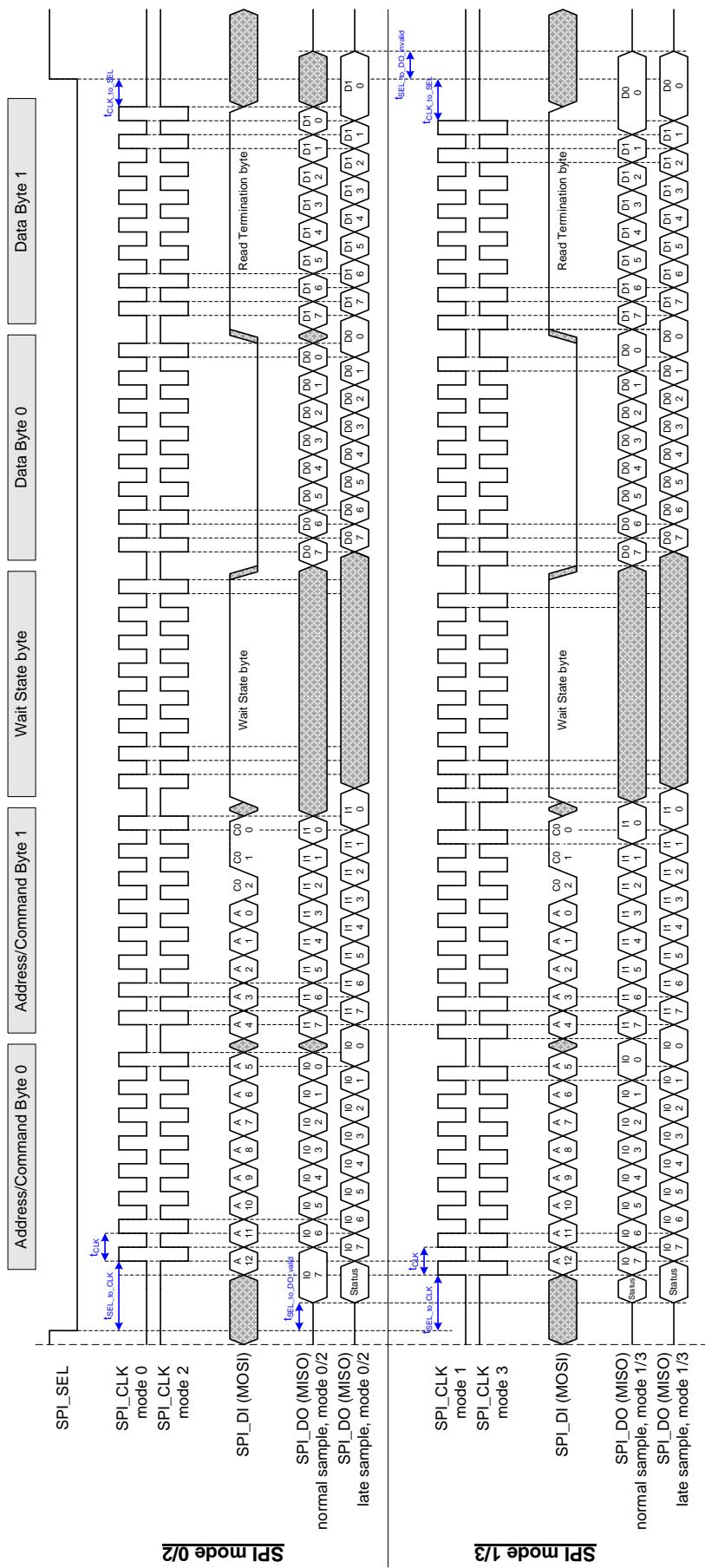


Figure 40: SPI read access (2 byte addressing, 2 byte read data) with Wait State byte

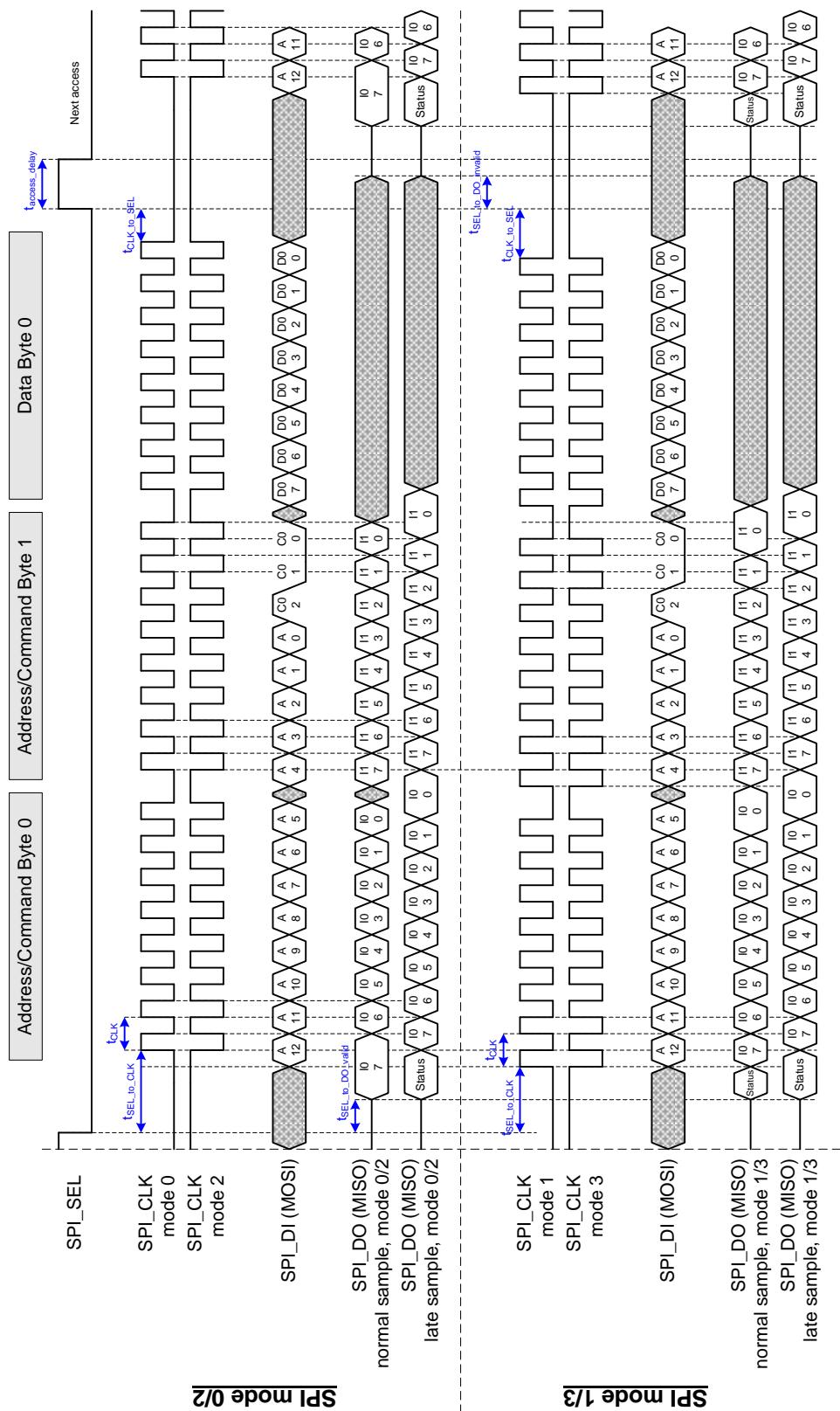


Figure 41: SPI write access (2 byte addressing, 1 byte write data)

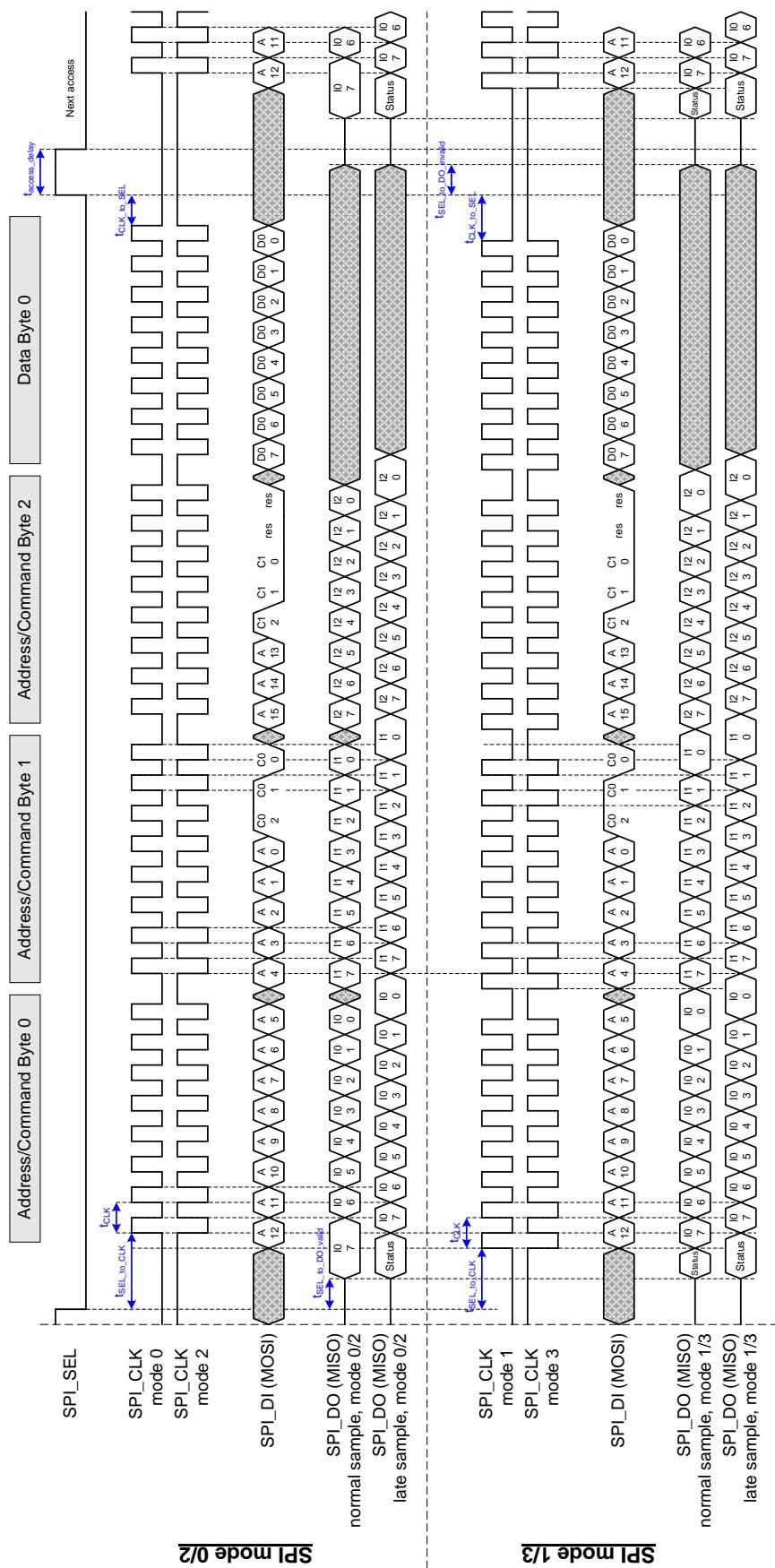


Figure 42: SPI write access (3 byte addressing, 1 byte write data)

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 are⁸:

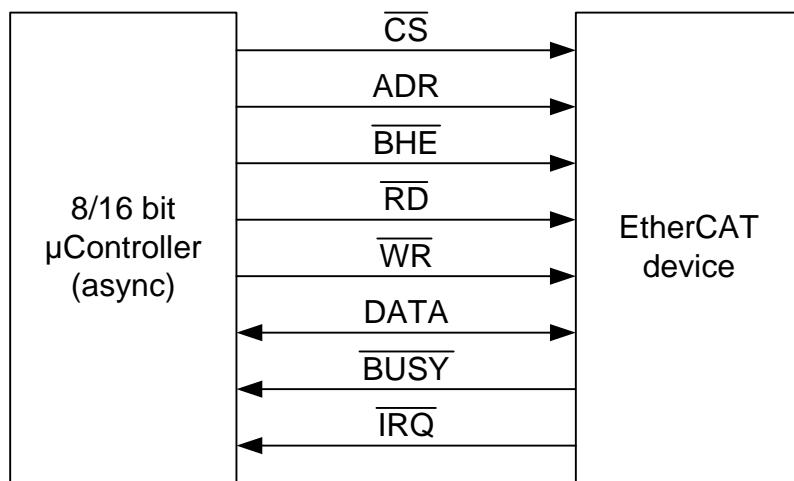


Figure 43: µController interconnection⁹

Table 52: µController signals

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

⁸ The prefix `PDI_uC_` or `PDI_uC_8` is added to the µController signals if the EtherCAT IP Core is used.

⁹ 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 53: 8 bit µController interface access types

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

Table 54: 16 bit µController interface access types

ADR[0]	BHE (act. low)	Access
0	0	16 bit access to ADR[15:0] and ADR[15:0]+1 (low and high byte)
0	1	8 bit access to ADR[15:0] (low byte, even address)
1	0	8 bit access to ADR[15:0] (high byte, odd address)
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 20 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.

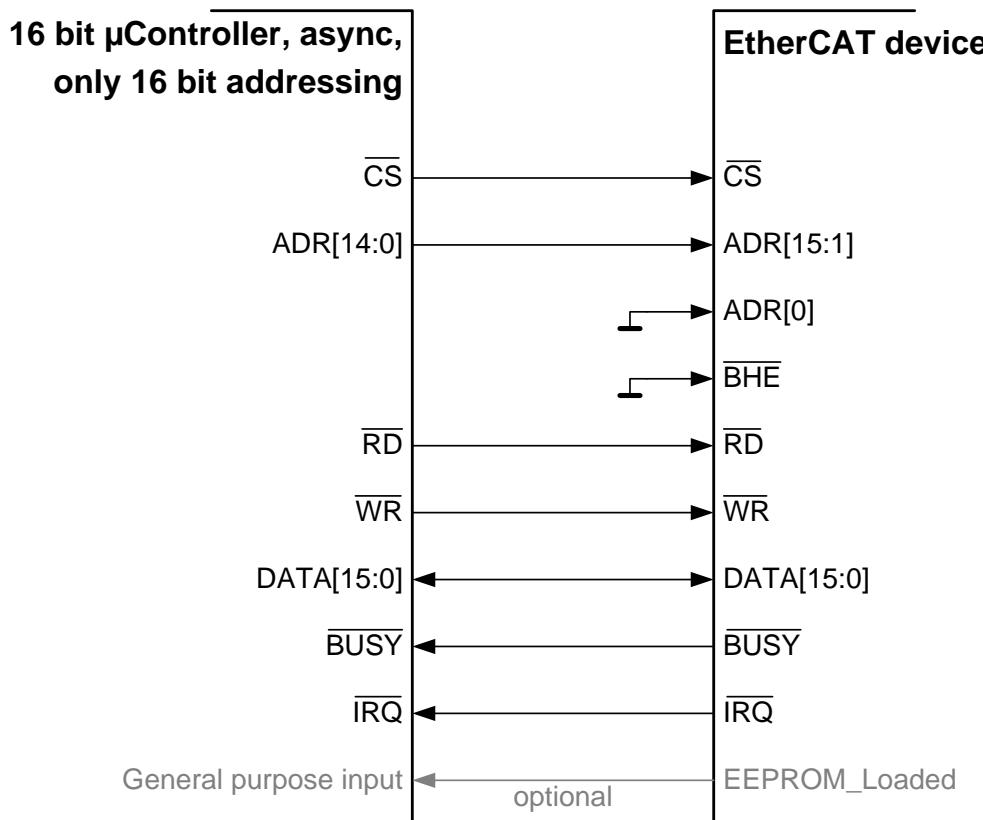


Figure 44: Connection with 16 bit µControllers without byte addressing

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.

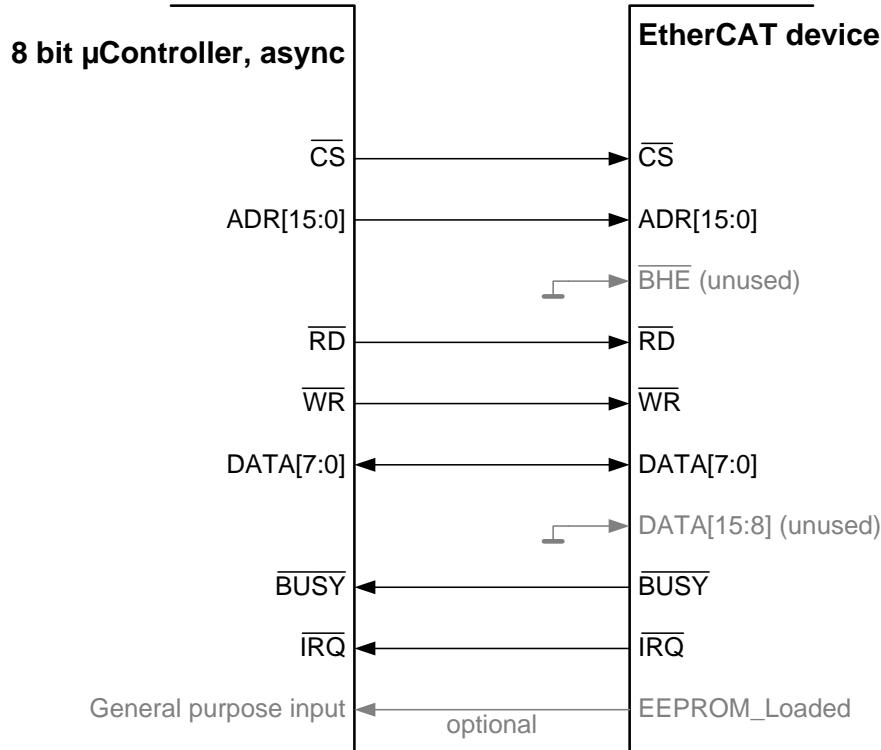


Figure 45: Connection with 8 bit µControllers (BHE and DATA[15:8] should not be left open)

10.3.9 Timing Specification

Table 55: µController timing characteristics IP Core

Parameter	Min	Max	Comment
$t_{CS_to_BUSY}$		x^{10}	BUSY driven and valid after CS assertion
$t_{ADR_BHE_setup}$	x^{10}		ADR and BHE valid before RD assertion
$t_{RD_to_DATA_driven}$	0 ns ¹¹		DATA bus driven after RD assertion
$t_{RD_to_BUSY}$	0 ns ¹¹	x^{10}	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) $t_{read_int}^{11}$	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) $t_{read_int} + t_{prec_write} + t_{Coll} - t_{WR_to_RD}^{11}$	b) with preceding write access and $t_{WR_to_RD} < t_{prec_write} + t_{Coll}$
		c) 420 ns ¹¹	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 ¹¹	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 ¹¹ b) 300 ns ¹¹	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^{10} b) $x^{10}-20$ 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 ¹¹		DATA invalid after ADR or BHE change
$t_{CS_RD_to_DATA_release}$	0 ns ¹¹	x^{10}	DATA bus released after CS deassertion or RD deassertion
$t_{CS_to_BUSY_release}$	0 ns ¹¹	x^{10}	BUSY released after CS deassertion
t_{CS_delay}	0 ns ¹¹		Delay between CS deassertion and assertion
t_{RD_delay}	x^{10}		Delay between RD deassertion and assertion
$t_{ADR_BHE_DATA_setup}$	x^{10}		ADR, BHE and Write DATA valid before WR deassertion

¹⁰ EtherCAT IP Core: time depends on synthesis results

¹¹ 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^{10}		ADR, BHE and Write DATA valid after WR deassertion
t_{WR_active}	x^{10}		WR assertion time
$t_{BUSY_to_WR_CS}$	0 ns ¹¹		WR or CS deassertion after BUSY deassertion
$t_{WR_to_BUSY}$		x^{10}	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) $t_{write_int} - t_{WR_delay}^{11}$	b) with preceding write access and $t_{WR_delay} < t_{write_int}$ (Write after rising edge of WR)
		c) 0 ns ¹¹	c) without preceding write access or $t_{WR_delay} \geq t_{write_int}$ (Write after rising edge of WR)
		d) 180 ns ¹¹	d) 8 bit access, absolute worst case with preceding write access ($t_{WR_delay} = \text{min}$, $t_{WR_int} = \text{max}$, Write after rising edge of WR)
		e) 260 ns ¹¹	e) 16 bit access, absolute worst case with preceding write access ($t_{WR_delay} = \text{min}$, $t_{WR_int} = \text{max}$, Write after rising edge of WR)
t_{write_int}		a) 180 ns ¹¹ b) 260 ns ¹¹	Internal write time a) 8 bit access b) 16 bit access
t_{WR_delay}	x^{10}		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^{10}		Time both CS and WR have to be deasserted simultaneously (only if CS is deasserted at all)
$t_{CS_RD_overlap}$	x^{10}		Time both CS and RD have to be deasserted simultaneously (only if CS is deasserted at all)

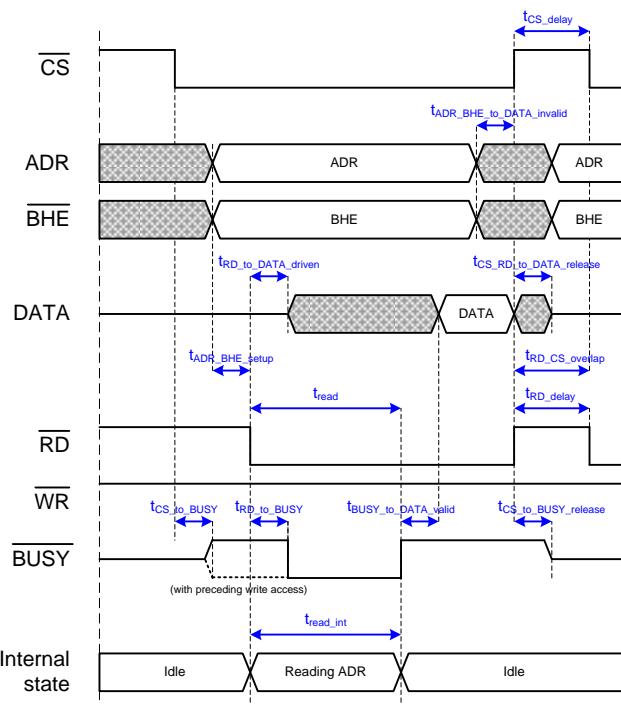


Figure 46: Read access (without preceding write access)

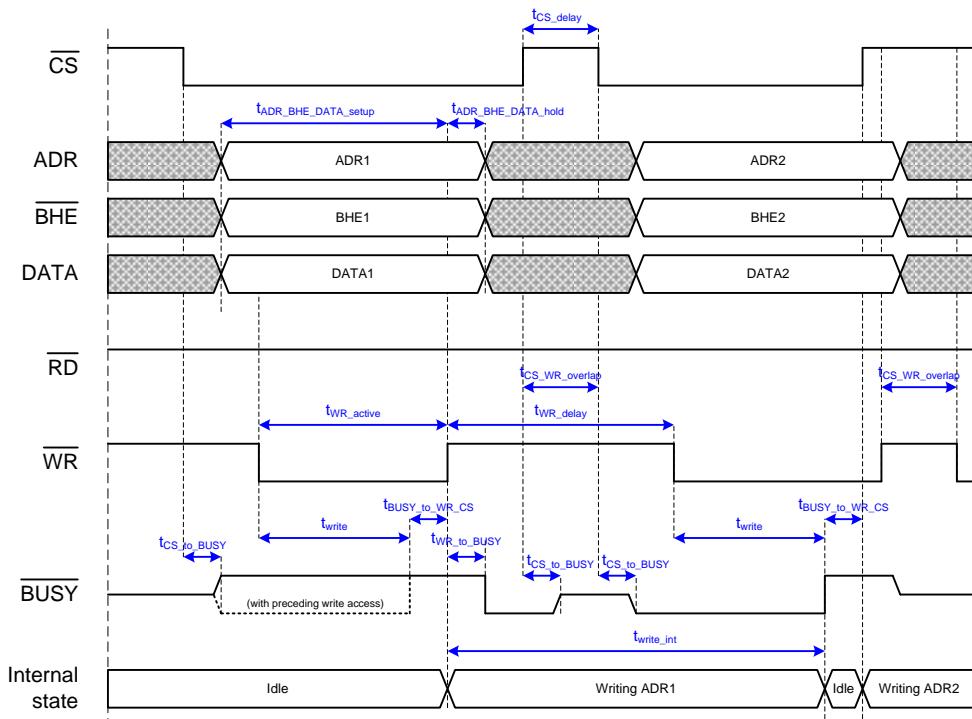


Figure 47: Write access (write after rising edge nWR, without preceding write access)

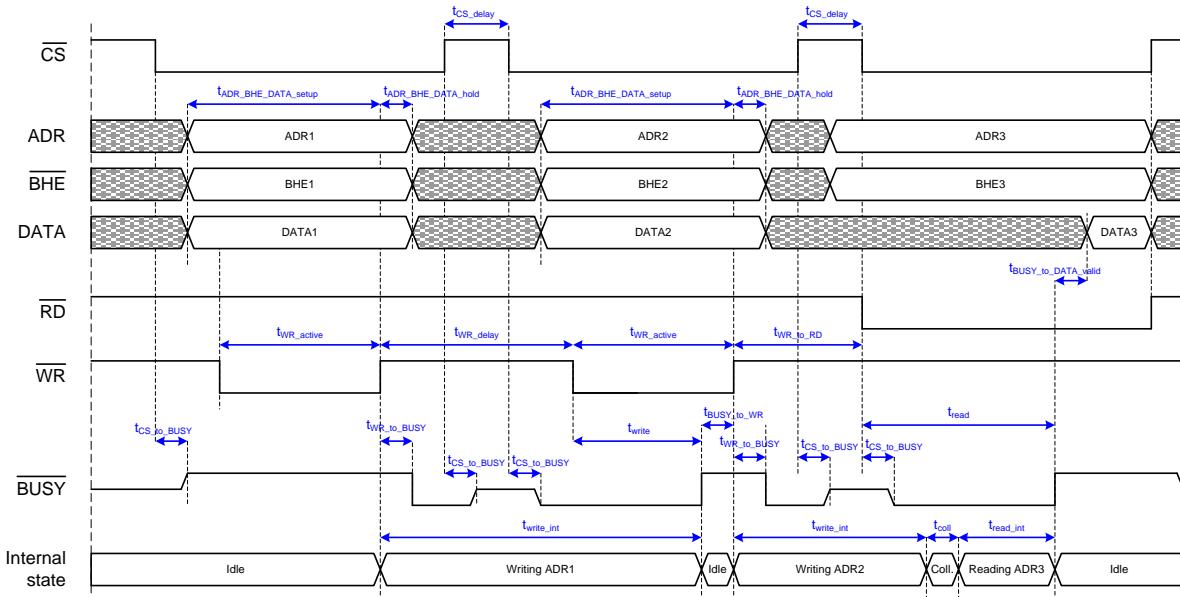


Figure 48: Sequence of two write accesses and a read access

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.

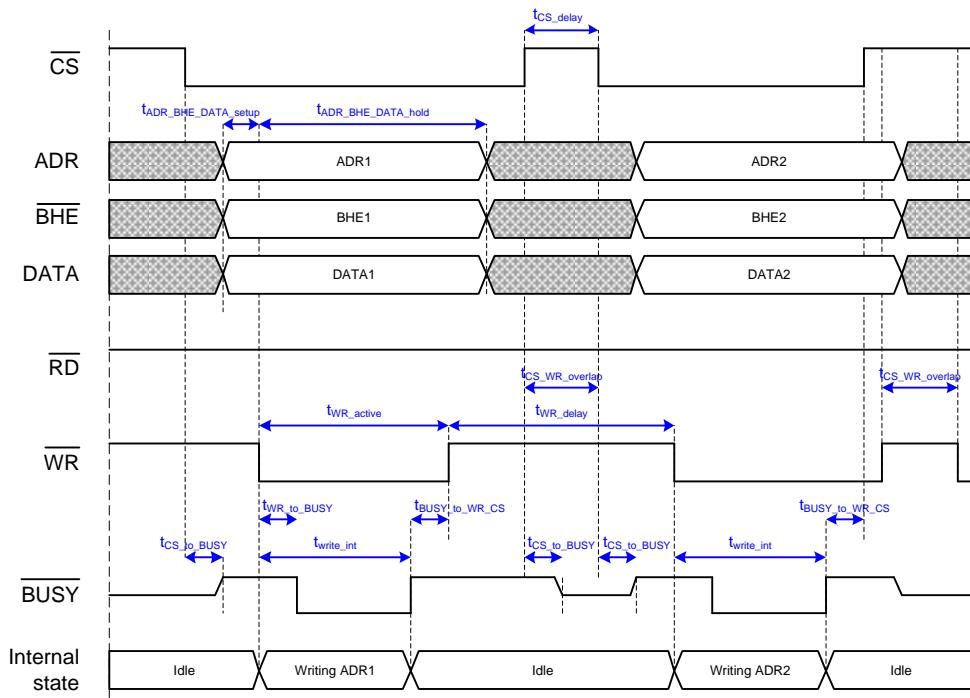


Figure 49: Write access (write after falling edge nWR)

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 are¹²:

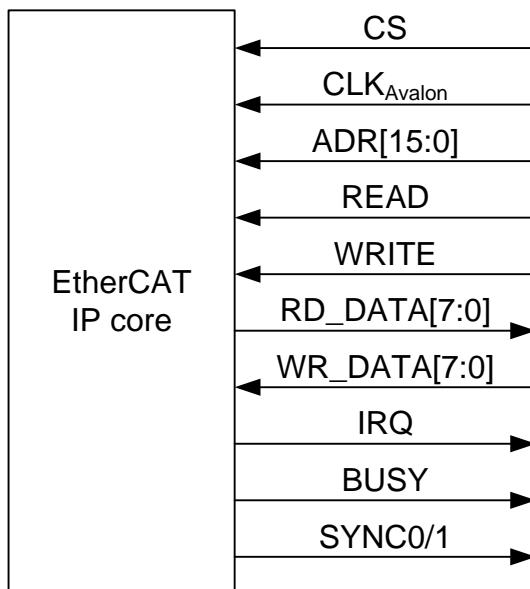


Figure 50: Avalon signals

Table 56: Avalon signals

Signal	Direction	Description	Signal polarity
CS	IN	Chip select	act. high
CLK _{Avalon}	IN	Avalon Bus clock (rising edge synchronous with rising edge of CLK25 of the IP Core)	
ADR[15:0]	IN	Address	
READ	IN	Read request	act. high
WRITE	IN	Write request	act. high
RD_DATA[7:0]	OUT	Read data	
WR_DATA[7:0]	IN	Write data	
IRQ	OUT	Interrupt	act. high
SYNC0/1	OUT	DC Sync0/1 used as interrupts	act. high
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>).

¹² The prefix `PDI_AVALON_` is added to the Avalon interface signals for the IP Core interface.

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:

$$\text{Avalon clock frequency} = N * 25 \text{ MHz} \quad (N=1\ldots31)$$

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 width 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.

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 Width 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 57: 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}	$\frac{1}{N * 25MHz}$ ¹³	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) 320 ns + $\frac{40ns}{N}$	32 Bit write access time (W=4) a) N=1 b) N>1
	80 ns	a) 160 ns b) 160 ns + $\frac{40ns}{N}$	16 Bit write access time (W=2) a) N=1 b) N>1
	t _{Clk}		8 Bit write access time (W=1)

¹³ EtherCAT IP Core: time depends on synthesis results

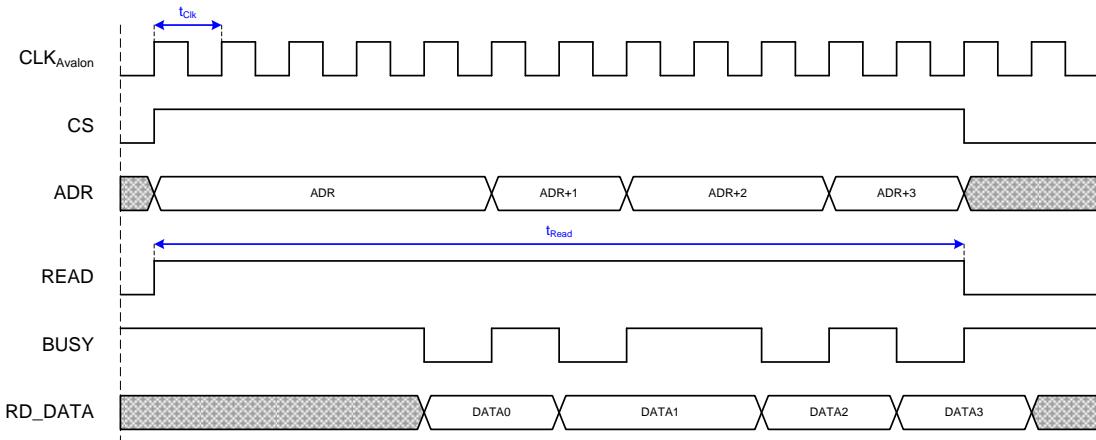


Figure 51: Avalon Read Access (32 Bit, W=4)

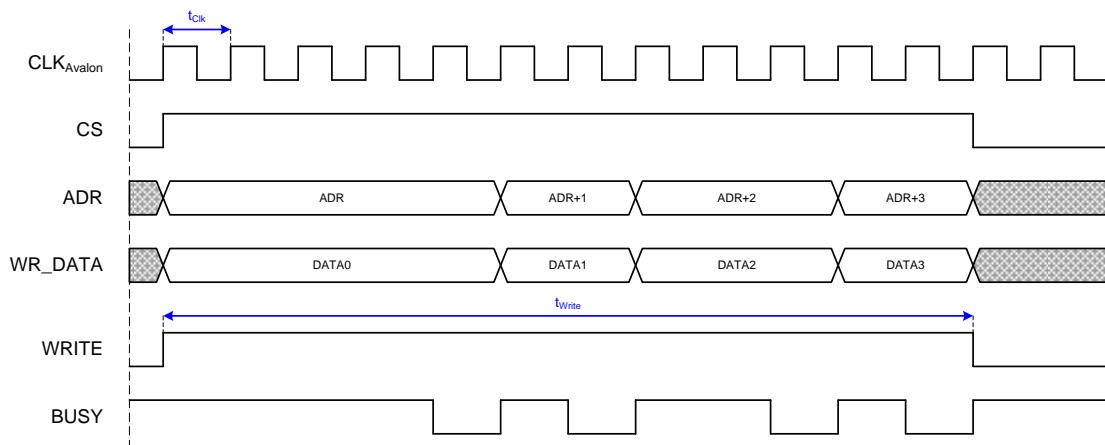


Figure 52: Avalon Write Access (32 Bit, W=4)

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):

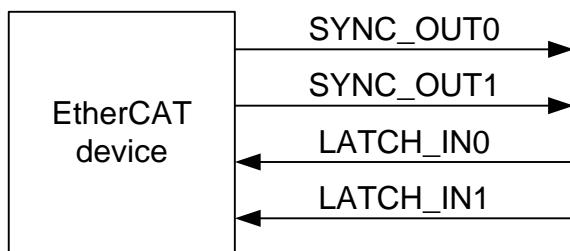


Figure 53: Distributed Clocks signals

Table 58: 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 59: DC SYNC/LATCH timing characteristics IP Core

Parameter	Min	Max	Comment
t_{DC_LATCH}	$12\text{ ns} + x^{14}$		Time between Latch0/1 events
$t_{DC_SYNC_Jitter}$		$11\text{ ns} + x^3$	SYNC0/1 output jitter

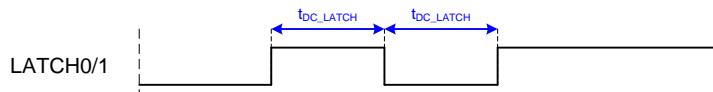


Figure 54: LatchSignal timing

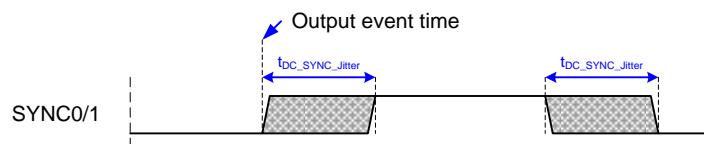


Figure 55: SyncSignal timing

¹⁴ 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:

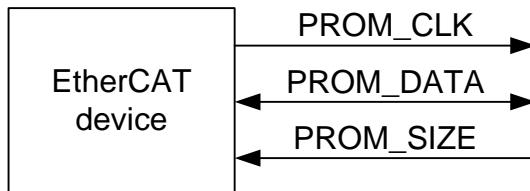


Figure 56: I²C EEPROM signals

Table 60: I²C EEPROM signals

Signal	Direction	Description
PROM_CLK	OUT	I ² C clock (alias EEPROM_CLK)
PROM_DATA	BIDIR	I ² C 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 Timing specifications

Table 61: EEPROM timing characteristics IP Core

Parameter	Typical		Comment
	Up to 16 kBit	32 kBit-4 MBit	
t _{Clk}	~ 6.72 µs		EEPROM clock period ($f_{Clk} \approx 150$ kHz)
t _{Write}	~ 250 µs	~ 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 62: AC Characteristics

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

Table 63: 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) $320+x^{15}$ b) $280+x^{15}$	a) $340+x^{15}$ b) $300+x^{15}$	a) $360+x^{15}$ b) $320+x^{15}$	ns

NOTE: Average timings are used for DC calculations.

¹⁵ EtherCAT IP Core: time depends on synthesis results

14 Synthesis Constraints

The following table contains basic IP Core constraints.

Table 64: 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.		I ² C clock. Actual ESC output clock is 6.72 µs (~ 150 kHz). Min. 2.5µs (400 KHz) for example I ² C 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 I ² C 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|altppll_component|auto_generated|pll1|clk[0] } 5 [get_ports
    nMII_LINK* ]
set_input_delay -clock { PLL_INST|altppll_component|auto_generated|pll1|clk[1] } 5 [get_ports
    MII_TX_CLK*]

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

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

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

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

#false paths
set_false_path -from {nRESET} -to
    {PLL:PLL_INST|altppll:altppll_component|altppll_bbcl:auto_generated|pll_lock_sync}
set_false_path -from
    {PLL:PLL_INST|altppll:altppll_component|altppll_bbcl:auto_generated|pll_lock_sync} -to
    [get_ports {PROM_DATA PROM_CLK}]
set_false_path -from [get_clocks {PLL_INST|altppll_component|auto_generated|pll1|clk[0]}] -to
    [get_clocks {DIGI_CLK}]
set_false_path -from [get_clocks {PLL_INST|altppll_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
Eiserstr. 5
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