Hardware Data Sheet

ET1100

EtherCAT® Slave Controller

Section I –
EtherCAT Slave Controller Technology

Section II –
EtherCAT Slave Controller Register Description

Section III –
ET1100 Hardware Description: Pinout, Interface description, electrical and mechanical specification, ET1100 register overview

Version 1.8
Date: 2010-05-03

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

DOCUMENT HISTORY

Version	Comment
0.6	Initial release (Section I V0.0, Section II V1.0, Section III V0.6)
0.7	Section update (Section I V0.0, Section II V1.3, Section III V0.7)
1.0	Section update (Section I V1.0, Section II V1.5, Section III V1.0)
1.1	Section update (Section I V1.1, Section II V1.6, Section III V1.1)
1.2	Section update (Section I V1.2, Section II V1.7, Section III V1.2)
1.2.1	Section I update V1.2.1
1.3	Section update (Section I V1.3, Section II V1.8, Section III V1.3), document organization added
1.4	Section update (Section I V1.6, Section II V2.0, Section III V1.4)
1.5	Section update (Section I V1.7, Section II V2.1, Section III V1.5)
1.6	Section update (Section II V2.2, Section III V1.6) Update to ET1100-0002
1.7	Section update (Section III V1.7)
1.8	Section update (Section I V1.9, Section II V2.4, Section III V1.8)

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,
ESI EEPROM, Distributed Clocks, etc.**

Version 1.9
Date: 2010-05-03

BECKHOFF

DOCUMENT HISTORY

Version	Comment
1.0	Initial release
1.1	• 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 ESI EEPROM is successfully loaded • Editorial changes
1.2	• 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	• Chapter 5.1.1: incompatible PHYs in footnote 1 deleted
1.3	• 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	• 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	• 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	• 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	• 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 • ESI 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	• 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	• 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

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

µC	Microcontroller
ADR	Address
ADS	Automation Device Specification (Beckhoff)
AL	Application Layer
APRD APWR APRW ARMW	Auto Increment Physical Read Auto Increment Physical Write Auto Increment Physical ReadWrite 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 BWR BRW	Broadcast Read Broadcast Write 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 ESC configuration and device description. Connected to the ESI interface.
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 ESI EEPROM (formerly known as SII)
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 FPWR FPRW FRMW	Configured Address Physical Read Configured Address Physical Write Configured Address Physical ReadWrite 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 GPO	General Purpose Input General Purpose Output
HW	Hardware
I²C	Inter-Integrated Circuit, serial bus used for ESI 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 LWR LRW	Logical Read Logical Write 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, replaced by ESI 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:

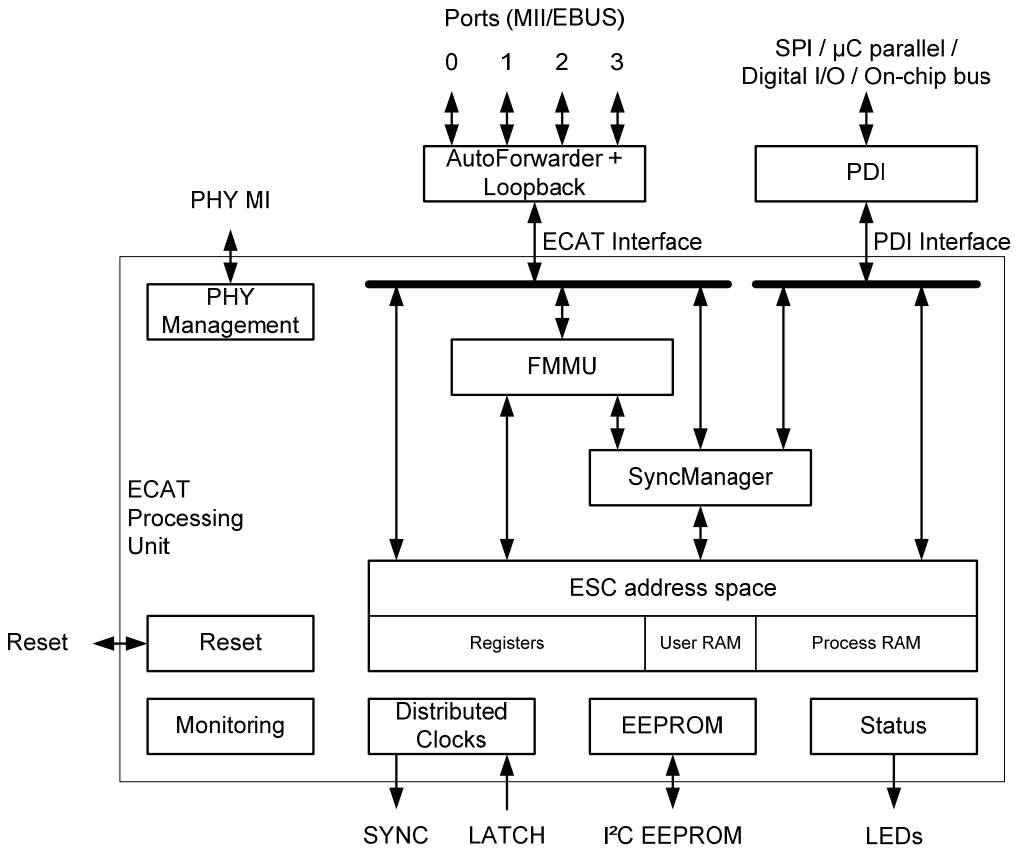


Figure 1: EtherCAT Slave Controller Block Diagram

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 and 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.

ESI EEPROM
One non-volatile memory is needed for ESC configuration and device description, 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 (V2.3.2)
- EtherCAT IP Core for Xilinx® FPGAs (V2.03c)
- 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 protocols[1].

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.

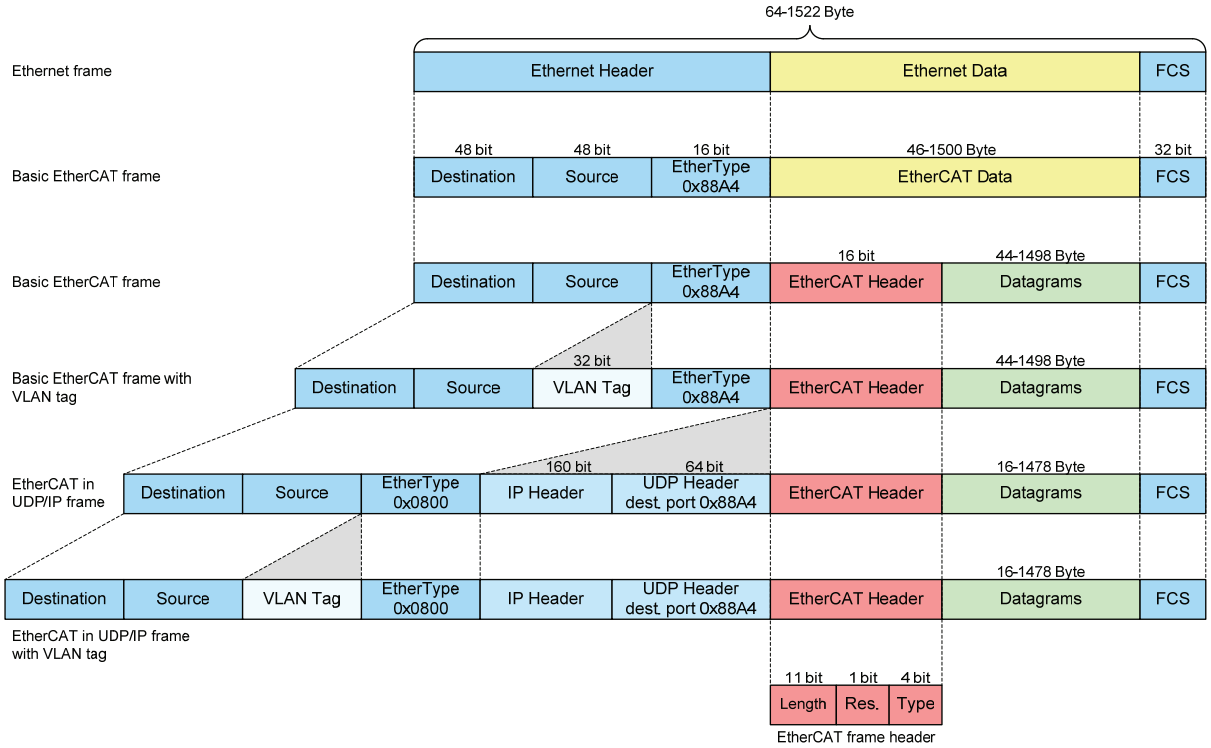


Figure 2: Ethernet Frame with EtherCAT Data

[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.

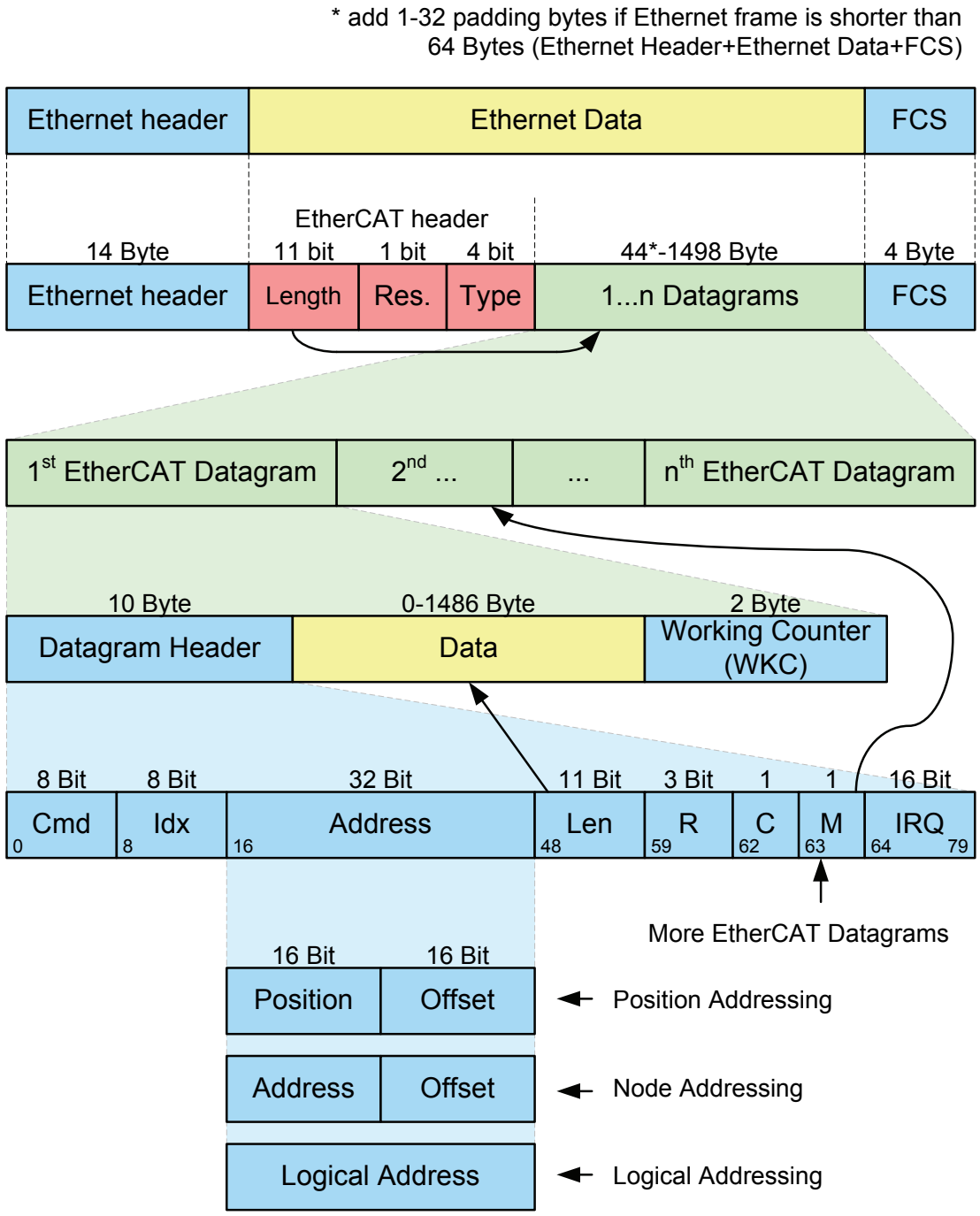


Figure 3: EtherCAT Datagram

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 datagrams 0: Last EtherCAT datagram 1: More EtherCAT datagrams 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 ESI 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 ESI 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		High Add. In+1	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. It

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.

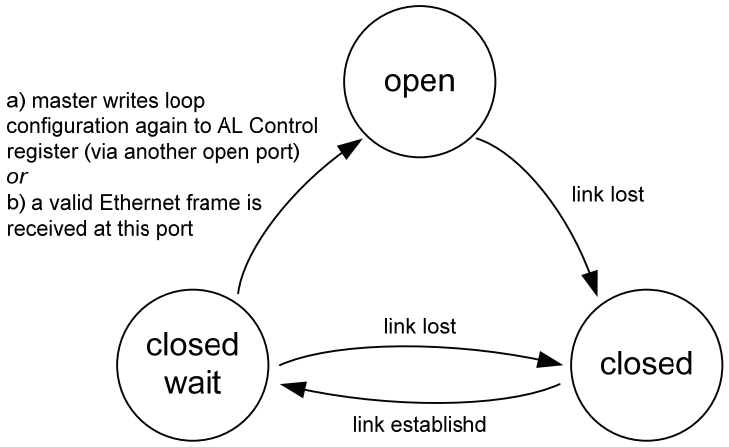


Figure 4: Auto close loop state transitions

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:

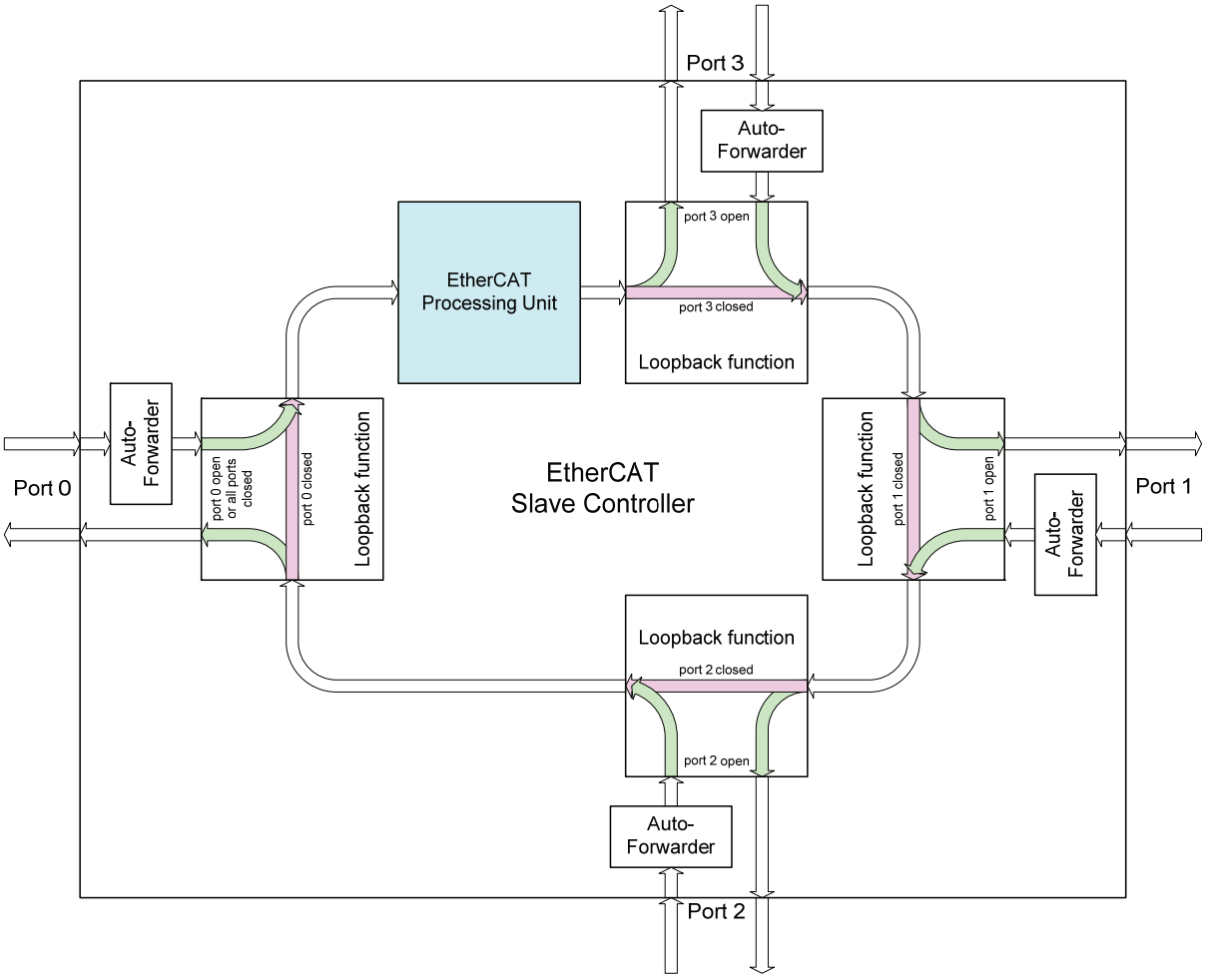


Figure 5: Frame Processing

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 ESI 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:

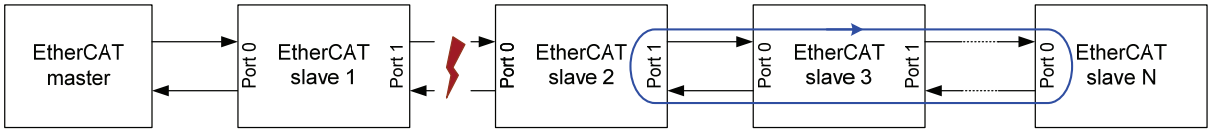


Figure 6: Circulating Frames

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. The Circulating bit of each frame is set at slave 3. Slave 1 detects this and destroys the frames.

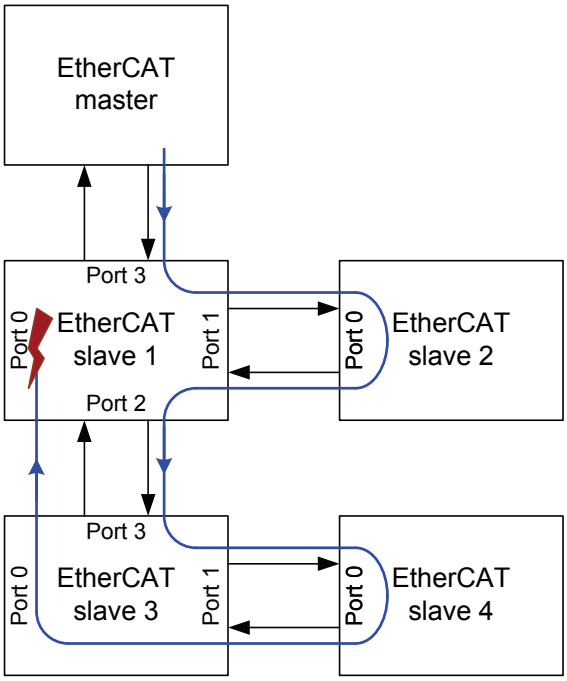


Figure 7: All frames are dropped because of Circulating Frame Prevention

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

<table>
<tr><th rowspan="3">Status register</th><th colspan="4">MII</th><th colspan="2" rowspan="2">EBUS</th></tr>
<tr><th colspan="2">LINK_MII signal</th><th colspan="2">Management Interface</th></tr>
<tr><th>Standard link detection</th><th>Enhanced link detection</th><th>Standard link detection</th><th>Enhanced link detection</th><th>Standard link detection</th><th>Enhanced link detection</th></tr>
<tr><td>ESC DL Status: Physical link 0x0110[7:4]</td><td colspan="2">LINK_MII signal state</td><td colspan="2">LINK_MII signal combined with MI Link Detection and Configuration result</td><td colspan="2">Result of the standard link detection mechanism</td></tr>
<tr><td>ESC DL Status: Communication established 0x0110[15,13,11,9]</td><td>LINK_MII signal state</td><td>LINK_MII signal state combined with RX_ERR threshold state</td><td>LINK_MII signal combined with MI Link Detection and Configuration result</td><td>LINK_MII signal combined with RX_ERR threshold state and MI Link Detection and Configuration result</td><td>Result of the standard link detection mechanism.</td><td>Result of the enhanced link detection mechanism.</td></tr>
<tr><td>PHY port status: physical link status 0x0518.0, 0x0519.0, 0x051A.0, 0x051B.0</td><td colspan="2" rowspan="2">n.a.</td><td colspan="2">PHY has detected link (PHY Status register 1.2)</td><td colspan="2" rowspan="2">n.a.</td></tr>
<tr><td>PHY port status: Link status 0x0518.1, 0x0519.1, 0x051A.1, 0x051B.1</td><td colspan="2">PHY has detected link, link is suitable for ECAT</td></tr>
</table>

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 ESI 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 ESI 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 RMII[1]) 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 operation[2].
- 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).

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

[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.

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 signals[1] shown in Figure 8:

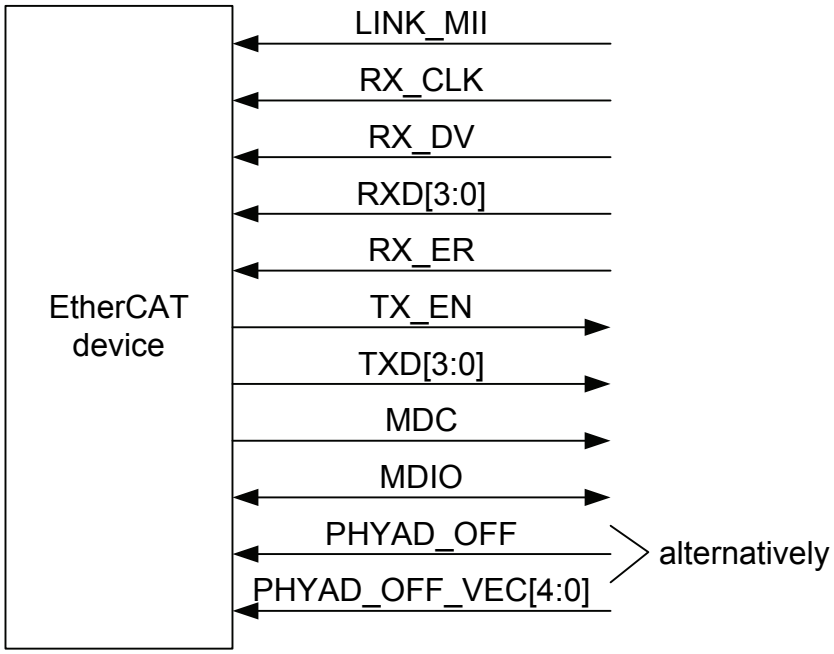


Figure 8: MII Interface signals

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

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 signals[1] shown in Figure 9:

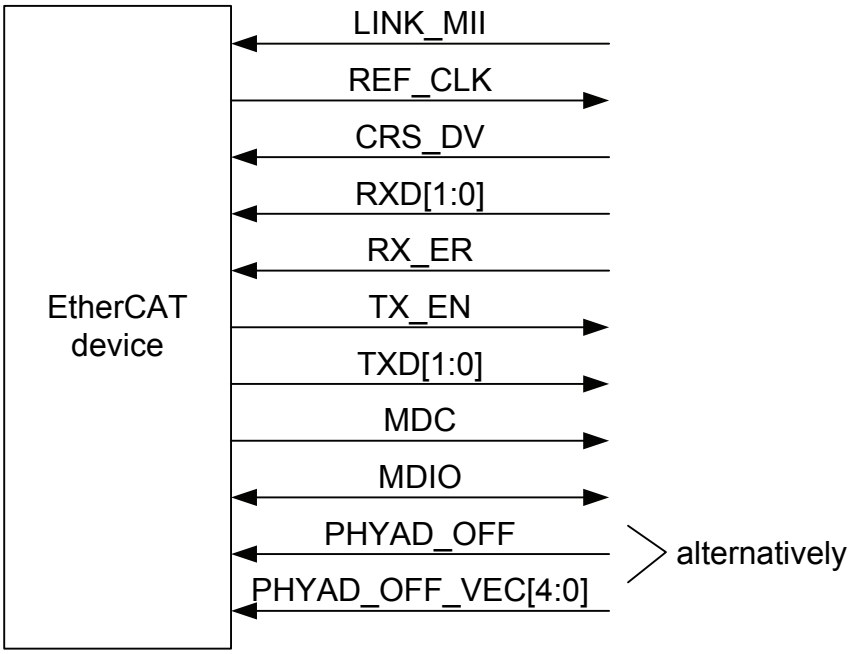


Figure 9: RMII Interface signals

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 looses 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 to 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 accesses 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_OFF). 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 can not 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 can not 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 EtherCAT slave device description.

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 registers[1].

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

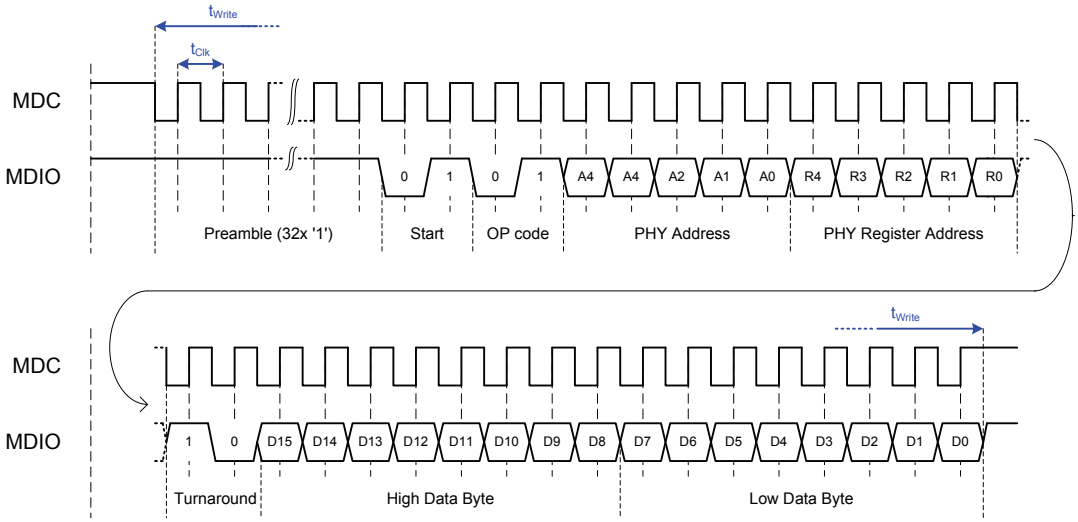


Figure 10: Write access

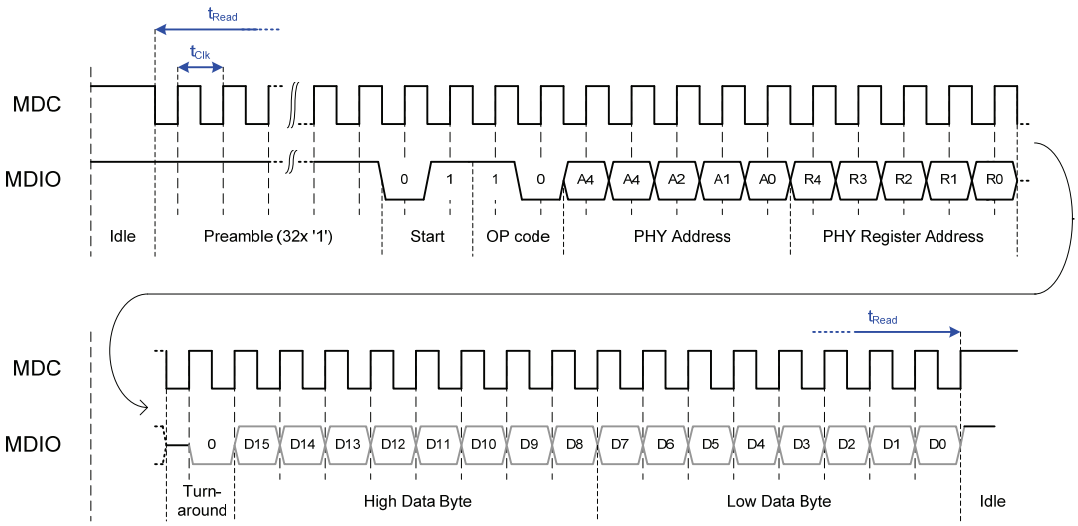


Figure 11: Read access

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 10nF/500V capacitor in parallel to a 1MΩ resistor. Shield is also connected to PE by a 10nF/500V capacitor in parallel to a 1MΩ resistor. Especially the values for the connection between Virtual GND and PE are subject to change to meet industrial requirements (EMC).

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

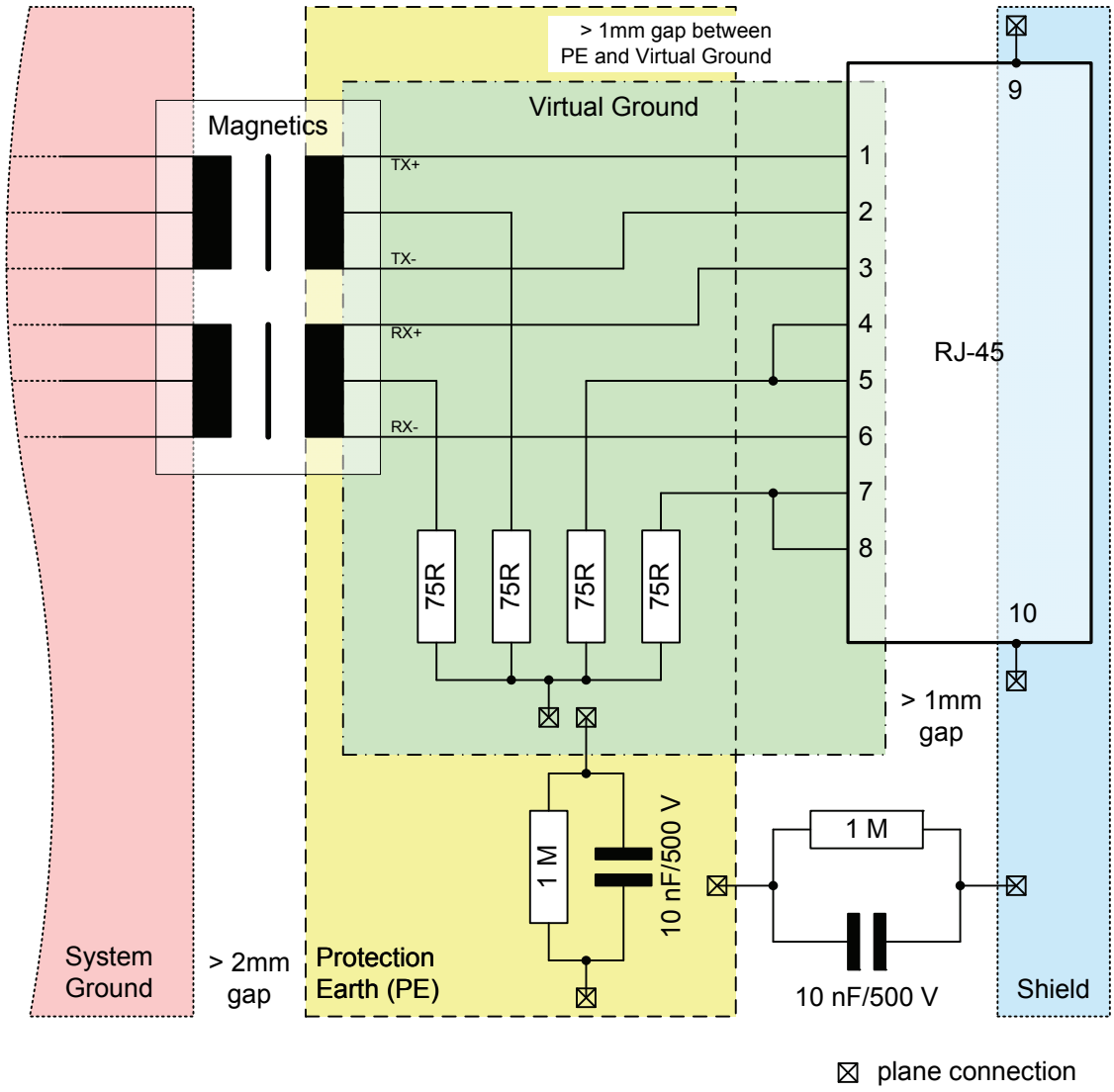


Figure 12: Termination and Grounding Recommendation

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.

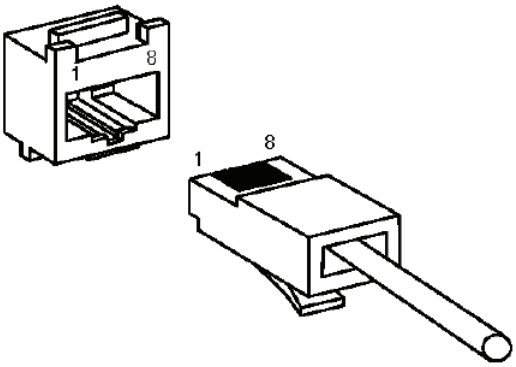


Figure 13: RJ45 Connector

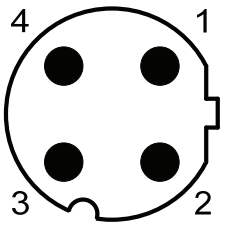


Figure 14: M12 D-code Connector

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.

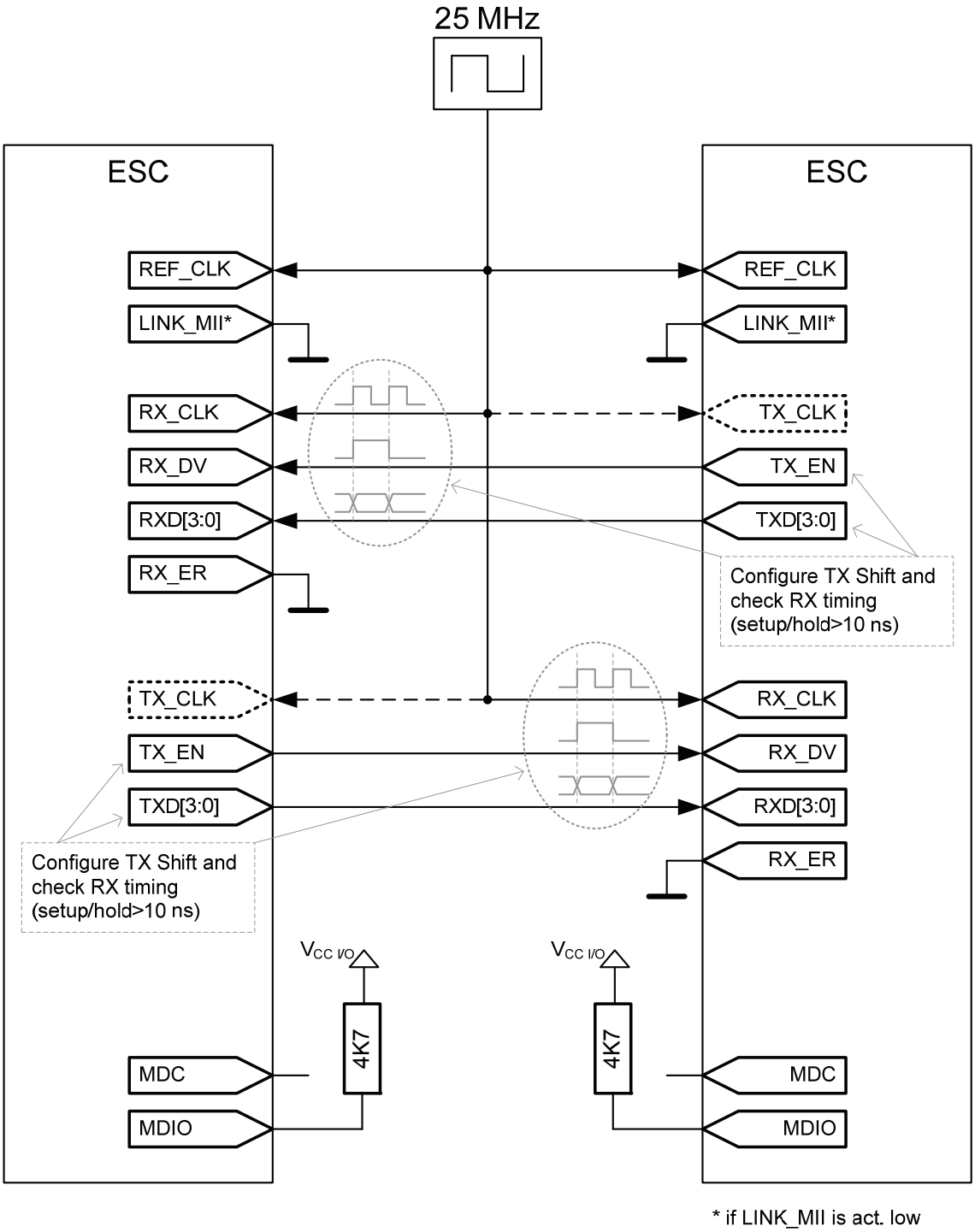


Figure 15: Back-to-Back MII Connection (two ESCs)

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.

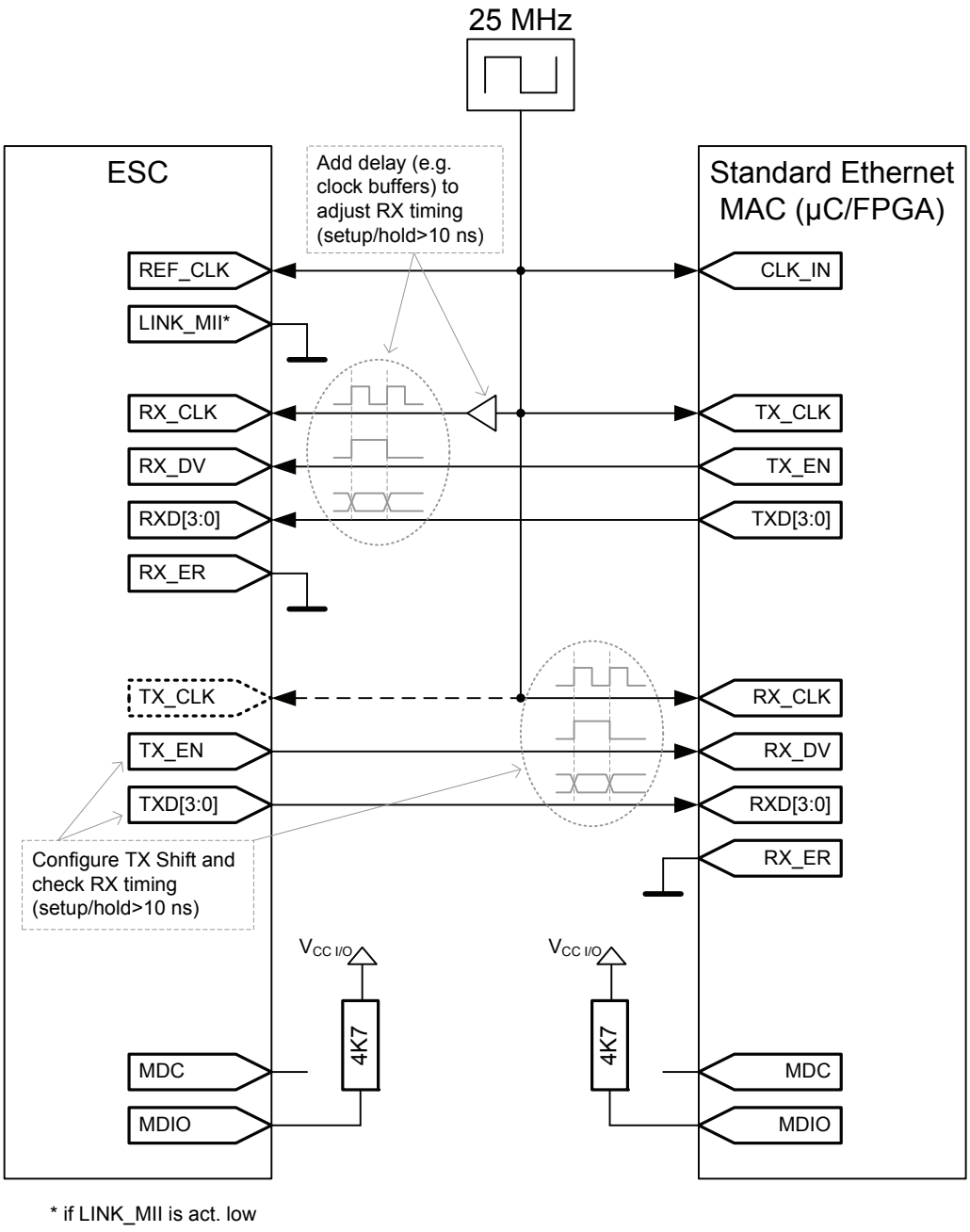


Figure 16: Back-to-Back MII Connection (ESC and standard MAC)

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:

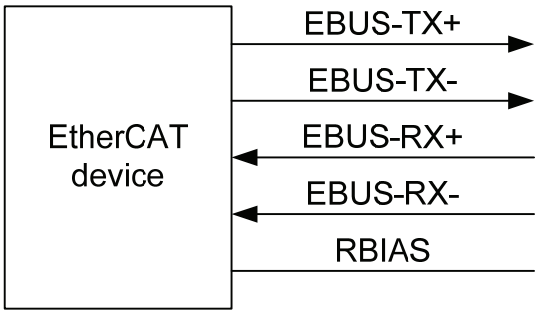


Figure 17: EBUS Interface Signals

Table 21: EBUS Interface signals

Signal	Direction	Description
EBUS-TX+ EBUS-TX-	OUT	EBUS/LVDS transmit signals
EBUS-RX+ EBUS-RX-	IN	EBUS/LVDS receive signals
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.

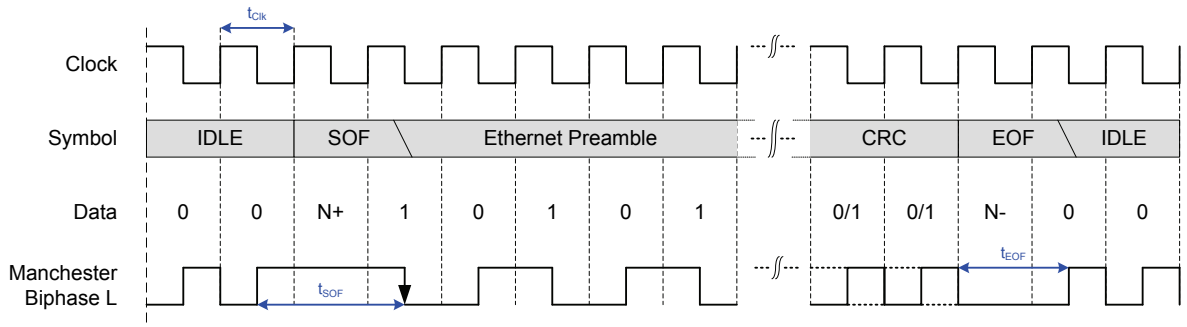


Figure 18: EBUS Protocol

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).

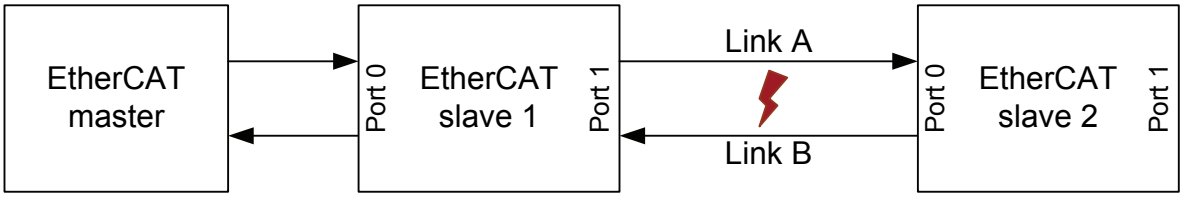


Figure 19: Example EtherCAT Network

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 ESI 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.

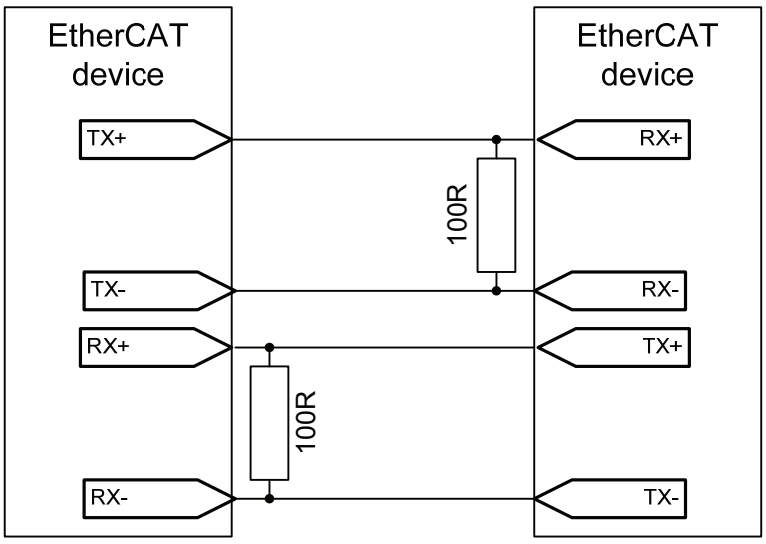


Figure 20: EBUS Connection

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.

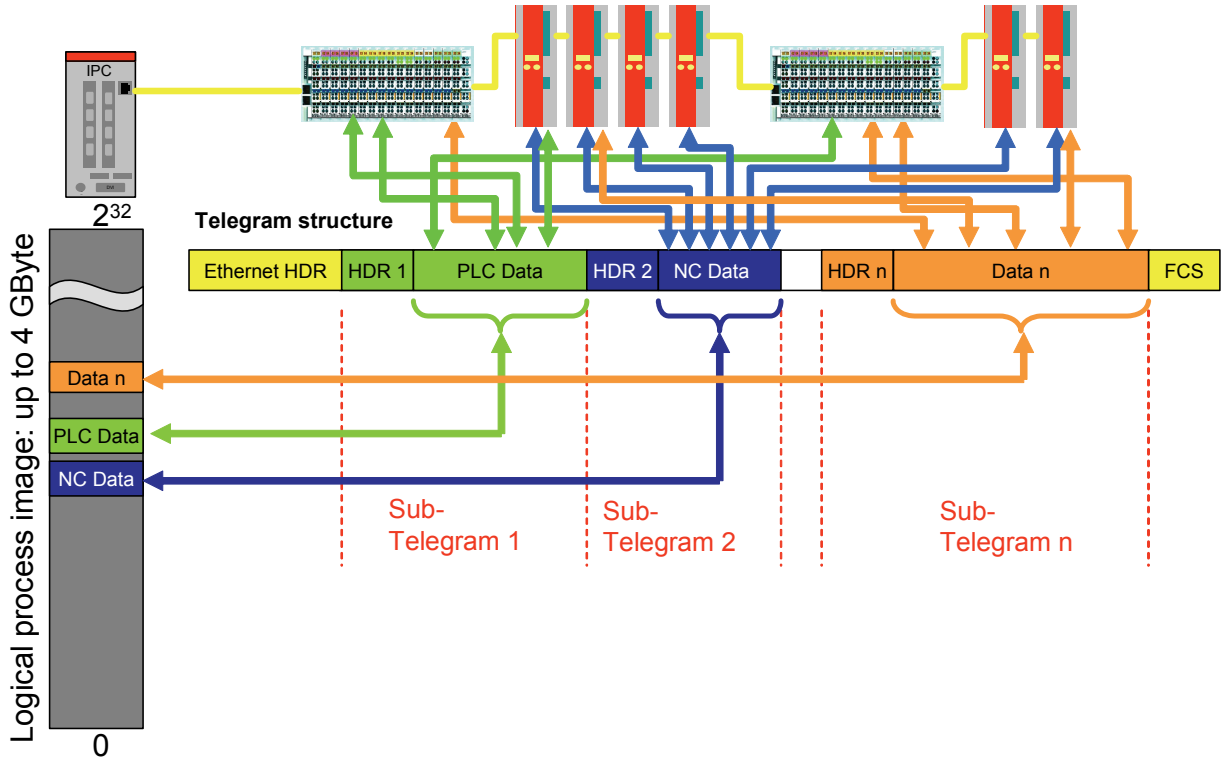


Figure 21: FMMU Mapping Principle

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.

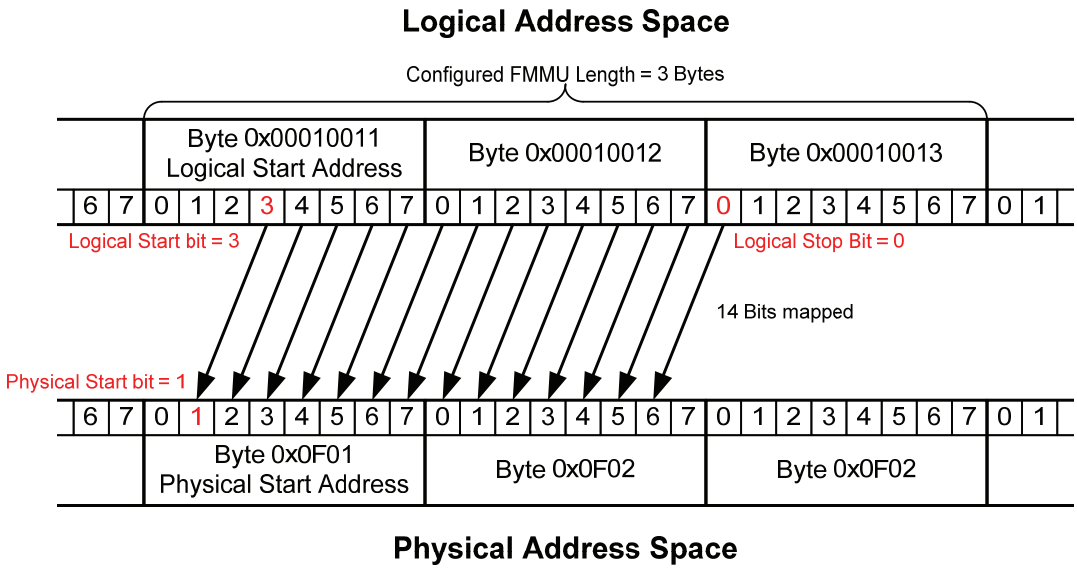


Figure 22: FMMU Mapping Example

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 know 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.

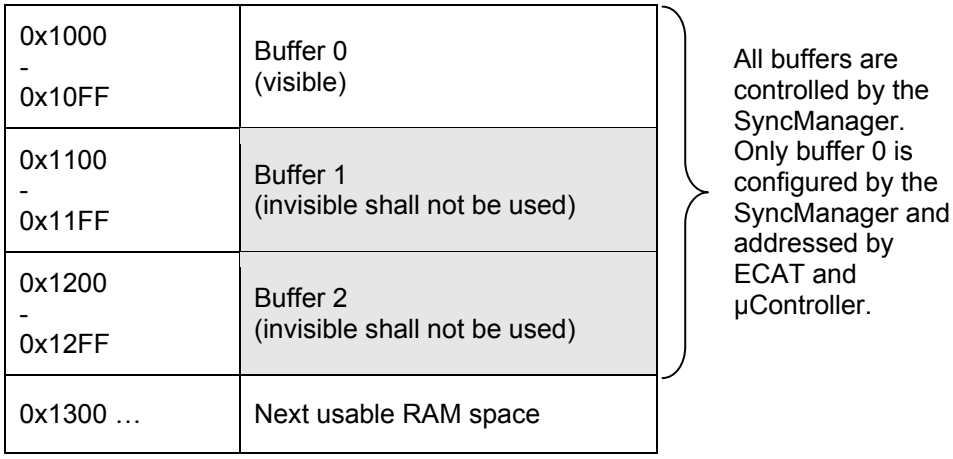


Figure 23: SyncManager Buffer allocation

The buffer interaction is shown in Figure 24:

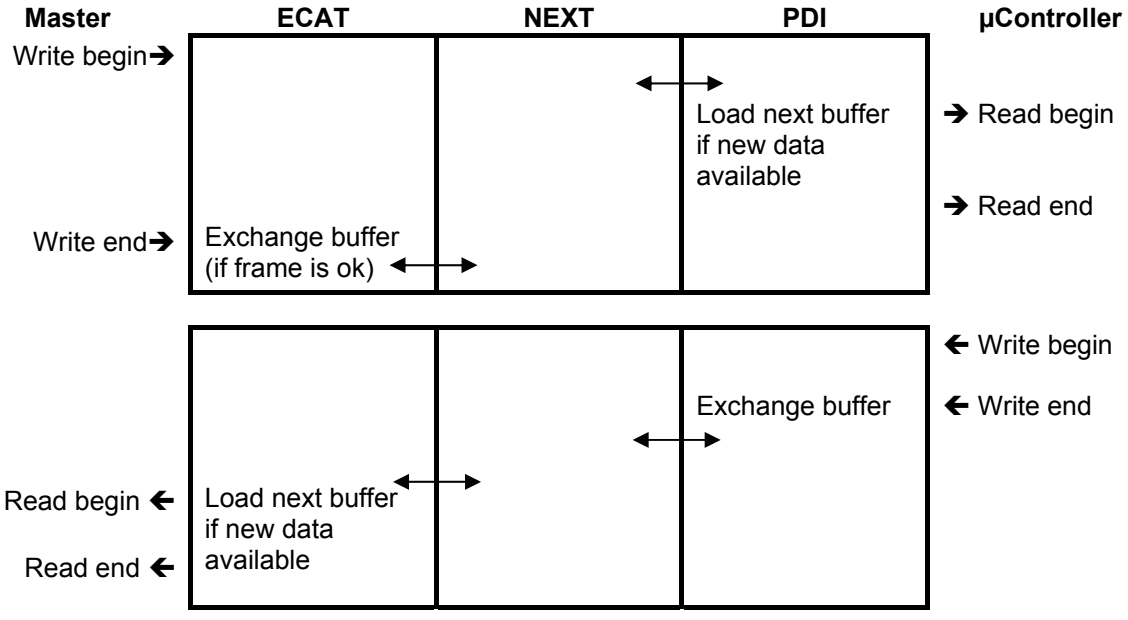


Figure 24: SyncManager Buffered Mode Interaction

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.

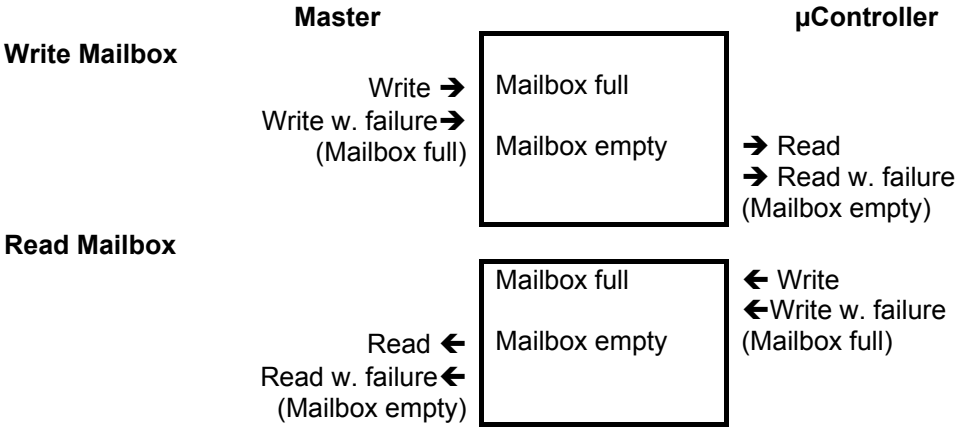


Figure 25: SyncManager Mailbox Interaction

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.

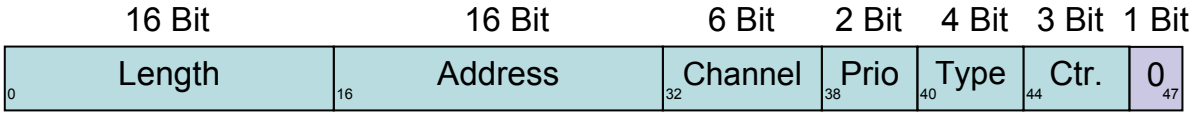


Figure 26: EtherCAT Mailbox Header (for all Types)

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 (CANopen 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 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.

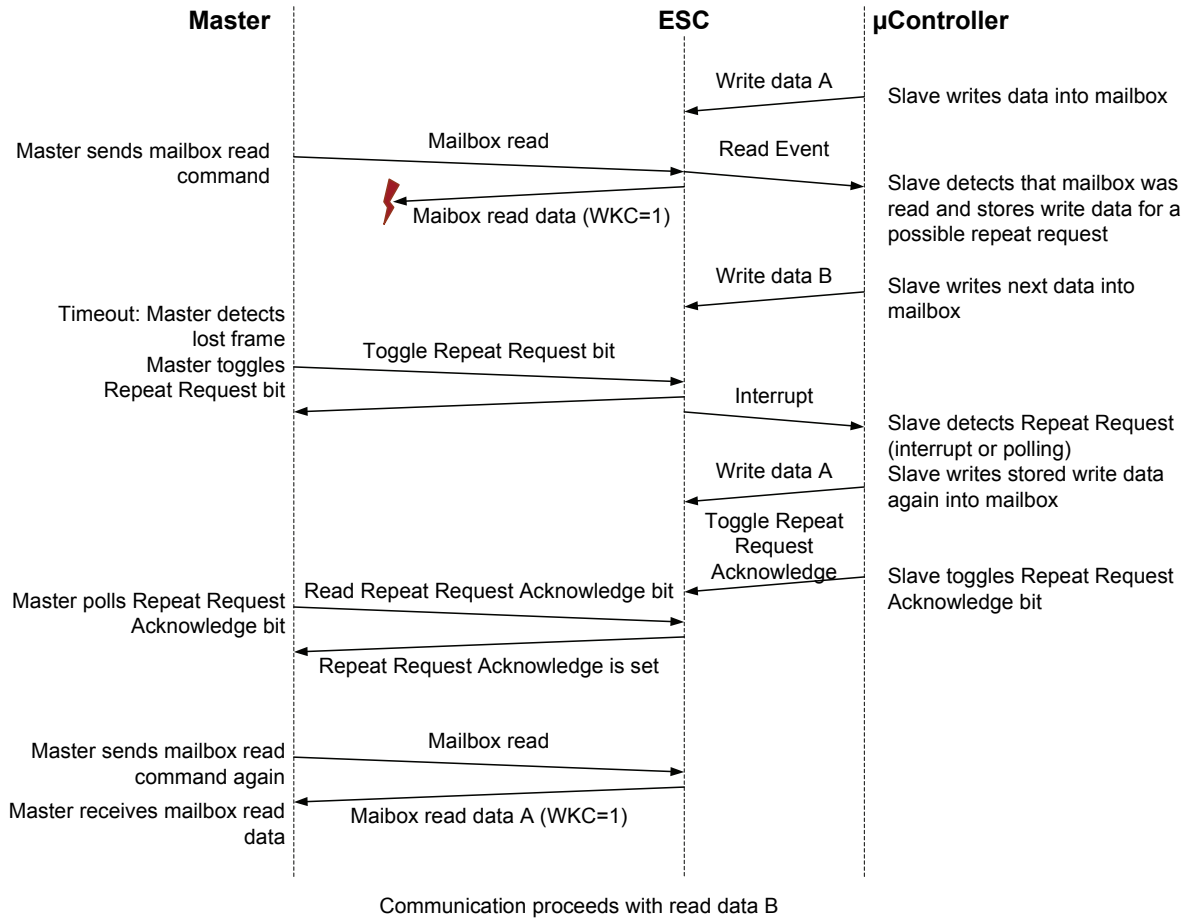


Figure 27: Handling of Write/Read Toggle with Read Mailbox

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.

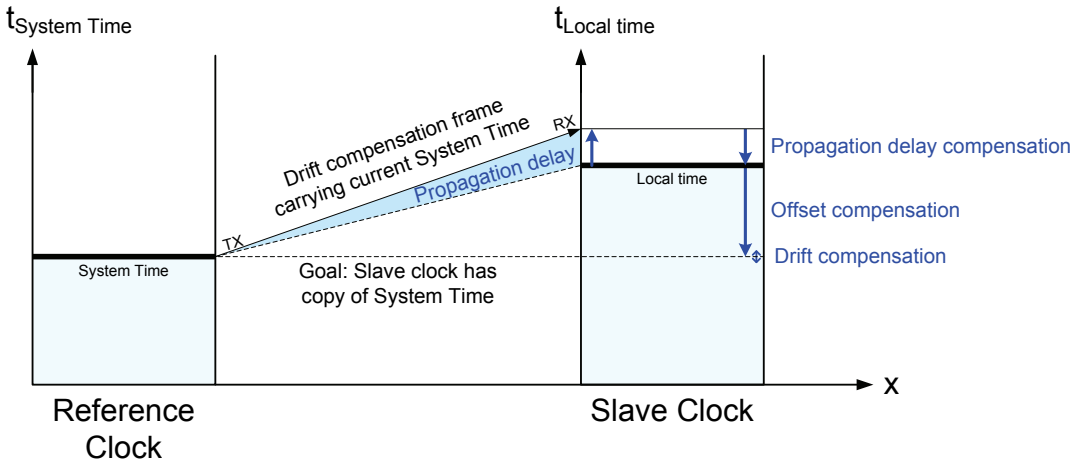
9.1.1 Clock Synchronization Process

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

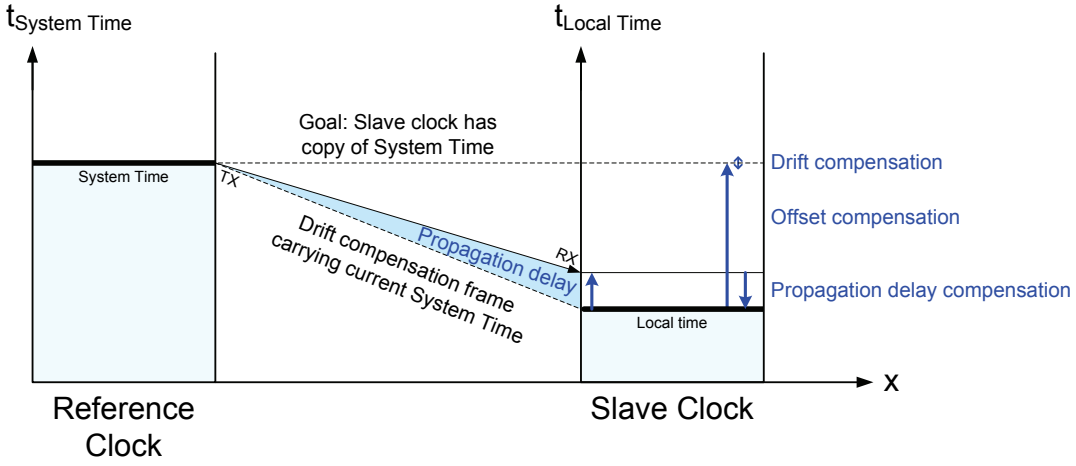


Figure 28: Propagation Delay, Offset, and Drift Compensation

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 no 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.

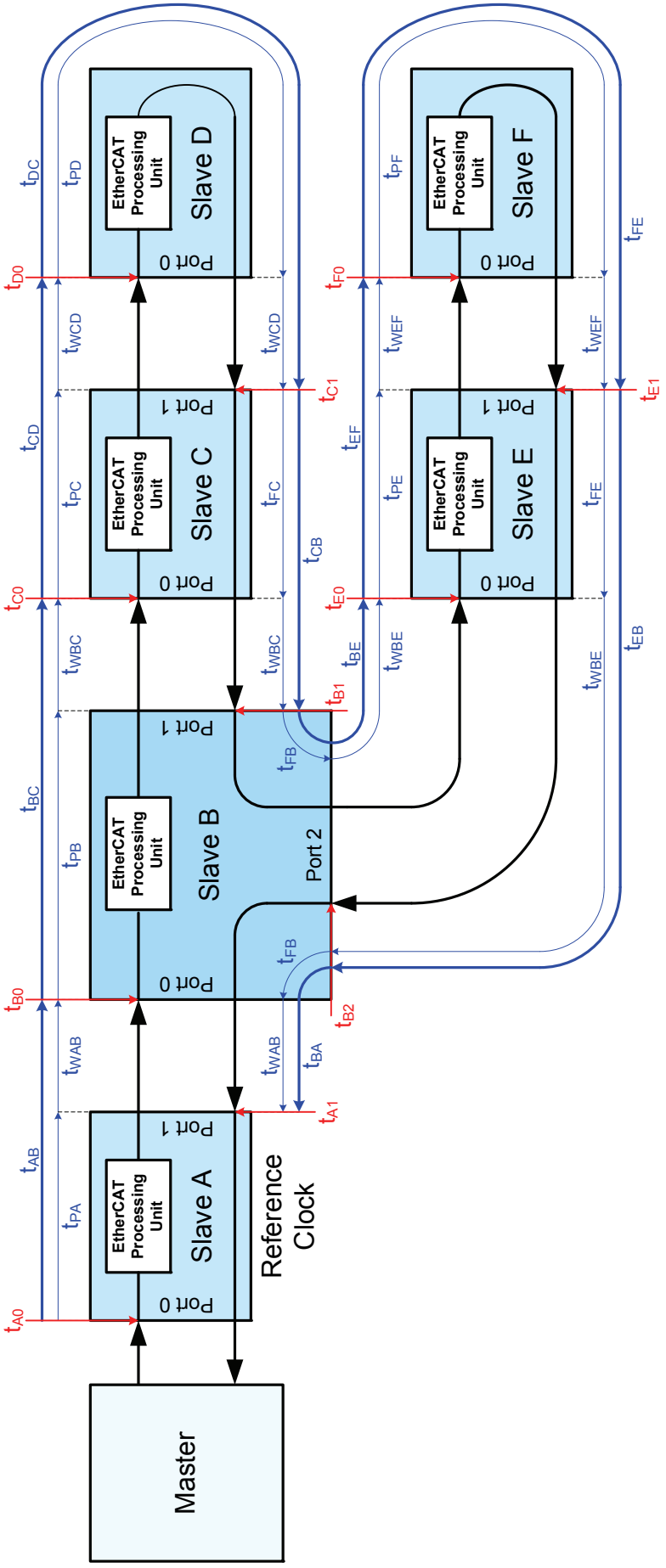


Figure 29: Propagation Delay Calculation

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 device description. 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 quarts and local quarts) 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:

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

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

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 Reverence 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 Reverence 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):

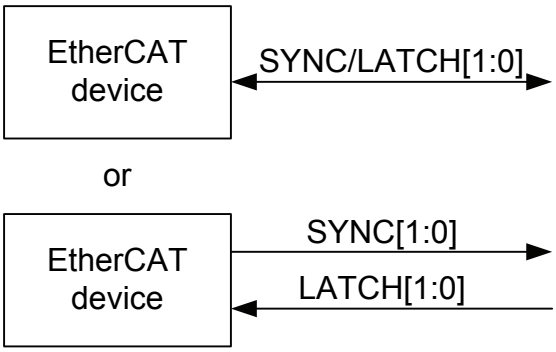


Figure 30: Distributed Clocks signals

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 ESI 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.

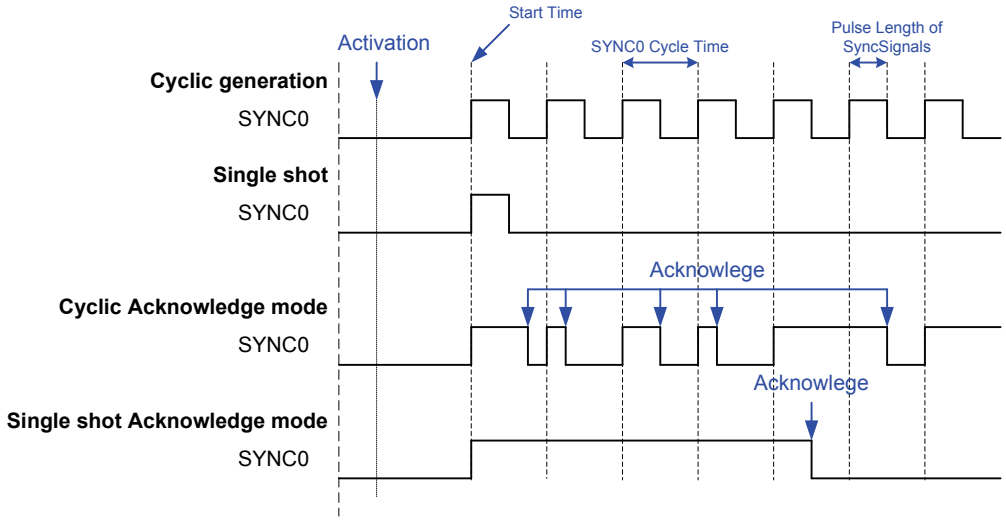


Figure 31: SyncSignal Generation Modes

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 ESI 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:

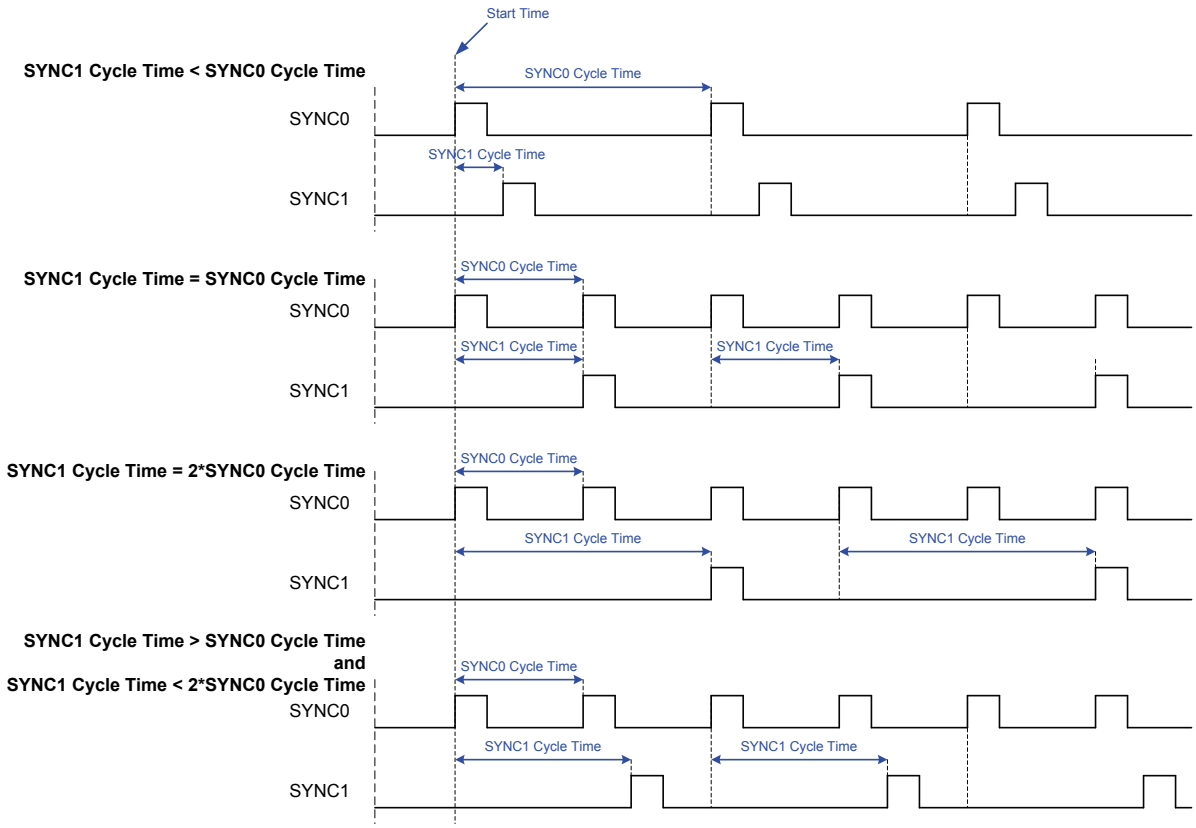


Figure 32: SYNC0/1 Cycle Time Examples

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 ESI 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 Device Description File)
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 ESI 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 µController/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 µController 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 µController 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.

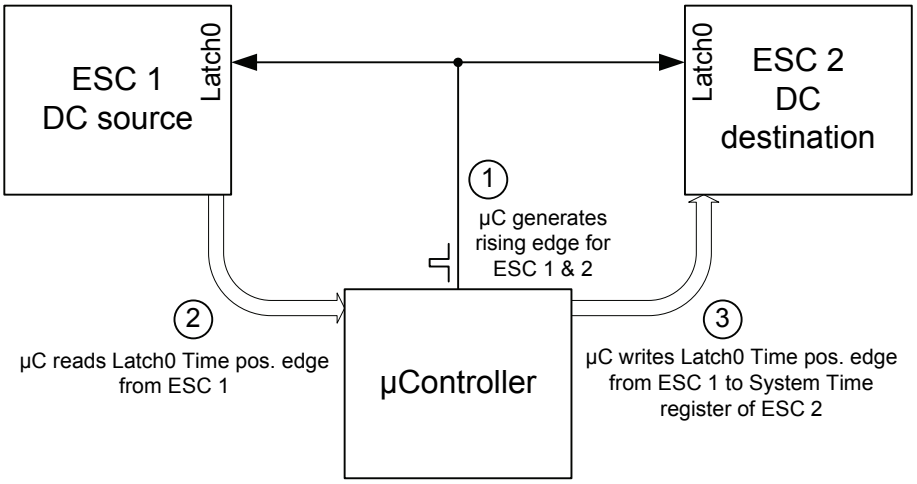


Figure 33: System Time PDI Controlled with three steps

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.

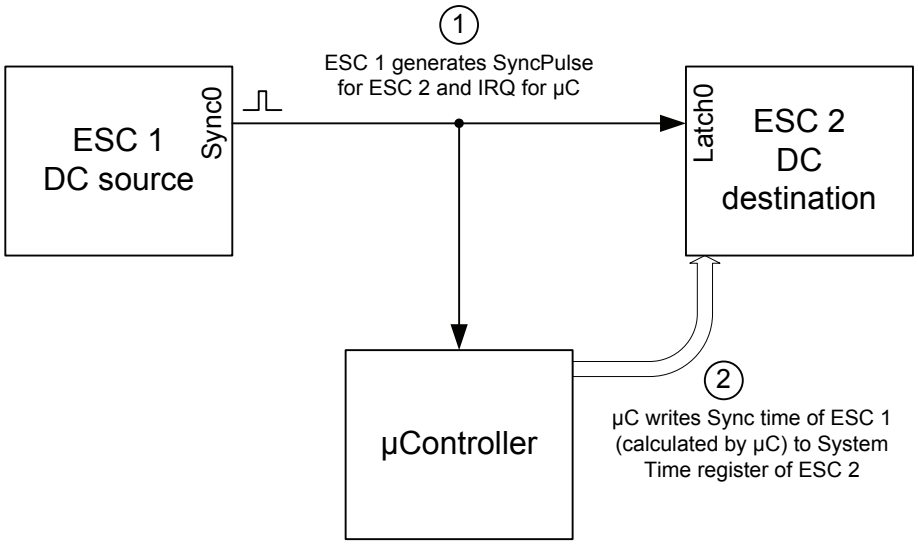


Figure 34: System Time PDI Controlled with two steps

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.

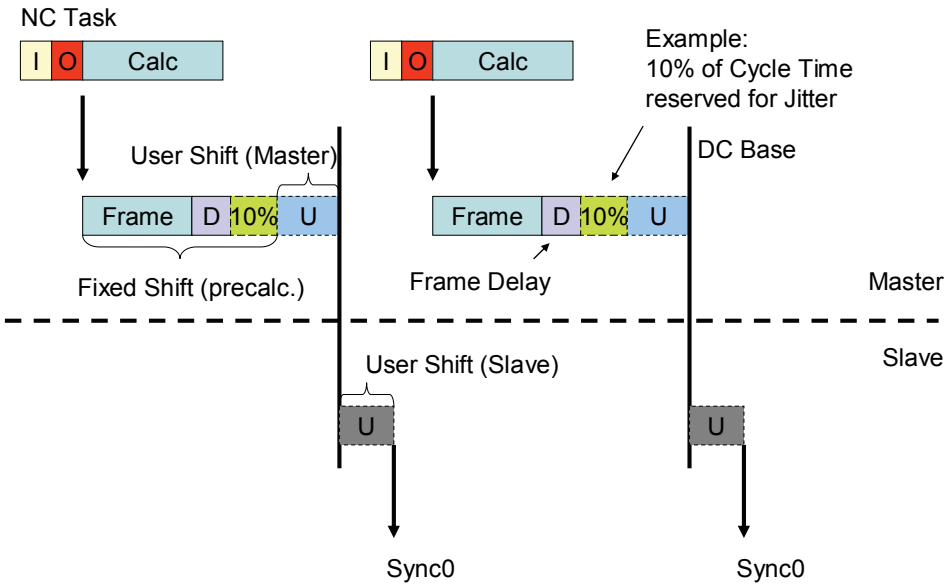


Figure 35: DC Timing Signals in relation to Communication

IO(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:

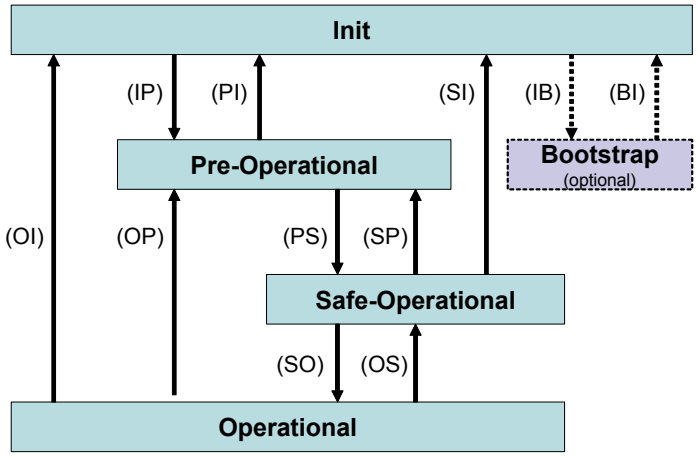


Figure 36: EtherCAT State Machine

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 ESI 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.ethercat.org, Guidelines and Protocol Enhancements).

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

Code	Description	Current state (or state change)	Resulting state
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, P→S	P + E
0x0019	No valid outputs	O, S→O	S + E
0x001A	Synchronization error	O, S→O	S + E
0x001B	Sync manager watchdog	O, S	S + E
0x001C	Invalid Sync Manager Types	O, S P→S	S + E P + E
0x001D	Invalid Output Configuration	O, S P→S	S + E P + 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	Free-Run not supported	P→S	P + E
0x0028	Synchronization not supported	P→S	P + E
0x0029	Free-Run 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
0x0030	Invalid DC SYNCH Configuration	O, S	S + E
0x0031	Invalid DC Latch Configuration	O, S	S + E

Code	Description	Current state (or state change)	Resulting state
0x0032	PLL Error	O, S	S + E
0x0033	Invalid DC IO Error	O, S	S + E
0x0034	Invalid DC Timeout Error	O, S	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	• No communication on Application Layer • Master has access to the DL-Information registers
INIT TO PREOP	• Master configures registers, at least: - DL Address register - SyncManager channels for Mailbox communication • Master initializes DC clock synchronization • Master requests ‘Pre-Operational’ state - Master sets AL Control register • wait for AL Status register confirmation
PREOP	• Mailbox communication on the Application Layer • No Process Data communication
PREOP TO SAFEOP	• Master configures parameters using the Mailbox: - e.g., Process Data Mapping • Master configures DL Register: - SyncManager channels for Process Data communication - FMMU channels • Master requests ‘Safe-Operational’ state • wait for AL Status register confirmation
SAFEOP	• Mailbox communication on the Application Layer • Process Data communication, but only Inputs are evaluated – Outputs remain in ‘Safe’ state
SAFEOP TO OP	• Master sends valid Outputs • Master requests ‘Operational’ state (AL Control/Status) • wait for AL Status register confirmation
OP	• Inputs and Outputs are valid
BOOT	Optional, but recommended if firmware updates are necessary. • 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 ESI EEPROM

EtherCAT slave controllers use a mandatory NVRAM (typically a serial EEPROM with I²C interface) to store device configuration and device description. 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.

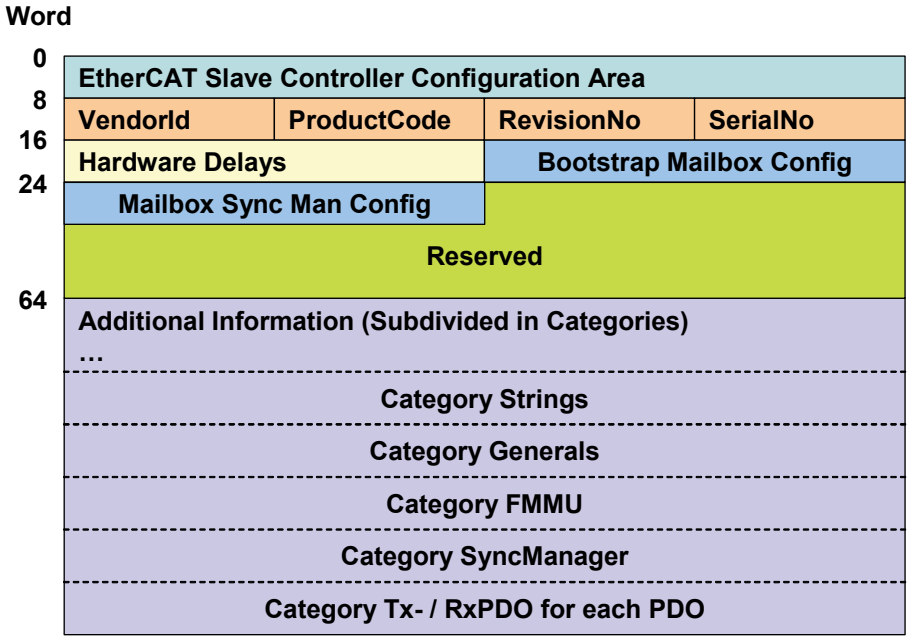


Figure 37: ESI EEPROM Layout

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 ESI 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 "EtherCAT Slave Device Description" specification, available from the EtherCAT Technology Group (http://www.ethercat.org), and to the EtherCAT Knowledge Base, also available at the EtherCAT Technology Group website (http://www.ethercat.org, Guidelines and Protocol Enhancements).

11.1 ESI 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.

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 ESI EEPROM content following the ESC Configuration area is shown in Table 40. For more information, refer to the EtherCAT Knowledge Base available at the EtherCAT Technology Group website (http://www.ethercat.org, Guidelines and Protocol Enhancements).

Table 40: ESI 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 ESI EEPROM Logical Interface

The ESI 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: ESI 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 ESI 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 0x0012:0x0013and PDI Control Bit 0x140.9 – enhanced link detection).

The ESI EEPROM interface has these error status bits:

Table 42: ESI EEPROM Interface Errors

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

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 ESI 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 ESI 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 Enables (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 ESI EEPROM Electrical Interface (I^2C)

The ESI 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 ESI EEPROM interface has the following signals[1]:

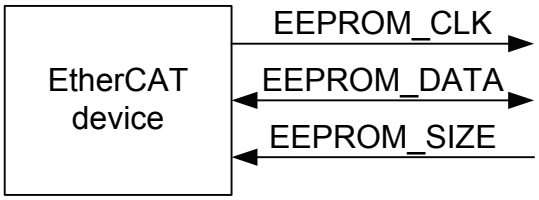


Figure 38: I²C EEPROM signals

Table 43: I²C EEPROM signals

Signal	Direction	Description
EEPROM_CLK	OUT	I²C clock
EEPROM_DATA	BIDIR	I²C 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. I²C 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 EERPOM 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 EERPOM:

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 EERPOM. At first, the address is written to the EEPROM, then the data is read (N=3 or N=7):

Table 47: I²C 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

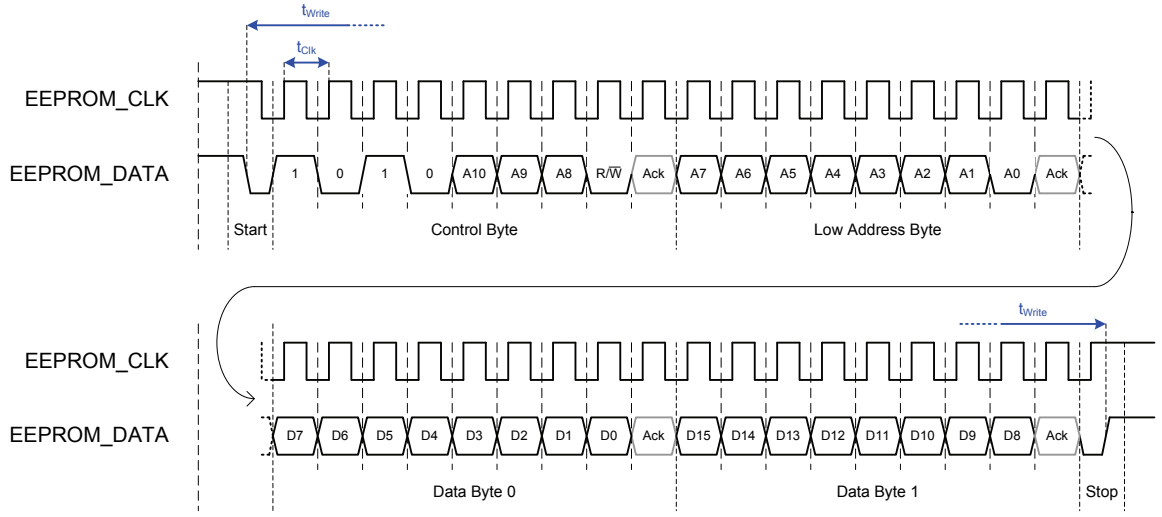


Figure 39: Write access (1 address byte, up to 16 kBit EEPROMs)

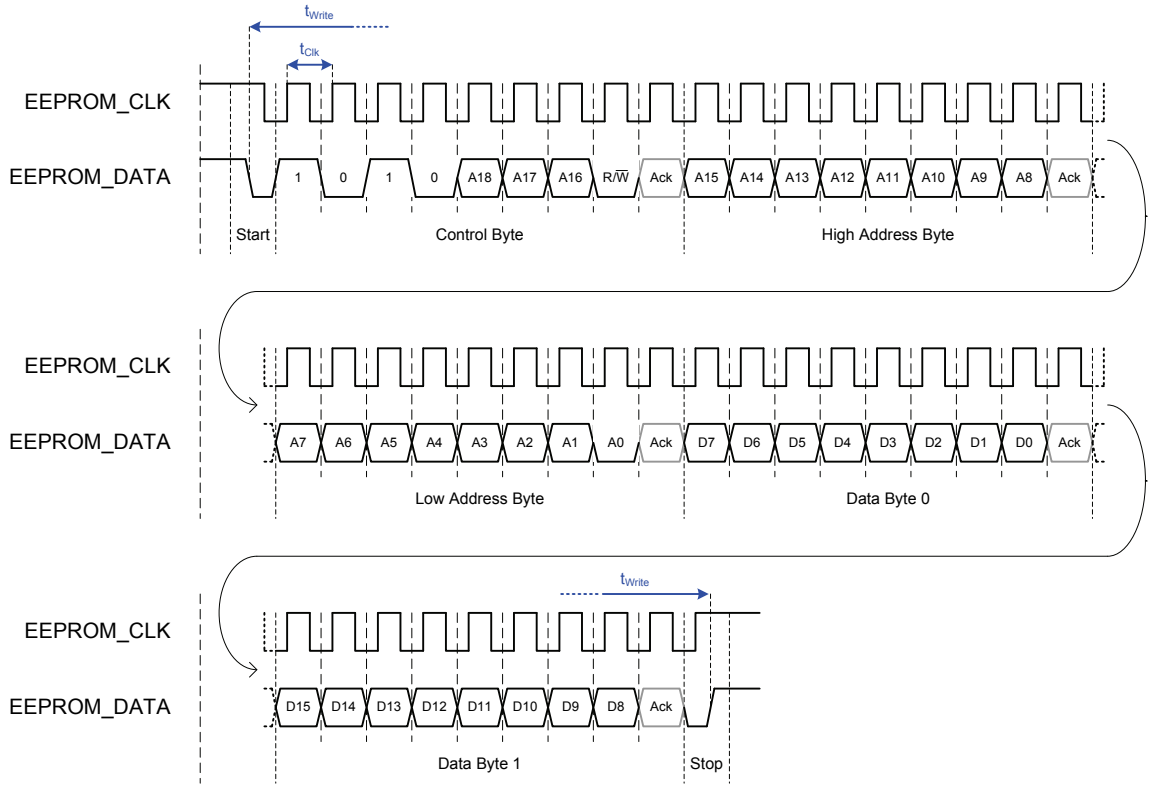


Figure 40: Write access (2 address bytes, 32 kBit - 4 MBit EEPROMs)

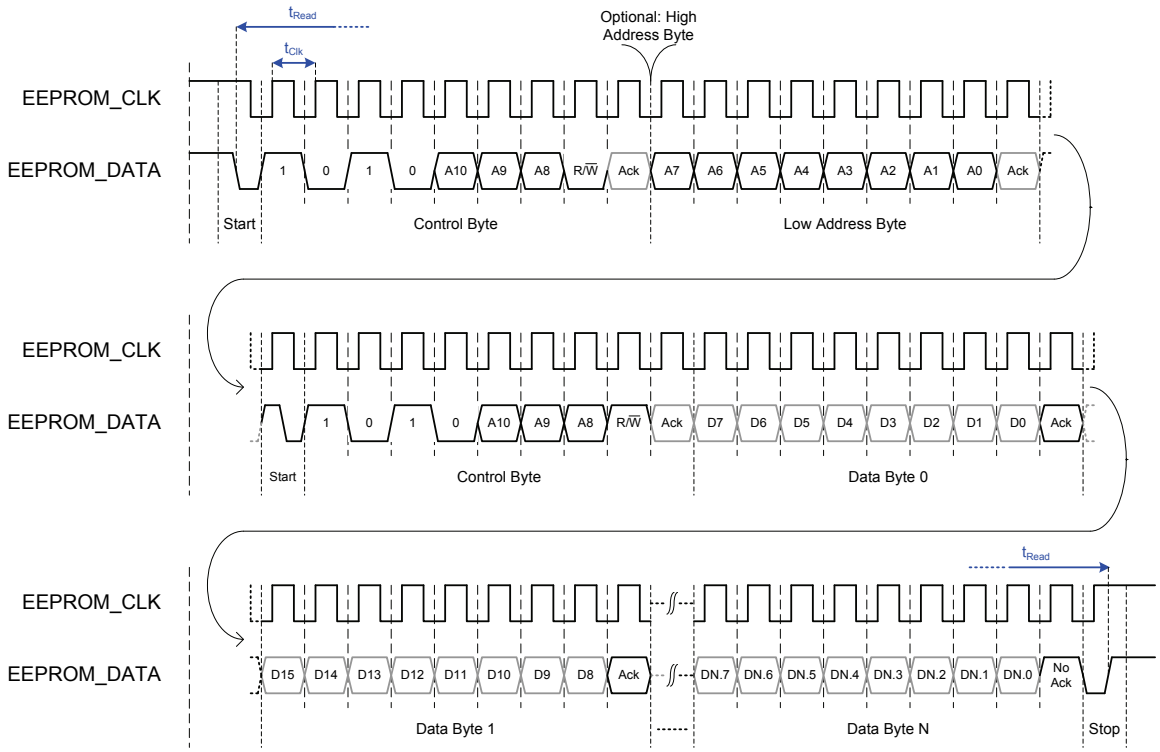


Figure 41: Read access (1 address byte, up to 16 kBit EEPROMs)

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.

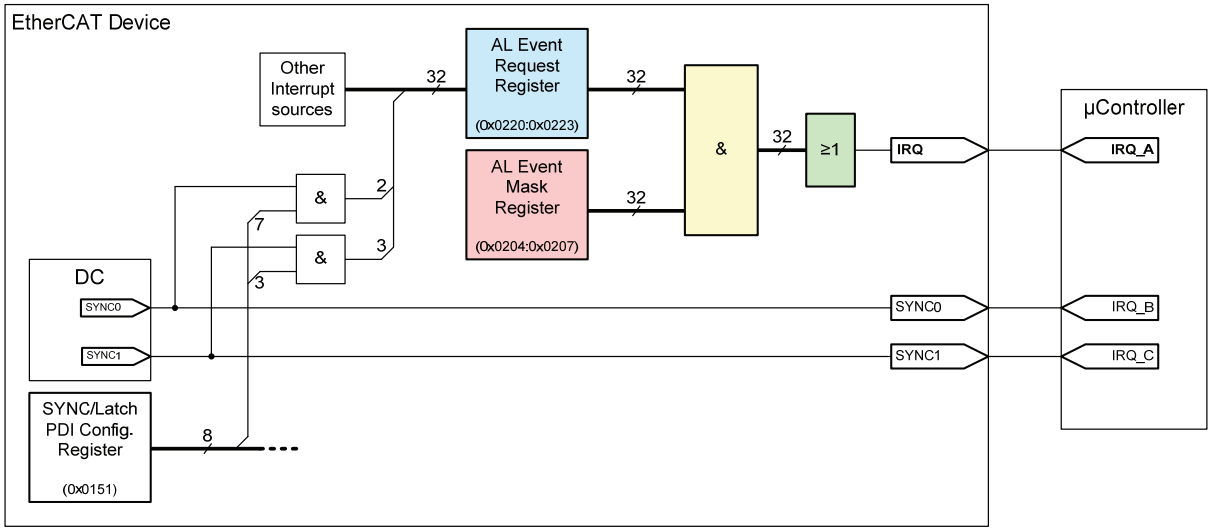


Figure 42: PDI Interrupt Masking and interrupt signals

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 ESI 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.

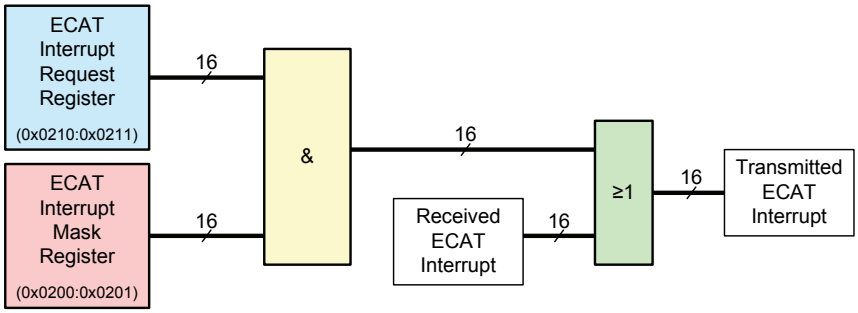


Figure 43: ECAT Interrupt Masking

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):

$$t_{WD\_Div} = (WD\_DIV + 2) * 40ns$$

$$t_{WD\_PDI} = [t_{WD\_Div} * WD\_PDI ; t_{WD\_Div} * WD\_PDI + t_{WD\_Div}]$$

$$t_{WD\_PD} = [t_{WD\_Div} * WD\_PD ; t_{WD\_Div} * WD\_PD + t_{WD\_Div}]$$

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)
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)	• Physical Layer Errors (RX Errors) • Too long frames (> ~ 2000 Byte) • CRC errors • Frames without Ethernet SOF	• 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 “EtherCAT Indicator Specification”, available from the ETG (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 EtherCAT indicator 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 EtherCAT indicator 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 ESI 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 ESI 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).

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.

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 ESI 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: • 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 do 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

Slave Controller

Section II – Register Description

Register overview and detailed description

Version 2.4
Date: 2010-05-03

BECKHOFF

DOCUMENT HISTORY

Version	Comment
1.0	Initial release
1.1	• 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	• 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	• 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	• Sync/Latch PDI configuration register: Latch configuration clarified • AL Control register: mailbox behavior described • Editorial changes
1.5	• ESC DL Control (0x0100:0x0103): FIFO Size description enhanced • IP Core: Extended Features (reset value of User RAM 0x0F80:0x0FFF) 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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	• 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

CONTENTS

TABLES

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.

Address[1]	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
		ESI 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
0x0F00:0x0F03	4	Digital I/O Output Data
0x0F10:0x0F17	1-8	General Purpose Outputs
0x0F18:0x0F1F	1-8	General Purpose Inputs
		User RAM/Extended ESC features
0x0F80:0x0FFF	128	User RAM/Extended ESC features
		Process Data RAM
0x1000:0x1003	4	Digital I/O Input Data
0x1000:0xFFFF [0x1000:0x13FF] [0x1000:0x1FFF] [0x1000:0x2FFF] [0x1000:0xFFFF]	0-60 KB 1 KB 4 KB 8 KB 1-60 KB	Process Data RAM ET1200 ESC20 ET1100 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 (V2.3.2)
- EtherCAT IP Core for Xilinx® FPGAs (V2.03c)
- 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. ESI 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. ESI 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 V2.3.2/V2.03c Register set S	M	L	ESC20
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	c	c	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	r/c	r/c	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	-	-	-	-
0x0130	5 bits [4:0]	AL Status	x	x	x	x	x	x
0x0130:0x0131	2	AL Status	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	r/c	r/c	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 V2.3.2/V2.03c Register set S	M	L	ESC20
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	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	ESI 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 0x09AE:0x09CF	36	DC – Latch In Unit	x	l	dc	dc	dc	x
0x09F0:0x09F3 0x09F8:0x09FF	12	DC – SyncManager Event Times	-	s/l	c	c	dc	-
0x0E00:0x0EFF	256	ESC specific registers (e.g., Power-On Values / Product and Vendor ID)	x	x	x	x	x	x
0x0F00:0x0F03	4	Digital I/O Output Data	x	x	io	io	io	x
0x0F10:0x0F17	8	General Purpose Outputs [Byte]	2	2	0-8	0-8	0-8	-
0x0F18:0x0F1F	8	General Purpose Inputs [Byte]	-	2	0-8	0-8	0-8	-
0x0F80:0x0FFF	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)
l	Available if DC Latch In Unit enabled (Register 0x0140.11=1)
s/l	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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		~~ESC20~~	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
	Port configuration: 00: Not implemented 01: Not configured (ESI 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)

ESC20	ET1100	ET1200	IP Core
			~~[8]~~ V2.2.0/ V2.02a

Bit	Description	ECAT	PDI	Reset Value
0	FMMU Operation: 0: Bit oriented 1: Byte oriented	r/-	r/-	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)

		ESC20	ET1100	ET1200	IP Core
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)

		ESC20	ET1100	ET1200	IP Core
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)

		ESC20	ET1100	ET1200	IP Core
		~~read~~			~~read~~
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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	
7:1	Reserved, write 0	r/-	r/-	0	

3.12 Write Register Protection (0x0021)

Table 15: Register Write Register Protection (0x0021)

	ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
0	Write register protection: 0: Protection disabled 1: Protection enabled Registers 0x0000-0x0F0F are write protected.	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)

ESC20	ET1100	ET1200	IP Core
~~read~~			~~read~~

Bit	Description	ECAT	PDI	Reset Value
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)

	ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
0	Write protect: 0: Protection disabled 1: Protection enabled	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)

		~~ESC20~~	ET1100	ET1200	IP Core
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)

		~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
					V2.2.0/ V2.02a
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)

ESC20	ET1100	ET1200	IP Core
~~[1]~~			[31:16] config.

Bit	Description	ECAT	PDI	Reset Value
0	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.	r/w	r/-	1
1	Temporary use of settings in Register 0x101: 0: permanent use 1: use for about 1 second, then revert to previous settings	r/w	r/-	0
7:2	Reserved, write 0	r/-	r/-	0
9:8	Loop Port 0: 00: Auto 01: Auto Close 10: Open 11: Closed 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	r/w*	r/-	00
11:10	Loop Port 1: 00: Auto 01: Auto Close 10: Open 11: Closed	r/w*	r/-	00
13:12	Loop Port 2: 00: Auto 01: Auto Close 10: Open 11: Closed	r/w*	r/-	00
15:14	Loop Port 3: 00: Auto 01: Auto Close 10: Open 11: Closed	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)

ESC20 **ET1100** **ET1200** IP Core

Bit	Description	ECAT	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)

		ESC20	ET1100	ET1200	IP Core
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)

	ESC20	ET1100	ET1200	IP Core
	~~[15:5]~~			~~[15:5]~~

Bit	Description	ECAT	PDI	Reset Value
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)

	ESC20	ET1100	ET1200	IP Core
	~~[15:5]~~			~~[15:5]~~

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core
			V1.0.0/ V1.01b

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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
			V2.3.0/ V2.03a

Bit	Description	ECAT	PDI	Reset Value
3:0	LED code: (FSM State:) 0x0: Off (1-Init) 0x1-0xC: Flash 1x – 12x (4-SafeOp 1x) 0xD: Blinking (2-PreOp) 0xE: Flickering (3-Bootrap) 0xF: On (8-Op)	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: 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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
			V2.3.0/ V2.03a

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)

ESC20	ET1100	ET1200	IP Core

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 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)

ESC20	ET1100	ET1200	IP Core
~~[7:1]~~	~~[7:4]~~	~~[7:2]~~	[7:4] V2.2.0/ V2.02a

Bit	Description	ECAT	PDI	Reset Value
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)

~~ESC20~~ **ET1100** **ET1200** IP Core

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)

~~ESC20~~ **ET1100** **ET1200** IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

~~ESC20~~ **ET1100** **ET1200** IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 [7:4]	ET1100	~~ET1200~~	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

~~ESC20~~	ET1100 [1]	~~ET1200~~	IP Core V2.2.0/ V2.02a [1] V2.3.0/ V2.03a

Bit	Description	ECAT	PDI	Reset Value
0	Read BUSY delay: 0: Normal read BUSY output 1: Delayed read BUSY output	r/-	r/-	IP Core: 0 Others: 0, later EEPROM ADR 0x0003
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)

~~ESC20~~ **ET1100** ~~ET1200~~ ~~IP Core~~

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)

~~ESC20~~ **ET1100** ~~ET1200~~ ~~IP Core~~

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)

~~ESC20~~ ~~ET1100~~ **ET1200** ~~IP Core~~

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)

~~ESC20~~ ~~ET1100~~ **ET1200** ~~IP Core~~

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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core

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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
			V1.1.1/ V2.00a

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)

ESC20 ET1100 ET1200 IP Core

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)

ESC20	ET1100	ET1200	IP Core
			write config.

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core
			write config.

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 | ET1100 | ET1200 | IP Core

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 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)	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)	r/-	r/-	0
4 5 … 11	Mirrors values of each SyncManager Status: 0: No Sync Channel 0 event 1: Sync Channel 0 event pending 0: No Sync Channel 1 event 1: Sync Channel 1 event pending … 0: No Sync Channel 7 event 1: Sync Channel 7 event pending	r/-	r/-	0
15:12	Reserved	r/-	r/-	0

3.31 AL Event Request (0x0220:0x0223)

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

ESC20	ET1100	ET1200	IP Core
~~[6:4]~~	~~[6:5]~~	~~[6:5]~~	[5:4] V2.0.0/ V2.00a; [6] V2.3.0/ V2.03a

Bit	Description	ECAT	PDI	Reset Value
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 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)	r/-	r/-	0
3	State of DC SYNC1 (if register 0x0151.7=1): (Bit is cleared by reading of SYNC1 status 0x098F)	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.)	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)	r/-	r/-	0
6	Watchdog Process Data: 0: Has not expired 1: Has expired (Bit is cleared by reading Watchdog Status Process Data 0x0440)	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 9 …. 23	SyncManager interrupts (SyncManager register offset 0x5, bit [0] or [1]): 0: No SyncManager 0 interrupt 1: SyncManager 0 interrupt pending 0: No SyncManager 1 interrupt 1: SyncManager 1 interrupt pending … 0: No SyncManager 15 interrupt 1: SyncManager 15 interrupt pending	r/-	r/-	0
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)

ESC20	ET1100	ET1200	IP Core

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)

~~ESC20~~	ET1100	ET1200	IP Core

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)

~~ESC20~~	ET1100	~~ET1200~~	IP Core

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)

ESC20	ET1100	ET1200	IP Core

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)

ESC20	ET1100	ET1200	IP Core
			V2.0.3/ V2.03a

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)

ESC20	ET1100	ET1200	IP Core
			V2.0.3/ V2.03a

Bit	Description	ECAT	PDI	Reset Value
	µC 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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
7:0	Lost Link counter of Port y (counting is stopped when 0xff is reached). 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)

ESC20	ET1100	ET1200	IP Core
			write config.

Bit	Description	ECAT	PDI	Reset Value
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

3.39 Watchdog Time PDI (0x0410:0x0411)

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

ESC20	ET1100	ET1200	IP Core

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

ESC20	ET1100	ET1200	IP Core

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

ESC20	ET1100	ET1200	IP Core

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)

~~ESC20~~ **ET1100** **ET1200** IP Core

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)

~~ESC20~~ **ET1100** **ET1200** IP Core

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 ESI EEPROM Interface (0x0500:0x050F)

Table 62: ESI 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 ESI 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)

ESC20 ET1100 ET1200 IP Core

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)

ESC20 ET1100 ET1200 IP Core

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)

		ESC20 [7]	ET1100	ET1200	IP Core
Bit	**Description**	**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 (I²C interface used) 1: PDI emulates EEPROM (I²C 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	
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)

		ESC20	ET1100	ET1200	IP Core
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])

		ESC20	ET1100	ET1200	IP Core
		~~[63:32]~~			~~[63:32]~~
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)

		~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
					~~[63:32]~~
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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 if Transparent Mode is enabled.

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

ESC20	ET1100	ET1200	IP Core
~~[13]~~	~~[13]~~	~~[13]~~	[13] V2.0.0/ V2.00a

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
			V2.0.0/ V2.00a

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)

ESC20	ET1100	ET1200	IP Core
			V2.0.0/ V2.00a

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)

ESC20	ET1100	ET1200	IP Core
			V2.0.0/ V2.00a; [5] V2.0.2/ V2.02a

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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
31:0	Logical start address within the EtherCAT Address Space.	r/w	r/-	0	

Table 79: Register Length FMMU y (0x06y4:0x06y5)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
15:0	Physical Start Address (mapped to logical Start address)	r/w	r/-	0

Table 83: Register Physical Start bit FMMU y (0x06yA)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 ET1100 ET1200 IP Core

Bit	Description	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)

ESC20 ET1100 ET1200 IP Core

Bit	Description	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)

ESC20 ET1100 ET1200 IP Core

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)

ESC20	ET1100	ET1200	IP Core
~~[7:6]~~	~~[7:6]~~	~~[7:6]~~	[7:6] V2.3.0/ V2.03a

Bit	Description	ECAT	PDI	Reset Value
0	Interrupt Write: 1: Interrupt after buffer was completely and successfully written 0: Interrupt cleared after first byte of buffer was read	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/-	0
2	Reserved	r/-	r/-	0
3	Mailbox mode: mailbox status: 0: Mailbox empty 1: Mailbox full Buffered mode: reserved	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/-	11
6	Read buffer in use (opened)	r/-	r/-	0
7	Write buffer in use (opened)	r/-	r/-	0

Table 92: Register Activate SyncManager y (0x0806+y*8)

ESC20	ET1100	ET1200	IP Core
~~[7:6]~~		~~[7:6]~~	

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20	ET1100	ET1200	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 ET1100 ET1200 IP Core

Bit	Description	ECAT	PDI	Reset Value
31:0	Write: 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. Write (ESC20, ET1200 exception): 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. Read: Local time of the beginning of the last receive frame containing a write access to this register. NOTE: The time stamps cannot be read in the same frame in which this register was written.	r/w (special function)	r/-	Undefined

Table 96: Register Receive Time Port 1 (0x0904:0x0907)

ESC20 ET1100 ET1200 IP Core

Bit	Description	ECAT	PDI	Reset Value
31:0	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. ESC20, ET1200 exception: 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.	r/-	r/-	Undefined

Table 97: Register Receive Time Port 2 (0x0908:0x090B)

~~ESC20~~ ET1100 ~~ET1200~~ IP Core

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)

~~ESC20~~	ET1100	ET1200	~~IP Core~~

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[0x910:0x0917] is described in the next chapter.

Table 99: Register Receive Time ECAT Processing Unit (0x0918:0x091F)

ESC20	ET1100	**ET1200**	IP Core
~~[63:32]~~			[63:32] config.

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 BWR or FPWR (configured address) 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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

Bit	Description	ECAT	PDI	Reset Value
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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 ET1100 **ET1200** IP Core

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). Minimum value: 0x0080	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)

ESC20 ET1100 **ET1200** IP Core

Bit	Description	ECAT	PDI	Reset Value
15:0	Representation of the deviation between local clock period and Reference Clock's clock period Range: ±(Speed Counter Start – 0x80)	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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20	ET1100	ET1200	IP Core
~~[7:3]~~	~~[7:3]~~	~~[7:3]~~	[7:3] V2.2.0/ V2.02a

Bit	Description	ECAT	PDI	Reset Value
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: ½ 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)

ESC20	ET1100	ET1200	IP Core

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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	IP Core
			V2.2.0/ V2.02a

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)

ESC20	ET1100	**ET1200**	IP Core

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)

ESC20	ET1100	**ET1200**	IP Core

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	ET1100	**ET1200**	IP Core
~~[63:32]~~			[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.
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	ET1100	ET1200	IP Core
~~[63:32]~~			[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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 ET1100 **ET1200** IP Core

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)

ESC20 [2]	ET1100	ET1200 [2]	IP Core

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 [2]	ET1100	ET1200 [2]	IP Core

Bit	Description	ECAT	PDI	Reset Value
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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

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])

ESC20	ET1100	ET1200	IP Core
~~[63:32]~~			[63:32] config.

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)

~~ESC20~~ ET1100 ~~ET1200~~ IP Core

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)

~~ESC20~~ ET1100 ~~ET1200~~ IP Core

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)

~~ESC20~~ ET1100 ~~ET1200~~ IP Core

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)

~~ESC20~~ ~~ET1100~~ **ET1200** ~~IP Core~~

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)

~~ESC20~~ **ET1100** ~~ET1200~~ ~~IP Core~~

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)

~~ESC20~~ ~~ET1100~~ ~~ET1200~~ **IP Core**

Bit	Description	ECAT	PDI	Reset Value
63:0	Product ID	r/-	r/-	Depends on configuration

Table 132: Register Vendor ID (0x0E08:0x0E0F)

~~ESC20~~ ~~ET1100~~ ~~ET1200~~ **IP Core**

Bit	Description	ECAT	PDI	Reset Value
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)

ESC20 ~~ET1100~~ ~~ET1200~~ ~~IP Core~~

Bit	Description	ECAT	PDI	Reset Value
	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)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
31:0	Output Data	r/w	r/-	0	

NOTE: Register size depends on PDI setting and/or device configuration.

3.51 General Purpose Outputs (0x0F10:0x0F17)

Table 135: Register General Purpose Outputs (0x0F10:0x0F17)

		~~ESC20~~	ET1100	ET1200	IP Core
			~~[63:16]~~ config.	~~[63:12]~~	
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
[7:0] [15:0] [31:0] [63:0]	General Purpose Output Data	r/w	r/w	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)

		~~ESC20~~	ET1100	~~ET1200~~	IP Core
			~~[63:16]~~ config.		
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
[7:0] [15:0] [31:0] [63:0]	General Purpose Input Data	r/-	r/-	0	

NOTE: Register size depends on PDI setting and/or device configuration

3.53 User RAM (0x0F80:0x0FFF)

Table 137: User RAM (0x0F80:0x0FFF)

		ESC20	ET1100	ET1200	IP Core
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
----	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)

~~ESC20~~	~~ET1100~~	~~ET1200~~	**IP Core**
			V1.1.0/ V1.01a

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)	
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	
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	

Bit	Description	Reset Value
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)

	ESC20	ET1100	ET1200	IP Core

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)

	ESC20	ET1100	ET1200	IP Core
	4 KB	8 KB	1 KB	config.

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

ET1100

EtherCAT® Slave Controller

Section III – Hardware Description

Pinout, Interface description, electrical and mechanical specification, ET1100 register overview

Version 1.8
Date: 2010-05-03

BECKHOFF

DOCUMENT HISTORY

Version	Comment
0.6	Editorial Changes
0.7	• Synchronous µController Interface LSB/MSB clarification table added • EEPROM_LOADED pull-down recommendation added • Chip label updated • $VCC_{I/O}$/$GND_{I/O}$ pins adjacent to LDO indicated • Frame processing order example corrected • I^2C EEPROM interface description added • MII management interface description added • Corrected Process RAM size in Register Overview • P_CONF does not correspond with physical ports. See new port configuration tables for details. • Revision/Build information added
0.8	• CLK25OUT1/2 availability completed • Recommendations for unused input pins added (should not be left open) • EEPROM_SIZE description corrected from kByte to kBit, possible EEPROM sizes range from 16kBit to 4Mbit • RoHS compliance added • Autonegotiation is mandatory for ESCs • Description of power supply options added • Electrical characteristics added/revised • SPI_IRQ delay added, support for SPI masters with 2 or 4 bytes added • TX Shift timing diagram and description added • Internal 27KΩ PU/PD resistors at EBUS-RX pins added • LED polarity depending on configuration pin setting described • Recommendation for voltage stabilization capacitors added • Description of Digital I/O behavior on watchdog expiration enhanced • 8 bit asynchronous µController PDI connection added • EBUS ports are open failsafe • Reset example schematic added • Ethernet PHY requirements and PHY connection schematic added • MI_DATA pull-up requirement added • µController PDI: DATA bus signal direction corrected • Pin/Signal description overview added • PERR(x) LEDs are only for testing/debugging • Editorial changes
1.0	• RUN, LINKACT/x) and PERR(x) LED activity level corrected: active high if pulled down, active low if pulled up • DC Characteristics enhanced: added $V_{Reset\ Core}$, V_{ID}, V_{IC} • Synchronous µController interface: timing characteristics enhanced • Note on RBIAS if no EBUS ports/only MII ports are used • DC SYNC/LATCH signal description and timing characteristics added • MII Interface chapter and MII timing characteristics added • EBUS Interface chapter added • Frame processing order, PHY requirements, EEPROM Interface description and MII Management Interface description moved to Section I • TX Shift description moved to MII Interface chapter • Ambient temperature range instead of junction temperature range • Editorial changes

Version	Comment
1.1	• Port configurations with 2 ports: P_CONF[3] erroneously named P_MODE[3] • Clarified I/O voltage with respect to I/O power supply (only 3.3V I/O with $V_{CCI/O}$=3.3V, and no 5V input tolerance unless $V_{CCI/O}$=5V) • Update to ET1100 stepping 1 • Added/revised OSC_IN, CLK25OUT1/2, and MII TX signal timings • Added soldering profile • PHY address configuration changed • Added feature detail overview, removed redundant feature details • PDI and DC SYNC/LATCH signals are not driven until EEPROM is loaded • Synchronous 8/16 bit µController interface: clarified that clock is CPU_CLK_IN • Editorial changes
1.2	• PHY address configuration chapter added, configuration revised • Enhanced link detection for MII available depending on PHY address configuration • Ethernet Management Interface: read and write times were interchanged • Reserved pins are input pins • Editorial changes
1.3	• Added reset timing figure and power-on value sample time • Distributed Clocks SYNC/LATCH signals are configurable and unidirectional • Information on CLK25OUT/CPU_CLK clock output during reset added • Description of internal PU/PD resistors at EBUS_RX pins enhanced • Added t_{Diff} timing characteristic • Power supply example schematic clarified • Enhanced package information: MSL, ball's material, and solder joint recommendation • Digital I/O PDI: added SOF/OUTVALID description, dispensable timings removed • Editorial changes
1.4	• Register 0x0980 is only available if DC Sync Unit is enabled (0x0140.10=1) • Updated solder joint recommendation • OSC_IN/OSC_OUT pin capacitance added, crystal connection note extended • Release Notes added • Timing requirement for asynchronous µController PDI ($t_{ADR\ BHE\ setup}$) relaxed • Input threshold voltage for OSC_IN added • Example schematic for transparent mode added • Renamed Err(x) LED to PERR(x) • Digital I/O PDI: OE_CONF functionality in bidirectional mode corrected • Digital I/O PDI: output event description corrected (EOF mode and WD_TRIG mode) • SPI PDI: access error if SPI_DI not 1 in the last read byte (not SPI_DO) • Async./sync. µC PDI: access error with A(0)=1 and nBHE=1 (not nBHE=0), timing requirements and diagrams clarified • Async. µC PDI: timing requirement for asynchronous µController PDI ($t_{ADR\ BHE\ setup}$) relaxed • AC timing: forwarding delay figures enhanced • Editorial changes
1.5	• Reset timing figure corrected • Maximum soldering profile added • SPI PDI updated • ESI EEPROM interface is a point-to-point connection • Editorial changes
1.6	• Update to ET1100-0002 • Editorial changes
1.7	• µC PDI timing updated • Editorial changes

Version	Comment
1.8	• Enhanced Link Detection must not be activated if EBUS ports are used • Enhanced Link Detection for MII ports requires PHY address offset = 0 • Digital Output principle schematic updated • Chip label updated • Editorial changes

CONTENTS

TABLES

FIGURES

ABBREVIATIONS

(x)	Physical Port x
[y]	Bit y
µC	Microcontroller
ADR	Address
AL	Application Layer
BD	Bidirectional
BGA	Ball Grid Array
BHE	Bus High Enable
CMD	Command
CS	Chip Select
DC	Distributed Clock
Dir.	Pin direction
DL	Data Link Layer
ECAT	EtherCAT
EMC	Electromagnetic Compatibility
EMI	Electromagnetic Interference
EOF	End of Frame
ESC	EtherCAT Slave Controller
ESI	EtherCAT Slave Information
FMMU	Fieldbus Memory Management Unit
GPI	General Purpose Input
GPO	General Purpose Output
I	Input
I/O	Input or Output
IRQ	Interrupt Request
LDO	Low Drop-Out regulator
LI-	LVDS RX-
LI+	LVDS RX+
LO-	LVDS TX-
LO+	LVDS TX+
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
n.a.	not available
n.c.	not connected
O	Output
PD	Pull-down
PDI	Process Data Interface
PLL	Phase Locked Loop
PU	Pull-up
QFN	Quad Flat package No leads
RD	Read
SM	SyncManager
SOF	Start of Frame
SPI	Serial Peripheral Interface
TA	Transfer Acknowledge
TFBGA	Thin-profile Fine-pitch BGA
TS	Transfer Start
UI	Unused Input (PDI: PD, others: GND)
WD	Watchdog
WPD	Weak Pull-down, sufficient only for configuration signals
WPU	Weak Pull-up, sufficient only for configuration signals
WR	Write

1 Overview

The ET1100 ASIC is an EtherCAT Slave Controller (ESC). It takes care of the EtherCAT communication as an interface between the EtherCAT fieldbus and the slave application. The ET1100 supports a wide range of applications. For example, it may be used as a 32 bit Digital I/O node without external logic using Distributed clocks, or as a part of a complex µController design with up to 4 EtherCAT communication ports.

Table 1: ET1100 Main Features

Feature	ET1100
Ports	2-4 ports (each EBUS or MII)
FMMUs	8
SyncManagers	8
RAM	8 KB
Distributed Clocks	Yes, 64 bit (power saving options with ESI EEPROM configuration)
Process Data Interfaces	• 32 Bit Digital I/O (unidirectional/bidirectional) • SPI Slave • 8/16 asynchronous/synchronous µController
Power supply	Integrated voltage regulator (LDO) for logic core/PLL (5V/3.3V to 2.5V), optional external power supply for logic core/PLL.
I/O	3.3V or 5V compatible I/O
Package	BGA128 (10x10 mm^2)
Other features	• Internal 1GHz PLL • Clock output for external devices (10, 20, 25 MHz)

The general functionality of the ET1100 EtherCAT Slave Controller (ESC) is shown in Figure 1:

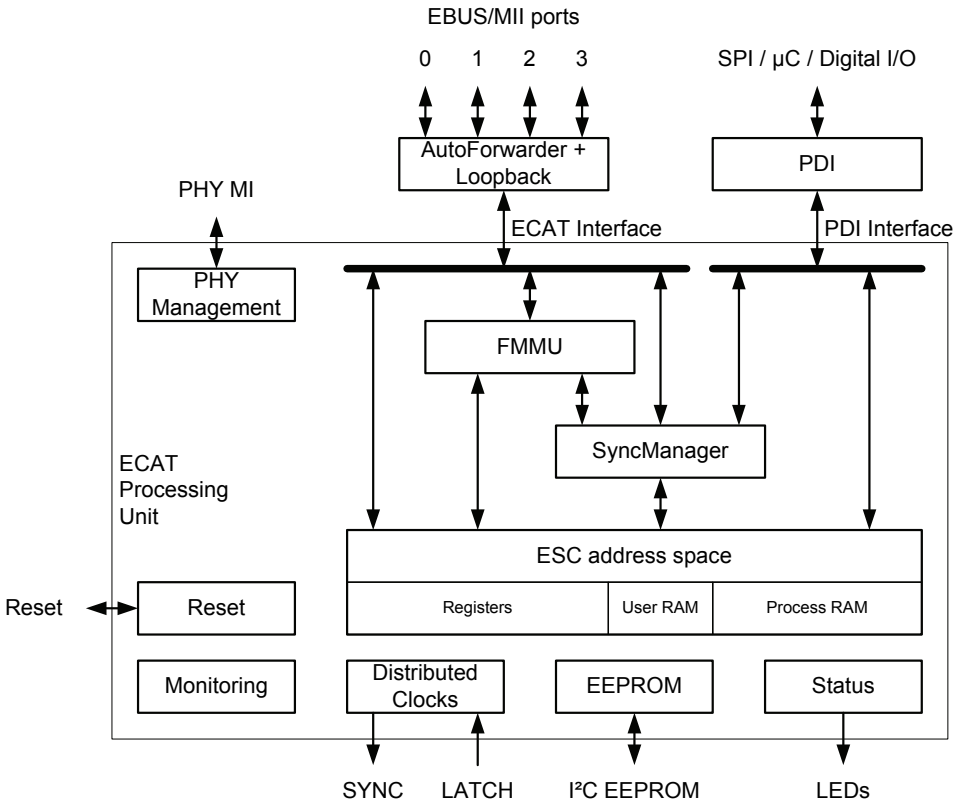


Figure 1: ET1100 Block Diagram

1.1 ET1100 Feature Details

Table 2: ET1100 Feature Details

Feature	ET1100
EtherCAT Ports	**2-4**
Permanent ports	2-4
Optional Bridge port 3 (EBUS or MII)	-
EBUS ports	0-4
MII ports	0-4
RMII ports	-
Ports 0, 1	x
Ports 0, 1, 2	x
Ports 0, 1, 3	x
Ports 0, 1, 2, 3	x
EtherCAT mode	Direct
Slave Category	Full Slave
Position addressing	x
Node addressing	x
Logical addressing	x
Broadcast addressing	x
Physical Layer General Features	
FIFO Size configurable (0x0100[18:16])	x
Auto-Forwarder checks CRC and SOF	x
Forwarded RX Error indication, detection and Counter (0x0308:0x030B)	x
Lost Link Counter (0x0310:0x0313)	x
Prevention of circulating frames	x
Fallback: Port 0 opens if all ports are closed	x
VLAN Tag and IP/UDP support	x
EBUS Features	
Low Jitter	x
Enhanced Link Detection supported	-
Enhanced Link Detection compatible	-
EBUS signal validation	x
LVDS Transceiver internal	x
LVDS sample rate [MHz]	1,000
General Ethernet Features (MII/RMII)	
MII Management Interface (0x0510:0x051F)	x
Supported PHY Address Offsets	0/16
Link Polarity configurable	x
Enhanced Link Detection	x
Link detection using PHY signal (LED)	x
MI link status and configuration	-
MI controllable by PDI (0x0516:0x0517)	-
MI read error (0x0510.13)	-
MI PHY configuration update status (0x0518.5)	-
Transparent Mode	x

Feature	ET1100
MII Features	
CLK25OUT as PHY clock source	x
Bootstrap TX Shift settings	x
Automatic TX Shift setting (with TX_CLK)	-
TX Shift not necessary (PHY TX_CLK as clock source)	-
PDI General Features	
Extended PDI Configuration (0x0152:0x0153)	x
PDI Error Counter (0x030D)	x
CPU_CLK output (10, 20, 25 MHz)	x
SOF, EOF, WD_TRIG and WD_STATE independent of PDI	-
Available PDIs and PDI features depending on port configuration	x
PDI selection at run-time (ESI EEPROM)	x
PDI active immediately (ESI EEPROM settings ignored)	-
Digital I/O PDI	**x**
Digital I/O width [bits]	8/16/24/32
PDI Control register value (0x0140:0x0141)	4
Control/Status signals:	7/0[1]
LATCH_IN	x[1]
SOF	x[1]
OUTVALID	x[1]
WD_TRIG	x[1]
OE_CONF	x[1]
OE_EXT	x[1]
EEPROM_ Loaded	x[1]
WD_STATE	-
EOF	-
Granularity of direction configuration [bits]	2
Bidirectional mode	x
Output high-Z if WD expired	x
Output 0 if WD expired	x
Output with EOF	x
Output with DC SyncSignals	x
Input with SOF	x
Input with DC SyncSignals	x

[1] Availability depending on port configuration

Feature	ET1100
SPI Slave PDI	**x**
Max. SPI clock [MHz]	20
SPI modes configurable (0x0150[1:0])	x
SPI_SEL polarity configurable (0x0150.4)	x
Data out sample mode configurable (0x0150.5)	x
Busy signaling	-
Wait State byte(s)	x
Number of address extension byte(s)	any
2/4 Byte SPI master support	x
Extended error detection (read busy violation)	x
SPI_IRQ delay	x
Status indication	x
EEPROM_ Loaded signal	x
8/16 bit asynchronous µController PDI	**x**
Extended µC configuration bits 0x0150[7:4], 0x0152:0x0153	x
ADR[15:13] available (000_b if not available)	x
EEPROM_LOADED signal	x
RD polarity configurable (0x0150.7)	x
Read BUSY delay (0x0152.0)	x
8/16 bit synchronous µController PDI	**x**
EEPROM_ Loaded signal	x
On-Chip Bus (Avalon/OPB) PDI	**-**
EtherCAT Bridge (port 3, EBUS/MII)	**-**
General Purpose I/O	**x**
GPO bits	0-16
GPI bits	0-16
GPIO available independent of PDI or port configuration	-
Concurrent access to GPO by ECAT and PDI	x
ESC Information	
Basic Information (0x0000:0x0006)	x
Port Descriptor (0x0007)	x
ESC Features supported (0x0008:0x0009)	x
Extended IP Core Configuration in User RAM (0x0F80 ff.)	-
Write Protection (0x0020:0x0031)	x
Data Link Layer Features	
ECAT Reset (0x0040)	x
PDI Reset (0x0041)	-
ESC DL Control (0x0100:0x0103) bytes	4
EtherCAT only mode (0x0100.0)	x
Temporary loop control (0x0100.1)	x
FIFO Size configurable (0x0100[18:16])	x
Configured Station Address (0x0010:0x0011)	x
Configured Station Alias (0x0100.24, 0x0012:0x0013)	x
Physical Read/Write Offset (0x0108:0x0109)	x
Application Layer Features	
Extended AL Control/Status bits (0x0120[15:5], 0x0130[15:5])	x
AL Status Emulation (0x0140.8)	x
AL Status Code (0x0134:0x0135)	x
Interrupts	
ECAT Event Mask (0x0200:0x0201)	x
AL Event Mask (0x0204:0x0207)	x
ECAT Event Request (0x0210:0x0211)	x
AL Event Request (0x0220:0x0223)	x
SyncManager activation changed (0x0220.4)	x
Error Counters	
RX Error Counter (0x0300:0x0307)	x
Forwarded RX Error Counter (0x0308:0x030B)	x
ECAT Processing Unit Error Counter (0x030C)	x
PDI Error Counter (0x030D)	x
Lost Link Counter (0x0310:0x0313)	x
Watchdog	
Watchdog Divider configurable (0x0400:0x0401)	x
Watchdog Process Data	x
Watchdog PDI	x
Watchdog Counter Process Data (0x0442)	x
Watchdog Counter PDI (0x0443)	x
ESI EEPROM Interface (0x0500:0x050F)	
EEPROM sizes supported	1 KB-4 Mbyte
EEPROM size reflected in 0x0502.7	x
EEPROM controllable by PDI	x
EEPROM Emulation by PDI	-
Read data bytes (0x0502.6)	8
Internal Pull-Ups for EEPROM_CLK and EEPROM_DATA	x
FMMUs	8
Bit-oriented operation	x
SyncManagers	8
Watchdog trigger generation for 1 Byte Mailbox configuration independent of reading access	x
SyncManager Event Times (+0x8[7:6])	x

Feature	ET1100
Distributed Clocks	**x**
Width	64
Sync/Latch signals	2
SyncManager Event Times (0x09F0:0x09FF)	x
DC Receive Times	x
DC Time Loop Control controllable by PDI	-
DC activation by EEPROM (0x0140[11:10])	x
Propagation delay measurement with traffic (BWR/FPWR 0x900 detected at each port)	x
LatchSignal state in Latch Status register (0x09AE:0x09AF)	x
SyncSignal Auto-Activation (0x0981.3)	-
SyncSignal 32 or 64 bit Start Time (0x0981.4)	-
SyncSignal Late Activation (0x0981[6:5])	-
SyncSignal debug pulse (0x0981.7)	-
SyncSignal Activation State 0x0984)	-
Reset filters after writing filter depth	-
ESC Specific Registers (0x0E00:0x0EFF)	
Product and Vendor ID	-
POR Values	x
FPGA Update (online)	-
Process RAM and User RAM	
Process RAM (0x1000 ff.) [KByte]	8
User RAM (0x0F80:0x0FFF)	x
Extended IP Core Configuration in User RAM	-

Feature	ET1100
Additional EEPROMs	1
ESI EEPROM (I²C)	x
FPGA configuration EEPROM	-
LED Signals	
RUN LED	x
Link/Activity(x) LED per port	x
PERR(x) LED per port	x
Device ERR LED	-
Clock supply	
Crystal	x
Crystal oscillator	x
TX_CLK from PHY	x
25ppm clock source accuracy	x
Internal PLL	x
Power Supply Voltages	1-2
I/O Voltage	
3.3 V	x
3.3V / 5V tolerant	-
5 V	x
Core Voltage	2.5V
Internal LDOs	1
LDO supply voltage	3.3V/5V
Core Voltage	x
I/O Voltage	-
Package	BGA128
Size [mm²]	10x10
Release date	3/2006
Configuration and Pinout calculator (XLS)	x
Register Configuration	fixed

Table 3: Legend

Symbol	Description
x	available
-	not available
c	configurable

1.2 ET1100 Release Notes

Table 4: ET1100 Release Notes

Stepping	Release notes
ET1100-0001	Restrictions: • Microchip 24xx16 I²C EEPROMs can not be used with the ET1100. Workaround: use Microchip 24xx32 (or even larger devices – configure EEPROM Size appropriately) or I²C EEPROMs from different vendors like ATMEL AT24C or STMicroelectronics M24. • Distributed Clocks SyncSignal generation with single shot mode (SYNC0 Cycle Time 0x09A0:0x09A3 = 0): deactivation fails unless the start time (0x0990:0x0997) is reached and the SYNC0 pulse is generated. A problem occurs if the start time is too far in the future. Workaround: set SYNC0 Cycle Time 0x09A0:0x09A3 to a value different from 0 before deactivation. Restore single shot mode afterwards by setting SYNC 0 Cycle Time back to 0. • DC Cyclic Unit Control register (0x0980) is not available if only DC Latch Unit is enabled (EEPROM value for PDI Control register 0x0140[11:10] = 10). The Latch units can not be assigned to the PDI in this case. Workaround: Enable DC Sync Unit: EEPROM value for PDI Control register 0x0140[11:10] = 11 • Build register 0x0002 has value 0x0000 for ET1100-0001 • Enhanced Link Detection for EBUS cannot be used. ESC feature register 0x0008[5] reflects this incorrectly.
ET1100-0002	New features: • Sample time for Power-On values delayed by ~80 ms to cope with PHYs which power up slow. As a consequence, the startup time is increased by ~80 ms • ESI EEPROM support for devices < 16 kBit • Microchip 24xx16 I²C EEPROMs can be used • Build register 0x0002 has value 0x0002 • SyncManagers can trigger process data watchdog for Digital I/O PDI properly even for 1 byte length in Mailbox mode (Buffered mode remains mandatory for watchdog trigger generation) • SPI timing requirements relaxed • MII Management write enable bit 0x0510[0] is permanently set to 1 if MII Management interface is assigned to PDI (Transparent mode enabled). The behaviour is now similar to the EEPROM Interface. • Distributed Clocks SyncSignal generation can always be deactivated Restrictions: • DC Cyclic Unit Control register (0x0980) is not available if only DC Latch Unit is enabled (EEPROM value for PDI Control register 0x0140[11:10] = 10). The Latch units can not be assigned to the PDI in this case. Workaround: Enable DC Sync Unit: EEPROM value for PDI Control register 0x0140[11:10] = 11 • Enhanced Link Detection for EBUS cannot be used. ESC feature register 0x0008[5] reflects this incorrectly.

1.3 Scope of this document

This documentation refers to stepping ET1100-0002.

2 Pin Description

For pin configuration there is a table calculation file (ET1100 configuration and pinout V<version>.xls) available to make pin configuration easier. This file can be downloaded from the Beckhoff homepage (http://www.beckhoff.com). This documentation supersedes the table calculation file.

Input pins should not be left open/floating. Unused input pins (denoted with direction UI) without external or internal pull-up/pull-down resistor should not be left open. Unused configuration pins should be pulled down if the application allows this (take care of configuration signals in the PDI[39:0] area when bidirectional Digital I/O is used). Unused PDI[39:0] input pins should be pulled down, all other input pins can be connected to GND directly.

Internal pull-up/pull-down resistor values shown in the pinout tables are nominal.

2.1 Overview

2.1.1 Pin Overview

Table 5: Pin Overview

Pin	Pin name	Dir.	Pin	Pin name	Dir.
A1	PDI[27]/RX_DV(3)/EBUS(3)-RX-	BD/LI-	D7	GND_{Core}	
A2	PDI[26]/TX_ENA(3)/EBUS(3)-TX+	BD/LO+	D8	Res. [7]	I
A3	PDI[24]/TX_D(3)[1]/EBUS(3)-TX-	BD/LO-	D9	$GND_{I/O}$	
A4	PDI[22]/TX_D(3)[3]	BD	D10	$V_{CC\ I/O}$	
A5	PDI[20]/RX_D(3)[3]	BD	D11	PDI[1]	BD
A6	PDI[18]/RX_D(3)[0]	BD	D12	PDI[0]	BD
A7	PDI[16]/RX_ERR(3)	BD	E1	TX_D(2)[1]/EBUS(2)-TX-	O/LO-
A8	PDI[14]	BD	E2	PDI[34]/TX_D(2)[0]/ CTRL_STATUS_MOVE	BD
A9	PDI[12]	BD	E3	LINKACT(2)/P_CONF[2]	BD
A10	PDI[10]	BD	E4	Res. [0]	I
A11	PDI[8]	BD	E9	$V_{CC\ I/O}$	
A12	PDI[6]	BD	E10	Res. [3]	I
B1	PDI[29]/RX_D(3)[1]/EBUS(3)-RX+	BD/LI+	E11	SYNC/LATCH[0]	BD
B2	PDI[28]/PERR(3)/TRANS(3)	BD	E12	SYNC/LATCH[1]	BD
B3	PDI[25]/TX_D(3)[0]	BD	F1	TX_ENA(2)/EBUS(2)-TX+	BD/LO+
B4	PDI[23]/TX_D(3)[2]	BD	F2	LINK_MII(2)/CLK25OUT1	BD
B5	PDI[21]/LINK_MII(3)	BD	F3	$V_{CC\ I/O}$ (T0)	
B6	PDI[19]/RX_D(3)[2]	BD	F4	Res. [6]	I
B7	PDI[17]/RX_CLK(3)	BD	F9	$GND_{I/O}$	
B8	PDI[15]	BD	F10	$V_{CC\ I/O}$	
B9	PDI[13]	BD	F11	EEPROM_DATA	BD
B10	PDI[9]	BD	F12	OSC_OUT	O
B11	PDI[7]/CPU_CLK	BD	G1	PDI[35]/RX_ERR(2)	BD
B12	PDI[4]	BD	G2	PDI[36]/RX_CLK(2)	BD
C1	PDI[31]/CLK25OUT2	BD	G3	Res. [1]	I
C2	PDI[30]/LINKACT(3)/P_CONF(3)	BD	G4	Res. [2]	I
C3	PERR(2)/TRANS(2)/PHYAD_OFF	BD	G9	GND_{PLL}	
C4	RBIAS		G10	$V_{CC\ PLL}$	
C5	$V_{CC\ I/O}$		G11	EEPROM_CLK	BD
C6	$V_{CC\ Core}$		G12	OSC_IN	I
C7	$V_{CC\ Core}$		H1	RX_DV(2)/EBUS(2)-RX-	I/LI-
C8	Res. [4]	I	H2	PDI[37]/RX_D(2)[0]	BD
C9	PDI[11]	BD	H3	TESTMODE	I
C10	PDI[5]	BD	H4	$GND_{I/O}$ (T1)	
C11	PDI[3]	BD	H9	$V_{CC\ I/O}$ (T3)	
C12	PDI[2]	BD	H10	Res. [5]	I
D1	PDI[32]/TX_D(2)[3]	BD	H11	RUN/EEPROM_SIZE	BD

Pin	Pin name	Dir.
D2	PDI[33]/TX_D(2)[2]	BD
D3	$V_{CC\ I/O}$	
D4	$GND_{I/O}$	
D5	$GND_{I/O}$	
D6	GND_{Core}	
J5	$GND_{I/O}$	
J6	GND_{Core}	
J7	GND_{Core}	
J8	$GND_{I/O}$	
J9	$GND_{I/O}$	
J10	$V_{CC\ I/O}$	
J11	PERR (0)/TRANS(0)/CLK_MODE[0]	BD
J12	LINKACT(0)/P_CONF[0]	BD
K1	PDI[39]/RX_D(2)[3]	BD
K2	PERR(1)/TRANS(1)/CLK_MODE(1)	BD
K3	LINK_MII(1)	I
K4	RX_CLK(1)	I
K5	$V_{CC\ I/O}$	
K6	$V_{CC\ Core}$	
K7	$V_{CC\ Core}$	
K8	$V_{CC\ I/O}$	
K9	$GND_{I/O}$ (T2)	
K10	RX_D(0)[0]	I
K11	MI_CLK/LINKPOL	BD
K12	MI_DATA	BD
L1	LINKACT(1)/P_CONF(1)	BD
L2	TX_D(1)[2]/P_MODE[0]	BD

Pin	Pin name	Dir.
H12	RESET	BD
J1	RX_D(2)[1]/EBUS(2)-RX+	I/LI+
J2	PDI[38]/RX_D(2)[2]	BD
J3	$V_{CC\ I/O}$	
J4	$GND_{I/O}$	
L3	TX_D(1)[0]/TRANS_MODE_ENA	BD
L4	RX_D(1)[0]	I
L5	RX_D(1)[2]	I
L6	RX_ERR(1)	I
L7	TX_D(0)[2]/C25_SHI[0]	BD
L8	TX_D(0)[0]/C25_ENA	BD
L9	LINK_MII(0)	I
L10	RX_CLK(0)	I
L11	RX_D(0)[2]	I
L12	RX_D(0)[3]	I
M1	TX_D(1)[3]/P_MODE[1]	BD
M2	TX_D(1)[1]/EBUS(1)-TX-	O/LO-
M3	TX_ENA(1)/EBUS(1)-TX+	BD/LO+
M4	RX_DV(1)/EBUS(1)-RX-	I/LI-
M5	RX_D(1)[1]/EBUS(1)-RX+	I/LI+
M6	RX_D(1)[3]	I
M7	TX_D(0)[3]/C25_SHI[1]	BD
M8	TX_D(0)[1]/EBUS(0)-TX-	O/LO-
M9	TX_ENA(0)/EBUS(0)-TX+	BD/LO+
M10	RX_ERR(0)	I
M11	RX_DV(0)/EBUS(0)-RX-	I/LI-
M12	RX_D(0)[1]/EBUS(0)-RX+	I/LI+

2.1.2 Signal Overview

Table 6: Signal Overview

Signal	Type	Dir.	Description
C25_ENA	Configuration	I	CLK25OUT2 Enable: Enable CLK25OUT2
C25_SHI[1:0]	Configuration	I	TX Shift: Shifting/phase compensation of MII TX signals
CLK_MODE[1:0]	Configuration	I	CPU_CLK configuration
CLK25OUT1/CLK25OUT2	MII	O	25 MHz clock source for Ethernet PHYs
CPU_CLK	PDI	O	Clock signal for µController
CTRL_STATUS_MOVE	Configuration	I	Move Digital I/O Control/Status signal to last available PDI byte
EBUS(3:0)-RX-	EBUS	LI-	EBUS LVDS receive signal -
EBUS(3:0)-RX+	EBUS	LI+	EBUS LVDS receive signal +
EBUS(3:0)-TX-	EBUS	LO-	EBUS LVDS transmit signal -
EBUS(3:0)-TX+	EBUS	LO+	EBUS LVDS transmit signal +
EEPROM_CLK	EEPROM	BD	EEPROM I^2C Clock
EEPROM_DATA	EEPROM	BD	EEPROM I^2C Data
EEPROM_SIZE	Configuration	I	EEPROM size configuration
PERR(3:0)	LED	O	Port receive error LED output (for testing)
GND_{Core}	Power		Core logic ground
$GND_{I/O}$	Power		I/O ground
GND_{PLL}	Power		PLL ground
LINK_MII(3:0)	MII	I	PHY signal indicating a link
LINKACT(3:0)	LED	O	Link/Activity LED output
LINKPOL	Configuration	I	LINK_MII(3:0) polarity configuration
MI_CLK	MII	O	PHY Management Interface clock
MI_DATA	MII	BD	PHY Management Interface data
OSC_IN	Clock	I	Clock source (crystal/oscillator)
OSC_OUT	Clock	O	Clock source (crystal)
P_CONF(3:0)	Configuration	I	Physical layer of logical ports
P_MODE[1:0]	Configuration	I	Number of physical ports and corresponding logical ports
PDI[39:0]	PDI	BD	PDI signal, depending on EEPROM content
PHYAD_OFF	Configuration	I	Ethernet PHY Address Offset
RBIAS	EBUS		BIAS resistor for LVDS TX current adjustment
Res. [7:0]	Reserved	I	Reserved pins
RESET	General	BD	Open collector Reset output/Reset input
RUN	LED	O	Run LED controlled by AL Status register
RX_CLK(3:0)	MII	I	MII receive clock
RX_D(3:0)[3:0]	MII	I	MII receive data
RX_DV(3:0)	MII	I	MII receive data valid
RX_ERR(3:0)	MII	I	MII receive error
SYNC/LATCH[1:0]	DC	I/O	Distributed Clocks SyncSignal output or LatchSignal input
TESTMODE	General	I	Reserved for testing, connect to GND
TRANS(3:0)	MII	I	MII interface sharing: share port enable
TRANS_MODE_ENA	Configuration	I	Enable MII interface sharing (and TRANS(3:0) signals)
TX_D(3:0)[3:0]	MII	O	MII transmit data
TX_ENA(3:0)	MII	O	MII transmit enable
$V_{CC\ Core}$	Power		Core logic power
$V_{CC\ I/O}$	Power		I/O power
$V_{CC\ PLL}$	Power		PLL power

2.1.3 PDI Signal Overview

Table 7: PDI signal overview

PDI	Signal	Dir.	Description
Digital I/O	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	I/O[31:0]	I/O/BD	Input/Output or Bidirectional data
	LATCH_IN	I	External data latch signal
	OE_CONF	I	Output Enable Configuration
	OE_EXT	I	Output Enable
	OUTVALID	O	Output data is valid/Output event
	SOF	O	Start of Frame
	WD_TRIG	O	Watchdog Trigger
SPI	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	SPI_CLK	I	SPI clock
	SPI_DI	I	SPI data MOSI
	SPI_DO	O	SPI data MISO
	SPI_IRQ	O	SPI interrupt
	SPI_SEL	I	SPI chip select
µC async.	CS	I	Chip select
	BHE	I	Byte High Enable (16 bit µController interface only)
	RD	I	Read command
	WR	I	Write command
	BUSY	O	EtherCAT device is busy
	IRQ	O	Interrupt
	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	DATA[7:0]	BD	Data bus for 8 bit µController interface
	ADR[15:0]	I	Address bus
	DATA[15:0]	BD	Data bus for 16 bit µController interface
µC sync.	ADR[15:0]	I	Address bus
	BHE	I	Byte High Enable
	CPU_CLK_IN	I	µController interface clock
	CS	I	Chip select
	DATA[15:0]	BD	Data bus for 16 Bit µController interface
	DATA[7:0]	BD	Data bus for 8 Bit µController interface
	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	IRQ	O	Interrupt
	RD/nWR	I	Read/Write access
	TA	O	Transfer Acknowledge
	TS	I	Transfer Start

2.2 Configuration Pins

The configuration pins are used to configure the ET1100 at power-on with pull-up or pull-down resistors. At power-on the ET1100 uses these pins as inputs to latch the configuration[2]. After power-on, the pins have their operation functionality which has been assigned to them, and therefore pin direction changes if necessary. The power-on phase finishes before the nRESET pin is released. In subsequent reset phases without power-on condition, the configuration pins still have their operation functionality, i.e., the ET1100 configuration is not latched again and output drivers remain active.

The configuration value 0 is realized by a pull-down resistor, a pull-up resistor is used for a 1. Since some configuration pins are also used as LED outputs, the polarity of the LED output depends on the configuration value.

2.2.1 Port Mode

Port Mode configures the number of physical ports and the corresponding logical ports. It is shown in Table 8.

Table 8: Port Mode

Description	Config signal	Pin name	Register	P_MODE[1:0] Values
Port Mode	P_MODE[0]	TX_D(1)[2]/P_MODE[0]	0x0E00[0]	00 = 2 ports (log. ports 0 and 1) 01 = 3 ports (log. ports 0,1, and 2) 10 = 3 ports (log. ports 0,1, and 3) 11 = 4 ports (log. ports 0, 1, 2, and 3)
	P_MODE[1]	TX_D(1)[3]/P_MODE[1]	0x0E00[1]	

NOTE: The term physical port in this document is only used for grouping ET1100 interface pins. The register set as well as any master/slave software is always based on logical ports. The distinction between physical and logical ports is made in order to increase the number of available PDI pins. Each logical port is associated with exactly one physical port, and it can be configured to be either EBUS or MII.

MII ports are always assigned to the lower physical ports, then EBUS ports are assigned. If any MII ports are configured, the lowest logical MII port is always connected to physical port 0, the next higher logical MII port is connected to physical port 1, and so on. Afterwards, the lowest logical EBUS port – if configured – is connected to the next physical port following the physical MII ports, i.e. port [number of MII ports]. Without MII ports, the EBUS ports are connected beginning with physical port 0.

If only EBUS or only MII ports are used, the physical port number is the same as the logical port number for P_MODE[1:0]=00, 01 or 11. Refer to the next chapter for more details.

2.2.2 Port Configuration

P_CONF[3:0] determines the physical layer configuration (MII or EBUS). P_CONF[0] determines the physical layer of logical port 0, P_CONF[1] determines logical port 1, P_CONF[2] determines the physical layer of the next available logical port (either 3 for P_MODE[1:0]=10, else 2), and P_CONF[3] determines logical port 3. If a physical port is not used, the corresponding P_CONF configuration signal is not used.

Table 9: Port Configuration

Description	Configuration signal	Pin name	Register	Values
Port Configuration	P_CONF[0]	LINKACT(0)/P_CONF[0]	0x0E00[2]	0 = EBUS 1 = MII
	P_CONF[1]	LINKACT(1)/P_CONF(1)	0x0E00[3]	
	P_CONF[2]	LINKACT(2)/P_CONF[2]	0x0E00[4]	
	P_CONF[3]	PDI[30]/LINKACT(3)/P_CONF(3)	0x0E00[5]	

[2] Take care of proper configuration: External devices attached to dual-purpose configuration pins might interfere sampling the intended configuration if they are e.g. not properly powered at the sample time (external device keeps configuration pin low although a pull-up resistor is attached). In such cases the ET1100 power-on value sampling time can be delayed by delaying power activation.

2.2.2.1 Configurations with 2 ports

For configurations with 2 ports, logical ports 0 and 1 are used. The port signals are available at physical ports 0 and 1, depending on the port configuration. P_MODE[1:0] has to be set to 00. P_CONF[1:0] determine the physical layer of logical ports (1:0). P_CONF[3:2] are not used, nevertheless, P_ CONF[2] should not be left open (connection to GND recommended). P_CONF[3] should be pulled down if possible (denoted with '-' in the table), if your application allows this.

Table 10: Configurations with 2 ports (P_MODE[1:0]=00)

Logical port		Physical port		P_CONF [3:0]
1	0	1	0	
$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	-000
$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(1)}$	$MII_{(0)}$	-001
$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(0)}$	$MII_{(1)}$	-010
$MII_{(1)}$	$MII_{(0)}$	$MII_{(1)}$	$MII_{(0)}$	-011

2.2.2.2 Configurations with 3 ports

For configurations with 3 ports, either logical ports 0, 1, and 2 (P_MODE[1:0]=01) or logical ports 0, 1, and 3 (P_MODE[1:0]=10) are used. The port signals are available at physical ports 0, 1 and 2, depending on the port configuration. P_CONF[2:0] determine the physical layer of logical ports 2, 1, 0, or logical ports 3, 1, 0, depending on the P_MODE settings (P_CONF[2] is either used for logical port 2 or logical port 3). P_ CONF[3] should be pulled down if possible (denoted with '-' in the tables), if your application allows this.

Table 11: Configurations with 3 ports (ports 0,1, and 2; P_MODE[1:0]=01)

Logical port			Physical port			P_CONF [3:0]
2	1	0	2	1	0	
$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	-000
$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	-001
$EBUS_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(2)}$	$EBUS_{(0)}$	$MII_{(1)}$	-010
$EBUS_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$EBUS_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	-011
$MII_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$MII_{(2)}$	-100
$MII_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(1)}$	$MII_{(2)}$	$MII_{(0)}$	-101
$MII_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(0)}$	$MII_{(2)}$	$MII_{(1)}$	-110
$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	-111

Table 12: Configurations with 3 ports (ports 0,1, and 3; P_MODE[1:0]=10)

Logical port			Physical port			P_CONF [3:0]
3	1	0	2	1	0	
$EBUS_{(3)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	-000
$EBUS_{(3)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$EBUS_{(1)}$	$MII_{(0)}$	-001
$EBUS_{(3)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(0)}$	$MII_{(1)}$	-010
$EBUS_{(3)}$	$MII_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$MII_{(1)}$	$MII_{(0)}$	-011
$MII_{(3)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$MII_{(3)}$	-100
$MII_{(3)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(1)}$	$MII_{(3)}$	$MII_{(0)}$	-101
$MII_{(3)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(0)}$	$MII_{(3)}$	$MII_{(1)}$	-110
$MII_{(3)}$	$MII_{(1)}$	$MII_{(0)}$	$MII_{(3)}$	$MII_{(1)}$	$MII_{(0)}$	-111

2.2.2.3 Configurations with 4 ports

For configurations with 4 ports, logical ports 0 to 3 are used. The port signals are available at physical ports 0 to 3, depending on the port configuration. P_MODE[1:0] has to be set to 11. P_CONF[3:0] determine the physical layer of logical ports (3:0).

Table 13: Configurations with 4 ports (P_MODE[1:0]=01)

Logical port				Physical port				P_CONF [3:0]
3	2	1	0	3	2	1	0	
$EBUS_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	0000
$EBUS_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	0001
$EBUS_{(3)}$	$EBUS_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(2)}$	$EBUS_{(0)}$	$MII_{(1)}$	0010
$EBUS_{(3)}$	$EBUS_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$EBUS_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	0011
$EBUS_{(3)}$	$MII_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$MII_{(2)}$	0100
$EBUS_{(3)}$	$MII_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$EBUS_{(1)}$	$MII_{(2)}$	$MII_{(0)}$	0101
$EBUS_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(3)}$	$EBUS_{(0)}$	$MII_{(2)}$	$MII_{(1)}$	0110
$EBUS_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$EBUS_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	0111
$MII_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$MII_{(3)}$	1000
$MII_{(3)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(2)}$	$EBUS_{(1)}$	$MII_{(3)}$	$MII_{(0)}$	1001
$MII_{(3)}$	$EBUS_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(2)}$	$EBUS_{(0)}$	$MII_{(3)}$	$MII_{(1)}$	1010
$MII_{(3)}$	$EBUS_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$EBUS_{(2)}$	$MII_{(3)}$	$MII_{(1)}$	$MII_{(0)}$	1011
$MII_{(3)}$	$MII_{(2)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$EBUS_{(1)}$	$EBUS_{(0)}$	$MII_{(3)}$	$MII_{(2)}$	1100
$MII_{(3)}$	$MII_{(2)}$	$EBUS_{(1)}$	$MII_{(0)}$	$EBUS_{(1)}$	$MII_{(3)}$	$MII_{(2)}$	$MII_{(0)}$	1101
$MII_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$EBUS_{(0)}$	$EBUS_{(0)}$	$MII_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	1110
$MII_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	$MII_{(3)}$	$MII_{(2)}$	$MII_{(1)}$	$MII_{(0)}$	1111

2.2.3 CPU_CLK MODE

CLK_MODE is used to provide a clock signal to an external microcontroller. If CLK_MODE is not 00, CPU_CLK is available on PDI[7], thus this pin is not available for PDI signals anymore. For µController PDIs, PDI[7] is ADR[15], which is treated to be 0 if CPU_CLK is selected. The CPU_CLK MODE is shown in Table 14.

Table 14: CPU_CLK Mode

Description	Config signal	Pin name	Register	Values
CPU_CLK_MODE	CLK_MODE[0]	PERR(0)/ TRANS(0)/ CLK_MODE[0]	0x0E00[6]	00 = off, PDI[7]/CPU_CLK available for PDI 01 = 25 MHz clock output at PDI[7]/CPU_CLK 10 = 20 MHz clock output at PDI[7]/CPU_CLK 11 = 10 MHz clock output at PDI[7]/CPU_CLK
	CLK_MODE[1]	PERR(1)/ TRANS(1)/ CLK_MODE(1)	0x0E00[7]	

2.2.4 TX Shift

Phase shift (0/10/20/30ns) of MII TX signals (TX_ENA, TX_D[3:0]) can be attained via the C25_SHI[x] signals. TX-Shift is explained in Table 15. It is recommended to support all C25_SHI[1:0] configurations by hardware options to enable later adjustments.

Table 15: TX Shift

Description	Config signal	Pin name	Register	Values
TX Shift	C25_SHI[0]	TX_D(0)[2]/C25_SHI[0]	0x0E01[0]	00 = MII TX signals not delayed 01 = MII TX signals delayed by 10 ns 10 = MII TX signals delayed by 20 ns 11 = MII TX signals delayed by 30 ns
	C25_SHI[1]	TX_D(0)[3]/C25_SHI[1]	0x0E01[1]	

2.2.5 CLK25OUT2 Enable

A 25MHz clock for Ethernet PHYs can be made available by the ET1100 on PDI[31]/CLK25OUT2 pin. This is only relevant if three MII ports are used. In cases with less than 3 MII ports, pin LINK_MII(2)/CLK25OUT1 provides CLK25OUT anyway, because LINK_MII(2) is not used. If 4 MII ports are used, PDI[31]/CLK25OUT2 provides CLK25OUT2 regardless of CLK25OUT2 Enable. CLK25OUT2 Enable is explained in Table 16.

Table 16: CLK25OUT2 Enable

Description	Config signal	Pin name	Register	Values
CLK25OUT2 Enable	C25_ENA	TX_D(0)[0]/C25_ENA	0x0E01[2]	0 = disable, PDI[31]/CLK25OUT2 is available for PDI 1 = enable, PDI[31]/CLK25OUT2 is 25 MHz clock output

2.2.6 Transparent Mode Enable

The ET1100 is capable of sharing the MII interfaces with other MACs on a per port basis. Typically, the Transparent mode is disabled, and the ET1100 has exclusive access to the MII interfaces of the PHYs. With the Transparent mode turned on, the MII interfaces can be assigned either to the ET1100 or to other MACs, e.g., µControllers with integrated MACs. Reassignment is not meant to be done whilst network traffic is processed.

The Transparent mode primarily affects the PERR(x)/TRANS(x) signals. If Transparent mode is enabled, PERR(x)/TRANS(x) becomes TRANS(x) (active low), which controls the transparent state of each port. PERR(x) is not available in Transparent mode.

TRANS(x) does only affect the TX_ENA(x)/TX_D(x) signals of the same port as well as MI_CLK/MI_DATA. RX_CLK(x), RX_DV(x), RX_D(x), and RX_ERR(x) are connected to both ET1100 and the other MAC.

Each MII interface behaves as usual as long as TRANS(x) is high, and the ET1100 controls the MII interface. If TRANS(x) is low, the port becomes transparent (or isolated), i.e., the ET1100 will no longer drive TX_ENA(x)/TX_D(x) actively, thus, the other MAC can drive these signals.

The Link/Act(x) LED will still be driven by the ET1100, because it samples RX_DV(x) and TX_ENA(x) (which becomes an input while a port is transparent) for detection of activity.

As long as at least one MII interface is not transparent, the ET1100 is in control of the MII management interface. With the Transparent mode turned on, the PHY management interface of the ET1100 can be accessed via the PDI interface, so a µController gets access to the management interface. If all MII interfaces are transparent, the ET1100 releases MI_CLK and MI_DATA, so they can be used by the other MAC, too.

Refer to example schematics for more details.

Table 17: Transparent Mode Enable

Description	Config signal	Pin name	Register	Values
Transparent Mode Enable	TRANS_MODE_ENA	TX_D(1)[0]/ TRANS_MODE_ENA	0x0E01[3]	0 = normal mode/Transparent mode disabled. ET1100 uses PHY exclusively 1 = Transparent mode enabled, ET1100 can share PHY with other MACs

2.2.7 Digital Control/Status Move

If more than 2 MII ports are used (PDI[39:32] are not available for PDI use), the Digital I/O PDI control and status signals can be made available at the highest available PDI byte with CTRL_STATUS_MOVE.

Digital Control/Status Move is explained in Table 18:

Table 18: Digital Control/Status Move

Description	Config signal	Pin name	Register	Values
Digital Control/ Status Move	CTRL_STATUS_MOVE	PDI[34]/TX_D(2)[0]/ CTRL_STATUS_MOVE	0x0E01[4]	0 = Digital I/O control/status signals are not moved: they are available at PDI[39:32] if less than 3 MII ports are used, otherwise they are not available 1 = Digital I/O control/status signals moved to last PDI byte if PDI[39:32] is used for MII(2). Digital I/O control/status signals are available in any configuration.

2.2.8 PHY Address Offset

The ET1100 supports two PHY address offset configurations, either 0 or 16. Refer to chapter 3.2 for details on PHY address configuration. PHY address offset must be 0 if Enhanced Link Detection is to be used.

PHY Address Offset is explained in Table 19:

Table 19: PHY Address Offset

Description	Config signal	Pin name	Register	Values
PHY Address Offset	PHYAD_OFF	PERR(2)/TRANS(2)/ PHYAD_OFF	0x0E01[5]	0 = PHY address offset 0 1 = PHY address offset 16

2.2.9 Link Polarity

Ethernet PHYs signal a 100 Mbit/s Full (Duplex Link(to the ET1100 by asserting LINK_MII(x). The polarity can be selected with LINKPOL.

Link Polarity is explained in Table 20:

Table 20: Link Polarity

Description	Config signal	Pin name	Register	Values
Link Polarity	LINKPOL	MI_CLK/LINKPOL	0x0E01[6]	0 = LINK_MII(x) is active low 1 = LINK_MII(x) is active high
Reserved	RESERVED	PDI[28]/PERR(3)/TRANS(3)	0x0E01[7]	reserved

2.2.10 ESI EEPROM Size

EEPROM_SSIZE determines the size of the EEPROM (and the number of I²C address bytes). EEPROM_SIZE is sampled at the beginning of the EEPROM access. EEPROM_SIZE is shown in Table 21:

Table 21: ESI EEPROM_SIZE

Description	Config signal	Pin name	Register	Values
E²PROM Size	EEPROM_SIZE	RUN/EEPROM_SIZE	0x0502[7]	0 = 1 address byte (1 kbit to 16 kbit EEPROM) 1 = 2 address bytes (32 kbit to 4 Mbit EEPROM)

2.2.11 Reserved

The reserved configuration pin should be pulled down when 4 ports are used. Otherwise it should be left open. It is shown in Table 22:

Table 22: Reserved

Description	Config signal	Pin name	Register	Values
Reserved	RESERVED	PDI[28]/PERR(3)/TRANS(3)	0x0E01[7]	0 for 4 port configurations

2.3 General ET1100 Pins

Table 23: General pins

Pin	Pin		Signal		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
G12	OSC_IN	I	OSC_IN	I		
F12	OSC_OUT	O	OSC_OUT	O		
H12	RESET	BD	RESET	BD		3.3 KΩ PU
C4	RBIAS		RBIAS			
H3	TESTMODE	I	TESTMODE	I		WPD

OSC_IN
Connection to external crystal or oscillator input (25 MHz). An oscillator as the clock source for both ET1100 and PHYs is mandatory if MII ports are used and CLK25OUT1/2 can not be used as the clock source for the PHYs. The 25 MHz clock source should have an initial accuracy of 25ppm or better.

OSC_OUT
Connection to external crystal. Should be left open if an oscillator is connected to OSC_IN.

RESET
The open collector RESET input/output (active low) signals the reset state of ET1100. The reset state is entered at power-on, if the power supply is to low, or if a reset was initiated using the reset register 0x0040. ET1100 also enters reset state if RESET pin is held low by external devices

RBIAS
Bias resistor for LVDS TX current adjustment, should be 11KΩ connected to GND.

TESTMODE
Reserved for testing, should be connected to GND.

2.4 ESI EEPROM Interface Pins

Table 24: ESI EEPROM pins

Pin	Pin		Signal		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
G11	EEPROM_CLK	BD	EEPROM_CLK	BD		3.3 KΩ PU
F11	EEPROM_DATA	BD	EEPROM_DATA	BD		3.3 KΩ PU

EEPROM_CLK
EEPROM I²C clock signal (open collector output).

EEPROM_DATA
EEPROM I²C data signal (open collector output).

2.5 MII Management Pins

The MII Management signals are only used if at least one MII port is configured.

Table 25: MII Management pins

Pin	Pin		No MII port used		MII port(s) used		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.	Signal	Dir.		
K11	MI_CLK/LINKPOL	BD		UI	MI_CLK	O	LINKPOL	WPD
K12	MI_DATA	BD		UI	MI_DATA	BD		WPU

MI_CLK/LINKPOL

During power on LINK Polarity configuration during power-up, PHY Management Interface clock afterwards.

MI_DATA

PHY Management Interface Data.

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

2.6 Distributed Clocks SYNC/LATCH Pins

Table 26: DC SYNC/LATCH pins

Pin	Pin		Signal		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
E11	SYNC/LATCH[0]	BD	SYNC[0]/ LATCH[0]	O/ I		
E12	SYNC/LATCH[1]	BD	SYNC[1]/ LATCH[1]	O/ I		

SYNC/LATCH[x]

Distributed Clocks SyncSignal output or LatchSignal input, depending on ESI EEPROM configuration. SYNC/LATCH signals are not driven (high impedance) until the EEPROM is loaded.

2.7 LED Signals

All LED signals are also used as configuration signals. The polarity of each LED signal depends on the configuration: LED is active high if pin is pulled down for configuration, and active low if pin is pulled up. Refer to the example schematics for LED connection details.

Table 27: LED pins

Pin	Pin		Signal		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
H11	RUN/EEPROM_SIZE	BD	RUN	O	EEPROM_SIZE	

NOTE: The pin locations for LINKACT(x) and PERR(x)/TRANS(x) are described in the Physical Port 0-3 chapters.

RUN/EEPROM_SIZE
ESI EEPROM Size configuration (either 1KBit-16KBit or 32KBit-4Mbit) sampled at the beginning of the EEPROM access. Otherwise RUN LED signal. RUN is active high if pin is pulled down, and active low if pin is pulled up. Refer to example schematics for connection details. RUN LED should be green.

LINKACT(x)
Link/Activity LED output (off=no link, on=link without activity, blinking=link and activity) for physical port x. LINKACT(x) is active high if pin is pulled down, and active low if pin is pulled up. Refer to example schematics for connection details. Link/Activity LED should be green.

PERR(x)/TRANS(x)
Error LED output of physical port x for EBUS ports, and for MII ports if TRANS_MODE_ENA=0. If TRANS_MODE_ENA=1, PERR(x)/TRANS(x) is used as TRANS(x) for MII physical port x, which puts port x into isolate/transparent operation. PERR(x) is not available in this case. PERR(x) is active high if pin is pulled down, and active low if pin is pulled up. Refer to example schematics for connection details.

NOTE: PERR(x) LEDs are not part of the EtherCAT indicator specification. They are only intended for testing and debugging. The PERR(x) LED flashes once if a physical layer receive error occurs. Do not confuse PERR(x) LEDs with application layer ERR LED, this is not supported by the ESCs and has to be controlled by a µController.

2.8 Physical Ports and PDI Pins

The ET1100 pin out is optimized in order to achieve an optimum of size and features. To obtain this, there is a number of pins where either communication or PDI functionality can be assigned to. Number and type of the communication ports might reduce/exclude one or more PDI possibilities.

The physical communication ports are numbered from port 0 to port 3. Port 0 and port 1 do not interfere with PDI pins, while port 2 and port 3 might overlap with PDI[39:16] and therefore limit the number of choices for the PDI.

Pin configuration for ports will overwrite pin configuration for PDI. Therefore, number and type of ports should be configured first.

The ET1100 has 40 PDI pins, PDI[39:0]. They are structured in 4 groups: PDI[15:0] (PDI byte 0/1), PDI[16:23] (PDI byte 2), PDI[24:31] (PDI byte 3), and PDI[32:39] (PDI byte 4).

Possible Physical Port / PDI combinations

Table 28: Combinations of physical ports and PDI

	Async. µC	Sync. µC	SPI	Digital I/O with CTLR_STATUS_MOVE=	
				0	1
2 ports or 3 ports with min. 1xEBUS	8 Bit 16Bit	8 Bit 16Bit	SPI +32 Bit GPI/O	32Bit I/O +control/status signals	
3xMII, 0xEBUS	8Bit	8Bit	SPI +24 Bit GPI/O	32Bit I/O	24Bit I/O + control/status signals
4 ports, min. 2xEBUS	-	-	SPI +16Bit GPI/O	24Bit I/O + control/status signals	
3xMII, 1xEBUS	-	-	SPI +16Bit GPI/O	24 Bit I/O	16Bit I/O + control/status signals
xMII	-	-	SPI +8Bit GPI/O	16Bit I/O	8Bit I/O + control/status signals

2.8.1 Physical Port Signals

2.8.2 MII Interface

LINK_MII(x)
Input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established. LINK_MII(x) polarity is configurable.

RX_CLK(x)
MII Receive Clock

RX_DV(x)
MII receive data valid.

RX_D(x)[3:0]
MII receive data.

RX_ERR(x)
MII receive error.

TX_ENA(x)
MII transmit enable output. Used as MII transmit enable input for controlling the Link/Activity LED if port is in transparent mode (TRANS_MODE_ENA=1 and TRANS(x)=0).

TX_D(x)[3:0]
MII transmit data.

2.8.2.1 CLK25OUT1/2 Signals

The ET1100 has to provide the Ethernet PHYs with a 25 MHz clock signal (CLK25OUT) if a 25 MHz crystal is used for clock generation. In case a 25 MHz oscillator is used, CLK25OUT is not necessary, because Ethernet PHYs and ET1100 can share the oscillator output. Depending on the port configuration and C25_ENA, CLK25OUT is available at different pins:

Table 29: CLK25OUT1/2 signal output

Conf.	C25_ENA=0	C25_ENA=1
0-2xMII	LINK_MII(2)/CLK25OUT1 provides CLK25OUT (PDI[31]/CLK25OUT2 also provides CLK25OUT if 4 ports are used)	LINK_MII(2)/CLK25OUT1 and PDI[31]/CLK25OUT2 provide CLK25OUT
3xMII	CLK25OUT not available, oscillator is mandatory	PDI[31]/CLK25OUT2 provides CLK25OUT
4xMII	PDI[31]/CLK25OUT2 provides CLK25OUT	

NOTE: Unused CLK25OUT pins should not be connected to reduce driver load.

The CLK25OUT pins provide a clock signal – if configured – during external or ECAT reset, clock output is only turned of during power-on reset.

2.8.3 EBUS Interface

The EBUS ports of the ET1100 are open failsafe, i.e., the ET1100 detects if an EBUS port is unconnected and closes the port internally (no physical link).

EBUS(x)-RX+/EBUS(x)-RX-
EBUS LVDS receive signals. EBUS_RX+ pins incorporate a pull-down resistor R_{LI+} and EBUS_RX- pins incorporate a pull-up resistor R_{LI-}, even if the pins are not configured for EBUS.

EBUS(x)-TX+/EBUS(x)-TX-
EBUS LVDS transmit signals.

2.8.4 PDI Pins

PDI[x]
The function of PDI[x] signals depends on the configuration stored in the device ESI EEPROM. PDI signals are not driven (high impedance) until the EEPROM is loaded. This has to be taken into account especially for Digital Outputs.

CPU_CLK
The ET1100 can provide a clock signal for µControllers on pin PDI[7]/CPU_CLK. The CPU_CLK output setting is controlled by the CLK_MODE configuration pin. If CPU_CLK is enabled, PDI[7] is not available for the PDI, i.e., ADR[15] cannot be used by µController PDIs (ADR[15] is treated to be 0 internally), and I/O[7] is not available for Digital I/O PDIs.

CPU_CLK provides a clock signal – if configured – during external or ECAT reset, clock output is only turned of during power-on reset.

2.8.5 Physical Port 0

Table 30 shows the pins for physical port 0. It can be configured as MII or EBUS and is always available. Use of this port does in no case clash with pins needed for PDI.

Table 30: Physical Port 0

Pin	Pin		MII		EBUS		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.	Signal	Dir.		
M9	TX_ENA(0)/ EBUS(0)-TX+	BD/LO+	TX_ENA(0)	O/I	EBUS(0)-TX+	LO+		
L8	TX_D(0)[0]/ C25_ENA	BD	TX_D(0)[0]	O			C25_ENA	
M8	TX_D(0)[1]/ EBUS(0)-TX-	O/LO-	TX_D(0)[1]	O	EBUS(0)-TX-	LO-		
L7	TX_D(0)[2]/ C25_SHI[0]	BD	TX_D(0)[2]	O			C25_SHI[0]	
M7	TX_D(0)[3]/ C25_SHI[1]	BD	TX_D(0)[3]	O			C25_SHI[1]	
K10	RX_D(0)[0]	I	RX_D(0)[0]	I		UI		
M12	RX_D(0)[1]/ EBUS(0)-RX+	I/LI+	RX_D(0)[1]	I	EBUS(0)-RX+	LI+		27 KΩ PD
L11	RX_D(0)[2]	I	RX_D(0)[2]	I		UI		
L12	RX_D(0)[3]	I	RX_D(0)[3]	I		UI		
M11	RX_DV(0)/ EBUS(0)-RX-	I/LI-	RX_DV(0]	I	EBUS(0)-RX-	LI-		27 KΩ PU
M10	RX_ERR(0)	I	RX_ERR(0)	I		UI		
L10	RX_CLK(0)	I	RX_CLK(0)	I		UI		
L9	LINK_MII(0)	I	LINK_MII(0)	I		UI		
J11	PERR(0)/ TRANS(0)/ CLK_MODE[0]	BD	PERR(0)/ TRANS(0)	O/ I	PERR(0)	O	CLK_MODE[0]	
J12	LINKACT(0)/ P_CONF[0]	BD	LINKACT(0)	O	LINKACT(0)	O	P_CONF[0]	

2.8.6 Physical Port 1

Table 31 shows the pins for physical port 1. It can be configured as MII or EBUS and is always available. Use of this port does in no case clash with pins needed for PDI.

Table 31: Physical Port 1

Pin	Pin		MII		EBUS		Configuration Signal	Internal PU/PD
	Name	Dir.	Signal	Dir.	Signal	Dir.		
M3	TX_ENA(1)/ EBUS(1)-TX+	BD/LO+	TX_ENA(1)	O/I	EBUS(1)-TX+	LO+		
L3	TX_D(1)[0]/ TRANS-MODE-ENA	BD	TX_D(1)[0]	O			TRANS_ MODE_ENA	
M2	TX_D(1)[1]/ EBUS(1)-TX-	O/LO-	TX_D(1)[1]	O	EBUS(1)-TX-	LO-		
L2	TX_D(1)[2]/ P_MODE[0]	BD	TX_D(1)[2]	O			P_MODE[0]	
M1	TX_D(1)[3]/ P_MODE[1]	BD	TX_D(1)[3]	O			P_MODE[1]	
L4	RX_D(1)[0]	I	RX_D(1)[0]	I		UI		
M5	RX_D(1)[1]/ EBUS(1)-RX+	I/LI+	RX_D(1)[1]	I	EBUS(1)-RX+	LI+		27 KΩ PD
L5	RX_D(1)[2]	I	RX_D(1)[2]	I		UI		
M6	RX_D(1)[3]	I	RX_D(1)[3]	I		UI		
M4	RX_DV(1)/ EBUS(1)-RX-	I/LI-	RX_DV(1)	I	EBUS(1)-RX-	LI-		27 KΩ PU
L6	RX_ERR(1)	I	RX_ERR(1)	I		UI		
K4	RX_CLK(1)	I	RX_CLK(1)	I		UI		
K3	LINK_MII(1)	I	LINK_MII(1)	I		UI		
K2	PERR(1)/ TRANS(1)/ CLK_MODE(1)	BD	PERR(1)/ TRANS(1)	O/ I	PERR(1)	O	CLK_MODE[1]	
L1	LINKACT(1)/ P_CONF(1)	BD	LINKACT(1)	O	LINKACT(1)	O	P_CONF[1]	

2.8.7 Physical Port 2 / PDI byte 4

Table 32 shows the pins for physical port 2 or for PDI byte 4 (PDI[39:32]). If used as communication port it can be configured as MII or EBUS.

Table 32: Physical Port 2/PDI byte 4

	Pin	Pin		PDI		MII		EBUS		Configu-ration Signal	Int. PU/PD
		Name	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.		
PDI Byte 4	D1	PDI[32]/ TX_D(2)[3]	BD	PDI[32]	BD	TX_D(2)[3]	O	PDI[32]	BD		
	D2	PDI[33]/ TX_D(2)[2]	BD	PDI[33]	BD	TX_D(2)[2]	O	PDI[33]	BD		
	E2	PDI[34]/ TX_D(2)[0]/ CTRL_STATUS_MOVE	BD	PDI[34]	BD	TX_D(2)[0]	O	PDI[34]	BD	CTRL_STATUS_MOVE	
	G1	PDI[35]/ RX_ERR(2)	BD	PDI[35]	BD	RX_ERR(2)	I	PDI[35]	BD		
	G2	PDI[36]/ RX_CLK(2)	BD	PDI[36]	BD	RX_CLK(2)	I	PDI[36]	BD		
	H2	PDI[37]/ RX_D(2)[0]	BD	PDI[37]	BD	RX_D(2)[0]	I	PDI[37]	BD		
	J2	PDI[38]/ RX_D(2)[2]	BD	PDI[38]	BD	RX_D(2)[2]	I	PDI[38]	BD		
	K1	PDI[39]/ RX_D(2)[3]	BD	PDI[39]	BD	RX_D(2)[3]	I	PDI[39]	BD		

Table 33: Physical Port 2

Pin	Pin		Only 2 ports		MII		EBUS		Configu-ration Signal	Int. PU/PD
	Name	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.		
F1	TX_ENA(2)/ EBUS(2)-TX+	BD/ LO+		UI	TX_ENA(2)	O/I	EBUS(2)-TX+	LO+		
E1	TX_D(2)[1]/ EBUS(2)-TX-	O/ LO-		n.c.	TX_D(2)[1]	O	EBUS(2)-TX-	LO-		
H1	RX_DV(2)/ EBUS(2)-RX-	I/LI-		UI	RX_DV(2)	I	EBUS(2)-RX-	LI-		27 KΩ PU
J1	RX_D(2)[1]/ EBUS(2)-RX+	I/LI+		UI	RX_D(2)[1]	I	EBUS(2)-RX+	LI+		27 KΩ PD
C3	PERR(2)/ TRANS(2)/ PHYAD_OFF	BD		O	PERR(2)/ TRANS(2)	O/ I	PERR(2)	O	PHYAD_OFF	
E3	LINKACT(2)/ P_CONF[2]	BD		O	LINKACT(2)	O	LINKACT(2)	O	P_CONF[2]	
F2	LINK_MII(2)/ CLK25OUT1	BD	CLK25OUT1	O	LINK_MII(2)	I	CLK25OUT1	O		

2.8.8 Physical Port 3 / PDI Bytes 2/3

Table 34 shows the pins for physical port 3 or for PDI bytes 2/3 (PDI[23:16], PDI[3117]). If used as communication port it can be configured as MII or EBUS.

Table 34: Physical Port 3 / PDI

	Pin	Pin		PDI		MII		EBUS		Configuration Signal	Int. PU/PD
		Name	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.		
PDI Byte 2	A7	PDI[16]/ RX_ERR(3)	BD	PDI[16]	BD	RX_ERR(3)	I	PDI[16]	BD		
	B7	PDI[17]/ RX_CLK(3)	BD	PDI[17]	BD	RX_CLK(3)	I	PDI[17]	BD		
	A6	PDI[18]/ RX_D(3)[0]	BD	PDI[18]	BD	RX_D(3)[0]	I	PDI[18]	BD		
	B6	PDI[19]/ RX_D(3)[2]	BD	PDI[19]	BD	RX_D(3)[2]	I	PDI[19]	BD		
	A5	PDI[20]/ RX_D(3)[3]	BD	PDI[20]	BD	RX_D(3)[3]	I	PDI[20]	BD		
	B5	PDI[21]/ LINK_MII(3)	BD	PDI[21]	BD	LINK_MII(3)	I	PDI[21]	BD		
	A4	PDI[22]/ TX_D(3)[3]	BD	PDI[22]	BD	TX_D(3)[3]	O	PDI[22]	BD		
	B4	PDI[23]/ TX_D(3)[2]	BD	PDI[23]	BD	TX_D(3)[2]	O	PDI[23]	BD		
PDI Byte 3	A3	PDI[24]/ TX_D(3)[1]/ EBUS(3)-TX-	BD/ LO-	PDI[24]	BD	TX_D(3)[1]	O	EBUS(3)-TX-	LO-		
	B3	PDI[25]/ TX_D(3)[0]	BD	PDI[25]	BD	TX_D(3)[0]	O		UI		
	A2	PDI[26]/ TX_ENA(3)/ EBUS(3)-TX+	BD/ LO+	PDI[26]	BD	TX_ENA(3)	O/I	EBUS(3)-TX+	LO+		
	A1	PDI[27]/ RX_DV(3)/ EBUS(3)-RX-	BD/ LI-	PDI[27]	BD	RX_DV(3)	I	EBUS(3)-RX-	LI-		27 KΩ PU
	B2	PDI[28]/ PERR(3)/ TRANS(3)	BD	PDI[28]	BD	PERR(3)/ TRANS(3)	O/ I	PERR(3)	O	RESER-VED	
	B1	PDI[29]/ RX_D(3)[1]/ EBUS(3)-RX+	BD/ LI+	PDI[29]	BD	RX_D(3)[1]	I	EBUS(3)-RX+	LI+		27 KΩ PD
	C2	PDI[30]/ LINKACT(3)/ P_CONF(3)	BD	PDI[30]	BD	LINKACT(3)	O	LINKACT(3)	O	P_CONF [3]	
	C1	PDI[31]/ CLK25OUT2	BD	PDI[31]/ CLK25 OUT2	BD	CLK25OUT2	O	CLK25OUT2	O		

2.8.9 PDI Bytes 0/1

Table 35 shows PDI byte 0 and byte 1 (PDI[15:0]).

The direction of all PDI pins depends on the PDI configuration stored in the ESI EEPROM.

Table 35: PDI pins

		Pin	Pin		PDI, CLK_MODE=00		PDI, CLK_MODE/=00	
			Name	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte O	PDI[7:0]	D12	PDI[0]	BD	PDI[0]	BD	PDI[0]	BD
		D11	PDI[1]	BD	PDI[1]	BD	PDI[1]	BD
		C12	PDI[2]	BD	PDI[2]	BD	PDI[2]	BD
		C11	PDI[3]	BD	PDI[3]	BD	PDI[3]	BD
		B12	PDI[4]	BD	PDI[4]	BD	PDI[4]	BD
		C10	PDI[5]	BD	PDI[5]	BD	PDI[5]	BD
		A12	PDI[6]	BD	PDI[6]	BD	PDI[6]	BD
		B11	PDI[7]/CPU_CLK	BD	PDI[7]	BD	CPU_CLK	O
PDI Byte 1	PDI[8:15]	A11	PDI[8]	BD	PDI[8]	BD	PDI[8]	BD
		B10	PDI[9]	BD	PDI[9]	BD	PDI[9]	BD
		A10	PDI[10]	BD	PDI[10]	BD	PDI[10]	BD
		C9	PDI[11]	BD	PDI[11]	BD	PDI[11]	BD
		A9	PDI[12]	BD	PDI[12]	BD	PDI[12]	BD
		B9	PDI[13]	BD	PDI[13]	BD	PDI[13]	BD
		A8	PDI[14]	BD	PDI[14]	BD	PDI[14]	BD
		B8	PDI[15]	BD	PDI[15]	BD	PDI[15]	BD

2.9 PDI Signal Pinout depending on selected PDI

The PDI signal pinout depends on the selected PDI (ESI EEPROM). The PDI selection and PDI signal pinout is subject to restrictions introduced by the port configuration. Digital I/O and SPI PDI are available in any configuration – although the I/O width can be reduced depending on the configuration. The µController PDIs are only available with up to 3 ports, the data bus width can be reduced depending on the configuration.

Refer to PDI descriptions for further PDI and PDI signal descriptions.

The SPI PDI supports additional general purpose I/O signals, which are not part of the SPI PDI description:

GPO[x]
General purpose output signals.

GPI[x]
General purpose input signals.

2.9.1 Digital I/O Pin Out

Table 36: Mapping of Digital I/O Interface (1)

	Digital IO	PDI signal	2 ports, or 3 ports with min. 1xEBUS		3xMII, 0xEBUS CTRL_STATUS_MOVE=			
					0		1	
			Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	I/O[0]	I/O/BD	I/O[0]	I/O/BD	I/O[0]	I/O/BD
		PDI[1]	I/O[1]	I/O/BD	I/O[1]	I/O/BD	I/O[1]	I/O/BD
		PDI[2]	I/O[2]	I/O/BD	I/O[2]	I/O/BD	I/O[2]	I/O/BD
		PDI[3]	I/O[3]	I/O/BD	I/O[3]	I/O/BD	I/O[3]	I/O/BD
		PDI[4]	I/O[4]	I/O/BD	I/O[4]	I/O/BD	I/O[4]	I/O/BD
		PDI[5]	I/O[5]	I/O/BD	I/O[5]	I/O/BD	I/O[5]	I/O/BD
		PDI[6]	I/O[6]	I/O/BD	I/O[6]	I/O/BD	I/O[6]	I/O/BD
		PDI[7]/ CPU_CLK	I/O[7]/ CPU_CLK	I/O/BD/ O	I/O[7]/ CPU_CLK	I/O/BD/ O	I/O[7]/ CPU_CLK	I/O/BD/ O
PDI Byte 1		PDI[8]	I/O[8]	I/O/BD	I/O[8]	I/O/BD	I/O[8]	I/O/BD
		PDI[9]	I/O[9]	I/O/BD	I/O[9]	I/O/BD	I/O[9]	I/O/BD
		PDI[10]	I/O[10]	I/O/BD	I/O[10]	I/O/BD	I/O[10]	I/O/BD
		PDI[11]	I/O[11]	I/O/BD	I/O[11]	I/O/BD	I/O[11]	I/O/BD
		PDI[12]	I/O[12]	I/O/BD	I/O[12]	I/O/BD	I/O[12]	I/O/BD
		PDI[13]	I/O[13]	I/O/BD	I/O[13]	I/O/BD	I/O[13]	I/O/BD
		PDI[14]	I/O[14]	I/O/BD	I/O[14]	I/O/BD	I/O[14]	I/O/BD
		PDI[15]	I/O[15]	I/O/BD	I/O[15]	I/O/BD	I/O[15]	I/O/BD
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	I/O[16]	I/O/BD	I/O[16]	I/O/BD	I/O[16]	I/O/BD
		PDI[17]	I/O[17]	I/O/BD	I/O[17]	I/O/BD	I/O[17]	I/O/BD
		PDI[18]	I/O[18]	I/O/BD	I/O[18]	I/O/BD	I/O[18]	I/O/BD
		PDI[19]	I/O[19]	I/O/BD	I/O[19]	I/O/BD	I/O[19]	I/O/BD
		PDI[20]	I/O[20]	I/O/BD	I/O[20]	I/O/BD	I/O[20]	I/O/BD
		PDI[21]	I/O[21]	I/O/BD	I/O[21]	I/O/BD	I/O[21]	I/O/BD
		PDI[22]	I/O[22]	I/O/BD	I/O[22]	I/O/BD	I/O[22]	I/O/BD
		PDI[23]	I/O[23]	I/O/BD	I/O[23]	I/O/BD	I/O[23]	I/O/BD
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	I/O[24]	I/O/BD	I/O[24]	I/O/BD	SOF	O
		PDI[25]	I/O[25]	I/O/BD	I/O[25]	I/O/BD	OE_EXT	I
		PDI[26]	I/O[26]	I/O/BD	I/O[26]	I/O/BD	OUTVALID	O
		PDI[27]	I/O[27]	I/O/BD	I/O[27]	I/O/BD	WD_TRIG	O
		PDI[28]	I/O[28]	I/O/BD	I/O[28]	I/O/BD	LATCH_IN	I
		PDI[29]	I/O[29]	I/O/BD	I/O[29]	I/O/BD	OE_CONF	I
		PDI[30]	I/O[30]	I/O/BD	I/O[30]	I/O/BD	EEPROM_ LOADED	O
		PDI[31]/ CLK25OUT2	I/O[31]/ CLK25OUT2	I/O/BD/ O	I/O[31]/ CLK25OUT2	I/O/BD/ O	--/ CLK25OUT2	--/ O
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	SOF	O	MII(2)		MII(2)	
		PDI[33]	OE_EXT	I				
		PDI[34]	OUTVALID	O				
		PDI[35]	WD_TRIG	O				
		PDI[36]	LATCH_IN	I				
		PDI[37]	OE_CONF	I				
		PDI[38]	EEPROM_LOADED	O				
		PDI[39]	--	--				

Table 37: Mapping of Digital I/O Interface (2)

	Digital IO	PDI signal	4 ports, min. 2x EBUS		3xMII, 1xEBUS CTRL_STATUS_MOVE			
					0		1	
			Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	I/O[0]	I/O/BD	I/O[0]	I/O/BD	I/O[0]	I/O/BD
		PDI[1]	I/O[1]	I/O/BD	I/O[1]	I/O/BD	I/O[1]	I/O/BD
		PDI[2]	I/O[2]	I/O/BD	I/O[2]	I/O/BD	I/O[2]	I/O/BD
		PDI[3]	I/O[3]	I/O/BD	I/O[3]	I/O/BD	I/O[3]	I/O/BD
		PDI[4]	I/O[4]	I/O/BD	I/O[4]	I/O/BD	I/O[4]	I/O/BD
		PDI[5]	I/O[5]	I/O/BD	I/O[5]	I/O/BD	I/O[5]	I/O/BD
		PDI[6]	I/O[6]	I/O/BD	I/O[6]	I/O/BD	I/O[6]	I/O/BD
		PDI[7]/ CPU_CLK	I/O[7]/ CPU_CLK	I/O/BD/ O	I/O[7]/ CPU_CLK	I/O/BD/ O	I/O[7]/ CPU_CLK	I/O/BD/ O
PDI Byte 1		PDI[8]	I/O[8]	I/O/BD	I/O[8]	I/O/BD	I/O[8]	I/O/BD
		PDI[9]	I/O[9]	I/O/BD	I/O[9]	I/O/BD	I/O[9]	I/O/BD
		PDI[10]	I/O[10]	I/O/BD	I/O[10]	I/O/BD	I/O[10]	I/O/BD
		PDI[11]	I/O[11]	I/O/BD	I/O[11]	I/O/BD	I/O[11]	I/O/BD
		PDI[12]	I/O[12]	I/O/BD	I/O[12]	I/O/BD	I/O[12]	I/O/BD
		PDI[13]	I/O[13]	I/O/BD	I/O[13]	I/O/BD	I/O[13]	I/O/BD
		PDI[14]	I/O[14]	I/O/BD	I/O[14]	I/O/BD	I/O[14]	I/O/BD
		PDI[15]	I/O[15]	I/O/BD	I/O[15]	I/O/BD	I/O[15]	I/O/BD
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	I/O[16]	I/O/BD	I/O[16]	I/O/BD	SOF	O
		PDI[17]	I/O[17]	I/O/BD	I/O[17]	I/O/BD	OE_EXT	I
		PDI[18]	I/O[18]	I/O/BD	I/O[18]	I/O/BD	OUTVALID	O
		PDI[19]	I/O[19]	I/O/BD	I/O[19]	I/O/BD	WD_TRIG	O
		PDI[20]	I/O[20]	I/O/BD	I/O[20]	I/O/BD	LATCH_IN	I
		PDI[21]	I/O[21]	I/O/BD	I/O[21]	I/O/BD	OE_CONF	I
		PDI[22]	I/O[22]	I/O/BD	I/O[22]	I/O/BD	EEPROM_ LOADED	O
		PDI[23]	I/O[23]	I/O/BD	I/O[23]	I/O/BD	--	--
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	EBUS(3)		EBUS(3)		EBUS(3)	
		PDI[25]						
		PDI[26]						
		PDI[27]						
		PDI[28]						
		PDI[29]						
		PDI[30]						
		PDI[31]/ CLK25OUT2						
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	SOF	O	MII(2)		MII(2)	
		PDI[33]	OE_EXT	I				
		PDI[34]	OUTVALID	O				
		PDI[35]	WD_TRIG	O				
		PDI[36]	LATCH_IN	I				
		PDI[37]	OE_CONF	I				
		PDI[38]	EEPROM_LOADED	O				
		PDI[39]	--	--				

Table 38: Mapping of Digital I/O Interface (3)

	Digital IO	PDI signal	4xMII CTRL_STATUS_MOVE=			
			0		1	
			Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	I/O[0]	I/O/BD	I/O[0]	I/O/BD
		PDI[1]	I/O[1]	I/O/BD	I/O[1]	I/O/BD
		PDI[2]	I/O[2]	I/O/BD	I/O[2]	I/O/BD
		PDI[3]	I/O[3]	I/O/BD	I/O[3]	I/O/BD
		PDI[4]	I/O[4]	I/O/BD	I/O[4]	I/O/BD
		PDI[5]	I/O[5]	I/O/BD	I/O[5]	I/O/BD
		PDI[6]	I/O[6]	I/O/BD	I/O[6]	I/O/BD
		PDI[7]/CPU_CLK	I/O[7]/CPU_CLK	I/O/BD/O	I/O[7]/CPU_CLK	I/O/BD/O
PDI Byte 1		PDI[8]	I/O[8]	I/O/BD	SOF	O
		PDI[9]	I/O[9]	I/O/BD	OE_EXT	I
		PDI[10]	I/O[10]	I/O/BD	OUTVALID	O
		PDI[11]	I/O[11]	I/O/BD	WD_TRIG	O
		PDI[12]	I/O[12]	I/O/BD	LATCH_IN	I
		PDI[13]	I/O[13]	I/O/BD	OE_CONF	I
		PDI[14]	I/O[14]	I/O/BD	EEPROM_LOADED	O
		PDI[15]	I/O[15]	I/O/BD	--	--
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	MII(3)		MII(3)	
		PDI[17]				
		PDI[18]				
		PDI[19]				
		PDI[20]				
		PDI[21]				
		PDI[22]				
		PDI[23]				
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	MII(3)		MII(3)	
		PDI[25]				
		PDI[26]				
		PDI[27]				
		PDI[28]				
		PDI[29]				
		PDI[30]				
		PDI[31]/CLK25OUT2				
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	MII(2)		MII(2)	
		PDI[33]				
		PDI[34]				
		PDI[35]				
		PDI[36]				
		PDI[37]				
		PDI[38]				
		PDI[39]				

2.9.2 8/16 Bit asynchronous µController

Figure 2: Mapping of asynchronous µC Interface to Port

	Async. µC	PDI signal	2 ports, or 3 ports with min. 1xEBUS				3xMII, 0xEBUS	
			8 bit		16 bit		8 bit	
			Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	CS	I	CS	I	CS	I
		PDI[1]	RD	I	RD	I	RD	I
		PDI[2]	WR	I	WR	I	WR	I
		PDI[3]	BUSY	O	BUSY	O	BUSY	O
		PDI[4]	IRQ	O	IRQ	O	IRQ	O
		PDI[5]	BHE	I	BHE	I	BHE	I
		PDI[6]	EEPROM_LOADED	O	EEPROM_LOADED	O	EEPROM_LOADED	O
		PDI[7]/ CPU_CLK	ADR[15]/ CPU_CLK	I/ O	ADR[15]/ CPU_CLK	I/ O	ADR[15]/ CPU_CLK	I/ O
PDI Byte 1		PDI[8]	ADR[14]	I	ADR[14]	I	ADR[14]	I
		PDI[9]	ADR[13]	I	ADR[13]	I	ADR[13]	I
		PDI[10]	ADR[12]	I	ADR[12]	I	ADR[12]	I
		PDI[11]	ADR[11]	I	ADR[11]	I	ADR[11]	I
		PDI[12]	ADR[10]	I	ADR[10]	I	ADR[10]	I
		PDI[13]	ADR[9]	I	ADR[9]	I	ADR[9]	I
		PDI[14]	ADR[8]	I	ADR[8]	I	ADR[8]	I
		PDI[15]	ADR[7]	I	ADR[7]	I	ADR[7]	I
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	ADR[6]	I	ADR[6]	I	ADR[6]	I
		PDI[17]	ADR[5]	I	ADR[5]	I	ADR[5]	I
		PDI[18]	ADR[4]	I	ADR[4]	I	ADR[4]	I
		PDI[19]	ADR[3]	I	ADR[3]	I	ADR[3]	I
		PDI[20]	ADR[2]	I	ADR[2]	I	ADR[2]	I
		PDI[21]	ADR[1]	I	ADR[1]	I	ADR[1]	I
		PDI[22]	ADR[0]	I	ADR[0]	I	ADR[0]	I
		PDI[23]	DATA[0]	BD	DATA[0]	BD	DATA[0]	BD
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	DATA[1]	BD	DATA[1]	BD	DATA[1]	BD
		PDI[25]	DATA[2]	BD	DATA[2]	BD	DATA[2]	BD
		PDI[26]	DATA[3]	BD	DATA[3]	BD	DATA[3]	BD
		PDI[27]	DATA[4]	BD	DATA[4]	BD	DATA[4]	BD
		PDI[28]	DATA[5]	BD	DATA[5]	BD	DATA[5]	BD
		PDI[29]	DATA[6]	BD	DATA[6]	BD	DATA[6]	BD
		PDI[30]	DATA[7]	BD	DATA[7]	BD	DATA[7]	BD
		PDI[31]/ CLK25OUT2	--/ CLK25OUT2	--/ O	--/ CLK25OUT2	--/ O	--/ CLK25OUT2	--/ O
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	--	--	DATA[8]	BD	MII(2)	
		PDI[33]	--	--	DATA[9]	BD		
		PDI[34]	--	--	DATA[10]	BD		
		PDI[35]	--	--	DATA[11]	BD		
		PDI[36]	--	--	DATA[12]	BD		
		PDI[37]	--	--	DATA[13]	BD		
		PDI[38]	--	--	DATA[14]	BD		
		PDI[39]	--	--	DATA[15]	BD		

2.9.3 8/16 Bit synchronous µController

Table 39: Mapping of synchronous µC Interface to Port

	Sync. µC	PDI signal	2 ports, or 3 ports with min. 1xEBUS				3xMII, 0xEBUS	
			8 bit		16 bit		8 bit	
			Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	CS	I	CS	I	CS	I
		PDI[1]	TS	I	TS	I	TS	I
		PDI[2]	RD/nWR	I	RD/nWR	I	RD/nWR	I
		PDI[3]	TA	O	TA	O	TA	O
		PDI[4]	IRQ	O	IRQ	O	IRQ	O
		PDI[5]	BHE	I	BHE	I	BHE	I
		PDI[6]	EEPROM_LOADED	O	EEPROM_LOADED	O	EEPROM_LOADED	O
		PDI[7]/ CPU_CLK	ADR[15]/ CPU_CLK	I/ O	ADR[15]/ CPU_CLK	I/O	ADR[15]/ CPU_CLK	I/ O
PDI Byte 1		PDI[8]	ADR[14]	I	ADR[14]	I	ADR[14]	I
		PDI[9]	ADR[13]	I	ADR[13]	I	ADR[13]	I
		PDI[10]	ADR[12]	I	ADR[12]	I	ADR[12]	I
		PDI[11]	ADR[11]	I	ADR[11]	I	ADR[11]	I
		PDI[12]	ADR[10]	I	ADR[10]	I	ADR[10]	I
		PDI[13]	ADR[9]	I	ADR[9]	I	ADR[9]	I
		PDI[14]	ADR[8]	I	ADR[8]	I	ADR[8]	I
		PDI[15]	ADR[7]	I	ADR[7]	I	ADR[7]	I
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	ADR[6]	I	ADR[6]	I	ADR[6]	I
		PDI[17]	ADR[5]	I	ADR[5]	I	ADR[5]	I
		PDI[18]	ADR[4]	I	ADR[4]	I	ADR[4]	I
		PDI[19]	ADR[3]	I	ADR[3]	I	ADR[3]	I
		PDI[20]	ADR[2]	I	ADR[2]	I	ADR[2]	I
		PDI[21]	ADR[1]	I	ADR[1]	I	ADR[1]	I
		PDI[22]	ADR[0]	I	ADR[0]	I	ADR[0]	I
		PDI[23]	DATA[0]	BD	DATA[0]	BD	DATA[0]	BD
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	DATA[1]	BD	DATA[1]	BD	DATA[1]	BD
		PDI[25]	DATA[2]	BD	DATA[2]	BD	DATA[2]	BD
		PDI[26]	DATA[3]	BD	DATA[3]	BD	DATA[3]	BD
		PDI[27]	DATA[4]	BD	DATA[4]	BD	DATA[4]	BD
		PDI[28]	DATA[5]	BD	DATA[5]	BD	DATA[5]	BD
		PDI[29]	DATA[6]	BD	DATA[6]	BD	DATA[6]	BD
		PDI[30]	DATA[7]	BD	DATA[7]	BD	DATA[7]	BD
		PDI[31]	CPU_CLK_IN	I	CPU_CLK_IN	I	CPU_CLK_IN	I
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	--	--	DATA[8]	BD	MII(2)	
		PDI[33]	--	--	DATA[9]	BD		
		PDI[34]	--	--	DATA[10]	BD		
		PDI[35]	--	--	DATA[11]	BD		
		PDI[36]	--	--	DATA[12]	BD		
		PDI[37]	--	--	DATA[13]	BD		
		PDI[38]	--	--	DATA[14]	BD		
		PDI[39]	--	--	DATA[15]	BD		

2.9.4 SPI Pin Out

Figure 3: Mapping of SPI Interface to Port (1)

	SPI	PDI signal	2 ports, or 3 ports with min. 1xEBUS		3xMII, 0xEBUS	
			Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	SPI_CLK	I	SPI_CLK	I
		PDI[1]	SPI_SEL	I	SPI_SEL	I
		PDI[2]	SPI_DI	I	SPI_DI	I
		PDI[3]	SPI_DO	O	SPI_DO	O
		PDI[4]	SPI_IRQ	O	SPI_IRQ	O
		PDI[5]	--	--	--	--
		PDI[6]	EEPROM_LOADED	O	EEPROM_LOADED	O
		PDI[7]/CPU_CLK	--/CPU_CLK	--/O	--/CPU_CLK	--/O
PDI Byte 1		PDI[8]	GPO[0]	O	GPO[0]	O
		PDI[9]	GPO[1]	O	GPO[1]	O
		PDI[10]	GPO[2]	O	GPO[2]	O
		PDI[11]	GPO[3]	O	GPO[3]	O
		PDI[12]	GPI[0]	I	GPI[0]	I
		PDI[13]	GPI[1]	I	GPI[1]	I
		PDI[14]	GPI[2]	I	GPI[2]	I
		PDI[15]	GPI[3]	I	GPI[3]	I
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	GPO[4]	O	GPO[4]	O
		PDI[17]	GPO[5]	O	GPO[5]	O
		PDI[18]	GPO[6]	O	GPO[6]	O
		PDI[19]	GPO[7]	O	GPO[7]	O
		PDI[20]	GPI[4]	I	GPI[4]	I
		PDI[21]	GPI[5]	I	GPI[5]	I
		PDI[22]	GPI[6]	I	GPI[6]	I
		PDI[23]	GPI[7]	I	GPI[7]	I
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	GPO[8]	O	GPO[8]	O
		PDI[25]	GPO[9]	O	GPO[9]	O
		PDI[26]	GPO[10]	O	GPO[10]	O
		PDI[27]	GPO[11]	O	GPO[11]	O
		PDI[28]	GPI[8]	I	GPI[8]	I
		PDI[29]	GPI[9]	I	GPI[9]	I
		PDI[30]	GPI[10]	I	GPI[10]	I
		PDI[31]/CLK25OUT2	GPI[11]/CLK25OUT2	I/O	GPI[11]/CLK25OUT2	I/O
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	GPO[12]	O	MII(2)	
		PDI[33]	GPO[13]	O		
		PDI[34]	GPO[14]	O		
		PDI[35]	GPO[15]	O		
		PDI[36]	GPI[12]	I		
		PDI[37]	GPI[13]	I		
		PDI[38]	GPI[14]	I		
		PDI[39]	GPI[15]	I		

Table 40: Mapping of SPI Interface to Port (2)

	SPI	PDI signal	4 ports, min. 2x EBUS		3xMII, 1xEBUS		4xMII	
			Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI Byte 0	PDI[15:0]	PDI[0]	SPI_CLK	I	SPI_CLK	I	SPI_CLK	I
		PDI[1]	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I
		PDI[2]	SPI_DI	I	SPI_DI	I	SPI_DI	I
		PDI[3]	SPI_DO	O	SPI_DO	O	SPI_DO	O
		PDI[4]	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O
		PDI[5]	--	--	--	--	--	--
		PDI[6]	EEPROM _LOADED	O	EEPROM _LOADED	O	EEPROM _LOADED	O
		PDI[7]/CPU_CLK	--/CPU_CLK	--/O	--/CPU_CLK	--/O	--/CPU_CLK	--/O
PDI Byte 1		PDI[8]	GPO[0]	O	GPO[0]	O	GPO[0]	O
		PDI[9]	GPO[1]	O	GPO[1]	O	GPO[1]	O
		PDI[10]	GPO[2]	O	GPO[2]	O	GPO[2]	O
		PDI[11]	GPO[3]	O	GPO[3]	O	GPO[3]	O
		PDI[12]	GPI[0]	I	GPI[0]	I	GPI[0]	I
		PDI[13]	GPI[1]	I	GPI[1]	I	GPI[1]	I
		PDI[14]	GPI[2]	I	GPI[2]	I	GPI[2]	I
		PDI[15]	GPI[3]	I	GPI[3]	I	GPI[3]	I
PDI Byte 2	PDI[23:16]/ MII(3)	PDI[16]	GPO[4]	O	GPO[4]	O	MII(3)	
		PDI[17]	GPO[5]	O	GPO[5]	O		
		PDI[18]	GPO[6]	O	GPO[6]	O		
		PDI[19]	GPO[7]	O	GPO[7]	O		
		PDI[20]	GPI[4]	I	GPI[4]	I		
		PDI[21]	GPI[5]	I	GPI[5]	I		
		PDI[22]	GPI[6]	I	GPI[6]	I		
		PDI[23]	GPI[7]	I	GPI[7]	I		
PDI Byte 3	PDI[31:24]/ MII(3)/ EBUS(3)	PDI[24]	EBUS(3)		EBUS(3)		MII(3)	
		PDI[25]						
		PDI[26]						
		PDI[27]						
		PDI[28]						
		PDI[29]						
		PDI[30]						
		PDI[31]/CLK25OUT2						
PDI Byte 4	PDI[39:32]/ MII(2)	PDI[32]	GPO[12]	O	MII(2)		MII(2)	
		PDI[33]	GPO[13]	O				
		PDI[34]	GPO[14]	O				
		PDI[35]	GPO[15]	O				
		PDI[36]	GPI[12]	I				
		PDI[37]	GPI[13]	I				
		PDI[38]	GPI[14]	I				
		PDI[39]	GPI[15]	I				

2.10 Power Supply

The ET1100 supports different power supply and I/O voltage options with either 3.3V I/O or 5V I/O and optionally single or dual power supply.

The $V_{CCI/O}$ supply voltage directly determines the I/O voltages for all inputs and outputs, i.e., with 3.3V $V_{CCI/O}$, the inputs are 3.3V I/O compliant and they are not 5V tolerant ($V_{CCI/O}$ has to be 5V if 5V compatible I/Os are required).

The core supply voltages $V_{CC\ Core}/V_{CC\ PLL}$ (nom. 2.5V) are generated from $V_{CC\ I/O}$ by an internal LDO. $V_{CC\ PLL}$ is always equal to $V_{CC\ Core}$. The internal LDO can not be switched off, it stops operating if external supply voltage is higher than the internal LDO output voltage, thus the external supply voltage ($V_{CC\ Core}/V_{CC\ PLL}$) has to be higher (at least 0.1V) than the internal LDO output voltage.

Using the internal LDO increases power dissipation, and power consumption for 5V I/O voltage is significantly higher than power consumption for 3.3V I/O. It is highly recommended to use 3.3V I/O voltage and the internal LDO for $V_{CC\ Core}/V_{CC\ PLL}$.

Voltage stabilization capacitors at all power pairs are necessary.

Table 41: Power supply options

$V_{CC\ I/O}$	$V_{CC\ Core}/V_{CC\ PLL}$	Input signals	Output signals	Comment
3.3V	**Internal LDO (2.5V)**	**3.3V only**	**3.3V only**	**Single power supply, low power dissipation**
3.3V	External 2.5V	3.3V only	3.3V only	Dual power supply, lowest power dissipation
Not recommended:				
5V	Internal LDO (2.5V)	5V only	5V only	Single power supply, highest power dissipation
5V	External 2.5V	5V only	5V only	Dual power supply, high power dissipation

2.10.1 I/O Power Supply

The I/O power supply pins can be connected to either 3.3V or 5.0V, depending on the desired interface voltage. All power pins must be connected, and voltage stabilization capacitors at $V_{CCI/O}$/$GND_{I/O}$ power pairs are necessary.

Table 42: I/O power supply

Pin	Pin name
C5	$V_{CC\ I/O}$
D5	$GND_{I/O}$
D3	$V_{CC\ I/O}$
D4	$GND_{I/O}$
J3	$V_{CC\ I/O}$
J4	$GND_{I/O}$
K5	$V_{CC\ I/O}$
J5	$GND_{I/O}$
K8	$V_{CC\ I/O}$
J8	$GND_{I/O}$
J10	$V_{CC\ I/O}$
J9	$GND_{I/O}$
F10	$V_{CC\ I/O}$*
F9	$GND_{I/O}$*
D10	$V_{CC\ I/O}$
D9	$GND_{I/O}$
E9	$V_{CC\ I/O}$*
H4	$GND_{I/O}$
F3	$V_{CC\ I/O}$
K9	$GND_{I/O}$
H9	$V_{CC\ I/O}$

c) NOTE: These pins are most adjacent to the internal LDO – this should be taken into account for voltage stabilization.

2.10.2 Logic Core Power Supply

Table 43 shows the pins for core power supply. Core supply voltage is 2.5V. The core power is either generated by the internal LDO, which is sourced by the I/O power supply, or externally. In both cases, voltage stabilization capacitors have to be connected to $V_{CC\ Core}$/GND_{Core} power pairs.

Table 43: Core Power Supply

Pin	Pin name
C6	$V_{CC\ Core}$
D6	GND_{Core}
K6	$V_{CC\ Core}$
J6	GND_{Core}
K7	$V_{CC\ Core}$
J7	GND_{Core}
C7	$V_{CC\ Core}$
D7	GND_{Core}

2.10.3 PLL Power Supply

Table 44 shows the pins for PLL power supply. PLL supply voltage is 2.5V. The PLL power is either generated by the internal LDO, which is sourced by the I/O power supply, or externally. In both cases, voltage stabilization capacitors have to be connected to $V_{CC\ PLL}$/GND_{PLL}.

Table 44: PLL Power Supply

Pin	Pin name
G10	$V_{CC\ PLL}$
G9	GND_{PLL}

2.11 Reserved Pins

Table 45 shows reserved Pins which are not used on the ET1100 and have to be connected to $GND_{I/O}$.

Table 45: Reserved Pins

Pin	Pin name	Dir.	Connect to
E4	Res. [0]	I	$GND_{I/O}$
G3	Res. [1]	I	$GND_{I/O}$
G4	Res. [2]	I	$GND_{I/O}$
E10	Res. [3]	I	$GND_{I/O}$
C8	Res. [4]	I	$GND_{I/O}$
H10	Res. [5]	I	$GND_{I/O}$
F4	Res. [6]	I	$GND_{I/O}$
D8	Res. [7]	I	$GND_{I/O}$

3 MII Interface

The ET1100 is connected with Ethernet PHYs using the MII interfaces. The MII interfaces of the ET1100 are optimized for low processing/forwarding delays by omitting a transmit FIFO. To allow this, the ET1100 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 ET1100:

- The clock source of the PHYs is either CLK25OUT1/2 of the ET1100, or the clock signal that is connected to OSC_IN if a quartz oscillator is used.
- The TX_CLK signal of the PHYs is not connected to the ET1100. The ET1100 does not use the MII interface for link detection or link configuration.

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

3.1 MII Interface Signals

The MII interface of the ET1100 has the following signals:

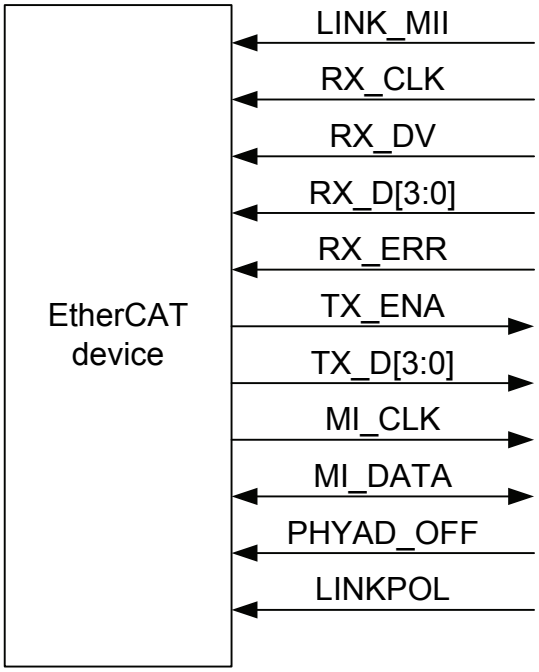


Figure 4: MII Interface signals

Table 46: 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
RX_D[3:0]	IN	Receive data (alias RXD)
RX_ERR	IN	Receive error (alias RX_ER)
TX_ENA	OUT	Transmit enable (alias TX_EN)
TX_D[3:0]	OUT	Transmit data (alias TXD)
MI_CLK	OUT	Management Interface clock (alias MCLK)
MI_DATA	BIDIR	Management Interface data (alias MDIO)
PHYAD_OFF	IN	Configuration: PHY address offset
LINKPOL	IN	Configuration: LINK_MII polarity

MI_DATA should have an external pull-up resistor (4.7 kΩ recommended for ESCs). MI_CLK is driven rail-to-rail, idle value is High.

3.2 PHY Address Configuration

The ET1100 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 0-3 are used.

A PHY address offset of 16 can be applied which moves the PHY addresses to 16-19 by inverting the MSB of the PHY address internally.

If both alternatives can not be used, the PHYs should be configured to use an actual PHY address offset of 1, i.e., PHY addresses 1-4. The PHY address offset configuration of the ET1100 remains 0. If configured PHY address offset or actual PHY address offset are not zero, enhanced link detection can not be used and has to be disabled.

Refer to Section I for more details about PHY addressing.

3.3 TX Shift Compensation

Since ET1100 and the Ethernet PHY share the same clock source, TX_CLK from the PHY has a fixed phase relation to TX_ENA/TX_D[3:0] from the ET1100. Thus, TX_CLK is not connected and the delay of a TX FIFO inside the ET1100 is saved. The phase shift between TX_CLK and TX_ENA/TX_D[3:0] can be compensated by an appropriate value for TX Shift, which will delay TX_ENA/TX_D[3:0] by 0, 10, 20, or 30 ns.

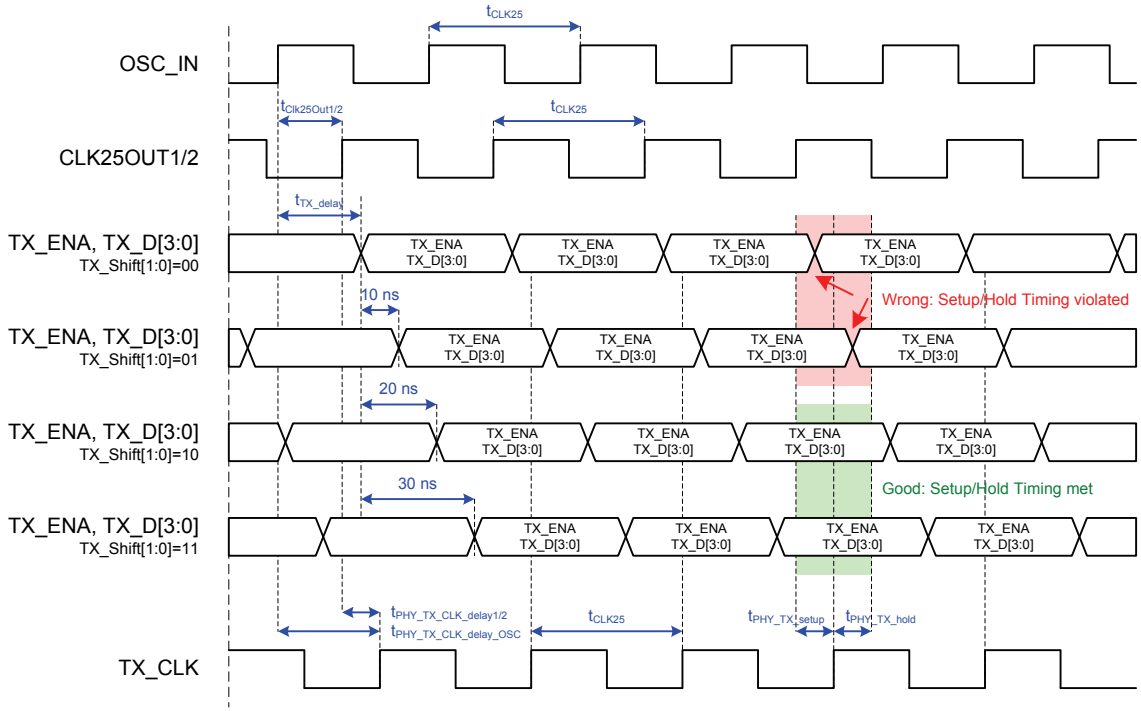


Figure 5: TX Shift Timing Diagram

Table 47: TX Shift Timing characteristics

Parameter	Comment
t_{CLK25}	25 MHz clock source period (OSC_IN, see f_{CLK25})
$t_{CLK25OUT1/2}$	CLK25OUT1/2 delay after OSC_IN (refer to AC characteristics)
t_{TX_delay}	TX_ENA/TX_D[3:0] delay after rising edge of OSC_IN (refer to AC characteristics)
$t_{PHY\_TX\_CLK\_delay1/2}$	Delay between PHY clock source CLK25OUT1/2 and TX_CLK output of the PHY, PHY dependent.
$t_{PHY\_TX\_CLK\_delay\_OSC}$	Delay between PHY clock source OSC_IN and TX_CLK output of the PHY, PHY dependent.
$t_{PHY\_TX\_setup}$	PHY setup requirement: TX_ENA/TX_D[3:0] with respect to TX_CLK. (PHY dependent, IEEE802.3 limit is 15 ns)
$t_{PHY\_TX\_hold}$	PHY hold requirement: TX_ENA/TX_D[3:0] with respect to TX_CLK. (PHY dependent, IEEE802.3 limit is 0 ns)

NOTE: TX Shift can be adjusted by displaying TX_CLK of a PHY and TX_ENA/TX_D[3:0] on an oscilloscope. TX_ENA/TX_D 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). Configure TX Shift so that TX_ENA/TX_D[3:0] change near the middle of this range. It is sufficient to check just one of the TX_ENA/TX_D[3:0] signals, because they are nearly generated at the same time.

3.4 Timing specifications

Table 48: MII timing characteristics

Parameter	Min	Typ	Max	Comment
t_{RX_CLK}	40 ns ± 100 ppm			RX_CLK period (100 ppm with maximum FIFO Size only)
t_{RX_setup}	9 ns			RX_DV/RX_DATA/RX_D[3:0] valid before rising edge of RX_CLK
t_{RX_hold}	3 ns			RX_DV/RX_DATA/RX_D[3:0] valid after rising edge of RX_CLK
t_{Clk}		~ 1.44 µs		MI_CLK period ($f_{Clk} \approx 700$ kHz)
t_{Write}		~ 92.16 µs		MI Write access time
t_{Read}		~ 91.44 us		MI Read access time

NOTE: For MI timing diagrams refer to Section I.

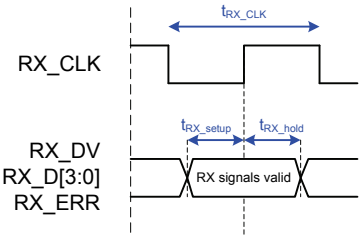


Figure 6: MII timing RX signals

4 EBUS/LVDS Interface

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

4.1 EBUS Interface Signals

The EBUS interface of the ET1100 has the following signals:

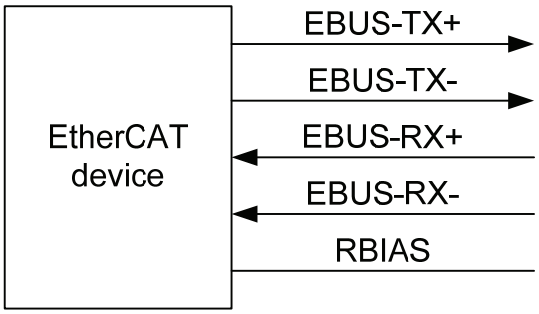


Figure 7: EBUS Interface Signals

Table 49: EBUS Interface signals

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

NOTE: An external LVDS termination with an impedance of 100 Ω between EBUS-RX+ and EBUS-RX- is necessary for EBUS ports. EBUS-RX+ incorporates a pull-down resistor and EBUS-RX- incorporated a pull-up resistor.

5 PDI description

Table 50: Available PDIs for ET1100

PDI number (PDI Control register 0x0140[7:0])	PDI name	ET1100
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	x
11	8 Bit sync. µC	x
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 or OPB)	
Others	Reserved	

5.1 Digital I/O Interface

5.1.1 Interface

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

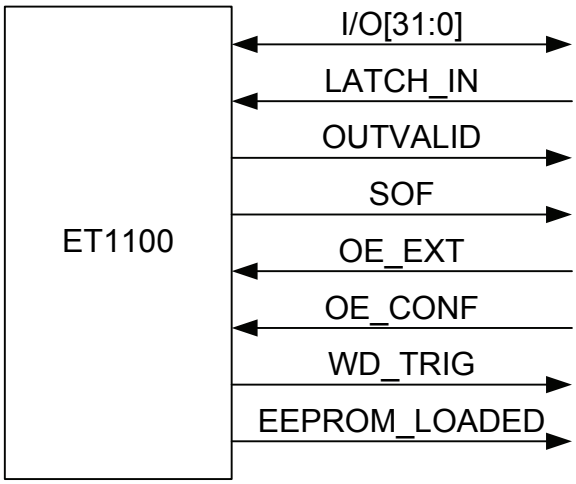


Figure 8: ET1100 Digital I/O signals

Table 51: ET1100 Digital I/O signals

Signal	Direction	Description	Signal polarity
I/O[31:0]	IN/OUT/BIDIR	Input/Output or Bidirectional 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
OE_CONF	IN	Output Enable Configuration	
WD_TRIG	OUT	Watchdog Trigger	act. high
EEPROM_LOADED	OUT	PDI is active, EEPROM is loaded	act. high

5.1.2 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.

5.1.3 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.

5.1.4 Digital Outputs

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

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

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

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

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

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

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

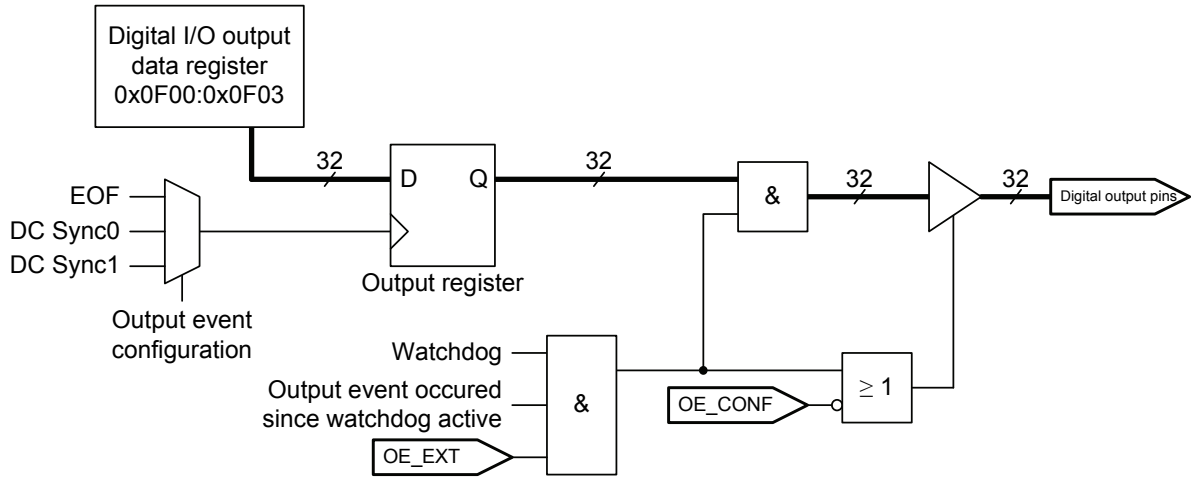


Figure 9: Digital Output Principle Schematic

NOTE: The Digital Outputs are not driven (high impedance) until the EEPROM is loaded. Depending on the configuration, the Digital Outputs are also not driven if the Watchdog is expired or if the outputs are disabled. This behaviour has to be taken into account when using digital output signals.

5.1.5 Bidirectional mode

In bidirectional mode, all DATA signals are bidirectional (individual input/output configuration is ignored). Input signals are connected to the ESC via series resistors, output signals are driven actively by the ESC. Output signals are permanently available if they are latched with OUTVALID (Flip-Flop or Latch).

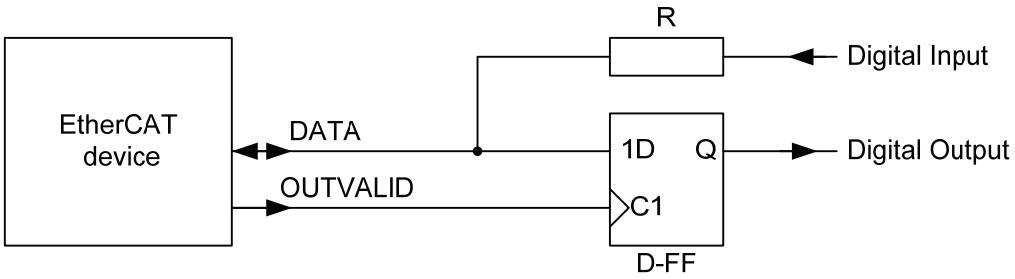


Figure 10: Bidirectional mode: Input/Output connection (R=4.7kΩ recommended)

Input sample event and output update event can be configured as described in the Digital Inputs/Digital Outputs chapter.

An output event is signaled by a pulse on OUTVALID even if the digital outputs remain unchanged. Overlapping input and output events will lead to corrupt input data.

5.1.6 Output Enable/Output Configuration

The ET1100 has an Output Enable signal OE_EXT and an Output Configuration signal OE_CONF. With the OE_EXT signal, the I/O signals can be cleared/put into a high impedance state. OE_CONF controls the output driver's behavior after the output enable signal OE_EXT is set to low or the SyncManager Watchdog is expired (and not disabled).

Table 52: Output Enable/Output Configuration combinations

OE_CONF	OE_EXT	
	0	1
0	I/O driver: ON I/O: 0	I/O driver: ON I/O: 0 if WD is expired, else output data
1	I/O driver: OFF	I/O driver: OFF if WD is expired or output event has not occurred since WD was last activated I/O: 0 if WD is expired, else output data

OE_CONF is ignored in bidirectional mode, I/O will be driven low during output events if OE_EXT is 0 or the watchdog is expired.

NOTE: I/O drivers are off until the EEPROM is loaded regardless of OE_CONF, OE_EXT, and watchdog.

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

5.1.8 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.

5.1.9 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.

5.1.10 EEPROM_LOADED

The EEPROM_LOADED signal indicates that the Digital I/O Interface is operational. Attach a pull-down resistor for proper function, since the PDI pin will not be driven until the EEPROM is loaded.

5.1.11 Timing specifications

Table 53: Digital I/O timing characteristics ET1100

Parameter	Min	Max	Comment
t_{DATA_setup}	7 ns		Input data valid before LATCH_IN
t_{DATA_hold}	3 ns		Input data valid after LATCH_IN
t_{LATCH_IN}	8 ns		LATCH_IN high time
t_{SOF}	35 ns	45 ns	SOF high time
$t_{SOF\_to\_DATA\_setup}$		1,2 µs	Input data valid after SOF, so that Inputs can be read in the same frame
$t_{SOF\_to\_DATA\_hold}$	1,6 µs		Input data invalid after SOF
$t_{input\_event\_delay}$	440 ns		Time between consecutive input events
$t_{OUTVALID}$	75 ns	85 ns	OUTVALID high time
$t_{DATA\_to\_OUTVALID}$	65 ns		Output data valid before OUTVALID
t_{WD_TRIG}	35 ns	45 ns	WD_TRIG high time
$t_{DATA\_to\_WD\_TRIG}$		35 ns	Output data valid after WD_TRIG
$t_{OE\_EXT\_to\_DATA\_invalid}$	0 ns	15 ns	Outputs zero or Outputs high impedance after OE_EXT set to low
$t_{output\_event\_delay}$	320 ns		Time between consecutive output events
$t_{BIDIR\_DATA\_valid}$	65 ns		Bidirectional mode: I/O valid before OUTVALID
$t_{BIDIR\_DATA\_invalid}$	65 ns		Bidirectional mode: I/O invalid after OUTVALID
$t_{BIDIR\_event\_delay}$	440 ns		Bidirectional mode: time between consecutive input and output events

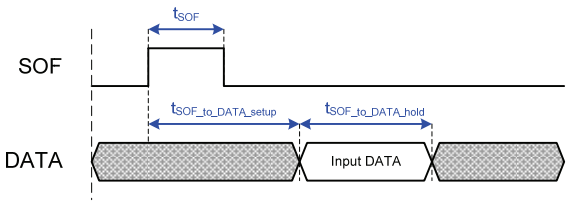


Figure 11: Digital Input: Input data sampled at SOF, I/O can be read in the same frame

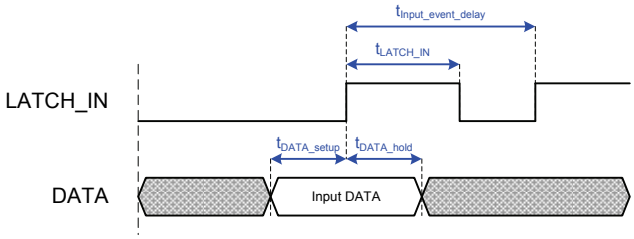


Figure 12: Digital Input: Input data sampled with LATCH_IN

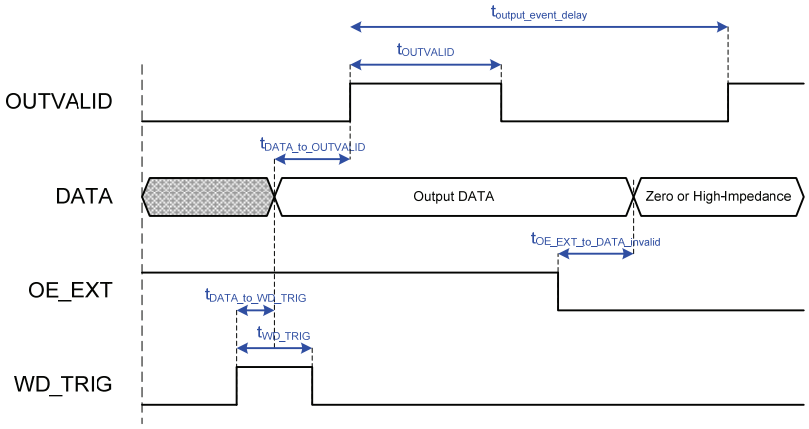


Figure 13: Digital Output timing

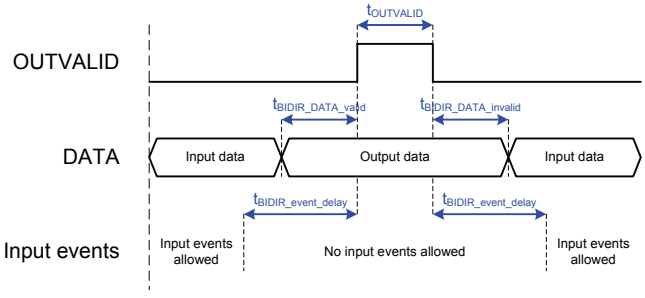


Figure 14: Bidirectional Mode timing

5.2 SPI Slave Interface

5.2.1 Interface

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

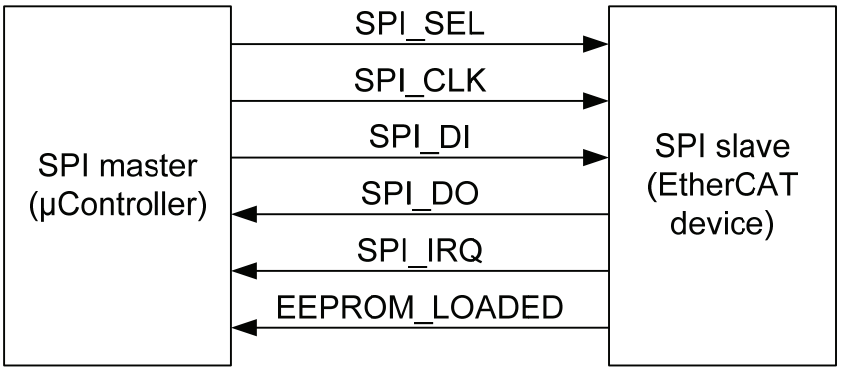


Figure 15: SPI master and slave interconnection

Table 54: 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
EEPROM_LOADED	OUT	(slave → master)	PDI is active, EEPROM is loaded	act. high

5.2.2 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.

5.2.3 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.

5.2.4 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 55: Address modes

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

5.2.5 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 56: 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

5.2.6 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 57: 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

5.2.7 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.

5.2.8 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).

5.2.8.1 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.

5.2.8.2 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.

5.2.9 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.

5.2.10 EEPROM_LOADED

The EEPROM_LOADED signal indicates that the SPI Interface is operational. Attach a pull-down resistor for proper function, since the PDI pin will not be driven until the EEPROM is loaded.

5.2.11 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 58.

Table 58: Write access for 2 and 4 Byte SPI Masters

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

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

5.2.12 Timing specifications

Table 59: SPI timing characteristics ET1100

Parameter	Min	Max	Comment
t_{CLK}	50 ns		SPI_CLK frequency (f_{CLK} ≤ 20 MHz)
$t_{SEL\_to\_CLK}$	6 ns		First SPI_CLK cycle after SPI_SEL asserted
$t_{CLK\_to\_SEL}$	a) 5 ns b) t_{CLK}/2+5 ns		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 Wait State Bytes are used.
$t_{SEL\_to\_DO\_valid}$		15 ns	Status/Interrupt Byte 0 bit 7 valid after SPI_SEL asserted
$t_{SEL\_to\_DO\_invalid}$	0 ns		Status/Interrupt Byte 0 bit 7 invalid after SPI_SEL deasserted
t_{STATUS_valid}	12 ns		Time until status of last access is valid. Can be ignored if status is not used.
t_{access_delay}	a) 15 ns b) 240 ns		Delay between SPI accesses a) typical b) If last access was shorter than 2 bytes, otherwise Interrupt Request Register value I0_[7:0] will not be valid.
t_{DI_setup}	9 ns		SPI_DI valid before SPI_CLK edge
t_{DI_hold}	3 ns		SPI_DI valid after SPI_CLK edge
$t_{CLK\_to\_DO\_valid}$		15 ns	SPI_DO valid after SPI_CLK edge
$t_{CLK\_to\_DO\_invalid}$	0 ns		SPI_DO invalid after SPI_CLK edge
$t_{EEPROM\ LOADED\ to\ access}$	0 ns		Time between EEPROM_LOADED and first access
t_{IRQ_delay}	160 ns		Internal delay between AL event and SPI_IRQ output to enable correct reading of the interrupt registers.

Table 60: Read/Write timing diagram symbols

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

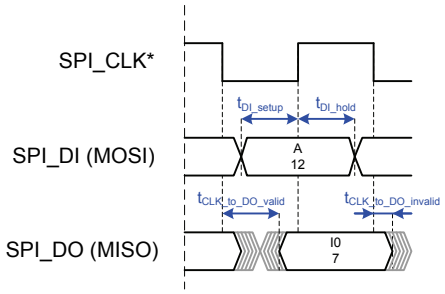


Figure 16: Basic SPI_DI/SPI_DO timing (*refer to timing diagram for relevant edges of SPI_CLK)

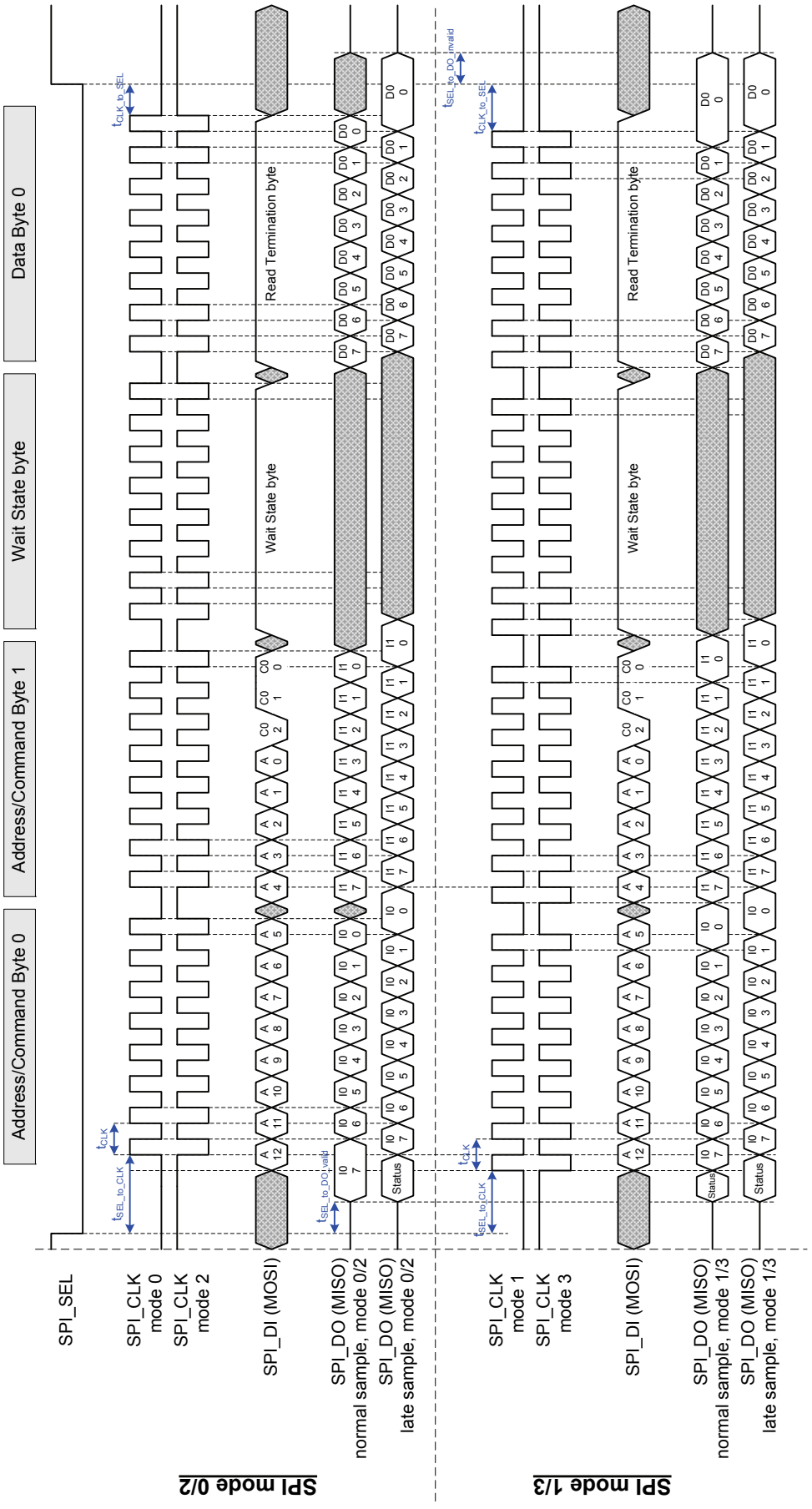


Figure 17: SPI read access (2 byte addressing, 1 byte read data) with Wait State byte

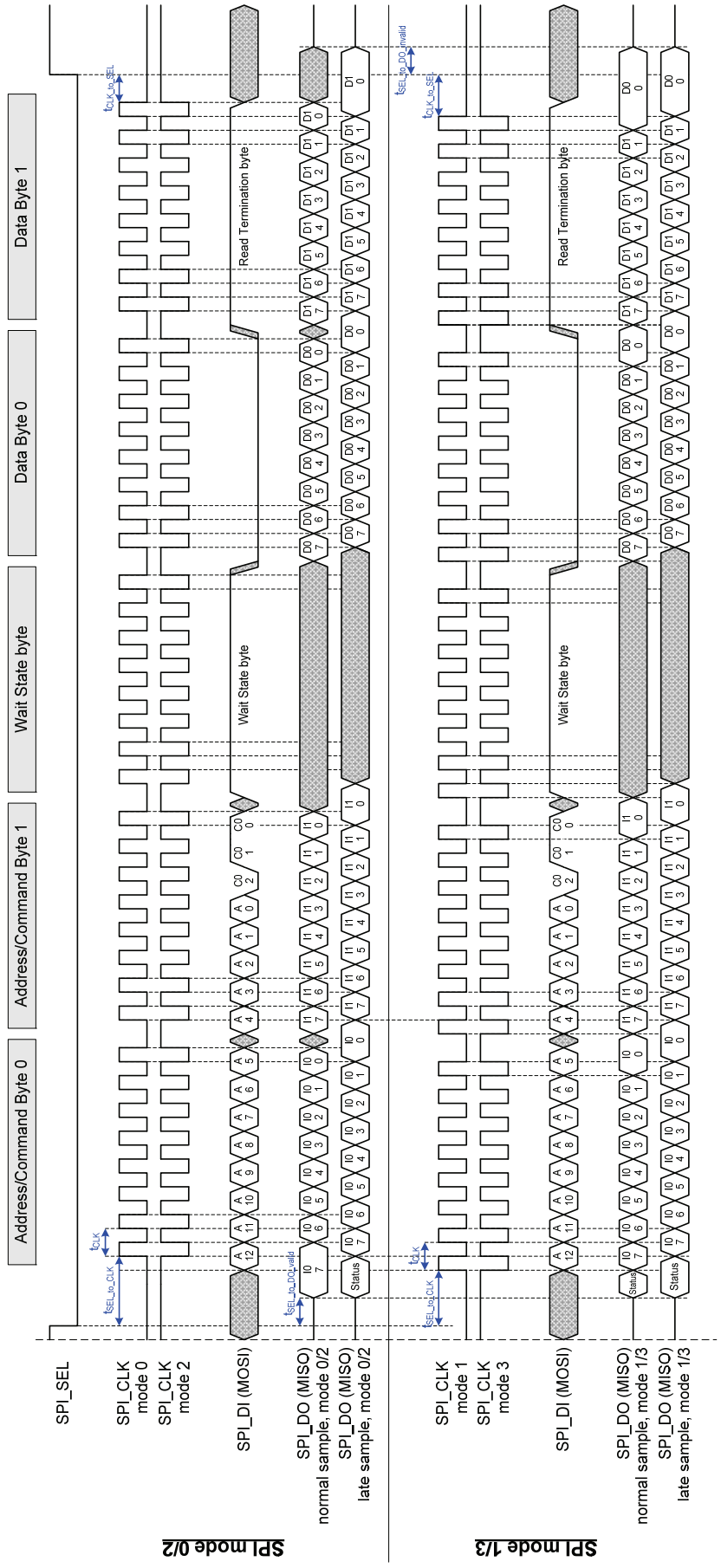


Figure 18: SPI read access (2 byte addressing, 2 byte read data) with Wait State byte

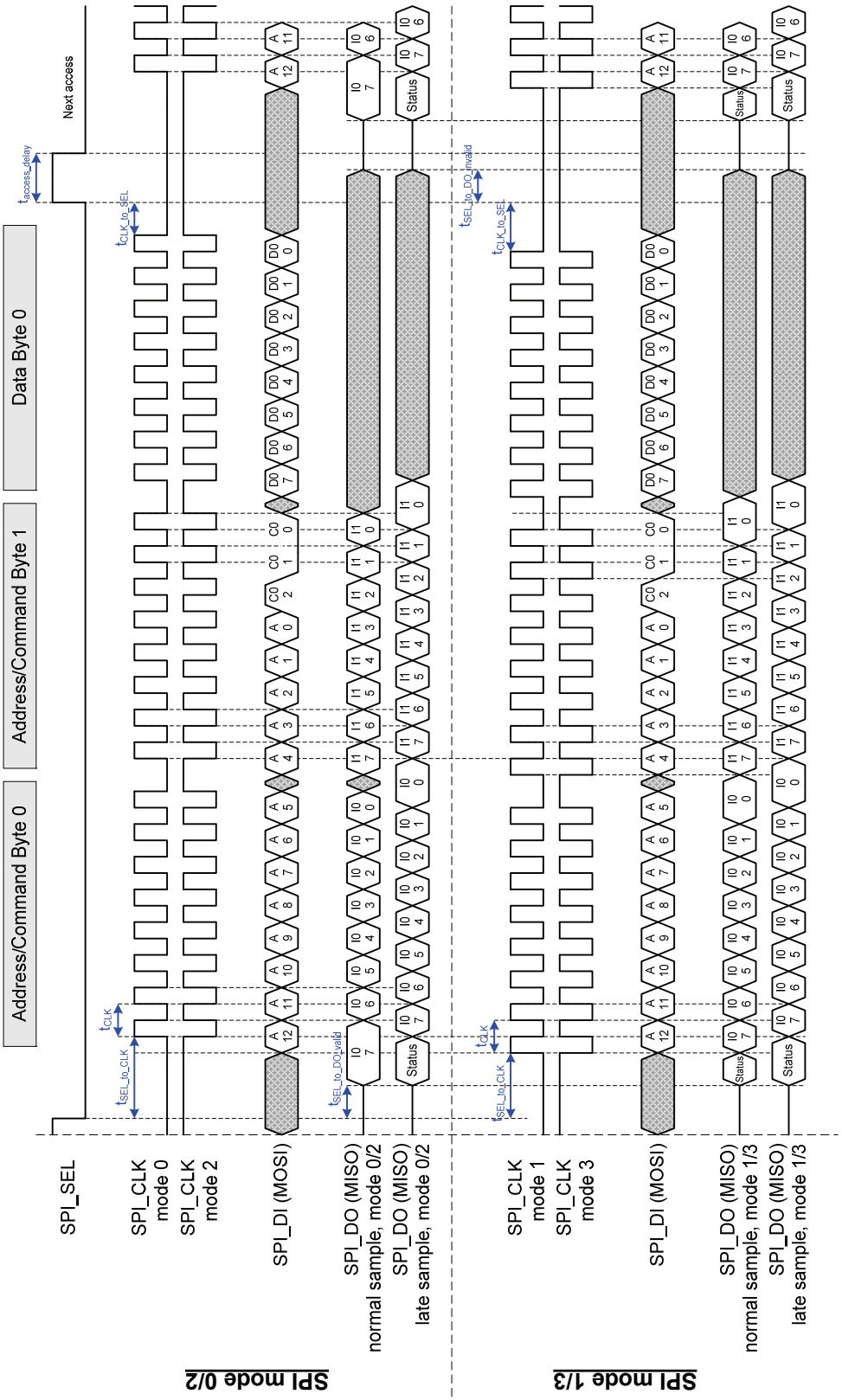


Figure 19: SPI write access (2 byte addressing, 1 byte write data)

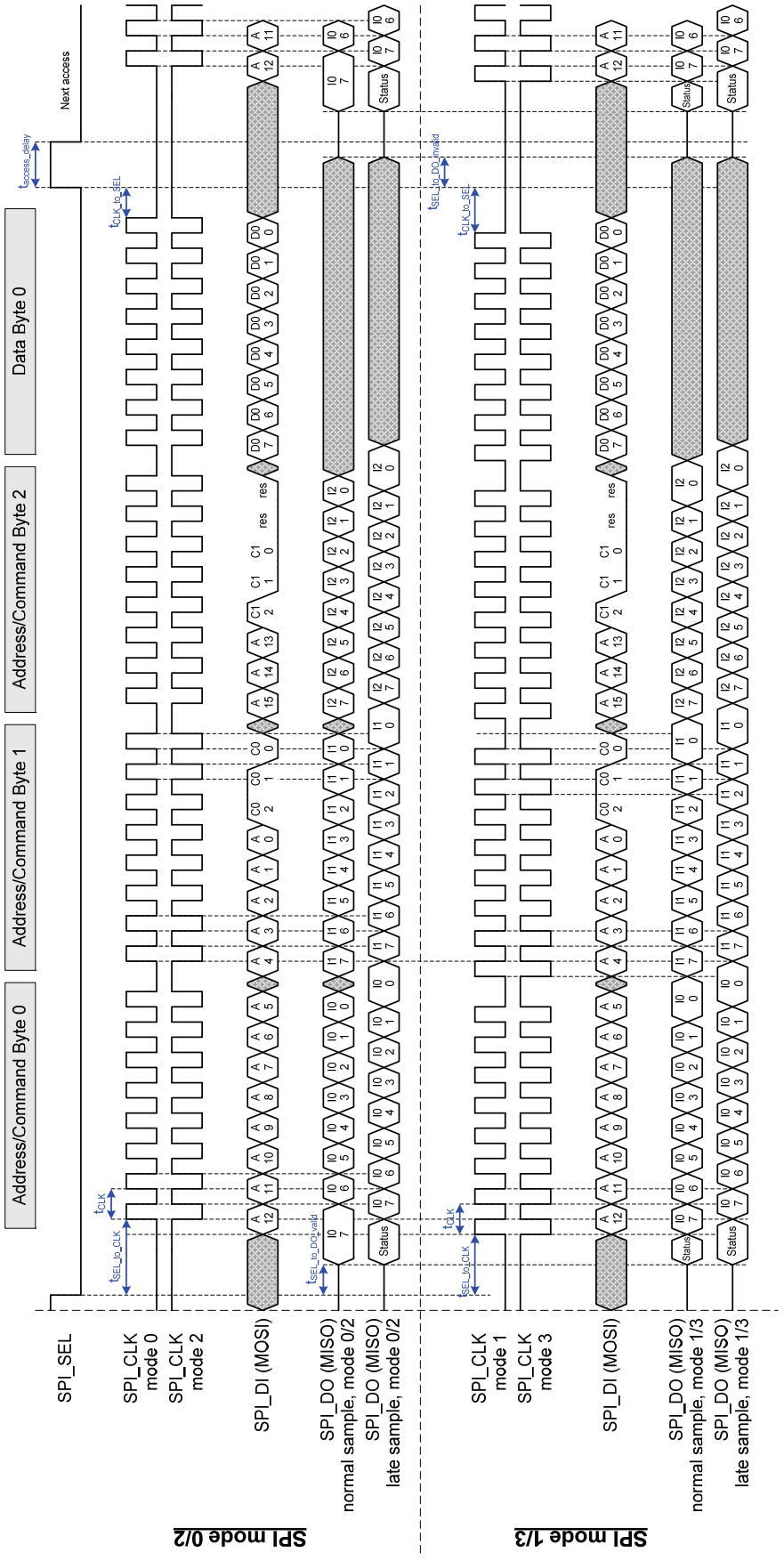


Figure 20: SPI write access (3 byte addressing, 1 byte write data)

5.3 Asynchronous 8/16 bit µController Interface

5.3.1 Interface

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

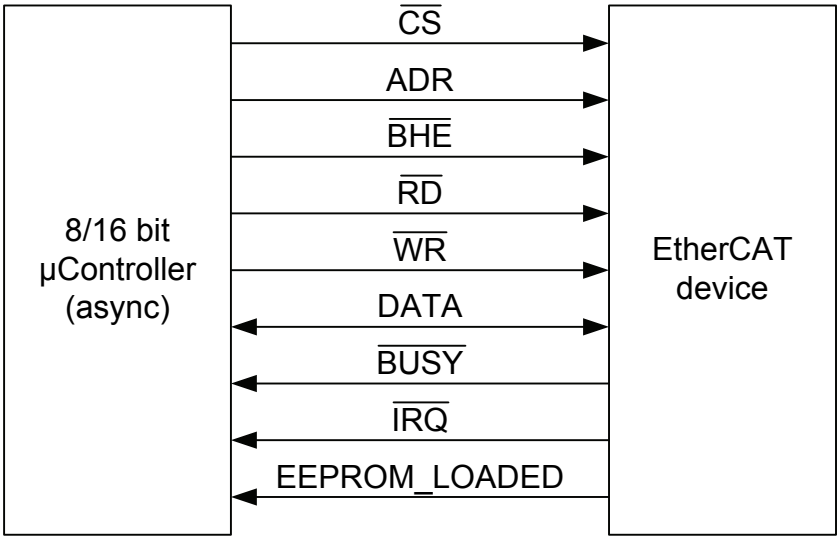


Figure 21: µController interconnection[3]

Table 61: µController signals

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

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

5.3.2 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.

[3] All signals are denoted with typical polarity configuration.

5.3.3 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 62: 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 63: 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

5.3.4 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.

Internally, the write access is performed after the rising edge of WR, this allows for fast write accesses. Nevertheless, an access following immediately will be delayed by the preceding write access (BUSY is active for a longer time).

5.3.5 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.

5.3.6 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.

5.3.7 EEPROM_LOADED

The EEPROM_LOADED signal indicates that the µController Interface is operational. Attach a pull-down resistor for proper function, since the PDI pin will not be driven until the EEPROM is loaded.

5.3.8 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.

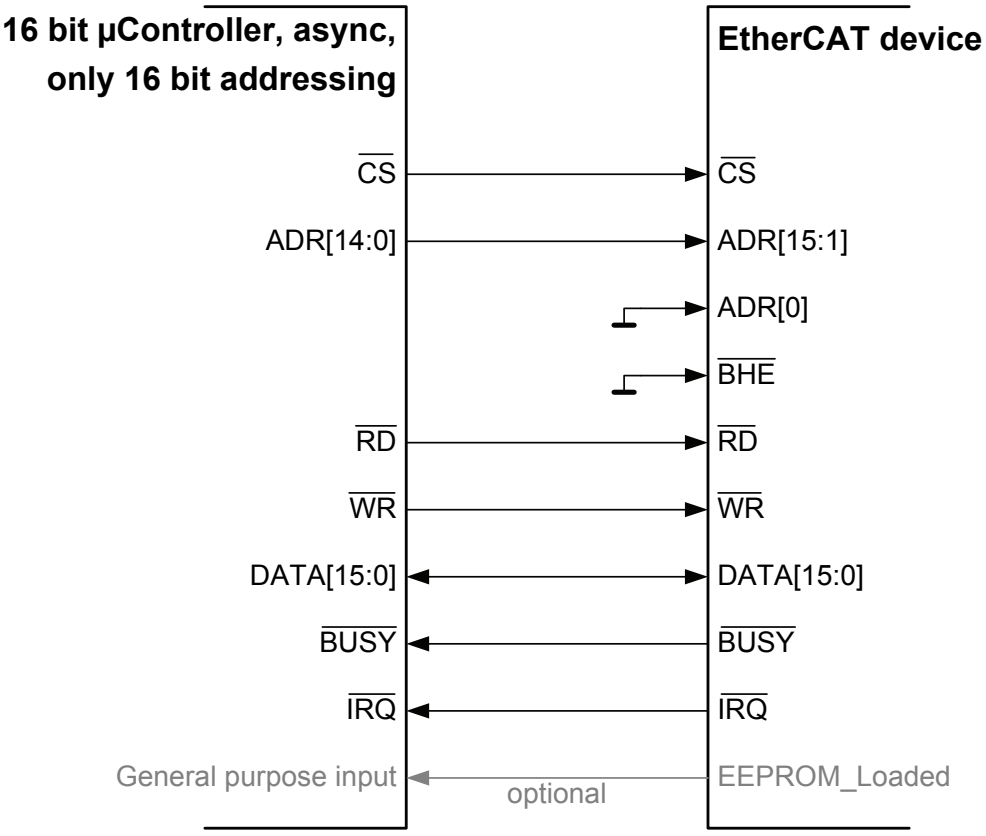


Figure 22: Connection with 16 bit µControllers without byte addressing

5.3.9 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.

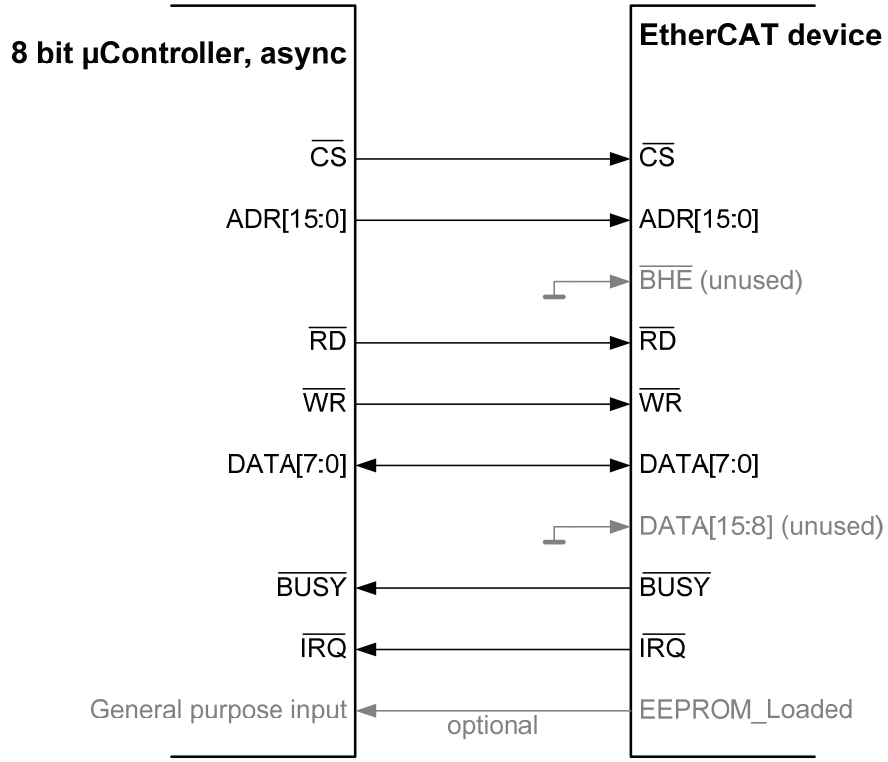


Figure 23: Connection with 8 bit µControllers (BHE and DATA[15:8] should not be left open)

5.3.10 Timing Specification

Table 64: µController timing characteristics ET1100

Parameter	Min	Max	Comment
$t_{CS\_to\_BUSY}$		15 ns	BUSY driven and valid after CS assertion
$t_{ADR\_BHE\_setup}$	-2 ns		ADR and BHE valid before RD assertion
$t_{RD\_to\_DATA\_driven}$	0 ns		DATA bus driven after RD assertion
$t_{RD\_to\_BUSY}$	0 ns	15 ns	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 with delayed read busy output.
		a) t_{read_int} + t_{prec_write} +t_{Coll} -$t_{WR\_to\_RD}$	a) with preceding write access and $t_{WR\_to\_RD} < t_{prec\_write} + t_{Coll}$
		b) t_{read_int}	b) without preceding write access or $t_{WR\_to\_RD} \geq t_{prec\_write} + t_{Coll}$
		c) 435 ns	c) 8 bit access, absolute worst case with preceding 8 bit write access ($t_{WR\ to\ RD}$=min, t_{prec_write} =max, t_{Coll}=max)
		d) 575 ns	d) 16 bit access, absolute worst case with preceding 16 bit write access ($t_{WR\ to\ RD}$=min, t_{prec_write} =max, t_{Coll}=0)
t_{read_int}		a) 235 ns b) 315 ns	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) 5 ns b) -15 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		DATA invalid after ADR or BHE change
$t_{CS\_RD\_to\_DATA\_release}$	0 ns		DATA bus released after CS deassertion or RD deassertion
$t_{CS\_to\_BUSY\_release}$	0 ns	15 ns	BUSY released after CS deassertion
t_{CS_delay}	0 ns		Delay between CS deassertion an assertion
t_{RD_delay}	10 ns		Delay between RD deassertion and assertion
$t_{ADR\_BHE\_DATA\_setup}$	10 ns		ADR, BHE and Write DATA valid before WR deassertion
$t_{ADR\_BHE\_DATA\_hold}$	3 ns		ADR, BHE and Write DATA valid after WR deassertion
t_{WR_active}	10 ns		WR assertion time
$t_{BUSY\_to\_WR\_CS}$	0 ns		WR or CS deassertion after BUSY deassertion
$t_{WR\_to\_BUSY}$		15 ns	BUSY assertion after WR deassertion
t_{write}	0 ns		External write time (WR assertion to BUSY deassertion)

Parameter	Min	Max	Comment
		a) t_{write_int} -t_{WR_delay}	a) with preceding write access and $t_{WR\_delay} < t_{write\_int}$
		b) 0 ns	b) without preceding write access or $t_{WR\_delay} \geq t_{write\_int}$
		c) 200 ns	c) 8 bit access, absolute worst case with preceding 8 bit write access ($t_{WR\ delay}$= min, t_{WR_int}=max)
		d) 280 ns	d) 16 bit access, absolute worst case with preceding 16 bit write access ($t_{WR\ delay}$=min, t_{WR_int}=max)
t_{write_int}		a) 200 ns b) 280 ns	Internal write time a) 8 bit access b) 16 bit access
t_{WR_delay}	10 ns		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
$t_{CS\_WR\_overlap}$	5 ns		Time both CS and WR have to be deasserted simultaneously (only if CS is deasserted at all)
$t_{CS\_RD\_overlap}$	5 ns		Time both CS and RD have to be deasserted simultaneously (only if CS is deasserted at all)
$t_{EEPROM\_LOADED\_to\_access}$	0 ns		Time between EEPROM_LOADED and first access
$t_{EEPROM\_LOADED\_to\_IRQ}$		0 ns	IRQ valid after EEPROM_LOADED

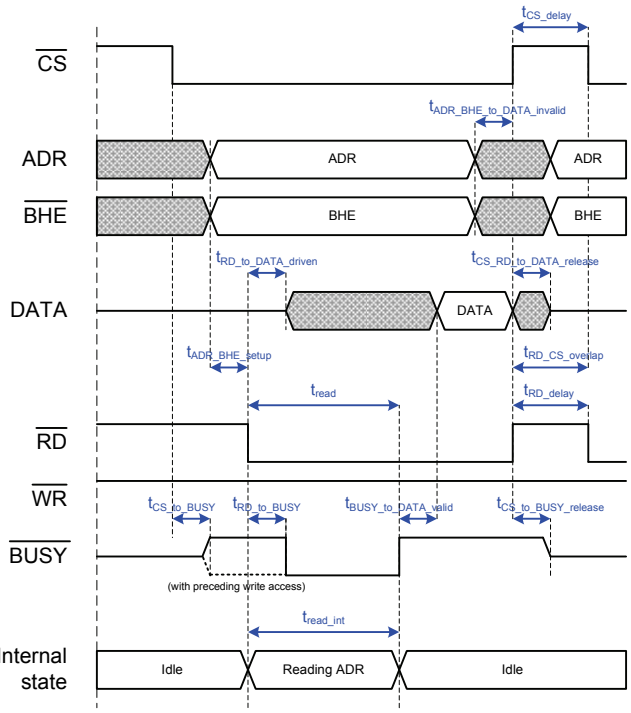


Figure 24: Read access (without preceding write access)

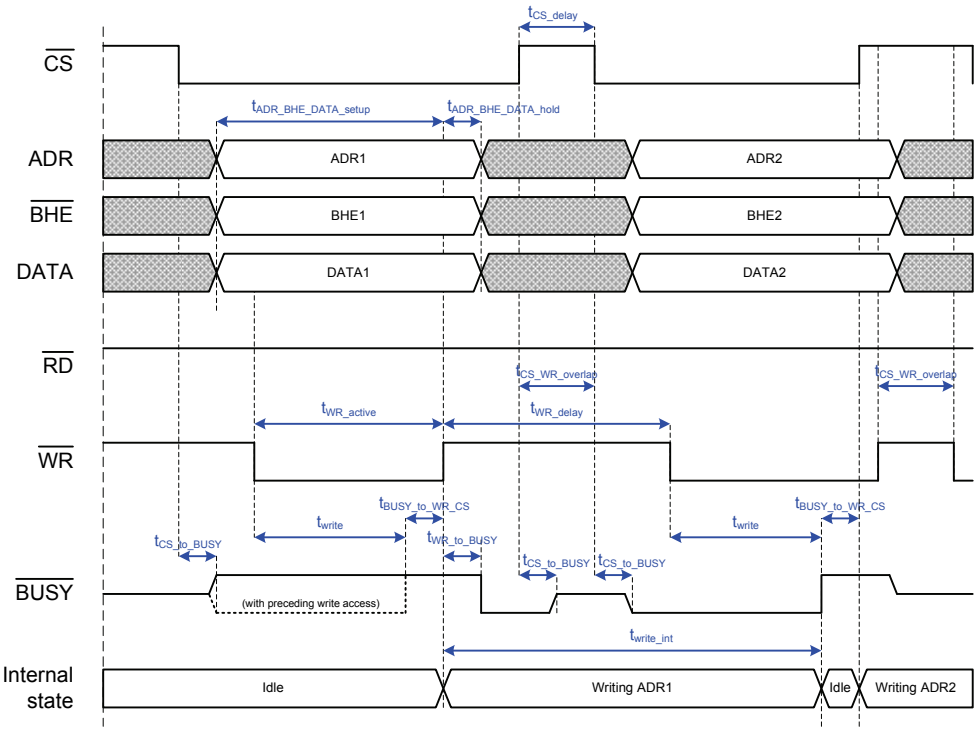


Figure 25: Write access (write after rising edge nWR, without preceding write access)

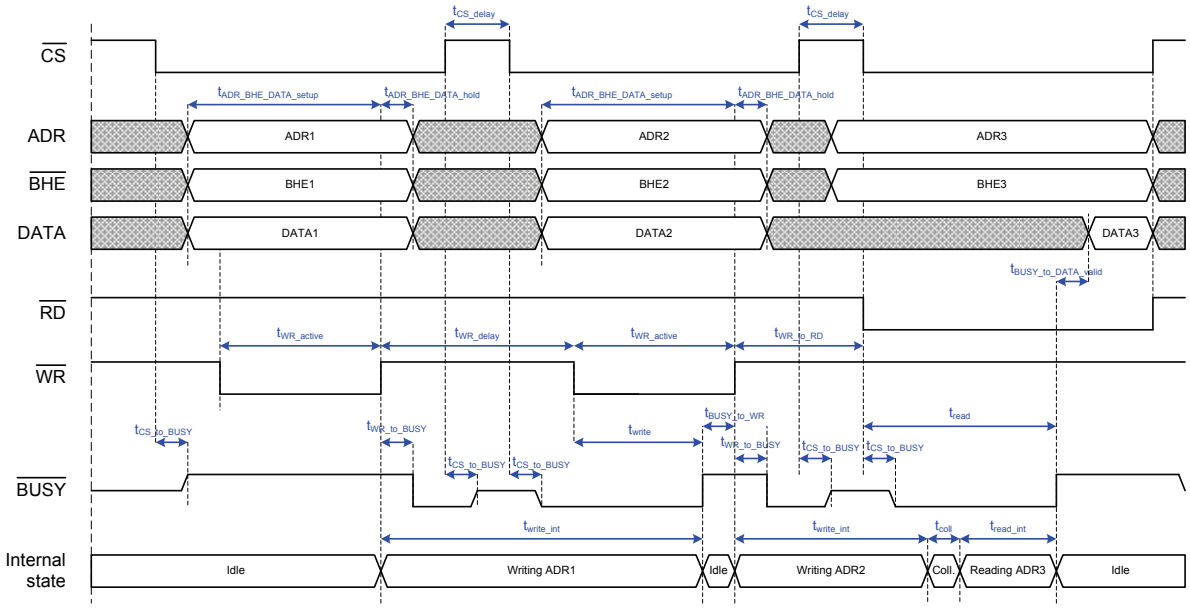


Figure 26: Sequence of two write accesses and a read access

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.

5.4 Synchronous 8/16 bit µController Interface

5.4.1 Interface

The synchronous 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 synchronous µController interface of EtherCAT devices are:

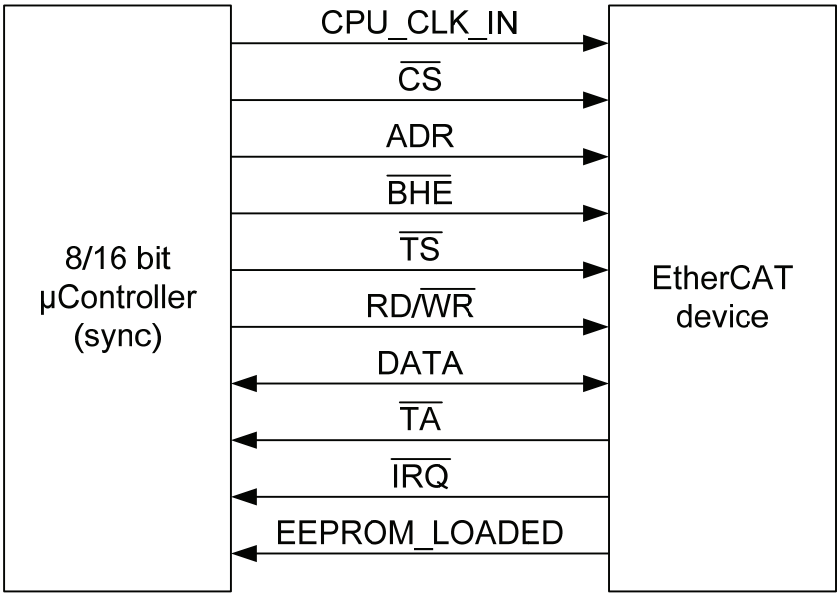


Figure 27: µController interconnection[4]

Table 65: µController signals

Signal sync I/F	Signal async I/F	Direction		Description	Signal polarity
CPU_CLK_IN	N/A	IN	(µC → ESC)	µController interface clock	
CS	CS	IN	(µC → ESC)	Chip select	Typical: act. low
ADR[15:0]	ADR[15:0]	IN	(µC → ESC)	Address bus	act. high
BHE	BHE	IN	(µC → ESC)	Byte High Enable	Typical: act. low
TS	RD	IN	(µC → ESC)	Transfer Start	Typical: act. low
RD/nWR	WR	IN	(µC → ESC)	Read/Write access	
DATA[15:0]	DATA[15:0]	BD	(µC ↔ ESC)	Data bus for 16 Bit µController interface	act. high
DATA[7:0]	DATA[7:0]	BD	(µC ↔ ESC)	Data bus for 8 Bit µController interface	act. high
TA	BUSY	OUT	(ESC → µC)	Transfer Acknowledge	Typical: act. low
IRQ	IRQ	OUT	(ESC → µC)	Interrupt	Typical: act. low
EEPROM_ LOADED	EEPROM_ LOADED	OUT	(ESC → µC)	PDI is active, EEPROM is loaded	act. high

5.4.2 Configuration

The 16 bit synchronous µController interface is selected with PDI type 0x0A in the PDI control register 0x0140, the 8 bit synchronous µController interface has PDI type 0x0B. It supports different configurations, which are located registers 0x0150 – 0x0153.

[4] All signals are denoted with typical polarity configuration.

5.4.3 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. The least significant address bit A[0] 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.

Table 66: 8 bit high/low byte and 16 bit access distinction

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

If Byte High Enable (BHE) is used, the Byte access mode configuration bit has to be set to zero (BHE or Byte Select mode).

EtherCAT devices use Little Endian byte ordering, even with the synchronous µController interface. The conversion between Little Endian and Big Endian, depending on the register size of 8, 16, 32, or 64 bit, has to be done in software.

NOTE: A µController with 32 Bit interface is used as an example connected to the synchronous µController interface. It is also possible to use 8 or 16 Bit µControllers.

NOTE: Please compare the bit ordering ([0:31] instead of [31:0]) of your µController with that used in this document, because it might be different. The MSB/LSB notation used below will help you.

Table 67: Corresponding Bytes and Bits

Address	0	1	2	3
µController	[31:24]	[23:16]	[15:8]	[7:0]
	[MSBit:LSBit]	[MSBit:LSBit]	[MSBit:LSBit]	[MSBit:LSBit]
	MSByte	:		LSByte
ESC 8 Bit access	[7:0]			
	[MSBit:LSBit]			
ESC 16 Bit access	[7:0]	[15:8]		
	[MSBit:LSBit]	[MSBit:LSBit]		
	LSByte :	MSByte		

Table 68: Byte ordering

Addr.	ESC (Little Endian)				sync. µController (Big Endian)			
	8 bit reg.	16 bit reg.	32 bit reg.	64 bit reg.	8 bit reg.	16 bit reg.	32 bit reg.	64 bit reg.
0	Byte 0	LSB	LSB	LSB	Byte 0	MSB	MSB	MSB
1	Byte 1	MSB	|	|	Byte 1	LSB	|	|
2	Byte 2	LSB	|	|	Byte 2	MSB	|	|
3	Byte 3	MSB	MSB	|	Byte 3	LSB	LSB	|
4	Byte 4	LSB	LSB	|	Byte 4	MSB	MSB	|
5	Byte 5	MSB	|	|	Byte 5	LSB	|	|
6	Byte 6	LSB	|	|	Byte 6	MSB	|	|
7	Byte 7	MSB	MSB	MSB	Byte 7	LSB	LSB	LSB

5.4.4 µController connection using Byte Select signals (BSn)

In case the µController does not provide Byte High Enable, and Byte Select signals (BS2, and BS3 for 32 bit µController) are available, they can be used to distinguish between 8 and 16 bit accesses. The signal BS3 (active low) is equivalent to ADR[0], and BS2 (active low) is equivalent to BHE (active low).

For Byte Select mode the Byte access mode configuration bit has to be set to zero (BHE or Byte Select mode).

Table 69: Byte Select vs. A[0] and BHE

µController			EtherCAT device		Access
ADR[0]	nBS3	nBS2	ADR[0]	BHE (act. low)	
0	0	0	0	0	16 bit access (low and high byte)
0	0	1	0	1	8 bit access (low byte, even address)
1	1	0	1	0	8 bit access (high byte, odd address)
1	1	1	1	1	invalid access

The following figure shows how a 32 bit µController can be connected with the EtherCAT synchronous 16 bit µController interface using Byte Select signals:

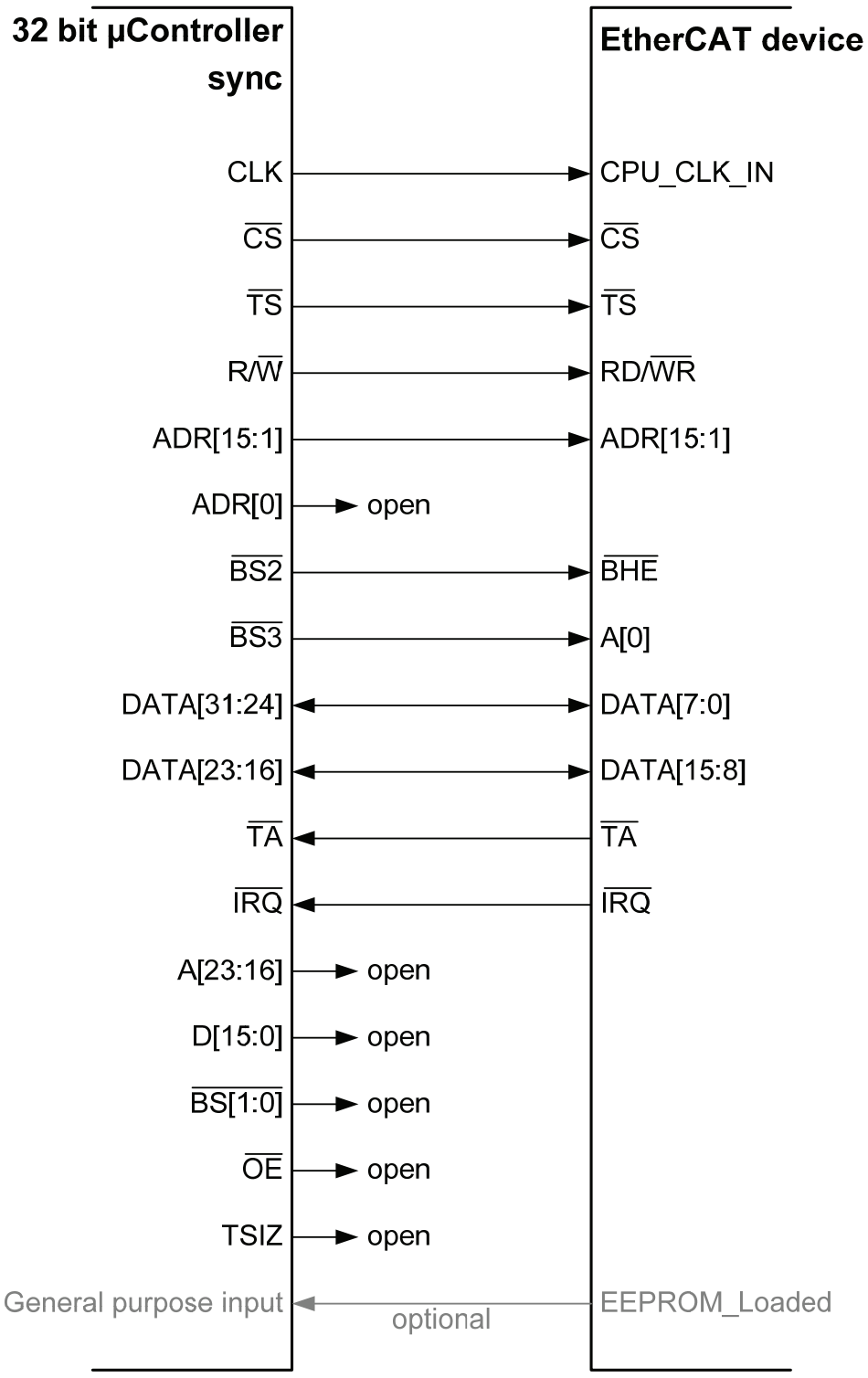


Figure 28: Synchronous 32 bit µController connection using Byte Select

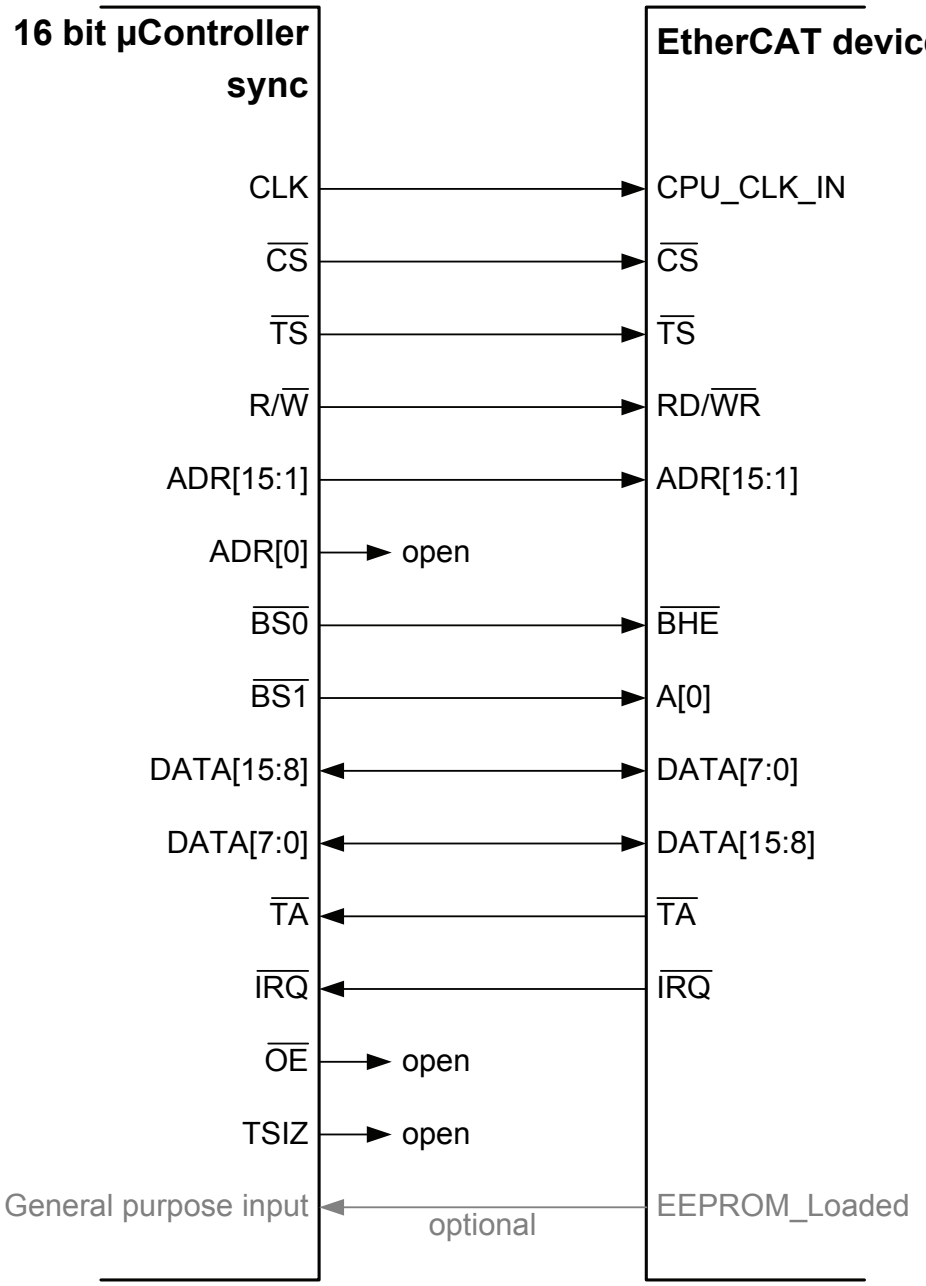


Figure 29: Synchronous 16 bit µController connection using Byte Select

5.4.5 µController connection using Transfer Size signals (SIZ)

In case the µController does not provide Byte High Enable, and Transfer Size signals (SIZ or TSIZ) are available, they can be used to distinguish between 8 and 16 bit accesses together with ADR[0]. An exclusive-or combination of ADR[0] and SIZ[0] is equivalent to BHE. This combination can be configured with Byte access mode set to one (Transfer Size mode).

Table 70: Byte Select vs. ADR[0] and BHE

µController			EtherCAT device		Access
ADR[0]	**SIZ[1:0]**	**ADR[0] xor SIZ[0]**	**ADR[0]**	**BHE (act. low)**	
0	10	0	0	0	16 bit access (low and high byte)
0	01	1	0	1	8 bit access (low byte, even address)
1	01	0	1	0	8 bit access (high byte, odd address)
0	00	0	0	0	32 bit access (splitted in two 16 bit accesses)

The following figure shows how a 32 bit µController can be connected with the EtherCAT synchronous 16 bit µController interface using SIZ signals:

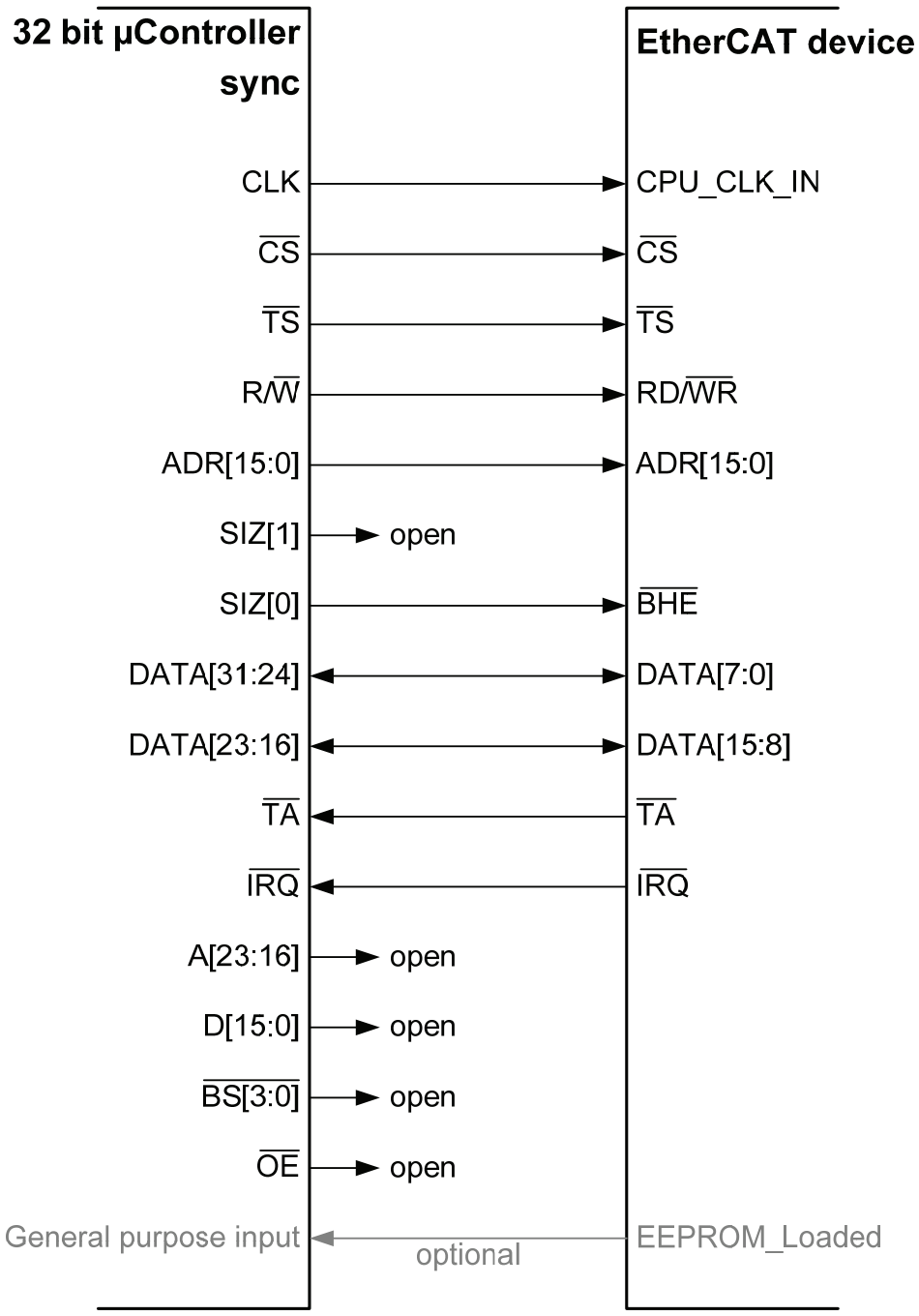


Figure 30: Synchronous 32 bit µController connection using Transfer Size

5.4.6 Write access

A write access starts with a Transfer Start (TS). Chip Select can be either together with TS or one clock cycle later (does not need to be configured). The CPU_CLK_IN edge at which CS is sampled can be configured. ADR, BHE and R/nW are valid together with TS. It is configurable if write DATA is also valid with CS or one cycle later. Once the EtherCAT device has finished the access, Transfer Acknowledge is asserted for one clock cycle. It may either be generated with the rising or falling edge of CPU_CLK_IN.

5.4.7 Read access

A read access starts with a Transfer Start (TS). Chip Select can be either together with TS or one clock cycle later (does not need to be configured). The CPU_CLK_IN edge at which CS is sampled can be configured. ADR, BHE and R/nW are valid together with TS. Once the EtherCAT device has finished the access, Transfer Acknowledge is asserted for one clock cycle together with the read DATA. TA may either be generated with the rising or falling edge of CPU_CLK_IN.

Some µControllers expect a read access always to be a 16 bit read access, regardless of the Byte Select signals. For this reason, it is configurable that the Byte Select signals are ignored and a read access is always a 16 bit access.

5.4.8 µController access errors

One reason for µController access errors is detected by the synchronous 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.

Such a wrong µController access will have these consequences:

- The PDI error counter 0x030D will be incremented.
- No access will be performed internally.

5.4.9 EEPROM_LOADED

The EEPROM_LOADED signal indicates that the µController Interface is operational. Attach a pull-down resistor for proper function, since the PDI pin will not be driven until the EEPROM is loaded.

5.4.10 Timing Specification

Table 71: µController timing characteristics ET1100

Parameter	Min	Max	Comment
t_{CLK}	25 ns		CPU_CLK_IN period ($f_{CLK} \leq 40$ MHz)
t_{IN_setup}	10 ns		Input signals valid before CPU_CLK_IN edge (TS, CS, ADR, BHE, R/nW, DATA)
t_{IN_hold}	3 ns		Input signals valid after CPU_CLK_IN edge (TS, CS, ADR, BHE, R/nW, DATA)
$t_{CLK\_to\_OUT\_valid}$		15 ns	Output signals valid after CPU_CLK_IN edge (TA, IRQ, DATA)
$t_{CLK\_to\_OUT\_invald}$	0 ns		Output signals invalid after CPU_CLK_IN edge (TA, IRQ, DATA)
t_{read}		a) t_{read_int} + t_{prec_write} +t_{Coll} b) t_{read_int} +t_{read_sync}	External read time (TS to TA) a) with preceeding write access and $t_{WR\_to\_RD} + t_{read\_sync} < t_{prec\_write} + t_{Coll}$ b) without preceeding write access or $t_{WR\_to\_RD} + t_{read\_sync} > t_{prec\_write} + t_{Coll}$
t_{read_sync}		$2.5*t_{CLK}$ a) $+0.5*t_{CLK}$ b) $+0.5*t_{CLK}$ c) $+t_{CLK}$	Extra read synchronization delay a) extra delay if 0x0152.11=1 b) extra delay if 0x0152.10=1 b) extra delay if CS asserted one CPU_CLK_IN cycle after TS
t_{read_int}		a) 235 ns b) 315 ns	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_{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_{write}		a) t_{write_int} b) t_{write_sync}	External write time (TS to TA) a) with preceeding write access and $t_{WR\_delay}+t_{write\_sync} < t_{write\_int}$ b) without preceeding write access or $t_{WR\_delay}+t_{write\_sync} \geq t_{write\_int}$
t_{write_sync}		$2.5*t_{CLK}$ a) $+t_{CLK}$ b) $+0.5*t_{CLK}$ c) $+0.5*t_{CLK}$ d) $+t_{CLK}$	Extra write synchronization delay a) extra delay if 0x0152.8=0 b) extra delay if 0x0152.10=1 c) extra delay if 0x0152.11=1 d) extra delay if CS asserted one CPU_CLK_IN cycle after TS
t_{write_int}		a) 200 ns b) 280 ns	Internal write time a) 8 bit access b) 16 bit access
$t_{write+read}$		$t_{write}+t_{read}$	Internal write/read time for a read access following a write access
$t_{EEPROM\_LOADED\_to\_access}$	0 ns		Time between EEPROM_LOADED and first access

Parameter	Min	Max	Comment
$t_{EEPROM\_LOADED\_to\_IRQ}$		0 ns	IRQ valid after EEPROM_LOADED

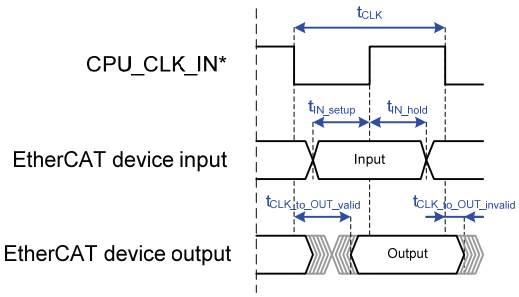


Figure 31: Basic synchronous µController interface timing (*refer to timing diagram for relevant CPU_CLK_IN edges)

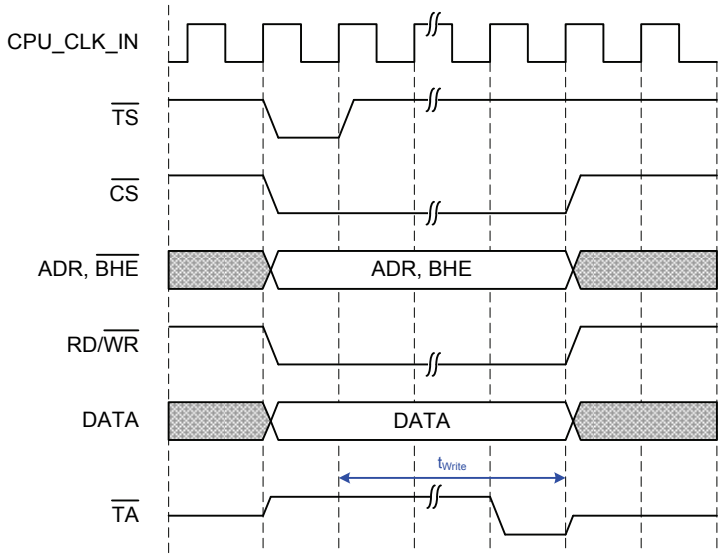


Figure 32: Write access (CS together with TS, Write DATA together with CS, CS and TA on rising edge)

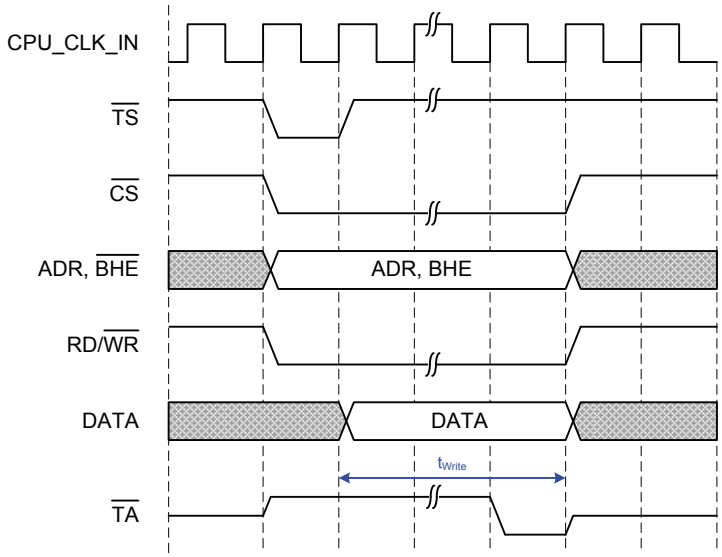


Figure 33: Write access (CS together with TS, Write DATA after CS, CS and TA on rising edge)

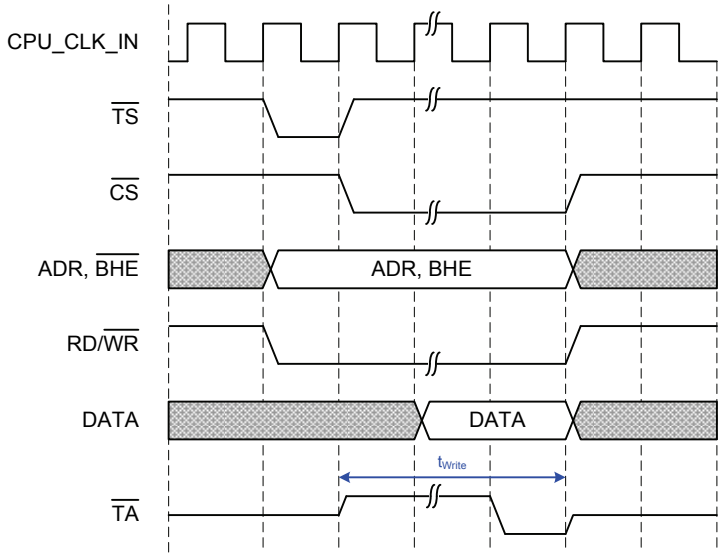


Figure 34: Write access (CS after TS, Write DATA after CS, CS and TA on rising edge)

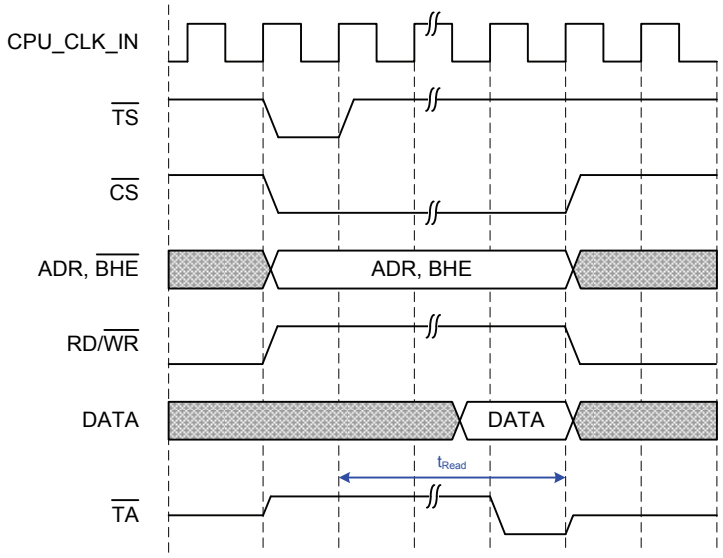


Figure 35: Read access (CS together wit TS, CS and TA on rising edge)

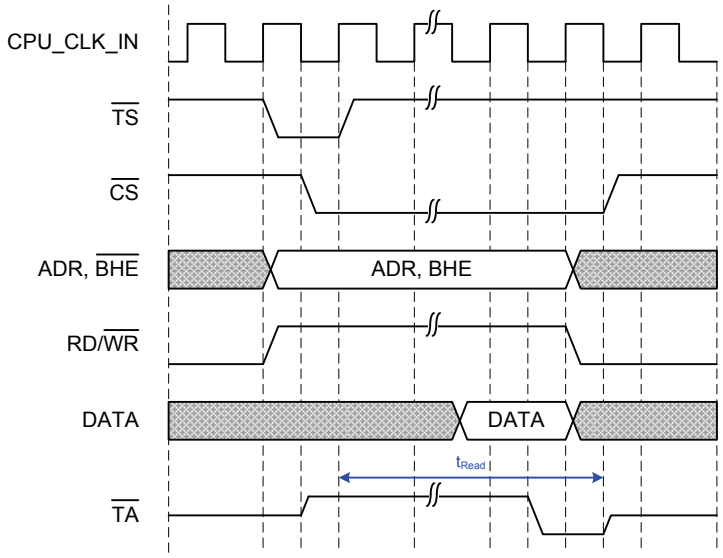


Figure 36: Read access (CS half a clock period after TS, CS and TA on falling edge)

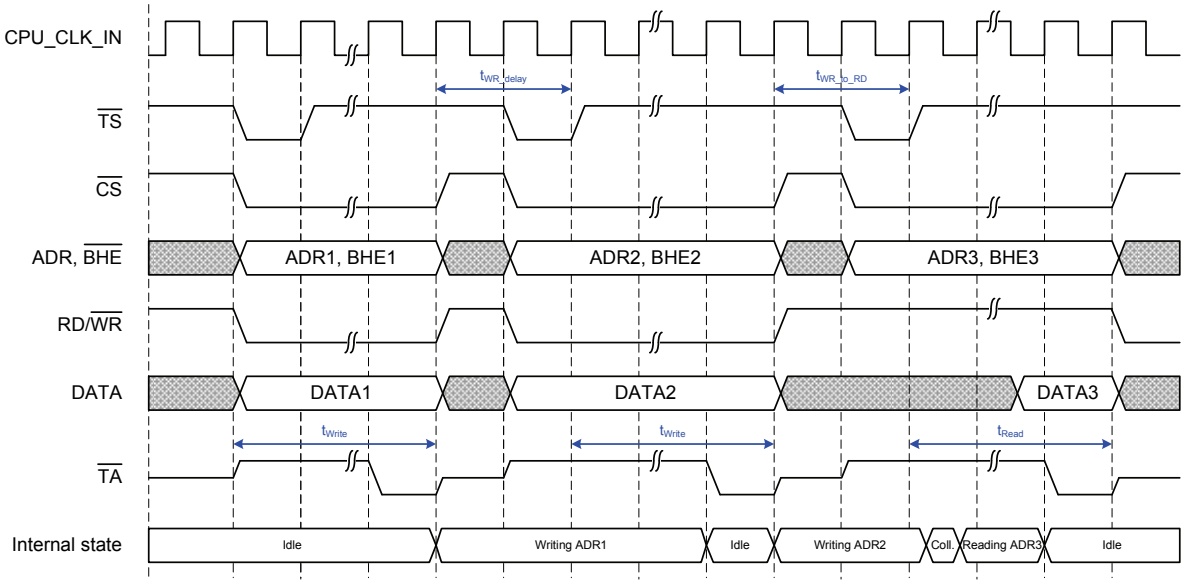


Figure 37: Sequence of two write accesses and a read access

Note: The first write access to ADR1 is performed after the first TA. After that, the ESC is internally busy writing to ADR1. After CS is deasserted, TA 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. After the second TA, the ESC is busy writing to ADR2.

The third access in this example is a read access. The ESC is still busy writing to ADR2 while the read access begins. In this case, the write access to ADR2 is finished first, and afterwards, the read access to ADR3 is performed. The ESC signals TA after both write and read access have finished.

6 Distributed Clocks SYNC/LATCH Signals

For details about the Distributed Clocks refer to Section I.

6.1 Signals

The Distributed Clocks unit of the ET1100 has the following external signals:

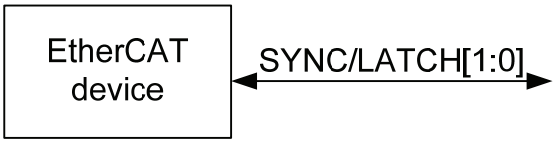


Figure 38: Distributed Clocks signals

Table 72: Distributed Clocks signals

Signal	Direction	Description
SYNC/LATCH[1:0]	OUT/IN	SyncSignal (OUT) or LatchSignal (IN), direction bitwise configurable via register 0x0151 / EEPROM.

NOTE: SYNC/LATCH signals are not driven (high impedance) until the ESI EEPROM is loaded.

6.2 Timing specifications

Table 73: DC SYNC/LATCH timing characteristics ET1100

Parameter	Min	Max	Comment
t_{DC_LATCH}	15 ns		Time between Latch0/1 events
$t_{DC\_SYNC\_Jitter}$		15 ns	SYNC0/1 output jitter

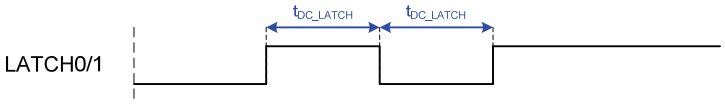


Figure 39: LatchSignal timing

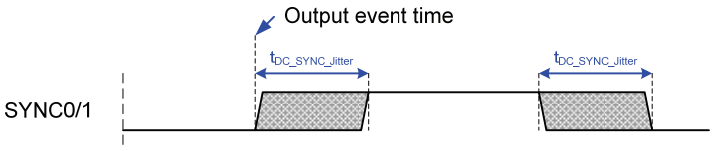


Figure 40: SyncSignal timing

7 ESI EEPROM Interface (I²C)

For details about the ESC ESI EEPROM Interface refer to Section I. The ESI EEPROM Interface is intended to be a point-to-point interface between ET1100 and I²C EEPROM. If other I²C masters are required to access the I²C bus, the ET1100 must be held in reset state (e.g. for in-circuit-programming of the EEPROM), otherwise access collisions will be detected by the ET1100.

7.1 Signals

The EEPROM interface of the ET1100 has the following signals:

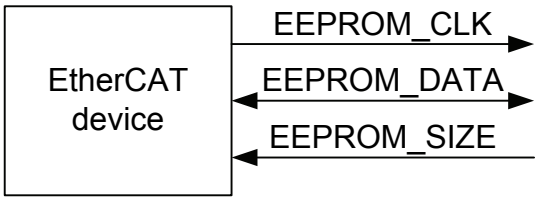


Figure 41: I²C EEPROM signals

Table 74: I²C EEPROM signals

Signal	Direction	Description
EEPROM_CLK	OUT	I²C clock
EEPROM_DATA	BIDIR	I²C data
EEPROM_SIZE	IN	EEPROM size configuration

The pull-up resistors for EEPROM_CLK and EEPROM_DATA are integrated into the ET1100. EEPROM_CLK must not be held low externally, because the ET1100 will detect this as an error.

7.2 Timing specifications

Table 75: ESI EEPROM timing characteristics

Parameter	Typical		Comment
	1 kBit-16 kBit	32 kBit-4 Mbit	
t_{Clk}	~ 6.72 µs		EEPROM clock period (f_{Clk} ≈ 150 kHz)
t_{Write}	~ 250 us	~ 310 µs	Write access time (without errors)
t_{Read}	a) ~ 680 µs b) ~ 1.16 ms	a) ~ 740 µs b) ~ 1.22 ms	Read access time (without errors): a) 4 words b) configuration (8 Words)
t_{Delay}	~ 168 ms		Time until configuration loading begins after Reset is gone

8 Example Schematics

8.1 Clock source

The layout of the clock source has the biggest influence on EMC/EMI of a system design.

Although a clock frequency of 25 MHz requires not extensive design efforts, the following rules shall help to improve system performance:

- Keep clock source and ESC as close as possible close together.
- Ground Layer should be seamless in this area.
- Power supply should be of low impedance for clock source and ESC clock supply.
- Capacitors shall be used as recommended by the clock source component.
- Capacities between clock source and ESC clock supply should be in the same size (values depend upon geometrical form of board).

The initial accuracy of the ET1100 clock source has to be 25ppm or better.

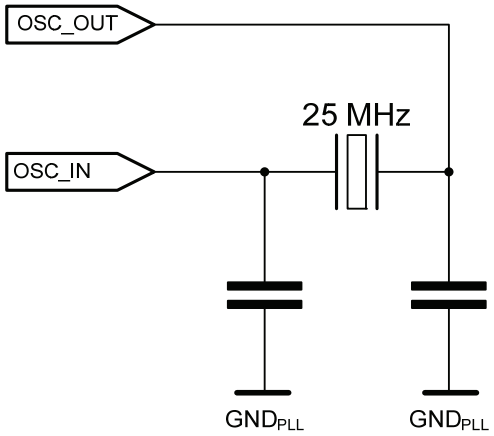


Figure 42: Quartz crystal connection

NOTE: The value of the load capacitors depends on the load capacitance of the crystal, the pin capacitance C_{OSC} of the ESC pins and the board design (typical 12pF each if C_L = 10pF).

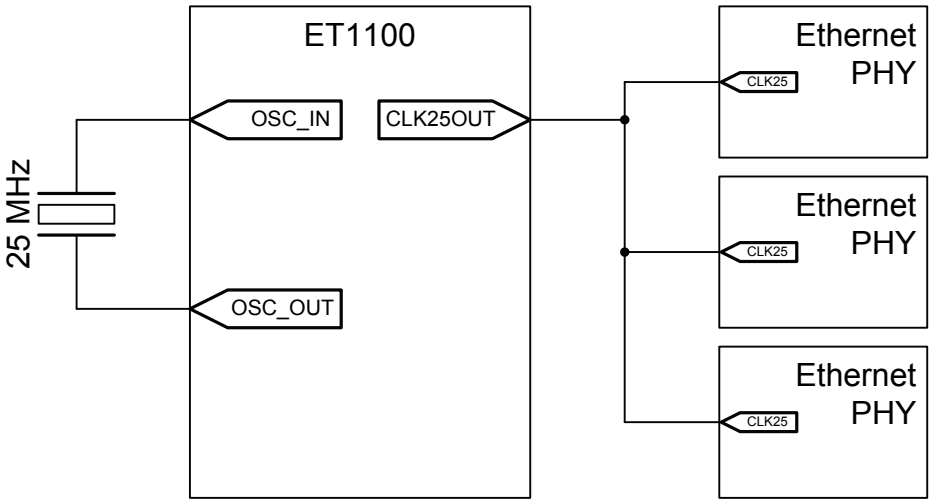


Figure 43: Quartz crystal Clock source for ET1100 and Ethernet PHYs

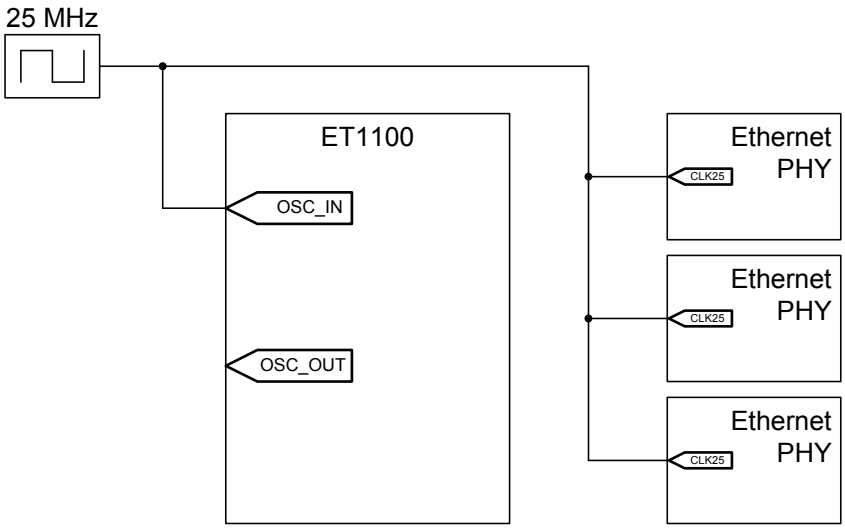


Figure 44: Oscillator clock source for ET1100 and Ethernet PHYs

8.2 Power supply

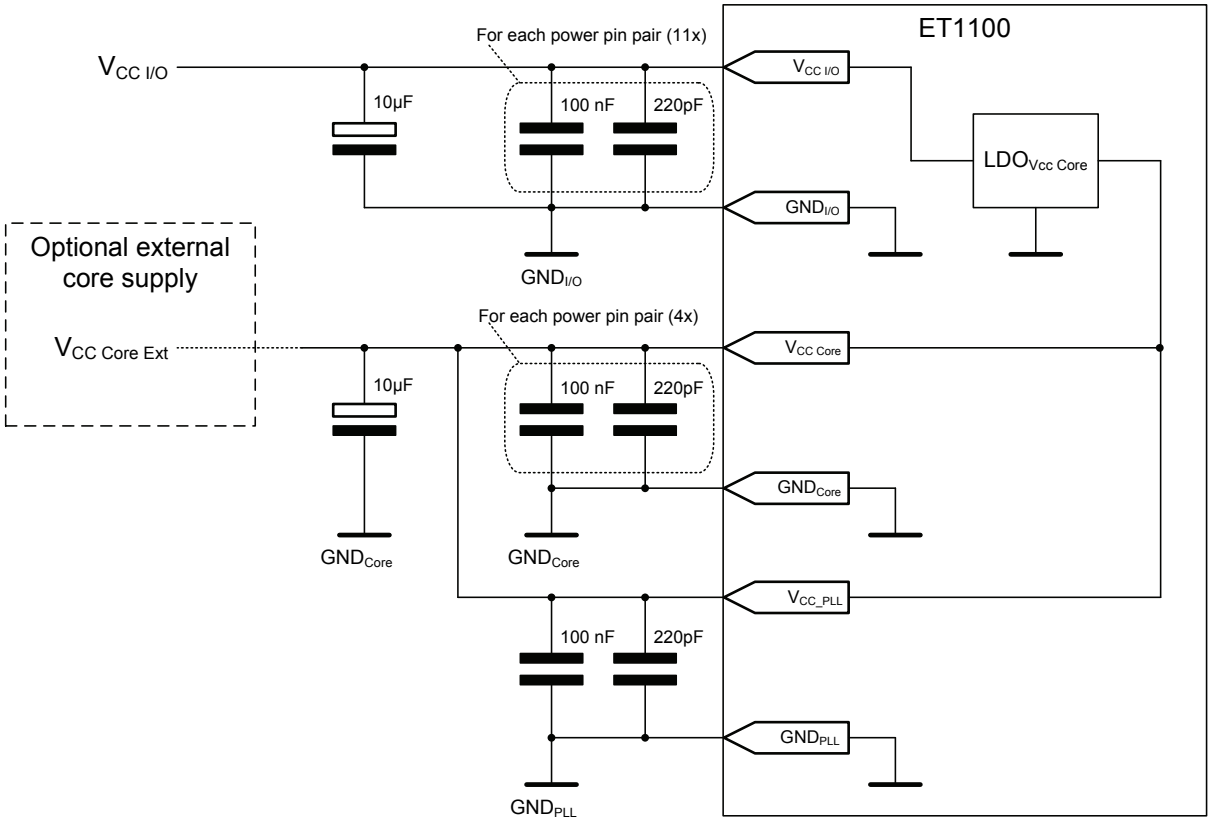


Figure 45: ET1100 power supply

Recommendation for voltage stabilization capacitors: 220pF and 100nF ceramic capacitors for each power pin pair, additional 10µF tantalum electrolytic capacitor for $V_{CC\ I/O}$, and $V_{CC\ Core}$/$V_{CC\ PLL}$, i.e., a total of two 10µF capacitors.

$GND_{I/O}$, GND_{Core}, and GND_{PLL} can be connected to a single GND potential.

The internal LDO is self-deactivating if the actual $V_{CC\ Core}$/$V_{CC\ PLL}$ voltage is higher than the nominal LDO output voltage.

8.3 Dual purpose configuration input/LED output pins

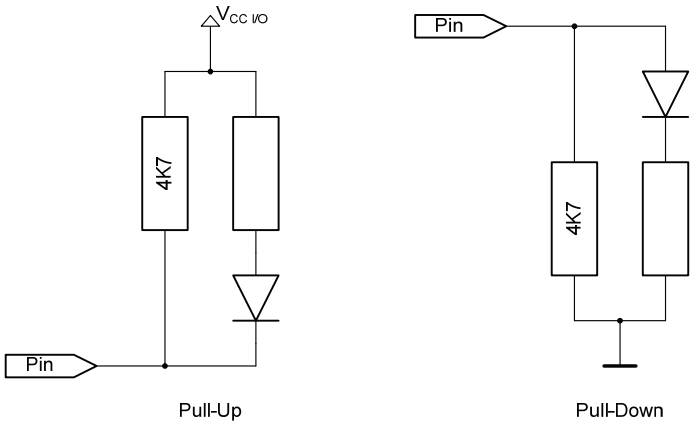


Figure 46: Dual purpose configuration input/LED output pins

8.4 PHY Connection

Refer to chapter 2.8.2 for more information on special markings (!). Take care of proper configuration of TX Shift, LINK_POL, and PHY addresses.

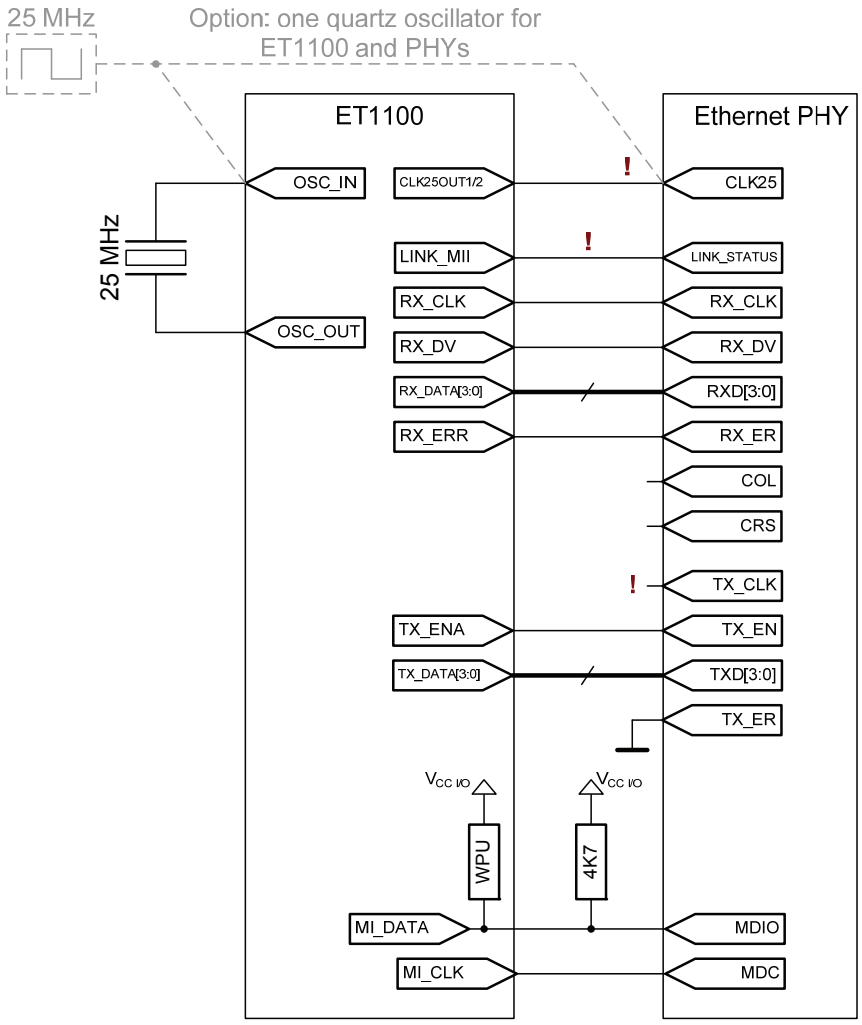


Figure 47: PHY Connection

8.5 LVDS termination

The LVDS termination with an impedance of 100 Ω is typically achieved by a resistor R_L=100 Ω. It is only necessary for EBUS ports and should be placed adjacent to the EBUS_RX inputs.

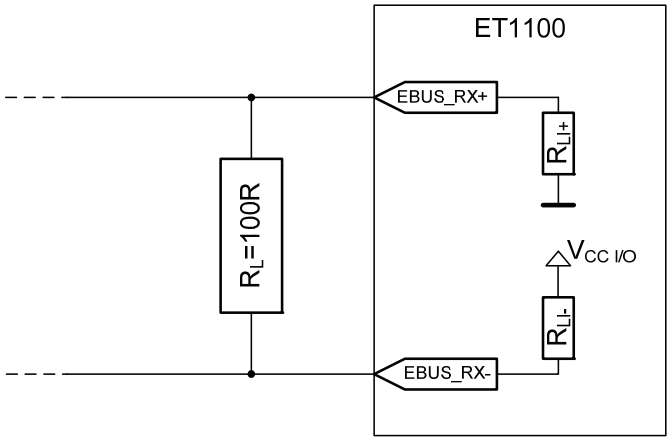


Figure 48: LVDS termination

8.6 RBIAS resistor

The LVDS RBIAS resistor should have a value of R_{BIAS}=11 kΩ.

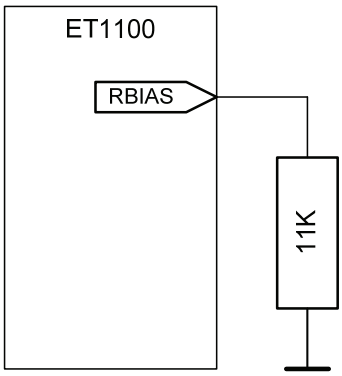


Figure 49: LVDS load resistor

NOTE: If only MII ports are used (no EBUS at all), the RBIAS resistor can be selected in the range of 10-15KΩ.

8.7 Reset Logic

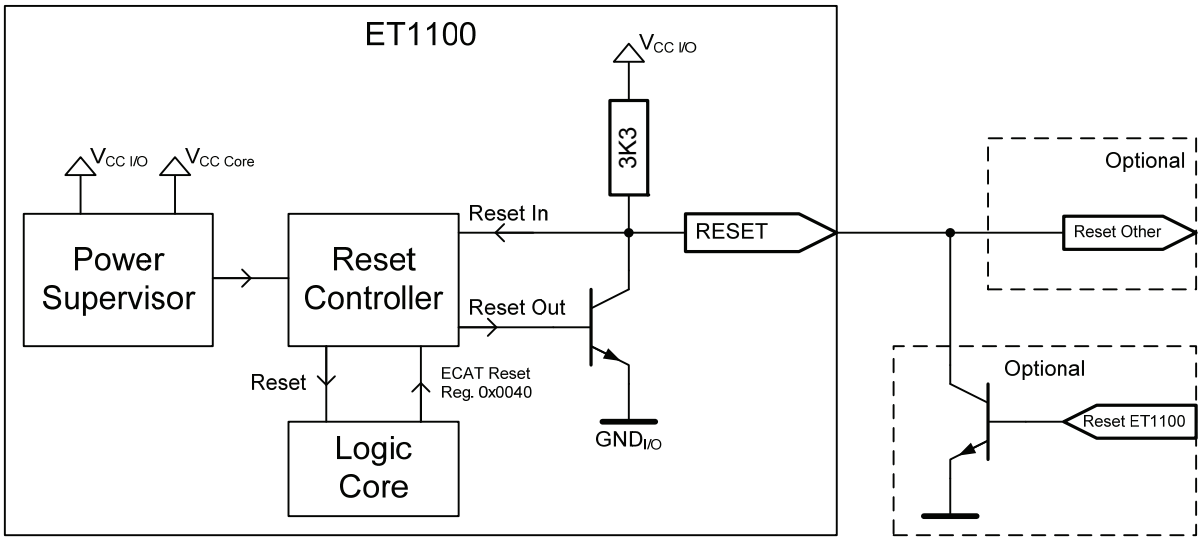


Figure 50: Reset Logic

8.8 Transparent Mode

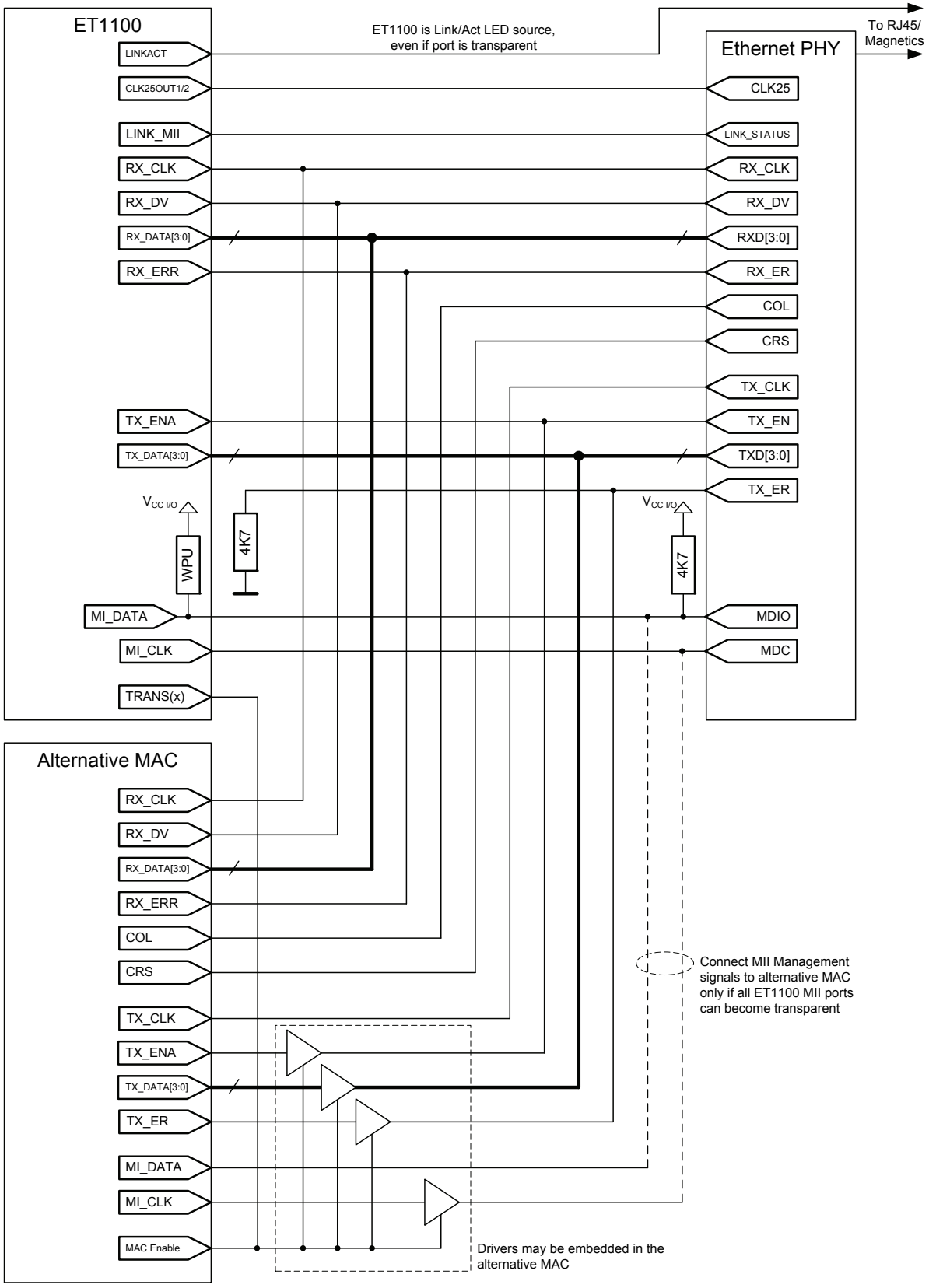


Figure 51: Transparent Mode

NOTE: MI_DATA output of alternative MAC has to be high-Z if ET1100 is controlling PHY. Observe alternative MAC's TX signal timings if extra drivers are used.

9 Electrical Specifications and Timings

9.1 Absolute Maximum Ratings

Table 76: Absolute Maximum Ratings

Symbol	Parameter	Condition	Min	Max	Units
$V_{CC\ I/O}$-V_{SS}	Supply voltage for internal LDO		-0.3	5.5	V
$I_{CC\ I/O}$	Supply current	Internal LDO for $V_{CC\ Core}$ used a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=5V		a) 170 b) 220	mA
$I_{CC\ Core}$	Supply current	$V_{CC\ Core}$ sourced externally		150	mA
$\vartheta_{Storage}$	Storage temperature		-65	150	°C
$\vartheta_{Soldering}$	Soldering temperature	max. 12 s		260	°C
V_{ESC}	ESD protection	Human body model, according to MIL-STD-883E-3015.7 Class 1	2		kV
I_{DC_ESD}	Permanent current into ESD protection diodes	Only in case of forward biased ESD diodes. Input voltage above $V_{CCI/O}$ or below V_{SS}		2	mA

NOTE: Supply current does not include output driver current for PDIs and LEDs.

9.2 Electrical Characteristics

Table 77: Operating Conditions

Symbol	Parameter	Condition	Min	Typ	Max	Units
$V_{CC\ I/O}$	I/O power supply		3.0	3.3	5.5	V
$V_{CC\ Core}$	Logic power supply		2.25	2.5	2.75	V
$V_{CC\ PLL}$	PLL power supply		2.25	2.5	2.75	V
$V_{CC\ Core\ Ext}$	External logic power supply		2.5	2.5	2.75	V
$V_{CC\ PLL\ Ext}$	External PLL power supply		2.5	2.5	2.75	V
ϑ_{A}	Ambient temperature		-40		85	°C

Table 78: DC Characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Units
$V_{CC\ Core\ LDO}$	Internal LDO output voltage $V_{CC\ Core}$/$V_{CC\ PLL}$			2.4		V
$V_{Reset\ I/O}$	Reset threshold for $V_{CC\ I/O}$			2.8		V
$V_{Reset\ Core}$	Reset threshold for $V_{CC\ Core}$			1.6		V
V_{IL}	Input Low voltage (not OSC_IN)				0.7	V
V_{IH}	Input High voltage (not OSC_IN)	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=5V	2.0		a) 3.6 b) 5.5	V
$V_{IT\ OSC\_IN}$	Input threshold voltage OSC_IN (no Schmitt trigger)	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=5V	a) 1.4 b) 2.2	a) 1.6 b) 2.5	a) 1.8 b) 2.8	V
V_{OL}	Output Low voltage				0.4	V
V_{OH}	Output High voltage		2.4			V
V_{OD}	LVDS differential output voltage	R_L=100 Ω R_{BIAS}=11 KΩ	245	350	455	mV
ΔV_{OD}	Change in V_{OD} between 1 and 0				±50	mV
V_{OC}	LVDS common mode output voltage		1.125	1.25	1.375	V
ΔV_{OC}	Change in V_{OC} between 1 and 0				±50	mV
V_{ID}	LVDS differential input voltage		100			mV
V_{IC}	LVDS input voltage range		0		2.4	V
I_{OH}	Output High current				4	mA
I_{OL}	Output Low current				-3	mA
I_{IL}	Input leakage current (without internal pull-up/pull-down resistors)				±10	µA
I_{OL}	Output leakage current (tristate, without internal PU/PD)				±10	µA
R_{PU}	Internal pull-up resistor		1.6	3.3	7	kΩ
R_{WPU}	Weak internal pull-up resistor	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=5V	a) 75 b) 50	a) 110 b) 70	a) 190 b) 120	kΩ
R_{WPD}	Weak internal pull-down resistor	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=5V	a) 60 b) 40	a) 95 b) 60	a) 180 b) 110	kΩ
R_{LI+}	Internal LVDS input pull-down resistor at EBUS_RX+ pins		15	27	45	kΩ
R_{LI-}	Internal LVDS input pull-up resistor at EBUS_RX- pins		15	27	45	kΩ
R_{BIAS}	External LVDS BIAS resistor			11		kΩ
R_L	LVDS RX load resistor			100		Ω
C_{OSC}	OSC_IN/OSC_OUT pin capacitance			1.2		pF

NOTE: R_{WPU}/R_{WPD} can not be used externally, their full effectiveness appears only inside the ET1100 (realized as transistors).

NOTE: Input and output characteristics without special indication apply to all non-LVDS I/O signals.

Table 79: DC Characteristics (Supply Current – Internal LDO used)

Symbol	Parameter	Condition	Typ	Units
$I_{CC\ I/O}$	Supply current examples: a) 2xMII, 1xFMMU, DC off b) 2xMII, 1xFMMU, DC S+L c) 4xMII, 8xFMMU, DC S+L d) 2xEBUS, 1xFMMU, DC off e) 2xEBUS, 1xFMMU, DC S+L f) 4xEBUS, 8xFMMU, DC S+L	$V_{CC\ I/O}$=3.3V, Internal LDO used	 a) 49 b) 63 c) 81 d) 83 e) 98 f) 149	mA
$I_{CC\ I/O}$	Supply current examples: a) 2xEBUS, 1xFMMU, DC off b) 2xEBUS, 1xFMMU, DC S+L c) 4xEBUS, 8xFMMU, DC S+L	$V_{CC\ I/O}$=5V, Internal LDO used	 a) 102 b) 117 c) 177	mA
$I_{CC\ I/O\ Base}$	Supply current calculation base	$V_{CC\ I/O}$=3.3V, Internal LDO used	32	mA
$I_{CC\ EBUS}$	Supply current add-on to $I_{CC\ I/O\ Base}$ per EBUS port		24	mA
$I_{CC\ MII}$	Supply current add-on to $I_{CC\ I/O\ Base}$ per MII port		7	mA
$I_{CC\ DC\ Cyclic}$	Supply current add-on to $I_{CC\ I/O\ Base}$ if DC Latch or Sync enabled		5	mA
$I_{CC\ DC\ Latch}$	Supply current add-on to $I_{CC\ I/O\ Base}$ if DC Latch unit enabled		4	mA
$I_{CC\ DC\ Sync}$	Supply current add-on to $I_{CC\ I/O\ Base}$ if DC Sync unit enabled		6	mA
$I_{CC\ FMMU}$	Supply current add-on to $I_{CC\ I/O\ Base}$ per FMMU		0.5	mA
$I_{CC\ Digital}$	Supply current add-on to $I_{CC\ I/O\ Base}$ if Digital I/O PDI is selected		2	mA
$I_{CC\ SPI}$	Supply current add-on to $I_{CC\ I/O\ Base}$ if SPI PDI is selected		5	mA
I_{CC_uC}	Supply current add-on to $I_{CC\ I/O\ Base}$ if µController PDI is selected		5	mA

NOTE: Supply current does not include output driver current for PDIs and LEDs.

Table 80: DC Characteristics (Supply Current – $V_{CC\ Core}$ sourced external)

Symbol	Parameter	Condition	Typ	Units
$I_{CC\ Core}$	Supply current examples (Digital I/O): a) 2xMII, 1xFMMU, DC off b) 2xMII, 1xFMMU, DC S+L c) 4xMII, 8xFMMU, DC S+L d) 2xEBUS, 1xFMMU, DC off e) 2xEBUS, 1xFMMU, DC S+L f) 4xEBUS, 8xFMMU, DC S+L	$V_{CC\ I/O}$=3.3V, $V_{CC\ Core}$=2.5V	 a) 42 b) 58 c) 70 d) 63 e) 79 f) 117	mA
$I_{CC\ I/O}$	Supply current examples (Digital I/O): a) 2xMII, 1xFMMU, DC off b) 2xMII, 1xFMMU, DC S+L c) 4xMII, 8xFMMU, DC S+L d) 2xEBUS, 1xFMMU, DC off e) 2xEBUS, 1xFMMU, DC S+L f) 4xEBUS, 8xFMMU, DC S+L	$V_{CC\ I/O}$=3.3V, $V_{CC\ Core}$=2.5V	 a) 14 b) 14 c) 14 d) 23 e) 23 f) 39	mA

NOTE: Supply current does not include output driver current for PDIs and LEDs.

Table 81: AC Characteristics

Symbol	Parameter	Min	Typ	Max	Units
f_{CLK25}	Clock source (OSC_IN) with initial accuracy		25 MHz ± 25 ppm		
$t_{CLK25OUT1}$	CLK25OUT1 rising edge after OSC_IN rising edge		5		ns
$t_{CLK25OUT2}$	CLK25OUT2 rising edge after OSC_IN rising edge		7		ns
t_{TX_delay}	TX_ENA/TX_D[3:0] edge (TX-Shift = 00) after rising edge of a) OSC_IN b) CLK25OUT1 c) CLK25OUT2		 a) 5 b) 0 c) 38		ns
t_{CPU_CLK}	CPU_CLK (25 MHz) rising edge after OSC_IN rising edge		5		ns
t_{POR_Sample}	POR value sample time after power good		84		ms
t_{Driver_Enable}	Output drivers enabled after POR values sampled (not PDI and not Sync/LatchSignals)		80		ns
t_{Reset_In}	External reset input time	50			ns
t_{Reset_Out}	ET1100 Reset output time	80	84		ms
t_{Reset_Func}	ET1100 functional after RESET signal high (EEPROM not loaded, PDI not functional)			50	µs
$t_{Startup}$	Startup time (PDI operational after power good)		340		ms

The AC characteristics of the PDIs, Distributed Clocks, EEPROM I^2C interface, and MII interface can be found in their respective chapters.

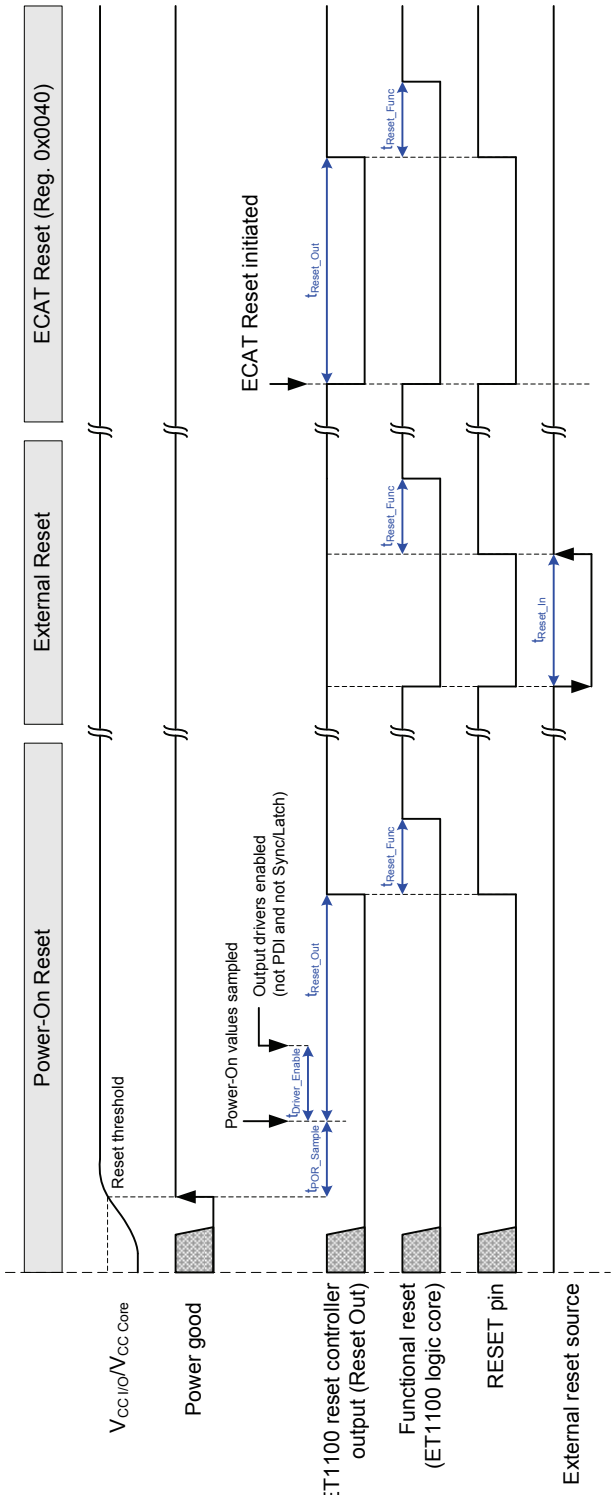


Figure 52: Reset Timing

NOTE: External clock source (quartz oscillator) is assumed to be operational at Power-good time. Otherwise t_{POR_Sampe} is delayed.

Table 82: Forwarding Delays

Symbol	Parameter	Min	Average	Max	Units
t_{Diff}	Average difference processing delay minus forwarding delay (without RX FIFO jitter) between any two ports a) at least one of the two ports is EBUS b) both ports are MII		a) 20 b) 40		ns
t_{EE}	EBUS port to EBUS port delay (FIFO size 7): a) Through ECAT Processing Unit (processing), Low Jitter off b) Alongside ECAT Processing Unit (forwarding), Low Jitter off c) Through ECAT Processing Unit (processing), Low Jitter on d) Alongside ECAT Processing Unit (forwarding), Low Jitter on	a) 140 b) 120 c) 150 d) 130	a) 150 b) 130 c) 155 d) 135	a) 160 b) 140 c) 160 d) 140	ns
t_{EM}	EBUS port to MII port delay (FIFO size 7, TX Shift=00): a) Through ECAT Processing Unit (processing), Low Jitter off b) Alongside ECAT Processing Unit (forwarding), Low Jitter off	a) 145 b) 125	a) 170 b) 150	a) 195 b) 175	ns
t_{ME}	MII port to EBUS port delay (FIFO size 7, TX Shift=00): a) Through ECAT Processing Unit (processing), Low Jitter off b) Alongside ECAT Processing Unit (forwarding), Low Jitter off c) Through ECAT Processing Unit (processing), Low Jitter on d) Alongside ECAT Processing Unit (forwarding), Low Jitter on	a) 255 b) 235 c) 265 d) 245	a) 280 b) 260 c) 290 d) 270	a) 305 b) 285 c) 315 d) 295	ns
t_{MM}	MII port to MII port delay (FIFO size 7, TX Shift=00): a) Through ECAT Processing Unit (processing), Low Jitter off b) Alongside ECAT Processing Unit (forwarding), Low Jitter off	a) 280 b) 240	a) 305 b) 265	a) 335 b) 295	ns

NOTE: Average timings are used for DC calculations.

10 Mechanical Specifications

10.1 Package Information

A 10mm x 10mm TFBGA (Thin-profile Fine-pitch BGA) with 128 balls is used for the ET1100. The pinout of the ET1100 is optimized for easy escape routing using 0.7mm/0.3mm vias inside the free center of the BGA, because the inner two ball rings are mainly used for power supply.

The ET1100 is RoHS compliant. The material of the balls is 95,5% Sn / 4% Ag / 0.5% Cu. The moisture sensitivity level of the ET1100 is MSL 3.

Non-solder mask defined pads (NSMD) with a copper pad diameter of 300 µm and an actual solder mask opening diameter of 400 µm (after widening) are recommended. Each pad (whether used or unused) should only be connected by a single trace, and the trace width should be small and identical for all pads, e.g. 125 µm.

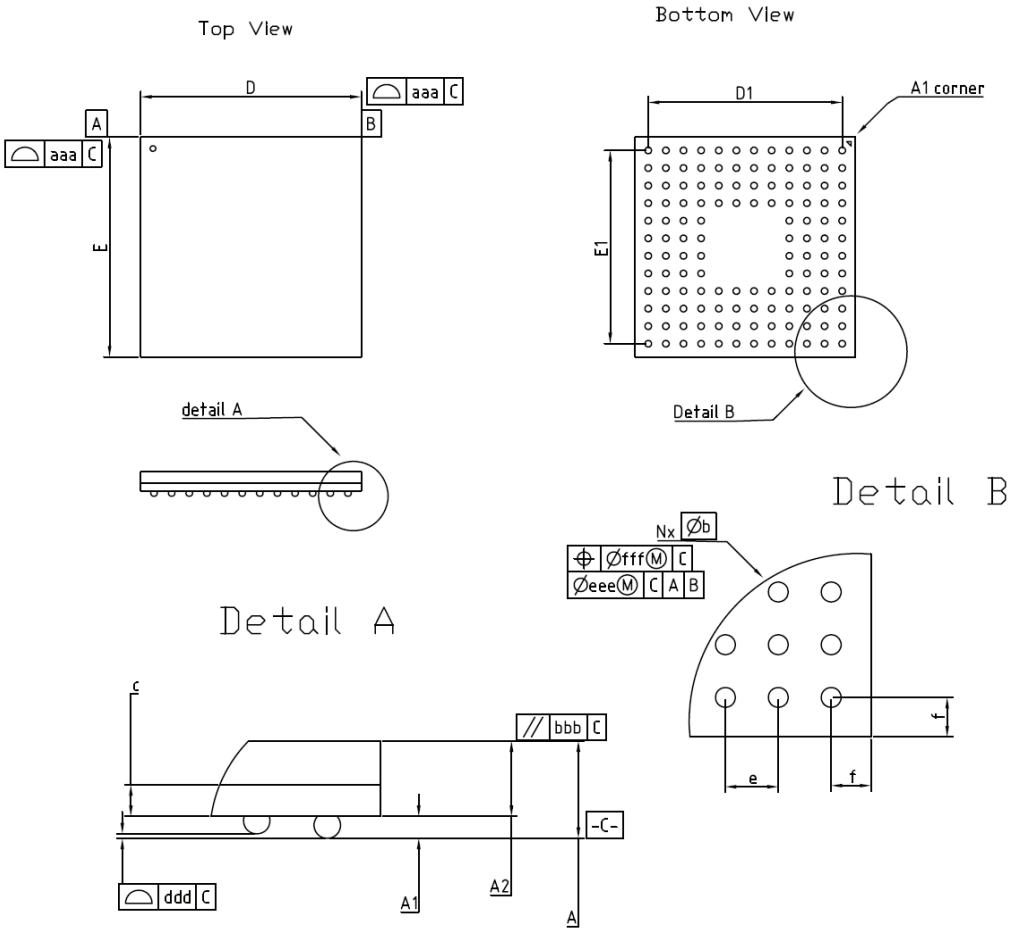


Figure 53: Package Outline

Table 83: Package Dimensions

Dimensional Ref.			
REF.	Min.	Nom	Max.
A			1.2
A1	0.2		
A2		0.82	
D		10.0	
D1		8.8	
E		10.0	
E1		8.8	
b		0.3	
c	0.28	0.32	0.36
e		0.8	
f		0.6	
m		12	
n		128	

Dimensional Tol.	
aaa	0.15
bbb	0.10
ddd	0.08
eee	0.15
fff	0.08

Notes
1. All dimensions in MM
2. 'e' represents the basic solder ball pitch
3. 'm' represents the basic solder ball matrix size. And 'n' is the number of attached solder balls
4. 'b' is measurable at the maximum solder ball diameter parallel the the primary datum -C-
5. Dimension 'aaa' is measured parallel to primary datum -C-
6. Primary datum -C- and the seating plane are defined by the spherical crowns of the solder balls
7. The package surface shall be matte finish charmilles 24 to 27
8. The over package thickness 'A' already considers collapse balls
9. Reference Jedec MO-205

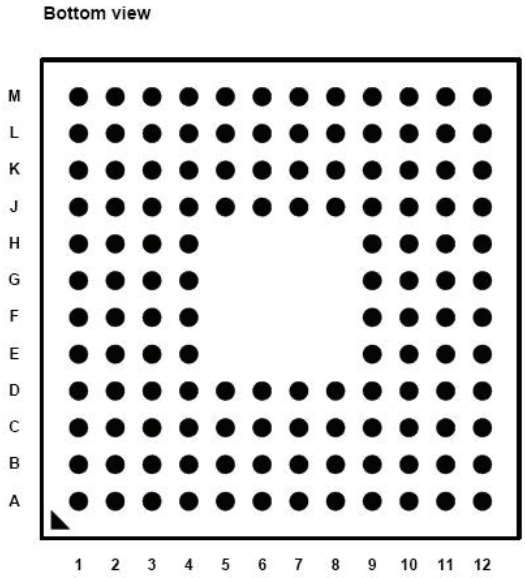


Figure 54: TFBGA 128 Pin Layout

The chip label contains the date code (X=stepping, YY=year, WW=week, optional: LLL…= lot ID).

Figure 55: Chip Label

10.2 Soldering Profile

The following soldering profile is a maximum soldering profile. For the actual soldering profile many factors have to be taken into consideration, e.g., solder paste characteristics, the PCB, other components, materials, and process type. An example soldering profile is shown below.

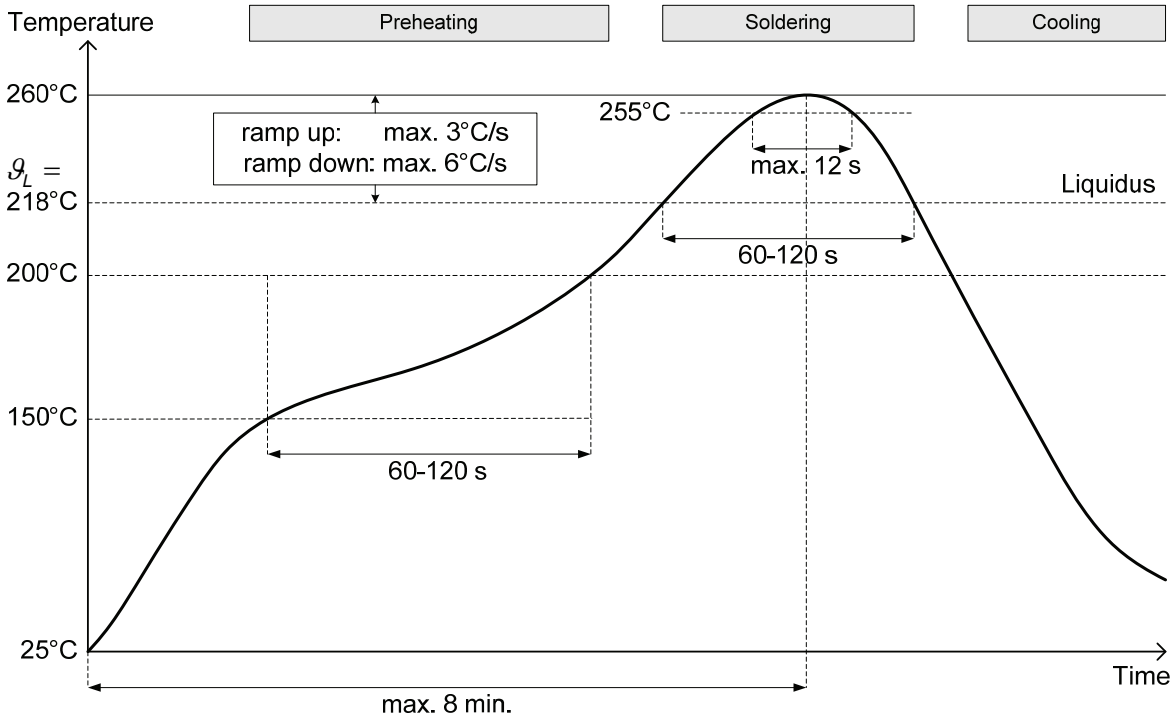


Figure 56: Maximum Soldering Profile

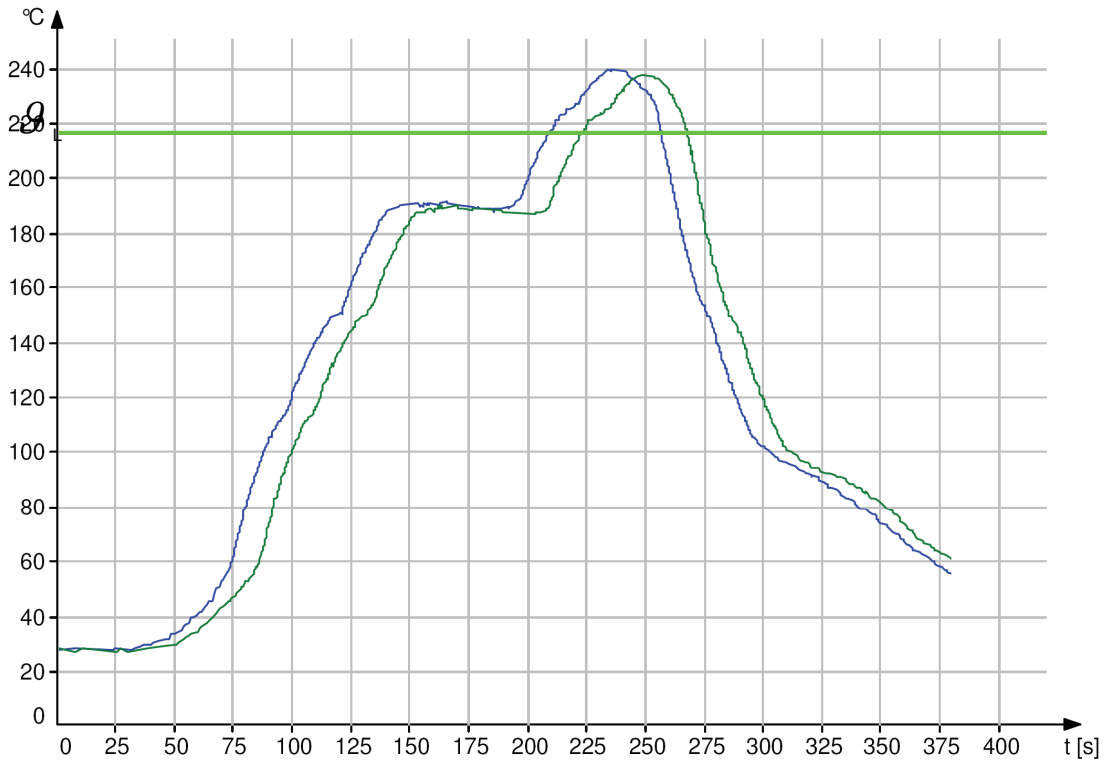


Figure 57: Example Soldering Profiles

Table 84: Example Soldering Profile

Symbol	Parameter	Example	Abs. Max.	Units
ϑ_L	Liquidus temperature	218		°C
t_L	Time above ϑ_L (TAL)	45		s
ϑ_P	Peak temperature	240	260	°C
t_P	Time at ϑ_P	10	12	s

11 Register 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 is 8KByte (end address 0x2FFF).

Table 86 gives an overview of the available registers.

Table 85: Register Overview Legend

Symbol	Description
x	Available
-	Not available
s	Available if DC SYNC Out Unit enabled (Register 0x0140.10=1)
l	Available if DC Latch In Unit enabled (Register 0x0140.11=1)
s/l	Available if DC SYNC Out Unit enabled and/or DC Latch In Unit enabled (Register 0x0140.10=1 and/or 0x0140.11=1)

Table 86: Register Overview

Register Address[5]	Length (Byte)	Description	Available
0x0000	1	Type	x
0x0001	1	Revision	x
0x0002:0x0003	2	Build	x
0x0004	1	FMMUs supported	x
0x0005	1	SyncManagers supported	x
0x0006	1	RAM Size	x
0x0007	1	Port Descriptor	x
0x0008:0x0009	2	ESC Features supported	x
0x0010:0x0011	2	Configured Station Address	x
0x0012:0x0013	2	Configured Station Alias	x
0x0020	1	Write Register Enable	x
0x0021	1	Write Register Protection	x
0x0030	1	ESC Write Enable	x
0x0031	1	ESC Write Protection	x
0x0040	1	ESC Reset ECAT	x
0x0041	1	ESC Reset PDI	-
0x0100:0x0103	4	ESC DL Control	x
0x0108:0x0109	2	Physical Read/Write Offset	x
0x0110:0x0111	2	ESC DL Status	x
0x0120:0x0121	2	AL Control	x
0x0130:0x0131	2	AL Status	x
0x0134:0x0135	2	AL Status Code	x
0x0140:0x0141	2	PDI Control	x

[5] Registers not listed here are reserved. They are not writable. A read access to reserved registers receives 0 as return value.

Register Address[5]	Length (Byte)	Description	Available
0x0150:0x0153	4	PDI Configuration	x
0x0200:0x0201	2	ECAT Event Mask	x
0x0204:0x0207	4	AL Event Mask	x
0x0210:0x0211	2	ECAT Event Request	x
0x0220:0x0223	4	AL Event Request	x
0x0300:0x0307	4x2	RX Error Counter[3:0]	x
0x0308:0x030B	4x1	Forwarded RX Error Counter[3:0]	x
0x030C	1	ECAT Processing Unit Error Counter	x
0x030D	1	PDI Error Counter	x
0x0310:0x0313	4x1	Lost Link Counter[3:0]	x
0x0400:0x0401	2	Watchdog Divider	x
0x0410:0x0411	2	Watchdog Time PDI	x
0x0420:0x0421	2	Watchdog Time Process Data	x
0x0440:0x0441	2	Watchdog Status Process Data	x
0x0442	1	Watchdog Counter Process Data	x
0x0443	1	Watchdog Counter PDI	x
0x0500:0x050F	16	ESI EEPROM Interface	x
0x0510:0x0515	6	MII Management Interface	x
0x0516:0x0517	2	MII Management Access State	-
0x0518:0x051B	4	PHY Port Status[3:0]	-
0x0600:0x067C	8x13(16)	FMMU[7:0]	x
0x0800:0x083F	8x8	SyncManager[7:0]	x
0x0900:0x090F	4x4	DC – Receive Times[3:0]	x
0x0910:0x0917	8	DC – System Time	s/l
0x0918:0x091F	8	DC – Receive Time EPU	s/l
0x0920:0x0935	24	DC – Time Loop Control Unit	s/l
0x0980	1	DC – Cyclic Unit Control	s
0x0981:0x0983	3	DC – SYNC Out Unit	s
0x0984	1	DC – Activation Status	-
0x098E:0x09A7	26	DC – SYNC Out Unit	s
0x09A8:0x09A9 0x09AE:0x09CF	36	DC – Latch In Unit	l
0x09F0:0x09F3 0x09F8:0x09FF	12	DC – SyncManager Event Times	s/l
0x0E00	1	Power-On values	x
0x0F00:0x0F03	4	Digital I/O Output Data	x
0x0F10:0x0F11	2	General Purpose Outputs	x
0x0F18:0x0F19	2	General Purpose Inputs	x
0x0F80:0x0FFF	128	User RAM	x

11.1 Revision/Build History

Table 87: Revision/Build History

Revision Register 0x0001	Build Register 0x0002:0x0003	Stepping
0x00	0x0000	ET1100-0000 ET1100-0001
0x00	0x0002	ET1100-0002

12 Appendix

12.1 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.

12.1.1 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.

12.2 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