Hardware data sheet

ET1150

Preliminary

EtherCAT® slave controller

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

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

Section III – Hardware description
Pinout, interface description, electrical and mechanical specification, ET1150 features and registers

Version 0.5
Date: 2026-06-22

BECKHOFF

DOCUMENT ORGANIZATION

The Beckhoff EtherCAT slave controller (ESC) documentation covers the following Beckhoff ESCs:

- ET1200
- ET1100, ET1150
- EtherCAT IP core for FPGAs
- ESC20

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

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

Section I – technology (all ESCs)

Section I deals with the basic EtherCAT technology. Starting with the EtherCAT protocol itself, the frame processing inside EtherCAT slaves is described. The features and interfaces of the physical layer with its two alternatives Ethernet and EBUS are explained afterwards. Finally, the details of the functional units of an ESC like FMMU, SyncManager, distributed clocks, slave information interface, interrupts, watchdogs, and so on, are described.

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

Section II – register description (all ESCs)

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

Section III – hardware description (specific ESC)

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

Additional documentation

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

DOCUMENT HISTORY

Version	Comment
0.1	Initial version
0.2	• Editorial changes
0.3	• Update SPI slave bidirectional mode description • Fix boundary scan EXTEST instruction code, add CLAMP instruction • Editorial changes
0.4	• Add voltage supervisor thresholds • Correctly indicate PDI error functions by changing pin names of LED_RUN/EEPROM SIZE/LED_PDI0_ERR/EE_EMU[2] and LED_ERR/LED_PDI1_ERR/PHYAD_OFF[0] • Adding descriptions for LED_PDI0_ERR and LED_PDI1_ERR • Editorial changes
0.5	• SPI master: correct description for number of MISO/MOSI signals depending on configuration • Update EEPROM load delay t_{Delay} • Improve EEPROM_SIZE strap pull-down description to avoid configuration mistakes • 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
SII	Slave Information Interface
SM	SyncManager
SOF	Start of Frame
SPD	Strap pull down
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 ET1150 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 ET1150 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.

The ET1150 is designed to be drop-in compatible with the ET1100, in most configurations, and potentially requiring small changes.

Table 1: ET1150 main features

Feature	ET1150
Ports	1-4 ports (each MII, or RGMII)
FMMUs	16
SyncManagers	16
RAM	15 Kbyte
Distributed clocks	Yes, 64 bit (power saving options with SII EEPROM configuration) • 0-4 distributed clocks SyncSignals • 0-4 distributed clocks LatchSignals
Process data Interfaces	Up to two interfaces: PDI0: • 32 bit digital I/O (unidirectional/bidirectional) • SPI slave (1/2/4/8 MISO/MOSI signals) • SPI master (1/2/4/8 MISO/MOSI signals) • 8/16 bit asynchronous/synchronous µController • 8/16/32 bit multiplexed asynchronous µController PDI1: • 32 bit digital I/O (unidirectional/bidirectional) • SPI slave (1/2/4/8 MISO/MOSI signals) • SPI master (1/2/4/8 MISO/MOSI signals) • 8 bit multiplexed asynchronous µController Up to 46 bit general purpose IO, up to 32 bit bidirectional general purpose IO
Power supply	• Single power supply with 2.5V or 3.3V I/O, dual power supply required for 1.8V I/O (either 2.5V or 3.3V) • Integrated voltage regulator (LDO) for logic core/PLL • Integrated switching regulator (DC/DC) for logic core/PLL • optional external power supply for logic core/PLL.
I/O	• Separate I/O banks for PDI and communication • Configurable I/O voltage for both banks (1.8V/2.5V/3.3V)
Package	BGA128 (10x10 mm^2)
Other features	• Internal 1GHz PLL • Configurable clock outputs for external devices

The general functionality of the ET1150 EtherCAT slave controller (ESC) is shown here:

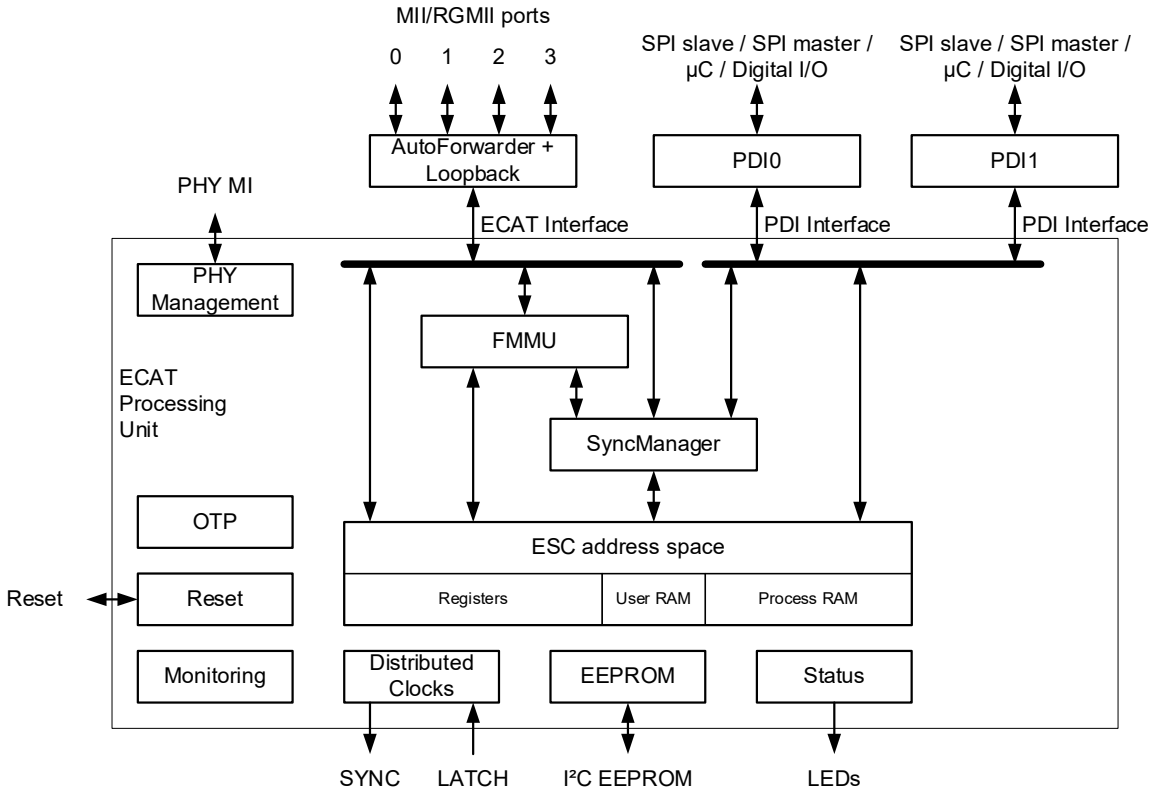

Figure 1: ET1150 block diagram

1.1 Frame processing order

The frame processing order of the ET1150 depends on the number of ports and the chip mode (logical port numbers are used):

Table 2: Frame processing order

Number of Ports	Frame processing order
1	0→EtherCAT processing unit→1
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

Figure 2 shows the frame processing in general:

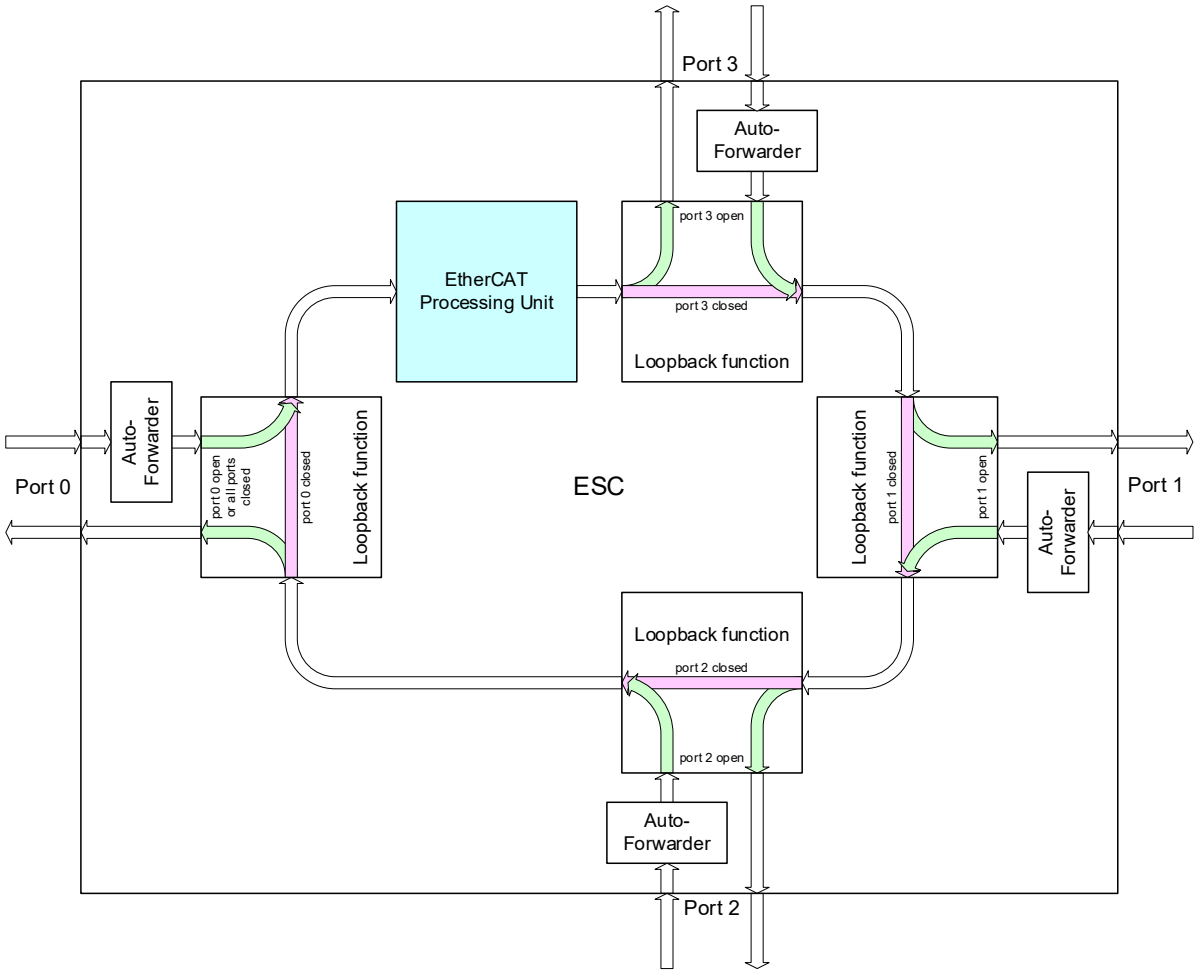

Figure 2: ET1150 frame processing

1.2 Scope of this document

This documentation refers to stepping ET1150-0002.

1.3 Revision/build history

Table 3: Revision/build history

Revision register 0x0001	Build register 0x0002:0x0003	JTAG INFO instruction [63:48]	Stepping/marking
0x02	0x0000	0x0000	ET1150-0002

The stepping code is printed on the devices, do not confuse the stepping code with the ordering codes.

1.4 Differences between ET1100 and ET1150

The ET1150 is designed to be drop-in compatible with the ET1100, in most configurations, and potentially requiring small changes. Most new features of the ET1150 need to be activated explicitly, default is ET1100 compatible behavior. The features are activated by new configuration pins, using former unused pins, or activation by EEPROM or register values.

1.5 Major improvements of ET1150 over ET1100

This is an overview of the major improvements, please refer to detailed descriptions in the specific parts of this data sheet, and the register description in section II:

- Lower power consumption, due to:
    - logic design improvements
    - advanced CMOS technology
    - optional use of an integrated DC/DC converter
- I/O voltage support for 1.8V, 2.5V, or 3.3V. Individually configurable for communication ports 0/1, and PDI.
- Integrated one-time-programmable memory for ESC identification, configuration, and application data. In-system programmable via EtherCAT, PDI, or JTAG.
- JTAG interface for boundary scan testing, OTP programming, and debugging.
  ➔ JTAG pins should be made available on the PCB, even if boundary scan testing is not used, e.g. as test points
- Enhanced EtherCAT core functions:
    - 16 FMMU
    - 16 SyncManager
    - 15 Kbyte process data RAM
    - 4x distributed clock SyncSignals
    - 4x distributed clock LatchSignals
    - SII EEPROM emulation supported (no I2C EEPROM required)
    - Extended error registers and LEDs
- Physical layer:
    - RGMII PHY support
    - Automatic MII TX_CLK phase compensation
    - Enhanced PHY management interface: PDI control, PHY configuration and link detection, FX link support
- PDI:
    - Significantly higher PDI performance due to higher internal bandwidth: 200 MHz/32 bit vs. ET1100 with 12.5 MHz/8 bit
    - Two PDI interfaces
    - SPI slave PDI with 1/2/4/8 data lines, unidirectional or bidirectional
    - SPI master PDI with 1/2/4/8 data lines, unidirectional or bidirectional
    - multiplexed, asynchronous µController PDI with 8/16/32 bit
    - 6 additional pins available for PDIs
    - unused PDI pins are available as general purpose I/Os, additional µController clock etc.

1.6 Limitations for ET1150 used as drop-in replacement in ET1100 designs

There are conditions which do not allow direct replacement of an ET1100 by the ET1150 ESC. This chapter describes these differences, and potentially required changes to the design.

Table 4: Critical differences between ET1100 and ET1150

Description	ET1100	ET1150
V_{CC_core}/V_{CC_PLL}	2.5V	1.1V
V_{CC_IO}=5V operation	Supported	Not supported
EBUS ports	Supported	Not supported
Internal processing data bus width	8 bit	32 bit

External core supply voltage (V_{CC_core}/V_{CC_PLL})

The core supply voltage of the ET1150 is lower than the ET1100 core supply voltage. If this voltage is not generated by the internal LDO, the voltage has to be adjusted.
Using internal LDO or DC-DC converter is recommended for future upgrades.

5V I/O voltage

The ET1100 allowed 5V I/O voltage, although it was not recommended. This I/O voltage is no longer supported.

EBUS ports

The ET1150 no longer supports EBUS.

Internal processing data bus width

In some cases, using the same SyncManager start addresses is not possible, due to the larger internal processing data bus width: the ET1150 treats each 32 bit word of the memory as a single address, and it is not allowed to configure multiple SyncManagers for the same 32 bit word.

Therefore, the start address of each SyncManager buffer has to be rounded down to the start of a 32 bit word, and the end address of each SyncManager buffer has to be rounded up to the end of a 32 bit word. In buffered mode, this extended memory area has to be multiplied by 3 (for buffer 1 and 2). The resulting memory area cannot be used for other SyncManagers. Their start position has to be moved, the SII EEPROM has to be updated, and potentially µController firmware.

Please note that there is no general restriction on the SyncManager start address, end address, or length – none of them has to be 32 bit aligned.

This example demonstrates a SyncManager configuration with start address 0x1003 and length 0xFA:

0x1000 - 0x1002	Alignment buffer 0 (cannot be used)	Required RAM for buffer 0 (buffered mode/mailbox mode)
0x1003 - 0x10FC	**Buffer 0** (visible)	
0x10FD - 0x10FF	Alignment buffer 0 (cannot be used)	
0x1100 - 0x1102	Alignment buffer 1 (cannot be used)	Required RAM for buffer 1 (buffered mode only; mailbox: usable RAM)
0x1103- 0x11FC	**Buffer 1** (invisible, cannot be used)	
0x11FD - 0x11FF	Alignment buffer 1 (cannot be used)	
0x1200 - 0x1202	Alignment buffer 2 (cannot be used)	Required RAM for buffer 2 (buffered mode only; mailbox: usable RAM)
0x1203 – 0x12FC	**Buffer 2** (invisible, cannot be used)	
0x12FD - 0x12FF	Alignment buffer 2 (cannot be used)	
0x1300 …	Next usable RAM space	

Figure 3: SyncManager buffer allocation

2 Features and registers

2.1 Features

Table 5: ET1150 feature details

Feature	ET1150	ET1100 -0003
EtherCAT ports	1-4	2-4
Permanent ports	1-4	2-4
Optional bridge port 3	-	-
EBUS ports	0	0-4
MII ports	0-4	0-4
RMII ports	-	-
RGMII ports	-	-
Port 0	x	-
Ports 0, 1	x	x
Ports 0, 1, 2	x	x
Ports 0, 1, 3	x	x
Ports 0, 1, 2, 3	x	x
EtherCAT mode	Direct	Direct
Slave category	Full slave	Full slave
Position addressing	x	x
Node addressing	x	x
Logical addressing	x	x
Broadcast addressing	x	x
Physical layer general features		
FIFO size configurable (0x0100[18:16])	x	x
FIFO size default from SII EEPROM	x	-
Auto-forwarder checks CRC and SOF	x	x
Forwarded RX error indication, detection and counter (0x0308:0x030B)	x	x
Prevention of circulating frames	x	x
Fallback: port 0 opens if all ports are closed	x	x
VLAN tag and IP/UDP support	x	x
Enhanced link detection per port configurable	x	-
General Ethernet features (MII/RMII/RGMII)		
PHY management interface (0x0510:0x051F)	x	x
Supported PHY address offsets	0/1/16/17	0/16
Individual port PHY addresses	OTP: x	-
Port PHY addresses readable	x	-
Link polarity configurable	x	x
Enhanced link detection supported	x	x
FX PHY support (native)	x	-
Fast hot connect support (native)	x	-
PHY reset out signals	FX	-
Link detection using PHY signal (LED)	x	x
Link detection using RGMII in-band status	-	-
MI link status and configuration	PROM per port	-
User MI initialization values	OTP: 4	-
MI controllable by PDI (0x0516:0x0517)	x	-
MI read error (0x0510[13])	x	-
MI PHY configuration update status (0x0518[5])	x	-
MI preamble suppression	x	-
Additional MCLK	x	x

Feature	ET1150	ET1100 -0003
Gigabit PHY configuration	c	-
Gigabit PHY register 9 detection	full	-
Clause 45 access	x	-
Transparent mode	c	c
MII features		
CLK25OUT as PHY clock source	x	x
Bootstrap TX shift settings	x	x
Automatic TX shift setting (with TX_CLK)	x	-
TX shift not necessary (PHY TX_CLK as clock source)	-	-
FIFO size reduction steps	2	1
MII back-to-back connections	x	x
PDI general features		
Internal PDI clock	25/50/100/ 200 MHz	25 MHz TD
Internal processing data bus width [bits]	32	8
Extended PDI configuration (0x0152:0x0153)	x	x
CPU_CLK output (10, 20, 25 MHz)	c	c
CPU_CLK2 output and CPU_nRESET_ OUT	c	-
SOF, EOF, WD_TRIG and WD_STATE independent of PDI	WD_ STATE	-
Available PDIs and PDI features depending on port configuration	x	x
PDI selection at run-time (SII EEPROM)	c	x
PDI active immediately (SII EEPROM settings ignored)	c	-
PDI function acknowledge by write	c	-
PDI function acknowledge SyncManager/register independently	c	-
PDI information register 0x014E:0x014F	x	-
Support for two PDIs	x	-

Feature	ET1150	ET1100 -0003
Digital I/O PDI	**PDI0, PDI1**	**x**
Digital I/O width [bits]	8/16/24/32	8/16/24/32
PDI control register value (0x0140:0x0141)	4	4
Control/status signals:	8/0[1]	7/0[1]
LATCH_IN	x[1]	x[1]
SOF	x[1]	x[1]
OUTVALID	x[1]	x[1]
WD_TRIG	x[1]	x[1]
OE_CONF	x[1]	x[1]
OE_EXT	x[1]	x[1]
EEPROM_ Loaded	x[1]	x[1]
WD_STATE	-	-
EOF	-	-
OUT_START	x[1]	-
Granularity of direction configuration [bits]	2	2
Bidirectional mode	x	x
Output high-Z if WD expired	x	x
Output 0 if WD expired	x	x
Output with EOF	x	x
Output with DC SyncSignals	x	x
Input with SOF	x	x
Input with DC SyncSignals	x	x
Digital I/O user mode from ECAT	x	-
Digital input register	PDI1	-
SPI slave PDI	**PDI0, PDI1**	**x**
Max. SPI clock [MHz]	50	20
SPI modes configurable (0x0150[1:0])	x	x
SPI_IRQ driver configurable (0x0150[3:2])	x	x
SPI_SEL polarity configurable (0x0150[4])	x	x
Data out sample mode configurable (0x0150[5])	x	x
Busy signaling	-	-
Wait state byte(s)	x	x
Number of address extension byte(s)	any	any
2/4 byte SPI master support	x	x
Extended error detection (read busy violation)	x	x
IRQ byte 0/1 swap	x	-
SPI user mode from PDI	x	-
SPI_IRQ delay	x	x
Status indication	x	x
EEPROM_ Loaded signal	x	x
Additional select signals	c	-
SPI data width	1/2/4/8	1
SPI data bidirectional	c	-
SPI master PDI	**PDI0, PDI1**	**-**
Asynchronous µController PDI	**PDI0: 8/16 bit**	**8/16 bit**
Extended µC configuration bits 0x0150[7:4], 0x0152:0x0153	x	x
ADR[15:13] available (000_b if not available)	x	x
EEPROM_Loaded signal	x	x
RD polarity configurable (0x0150[7])	x	x
Read BUSY delay (0x0152[0])	x	x
Write after first edge (0x0152[2])	c	-
Default BUSY state	c	-
ET1100 compatible timing	c	x
Synchronous µController PDI	**PDI0: 8/16 bit**	**8/16 bit**
EEPROM_ Loaded signal	x	x
BUSY until access starts	c	-
ET1100 compatible timing	c	x
Multiplexed asynchronous µController PDI	**PDI0: 8/16/32 bit PDI1: 8 bit**	**-**
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])	c	
Default BUSY state	c	
On-chip bus PDI	**-**	**-**
Bridge port 3	**-**	**-**
Port open/closed configurable		
General purpose I/O	**x**	**x**
GPO bits	0-46	0-16
GPI bits	0-46	0-16
GPIO available independent of PDI or port configuration	-	-
GPIO available without PDI	x	-
Concurrent access to GPO by ECAT and PDI	x	x
Bidirectional GPIO bits	x	-
ESC information		
Basic information (0x0000:0x0006)	x	x
Port descriptor (0x0007)	x	x
ESC features supported (0x0008:0x0009)	x	x
Extended ESC feature availability in user RAM (0x0F80 ff.)	-	-
Write protection (0x0020:0x0031)	x	x
Data link layer features		
ECAT reset (0x0040)	x	x
PDI reset (0x0041)	x	-
ESC DL control (0x0100:0x0103) bytes	4	4
EtherCAT only mode (0x0100[0])	x	x
Temporary loop control (0x0100[1])	x	x
FIFO size configurable (0x0100[18:16])	x	x
Configured station address (0x0010:0x0011)	x	x
Configured station alias (0x0100[24], 0x0012:0x0013)	x	x
Physical read/write offset (0x0108:0x0109)	x	x
Application layer features		
Extended AL control/status bits (0x0120[15:5], 0x0130[15:5])	x	x
AL status emulation (0x0140[8])	x	x
AL status code (0x0134:0x0135)	x	x

[1] Availability depending on port configuration

Feature	ET1150	ET1100 -0003
Interrupts		
ECAT event mask (0x0200:0x0201)	x	x
AL event mask (0x0204:0x0207)	x	x
ECAT event request (0x0210:0x0211)	x	x
AL event request (0x0220:0x0223)	x	x
SyncManager activation changed (0x0220[4])	x	x
SyncManager watchdog expiration (0x0220[6])	x	-
DC SYNC interrupt	[3:0]	[1:0]
Error counters		
RX error counter (0x0300:0x0307)	x	x
Forwarded RX error counter (0x0308:0x030B)	x	x
ECAT processing unit error counter (0x030C)	x	x
PDI error counter (0x030D)	x	x
PDI error code (0x030E:0x030F)	x	-
Lost link counter (0x0310:0x0313)	x	x
Extended RX error counters (0x0314:0x0317)	x	-
RX error code (0x0320:0x0327)	x	-
PDI clearing error counters	-	-
Watchdog		
Watchdog divider configurable (0x0400:0x0401)	x	x
Watchdog process data	x	x
Watchdog PDI	x	x
Watchdog counter process data (0x0442)	x	x
Watchdog counter PDI (0x0443)	x	x
Watchdog status (0x0448)	x	-
SII EEPROM interface (0x0500:0x050F)		
I²C EEPROM sizes supported	1 Kbit-4 Mbit	1 Kbit-4 Mbit
I²C EEPROM size reflected in 0x0502[7]	x	x
EEPROM controllable by PDI	x	x
EEPROM emulation by PDI	x	-
EEPROM emulation CRC error 0x0502[11] PDI writable	x	-
Read data bytes (0x0502[6])	8	8
Internal pull-up resistors for EEPROM_CLK and EEPROM_DATA	x	x
ESC configuration area A	x	x
ESC configuration area B	x	-
I2C base address	0	0
I2C half speed	-	-
FMMUs	16	8
Bit-oriented operation	x	x
SyncManagers	16	8
Watchdog trigger generation for 1 byte Mailbox configuration independent of reading access	x	x
SyncManager event times (+0x8[7:6])	x	x
Buffer state (+0x5[7:6])	x	-
SyncManager sequential mode	x	-
SyncManager deactivation delay	x	-

Feature	ET1150	ET1100 -0003
Distributed clocks	x	x
Width	64	64
Sync/Latch signals	0-8 (0-4 Sync-Signals, 0-4 Latch-Signals)	2
SyncManager event times (0x09F0:0x09FF)	x	x
DC receive times	x	x
DC time loop control controllable by PDI	-	-
DC sync/latch activation (0x0140[11:10])	x	x
DC time loop activation (0x0142[8])	x	-
Propagation delay measurement with traffic (BWR/FPWR 0x900 detected at each port)	x	x
LatchSignal state in latch status register (0x09AE:0x09AF)	x	x
SyncSignal auto-activation (0x0981[3])	x	-
SyncSignal 32 or 64 bit start time (0x0981[4])	x	-
SyncSignal late activation (0x0981[6:5])	x	-
SyncSignal debug pulse (0x0981[7])	x	-
SyncSignal activation state 0x0984)	x	-
Reset filters after writing filter depth	x	-
Speed counter diff direct control (0x0938:0x0939)	x	-
ESC specific registers (0x0E00:0x0EFF)		
Product and vendor ID	-	-
POR values	x	x
FPGA update	-	-
Process RAM and user RAM		
Process RAM (0x1000 ff.) [Kbyte]	15	8
User RAM (0x0F80:0x0FFF)	x	x
User RAM ESC feature initialization	-	-
RAM BIST	x	-
Additional EEPROMs	1	1
SII EEPROM (I²C)	c (EEPROM of µC used)	x
FPGA configuration EEPROM	-	-
LED signals		
RUN LED	x	x
RUN LED override	x	-
Link/activity(x) LED per port	x	x
PERR(x) LED per port	x	x
Device ERR LED	x	-
STATE_RUN LED	x	-
PDI_ERR LED per PDI	x	-
Optional LED states		
RUN LED: bootstrap	x	x
RUN LED: booting	x	-
RUN LED: device identification	x	-
RUN LED: loading SII EEPROM	x	-
RUN LED: POR values signaling	x	-
Error LED: SII EEPROM loading error	x	-
Error LED: invalid hardware configuration	x	-
Error LED: process data watchdog timeout	x	-

Feature	ET1150	ET1100 -0003
Error LED: PDI watchdog timeout	x	-
Error LED: BIST error	x	-
Error LED: error indication 0x0130[4]	x	-
Link/activity: port closed	-	-
Link/activity: local auto-negotiation error	-	-
Link/activity: remote auto-negotiation error	-	-
Link/activity: unknown PHY auto-negotiation error	-	-
LED test	-	-
Clock supply		
Crystal	x	x
Crystal oscillator	x	x
TX_CLK from PHY	x	x
25ppm clock source accuracy	x	x
Internal PLL	x	x
Power supply voltages	1-3	1-2
I/O voltages		
1.8 V	x	-

Feature	ET1150	ET1100 -0003
2.5 V	x	-
3.3 V	x	x
5 V	-	(x)
Separate I/O voltages	PDI+COM	-
Core voltage	1.1V	2.5V
Internal regulators	1x LDO/ DC-DC	1x LDO
Input voltage	2.5V/3.3V/-	3.3V/5V
Output core Voltage	x	x
Output I/O Voltage	-	-
JTAG	x	-
Package	BGA128	BGA128
Size [mm²]	10x10	10x10
Original release date	2025	3/2007
Configuration and pinout calculator	x	x
Register configuration	fixed	fixed
Internal tri-strate drivers	x	x

Table 6: Legend

Symbol	Description
x	available
-	not available
c	configurable
POR	value can be changed by power-on reset values (strapping)
PROM	Value can be changed by SII EEPROM content
OTP	value can be changed by OTP content
red	Feature changed in this version

2.2 Register overview

An EtherCAT slave controller (ESC) has an address space of 64 Kbyte. The first block of 4 Kbyte (0x0000:0x0FFF) is reserved for registers. The process data RAM starts at address 0x1000, its size is 15 Kbyte (end address 0x4BFF).

Table 8 gives an overview of the available registers.

Table 7: Register overview legend

Symbol	Description	ET1150 EEPROM setting	ET1100 EEPROM setting
x	Available		
-	Not available		
r	Read only		
c	Configurable		
RT	DC receive times enabled	x	x
DC	DC time loop control enabled	0x0140[10]=1 or 0x0140[11]=1 or 0x0142[8]=1	0x0140[10]=1 or 0x0140[11]=1
SL	DC sync unit and/or latch unit enabled	0x0140[10]=1 or 0x0140[11]=1	0x0140[10]=1 or 0x0140[11]=1
S	DC sync unit enabled	0x0140[10]=1	0x0140[10]=1
S#	DC sync unit instance # enabled	0x0140[10]=1	0x0140[10]=1
L	DC latch In unit enabled	0x0140[11]=1	0x0140[11]=1
L#	DC latch In unit instance # enabled	0x0140[11]=1	0x0140[11]=1
ET	DC SyncManager event times are enabled	0x0140[10]=1 or 0x0140[11]=1	0x0140[10]=1 or 0x0140[11]=1
io	Available if Digital I/O PDI is selected		
red	Register changed in this version		

Table 8: Register overview

Address	Length (byte)	Description	ET1150	ET1100
0x0000	1	Type	x	x
0x0001	1	Revision	x	x
0x0002:0x0003	2	Build	x	x
0x0004	1	FMMUs supported	x	x
0x0005	1	SyncManagers supported	x	x
0x0006	1	RAM size	x	x
0x0007	1	Port descriptor	x	x
0x0008:0x0009	2	ESC features supported	x	x
0x0010:0x0011	2	Configured station address	x	x
0x0012:0x0013	2	Configured station alias	x	x
0x0020	1	Write register enable	x	x
0x0021	1	Write register protection	x	x
0x0030	1	ESC write enable	x	x

Address	Length (byte)	Description	ET1150	ET1100
0x0031	1	ESC write protection	x	x
0x0040	1	ESC reset ECAT	x	x
0x0041	1	ESC reset PDI	x	-
0x0100:0x0101	2	ESC DL control	x	x
0x0102:0x0103	2	Extended ESC DL control	x	x
0x0108:0x0109	2	Physical read/write offset	x	x
0x0110:0x0111	2	ESC DL status	x	x
0x0120	5 bits [4:0]	AL control	x	x
0x0120:0x0121	2	AL control	x	x
0x0130	5 bits [4:0]	AL status	x	x
0x0130:0x0131	2	AL status	x	x
0x0134:0x0135	2	AL status Code	x	x
0x0138	1	RUN LED override	x	-
0x0139	1	ERR LED override	x	-
0x0140	1	PDI0 control	x	x
0x0141	1	ESC configuration A0	x	x
0x0142:0x0143	2	ESC configuration A5	x	-
0x0144:0x0145	2	ESC configuration A6	x	-
0x014E:0x014F	2	PDI0 information	x	-
0x0150	1	PDI0 configuration	x	x
0x0151	1	DC sync/latch configuration	x	x
0x0152:0x0153	2	Extended PDI0 configuration	x	x
0x0158:0x0159	2	PDI0 user mode from ECAT	x	-
0x015C:0x015D	2	PDI0 user mode from PDI	x	-
0x0180	1	PDI1 control	x	-
0x0181	1	ESC configuration B0	x	-
0x0182:0x0183	2	ESC configuration B5	x	-
0x0188:0x0189	2	ESC configuration B4	x	-
0x018E:0x018F	2	PDI1 information	x	-
0x0190	1	PDI1 configuration	x	-
0x0191	1	Reserved	-	-
0x0192:0x0193	2	Extended PDI1 configuration	x	-
0x0198:0x0199	2	PDI1 user mode from ECAT	x	-
0x019C:0x019D	2	PDI1 user mode from PDI	x	-
0x0200:0x0201	2	ECAT event mask	x	x
0x0204:0x0207	4	PDI0 AL event mask	x	x
0x020A:0x020D	4	PDI1 AL event mask	x	-
0x0210:0x0211	2	ECAT event request	x	x
0x0220:0x0223	4	AL event request	x	x
0x0300:0x0307	4x2	RX error counter[3:0]	x	x
0x0308:0x030B	4x1	Forwarded RX error counter[3:0]	x	x
0x030C	1	ECAT processing unit error counter	x	x

Address	Length (byte)	Description	ET1150	ET1100
0x030D	1	PDI0 error counter	x	x
0x030E:0x030F	2	PDI0 error code	x	-
0x0310:0x0313	4x1	Lost link counter[3:0]	x	x
0x0314:0x0317	4x1	Extended RX error counter[3:0]	x	-
0x0320:0x0327	4x2	RX error code[3:0]	x	-
0x0340	1	PDI1 error counter	x	-
0x0341	1	PDI1 error code	x	-
0x0400:0x0401	2	Watchdog divider	x	x
0x0410:0x0411	2	Watchdog time PDI0	x	x
0x0412:0x0413	2	Watchdog time PDI1	x	-
0x0420:0x0421	2	Watchdog time process data	x	x
0x0440:0x0441	2	Watchdog status process data	x	x
0x0442	1	Watchdog counter process data	x	x
0x0443	1	Watchdog counter PDI0	x	x
0x0444	1	Watchdog counter PDI1	x	-
0x0448	1	Watchdog status PDI	x	-
0x0500:0x050F	16	SII EEPROM interface	x	x
0x0510:0x0515	6	PHY management interface	x	x
0x0516:0x0517	2	PHY management access state	x	-
0x0518:0x051B	4	PHY port status[3:0]	x	-
0x0600:0x06FC	16x13	FMMU[15:0]	16	8
0x0800:0x087F	16x8	SyncManager[15:0]	16	8
0x0900:0x090F	4x4	DC – receive times	x	x
0x0910:0x0917	8	DC – system time	DC	SL
0x0918:0x091F	8	DC – receive time EPU	DC	SL
0x0920:0x0927	8	DC – system time offset	DC	SL
0x0928:0x092B	4	DC – system time delay	DC	SL
0x092C:0x092F	4	DC – system time difference	DC	SL
0x0930:0x0931	2	DC – speed counter start	DC	SL
0x0932:0x0933	2	DC – speed counter diff	DC	SL
0x0934	1	DC – system time difference filter depth	DC	SL
0x0935	1	DC – speed counter filter depth	DC	SL
0x0936	1	DC – receive time latch mode	-	-
0x0938:0x0939	2	DC – speed counter diff direct control	DC	-
0x0940:0x0943	4	DC – SYNC2 cycle time	S0	-
0x0944:0x0947	4	DC – SYNC3 cycle time	S0	-
0x0950:0x0957	8	DC – next SYNC2 pulse	S0	-
0x0958:0x095F	8	DC – next SYNC3 pulse	S0	-
0x0980	1	DC – cyclic unit control	SL	S0
0x0981	1	DC – activation	S0	S0
0x0982:0x0983	2	DC – pulse length of SyncSignals	S0	S0

Address	Length (byte)	Description	ET1150	ET1100
0x0984	1	DC – activation status	S0	-
0x0986	1	DC – activation SYNC2/3	S0	-
0x098C	1	DC – SYNC2 status	S0	-
0x098D	1	DC – SYNC3 status	S0	-
0x098E	1	DC – SYNC0 status	S0	S0
0x098F	1	DC – SYNC1 status	S0	S0
0x0990:0x0997	8	DC – start time cyclic operation/next SYNC0 pulse	S0	S0
0x0998:0x099F	8	DC – next SYNC1 pulse	S0	S0
0x09A0:0x09A3	4	DC – SYNC0 cycle time	S0	S0
0x09A4:0x09A7	4	DC – SYNC1 cycle time	S0	S0
0x09A8	1	DC – latch0 control	L0	L0
0x09A9	1	DC – latch1 control	L0	L0
0x09AA	1	DC – latch2 control	L0	-
0x09AB	1	DC – latch3 control	L0	-
0x09AC	1	DC – latch2 status	L0	-
0x09AD	1	DC – latch3 status	L0	-
0x09AE	1	DC – latch0 status	L0	L0
0x09AF	1	DC – latch1 status	L0	L0
0x09B0:0x09B7	8	DC – latch0 positive edge	L0	L0
0x09B8:0x09BF	8	DC – latch0 negative edge	L0	L0
0x09C0:0x09C7	8	DC – latch1 positive edge	L0	L0
0x09C8:0x09CF	8	DC – latch1 negative edge	L0	L0
0x09D0:0x09D7	8	DC – latch2 positive edge	L0	-
0x09D8:0x09DF	8	DC – latch2 negative edge	L0	-
0x09E0:0x09E7	8	DC – latch3 positive edge	L0	-
0x09E8:0x09EF	8	DC – latch3 negative edge	L0	-
0x09F0:0x09F3	4	DC – EtherCAT buffer change event time	DC	SL
0x09F8:0x09FB	4	DC – PDI buffer start event time	DC	SL
0x09FC:0x09FF	4	DC – PDI buffer change event time	DC	SL
0x0E00:0x0E07	8	Power-on values [bits]	40	16
0x0E00:0x0E07	8	Product ID	-	-
0x0E08:0x0E0F	8	Vendor ID	-	-
0x0E10:0x0E17	8	ESC health status	x	-
0x0F00:0x0F03	4	Digital I/O output data	x	x
0x0F08:0x0F0B	4	Digital I/O input data	x	-
0x0F10:0x0F17	8	General purpose outputs [byte]	8	2
0x0F18:0x0F1F	8	General purpose inputs [byte]	8	2
0x0F80:0x0FFF	128	User RAM	x	x
0x1000:0x1003	4	Digital I/O input data	io	io
0x1000 ff.		Process data RAM [Kbyte]	15	8

3 Package pinout

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[45:0] area when bidirectional digital I/O is used). Unused PDI[45:0] input pins should be pulled down or connected to GND, all other input pins can be connected to GND directly.

Pull-up resistors must connect to $V_{CC\ I/O}$, not to a different power source. Otherwise the ET1150 could be powered via the resistors and the internal clamping diodes as long as $V_{CC\ I/O}$ is below the other power source.

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

NOTE: The term physical port in this document is only used for grouping ET1150 interface pins. The register set as well as any master/slave software is always based on logical ports.

Table 9: Pinout

Pin	Pin name	Dir.	Voltage reference	Strap signal	Internal PU/PD
	General pins				
G11	EEPROM_CLK/EE_EMU[0]	BD	$V_{CC\_IO\_COM}$	EE_EMU[0]	3.3 kΩ PU
F11	EEPROM_DATA/EE_EMU[1]	BD	$V_{CC\_IO\_COM}$	EE_EMU[1]	3.3 kΩ PU
H10	LED_ERR/LED_PDI1_ERR/PHYAD_OFF[0]	BD	$V_{CC\ IO\ COM}$	PHYAD_OFF[0]	SPD
H11	LED_RUN/EEPROM_SIZE/ LED_PDI0_ERR/EE_EMU[2]	BD	$V_{CC\_IO\_COM}$	EEPROM_SIZE/ EE_EMU[2]	SPD
D8	LED_STATE_RUN/EE_EMU_ENA	BD	$V_{CC\ IO\ COM}$	EE_EMU_ENA	SPD
K11	MI_CLK/LINK_POL	BD	$V_{CC\ IO\ COM}$		SPD
K12	MI_DATA	BD	$V_{CC\ IO\ COM}$		WPU
E11	DC_SYNC/LATCH[0]	BD	$V_{CC\ IO\ PDI}$		
E12	DC_SYNC/LATCH[1]	BD	$V_{CC\ IO\ PDI}$		
E10	WD_STATE/POR_ET1100	BD	$V_{CC\ IO\ COM}$	POR_ET1100	SPD
C4	n.c.	BD	$V_{CC\ IO\ COM}$		WPU
	PDI 0-15				
D12	PDI[0]	BD	$V_{CC\ IO\ PDI}$		
D11	PDI[1]	BD	$V_{CC\ IO\ PDI}$		
C12	PDI[2]	BD	$V_{CC\ IO\ PDI}$		
C11	PDI[3]	BD	$V_{CC\ IO\ PDI}$		
B12	PDI[4]	BD	$V_{CC\ IO\ PDI}$		
C10	PDI[5]	BD	$V_{CC\ IO\ PDI}$		
A12	PDI[6]	BD	$V_{CC\ IO\ PDI}$		
B11	PDI[7]/CPU_CLK_OUT1	BD	$V_{CC\ IO\ PDI}$		
A11	PDI[8]	BD	$V_{CC\ IO\ PDI}$		
B10	PDI[9]	BD	$V_{CC\ IO\ PDI}$		
A10	PDI[10]	BD	$V_{CC\ IO\ PDI}$		
C9	PDI[11]	BD	$V_{CC\ IO\ PDI}$		
A9	PDI[12]	BD	$V_{CC\ IO\ PDI}$		
B9	PDI[13]	BD	$V_{CC\ IO\ PDI}$		
A8	PDI[14]	BD	$V_{CC\ IO\ PDI}$		
B8	PDI[15]	BD	$V_{CC\ IO\ PDI}$		
	PDI 16-31, physical port 3				
A7	PDI[16]/RX_ERR(3)	BD	$V_{CC\ IO\ PDI}$		
B7	PDI[17]/RX_CLK(3)	BD	$V_{CC\ IO\ PDI}$		
A6	PDI[18]/RX_D(3)[0]	BD	$V_{CC\ IO\ PDI}$		
B6	PDI[19]/RX_D(3)[2]	BD	$V_{CC\ IO\ PDI}$		
A5	PDI[20]/RX_D(3)[3]	BD	$V_{CC\ IO\ PDI}$		
B5	PDI[21]/LINK_MII(3)	BD	$V_{CC\ IO\ PDI}$		
A4	PDI[22]/TX_D(3)[3]	BD	$V_{CC\ IO\ PDI}$		
B4	PDI[23]/TX_D(3)[2]	BD	$V_{CC\ IO\ PDI}$		
A3	PDI[24]/TX_D(3)[1]/RGMII[3]	BD/LO-	$V_{CC\ IO\ PDI}$	RGMII[3]	
B3	PDI[25]/TX_D(3)[0]	BD	$V_{CC\ IO\ PDI}$		
A2	PDI[26]/TX_ENA(3)/FX[3]	BD/LO+	$V_{CC\ IO\ PDI}$	FX[3]	
A1	PDI[27]/RX_DV(3)	BD/LI-	$V_{CC\ IO\ PDI}$		
B2	PDI[28]/LED_PERR(3)/TRANS(3)/PHY_RES(3)/ RES_CONF	BD	$V_{CC\_IO\_PDI}$	RES_CONF	

Pin	Pin name	Dir.	Voltage reference	Strap signal	Internal PU/PD
B1	PDI[29]/RX_D(3)[1]	BD/LI+	$V_{CC\_IO\_PDI}$		
C2	PDI[30]/LED_LINKACT(3)/P_CONF[3]	BD	$V_{CC\_IO\_PDI}$	P_CONF[3]	
C1	PDI[31]/CLK25OUT2	BD	$V_{CC\_IO\_PDI}$		
C8	TX_CLK(3)	BD	$V_{CC\_IO\_PDI}$		
	PDI 32-45, physical port 2				
D1	PDI[32]/TX_D(2)[3]	BD	$V_{CC\_IO\_PDI}$		
D2	PDI[33]/TX_D(2)[2]	BD	$V_{CC\_IO\_PDI}$		
E2	PDI[34]/TX_D(2)[0]/CTRL_STATUS_MOVE	BD	$V_{CC\_IO\_PDI}$	CTRL_STATUS_MOVE	
G1	PDI[35]/RX_ERR(2)	BD	$V_{CC\_IO\_PDI}$		
G2	PDI[36]/RX_CLK(2)	BD	$V_{CC\_IO\_PDI}$		
H2	PDI[37]/RX_D(2)[0]	BD	$V_{CC\_IO\_PDI}$		
J2	PDI[38]/RX_D(2)[2]	BD	$V_{CC\_IO\_PDI}$		
K1	PDI[39]/RX_D(2)[3]	BD	$V_{CC\_IO\_PDI}$		
F1	PDI[40]/TX_ENA(2)/FX[2]	BD/LO+	$V_{CC\_IO\_PDI}$	FX[2]	
E1	PDI[41]/TX_D(2)[1]/ RGMII[2]	O/LO-	$V_{CC\_IO\_PDI}$	RGMII[2]	
H1	PDI[42]/RX_DV(2)	I/LI-	$V_{CC\_IO\_PDI}$		
J1	PDI[43]/RX_D(2)[1]	I/LI+	$V_{CC\_IO\_PDI}$		
C3	PDI[44]/LED_PERR(2)/TRANS(2)/PHY_RES(2)/ PHYAD_OFF[4]	BD	$V_{CC\_IO\_PDI}$	PHYAD_OFF[4]	
E3	PDI[45]/LED_LINKACT(2)/P_CONF[2]	BD	$V_{CC\_IO\_PDI}$	P_CONF[2]	
F2	LINK_MII(2)/CLK25OUT1	BD	$V_{CC\_IO\_PDI}$		
G3	TX_CLK(2)	BD	$V_{CC\_IO\_PDI}$		
	Physical port 1				
L1	LED_LINKACT(1)/P_CONF[1]	BD	$V_{CC\_IO\_COM}$	P_CONF[1]	SPD
K2	LED_PERR(1)/TRANS(1)/ PHY_RES(2)/CLK_MODE[1]	BD	$V_{CC\_IO\_COM}$	CLK_MODE[1]	SPD
K3	LINK_MII(1)	I	$V_{CC\_IO\_COM}$		
K4	RX_CLK(1)	I	$V_{CC\_IO\_COM}$		
L4	RX_D(1)[0]	I	$V_{CC\_IO\_COM}$		
M5	RX_D(1)[1]	I/LI+	$V_{CC\_IO\_COM}$		
L5	RX_D(1)[2]	I	$V_{CC\_IO\_COM}$		
M6	RX_D(1)[3]	I	$V_{CC\_IO\_COM}$		
M4	RX_DV(1)	I/LI-	$V_{CC\_IO\_COM}$		
L6	RX_ERR(1)	I	$V_{CC\_IO\_COM}$		
G4	TX_CLK(1)	BD	$V_{CC\_IO\_COM}$		
L3	TX_D(1)[0]/TRANS_MODE_ENA	BD	$V_{CC\_IO\_COM}$	TRANS_MODE_ENA	SPD
M2	TX_D(1)[1]/RGMII[1]	O/LO-	$V_{CC\_IO\_COM}$	RGMII[1]	SPD
L2	TX_D(1)[2]/P_MODE[0]	BD	$V_{CC\_IO\_COM}$	P_MODE[0]	SPD
M1	TX_D(1)[3]/P_MODE[1]	BD	$V_{CC\_IO\_COM}$	P_MODE[1]	SPD
M3	TX_ENA(1)/FX[1]	BD/LO+	$V_{CC\_IO\_COM}$	FX[1]	SPD
	Physical port 0				
J12	LED_LINKACT(0)/P_CONF[0]	BD	$V_{CC\_IO\_COM}$	P_CONF[0]	SPD
J11	LED_PERR(0)/TRANS(0)/ PHY_RES(2)/CLK_MODE[0]	BD	$V_{CC\_IO\_COM}$	CLK_MODE[0]	SPD
L9	LINK_MII(0)	I	$V_{CC\_IO\_COM}$		
L10	RX_CLK(0)	I	$V_{CC\_IO\_COM}$		
K10	RX_D(0)[0]	I	$V_{CC\_IO\_COM}$		
M12	RX_D(0)[1]	I/LI+	$V_{CC\_IO\_COM}$		
L11	RX_D(0)[2]	I	$V_{CC\_IO\_COM}$		
L12	RX_D(0)[3]	I	$V_{CC\_IO\_COM}$		
M11	RX_DV(0)	I/LI-	$V_{CC\_IO\_COM}$		
M10	RX_ERR(0)	I	$V_{CC\_IO\_COM}$		
K9	TX_CLK(0)	BD	$V_{CC\_IO\_COM}$		
L8	TX_D(0)[0]/C25_ENA	BD	$V_{CC\_IO\_COM}$	C25_ENA	SPD
M8	TX_D(0)[1]/RGMII[0]	O/LO-	$V_{CC\_IO\_COM}$	RGMII[0]	SPD
L7	TX_D(0)[2]/C25_SHI[0]	BD	$V_{CC\_IO\_COM}$	C25_SHI[0]	SPD
M7	TX_D(0)[3]/C25_SHI[1]	BD	$V_{CC\_IO\_COM}$		
M9	TX_ENA(0)/FX[0]	BD/LO+	$V_{CC\_IO\_COM}$	FX[0]	SPD
	JTAG				
E4	JTAG_TCK	I	$V_{CC\_IO\_COM}$		
H9	JTAG_TDI	I	$V_{CC\_IO\_COM}$		
F4	JTAG_TDO	O	$V_{CC\_IO\_COM}$		
F3	JTAG_TMS	I	$V_{CC\_IO\_COM}$		
	Clock and reset				
G12	OSC_IN	I	$V_{CC\_REG\_IN}$		

Pin	Pin name	Dir.	Voltage reference	Strap signal	Internal PU/PD
F12	OSC_OUT	O	$V_{CC\_REG\_IN}$		
H12	RESET	BD	$V_{CC\_IO\_COM}$		3.3 kΩ PU
	Power supply configuration				
H4	VCC_CONF[0]	I3	$V_{CC\_REG\_IN}$	VCC_CONF[0]	
H3	VCC_CONF[1]	I3	$V_{CC\_REG\_IN}$	VCC_CONF[1]	
	Regulator supply				
E9	$V_{CC\_REG\_IN}$				
C7	$V_{CC\_REG\_OUT}$				
D7	GND_{REG}				
	core supply				
C6	V_{CC_CORE}				
K6	V_{CC_CORE}				
K7	$V_{CC\_CORE\_ANA}$				
D6	GND_{CORE}				
J6	GND_{CORE}				
J7	GND_{CORE_ANA}				
	PLL supply				
G10	V_{CC_PLL}				
G9	GND_{PLL}				
	IO COM supply				
J3	$V_{CC\_IO\_COM}$				
J10	$V_{CC\_IO\_COM}$				
K5	$V_{CC\_IO\_COM}$				
K8	$V_{CC\_IO\_COM}$				
J4	GND_{IO_COM}				
J5	GND_{IO_COM}				
J8	GND_{IO_COM}				
J9	GND_{IO_COM}				
	IO PDI supply				
C5	$V_{CC\_IO\_PDI}$				
D3	$V_{CC\_IO\_PDI}$				
D10	$V_{CC\_IO\_PDI}$				
F10	$V_{CC\_IO\_PDI}$				
D4	GND_{IO_PDI}				
D5	GND_{IO_PDI}				
D9	GND_{IO_PDI}				
F9	GND_{IO_PDI}				

Table 10: Pin legend

Dir.	Description
BD	Bidirectional signal
O	Output
I	Input
I3	Input with 3 level detection (GND, V_{CC_REG}, open)
PU	Pull-up resistor
PD	Pull-down resistor
SPD	Pull-down resistor active during strapping (weak)
WPU	Pull-up resistor (weak)
WPD	Pull-down resistor (weak)

4 Signal description

4.1 Signal overview

Table 11: 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_OUT1 configuration
CLK25OUT1/CLK25OUT2	MII/RGMII	O	25 MHz clock source for Ethernet PHYs
CPU_CLK_OUT1	PDI	O	Clock signal for µController
CTRL_STATUS_MOVE	Configuration	I	Move digital I/O control/status signal to last available PDI byte
EEPROM_CLK	EEPROM	BD	EEPROM I^2C clock
EEPROM_DATA	EEPROM	BD	EEPROM I^2C data
EEPROM_SIZE	Configuration	I	EEPROM size configuration
EE_EMU_ENA	Configuration	I	EEPROM emulation enable
EE_EMU[2:0]	Configuration	I	EEPROM emulation configuration
FX(3:0)	Configuration	I	FX mode
GND_{CORE}	Supply		Core logic ground
GND_{CORE_ANA}	Supply		Core logic ground (regulator feedback)
GND_{IO_COM}	Supply		Communication I/O ground
GND_{IO_PDI}	Supply		PDI I/O ground
GND_{PLL}	Supply		PLL ground
GND_{REG}	Supply		LDO/DC-DC regulator ground
LED_ERR	LED	O	EtherCAT ERROR LED
LED_LINKACT(3:0)	LED	O	Port link/activity LED output
LED_PDI0_ERR	LED	O	PDI0 error LED output (for testing)
LED_PDI1_ERR	LED	O	PDI1 error LED output (for testing)
LED_PERR(3:0)	LED	O	Port receive error LED output (for testing)
LED_RUN	LED	O	EtherCAT RUN LED
LED_STATE_RUN	LED	O	Application signal for combined bicolor EtherCAT STATUSLEDs
LINK_MII(3:0)	MII	I	PHY signal indicating a link
LINKPOL	Configuration	I	LINK_MII(3:0) polarity configuration
RES_CONF	Configuration	I	Reserved configuration bit
MI_CLK	MII/RGMII	O	PHY management interface clock
MI_DATA	MII/RGMII	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[45:0]	PDI	BD	PDI signals
PHYAD_OFF[4, 0]	Configuration	I	Ethernet PHY address offset bit 4 and bit 0
PHY_RES	MII/RGMII	O	PHY reset, when FX mode is enabled
POR_ET1100	Configuration	I	Disable ET1100 compatible POR strapping
RESET	General	BD	Open collector reset output/reset input
RGMII(3:0)	Configuration	I	RGMII mode
RX_CLK(3:0)	MII/RGMII	I	MII/RGMII receive clock
RX_CTL(3:0)	RGMII	I	RGMII receive control
RX_D(3:0)[3:0]	MII/RGMII	I	MII/RGMII receive data
RX_DV(3:0)	MII	I	MII receive data valid
RX_ERR(3:0)	MII	I	MII receive error
DC_SYNC/LATCH[1:0]	DC	I/O	Distributed clocks SyncSignal output or LatchSignal input
JTAG_TCK	JTAG	I	JTAG clock
JTAG_TDI	JTAG	I	JTAG input
JTAG_TDO	JTAG	O	JTAG output
JTAG_TMS	JTAG	I	JTAG select
TRANS(3:0)	MII	I	MII/RGMII interface sharing: share port enable
TRANS_MODE_ENA	Configuration	I	Enable MII/RGMII interface sharing (and TRANS(3:0) signals)
TX_CLK(3:0)	MII/RGMII	I/O	MII/RGMII TX_CLK
TX_D(3:0)[3:0]	MII/RGMII	O	MII/RGMII transmit data
TX_CTL(3:0)	RGMII	O	RGMII transmit control
TX_ENA(3:0)	MII	O	MII transmit enable
V_{CC_CORE}	Supply	I	Core logic supply
$V_{CC\_CORE\_ANA}$	Supply	I	Core logic supply (regulator feedback)
$V_{CC\_IO\_COM}$	Supply	I	Communication I/O supply

Signal	Type	Dir.	Description
$V_{CC\ IO\ PDI}$	Supply	I	PDI I/O supply
$V_{CC\ REG\ IN}$	Supply	I	LDO/DC-DC regulator input
$V_{CC\ REG\ OUT}$	Supply	O	LDO/DC-DC regulator output
$V_{CC\ PLL}$	Supply	I	PLL supply
VCC_CONF[1:0]	Configuration	I3	LDO/DC-DC and IO voltage configuration

4.2 PDI signal overview

Table 12: 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
	OUT_START	I	Start output cycle
	OUTVALID	O	Output data is valid/output event
	SOF	O	Start of frame
	WD_TRIG	O	Watchdog trigger
SPI slave	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	SPI_CLK	I	SPI clock
	SPI_MOSI[7:0]	I	Unidirectional SPI data MOSI
	SPI_MISO[7:0]	O	Unidirectional SPI data MISO
	SPI_MISO_ENA	O	Bidirectional SPI data MISO driver active
	SPI_D[7:0]	BD	Bidirectional SPI data
	SPI_IRQ	O	SPI interrupt
	SPI_SEL	I	SPI general chip select
	SPI_SEL_MOSI	I	SPI MOSI chip select
	SPI_SEL_MISO	I	SPI MISO chip select
SPI master	SPI_CLK	O	SPI clock
	SPI_MOSI[7:0]	O	Unidirectional SPI data MOSI
	SPI_MISO[7:0]	I	Unidirectional SPI data MISO
	SPI_MISO_VALID	I	MISO data valid status
	SPI_MOSI_ENA	O	Bidirectional SPI data MOSI driver active
	SPI_D[7:0]	BD	Bidirectional SPI data
	SPI_SEL	I	SPI general chip select
	SPI_SEL_MOSI	I	SPI MOSI chip select
	SPI_SEL_MISO	I	SPI MISO chip select
	SPI_START_MOSI	I	Start MOSI cycle
	SPI_START_MISO	I	Start MISO cycle
	SPI_WD_TRIGGER	O	Watchdog trigger
µC async.	ADR[15:0]	I	Address bus
	BHE	I	Byte high enable (16 bit µController interface only)
	BUSY	O	EtherCAT device is busy
	CS	I	Chip select
	DATA[15:0]	BD	Data bus, 8 bit or 16 bit used
	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	IRQ	O	Interrupt
	RD	I	Read command
	WR	I	Write command

PDI	Signal	Dir.	Description
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, 8 bit or 16 bit used
	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
µC async. mux.	A[15:8]	I	Address bus high byte for 8 bit µC interface
	AD[31:0]	BD	Address and data bus. 16 bit used for address, 8/16/32 bit used for data
	ALE	I	Address latch enable
	BE[3:0]	I	Byte enable. 1/2/4 bit used for 8/16/32 bit µC interface
	BUSY	O	EtherCAT device is busy
	CS	I	Chip select
	EEPROM_LOADED	O	PDI is active, EEPROM is loaded
	IRQ	O	Interrupt
	RD	I	Read command
	WR	I	Write command
µC sync.	ADR[15:0]	I	Address bus
	CPU_RESET	O	Reset signal for µController
	DC_SYNC[3:0]	O	DC SyncSignals
	DC_LATCH[3:0]	I	DC LatchSignals
	GPIO	BD	General purpose input/output
	GPI	I	General purpose input
	GPO	O	General purpose output

4.3 Power supply

The ET1150 supports different power supply voltages, but not all combinations are valid:

- $V_{CC\_REG\_IN}$ supports 3.3V or 2.5V
- $V_{CC\_IO\_COM}$ supports 3.3V, 2.5V, or 1.8V
- $V_{CC\_IO\_PDI}$ supports 3.3V, 2.5V, or 1.8V
- V_{CC_CORE}/V_{CC_PLL} is 1.1V, and it can be generated using the internal LDO, using the internal DC-DC regulator, or they can be supplied externally

Supply voltages are monitored internally. Therefore, the I/O voltage level must be configured properly using VCC_CONF[1:0]. These pins also select LDO or DC-DC operation.

Table 13: Power supply combinations (nominal voltages)

$V_{CC\_REG\_IN}$	VCC_CONF		LDO/ DC-DC	$V_{CC\_IO\_COM}$	$V_{CC\_IO\_PDI}$	Reg. 0x0E02	Comment
	[1]	[0]				[7:4]	
3.3V max							
3.3V	GND_{REG}	GND_{REG}	LDO	3.3V	3.3V	0000	Single supply, compatible with ET1100
3.3V	GND_{REG}	$V_{CC\_REG\_IN}$	LDO	2.5V	2.5V	0001	
3.3V	GND_{REG}	open	LDO	3.3V	2.5V	0010	
3.3V	open	GND_{REG}	LDO	3.3V	1.8V	1000	
3.3V	open	$V_{CC\_REG\_IN}$	LDO	2.5V	1.8V	1001	
3.3V	open	open	LDO	1.8V	1.8V	1010	
3.3V	$V_{CC\_REG\_IN}$	GND_{REG}	DC-DC	3.3V	3.3V	0100	Single supply
3.3V	$V_{CC\_REG\_IN}$	$V_{CC\_REG\_IN}$	DC-DC	2.5V	2.5V	0101	
3.3V	$V_{CC\_REG\_IN}$	open	DC-DC	3.3V	2.5V	0110	
2.5V max							
2.5V	GND_{REG}	$V_{CC\_REG\_IN}$	LDO	2.5V	2.5V	0001	Single supply
2.5V	open	$V_{CC\_REG\_IN}$	LDO	2.5V	1.8V	1001	
2.5V	open	open	LDO	1.8V	1.8V	1010	
2.5V	$V_{CC\_REG\_IN}$	$V_{CC\_REG\_IN}$	DC-DC	2.5V	2.5V	0101	Single supply

The $V_{CC\_IO\_COM}$/$V_{CC\_IO\_PDI}$ supply voltages directly determine the I/O voltage for all inputs and outputs of the specific I/O rail. E.g., with $V_{CC\_IO\_COM}$=2.5V, the inputs of the COM rail are 2.5V I/O compliant, but they are not 3.3V tolerant.

The core supply voltages V_{CC_CORE}/V_{CC_PLL} are usually generated by an internal LDO or internal DC-DC regulator – this is recommended for compatibility with future ESC updates. The internal LDO/DC-DC regulator is supplied by $V_{CC\_REG\_IN}$, and it delivers the output voltage for V_{CC_CORE}/V_{CC_PLL} on the $V_{CC\_REG\_OUT}$ pin.

For external supply of V_{CC_CORE}/V_{CC_PLL}, the LDO mode is selected, but $V_{CC\_REG\_OUT}$ is left unused, while $V_{CC\_REG\_IN}$ still must be supplied.

$V_{CC\_CORE\_ANA}$ is treated identical to V_{CC_CORE}, and GND_{CORE_ANA} is treated identically as GND_{CORE}.

All power supply pins must be connected. Voltage stabilization capacitors at all power pairs are necessary. V_{CC_PLL} should be decoupled from V_{CC_CORE}.

Recommendation for voltage stabilization capacitors: 220pF and 100nF ceramic capacitors for each power pin pair, additional 10µF capacitor for each power rail. All GND signals shall be connected to a common GND potential.

4.3.1 LDO operation

The ET1150 has an integrated LDO for generation of V_{CC_CORE}/V_{CC_PLL}. The LDO uses $V_{CC\_CORE\_ANA}$ as feedback signal for voltage regulation.

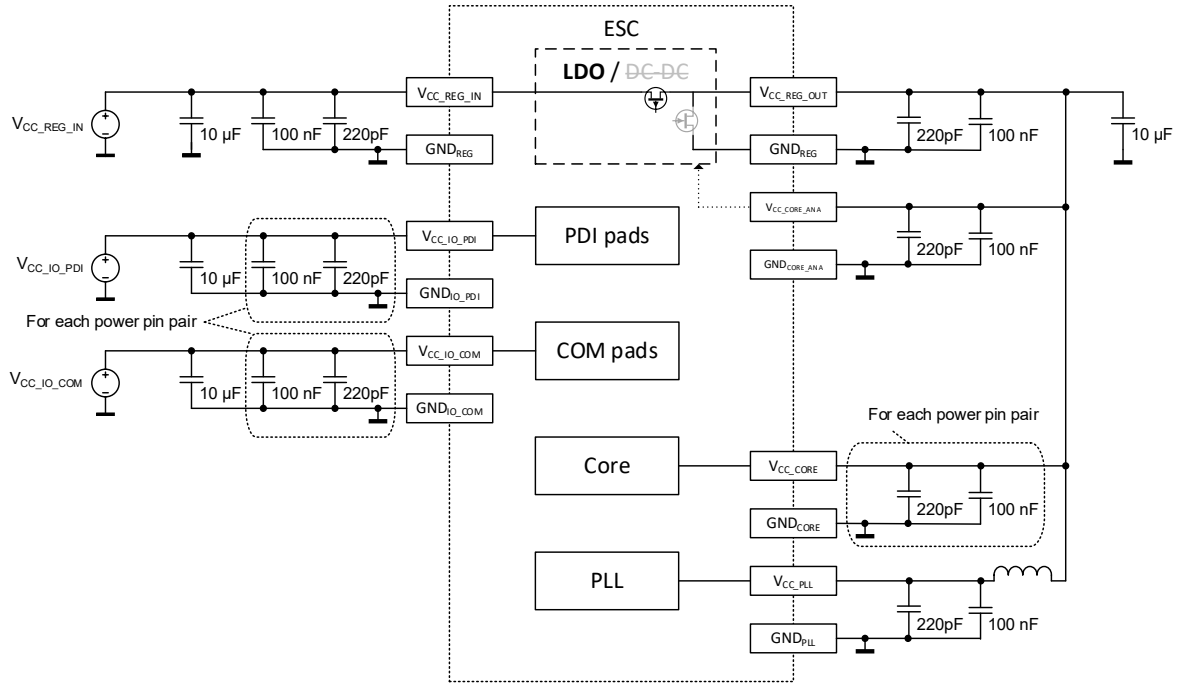

Figure 4: ET1150 power supply LDO operation

4.3.2 DC-DC operation

The ET1150 has an integrated DC-DC regulator, which can be used instead of the LDO for generation of V_{CC_CORE}/V_{CC_PLL}. The DC-DC regulator uses $V_{CC\_CORE\_ANA}$ as feedback signal for voltage regulation. An external inductor is required.

The DC-DC regulator is digitally controlled, with a slow start-up, and overload protection.

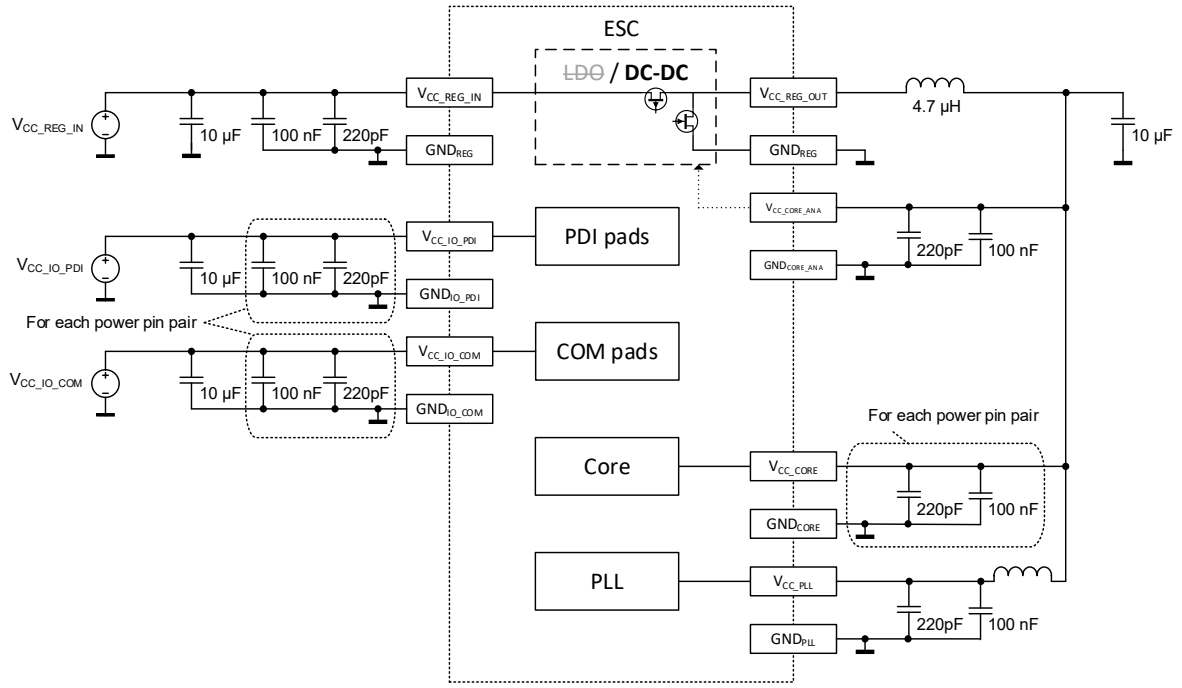

Figure 5: ET1150 power supply DC-DC operation

4.3.3 Electrical characteristics DC-DC regulator

Table 14: DC characteristics ET1150

Symbol	Parameter	Condition	Min	Typ	Max	Units
PRELIMINARY VALUES						
L_{DC-DC}	DC-DC output inductor		2.2	4.7	10	µH
C_{DC-DC}	DC-DC output capacitor			10		µF
ESR_{DC-DC}	DC-DC output capacitor ESR			10		mΩ
$I_{REG\_OUT\_DC-DC}$	DC-DC regulator output current		0	100	200	mA
$V_{REG\_OUT\_DC-DC}$	DC-DC regulator output voltage		0.99	1.1	1.21	V
F_{DC-DC}	DC-DC switching frequency				2	MHz

4.3.4 ET1100 compatible LDO operation

The ET1150 can be used as a replacement for ET1100. On an ET1100 footprint, supply voltages are automatically connected as shown, and VCC_CONF[1:0] are automatically set to GND, as required.

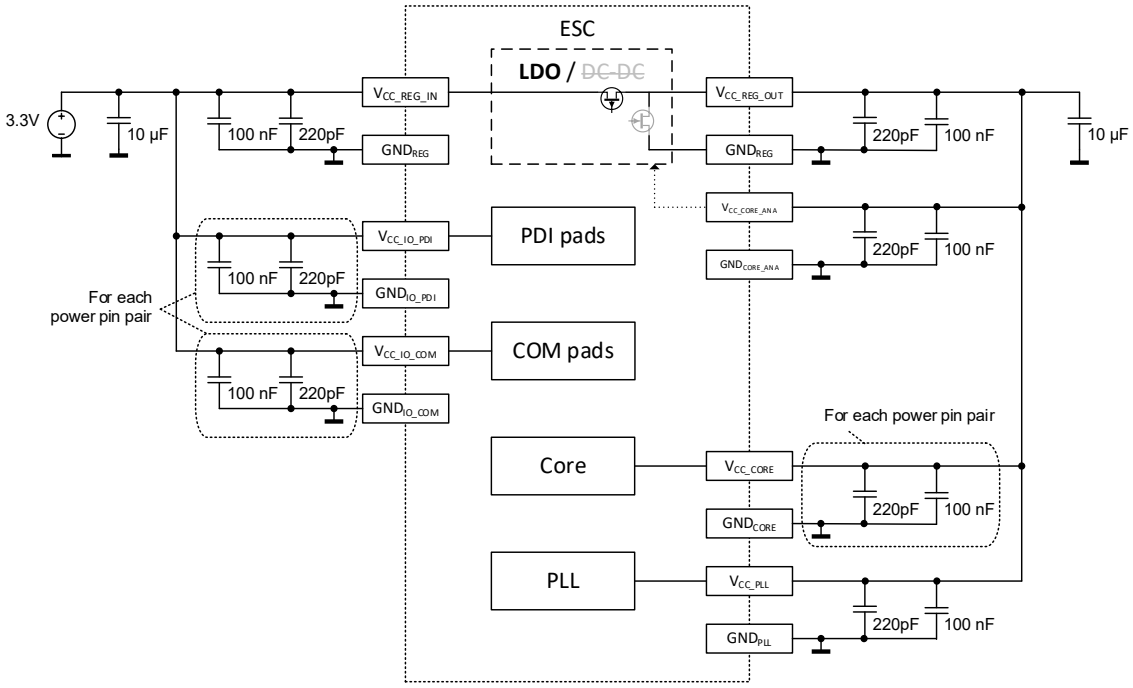

Figure 6: ET1150 power supply for ET1100 footprints

4.3.5 External V_{CC_CORE} supply

External V_{CC_CORE} supply is not recommended, because of potential incompatibility with future ESCs. $V_{CC\_REG\_OUT}$ is left open/unconnected in this case.

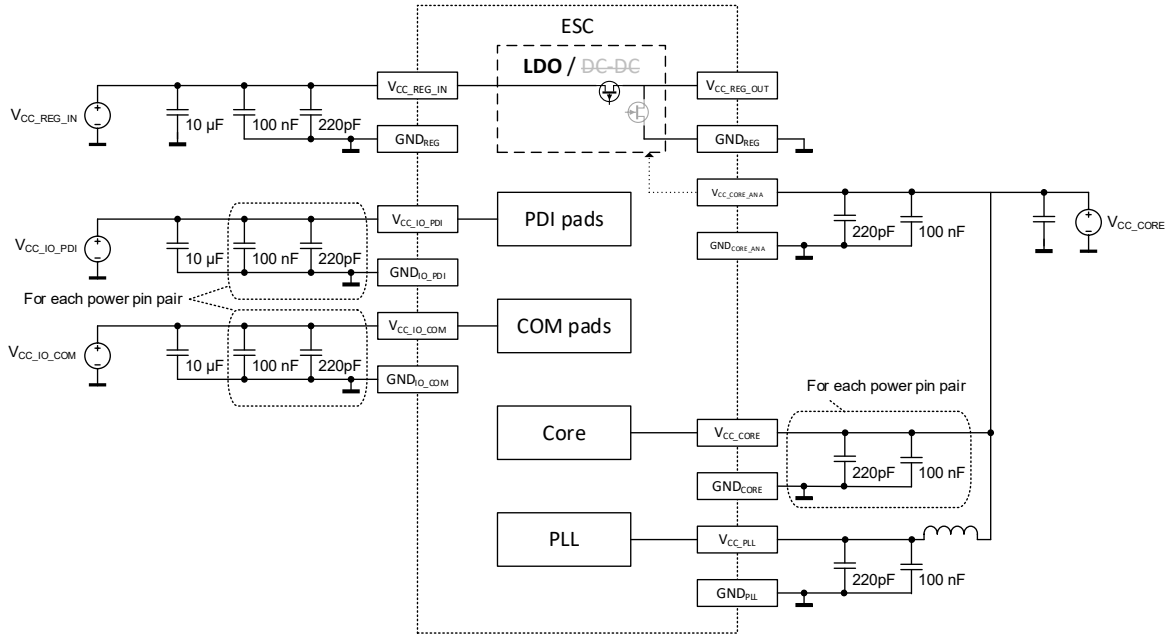

Figure 7: ET1150 power supply with external V_{CC_CORE}

4.4 Clock supply

An external crystal or a clock signal from an external oscillator can be used for the ET1150, both with 25 MHz. The 25 MHz clock source should have an initial accuracy of 25ppm or better (at room temperature).

An oscillator as the clock source for both ET1150 and PHYs is mandatory if MII ports are used and CLK25OUT1/2 are not used as the clock source for the PHYs.

Table 15: Clock supply 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		

OSC_IN
Connection to external crystal or oscillator input (25 MHz).

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

4.4.1 Example schematics for clock supply

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

Although a clock frequency of 25 MHz does not require extensive design efforts, the following rules shall help improve the 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 ET1150 clock source has to be 25ppm or better.

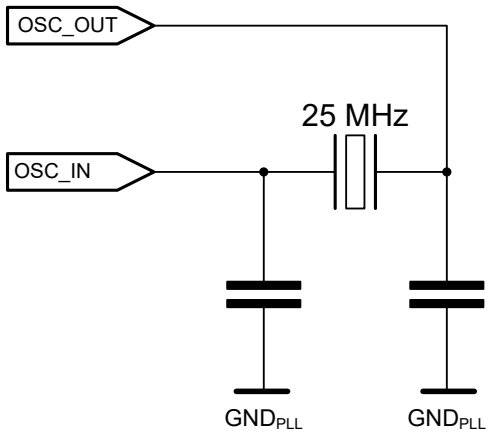

Figure 8: Quartz crystal connection

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).

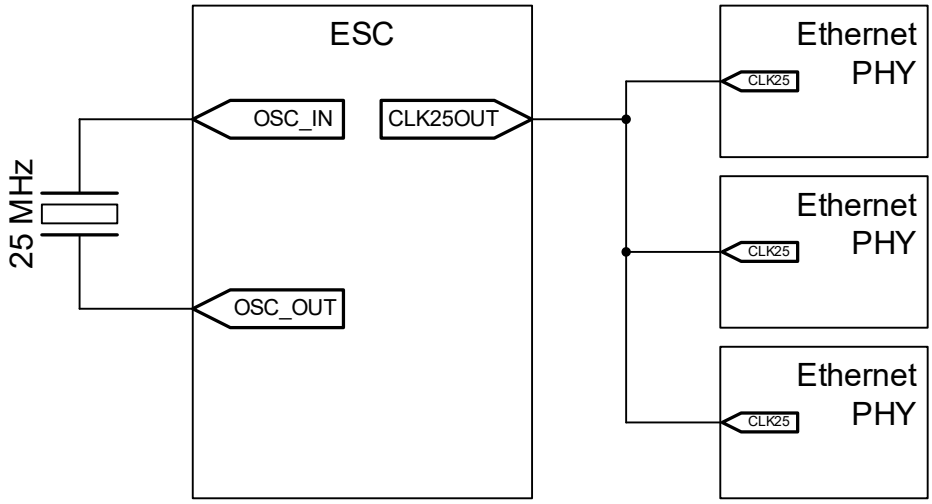

Figure 9: Quartz crystal clock source for ET1150 and Ethernet PHYs

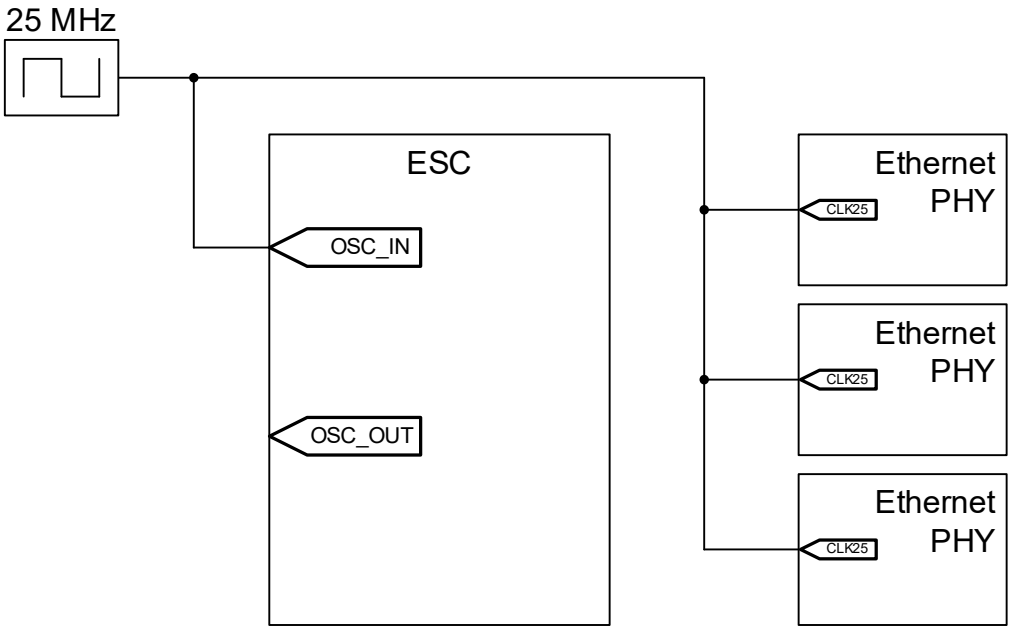

Figure 10: Oscillator clock source for ET1150 and Ethernet PHYs

4.5 RESET pin

Table 16: Reset pin

Pin	Pin		Signal		Configuration signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
H12	RESET	BD	RESET	BD		3.3 kΩ PU

The open collector RESET input/output (active low) signals the reset state of ET1150. The reset state is entered at power-on, if one of the power supplies is too low (V_{CC_REG}; $V_{CC\_IO\_COM}$, $V_{CC\_IO\_PDI}$, V_{CC_CORE}, V_{CC_PLL}), or if a reset was initiated using the reset registers 0x0040/0x0041. ET1150 also enters reset state if RESET pin is held low by external devices.

4.5.1 Internal reset logic and example schematic for RESET pin

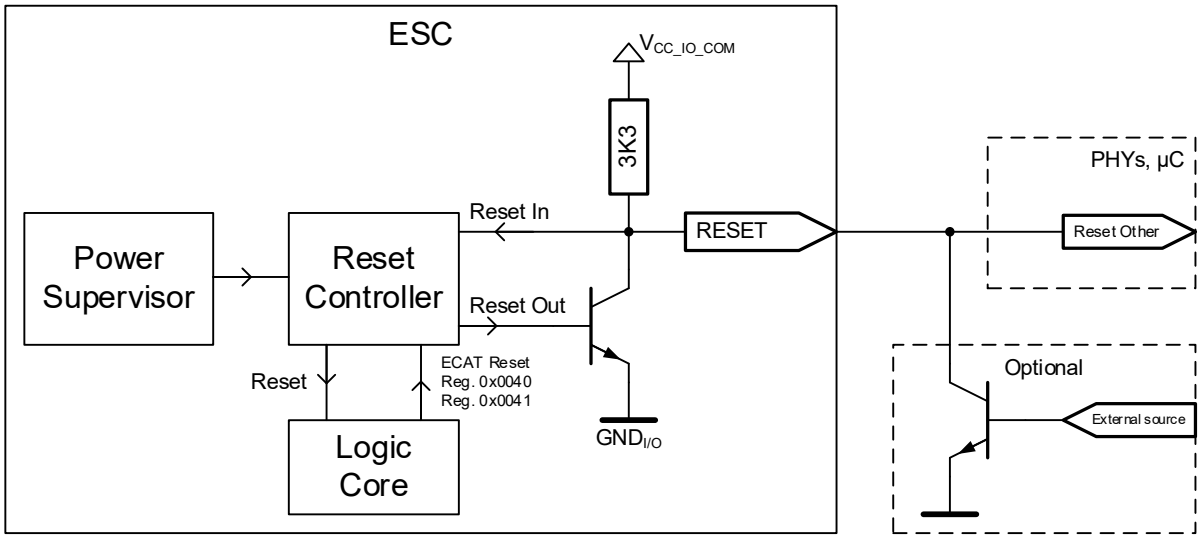

Figure 11: ET1150 reset logic

It is recommended to connect the PHYs and the µController to the RESET pin. This makes sure that the PHYs are not communicating while the ET1150 is in reset (causing lost frames), and it allows for resetting the whole EtherCAT slave device via EtherCAT in case of an exceptional condition.

4.6 Configuration pins

The configuration pins are used to configure the ET1150 at power-on with pull-up or pull-down resistors. At power-on the ET1150 uses these pins as inputs to latch the configuration. 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 ET1150 configuration is not latched again and output drivers remain active.

The configuration value 0 is implemented 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.

Some configuration pins have an integrated strap pull-down resistor (SPD), which is only active during strap phase, it is not enabled during operation.

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 ET1150 power-on value sampling time can be delayed by delaying power activation.

4.6.1 Example schematics for configuration input/LED output pins

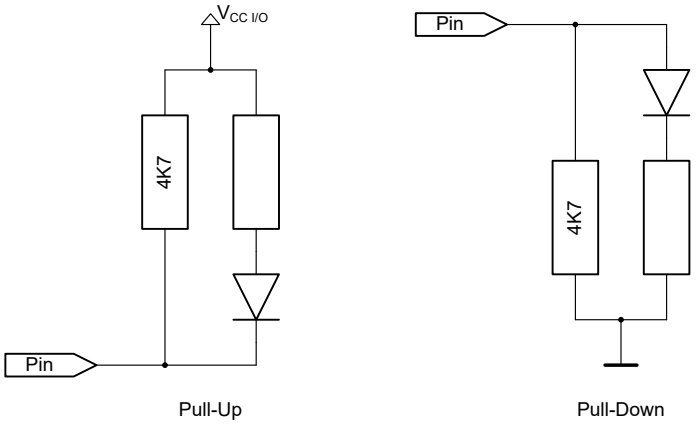

Figure 12: Dual purpose configuration input/LED output pins

4.6.2 Voltage configuration VCC_CONF[1:0]

Please refer to Table 13 for VCC_CONF details.

4.6.3 Port mode

Port mode configures the number of physical/logical ports.

Table 17: Port mode

Pin	Config signal	Pin name	Internal PU/PD	Register	P_MODE[1:0] Values
L2	P_MODE[0]	TX_D(1)[2]/P_MODE[0]	SPD	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)
M1	P_MODE[1]	TX_D(1)[3]/P_MODE[1]	SPD	0x0E00[1]	

4.6.4 Port configuration

P_CONF[3:0] determines the physical layer configuration (MII/RGMII). 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 18: Port configuration

Pin	Configuration signal	Pin name	Internal PU/PD	Register	Values
J12	P_CONF[0]	LED_LINKACT(0)/P_CONF[0]	SPD	0x0E00[2]	
L1	P_CONF[1]	LED_LINKACT(1)/P_CONF(1)	SPD	0x0E00[3]	0 = reserved 1 = MII/RGMII
E3	P_CONF[2]	LED_LINKACT(2)/P_CONF[2]		0x0E00[4]	
C2	P_CONF[3]	PDI[30]/LED_LINKACT(3)/P_CONF(3)		0x0E00[5]	

The term physical port in this document is only used for grouping ET1150 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 MII or RGMII.

The physical port number is the same as the logical port number for P_MODE[1:0]=00, 01 or 11.

4.6.4.1 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 19: Configurations with 2 ports (P_MODE[1:0]=00)

Logical port 1		Logical port 0		P_CONF [3:0]
Type	Physical port	Type	Physical port	
MII/RGMII	1	MII/RGMII	0	-011

4.6.4.2 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 20: Configurations with 3 ports (ports 0,1, and 2; P_MODE[1:0]=01)

Logical port 2		Logical port 1		Logical port 0		P_CONF [3:0]
Type	Physical port	Type	Physical port	Type	Physical port	
MII/RGMII	2	MII/RGMII	1	MII/RGMII	0	-111

Table 21: Configurations with 3 ports (ports 0, 1, and 3; P_MODE[1:0]=10)

Logical port 3		Logical port 1		Logical port 0		P_CONF [3:0]
Type	Physical port	Type	Physical port	Type	Physical port	
MII/RGMII	3	MII/RGMII	1	MII/RGMII	0	-111

4.6.4.3 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 22: Configurations with 4 ports (P_MODE[1:0]=01)

Logical port 3		Logical port 2		Logical port 1		Logical port 0		P_CONF [3:0]
Type	Phys. port	Type	Phys. port	Type	Phys. port	Type	Phys. port	
MII/RGMII	3	MII/RGMII	2	MII/RGMII	1	MII/RGMII	0	1111

4.6.5 RGMII mode

The selection between MII and RGMII operation mode can be made for each physical port individually.

Table 23: RGMII mode

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
M8	RGMII[0]	TX_D(0)[1]/RGMII[0]	SPD	0x0E03[0]	0 = MII 1 = RGMII
M2	RGMII[1]	TX_D(1)[1]/RGMII[1]	SPD	0x0E03[1]	0 = MII 1 = RGMII
E1	RGMII[2]	PDI[41]/TX_D(2)[1]/RGMII[2]		0x0E03[2]	0 = MII 1 = RGMII
A3	RGMII[3]	PDI[24]/TX_D(3)[1]/RGMII[3]		0x0E03[3]	0 = MII 1 = RGMII

4.6.6 FX mode

For MII/RGMII ports, FX operation mode can be selected for each physical port individually, which changes enhanced link detection behavior. LED_PERR(x) signals are replaced by PHY_RES(x) output (active low), when FX mode is selected. Therefore, FX mode cannot be used in combination with transparent mode.

Table 24: FX mode

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
M9	FX[0]	TX_ENA(0)/FX[0]	SPD	0x0E03[4]	0 = TX 1 = FX
M3	FX[1]	TX_ENA(1)/FX[1]	SPD	0x0E03[5]	0 = TX 1 = FX
F1	FX[2]	PDI[40]/TX_ENA(2)/FX[2]		0x0E03[6]	0 = TX 1 = FX
A2	FX[3]	PDI[26]/TX_ENA(3)/FX[3]		0x0E03[7]	0 = TX 1 = FX

4.6.7 CLK25OUT2 enable

A 25MHz clock for Ethernet PHYs can be made available by the ET1150 on PDI[31]/CLK25OUT2 pin. This is only relevant if three MII/RGMII ports are used. In cases with less than 3 MII/RGMII ports, pin LINK_MII(2)/CLK25OUT1 provides CLK25OUT anyway, because LINK_MII(2) is not used. If 4 MII/RGMII ports are used, PDI[31]/CLK25OUT2 provides CLK25OUT2 regardless of CLK25OUT2 enable.

Table 25: CLK25OUT2 enable

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
L8	C25_ENA	TX_D(0)[0]/C25_ENA	SPD	0x0E01[2]	0 = disable, PDI[31]/CLK25OUT2 is available for PDI 1 = enable, PDI[31]/CLK25OUT2 is 25 MHz clock output

4.6.8 CPU_CLK MODE

CLK_MODE is used to provide a clock signal to an external microcontroller. If CLK_MODE is not 00, CPU_CLK_OUT1 is available on PDI[7]. Thus this pin is not available for PDI signals anymore.

Table 26: CPU_CLK mode

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
J11	CLK_MODE[0]	LED_PERR(0)/TRANS(0)/ PHY_RES(0)/ CLK_MODE[0]	SPD	0x0E00[6]	00 = off, PDI[7]/CPU_CLK_OUT1 available for PDI 01 = 25 MHz clock output at PDI[7]/CPU_CLK_OUT1 10 = 20 MHz clock output at PDI[7]/CPU_CLK_OUT1 11 = 10 MHz clock output at PDI[7]/CPU_CLK_OUT1
K2	CLK_MODE[1]	LED_PERR(1)/TRANS(1)/ PHY_RES(1)/ CLK_MODE(1)	SPD	0x0E00[7]	

4.6.9 Link polarity

Ethernet PHYs signal a 100 Mbit/s Full (Duplex link) to the ET1150 by asserting LINK_MII(x). Use the LINKPOL config pin to select the desired polarity.

Table 27: Link polarity

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
K11	LINKPOL	MI_CLK/LINKPOL	SPD	0x0E01[6]	0 = LINK_MII(x) is active low 1 = LINK_MII(x) is active high

Link polarity configuration is ignored if transparent mode is enabled (TRANS_MODE_ENA=1), and POR_ET1100 is disabled (POR_ET1100=0). In this case, LINK_MII(x) is active low, regardless of LINKPOL, while 0x0E01[6] shows the configured LINK_POL setting. This behavior is compatible with ET1100.

4.6.10 MII TX shift

It is recommended to enable automatic TX phase shift by connecting TX_CLK to the PHYs. Nevertheless, it is still possible to tune the phase shift of MII TX signals (TX_ENA, TX_D[3:0]) manually (+0/+10/+20/+30ns) using the C25_SHI[x] signals. In that case, it is also recommended to support all C25_SHI[1:0] configurations by hardware options to enable later adjustments, unless automatic TX phase shift is used (TX_CLK connected).

Table 28: TX shift

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
L7	C25_SHI[0]	TX_D(0)[2]/C25_SHI[0]	SPD	0x0E01[0]	00 = MII TX signals not delayed additionally. Configure this setting when automatic TX phase shift is used. 01 = MII TX signals delayed by +10 ns 10 = MII TX signals delayed by +20 ns 11 = MII TX signals delayed by +30 ns
M7	C25_SHI[1]	TX_D(0)[3]/C25_SHI[1]	SPD	0x0E01[1]	

4.6.11 PHY address offset

The ET1150 supports four PHY address offset configurations, either 0, 1, 16, or 17. Two configuration signals are used for bit 0 and bit 4 of the PHY address offset value. Bits 1 to 3 are always zero. Refer to chapter 5.1.1 for details on PHY address configuration.

Table 29: PHY address offset

Pin	Config signal	Pin name	Internal PU/PD	Register	PHYAD_OFF[4:0]
H10	PHYAD_OFF[0]	LED_ERR/LED_PDI1_ERR/ PHYAD_OFF[0]	SPD	0x0E04[2]	00000 = PHY address offset 0 00001 = PHY address offset 1
C3	PHYAD_OFF[4]	PDI[44]/LED_PERR(2)/TRANS(2)/ PHY_RES(2)/PHYAD_OFF[4]		0x0E01[5]	10000 = PHY address offset 16 10001 = PHY address offset 17

4.6.12 Digital I/O PDI control/status move

If more than 2 MII/RGMII 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.

Table 30: Digital control/status move

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
E2	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/RGMII 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/RGMII(2). Digital I/O control/status signals are available in any configuration.

4.6.13 EEPROM emulation enable

EEPROM emulation mode can be used if a non-volatile memory (NVRAM) is attached to the µController. The ESC configuration and the device description can be stored in this NVRAM, an I²C EEPROM chip is no longer required then.

Table 31: EEPROM emulation enable

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
D8	EE_EMU_ENA	LED_STATE_RUN/EE_EMU_ENA	SPD	0x0E02[3]	0 = EEPROM emulation disabled 1 = EEPROM emulation enabled

EEPROM emulation requires a fixed PDI configuration, which is selected using EE_EMU[2:0] configuration signals. If EEPROM emulation is disabled (EE_EMU_ENA=0), these config pins will be ignored and EE_EMU[2:0] will be set to 000 internally.

Table 32: EEPROM emulation PDI selection

Config signal	Register	Value	PDI	PDI type
EE_EMU[2:0]	0x0E02[18:16]	000	SPI	0x05
		001	8 bit async µC	0x09
		010	16 bit async µC	0x08
		011	8 bit sync µC	0x0B
		100	16 bit sync µC	0x0A
		101	8 bit mux async µC	0x0D
		110	16 bit mux async µC	0x0C
		111	32 bit mux async µC	0x0D

Table 33: EEPROM emulation PDI configuration

Value	PDI	PDI type	PDI configuration 0x0150		PDI configuration 0x0153:0x0152	
			Value	Mask	Value	Mask
000	SPI	0x05	0x07	0x33	0x0000	0x0000
001	8 bit async µC	0x09	0x05	0xF0	0x0001	0x0F0C
010	16 bit async µC	0x08	0x05	0xF0	0x0001	0x0F0C
011	8 bit sync µC	0x0B	0x05	0xF0	0x0001	0x0F0C
100	16 bit sync µC	0x0A	0x05	0xF0	0x0001	0x0F0C
101	8 bit mux async µC	0x0D	0x05	0xF0	0x0001	0x0F0C
110	16 bit mux async µC	0x0C	0x05	0xF0	0x0001	0x0F0C
111	32 bit mux async µC	0x0D	0x05	0xF0	0x0009	0x0F0C

NOTE: Mask bit = 0: configuration can be changed by loaded values
Mask bit = 1: configuration cannot be changed by loaded values

4.6.14 EEPROM emulation configuration EE_EMU[1:0]

EE_EMU[1:0] determines the PDI selection for EEPROM emulation, in combination with EE_EMU[2]. Since a pull-up resistor is active on these pins during strapping, EE_EMU[1:0] are configured by direct connections to GND or $V_{CC\_IO\_COM}$. The pull-up resistors are disabled after strapping, if EE_EMU_ENA=1.

Table 34: EEPROM emulation configuration EE_EMU[0]

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
G11	EE_EMU[0]	EEPROM_CLK/EE_EMU[0]	PU 3K3 if EE_EMU_ENA=0	0x0E02[0]	Refer to EE_EMU_ENA for details
F11	EE_EMU[1]	EEPROM_DATA/EE_EMU[1]	PU 3K3 if EE_EMU_ENA=0	0x0E02[1]	Refer to EE_EMU_ENA for details

4.6.15 EEPROM size or EEPROM emulation configuration EE_EMU[2]

The pin LED_RUN/EEPROM_SIZE/LED_PDI0_ERR/EE_EMU[2] is used to configure either EEPROM_SIZE or EE_EMU[2], depending on EEPROM emulation configuration.

EEPROM_SIZE determines the size of the EEPROM (and the number of I²C address bytes, either 1 Kbit-16 Kbit or 32 KBit-4 Mbit). EEPROM_SIZE is sampled during operation, before the initial EEPROM configuration area load access, not during power-on configuration – that is later than the other config signals.

EE_EMU[2] determines the PDI selection for EEPROM emulation, EE_EMU[2] is sampled at the same time as the other config signals.

Table 35: EEPROM emulation configuration EE_EMU[1:0]

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
EEPROM emulation disabled (EE_EMU_ENA=0):					
H11	EEPROM_SIZE	LED_RUN/EEPROM_SIZE/ LED_PDI0_ERR/EE_EMU[2]		0x0502[7]	0 = 1 address byte (1 Kbit to 16 Kbit EEPROM) 1 = 2 address bytes (32 Kbit to 4 Mbit EEPROM)
EEPROM emulation enabled (EE_EMU_ENA=1):					
H11	EE_EMU[2]	LED_RUN/EEPROM_SIZE/ LED_PDI0_ERR/EE_EMU[2]	SPD	0x0E02[2]	Refer to EE_EMU_ENA for details

NOTE: EEPROM_SIZE has a strap pull-down active even when EE_EMU_ENA=0, but this strap pull-down is already disabled when EEPROM_SIZE is sampled – so an external pull-down resistor has to be used for EEPROM_SIZE=0 .

4.6.16 Transparent mode

The ET1150 is capable of sharing the MII/RGMII interfaces with other MACs on a per port basis. Typically, the Transparent mode is disabled, and the ET1150 has exclusive access to the MII/RGMII interfaces of the PHYs. With the Transparent mode turned on, the MII/RGMII interfaces can be assigned either to the ET1150 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 LED_PERR(x)/TRANS(x)/PHY_RES(x) signals. If Transparent mode is enabled, LED_PERR(x)/TRANS(x)/PHY_RES(x) becomes TRANS(x) (active low), which controls the transparent state of each port. LED_PERR(x) is not available in Transparent mode or in FX 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 ET1150 and the other MAC.

Each MII/RGMII interface behaves as usual as long as TRANS(x) is high, and the ET1150 controls the MII/RGMII interface. If TRANS(x) is low, the port becomes transparent (or isolated), i.e., the ET1150 will no longer drive TX_ENA(x)/TX_D(x)/TX_CTL(x)/TX_CLK(x) actively. Thus, the other MAC can drive these signals.

The link/Act(x) LED will still be driven by the ET1150, 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/RGMII interface is not transparent, the ET1150 is in control of the PHY management interface. With the Transparent mode turned on, the PHY management interface of the ET1150 can be accessed via the PDI interface, so a µController gets access to the management interface. If all MII/RGMII interfaces are transparent, the ET1150 releases MI_CLK and MI_DATA drivers, so they can be driven by the other MAC.

Refer to example schematic for more details.

Table 36: Transparent mode enable

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
L3	TRANS_MODE_ENA	TX_D(1)[0]/ TRANS_MODE_ENA	SPD	0x0E01[3]	0 = normal mode/Transparent mode disabled. ET1150 uses PHY exclusively 1 = Transparent mode enabled, ET1150 can share PHY with other MACs

4.6.16.1 Example schematic for transparent mode

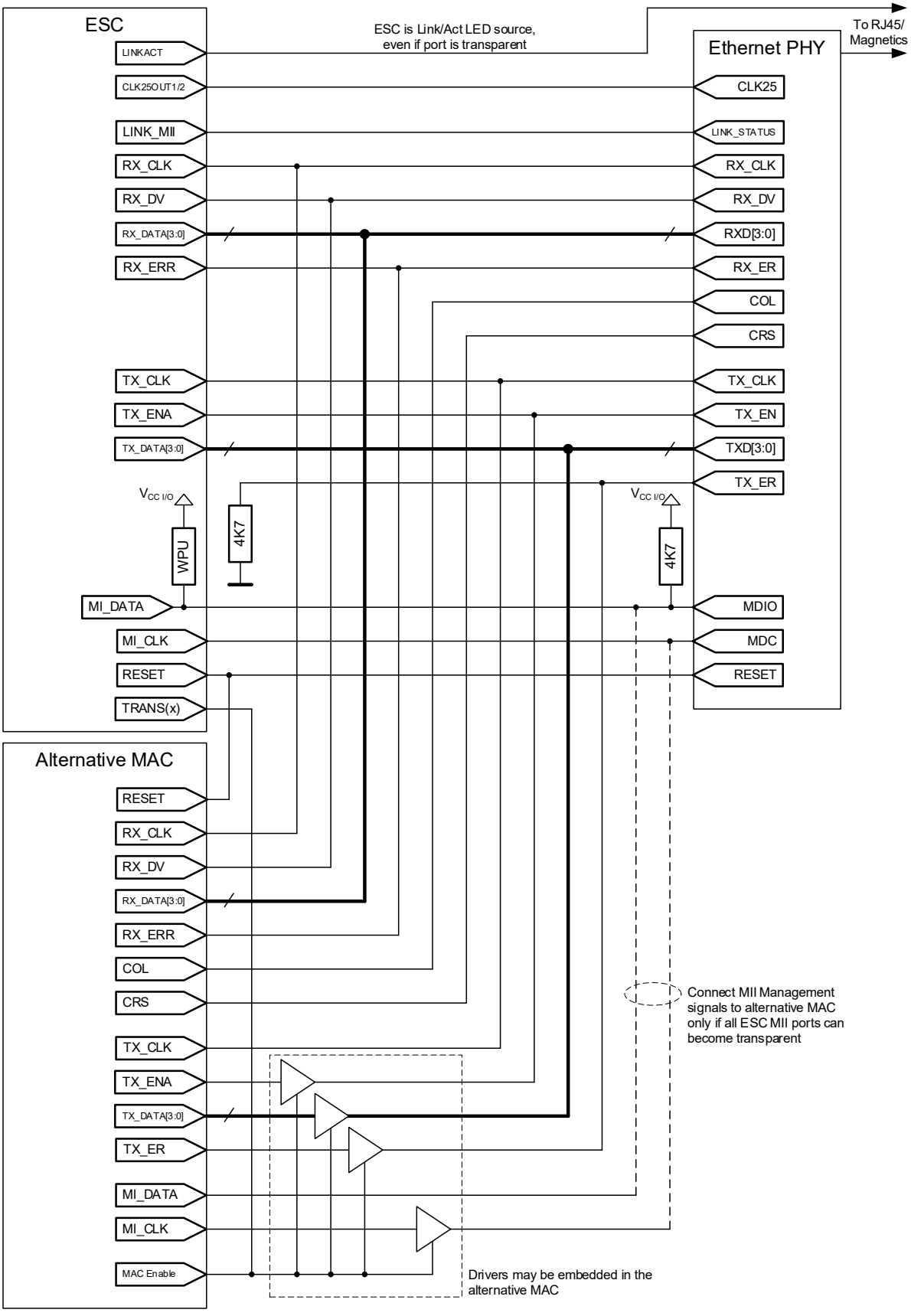

Figure 13: ET1150 transparent mode

NOTE: MI_DATA outputs of alternative MAC have to be high-Z if ET1150 is controlling PHY management interface, otherwise add driver (like MI_CLK). Check alternative MAC's TX timings when extra drivers are used.

4.6.17 POR_ET1100 for compatible pinout with ET1100

The configuration signal POR_ET1100 is used to indicate compatible operation with an ET1100:

Table 37: POR_ET1100

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
E10	POR_ET1100	WD_STATE/POR_ET1100	SPD	0x0E04[0]	0 = ET1100 compatible pinout 1 = additional configuration and signals available

On an ET1100 footprint, POR_ET1100 is automatically connected to GND, thus enabling compatible operation. Some strapping options are disabled with POR_ET1100, and some output signals are not activated:

Table 38: POR_ET1100 effect

Config signal/signal	POR_ET1100 = 0	POR_ET1100 = 1
FX_MODE(3:0)	0000	configurable
RGMII(3:0)	0000	configurable
RES_CONF	0	configurable
LINK_POL when TRANS_MODE_ENA=1	0	configurable
LINK_POL when TRANS_MODE_ENA=0	configurable	configurable
LED_ERR/LED_PDI1_ERR	Not available, driver disabled	Available, driver enabled
LED_STATE_RUN	Not available, driver disabled	Available, driver enabled
WD_STATE	Not available, driver disabled	Available, driver enabled

4.6.18 Reserved configuration pin

The reserved configuration pin should be pulled down when 4 ports are used. Otherwise it should be left open.

Table 39: Reserved

Pin	Config signal	Pin name	Internal PU/PD	Register	Values
B2	RES_CONF	PDI[28]/LED_PERR(3)/ TRANS(3)/PHY_RES(3)/ RES_CONF		0x0E01[7]	0 for 4 port configurations

4.7 SII EEPROM interface signals

Table 40: SII EEPROM pins

Pin	Pin		Signal		Configuration signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
G11	EEPROM_CLK/EE_EMU[0]	BD	EEPROM_CLK	BD		3.3 kΩ PU
F11	EEPROM_DATA/EE_EMU[1]	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).

4.8 PHY management interface signals

The PHY management signals are only used if at least one MII/RGMII port is configured.

Table 41: PHY management pins

Pin	Pin		No MII/RGMII port used		MII/RGMII 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 (4.7 kΩ recommended for ESCs).

4.9 Distributed clocks DC_SYNC/LATCH[1:0] signals

Table 42: DC SYNC/LATCH pins

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

DC_SYNC/DC_LATCH[1:0]
Distributed clocks SyncSignal[1:0] output or LatchSignal[1:0] input, depending on SII EEPROM configuration. DC_SYNC/LATCH signals are not driven (high impedance) until the EEPROM is loaded.
Additional DC_SYNC/LATCH can be made available as part of the DC signal block. Refer to chapter 4.14.4 for details.

4.10 LED signals

Most 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 chapter 4.6.1 for LED connection details.

Table 43: LED pins

Pin	Pin		LED signal	
	Name	Dir.	Signal	Dir.
H11	LED_RUN/EEPROM_SIZE/LED_PDI0_ERR/EE_EMU[2	BD	LED_RUN/LED_PDI0_ERR	O
H10	LED_ERR/LED_PDI1_ERR/PHYAD_OFF[0]	BD	LED_ERR/ LED_PDI1_ERR	O
D8	LED_STATE_RUN/EE_EMU_ENA	BD	LED_STATE_RUN	O
K2	LED_LINKACT(0)/P_CONF[0]	BD	LED_LINKACT(0)	O
L1	LED_LINKACT(1)/P_CONF[1]	BD	LED_LINKACT(1)	O
E3	PDI[45]/LED_LINKACT(2)/P_CONF[2]	BD	LED_LINKACT(2)	O
C2	PDI[30]/LED_LINKACT(3)/P_CONF[3]	BD	LED_LINKACT(3)	O
J11	LED_PERR(0)/TRANS(0)/PHY_RES(0)/CLK_MODE[0]	BD	LED_PERR(0)	O
J12	LED_PERR(1)/TRANS(1)/PHY_RES(1)/CLK_MODE[1]	BD	LED_PERR(1)	O
C3	PDI[44]/LED_PERR(2)/TRANS(2)/PHY_RES(2)/PHYAD_OFF[4]	BD	LED_PERR(2)	O
B2	PDI[28]/LED_PERR(3)/TRANS(3)/PHY_RES(3)/RES_CONF	BD	LED_PERR(3)	O

LED_RUN
EtherCAT RUN LED signal. RUN LED shall be green. The RUN LED is automatically controlled by the ESC, based on AL status register 0x0130:0x0131, RUN LED override register 0x0138, and EEPROM loading state.
For debugging, the LED_RUN signal can be configured to provide the LED_PDI0_ERR signal instead, using ESC configuration B4[11]. In this case, the occurrence of a PDI error (increment of the error counter) at PDI0 is signaled on the LED_RUN pin.

LED_ERR
EtherCAT ERROR LED signal. ERROR LED shall be red. The ERROR LED is automatically controlled by the ESC, based on AL status register 0x0130:0x0131, ERR LED override register 0x0139, and ET1150 error conditions.
For debugging, the LED_ERR signal can be configured to provide the LED_PDI1_ERR signal instead, using ESC configuration B4[11]. In this case, the occurrence of a PDI error (increment of the error counter) at PDI 1 is signaled on the LED_ERR pin.

LED_STATE_RUN
Signal for EtherCAT STATUS LED, a bicolor LED combining RUN (green) and ERROR (red). The bicolor LED is connected to LED_STATE_RUN and LED_ERR signals. The difference between LED_STATE_RUN and LED_RUN is that LED_STATE_RUN is off while LED_ERR is on.

LED_LINKACT(x)
EtherCAT link/activity LED output (off=no link, on=link without activity, blinking=link and activity) for physical port x. Link/activity LED shall be green.

LED_PERR(x)/TRANS(x)
Port error LED output of physical port x for MII/RGMII ports if TRANS_MODE_ENA=0 and FX(x)=0. If TRANS_MODE_ENA=1, LED_PERR(x)/TRANS(x)/PHY_RES(x) is used as TRANS(x). LED_PERR(x) is not available in this case.
If FX mode is used for a port, LED_PERR(x)/TRANS(x)/PHY_RES(x) becomes a PHY_RES(x) output, LED_PERR(x) is not available in this mode.

NOTE: LED_PERR(x) LEDs are not part of the EtherCAT indicator specification. They are only intended for testing and debugging. The LED_PERR(x) LED flashes once if a physical layer receive error occurs. Do not confuse LED_PERR(x) LEDs with application layer ERROR LED. The LED_PERR flash is too short to be visible, it is intended for oscilloscopes.

4.11 Clock output signals

Table 44: Clock output pins

Pin	Pin		Clock signal		Configuration signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
F2	LINK_MII(2)/CLK25OUT1	BD	CLK25OUT1	O		
C1	PDI[31]/CLK25OUT2	BD	CLK25OUT2	O		
B11	PDI[7]/CPU_CLK_OUT1	BD	CPU_CLK_OUT1	O		
div.	PDI[15]/PDI[23]/PDI[31]/PDI[39]/PDI[45]	BD	CPU_CLK_OUT2	O		
div.	PDI[14]/PDI[22]/PDI[30]/PDI[38]/PDI[44]	BD	CPU_RESET	O		

4.11.1 CLK25OUT1/2 signals

The ET1150 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 ET1150 can share the oscillator output. Depending on the port configuration and C25_ENA, CLK25OUT is available at different pins:

Table 45: CLK25OUT1/2 signal output

Conf.	C25_ENA=0	C25_ENA=1
0-2xMII/RGMII	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/RGMII	CLK25OUT not available, oscillator is mandatory	PDI[31]/CLK25OUT2 provides CLK25OUT
4xMII/RGMII	PDI[31]/CLK25OUT2 provides CLK25OUT	

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

The CLK25OUT pins (if enabled) provide a clock signal during external or ECAT reset, clock output is only turned off during power-on reset.

4.11.2 CPU_CLK_OUT1

The ET1150 can provide a clock signal for µControllers on pin PDI[7]/CPU_CLK_OUT1. The CPU_CLK_OUT1 output setting is controlled by the CLK_MODE configuration pin. If CPU_CLK_OUT1 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_OUT1 (if enabled) provides a clock signal during external or ECAT reset, clock output is only turned off during power-on reset.

4.11.3 CPU_CLK_OUT2

Another clock signal CPI_CLK_OUT2 for µControllers, combined with a reset output CPU_RESET, can be configured as part of the DC signal block. Refer to chapter 4.14.4 for details.

4.12 Physical port signals

4.12.1 MII signals

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 input

RX_DV(x)
MII receive data valid.

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

RX_ERR(x)
MII receive error.

TX_CLK(x)
MII transmit clock input, only used for automatic TX shift compensation. There is no TX FIFO. Connect to GND if manual TX shift compensation is used.

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.

4.12.1.1 Example schematic for MII connection

Refer to chapters 4.11 and 4.12.1 for more information on special markings (!). Take care of proper configuration of LINK_POL, and PHY addresses.

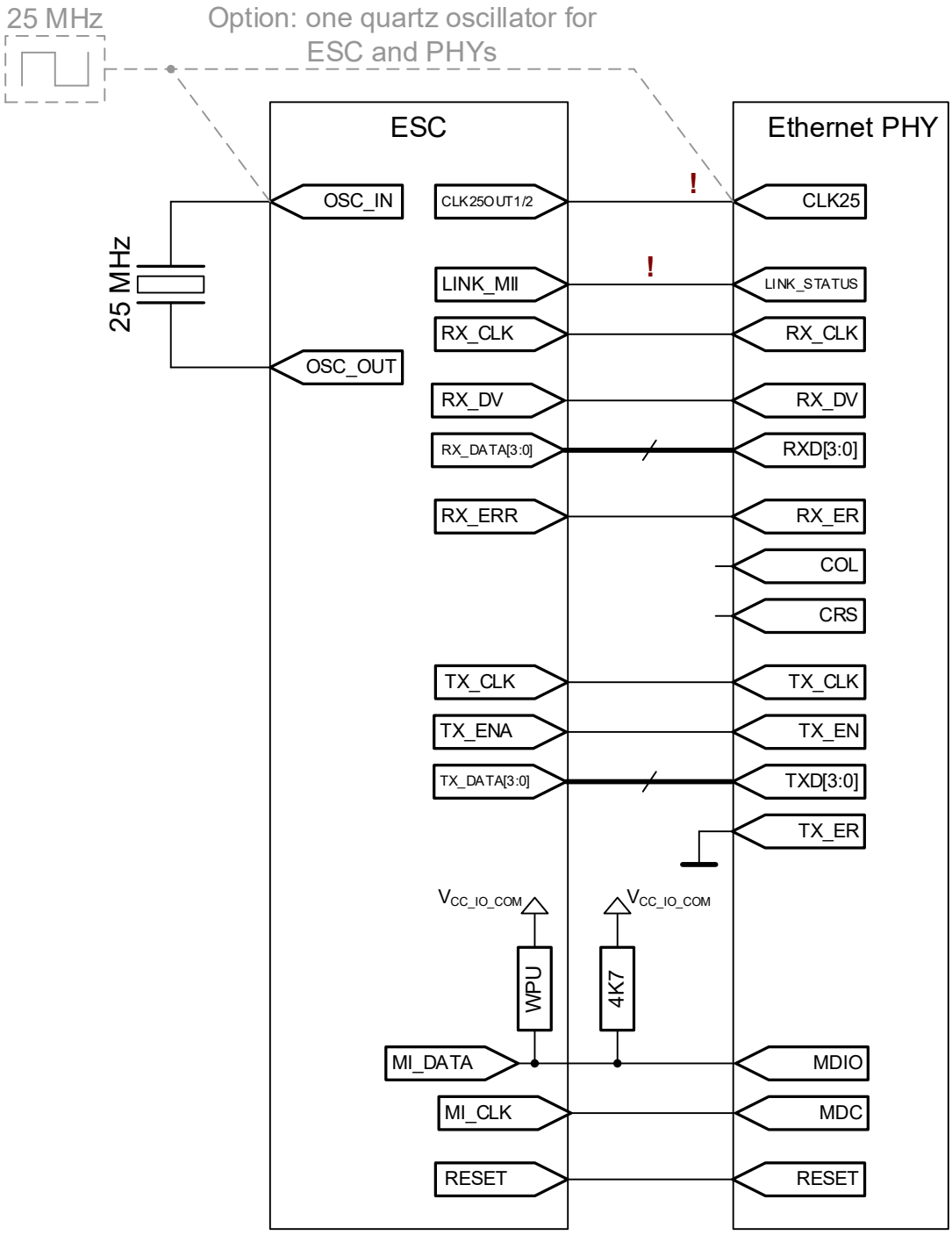

Figure 14: MII PHY connection

4.12.2 RGMII signals

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

If RGMII in-band status is available, set LINK_RGMII(x) to “no link”, according to configured link polarity. Link detection is performed based on RGMII in-band status signaling (recommended).

RX_CLK(x)
RGMII receive clock input

RX_CTL(x)
RGMII receive control.

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

TX_CLK(x)
RGMII transmit clock output

TX_CTL(x)
RGMII transmit control output. Used as RGMII transmit control 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]
RGMII transmit data.

4.12.2.1 Example schematic for RGMII connection

Refer to chapter 4.12.2 for more information on special markings (!). Take care of proper configuration of LINK_POL, and PHY addresses.

If RGMII in-band status is available, set LINK_RGMII to “no link”, depending on link polarity.
PHY clock supply from ET1150 (CLK25OUT1/2, or shared clock source) is recommended, to reduce jitter, but it is not required.

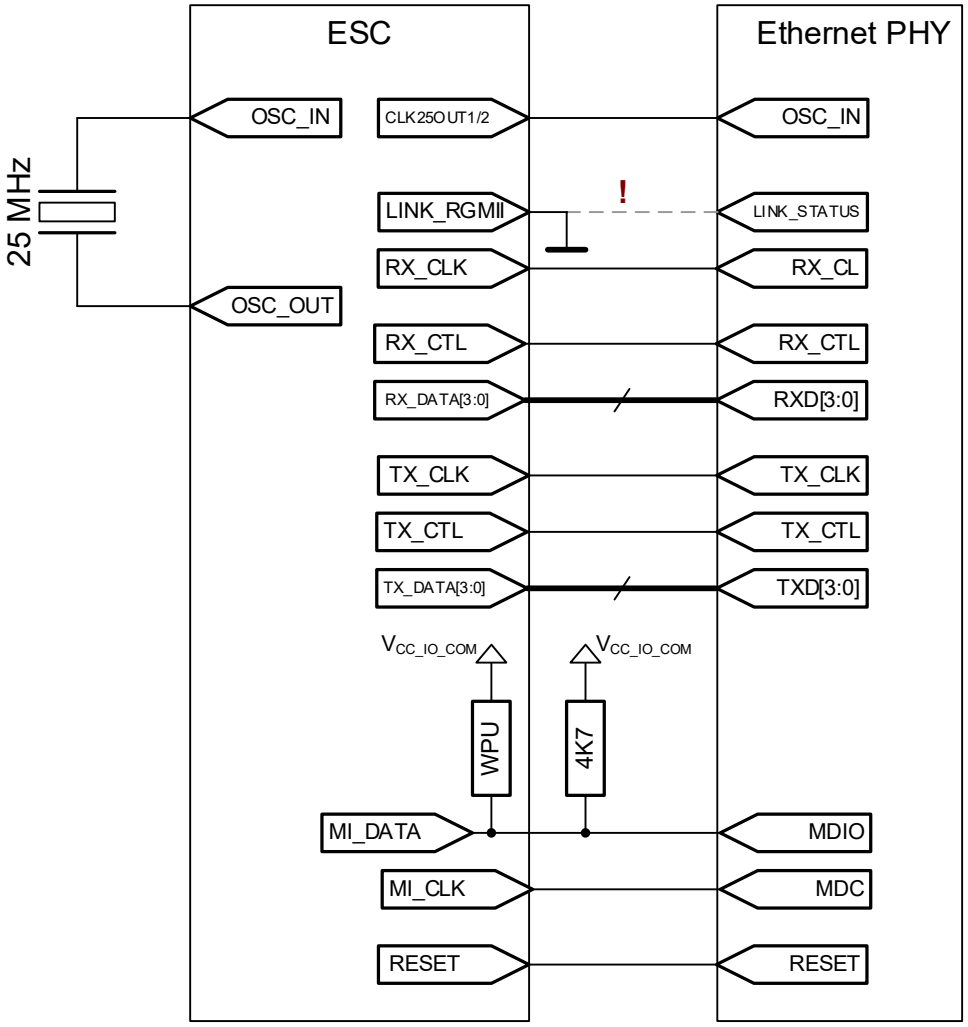

Figure 15: RGMII PHY connection

4.13 Physical ports 0-1

4.13.1 Physical port 0

Table 46 shows the pins for physical port 0. It can be configured as MII, or RGMII, and is always available.

Table 46: Physical port 0

Pin	Pin		MII		RGMII	
	Name	Dir.	Signal	Dir.	Signal	Dir.
M9	TX_ENA(0)	BD/LO+	TX_ENA(0)	O/I	TX_CTL(0)	O/I
L8	TX_D(0)[0]/ C25_ENA	BD	TX_D(0)[0]	O	TX_D(0)[0]	O
M8	TX_D(0)[1]	O/LO-	TX_D(0)[1]	O	TX_D(0)[1]	O
L7	TX_D(0)[2]/ C25_SHI[0]	BD	TX_D(0)[2]	O	TX_D(0)[2]	O
M7	TX_D(0)[3]/ C25_SHI[1]	BD	TX_D(0)[3]	O	TX_D(0)[3]	O
K10	RX_D(0)[0]	I	RX_D(0)[0]	I	RX_D(0)[0]	I
M12	RX_D(0)[1]	I/LI+	RX_D(0)[1]	I	RX_D(0)[1]	I
L11	RX_D(0)[2]	I	RX_D(0)[2]	I	RX_D(0)[2]	I
L12	RX_D(0)[3]	I	RX_D(0)[3]	I	RX_D(0)[3]	I
M11	RX_DV(0)	I/LI-	RX_DV(0]	I	RX_CTL(0]	I
M10	RX_ERR(0)	I	RX_ERR(0)	I	GND_{IO_COM}	UI
L10	RX_CLK(0)	I	RX_CLK(0)	I	RX_CLK(0)	I
L9	LINK_MII(0)	I	LINK_MII(0)	I	LINK_RGMII(0)	I
J11	LED_PERR(0)/ TRANS(0)/ PHY_RES(0)/ CLK_MODE[0]	BD	LED_PERR(0)/ TRANS(0) PHY_RES(0)/	O/ I	LED_PERR(0)/ TRANS(0) PHY_RES(0)/	O/ I
J12	LED_LINKACT(0)/ P_CONF[0]	BD	LED_LINKACT(0)	O	LED_LINKACT(0)	O
K9	TX_CLK(0)	BD	TX_CLK(0)	I	TX_CLK(0)	O

NOTE: Connection of unused inputs (UI) to GND is optional but recommended for lower power consumption.

4.13.2 Physical port 1

Table 47 shows the pins for physical port 1. It can be configured as MII, or RGMII, and is always available.

Physical port 1 can be disabled using register ESC configuration A5[1], but it is active until the EEPROM is loaded – proper configuration is required (especially no link).

Table 47: Physical port 1

Pin	Pin		MII		RGMII	
	Name	Dir.	Signal	Dir.	Signal	Dir.
M3	TX_ENA(1)	BD/LO+	TX_ENA(1)	O/I	TX_CTL(1)	O/I
L3	TX_D(1)[0]/ TRANS-MODE-ENA	BD	TX_D(1)[0]	O	TX_D(1)[0]	O
M2	TX_D(1)[1]	O/LO-	TX_D(1)[1]	O	TX_D(1)[1]	O
L2	TX_D(1)[2]/ P_MODE[0]	BD	TX_D(1)[2]	O	TX_D(1)[2]	O
M1	TX_D(1)[3]/ P_MODE[1]	BD	TX_D(1)[3]	O	TX_D(1)[3]	O
L4	RX_D(1)[0]	I	RX_D(1)[0]	I	RX_D(1)[0]	I
M5	RX_D(1)[1]	I/LI+	RX_D(1)[1]	I	RX_D(1)[1]	I
L5	RX_D(1)[2]	I	RX_D(1)[2]	I	RX_D(1)[2]	I
M6	RX_D(1)[3]	I	RX_D(1)[3]	I	RX_D(1)[3]	I
M4	RX_DV(1)	I/LI-	RX_DV(1)	I	RX_CTL(1)	I
L6	RX_ERR(1)	I	RX_ERR(1)	I	GND_{IO_COM}	UI
K4	RX_CLK(1)	I	RX_CLK(1)	I	RX_CLK(1)	I
K3	LINK_MII(1)	I	LINK_MII(1)	I	LINK_RGMII(1)	I
K2	LED_PERR(1)/ TRANS(1)/ PHY_RES(1)/ CLK_MODE(1)	BD	LED_PERR(1)/ TRANS(1)/ PHY_RES(1)	O/I	LED_PERR(1)/ TRANS(1)/ PHY_RES(1)	O/I
L1	LED_LINKACT(1)/ P_CONF(1)	BD	LED_LINKACT(1)	O	LED_LINKACT(1)	O
G4	TX_CLK(1)	BD	TX_CLK(1)	I	TX_CLK(1)	O

NOTE: Connection of unused inputs (UI) to GND is optional but recommended for lower power consumption.

4.14 PDI[45:0] pins for physical ports 2 and 3, PDI0, PDI1, DC signal block, GPIO

The ET1150 pin out is optimized to achieve an optimum of package size and features. To obtain this, the 46 PDI[45:0] pins are used for multiple purposes/function blocks, ordered by precedence:

1. Physical port 2, port 3, and CPU_CLK_OUT1
2. PDI0 (digital I/O, SPI slave, SPI master, µController)
3. PDI1 (digital I/O, SPI slave, SPI master, µController)
4. DC signal block (additional distributed clocks SyncSignals/LatchSignals, CPU_CLK_OUT2, CPU_RESET)
5. General purpose I/O (input/output)

Function blocks at the top of the priority list occupy PDI[45:0] pins first, remaining unused pins are available for functions lower on the priority list. Most function blocks are arranged as a block of a certain length, and defined start positions in the PDI[45:0] area.

Function blocks on priority 1 (port 2, 3, and CPU_CLK_OUT1) have a fixed length and position within PDI[45:0].

The PDI0 function block starts at PDI[0], but the length depends on the selected PDI0 and its configuration.

The PDI1 function block starts at the next free possible start position (PDI[8], PDI[24], PDI[32], or PDI[40]), after PDI0 occupies its pins.

The DC signal block starts at the next free possible start position (PDI[8], PDI[16], PDI[24], PDI[32], or PDI[40]), after PDI0 and PDI1 occupied their pins. Optionally, the DC signal block can be moved to a start position 8 pins lower, partially overlapping with PDI0/PDI1 signals. The DC signal block is only partially available then, but more pins remain available as GPIO.

Table 48: PDI[45:0] pin usage priority

Pin	PDI signal	1. Priority	2. Priority	3. Priority	4. Priority	5. Priority
		MII/RGMII ports, CPU_CLK_OUT1	PDI0	PDI1	DC block	GPIO
D12	PDI[0]		PDI0 start pin			GPIO[0]
D11	PDI[1]					GPIO[1]
C12	PDI[2]					GPIO[2]
C11	PDI[3]					GPIO[3]
B12	PDI[4]					GPIO[4]
C10	PDI[5]					GPIO[5]
A12	PDI[6]					GPIO[6]
B11	PDI[7]	CPU_CLK_OUT1				GPIO[7]
A11	PDI[8]			Possible PDI1 start pin	Possible DC block start pin	GPIO[8]
B10	PDI[9]					GPIO[9]
A10	PDI[10]					GPIO[10]
C9	PDI[11]					GPIO[11]
A9	PDI[12]					GPIO[12]
B9	PDI[13]					GPIO[13]
A8	PDI[14]					GPIO[14]
B8	PDI[15]					GPIO[15]
A7	PDI[16]	MII/RGMII port 3		-	Possible DC block start pin	GPIO[16]
B7	PDI[17]					GPIO[17]
A6	PDI[18]					GPIO[18]
B6	PDI[19]					GPIO[19]
A5	PDI[20]					GPIO[20]
B5	PDI[21]					GPIO[21]
A4	PDI[22]					GPIO[22]
B4	PDI[23]					GPIO[23]
A3	PDI[24]			Possible PDI1 start pin	Possible DC block start pin	GPIO[24]
B3	PDI[25]					GPIO[25]
A2	PDI[26]					GPIO[26]
A1	PDI[27]					GPIO[27]
B2	PDI[28]					GPIO[28]
B1	PDI[29]					GPIO[29]
C2	PDI[30]					GPIO[30]
C1	PDI[31]					GPIO[31]
D1	PDI[32]	MII/RGMII port 2		Possible PDI1 start pin	Possible DC block start pin	GPIO[32]
D2	PDI[33]					GPIO[33]
E2	PDI[34]					GPIO[34]
G1	PDI[35]					GPIO[35]
G2	PDI[36]					GPIO[36]
H2	PDI[37]					GPIO[37]
J2	PDI[38]					GPIO[38]
K1	PDI[39]					GPIO[39]
F1	PDI[40]			Possible PDI1 start pin	Possible DC block start pin	GPIO[40]
E1	PDI[41]					GPIO[41]
H1	PDI[42]					GPIO[42]
J1	PDI[43]					GPIO[43]
C3	PDI[44]					GPIO[44]
E3	PDI[45]					GPIO[45]

4.14.1 Priority 1 (port 2/3, CPU_CLK_OUT1)

4.14.1.1 CPU_CLK_OUT1 / PDI[7]

Table 49: CPU_CLK_OUT1 / PDI[7]

Pin	Pin		CLK_MODE = 00		CLK_MODE /= 00	
	Name	Dir.	Signal	Dir.	Signal	Dir.
B11	PDI[7]/CPU_CLK_OUT1	BD	PDI[7]	BD	CPU_CLK_OUT1	O

4.14.1.2 Physical port 3 / PDI[31:16]

Table 50 shows the pins for physical port 3, located at PDI[31:16]. If used as communication port, it can be configured as MII, or RGMII.

Table 50: Physical port 3 / PDI[31:16]

Pin	Pin		No port 3		MII		RGMII	
	Name	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.
A7	PDI[16]/ RX_ERR(3)	BD	PDI[16]	BD	RX_ERR(3)	I	- (GND_{PDI})	UI
B7	PDI[17]/ RX_CLK(3)	BD	PDI[17]	BD	RX_CLK(3)	I	RX_CLK(3)	I
A6	PDI[18]/ RX_D(3)[0]	BD	PDI[18]	BD	RX_D(3)[0]	I	RX_D(3)[0]	I
B6	PDI[19]/ RX_D(3)[2]	BD	PDI[19]	BD	RX_D(3)[2]	I	RX_D(3)[2]	I
A5	PDI[20]/ RX_D(3)[3]	BD	PDI[20]	BD	RX_D(3)[3]	I	RX_D(3)[3]	I
B5	PDI[21]/ LINK_MII(3)	BD	PDI[21]	BD	LINK_MII(3)	I	LINK_RGMII(3)	I
A4	PDI[22]/ TX_D(3)[3]	BD	PDI[22]	BD	TX_D(3)[3]	O	TX_D(3)[3]	O
B4	PDI[23]/ TX_D(3)[2]	BD	PDI[23]	BD	TX_D(3)[2]	O	TX_D(3)[2]	O
A3	PDI[24]/ TX_D(3)[1]	BD/ LO-	PDI[24]	BD	TX_D(3)[1]	O	TX_D(3)[1]	O
B3	PDI[25]/ TX_D(3)[0]	BD	PDI[25]	BD	TX_D(3)[0]	O	TX_D(3)[0]	O
A2	PDI[26]/ TX_ENA(3)	BD/ LO+	PDI[26]	BD	TX_ENA(3)	O/ I	TX_CTL(3)	O/ I
A1	PDI[27]/ RX_DV(3)	BD/ LI-	PDI[27]	BD	RX_DV(3)	I	RX_CTL(3)	I
B2	PDI[28]/ LED_PERR(3)/ TRANS(3)/ PHY_RES(3)	BD	PDI[28]	BD	LED_PERR(3)/ TRANS(3)/ PHY_RES(3)	O/ I	LED_PERR(3)/ TRANS(3)/ PHY_RES(3)	O/ I
B1	PDI[29]/ RX_D(3)[1]	BD/ LI+	PDI[29]	BD	RX_D(3)[1]	I	RX_D(3)[1]	I
C2	PDI[30]/ LED_ LINKACT(3)/ P_CONF(3)	BD	PDI[30]	BD	LED_ LINKACT(3)	O	LED_ LINKACT(3)	O
C1	PDI[31]/ CLK25OUT2	BD	PDI[31]/ CLK25 OUT2	BD	CLK25OUT2	O	CLK25OUT2	O
Additional signals related to physical port 3								
C8	TX_CLK(3)	BD	- (GND_{PDI})	UI	TX_CLK(3)	I	TX_CLK(3)	O

4.14.1.3 Physical port 2 / PDI[45:32]

Table 51 shows the pins for physical port 2, located at PDI[45:32]. If used as communication port, it can be configured as MII, or RGMII

Table 51: Physical port 2 / PDI[45:32]

Pin	Pin		No port 2		MII		RGMII	
	Name	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.
D1	PDI[32]/ TX_D(2)[3]	BD	PDI[32]	BD	TX_D(2)[3]	O	TX_D(2)[3]	O
D2	PDI[33]/ TX_D(2)[2]	BD	PDI[33]	BD	TX_D(2)[2]	O	TX_D(2)[2]	O
E2	PDI[34]/ TX_D(2)[0]/ CTRL_STATUS_ MOVE	BD	PDI[34]	BD	TX_D(2)[0]	O	TX_D(2)[0]	O
G1	PDI[35]/ RX_ERR(2)	BD	PDI[35]	BD	RX_ERR(2)	I	- (GND_{PDI})	UI
G2	PDI[36]/ RX_CLK(2)	BD	PDI[36]	BD	RX_CLK(2)	I	RX_CLK(2)	I
H2	PDI[37]/ RX_D(2)[0]	BD	PDI[37]	BD	RX_D(2)[0]	I	RX_D(2)[0]	I
J2	PDI[38]/ RX_D(2)[2]	BD	PDI[38]	BD	RX_D(2)[2]	I	RX_D(2)[2]	I
K1	PDI[39]/ RX_D(2)[3]	BD	PDI[39]	BD	RX_D(2)[3]	I	RX_D(2)[3]	I
F1	PDI[40]/ TX_ENA(2)	BD/ LO+	PDI[40]	BD	TX_ENA(2)	O/I	TX_CTL(2)	O/I
E1	PDI[41]/ TX_D(2)[1]	O/ LO-	PDI[41]	BD	TX_D(2)[1]	O	TX_D(2)[1]	O
H1	PDI[42]/ RX_DV(2)	I/LI-	PDI[42]	BD	RX_DV(2)	I	RX_CTL(2)	I
J1	PDI[43]/ RX_D(2)[1]	I/LI+	PDI[43]	BD	RX_D(2)[1]	I	RX_D(2)[1]	I
C3	PDI[44]/ LED_PERR(2)/ TRANS(2)/ PHY_RES(2)/ PHYAD_OFF	BD	PDI[44]	BD	LED_PERR(2)/ TRANS(2)/ PHY_RES(2)	O/ I	LED_PERR(2)/ TRANS(2)/ PHY_RES(2)	O/ I
E3	PDI[45]/ LED_ LINKACT(2)/ P_CONF[2]	BD	PDI[45]	BD	LED_ LINKACT(2)	O	LED_ LINKACT(2)	O
Additional signals related to physical port 2								
F2	LINK_MII(2)/ CLK25OUT1	BD	CLK25 OUT1	O	LINK_MII(2)	I	LINK_RGMII(2)	I
G2	TX_CLK(2)	BD	- (GND_{PDI})	UI	TX_CLK(3)	I	TX_CLK(3)	O

4.14.2 Priority 2 (PDI0)

The PDI0 signal pinout depends on the selected PDI0 (SII EEPROM). Refer to PDI descriptions for further PDI and PDI signal descriptions.

4.14.2.1 Digital I/O PDI0

The digital I/O PDI0 has up to 32 data signals, and a set of eight control/status signals.

Table 52: Digital I/O PDI0 data signals

PDI[31:0]	Digital I/O signal	
	Signal	Dir.
31:0	I/O[31:0]	I/O/BD

The set of control/status signals is usually available on PDI[39:32]. If this block is occupied by communication port 2, the control/status signals are not available. With the configuration signal CTRL_STATUS_MOVE=1, the set of control/status signals can be moved to the last free 8 signals block, starting from PDI[8] – but the number of data signals is reduced.

Table 53: Digital I/O PDI0 control/status signals

Relative position	Digital I/O signal		Default value if control signals are not available
	Signal	Dir.	
0	SOF	O	-
1	OE_EXT	I	1
2	OUTVALID	O	-
3	WD_TRIG	O	-
4	LATCH_IN	I	0 (mode cannot be used)
5	OE_CONF	I	0
6	EEPROM_LOADED	O	-
7	OUT_START	I	0 (mode cannot be used)

The different pinout variants are shown in the next table. If CPU_CLK_OUT1 is used, PDI[7] is no longer available for digital I/O – but all other signals are still available.

Table 54: Digital I/O mapping for PDI0

PDI signal	3 ports MII/RGMII CTRL_STATUS_MOVE		4 ports MII/RGMII CTRL_STATUS_MOVE	
	0	1	0	1
PDI[0]	I/O[0]	I/O[0]	I/O[0]	I/O[0]
PDI[1]	I/O[1]	I/O[1]	I/O[1]	I/O[1]
PDI[2]	I/O[2]	I/O[2]	I/O[2]	I/O[2]
PDI[3]	I/O[3]	I/O[3]	I/O[3]	I/O[3]
PDI[4]	I/O[4]	I/O[4]	I/O[4]	I/O[4]
PDI[5]	I/O[5]	I/O[5]	I/O[5]	I/O[5]
PDI[6]	I/O[6]	I/O[6]	I/O[6]	I/O[6]
PDI[7]	I/O[7]	I/O[7]	I/O[7]	I/O[7]
PDI[8]	I/O[8]	I/O[8]	I/O[8]	SOF
PDI[9]	I/O[9]	I/O[9]	I/O[9]	OE_EXT
PDI[10]	I/O[10]	I/O[10]	I/O[10]	OUTVALID
PDI[11]	I/O[11]	I/O[11]	I/O[11]	WD_TRIG
PDI[12]	I/O[12]	I/O[12]	I/O[12]	LATCH_IN
PDI[13]	I/O[13]	I/O[13]	I/O[13]	OE_CONF
PDI[14]	I/O[14]	I/O[14]	I/O[14]	EEPROM LOADED
PDI[15]	I/O[15]	I/O[15]	I/O[15]	OUT_START
PDI[16]	I/O[16]	I/O[16]	MII/RGMII port 3	MII/RGMII port 3
PDI[17]	I/O[17]	I/O[17]		
PDI[18]	I/O[18]	I/O[18]		
PDI[19]	I/O[19]	I/O[19]		
PDI[20]	I/O[20]	I/O[20]		
PDI[21]	I/O[21]	I/O[21]		
PDI[22]	I/O[22]	I/O[22]		
PDI[23]	I/O[23]	I/O[23]		
PDI[24]	I/O[24]	SOF		
PDI[25]	I/O[25]	OE_EXT		
PDI[26]	I/O[26]	OUTVALID		
PDI[27]	I/O[27]	WD_TRIG		
PDI[28]	I/O[28]	LATCH_IN		
PDI[29]	I/O[29]	OE_CONF		
PDI[30]	I/O[30]	EEPROM_ LOADED		
PDI[31]	I/O[31]	OUT_START		
PDI[32]	MII/RGMII port 2	MII/RGMII port 2	MII/RGMII port 2	MII/RGMII port 2
PDI[33]				
PDI[34]				
PDI[35]				
PDI[36]				
PDI[37]				
PDI[38]				
PDI[39]				
PDI[40]				
PDI[41]				
PDI[42]				
PDI[43]				
PDI[44]				
PDI[45]				

4.14.2.2 SPI slave PDI0

The SPI slave PDI0 supports a compatibility mode that has the same fixed GPIO numbers and mapping as ET1100. This mode will be enabled if ESC configuration area B is not configured (0x144[0]=0) and SPI additional features are disabled (0x0150[7]=0). Otherwise, flexible GPIO mapping will be used. It can also be configured to support additional features (SEL_MISO/SEL_MOSI, data width x1/x2/x4/x8, and unidirectional/bidirectional operation).

Table 55: SPI slave mapping for PDI0, ET1100 compatible

PDI signal	ET1100 compatible: 0x0144[0]=0 and 0x0150[7]=0		Corresponding register	
	Signal	Dir.	GPO register	GPI register
PDI[0]	SPI_CLK	I		
PDI[1]	SPI_SEL	I		
PDI[2]	SPI_MOSI[0]	I		
PDI[3]	SPI_MISO[0]	O		
PDI[4]	SPI_IRQ	O		
PDI[5]	-	UI		
PDI[6]	EEPROM_LOADED	O		
PDI[7]	-	UI		
PDI[8]	GPO[0]	O	0x0F10[0]	
PDI[9]	GPO[1]	O	0x0F10[1]	
PDI[10]	GPO[2]	O	0x0F10[2]	
PDI[11]	GPO[3]	O	0x0F10[3]	
PDI[12]	GPI[0]	I		0x0F18[0]
PDI[13]	GPI[1]	I		0x0F18[1]
PDI[14]	GPI[2]	I		0x0F18[2]
PDI[15]	GPI[3]	I		0x0F18[3]
PDI[16]	GPO[4]	O	0x0F10[4]	
PDI[17]	GPO[5]	O	0x0F10[5]	
PDI[18]	GPO[6]	O	0x0F10[6]	
PDI[19]	GPO[7]	O	0x0F10[7]	
PDI[20]	GPI[4]	I		0x0F18[4]
PDI[21]	GPI[5]	I		0x0F18[5]
PDI[22]	GPI[6]	I		0x0F18[6]
PDI[23]	GPI[7]	I		0x0F18[7]
PDI[24]	GPO[8]	O	0x0F10[8]	
PDI[25]	GPO[9]	O	0x0F10[9]	
PDI[26]	GPO[10]	O	0x0F10[10]	
PDI[27]	GPO[11]	O	0x0F10[11]	
PDI[28]	GPI[8]	I		0x0F18[8]
PDI[29]	GPI[9]	I		0x0F18[9]
PDI[30]	GPI[10]	I		0x0F18[10]
PDI[31]	GPI[11]	I		0x0F18[11]
PDI[32]	GPO[12]	O	0x0F10[12]	
PDI[33]	GPO[13]	O	0x0F10[13]	
PDI[34]	GPO[14]	O	0x0F10[14]	
PDI[35]	GPO[15]	O	0x0F10[15]	
PDI[36]	GPI[12]	I		0x0F18[12]
PDI[37]	GPI[13]	I		0x0F18[13]
PDI[38]	GPI[14]	I		0x0F18[14]
PDI[39]	GPI[15]	I		0x0F18[15]
PDI[40]	-	UI		
PDI[41]	-	UI		
PDI[42]	-	UI		
PDI[43]	-	UI		
PDI[44]	-	UI		
PDI[45]	-	UI		

Table 56: SPI slave mapping for PDI0, unidirectional (0x0152[6]=0)

PDI signal	x1 0x0150[7]=0 ESC configuration area B configured		x1 0x0150[7]=1 0x0152[1:0]=00		x2 0x0150[7]=1 0x0152[1:0]=01		x4 0x0150[7]=1 0x0152[1:0]=10		x8 0x0150[7]=1 0x0152[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
PDI[0]	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I
PDI[1]	**SPI_SEL**	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I
PDI[2]	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I
PDI[3]	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O
PDI[4]	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O
PDI[5]	-	UI	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I
PDI[6]	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O
PDI[7]	-	UI	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I
PDI[8]	Unused. Available for lower priority function blocks		Unused. Available for lower priority function blocks		**SPI_MOSI[1]**	I	**SPI_MOSI[1]**	I	**SPI_MOSI[1]**	I
PDI[9]					**SPI_MISO[1]**	O	**SPI_MISO[1]**	O	**SPI_MISO[1]**	O
PDI[10]					Unused. Available for lower priority function blocks		**SPI_MOSI[2]**	I	**SPI_MOSI[2]**	I
PDI[11]							**SPI_MISO[2]**	O	**SPI_MISO[2]**	O
PDI[12]							**SPI_MOSI[3]**	I	**SPI_MOSI[3]**	I
PDI[13]							**SPI_MISO[3]**	O	**SPI_MISO[3]**	O
PDI[14]							Unused. Available for lower priority function blocks		**SPI_MOSI[4]**	I
PDI[15]									**SPI_MISO[4]**	O
PDI[16]									**SPI_MOSI[5]**	I
PDI[17]									**SPI_MISO[5]**	O
PDI[18]									**SPI_MOSI[6]**	I
PDI[19]									**SPI_MISO[6]**	O
PDI[20]									**SPI_MOSI[7]**	I
PDI[21]									**SPI_MISO[7]**	O
PDI[45:22]									Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

Table 57: SPI slave mapping for PDI0, bidirectional (0x0152[6]=1)

PDI signal	x1 0x0150[7]=1 0x0152[1:0]=00		x2 0x0150[7]=1 0x0152[1:0]=01		x4 0x0150[7]=1 0x0152[1:0]=10		x8 0x0150[7]=1 0x0152[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
PDI[0]	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I
PDI[1]	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I
PDI[2]	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD
PDI[3]	SPI_MISO_ENA	O	SPI_MISO_ENA	O	SPI_MISO_ENA	O	SPI_MISO_ENA	O
PDI[4]	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O
PDI[5]	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I
PDI[6]	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O
PDI[7]	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I
PDI[8]	Unused. Available for lower priority function blocks		**SPI_D[1]**	BD	**SPI_D[1]**	BD	**SPI_D[1]**	BD
PDI[9]			Unused. Available for lower priority function blocks		**SPI_D[2]**	BD	**SPI_D[2]**	BD
PDI[10]					**SPI_D[3]**	BD	**SPI_D[3]**	BD
PDI[11]					Unused. Available for lower priority function blocks		**SPI_D[4]**	BD
PDI[12]							**SPI_D[5]**	BD
PDI[13]							**SPI_D[6]**	BD
PDI[14]							**SPI_D[7]**	BD
PDI[45:15]							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.2.3 SPI master PDI0

Table 58: SPI master number of MISO/MOSI signals

Number of MISO/MOSI signals	0x0152[1:0]	0x0152[3:2]
x1	00	00
x2	00	01
	01	00
x4	00	10
	01	01
	10	00
x8	00	11
	01	10
	10	01
	11	00

Table 59: SPI master mapping for PDI0, unidirectional (0x0152[6]=0)

PDI signal	MISO/MOSI signals x1		x2		x4		x8	
	Signal	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI[0]	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O
PDI[1]	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O
PDI[2]	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O
PDI[3]	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I
PDI[4]	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I
PDI[5]	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I
PDI[6]	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O
PDI[7]	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I
PDI[8]	Unused. Available for lower priority function blocks		**SPI_MOSI[1]**	O	**SPI_MOSI[1]**	O	**SPI_MOSI[1]**	O
PDI[9]			**SPI_MISO[1]**	I	**SPI_MISO[1]**	I	**SPI_MISO[1]**	I
PDI[10]			Unused. Available for lower priority function blocks		**SPI_MOSI[2]**	O	**SPI_MOSI[2]**	I
PDI[11]					**SPI_MISO[2]**	I	**SPI_MISO[2]**	O
PDI[12]					**SPI_MOSI[3]**	O	**SPI_MOSI[3]**	I
PDI[13]					**SPI_MISO[3]**	I	**SPI_MISO[3]**	O
PDI[14]					Unused. Available for lower priority function blocks		**SPI_MOSI[4]**	O
PDI[15]							**SPI_MISO[4]**	I
PDI[16]							**SPI_MOSI[5]**	O
PDI[17]							**SPI_MISO[5]**	I
PDI[18]							**SPI_MOSI[6]**	O
PDI[19]							**SPI_MISO[6]**	I
PDI[20]							**SPI_MOSI[7]**	O
PDI[21]							**SPI_MISO[7]**	I
PDI[45:22]							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

Table 60: SPI master mapping for PDI0, bidirectional (0x0152[6]=1)

PDI signal	x1 0x0152[1:0]=00		x2 0x0152[1:0]=01		x4 0x0152[1:0]=10		x8 0x0152[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
PDI[0]	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O
PDI[1]	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O
PDI[2]	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD
PDI[3]	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O
PDI[4]	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I
PDI[5]	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I
PDI[6]	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O
PDI[7]	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I
PDI[8]	Unused. Available for lower priority function blocks		**SPI_D[1]**	BD	**SPI_D[1]**	BD	**SPI_D[1]**	BD
PDI[9]			Unused. Available for lower priority function blocks		**SPI_D[2]**	BD	**SPI_D[2]**	BD
PDI[10]					**SPI_D[3]**	BD	**SPI_D[3]**	BD
PDI[11]					Unused. Available for lower priority function blocks		**SPI_D[4]**	BD
PDI[12]							**SPI_D[5]**	BD
PDI[13]							**SPI_D[6]**	BD
PDI[14]							**SPI_D[7]**	BD
PDI[45:15]							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.2.4 8/16 bit asynchronous/synchronous µController PDI0

Table 61: Mapping of asynchronous/synchronous µC PDI0

PDI signal	Async. µController				Sync. µController			
	8 bit 0x0140=0x09 0x0152[3]=0		16 bit 0x0140=0x0 0x0152[3]=0		8 bit 0x0140=0x0B 0x0152[3]=0		16 bit 0x0140=0x0A 0x0152[3]=0	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
PDI[0]	**CS**	I	**CS**	I	**CS**	I	**CS**	I
PDI[1]	**RD**	I	**RD**	I	**TS**	I	**TS**	I
PDI[2]	**WR**	I	**WR**	I	**RD/nWR**	I	**RD/nWR**	I
PDI[3]	BUSY	O	BUSY	O	TA	O	TA	O
PDI[4]	IRQ	O	IRQ	O	IRQ	O	IRQ	O
PDI[5]	BHE	I	BHE	I	-	UI	BHE	I
PDI[6]	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O
PDI[7]	ADR[15]	I	ADR[15]	I	ADR[15]	I	ADR[15]	I
PDI[8]	ADR[14]	I	ADR[14]	I	ADR[14]	I	ADR[14]	I
PDI[9]	**ADR[13]**	I	**ADR[13]**	I	**ADR[13]**	I	**ADR[13]**	I
PDI[10]	**ADR[12]**	I	**ADR[12]**	I	**ADR[12]**	I	**ADR[12]**	I
PDI[11]	**ADR[11]**	I	**ADR[11]**	I	**ADR[11]**	I	**ADR[11]**	I
PDI[12]	**ADR[10]**	I	**ADR[10]**	I	**ADR[10]**	I	**ADR[10]**	I
PDI[13]	**ADR[9]**	I	**ADR[9]**	I	**ADR[9]**	I	**ADR[9]**	I
PDI[14]	**ADR[8]**	I	**ADR[8]**	I	**ADR[8]**	I	**ADR[8]**	I
PDI[15]	**ADR[7]**	I	**ADR[7]**	I	**ADR[7]**	I	**ADR[7]**	I
PDI[16]	**ADR[6]**	I	**ADR[6]**	I	**ADR[6]**	I	**ADR[6]**	I
PDI[17]	**ADR[5]**	I	**ADR[5]**	I	**ADR[5]**	I	**ADR[5]**	I
PDI[18]	**ADR[4]**	I	**ADR[4]**	I	**ADR[4]**	I	**ADR[4]**	I
PDI[19]	**ADR[3]**	I	**ADR[3]**	I	**ADR[3]**	I	**ADR[3]**	I
PDI[20]	**ADR[2]**	I	**ADR[2]**	I	**ADR[2]**	I	**ADR[2]**	I
PDI[21]	**ADR[1]**	I	**ADR[1]**	I	**ADR[1]**	I	**ADR[1]**	I
PDI[22]	**ADR[0]**	I	**ADR[0]**	I	**ADR[0]**	I	**ADR[0]**	I
PDI[23]	**DATA[0]**	BD	**DATA[0]**	BD	**DATA[0]**	BD	**DATA[0]**	BD
PDI[24]	**DATA[1]**	BD	**DATA[1]**	BD	**DATA[1]**	BD	**DATA[1]**	BD
PDI[25]	**DATA[2]**	BD	**DATA[2]**	BD	**DATA[2]**	BD	**DATA[2]**	BD
PDI[26]	**DATA[3]**	BD	**DATA[3]**	BD	**DATA[3]**	BD	**DATA[3]**	BD
PDI[27]	**DATA[4]**	BD	**DATA[4]**	BD	**DATA[4]**	BD	**DATA[4]**	BD
PDI[28]	**DATA[5]**	BD	**DATA[5]**	BD	**DATA[5]**	BD	**DATA[5]**	BD
PDI[29]	**DATA[6]**	BD	**DATA[6]**	BD	**DATA[6]**	BD	**DATA[6]**	BD
PDI[30]	**DATA[7]**	BD	**DATA[7]**	BD	**DATA[7]**	BD	**DATA[7]**	BD
PDI[31]	Unused. Available for lower priority function blocks		-	UI	**CPU_CLK_IN**	I	**CPU_CLK_IN**	I
PDI[32]			**DATA[8]**	BD	Unused. Available for lower priority function blocks		**DATA[8]**	BD
PDI[33]			**DATA[9]**	BD			**DATA[9]**	BD
PDI[34]			**DATA[10]**	BD			**DATA[10]**	BD
PDI[35]			**DATA[11]**	BD			**DATA[11]**	BD
PDI[36]			**DATA[12]**	BD			**DATA[12]**	BD
PDI[37]			**DATA[13]**	BD			**DATA[13]**	BD
PDI[38]			**DATA[14]**	BD			**DATA[14]**	BD
PDI[39]			**DATA[15]**	BD			**DATA[15]**	BD
PDI[45:40]			Unused. Available for lower priority function blocks				Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.2.5 8/16/32 bit asynchronous multiplexed µController PDI0

Table 62: Mapping of asynchronous multiplexed µController PDI0

PDI signal	Async. multiplexed µController					
	8 bit 0x0140=0x0D 0x0152[3]=0		16 bit 0x0140=0x0C 0x0152[3]=0		32 bit 0x0140=0x0D) 0x0152[3]=1	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
PDI[0]	**CS**	I	**CS**	I	**CS**	I
PDI[1]	**RD**	I	**RD**	I	**TS**	I
PDI[2]	**WR**	I	**WR**	I	**RD/nWR**	I
PDI[3]	BUSY	O	BUSY	O	TA	O
PDI[4]	IRQ	O	IRQ	O	IRQ	O
PDI[5]	**ALE**	I	**ALE**	I	**ALE**	I
PDI[6]	EEPROM_LOADED	O	EEPROM_LOADED	O	EEPROM_LOADED	O
PDI[7]	ADR[15]	I	**AD[15]**	BD	**AD[15]**	BD
PDI[8]	ADR[14]	I	**AD[14]**	BD	**AD[14]**	BD
PDI[9]	**ADR[13]**	I	**AD[13]**	BD	**AD[13]**	BD
PDI[10]	**ADR[12]**	I	**AD[12]**	BD	**AD[12]**	BD
PDI[11]	**ADR[11]**	I	**AD[11]**	BD	**AD[11]**	BD
PDI[12]	**ADR[10]**	I	**AD[10]**	BD	**AD[10]**	BD
PDI[13]	**ADR[9]**	I	**AD[9]**	BD	**AD[9]**	BD
PDI[14]	**ADR[8]**	I	**AD[8]**	BD	**AD[8]**	BD
PDI[15]	**AD[7]**	BD	**AD[7]**	BD	**AD[7]**	BD
PDI[16]	**AD[6]**	BD	**AD[6]**	BD	**AD[6]**	BD
PDI[17]	**AD[5]**	BD	**AD[5]**	BD	**AD[5]**	BD
PDI[18]	**AD[4]**	BD	**AD[4]**	BD	**AD[4]**	BD
PDI[19]	**AD[3]**	BD	**AD[3]**	BD	**AD[3]**	BD
PDI[20]	**AD[2]**	BD	**AD[2]**	BD	**AD[2]**	BD
PDI[21]	**AD[1]**	BD	**AD[1]**	BD	**AD[1]**	BD
PDI[22]	**AD[0]**	BD	**AD[0]**	BD	**AD[0]**	BD
PDI[23]	Unused. Available for lower priority function blocks		-	UI	-	UI
PDI[24]			BE[0]	I	**BE[0]**	I
PDI[25]			**BE[1]**	I	**BE[1]**	I
PDI[26]			Unused. Available for lower priority function blocks		**BE[2]**	I
PDI[27]					**BE[3]**	I
PDI[28]					**AD[31]**	BD
PDI[29]					**AD[30]**	BD
PDI[30]					**AD[29]**	BD
PDI[31]					**AD[28]**	BD
PDI[32]					**AD[27]**	BD
PDI[33]					**AD[26]**	BD
PDI[34]					**AD[25]**	BD
PDI[35]					**AD[24]**	BD
PDI[36]					**AD[23]**	BD
PDI[37]					**AD[22]**	BD
PDI[38]					**AD[21]**	BD
PDI[39]					**AD[20]**	BD
PDI[40]					**AD[19]**	BD
PDI[41]					**AD[18]**	BD
PDI[42]					**AD[17]**	BD
PDI[43]					**AD[16]**	BD
PDI[44]					Unused. Available for lower priority function blocks	
PDI[45]						

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.3 Priority 3 (PDI1)

The PDI1 signal pinout depends on the selected PDI1 (SII EEPROM). Refer to PDI descriptions for further PDI and PDI signal descriptions.

The pinout or PDI1 is given as a relative position to the PDI1 start pin, which can be 8, 24, 32, or 40, depending on PDI0.

4.14.3.1 Digital I/O PDI1

The digital I/O PDI1 has up to 32 data signals, without control/status signals. I/O signals, which would be mapped beyond PDI[45] are not available.

Digital I/O PDI1 input data is stored in digital I/O input data register 0x0F08:0x0F0B. Digital I/O PDI1 output data is taken from digital I/O output data register 0x0F00:0x0F003.

Table 63: Digital I/O PDI1 data signals

Relative position	Digital I/O signal	
	Signal	Dir.
0	I/O[0]	I/O
1	I/O[1]	I/O
2	I/O[2]	I/O
3	I/O[3]	I/O
4	I/O[4]	I/O
5	I/O[5]	I/O
6	I/O[6]	I/O
7	I/O[7]	I/O
8	I/O[8]	I/O
9	I/O[9]	I/O
10	I/O[10]	I/O
11	I/O[11]	I/O
12	I/O[12]	I/O
13	I/O[13]	I/O
14	I/O[14]	I/O
15	I/O[15]	I/O
16	I/O[16]	I/O
17	I/O[17]	I/O
18	I/O[18]	I/O
19	I/O[19]	I/O
20	I/O[20]	I/O
21	I/O[21]	I/O
22	I/O[22]	I/O
23	I/O[23]	I/O
24	I/O[24]	I/O
25	I/O[25]	I/O
26	I/O[26]	I/O
27	I/O[27]	I/O
28	I/O[28]	I/O
29	I/O[29]	I/O
30	I/O[30]	I/O
31	I/O[31]	I/O

4.14.3.2 SPI slave PDI1

Table 64: SPI slave mapping for PDI1, unidirectional (0x0192[6]=0)

Relative position	x1 0x0190[7]=0		x1 0x0190[7]=1 0x0192[1:0]=00		x2 0x0190[7]=1 0x0192[1:0]=01		x4 0x0190[7]=1 0x0192[1:0]=10		x8 0x0190[7]=1 0x0192[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
0	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I
1	**SPI_SEL**	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I
2	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I	**SPI_MOSI[0]**	I
3	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O	**SPI_MISO[0]**	O
4	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O
5	-	UI	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I
6	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O
7	-	UI	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I
8	Unused. Available for lower priority function blocks		Unused. Available for lower priority function blocks		**SPI_MOSI[1]**	I	**SPI_MOSI[1]**	I	**SPI_MOSI[1]**	I
9					**SPI_MISO[1]**	O	**SPI_MISO[1]**	O	**SPI_MISO[1]**	O
10					Unused. Available for lower priority function blocks		**SPI_MOSI[2]**	I	**SPI_MOSI[2]**	I
11							**SPI_MISO[2]**	O	**SPI_MISO[2]**	O
12							**SPI_MOSI[3]**	I	**SPI_MOSI[3]**	I
13							**SPI_MISO[3]**	O	**SPI_MISO[3]**	O
14							Unused. Available for lower priority function blocks		**SPI_MOSI[4]**	I
15									**SPI_MISO[4]**	O
16									**SPI_MOSI[5]**	I
17									**SPI_MISO[5]**	O
18									**SPI_MOSI[6]**	I
19									**SPI_MISO[6]**	O
20									**SPI_MOSI[7]**	I
21									**SPI_MISO[7]**	O
22-									Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

Table 65: SPI slave mapping for PDI1, bidirectional (0x0192[6]=1)

Relative position	x1 0x0190[7]=1 0x0192[1:0]=00		x2 0x0190[7]=1 0x0192[1:0]=01		x4 0x0190[7]=1 0x0192[1:0]=10		x8 0x0190[7]=1 0x0192[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
0	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I	**SPI_CLK**	I
1	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I	SPI_SEL	I
2	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD
3	SPI_MISO_ENA	O	SPI_MISO_ENA	O	SPI_MISO_ENA	O	SPI_MISO_ENA	O
4	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O	SPI_IRQ	O
5	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I	SPI_SEL_MISO	I
6	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O	EEPROM_ LOADED	O
7	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I	SPI_SEL_MOSI	I
8	Unused. Available for lower priority function blocks		**SPI_D[1]**	BD	**SPI_D[1]**	BD	**SPI_D[1]**	BD
9			Unused. Available for lower priority function blocks		**SPI_D[2]**	BD	**SPI_D[2]**	BD
10					**SPI_D[3]**	BD	**SPI_D[3]**	BD
11					Unused. Available for lower priority function blocks		**SPI_D[4]**	BD
12							**SPI_D[5]**	BD
13							**SPI_D[6]**	BD
14							**SPI_D[7]**	BD
15-							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.3.3 SPI master PDI1

Table 66: SPI master number of MISO/MOSI signals

Number of MISO/MOSI signals	0x0192[1:0]	0x0192[3:2]
x1	00	00
x2	00	01
	01	00
x4	00	10
	01	01
	10	00
x8	00	11
	01	10
	10	01
	11	00

Table 67: SPI master mapping for PDI1, unidirectional (0x0192[6]=0)

Relative position	MISO/MOSI signals x1		x2		x4		x8	
	Signal	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.
0	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O
1	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O
2	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O	**SPI_MOSI[0]**	O
3	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I	**SPI_MISO[0]**	I
4	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I
5	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I
6	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O
7	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I
8	Unused. Available for lower priority function blocks		**SPI_MOSI[1]**	O	**SPI_MOSI[1]**	O	**SPI_MOSI[1]**	O
9			**SPI_MISO[1]**	I	**SPI_MISO[1]**	I	**SPI_MISO[1]**	I
10			Unused. Available for lower priority function blocks		**SPI_MOSI[2]**	O	**SPI_MOSI[2]**	I
11					**SPI_MISO[2]**	I	**SPI_MISO[2]**	O
12					**SPI_MOSI[3]**	O	**SPI_MOSI[3]**	I
13					**SPI_MISO[3]**	I	**SPI_MISO[3]**	O
14					Unused. Available for lower priority function blocks		**SPI_MOSI[4]**	O
15							**SPI_MISO[4]**	I
16							**SPI_MOSI[5]**	O
17							**SPI_MISO[5]**	I
18							**SPI_MOSI[6]**	O
19							**SPI_MISO[6]**	I
20							**SPI_MOSI[7]**	O
21							**SPI_MISO[7]**	I
22-							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

Table 68: SPI master mapping for PDI1, bidirectional (0x0192[6]=1)

Relative position	x1 0x0192[1:0]=00		x2 0x0192[1:0]=01		x4 0x0192[1:0]=10		x8 0x0192[1:0]=11	
	Signal	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**	**Signal**	**Dir.**
0	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O	**SPI_CLK**	O
1	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O	SPI_SEL	O
2	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD	**SPI_D[0]**	BD
3	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O	SPI_MOSI_ENA	O
4	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I	SPI_MISO_VALID	I
5	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I	SPI_SEL_MOSI/ START_MOSI	O/ I
6	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O	SPI_WD_ TRIGGER	O
7	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I	SPI_SEL_MISO/ START_MISO	O/ I
8	Unused. Available for lower priority function blocks		**SPI_D[1]**	BD	**SPI_D[1]**	BD	**SPI_D[1]**	BD
9			Unused. Available for lower priority function blocks		**SPI_D[2]**	BD	**SPI_D[2]**	BD
10					**SPI_D[3]**	BD	**SPI_D[3]**	BD
11					Unused. Available for lower priority function blocks		**SPI_D[4]**	BD
12							**SPI_D[5]**	BD
13							**SPI_D[6]**	BD
14							**SPI_D[7]**	BD
15-							Unused. Available for lower priority function blocks	

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.3.4 8 bit asynchronous multiplexed µController PDI1

The 8 bit asynchronous multiplexed µController PDI1 starts at pin PDI[24]

Table 69: Mapping of asynchronous multiplexed µController PDI1

Relative position	Async. multiplexed µController	
	8 bit 0x0180=0x0D 0x0192[3]=0	
	Signal	**Dir.**
PDI[23:0]	Unused by PDI1, available for PDI0 and GPIO.	
PDI[24]	**CS**	I
PDI[25]	**RD**	I
PDI[26]	**WR**	I
PDI[27]	BUSY	O
PDI[28]	IRQ	O
PDI[29]	ALE	I
PDI[30]	ADR[15]	I
PDI[31]	ADR[14]	I
PDI[32]	**ADR[13]**	I
PDI[33]	**ADR[12]**	I
PDI[34]	**ADR[11]**	I
PDI[35]	**ADR[10]**	I
PDI[36]	**ADR[9]**	I
PDI[37]	**ADR[8]**	I
PDI[38]	**AD[7]**	BD
PDI[39]	**AD[6]**	BD
PDI[40]	**AD[5]**	BD
PDI[41]	**AD[4]**	BD
PDI[42]	**AD[3]**	BD
PDI[43]	**AD[2]**	BD
PDI[44]	**AD[1]**	BD
PDI[45]	**AD[0]**	BD

NOTE: Bold signal names are used for signals which are required for the PDI. If one or more of the bold signals is not available in the final pinout, an ESC pin sharing error is detected.

4.14.4 Priority 4 (DC signal block)

The DC signal block starts at the next free possible start position (PDI[8], PDI[16], PDI[24], PDI[32], or PDI[40]), after PDI0 and PDI1 occupied their pins. Optionally, the DC signal block can be moved to a start position 8 pins lower (ESC configuration B4 register 0x0188[7]), then partially overlapping with PDI0/PDI1 signals. The DC signal block is only partially available then, but more pins remain available as GPIO.

NOTE: It is possible to have the DC signal block between PDI0 and PDI1 pins: the DC block will start at PDI[16], if this pin is free (not occupied by PDI0), regardless of PDI1. Since PDI1 cannot start at PDI[16], it will start at PDI[24] – leaving pins PDI[16:23] usable by the DC signal block.

Any signal of the DC signal block which is not enabled, will be available for the next priority (GPIO).

DC_LATCH/DC_SYNC

- DC_LATCH[3:2] are enabled by ESC configuration B4 register 0x0188[8].
- DC_SYNC[1:0] are enabled by ESC configuration B4 register 0x0188[9].
- DC_SYNC[3:2] are enabled by ESC configuration B4 register 0x0188[10].

When the pins DC_SYNC_LATCH[1:0] are configured to provide DC_LATCH[0] and DC_LATCH[1], all 4 LatchSignals and all 4 SyncSignals can be assigned to pins.

CPU_CLK_OUT2

CPU_CLK_OUT2 is configured by ESC configuration B4 register 0x0188[15:12]. When CPU_CLK_OUT2 is enabled, CPU_RESET is also enabled.

CPU_RESET

CPU_RESET is active low. With an external pull-down, a reset for a µController can be generated, which is active until the SII EEPROM is loaded and CPU_RESET is driven. CPU_RESET will be released sometime after CPU_CLK_OUT2 is generated.

Table 70: DC signal block

Pin	Start pin 8		Start pin 16		Start pin 24		Start pin 32		Start pin 40	
	Signal	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.	Signal	Dir.
PDI[0]	Unused. Available for other function blocks		Unused. Available for other function blocks		Unused. Available for other function blocks		Unused. Available for other function blocks		Unused. Available for other function blocks	
PDI[1]										
PDI[2]										
PDI[3]										
PDI[4]										
PDI[5]										
PDI[6]										
PDI[7]										
PDI[8]	DC_LATCH[2]	I								
PDI[9]	DC_LATCH[3]	I								
PDI[10]	DC_SYNC[0]	O								
PDI[11]	DC_SYNC[1]	O								
PDI[12]	DC_SYNC[2]	O								
PDI[13]	DC_SYNC[3]	O								
PDI[14]	CPU_RESET	O								
PDI[15]	CPU_CLK_OUT2	O								
PDI[16]	Unused. Available for lower priority function blocks		DC_LATCH[2]	I						
PDI[17]			DC_LATCH[3]	I						
PDI[18]			DC_SYNC[0]	O						
PDI[19]			DC_SYNC[1]	O						
PDI[20]			DC_SYNC[2]	O						
PDI[21]			DC_SYNC[3]	O						
PDI[22]			CPU_RESET	O						
PDI[23]			CPU_CLK_OUT2	O						
PDI[24]			Unused. Available for lower priority function blocks		DC_LATCH[2]	I				
PDI[25]					DC_LATCH[3]	I				
PDI[26]					DC_SYNC[0]	O				
PDI[27]					DC_SYNC[1]	O				
PDI[28]					DC_SYNC[2]	O				
PDI[29]					DC_SYNC[3]	O				
PDI[30]					CPU_RESET	O				
PDI[31]					CPU_CLK_OUT2	O				
PDI[32]					Unused. Available for lower priority function blocks		DC_LATCH[2]	I		
PDI[33]							DC_LATCH[3]	I		
PDI[34]							DC_SYNC[0]	O		
PDI[35]							DC_SYNC[1]	O		
PDI[36]							DC_SYNC[2]	O		
PDI[37]							DC_SYNC[3]	O		
PDI[38]							CPU_RESET	O		
PDI[39]							CPU_CLK_OUT2	O		
PDI[40]							Unused. Available for lower priority function blocks		DC_LATCH[2]	I
PDI[41]									DC_LATCH[3]	I
PDI[42]									DC_SYNC[0]	O
PDI[43]									DC_SYNC[1]	O
PDI[44]									CPU_RESET	O
PDI[45]									CPU_CLK_OUT2	O

NOTE: The DC signal block is different on start pin 40.

4.14.5 Priority 5 (GPIO)

The SPI slave PDI0 supports a ET1100 compatible GPIO mapping, please refer to chapter 4.14.2.2 regarding that mapping. This chapter is about the new GPIO features of the ET1150.

Most unused PDI[45:0] pins are available as general purpose input/output pins (GPI/GPO). The number of inputs (GPI)/outputs (GPO) can be configured using ESC configuration B4 register 0x0188[5:0]. If the register has the value x, PDI[0] to PDI[x-1] will be GPI. The remaining pins, PDI[x] to PDI[45], will be GPO. This will only apply to pins that are not used by a higher priority function, e.g. if you configure 10 inputs and PDI[7:0] are used for PDI0, only PDI[9:8] will be available as GPI.

There is no mapping/stuffing of GPIO pins to GPIO registers, each GPIO is mapped 1:1 based on its position (i.e., PDI[17] GPIO will always map to GPIO output register 0x0F12[2], or to GPIO input register 0x0F1A[2]).

GPO[x]
General purpose output signals.

GPI[x]
General purpose input signals.

Table 71: GPIO mapping (not ET1100 compatible: 0x0140[7:0]≠0x05 or 0x0144[0]=1 or 0x0150[7]=0)

PDI signal	0x0188[5:0]=0 only GPO		0x0188[5:0]=x any mix of GPIO		0x0188[5:0]=46 only GPI		Corresponding register	
	Signal	Dir.	Signal	Dir.	Signal	Dir.	GPO register	GPI register
PDI[0]	GPO[0]	O	GPI[x-1:0]	I	GPI[0]	I	0x0F10[0]	0x0F18[0]
PDI[1]	GPO[1]	O			GPI[1]	I	0x0F10[1]	0x0F18[1]
PDI[2]	GPO[2]	O			GPI[2]	I	0x0F10[2]	0x0F18[2]
PDI[3]	GPO[3]	O			GPI[3]	I	0x0F10[3]	0x0F18[3]
PDI[4]	GPO[4]	O			GPI[4]	I	0x0F10[4]	0x0F18[4]
PDI[5]	GPO[5]	O			GPI[5]	I	0x0F10[5]	0x0F18[5]
PDI[6]	GPO[6]	O			GPI[6]	I	0x0F10[6]	0x0F18[6]
PDI[7]	GPO[7]	O			GPI[7]	I	0x0F10[7]	0x0F18[7]
PDI[8]	GPO[8]	O			GPI[8]	I	0x0F10[8]	0x0F18[8]
PDI[9]	GPO[9]	O			GPI[9]	I	0x0F10[9]	0x0F18[9]
PDI[10]	GPO[10]	O			GPI[10]	I	0x0F10[10]	0x0F18[10]
PDI[11]	GPO[11]	O			GPI[11]	I	0x0F10[11]	0x0F18[11]
PDI[12]	GPO[12]	O			GPI[12]	I	0x0F10[12]	0x0F18[12]
PDI[13]	GPO[13]	O			GPI[13]	I	0x0F10[13]	0x0F18[13]
PDI[14]	GPO[14]	O			GPI[14]	I	0x0F10[14]	0x0F18[14]
PDI[15]	GPO[15]	O			GPI[15]	I	0x0F10[15]	0x0F18[15]
PDI[16]	GPO[16]	O			GPI[16]	I	0x0F10[16]	0x0F18[16]
PDI[17]	GPO[17]	O			GPI[17]	I	0x0F10[17]	0x0F18[17]
PDI[18]	GPO[18]	O			GPI[18]	I	0x0F10[18]	0x0F18[18]
PDI[19]	GPO[19]	O			GPI[19]	I	0x0F10[19]	0x0F18[19]
PDI[20]	GPO[20]	O			GPI[20]	I	0x0F10[20]	0x0F18[20]
PDI[21]	GPO[21]	O			GPI[21]	I	0x0F10[21]	0x0F18[21]
PDI[22]	GPO[22]	O			GPI[22]	I	0x0F10[22]	0x0F18[22]
PDI[23]	GPO[23]	O			GPI[23]	I	0x0F10[23]	0x0F18[23]
PDI[24]	GPO[24]	O	GPO[45:x]	O	GPI[24]	I	0x0F10[24]	0x0F18[24]
PDI[25]	GPO[25]	O			GPI[25]	I	0x0F10[25]	0x0F18[25]
PDI[26]	GPO[26]	O			GPI[26]	I	0x0F10[26]	0x0F18[26]
PDI[27]	GPO[27]	O			GPI[27]	I	0x0F10[27]	0x0F18[27]
PDI[28]	GPO[28]	O			GPI[28]	I	0x0F10[28]	0x0F18[28]
PDI[29]	GPO[29]	O			GPI[29]	I	0x0F10[29]	0x0F18[29]
PDI[30]	GPO[30]	O			GPI[30]	I	0x0F10[30]	0x0F18[30]
PDI[31]	GPO[31]	O			GPI[31]	I	0x0F10[31]	0x0F18[31]
PDI[32]	GPO[32]	O			GPI[32]	I	0x0F10[32]	0x0F18[32]
PDI[33]	GPO[33]	O			GPI[33]	I	0x0F10[33]	0x0F18[33]
PDI[34]	GPO[34]	O			GPI[34]	I	0x0F10[34]	0x0F18[34]
PDI[35]	GPO[35]	O			GPI[35]	I	0x0F10[35]	0x0F18[35]
PDI[36]	GPO[36]	O			GPI[36]	I	0x0F10[36]	0x0F18[36]
PDI[37]	GPO[37]	O			GPI[37]	I	0x0F10[37]	0x0F18[37]
PDI[38]	GPO[38]	O			GPI[38]	I	0x0F10[38]	0x0F18[38]
PDI[39]	GPO[39]	O			GPI[39]	I	0x0F10[39]	0x0F18[39]
PDI[40]	GPO[40]	O			GPI[40]	I	0x0F10[40]	0x0F18[40]
PDI[41]	GPO[41]	O			GPI[41]	I	0x0F10[41]	0x0F18[41]
PDI[42]	GPO[42]	O			GPI[42]	I	0x0F10[42]	0x0F18[42]
PDI[43]	GPO[43]	O			GPI[43]	I	0x0F10[43]	0x0F18[43]
PDI[44]	GPO[44]	O			GPI[44]	I	0x0F10[44]	0x0F18[44]
PDI[45]	GPO[45]	O			GPI[45]	I	0x0F10[45]	0x0F18[45]

NOTE: Some PDI pins might not be available for GPIO because they are used for a higher priority function. Output values are ignored for such pins, and input value is always zero.

4.14.5.1 Bidirectional GPIO

It is also possible to use up to 32 bidirectional GPIO pins on PDI[31:0], enabled by ESC configuration B4 register 0x0188[6]. In that case, GPIO output register 0x0F10[31:0] is used as output data, while 0x0F10[63:32] is used as output enable.

PDI[45:32] are fixed to general purpose input when bidirectional GPIOs are used.

Table 72: Bidirectional GPIO driver

Pin	Output enable 0x0F10[63:32] = 0		Output enable 0x0F10[63:32] = 1	
	Output data 0x0F10[31:0] = 0	Output data 0x0F10[31:0] = 1	Output data 0x0F10[31:0] = 0	Output data 0x0F10[31:0] = 1
PDI[31:0]	Pin is high-impedance	Pin is pulled down. Do not enable for PDI pins which are used for physical port 2 or 3.	Pin is driven low	Pin is driven high
PDI[45:32]	GPI[45:32]			

GPIO[x]
Bidirectional general purpose signals.

4.15 JTAG signals

Table 73: JTAG signals

Pin	Pin		Signal		Configuration signal	Internal PU/PD
	Name	Dir.	Signal	Dir.		
E4	JTAG_TCK	I	JTAG_TCK	I		WPD
F3	JTAG_TMS	I	JTAG_TMS	I		WPU
F4	JTAG_TDO	O	JTAG_TDO	O		
H9	JTAG_TDI	I	JTAG_TDI	I		WPU

JTAG_TCK
JTAG clock signal

JTAG_TMS
JTAG select signal

JTAG_TDO
JTAG data output signal

JTAG_TDI
JTAG data input signal

5 Ethernet interface

The ET1150 is connected with Ethernet PHYs using MII, or RGMII interfaces. MII is recommended since the PHY delay (and delay jitter) is smaller in comparison to RGMII.

5.1 PHY management interface

The PHY management interface of the ET1150 has the following signals:

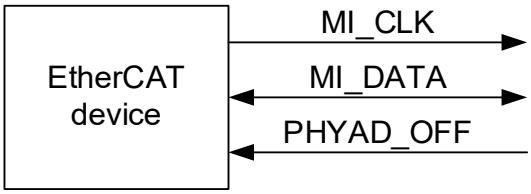

Figure 16: PHY management interface signals

Table 74: PHY management interface signals

Signal	Direction	Description
MI_CLK	OUT	Management interface clock (alias MCLK)
MI_DATA	BIDIR	Management interface data (alias MDIO)
PHYAD_OFF	IN	Configuration: PHY address offset

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

5.1.1 PHY address configuration

The ET1150 addresses Ethernet PHYs using logical port number (or PHY address register value) plus PHY address offset. Ideally, the Ethernet PHY addresses should correspond with the logical port number, so PHY addresses 0-3 are used.

Refer to section I for more details about PHY addressing.

The ET1150 supports four PHY address offset configurations, either 0, 1, 16, or 17. The ET1150 expects logical port 0 to have PHY address 0 plus PHY address offset (and so on):

Table 75: PHY configuration (logical port)

ET1150 PHY address offset	PHY port 0 configured address	PHY port 1 configured address	PHY port 2 configured address	PHY port 3 configured address
0	0	1	2	3
1	1	2	3	4
16	16	17	18	19
17	17	18	19	20

5.1.2 MI link detection and configuration

MI link detection and configuration checks and writes the following PHY registers.

Table 76: MI link detection for 100Base-TX

PHY register	Read mask	Read value	Write value	Comment
0	0xDEA0	0x1000	0x3300	Write only if any register does not have expected value
1	0x403D	0x402D		Check link up status
4	0x5FF	0x0101	0x0101	Advertise 100BaseTX full duplex only
5	0x6101	0x4101		Check auto negotiation finished
9 (port 0)	0xE300	0x0000	0x0000	Prefer slave
9 (port 1-3)	0xE300	0x0000	0x0400	Prefer master
15				Check for support of 1000Base-T

Table 77: MI link detection for 100Base-FX

PHY register	Read mask	Read value	Write value	Comment
0	0xFFC0	0x2100	0x2100	Write only if any register does not have expected value
1	0x4004	0x4004		Check link up status

5.1.3 MII management timing specifications

For MII management interface timing diagrams refer to section I.

Table 78: PHY management timing characteristics

Parameter	Min	Typ	Max	Comment
PRELIMINARY TIMING				
t_{Clk}		a) ~ 1.44 µs b) 400 ns		MI_CLK period a) No MI link detection: $f_{Clk} \approx 700$ kHz b) With MI link detection: $f_{Clk} = 2.5$ MHz
t_{Write}		a) ~ 92.16 µs b) ~ 25.6 µs		MI write access time a) No MI link detection b) With MI link detection
t_{Read}		a) ~ 91.44 us b) ~ 25.4 µs		MI read access time a) No MI link detection b) With MI link detection

5.2 PHY reset

PHY reset signals are available if FX mode is configured. The individual PHY_RES reset signals for each port have this functionality:

- Keep the PHY in reset while the ET1150 is in reset.
- If enhanced link detection is active, a reset for the PHY and the transceiver is applied when too many RX_ERR are received.
- The PHY reset is extended to a minimum time $t_{PHY\_Reset\_Out}$.

Table 79: PHY reset signals

Signal	Direction	Description
PHY_RES	OUT	PHY reset (act. low), resets PHY while ESC is in reset state, and, for FX PHYs, if enhanced link detection detects a lost link

NOTE: A pull-down resistor is required for PHY_RES to hold the PHY in reset state while the ET1150 is unpowered.

NOTE: If FX mode is not used, you must use the global reset of the ESC to reset the PHYs.

5.2.1 PHY reset timing specifications

Table 80: PHY reset timing characteristics

Parameter	Min	Typ	Max	Comment
PRELIMINARY TIMING				
$t_{PHY\_Reset\_Out}$	1.2 ms			PHY reset active time

5.3 MII interface

The MII interfaces of the ET1150 are optimized for low processing/forwarding delays by omitting a transmit FIFO. To allow this, the ET1150 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 ET1150:

- The clock source of the PHYs is either CLK25OUT1/2 of the ET1150, or the clock signal that is connected to OSC_IN if a quartz oscillator is used.
- The TX_CLK signal of the PHYs is usually connected to the ET1150, otherwise connected to GND
- The ET1150 can be configured to use the MII management interface for link detection and link configuration (ESC configuration B5 register 0x0182:0x0183[11:8]).

For details about the ESC MII interface refer to section I.

5.3.1 MII interface signals

The MII interface of the ET1150 has the following signals:

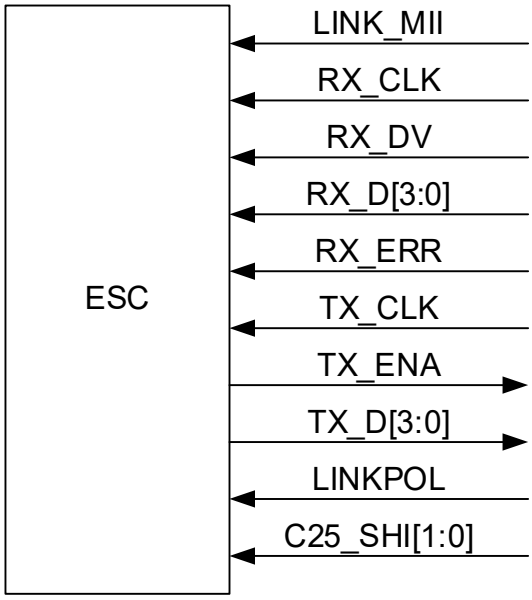

Figure 17: MII interface signals

Table 81: 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_CLK	IN	Transmit clock (automatic TX shift compensation)
TX_ENA	OUT	Transmit enable (alias TX_EN)
TX_D[3:0]	OUT	Transmit data (alias TXD)
LINKPOL	IN	Configuration: LINK_MII polarity
C25_SHI[1:0]	IN	Configuration: Manual TX shift compensation

5.3.2 TX shift compensation

Since ET1150 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 ET1150. Therefore, a TX FIFO inside the ET1150 is not necessary, and the transmit latency is reduced.

The phase shift between TX_CLK and TX_ENA/TX_D[3:0] is compensated by the ET1150 by an appropriate value for TX shift, which will delay TX_ENA/TX_D[3:0] by 0, 10, 20, or 30 ns.

There are two options to determine the appropriate value for TX shift:

- Automatic TX shift compensation (recommended)
- Manual TX shift compensation

5.3.2.1 Automatic TX shift compensation

The ESC detects the required phase shift automatically, based on the TX_CLK signal. This is not the same function as a TX FIFO, the clock source still must be the same. But the PHYs are not required to have the same fixed phase relation each time they are powered on/establish a link.

Connect TX_CLK with the PHY, to use automatic TX shift compensation. Set C25_SHI[1:0]=00, otherwise additional delay would be added. No further adjustments are required.

5.3.2.2 Manual TX shift compensation

A static signal delay for TX_EN/TXD by 0/10/20/30 ns is configured, 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 the clock input of the PHY. The phase relation must be the same each time the PHYs are powered up/establish a link. The TX_CLK signal is not connected to the ESC.

Connect TX_CLK to GND to disable automatic TX shift compensation. TX shift configuration is made by C25_SHI[1:0]. It is recommended to provide all C25_SHI options on your PCB, to allow for later adjustments.

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, it may contain relaxed timing requirements). 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 generated nearly at the same time.

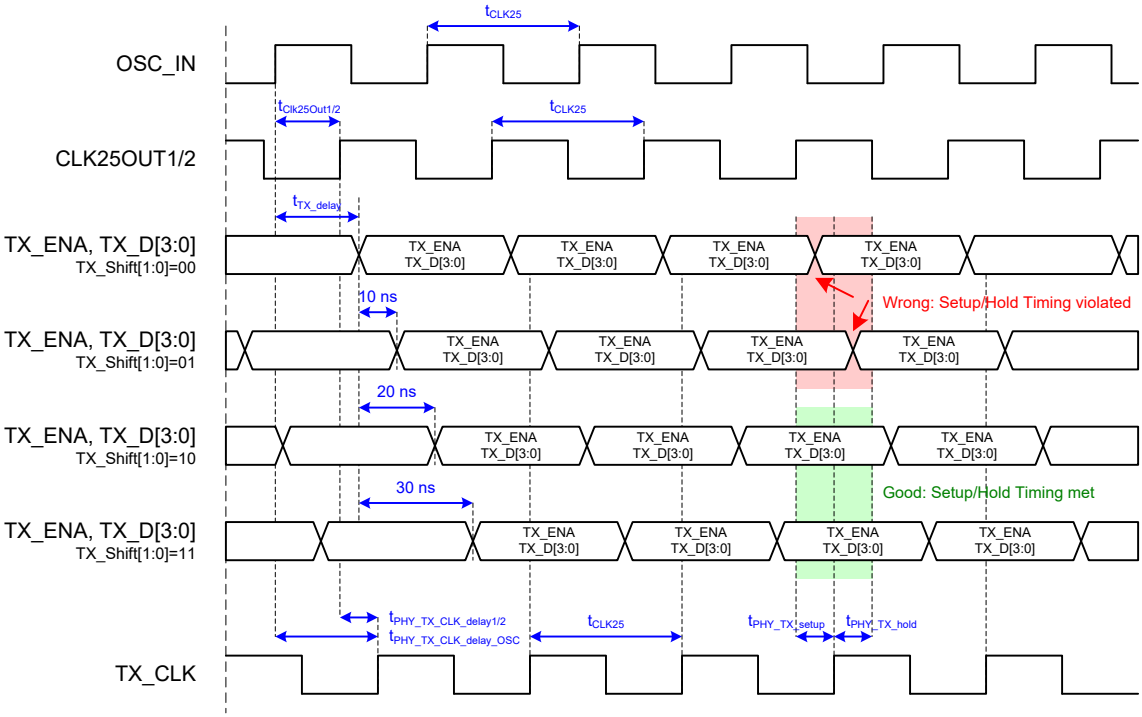

Figure 18: TX shift timing diagram

Table 82: 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)

5.3.3 Timing specifications

Table 83: MII timing characteristics

Parameter	Min	Typ	Max	Comment
PRELIMINARY TIMING				
t_{RX_CLK}	40 ns ± 100 ppm			RX_CLK period (100 ppm with maximum FIFO size only)
t_{RX_setup}	6 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

NOTE: For MI timing diagrams refer to section I.

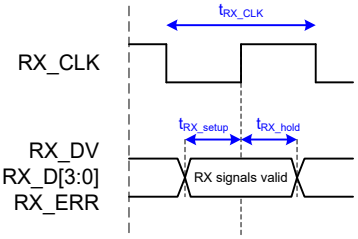

Figure 19: MII timing RX signals

5.4 RGMII interface

The ET1150 supports RGMII at 100 Mbit/s. Nevertheless, MII is recommended since the PHY delay (and delay jitter) is smaller in comparison to RGMII.

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

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

Additional information regarding the ET1150:

- The ET1150 can be configured to use the MII management interface for link detection and link configuration (ESC configuration B5 register 0x0182:0x0183[11:8]).
- A Gigabit Ethernet PHY has to be restricted to establish only 100 Mbit/s links (e.g. by using MI link detection and configuration).
- The ET1150 only accepts 100 Mbit/s full-duplex links.
- The ET1150 uses internal delay for RX and TX (RGMII-ID). Additional PCB delay for RX_CLK/TX_CLK is not required. The PHYs should not use RGMII-ID, although it is possible since the link speed is only 100 Mbit/s.
- If RGMII in-band status is available, set LINK_RGMII(x) to "no link", depending on link polarity. Link detection is performed based on RGMII in-band status signaling (recommended).

For details about the ESC RGMII interface refer to section I.

5.4.1 RGMII interface signals

The RGMII interface of the ET1150 has the following signals:

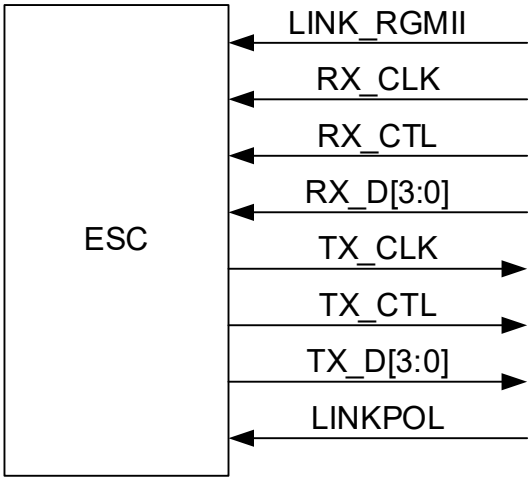

Figure 20: RGMII interface signals

Table 84: RGMII interface signals

Signal	Direction	Description
LINK_RGMII	IN	Without RGMII in-band status: input signal provided by the PHY if a 100 Mbit/s (Full Duplex) link is established With RGMII in-band status: set to “no link”, according to LINKPOL configuration
RX_CLK	IN	Receive clock (alias RXC)
RX_CTL	IN	Receive data valid
RX_D[3:0]	IN	Receive data (alias RD)
TX_CLK	OUT	Transmit clock (alias TXC)
TX_CTL	OUT	Transmit enable
TX_D[3:0]	OUT	Transmit data (alias TD)
LINKPOL	IN	Configuration: LINK_RGMII polarity

5.4.2 Timing specifications

Table 85: RGMII timing characteristics

Parameter	Min	Typ	Max	Comment
PRELIMINARY TIMING				
t_{RX_CLK}	40 ns ± 100 ppm			RX_CLK period (100 ppm with maximum FIFO size only)
t_{RX_setup}	-1 ns			RX_CTL/RX_D[3:0] valid before rising/falling edge of RX_CLK
t_{RX_hold}	15 ns			RX_CTL/RX_D[3:0] valid after rising/falling edge of RX_CLK
t_{TX_CLK}		40 ns		TX_CLK period
t_{TX_valid}	1.5 ns			TX_CTL/TX_D[3:0] valid before rising/falling edge of TX_CLK
$t_{TX\_invalid}$			14.5 ns	TX_CTL/TX_D[3:0] invalid before rising/falling edge of TX_CLK

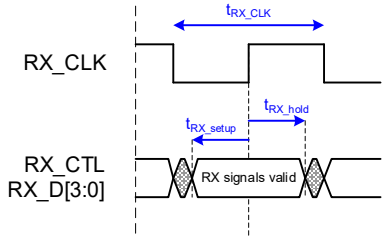

Figure 21: RGMII timing RX signals

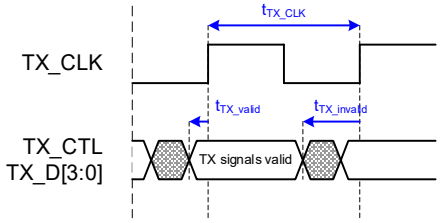

Figure 22: RGMII timing TX signals

6 PDI description

Table 86: Available PDIs for ET1150

PDI type	PDI name	ET1150	ET1100
	PDI0 control register 0x0140		
0x00	Interface deactivated (no PDI)	x	x
0x01	4 digital input		
0x02	4 digital output		
0x03	2 digital input and 2 digital output		
0x04	Digital I/O	x	x
0x05	SPI slave	x	x
0x06	Oversampling I/O		
0x07	Bridge port 3		
0x08	16/64 Bit asynchronous microcontroller interface	16	x
0x09	8/32 Bit asynchronous microcontroller interface	8	x
0x0A	16/64 Bit synchronous microcontroller interface	16	x
0x0B	8/32 Bit synchronous microcontroller interface	8	x
0x0C	16/64 Bit multiplexed asynchronous microcontroller interface	16	
0x0D	8/32 Bit multiplexed asynchronous microcontroller interface	8/32	
0x0E	16/64 Bit multiplexed synchronous microcontroller interface		
0x10	32 digital input/0 digital output		
0x11	24 digital input/8 digital output		
0x12	16 digital input/16 digital output		
0x13	8 digital input/24 digital output		
0x14	0 digital input/32 digital output		
0x15	SPI master	x	
0x80	On-chip bus (Avalon, OPB, PLB, AXI)		
Others	Reserved		
	PDI1 control register 0x0180		
0x00	Interface deactivated (no PDI)	x	
0x04	Digital I/O	x	
0x05	SPI slave	x	
0x08	16/64 Bit asynchronous microcontroller interface		
0x09	8/32 Bit asynchronous microcontroller interface		
0x0A	16/64 Bit synchronous microcontroller interface		
0x0B	8/32 Bit synchronous microcontroller interface		
0x0C	16/64 Bit multiplexed asynchronous microcontroller interface		
0x0D	8/32 Bit multiplexed asynchronous microcontroller interface	8	
0x0E	16/64 Bit multiplexed synchronous microcontroller interface		
0x15	SPI master	x	
0x80	On-chip bus (Avalon, OPB, PLB, AXI)		
Others	Reserved		

6.1 PDI deactivated

The PDI is deactivated with PDI type 0x00. The PDI pins are not driven (high impedance).

6.2 Digital I/O interface

6.2.1 Interface

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

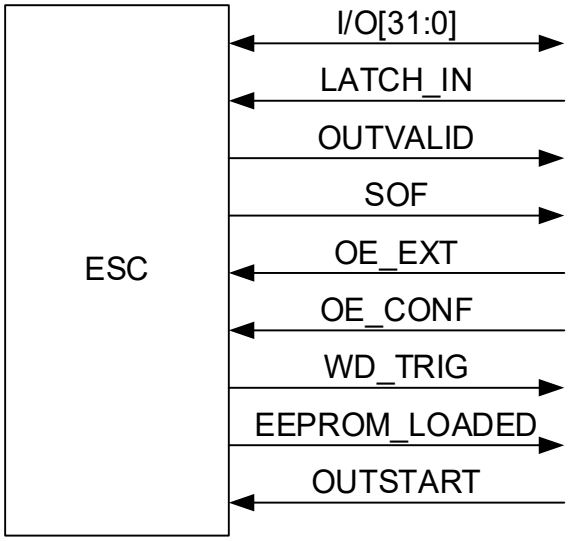

Figure 23: ET1150 digital I/O signals

Table 87: ET1150 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	configurable
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
OUT_START	IN	Start output cycle	act. high

6.2.2 Configuration

The digital I/O interface is selected with PDI type 0x04 in the PDI0 control register 0x0140, or PDI1 control register 0x0180. It supports different configurations, which are located in registers 0x0150 – 0x0153 / 0x0190 – 0x0193.

6.2.3 Digital inputs

Digital input values of PDI0 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 of PDI1 are transferred to the digital I/O input data register 0x0F08:0x0F0B.

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 0x1000:0x1003/0x0F08:0x0F0B 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 SyncSignal 0 events.
- Digital inputs are sampled at distributed clocks SyncSignal 1 events.

For inputs based on distributed clock SyncSignals, SyncSignal generation must be activated (register 0x0981). SYNC output is not necessary (register 0x0151). SyncSignal pulse length (registers 0x0982:0x0983) should not be set to 0, because acknowledging of SyncSignal events is not possible with digital I/O PDI. Sample time is the beginning of the SyncSignal event.

6.2.4 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 5 ways:

- Digital outputs are updated at the end of each EtherCAT frame (EOF mode).
- Digital outputs are updated at a rising edge of OUT_START signal (OUT_START mode)
- Digital outputs are updated with distributed clocks SyncSignal 0 events.
- Digital outputs are updated with distributed clocks SyncSignal 1 events.
- 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 outputs based on distributed clock SyncSignals, SyncSignal generation must be activated (register 0x0981). SYNC output is not necessary (register 0x0151). SyncSignal pulse length (registers 0x0982:0x0983) should not be set to 0, because acknowledging of SyncSignal events is not possible with digital I/O PDI. Output time is the beginning of the SyncSignal 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.

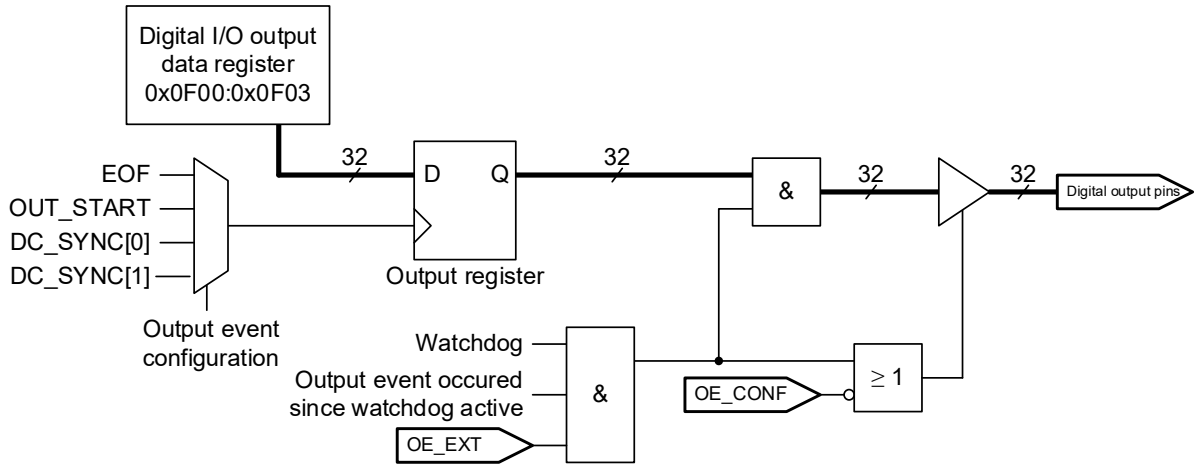

Figure 24: ET1150 digital output principle schematic

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 behavior has to be taken into account when using digital output signals.

6.2.5 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).

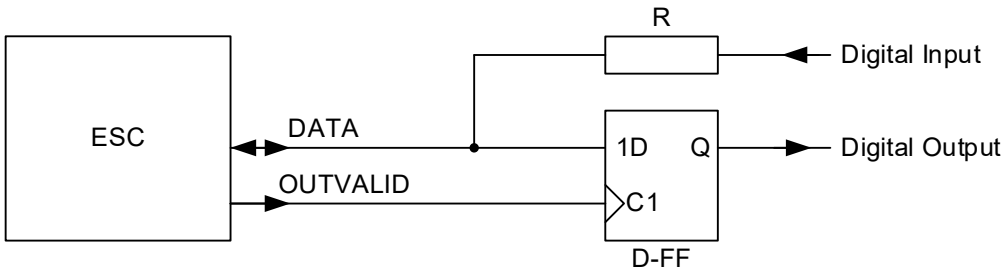

Figure 25: Bidirectional mode: input/output connection (R=4.7 kΩ recommended)

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.

6.2.6 Output enable/output configuration

The ET1150 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 88: 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.

6.2.7 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.

Immediate output reset after watchdog timeout is not available if OUTVALID mode set to watchdog trigger (0x0150[1]=1/0x0190[1]=1).

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.

6.2.8 SOF

SOF indicates the start of an Ethernet/EtherCAT frame. It is asserted shortly after RX_DV=1. 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.

6.2.9 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, 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. The polarity of OUTVALID is configurable using SII EEPROM A1[0]/PDI0 configuration register 0x0150[0].

6.2.10 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.

6.2.11 Timing specifications

Table 89: Digital I/O timing characteristics ET1150

Parameter	Min	Max	Comment
		PRELIMINARY TIMING	
t_{DATA_setup}	0 ns		Input data valid before LATCH_IN
t_{DATA_hold}	7 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
$t_{input\_event\_delay}$	320 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}	5 ns	45 ns	WD_TRIG high time
$t_{DATA\_to\_WD\_TRIG}$		45 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
$t_{BIDIR\_event\_delay}$	320 ns		Bidirectional mode: time between consecutive input and output events

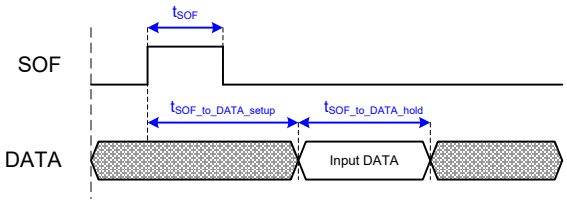

Figure 26: Digital input: input data sampled at SOF, input data can be read in the same frame

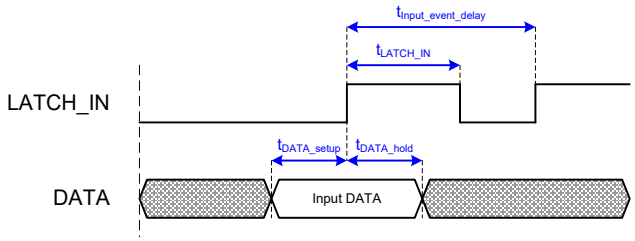

Figure 27: Digital input: input data sampled with LATCH_IN

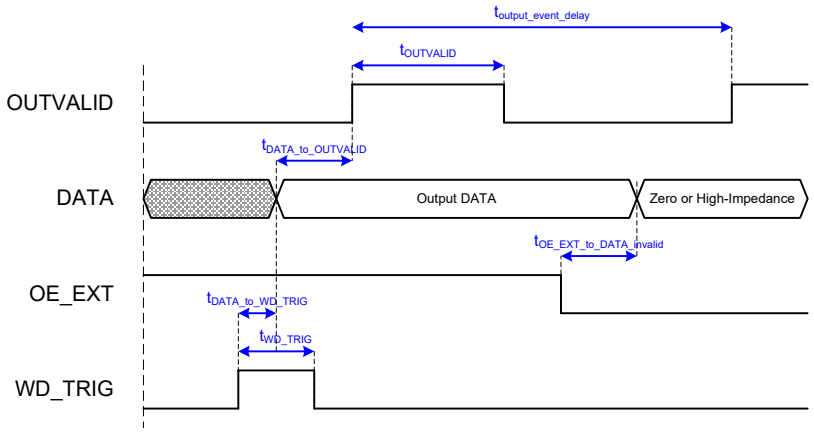

Figure 28: Digital output timing

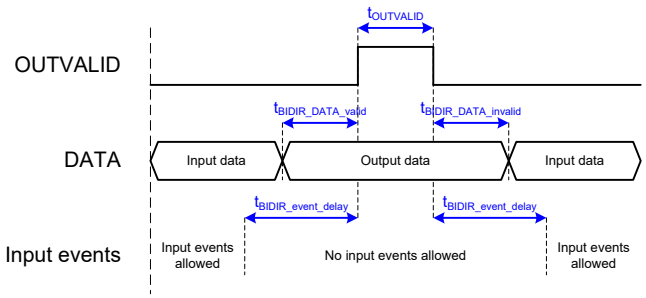

Figure 29: Bidirectional mode timing

6.3 SPI slave interface

6.3.1 Interface

The SPI slave PDI is selected with PDI type 0x05. The signals of the SPI slave interface are:

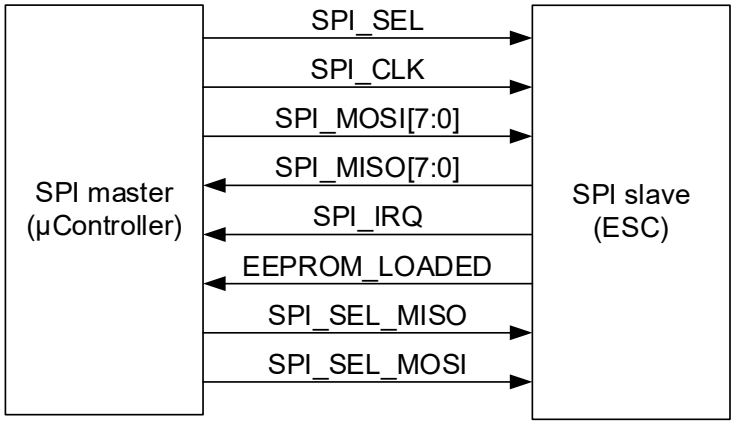

Figure 30: SPI master and slave interconnection (unidirectional)

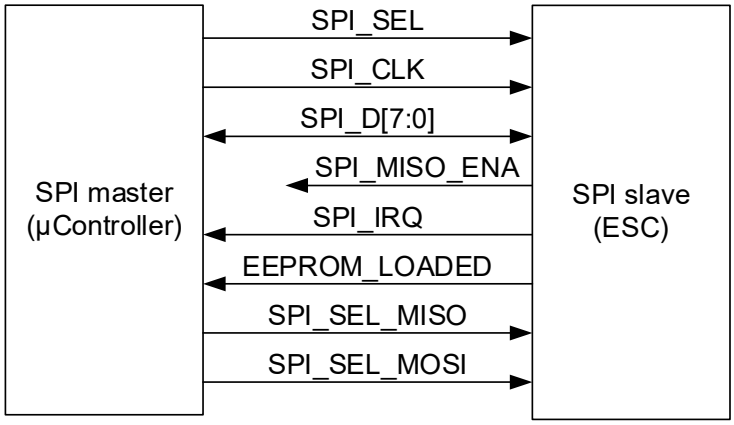

Figure 31: SPI master and slave interconnection (bidirectional)

Table 90: SPI slave signals

Signal	Direction		Description	Signal polarity
SPI_CLK	IN	(master → slave)	SPI clock	
SPI_SEL	IN	(master → slave)	SPI random access chip select	Typical: act. low
SPI_SEL_MISO	IN	(master → slave)	MISO chip select	Same as SPI_SEL
SPI_SEL_MOSI	IN	(master → slave)	MOSI chip select	
SPI_MOSI[7:0]	IN	(master → slave)	MOSI data (unidir)	
SPI_MISO[7:0]	OUT	(slave → master)	MISO data (unidir)	
SPI_MISO_ENA	OUT		MISO driver enabled	act. high
SPI_D[7:0]	BIDIR		MISO/MOSI data (bidirectional)	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

6.3.2 Configuration

The SPI slave interface is selected with PDI type 0x05 in the PDI0 control register 0x0140, or PDI1 control register 0x0180. It supports different data widths, operation modes, and signal polarities. The SPI slave configuration is located in register 0x0150:0x0153 / 0x0190:0x0193.

6.3.3 Access types

A regular SPI access to any ESC register or ESC RAM uses SPI_SEL as select signal. This select signal indicates that a address/command phase will be used before data is exchanged.

The ET1150 also supports direct SyncManager access using SPI_SEL_MISO and SPI_SEL_MOSI select signals. When these signals are used, no address phase or command phase is required, instead, MISO or MOSI data from a specific SyncManager per direction is immediately accessible.

SPI_SEL_MISO and SPI_SEL_MOSI might be activated simultaneously (within a defined period of time), indicating bidirectional transfer.

But, SPI_SEL must never be activated overlapping with either SPI_SEL_MISO or SPI_SEL_MOSI.

6.3.4 SPI_SEL random access

Each SPI_SEL random 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, MISO data is presented by the SPI slave (read command) or MOSI 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 de-asserting 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 x1 data width, or only once for x8 data width. In each clock cycle, both master and slave transmit 1/2/4/8 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 bits of a byte are transmitted first, the least significant bits last, the byte order is low byte first. EtherCAT devices use little endian byte ordering.

6.3.4.1 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 91: 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 (not recommended)
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

6.3.4.2 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 92: Address modes without wait state byte (read access without wait state byte)

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

Table 93: Address modes for read access with wait state byte

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 command: 011b	A[4:0] CMD0[2:0]	address bits [4:0] 3 byte addressing: 110b
2	0xFF	wait state byte	A[15:13] CMD1[2:0] res[1:0]	address bits [15:13] read command: 011b two reserved bits, set to 00b
3	D0[7:0]	MISO data byte 0	0xFF	wait state byte
4	D1[7:0]	MISO data byte 1	D0[7:0]	MISO data byte 0
5 ff.	D2[7:0]	MISO data byte 2	D1[7:0]	MISO data byte 1

6.3.4.3 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_MISO:

Table 94: Interrupt request register transmission

Byte	2 byte address mode			3 byte address mode		
	SPI_MOSI	SPI_MISO		SPI_MOSI	SPI_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

The order of PDI interrupt request registers 0x0220 and 0x0221 can be swapped by SPI slave configuration 0x0150[6], so SyncManager interrupts can be polled by a short one byte SPI access (terminated after the first byte).

6.3.4.4 Write access

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

6.3.4.5 Read access

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

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, using different read commands:

- The SPI master waits 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_MOSI.
  Using the wait state byte is recommended.

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

6.3.4.6 Bidirectional operation with SPI_SEL

The SPI slave also supports bidirectional operation using SPI_SEL. The protocol is again the same, just with a few restrictions:

- The interrupt request register values are not available as MISO data during address phase.
- Every read command requires a wait state byte, the normal read without wait state byte is not supported. The wait state byte is required for bus driver change (SPI master drives during address phase, SPI slave drives during data phase).
- **Read termination is not supported/available. The SPI slave internally fetches the next data bytes, even if they are not transferred later.**

This has to be taken into account e.g. for adjacent SyncManagers (reading the last byte of a SyncManager may also cause the chip reading the first by of the next SyncManager), or accidentally acknowledging interrupts (reading SyncManager status register accidentally acknowledges SyncManager enable interrupts). If this cannot be avoided, PDI function acknowledge by write may be configured.

The SPI_MISO_ENA signal might be used for drivers in the data path between master and slave, it is active while the SPI slave drives SPI_D.

6.3.4.7 SPI masters with 2 byte and 4 byte minimum access

The address phase can be extended by an arbitrary number of address extension bytes (command 0b110) to achieve a total access length that is supported by the SPI master. The address portion of the last address extension byte is used for the access.

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 95.

Table 95: 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).

6.3.5 SPI_SEL_MISO/SPI_SEL_MOSI configured SyncManager access

Two additional select signals become available when SPI slave configuration register 0x0150[7]=1. In contrast to the random access protocol used with SPI_SEL, the new select signals do not require an address phase/command phase. Instead, these signals allow immediately reading/writing SyncManager data, i.e., they are ideal for cyclic process data access.

Typically, a µController will use both SPI_SEL, and one or two of the additional select signals.

The corresponding SyncManagers for SPI_SEL_MISO/SPI_SEL_MOSI depend on the PDI (instance PDI0 or instance PDI1), and on the select signal:

Table 96: SyncManager used by SPI slave for SPI_SEL_MISO/SPI_SEL_MOSI

Access	PDI0	PDI1	SyncManger direction
SPI_SEL_MISO	SyncManager 15	SyncManager 13	ECAT write, PDI read
SPI_SEL_MOSI	SyncManager 14	SyncManager 12	ECAT read, PDI write

Each of the two select signals can be used from the moment when the corresponding SyncManager is activated by the master (there is no dependency on the AL status register 0x0130).

The SyncManager configuration does not only define the start address of an SPI access, it also defines the length. MISO data will be zero for any byte beyond the configured SyncManager buffer length. MOSI data will not be written beyond the SyncManager buffer length.

The SPI slave PDI will check SyncManager direction and activation before using it.

It is allowed and potentially beneficial to assert SPI_SEL_MISO and SPI_SEL_MOSI simultaneously. In unidirectional mode, MISO and MOSI data exchange happens during the same SPI_CLK cycles, maximizing throughput on the interface. In bidirectional mode, MISO and MOSI data exchange happens one after the other. Both select signals have to be active during the whole access (before first SPI_CLK edge until after last SPI_CLK edge), and they need to have the active edge within a short time window, which allows the ESC to align the required accesses. The length of the SyncManager buffers does not have to be identical for simultaneous assertion of SPI_SEL_MISO/SPI_SEL_MOSI.

6.3.5.1 SPI_SEL_MISO access

Between the active edge of SPI_SEL_MISO and the first data byte, 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 waits for the specified worst case internal read time t_{read} before the first clock edge. Depending on the SPI master, and the SPI_CLK frequency, the time between SPI_SEL and first clock edge might be sufficient for the ESC to fetch the data. In this case, this setting is preferred.
- The SPI master inserts one wait state byte before the first actual data byte. The wait state byte has to be configured using SPI slave extended configuration register 0x0152[2] / 0x0192[2]. The data presented during the wait state byte has to be ignored.
  Using the wait state byte is usually recommended.

6.3.5.2 Bidirectional operation with SPI_SEL_MISO/SPI_SEL_MOSI

Bidirectional operation can be selected when using SPI_SEL_MISO/SPI_SEL_MOSI. If only one of these two select signals is used, there is no difference to unidirectional mode (except for the pinout of course).

Simultaneous assertion of SPI_SEL_MISO and SPI_SEL _MOSI signals is also supported, as long as the active edge is within a short time window, which allows the ESC to detect the simultaneous access. First, the MISO data exchange will be executed. A turnaround byte without data exchange will then allow the bus driver to change, followed by the transfer of the MOSI data bytes. Optionally, the MISO data exchange starts with a wait state byte (configured by 0x0152[2]/0x192[2]).

This example diming diagram shows the byte ordering, when both SPI_SEL_MISO and SPI_SEL_MOSI are asserted. SPI_CLK is using x4 operation (2 clocks per byte), clock edge reference is not accurate.

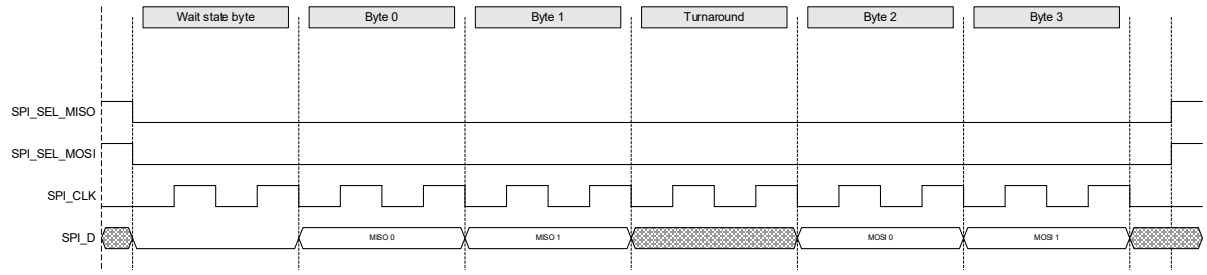

Figure 32: SPI slave bidirectional mode with simultaneous SEL_MISO/SEL_MOSI access

The SPI_MISO_ENA signal might be used for drivers in the data path between master and slave, it is active while the SPI slave drives SPI_D.

6.3.6 MISO/MOSI data width x1, x2, x4, x8

The SPI slave built in the ET1150 supports different data width modes: 1, 2, 4, or 8 bit. The data width is selected by SPI slave configuration, it cannot be selected during operation, and it does not change within an access.

The protocol used for 1 bit operation is also used for other data widths, only the number of SPI_CLK cycles is reduced.

6.3.7 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/SPI_SEL_MISO/SPI_SEL_MOSI 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 an SPI_SEL read access, the data phase was not terminated by setting SPI_MOSI to high for the last byte.
- For an SPI_SEL read access, additional bytes were read after termination of the access.
- Invalid combination of select signals SPI_SEL, SPI_SEL_MISO, SPI_SEL_MOSI
- SPI_SEL_MISO and SPI_SEL_MOSI active at the same time, but not simultaneously (too much skew)
- For SPI_SEL_MISO/SPI_SEL_MOSI access: SyncManager configuration is not valid

A wrong SPI access will have these consequences:

- The PDI error counter 0x030D/0x0340 will be incremented. For the first error, an SPI PDI error code will be stored in the PDI error code register 0x030E:0x030F/0x0341:0x0342, indicating details of the error reason.
- A status flag will indicate the error until the next access (not for SPI mode 0/2 with normal data out sample)
- For debugging, ESC configuration B4[11] can be used to enable a PDI error signal on the LED_RUN/LED_ERR pins, for PDI0/PDI1 respectively.

A wrong SPI access will also have these consequences, if PDI configuration 0x0150[7]=0 and PDI1 is not used:

- Registers will not take-over write data (nevertheless, RAM will be written).
- Special functions are not executed (e.g., SyncManager buffer switching).

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_MISO after the slave is selected (SPI_SEL or SPI_SEL_MISO) 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 SPI_SEL/SPI_SEL_MISO access just before the first clock cycle. The status flag will be high for a good access, and low for a wrong access. If you only want to get the status flag, prefer using SPI_SEL because SPI_SEL_MISO starts fetching the data internally (even without any SPI clock).

6.3.8 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.

6.3.9 Timing specifications

Table 97: SPI timing characteristics ET1150

Parameter	Min	Max	Comment
PRELIMINARY TIMING			
t_{CLK}	20 ns		SPI_CLK frequency (f_{CLK} ≤ 50 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		De-assertion 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}	60 ns		Only for read access between address/command and first data byte. Can be ignored if wait state bytes are used.
$t_{SEL\_to\_MISO\_valid}$		15 ns	Status/interrupt byte 0 bit 7 valid after SPI_SEL asserted
$t_{SEL\_to\_MISO\_invalid}$	0 ns		Status/interrupt byte 0 bit 7 invalid after SPI_SEL de-asserted
t_{STATUS_valid}	15 ns		Time until status of last access is valid. Can be ignored if status is not used.
t_{access_delay}	10 ns		Delay between SPI accesses
t_{MOSI_setup}	4 ns		SPI_MOSI valid before SPI_CLK edge
t_{MOSI_hold}	2 ns		SPI_MOSI valid after SPI_CLK edge
$t_{CLK\_to\_MISO\_valid}$	a) 20 ns b) 0 ns	a) 35 ns + 13 ns b) 13 ns	SPI_MISO valid after SPI_CLK edge a) x8 normal sample b) other
$t_{CLK\_to\_MISO\_invalid}$	0 ns		SPI_MISO invalid after SPI_CLK edge
$t_{EEPROM\_LOADED\_to\_access}$	0 ns		Time between EEPROM_LOADED and first access
t_{IRQ_delay}	0 ns		Internal delay between AL event and SPI_IRQ output to enable correct reading of the interrupt registers.

Table 98: Read/write timing diagram symbols

Symbol	Comment
A[15:0]	Address [15:0]
D0[7:0] D1[7:0]	Data byte 0 [7:0] Data byte 1 [7:0]
I0[7:0] I1[7:0] I2[7:0]	Interrupt request register 0x0220 [7:0] Interrupt request register 0x0221 [7:0] Interrupt request register 0x0222 [7:0]
C0[2:0] C1[2:0]	Command 0 [2:0] Command 1 [2:0] (3 byte addressing)
Status	0: last SPI access had errors 1: last SPI access was correct

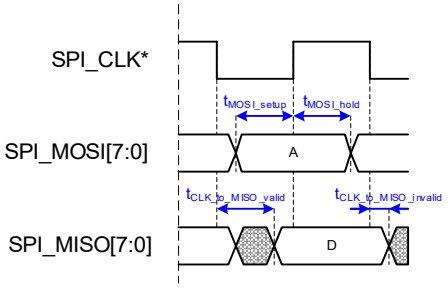

Figure 33: Basic SPI_MOSI/SPI_MISO timing (*refer to access diagrams for relevant edges of SPI_CLK)

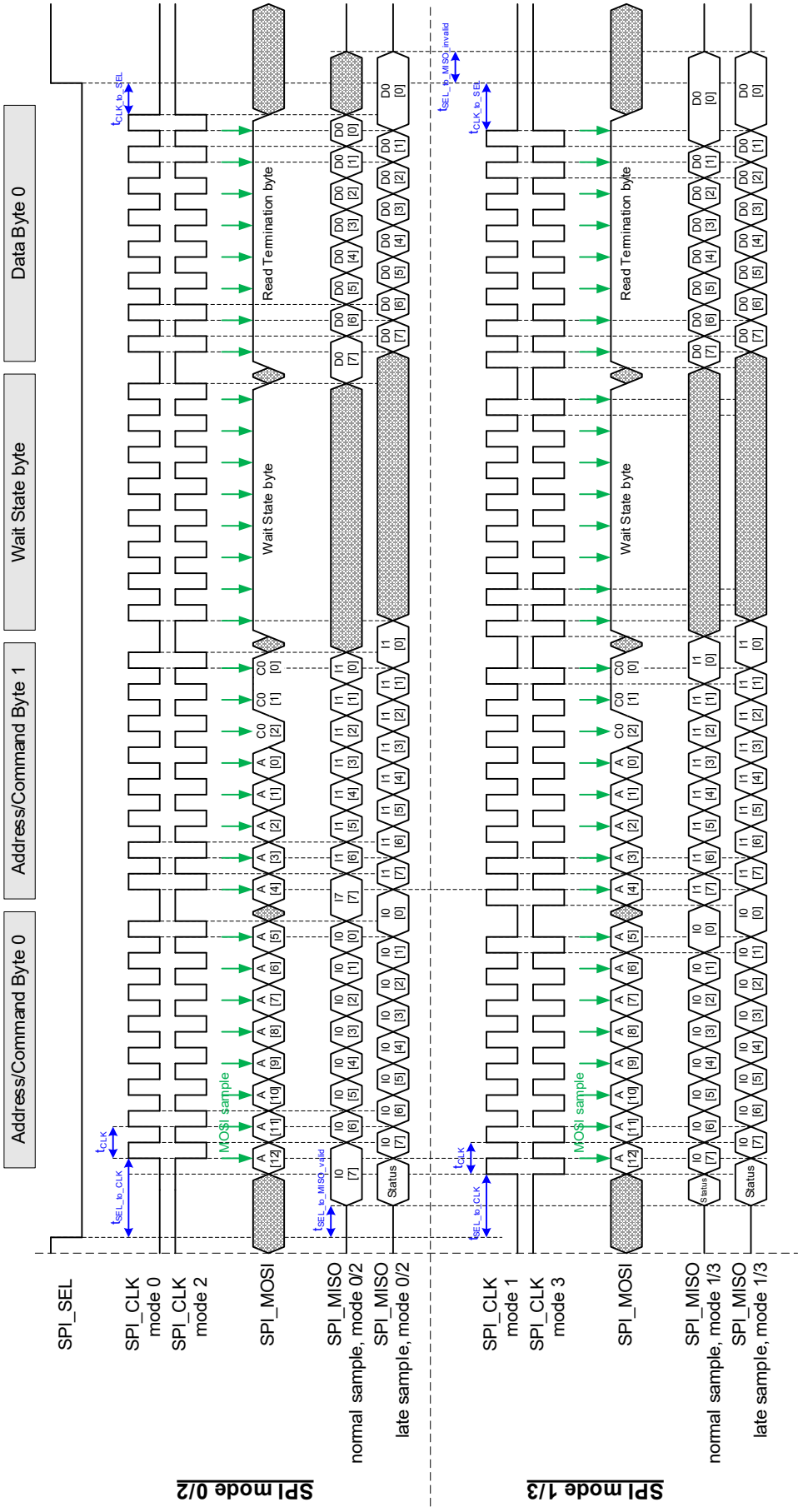

Figure 34: SPI_SEL random read access (2 byte addressing, 1 byte read data) with wait state byte

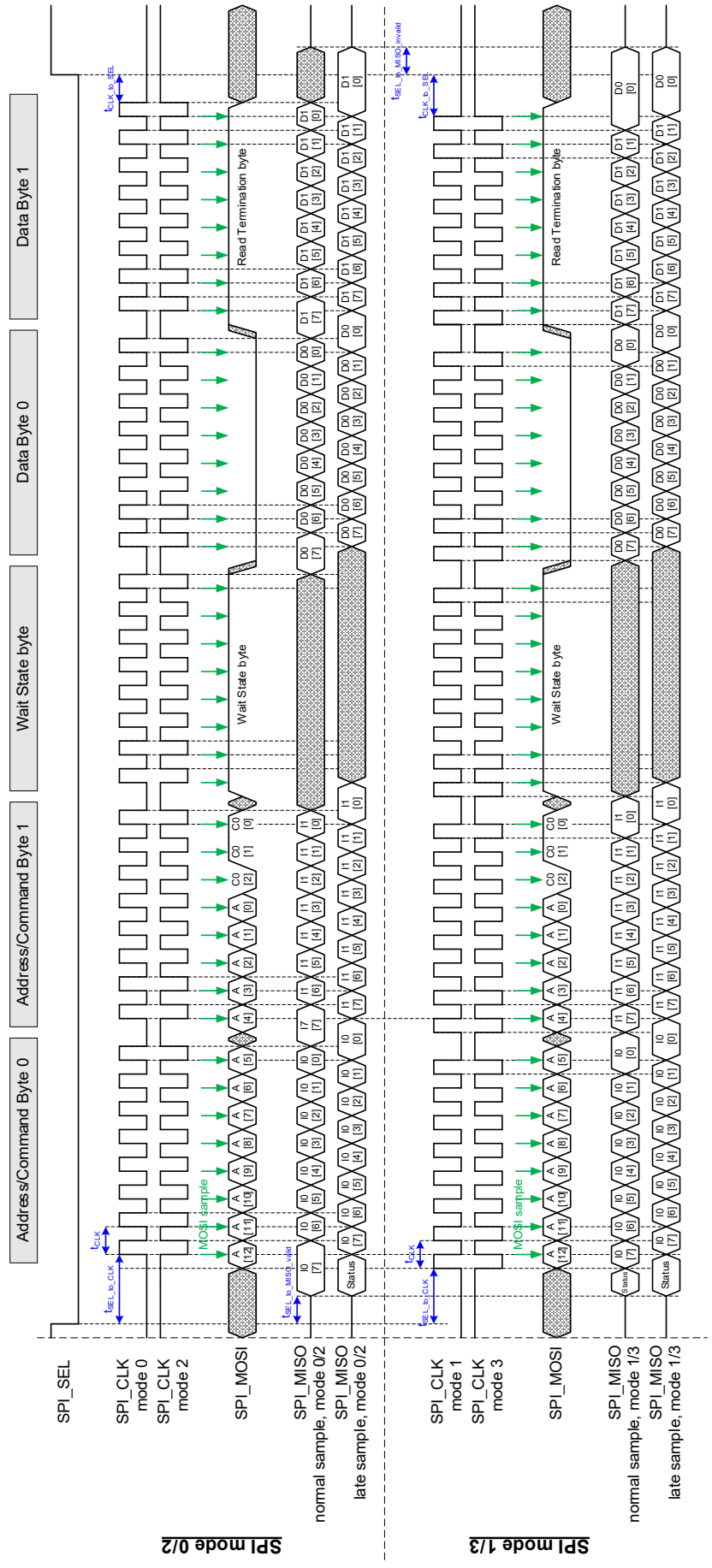

Figure 35: SPI_SEL random read access (2 byte addressing, 2 byte read data) with wait state byte

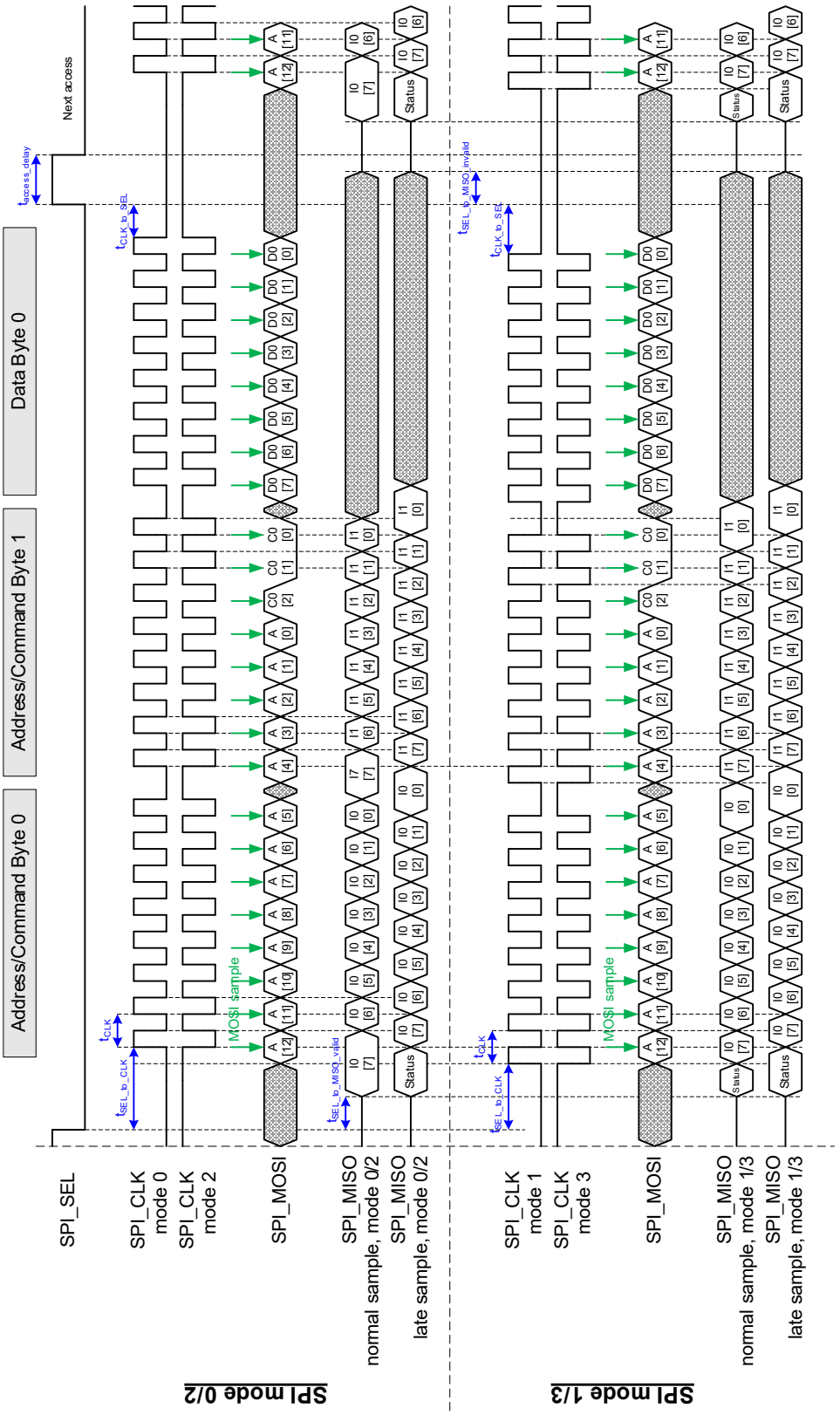

Figure 36: SPI_SEL random write access (2 byte addressing, 1 byte write data)

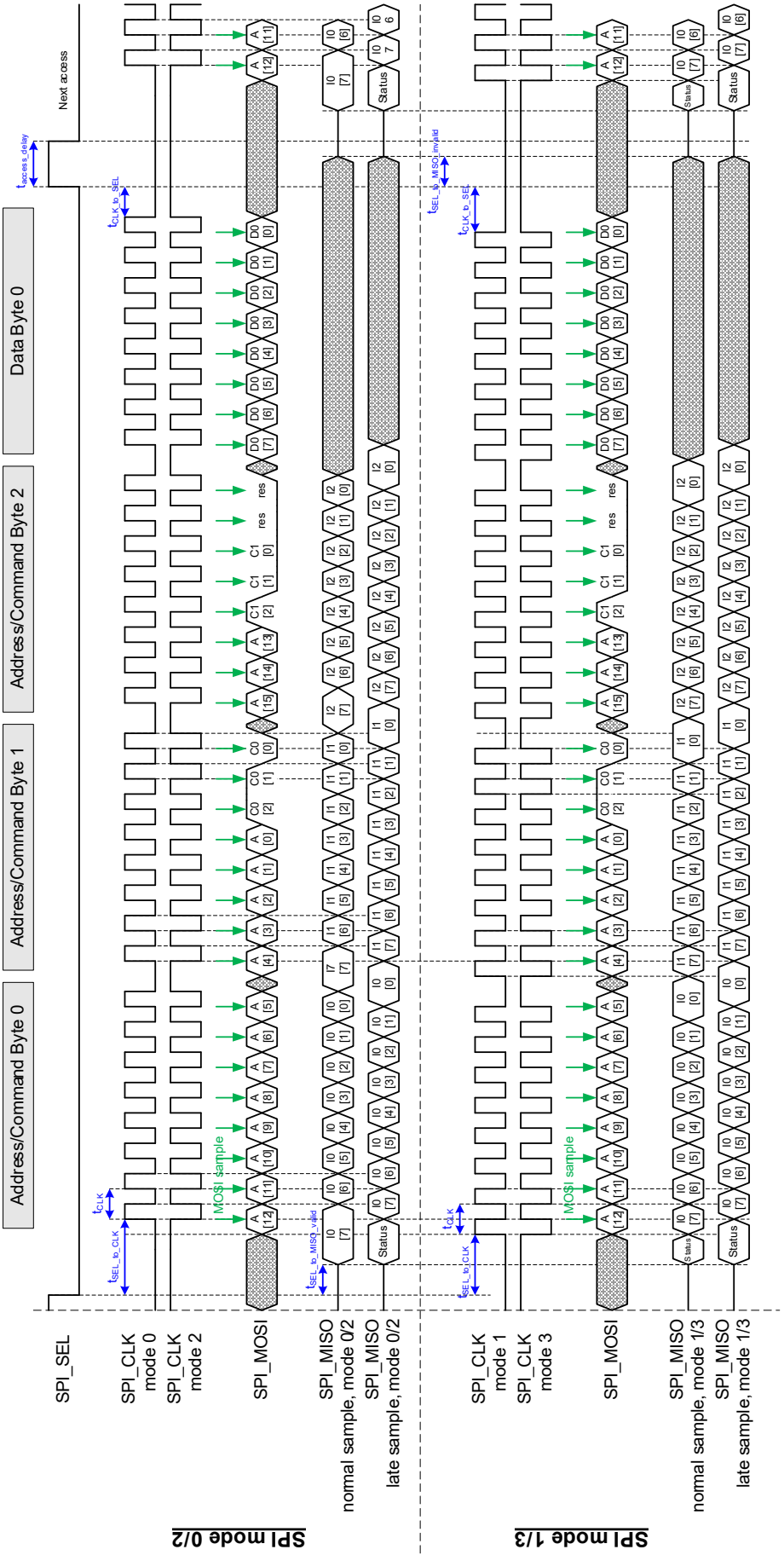

Figure 37: SPI_SEL random write access (3 byte addressing, 1 byte write data)

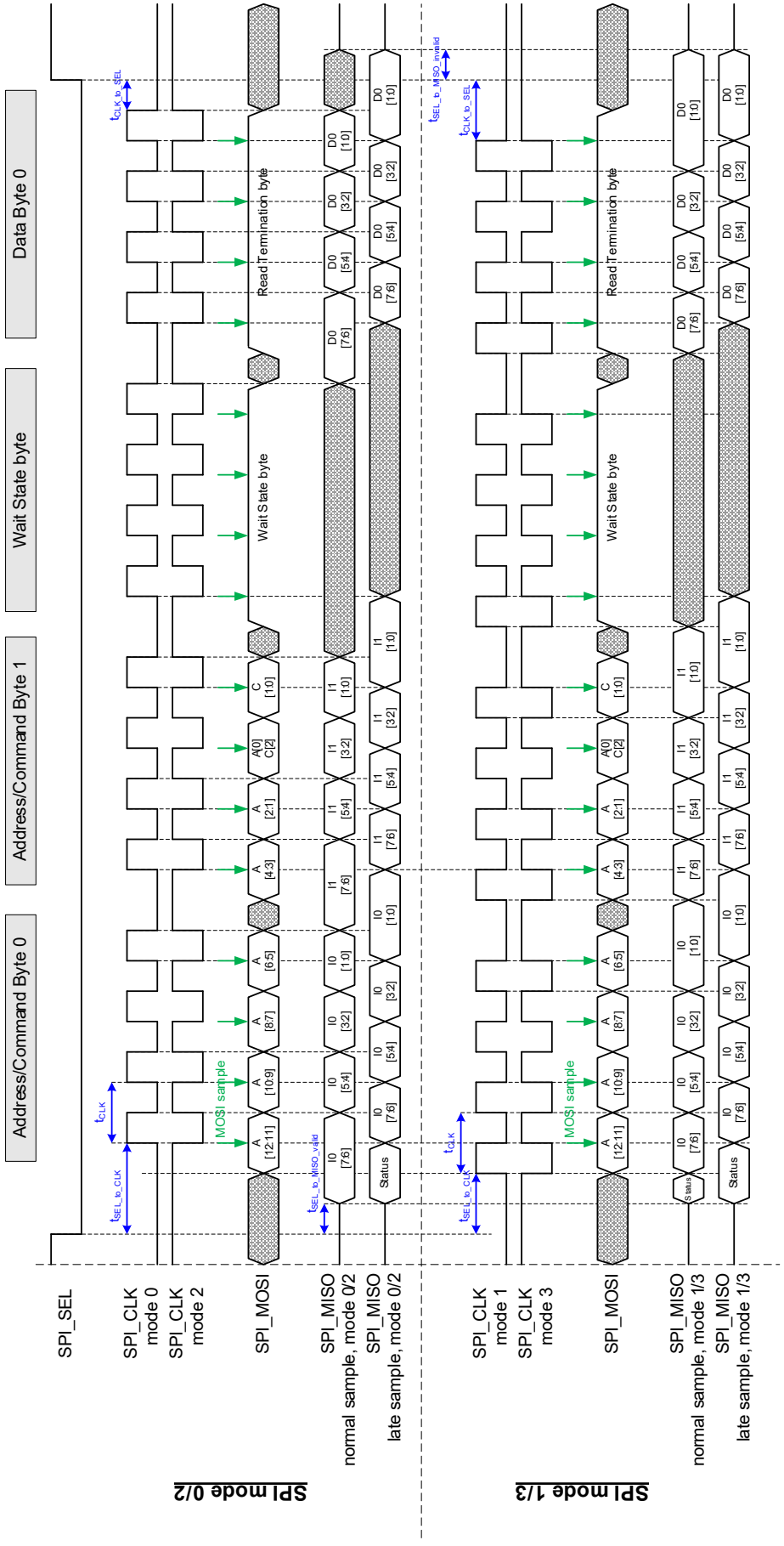

Figure 38: SPI_SEL random read access (2 bit data width, with wait state byte)

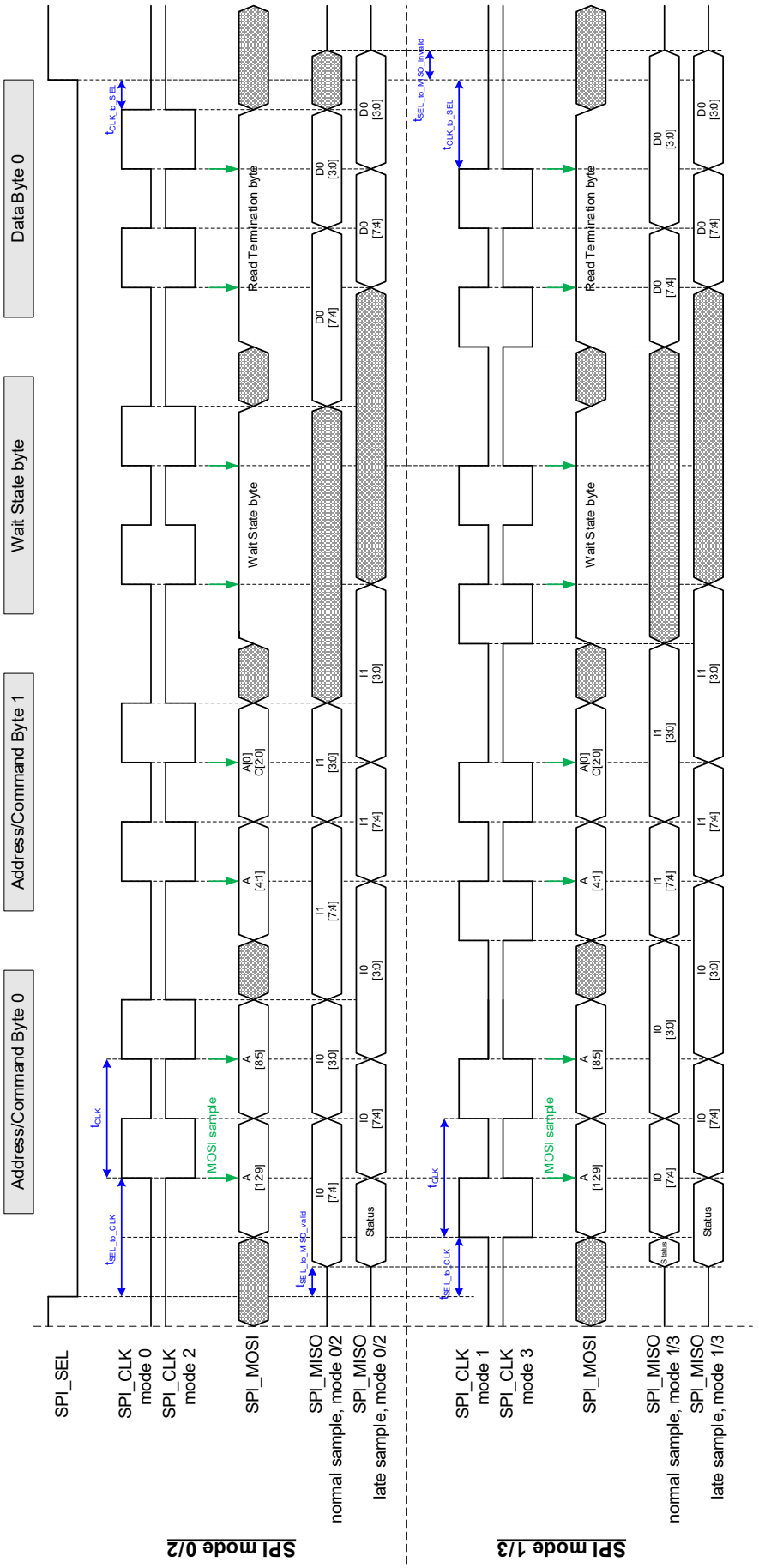

Figure 39: SPI_SEL random read access (4 bit data width, with wait state byte)

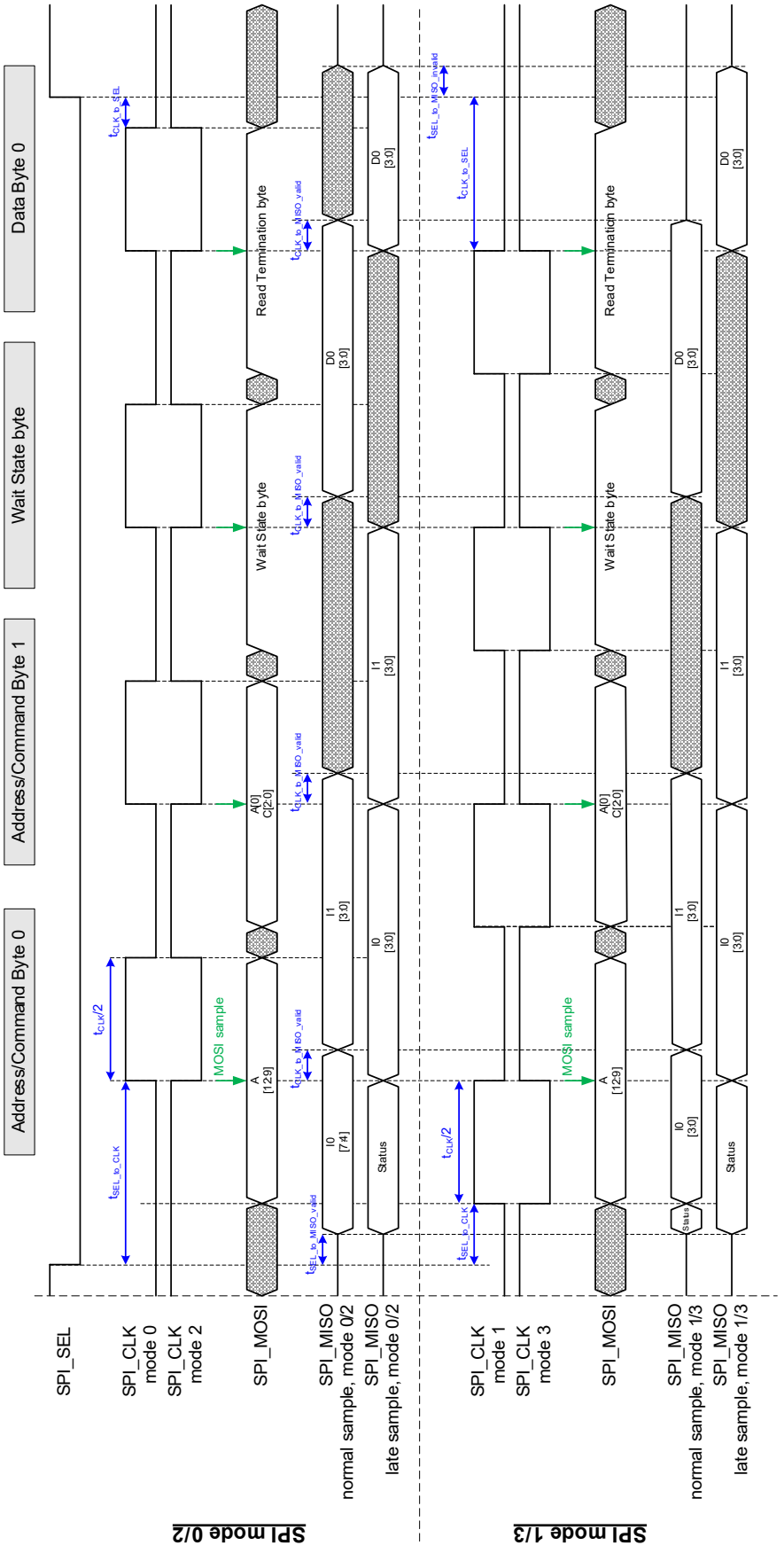

Figure 40: SPI_SEL random read access (8 bit data width, with wait state byte)

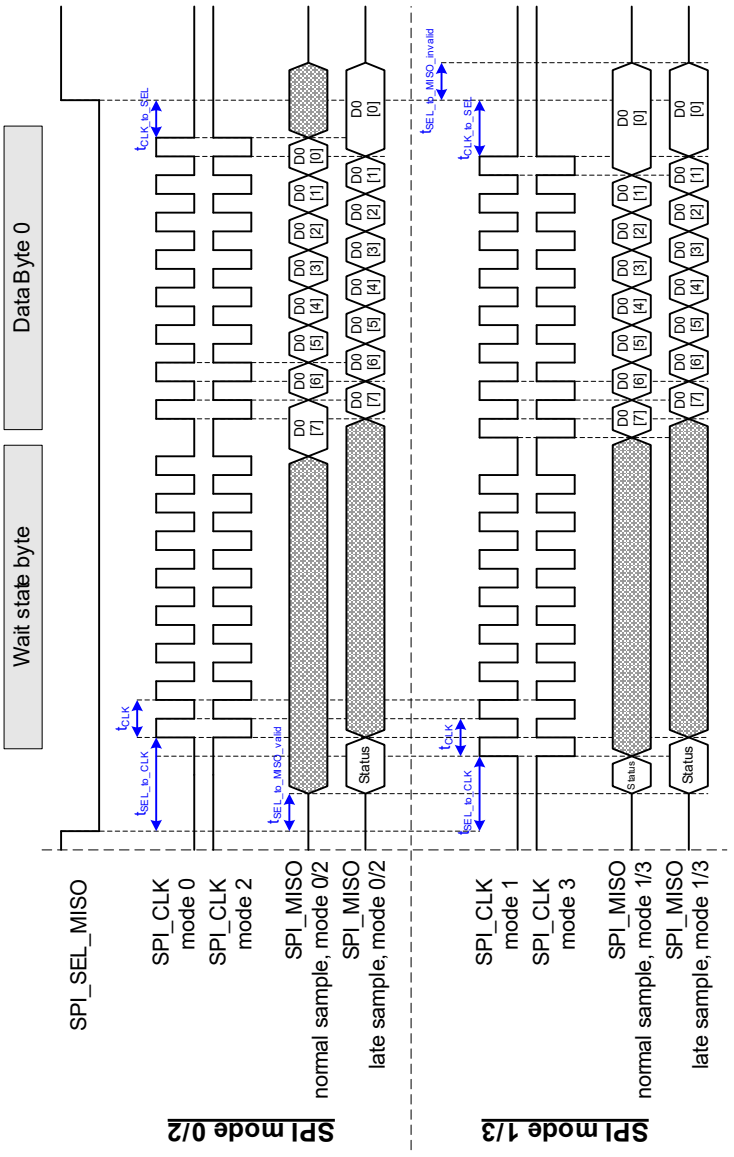

Figure 41: SPI_SEL_MISO access (wait state byte)

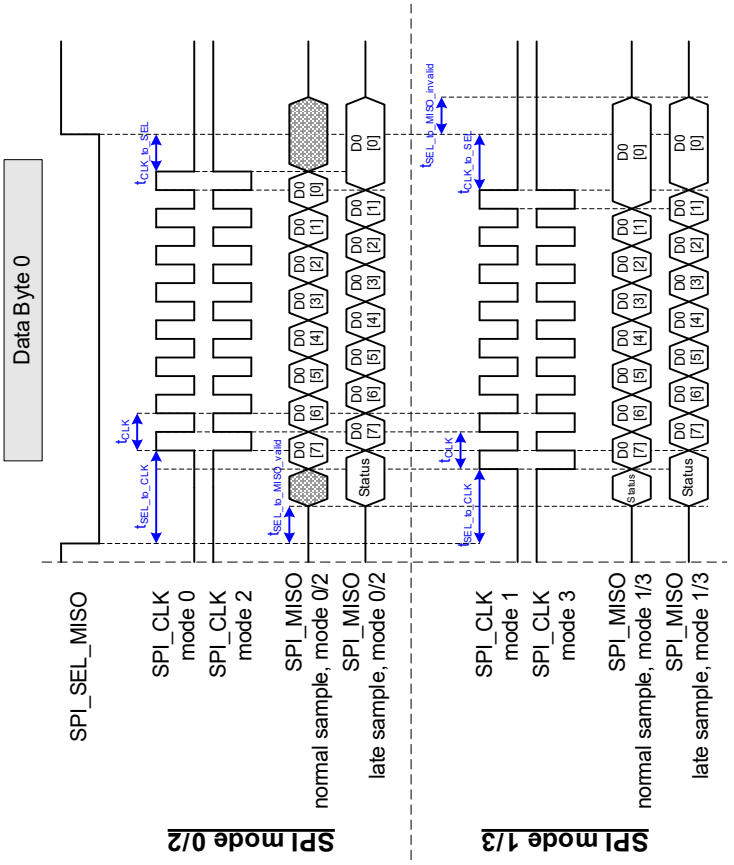

Figure 42: SPI_SEL_MISO access (no wait state byte)

6.4 SPI master interface

The SPI master PDI is selected with PDI type 0x15. The signals of the SPI master interface are:

6.4.1 Interface

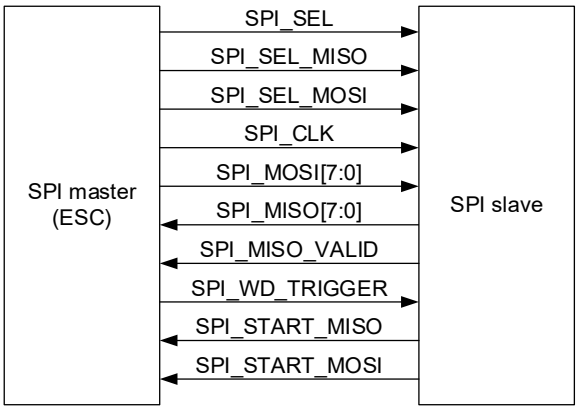

Figure 43: SPI master and slave interconnection (unidirectional)

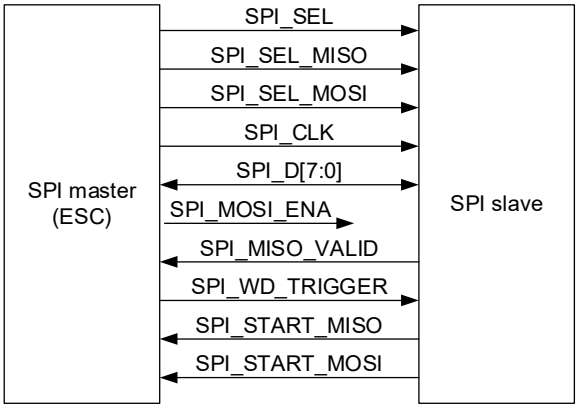

Figure 44: SPI master and slave interconnection (bidirectional)

Table 99: SPI master signals

Signal	Direction		Description	Signal polarity
SPI_CLK	OUT	(master → slave)	SPI clock	
SPI_SEL	OUT	(master → slave)	SPI random access chip select	Typical: act. low
SPI_SEL_MISO	OUT	(master → slave)	MISO chip select	Same as SPI_SEL
SPI_SEL_MOSI	OUT	(master → slave)	MOSI chip select	
SPI_MOSI[7:0]	OUT	(master → slave)	MOSI data (unidir)	act. high
SPI_MISO[7:0]	IN	(slave → master)	MISO data (unidir)	act. high
SPI_MISO_VALID	IN	(slave → master)	MISO data valid	act. high
SPI_MOSI_ENA	OUT		MOSI driver enabled	act. high
SPI_D[7:0]	BIDIR		MISO/MOSI data (bidirectional)	act. high
SPI_START_MISO	IN	(slave → master)	Request MISO access	act. high
SPI_START_MOSI	IN	(slave → master)	Request MOSI access	act. high
SPI_WD_TRIGGER	OUT		Watchdog trigger	act. high

6.4.2 Configuration

The SPI master interface is selected with PDI type 0x15 in the PDI0 control register 0x0140, or PDI1 control register 0x0180. It supports different data widths, operation modes, and signal polarities. The SPI master configuration is located in register 0x0150:0x0153 / 0x0190:0x0193.

The relevant edges of SPI_CLK for master and slave can be configured by selecting SPI mode and MISO sample delay in SPI master configuration register 0x0150.

The SPI_CLK frequency is configured using SPI master extended configuration register 0x0152:0x0153[15:12] / 0x0192:0x0193[15:12].

The SPI master built in the ET1150 supports different numbers of MISO/MOSI signals: 1, 2, 4, or 8 , connected to 1, 2, 4, or 8 SPI slaves. The number of bytes per slave can be configured, to simplify data mapping with multiple slaves. The number of signals, slaves, and bytes per slave is selected by SPI master configuration, it cannot be selected during operation, and it does not change within an access.

6.4.3 SPI master operation

The SPI master PDI is intended to connect with SPI slaves (either serial I/O devices, or the SPI slave PDI of other EtherCAT slave controllers). It does not support specific protocols directly, instead, the raw data bytes are transferred from/into SyncManager buffers. The SPI master PDI is only capable of transferring complete bytes.

The corresponding SyncManagers for MISO and MOSI data depend on the PDI (instance PDI0 or instance PDI1):

Table 100: SyncManager used by SPI master for SPI_SEL_MISO/SPI_SEL_MOSI

Direction	PDI0	PDI1	SyncManger direction
MOSI data	15	13	ECAT write, PDI read
MISO data	14	12	ECAT read, PDI write

The SPI 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 cycles SPI_CLK eight times for each byte transfer in x1 data width, or only once for x8 data width. In each clock cycle, both master and slave transmit 1/2/4/8 bit to the other side (full duplex).

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

6.4.4 MISO/MOSI access requirements

The SPI master automatically performs MISO and/or MOSI accesses when the requirements are met. If only MOSI requirements are met, a MOSI access is started. If only MISO requirements are met, a MISO access is started. If both requirements are met, and the start events occur at the same time, a combined MISO/MOSI access is performed.

MOSI access requirements:

- Process data watchdog active (triggered, register watchdog status process data 0x0440[0]=1)
- MOSI SyncManager configured properly and activated
- MOSI SyncManager has data (SyncManager buffer was written)
- MOSI access start event occurred

MISO access requirements:

- MISO SyncManager configured properly and activated
- SPI_MISO_VALID signal active
- MISO access start event occurred

NOTE: There is no dependency on the AL status register 0x0130, e.g., Operational mode.

6.4.5 Access start events

Possible combinations of MISO/MOSI start events are configured in SPI master extended configuration register 0x0152:0x0153[11:8] / register 0x0192:0x0193[11:8]. External SPI_START_MISO/SPI_START_MOSI signals are also supported, they can be activated simultaneously, as long as the active edge is within a short time window, which allows the ESC to detect the simultaneous access start.

When SPI_START_MISO/SPI_START_MOSI events are selected, the SPI_SEL_MISO/ SPI_SEL_MOSI signals are no longer available. The SPI_START_MISO/SPI_START_MOSI are triggering at the rising edge, so they can remain active during the access (potentially replacing SPI_SEL_MISO/SPI_SEL_MOSI in some cases).

Start configuration “continuous access” means that the accesses are performed back-to-back. The configuration “periodically every 10th cycle” is similar to continuous access, but only every 10th cycle will be performed.

6.4.6 Select signals and MISO/MOSI access order

The SPI master PDI supports three different select signals:

- SPI_SEL is a general select signal, active during the whole transfer of MISO and MOSI data
- SPI_SEL_MISO is a special select signal, only active while MISO bytes are transferred
- SPI_SEL_MOSI is a special select signal, only active while MOSI bytes are transferred

Depending on the application, one, two, or all three select signals are used. SPI_SEL is always active in combination with SPI_SEL_MISO/SPI_SEL_MOSI. It cannot be used independently.

NOTE: SPI_SEL can be used as a combined latch signal, using the leading edge to latch input signals, and using the trailing edge to apply output values.

When a combined MISO/MOSI access is performed, especially when the length is different for each direction, the byte order is relevant. It can be configured in two ways, described in the next chapters.

6.4.6.1 SPI master configuration register 0x0150[2]=0/0x0190[2]=0

In normal access order (for serial I/O devices), the MISO access is always performed from the beginning. SPI_SEL_MISO will be released earlier when MISO is shorter than MOSI.
The MOSI access is always aligned to the end of the access. SPI_SEL_MOSI will be asserted later when MOSI is shorter than MISO.

This example diming diagram shows the byte ordering, together with the select signals. SPI_CLK is using x4 operation (2 clocks per byte), clock edge reference is not accurate.

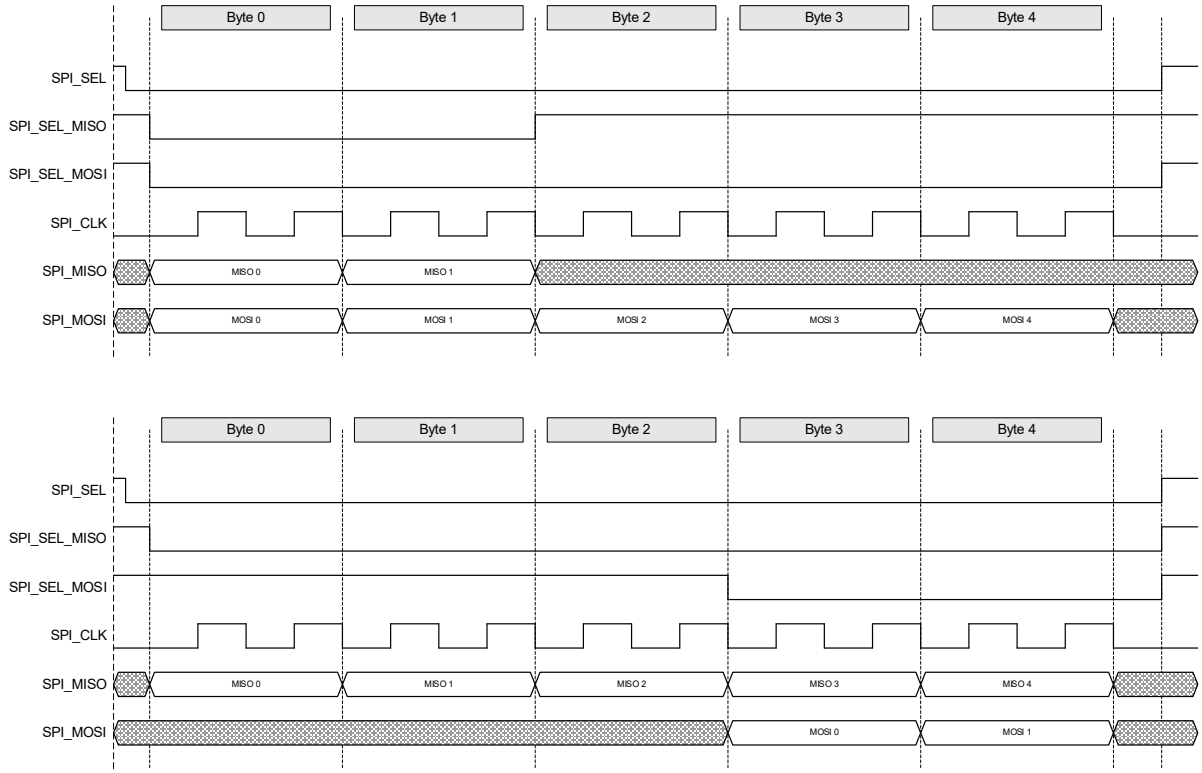

Figure 45: SPI master combined MISO/MOSI access (0x0150[2]=0, top: MOSI is longer, bottom: MISO is longer)

6.4.6.2 SPI master configuration register 0x0150[2]=1/0x0190[2]=1

In Beckhoff ESC SPI slave compatible mode, both MISO and MOSI accesses are performed from the beginning. The select signals are active until both MISO and MOSI access are finished. MOSI data is zero if no more MOSI bytes are to be transferred. MISO data is ignored after MISO length.

This example diming diagram shows the byte ordering, together with the select signals. SPI_CLK is using x4 operation (2 clocks per byte), clock edge reference is not accurate.

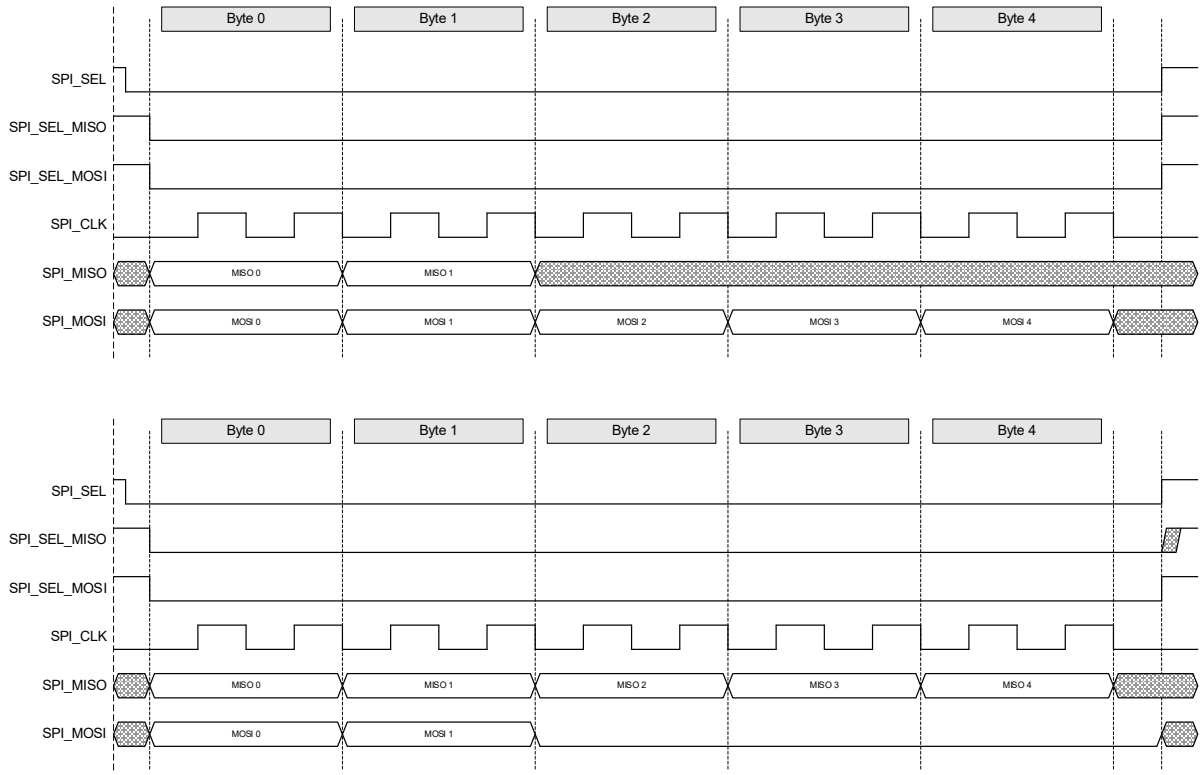

Figure 46: SPI master combined MISO/MOSI access (0x0150[2]=1, top: MOSI is longer, bottom: MISO is longer)

Bidirectional operation cannot be used for connection with Beckhoff ESCs.

6.4.7 Bidirectional operation

The SPI master also supports bidirectional operation, activated in SPI master extended configuration register 0x0152:0x0153[6] / 0x0192:0x0193[6].

In bidirectional mode, the same data signals are used for both directions, so data exchange cannot be performed in both directions simultaneously. Instead, the MISO access is performed first, followed by a turnaround period of approx. one byte length. Then the MOSI access is performed. SPI_SEL is asserted during all three phases.

The configuration option in SPI master configuration register 0x0150[2] has no influence on bidirectional mode.

The SPI_MOSI_ENA signal might be used for drivers in the data path between master and slave, it is active while the SPI master drives SPI_D.

This example diming diagram shows the byte ordering, together with the select signals. SPI_CLK is using x4 operation (2 clocks per byte), clock edge reference is not accurate.

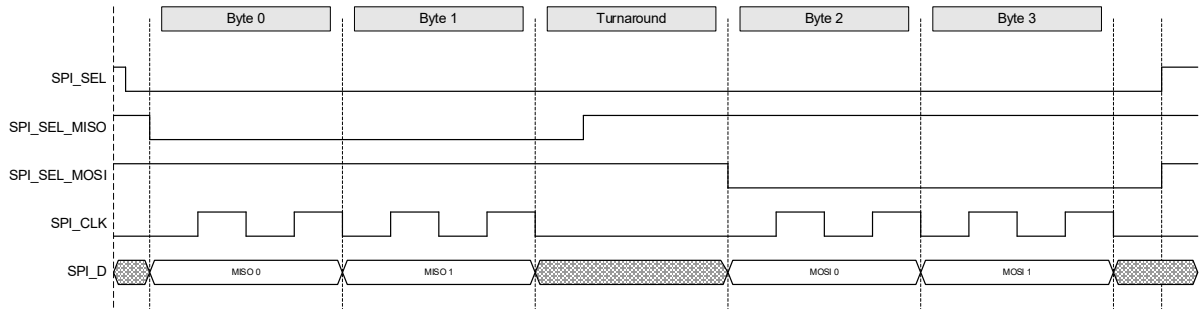

Figure 47: SPI master bidirectional mode with combined MISO/MOSI access

6.4.8 Deactivation

When either the process data watchdog expires, or the MOSI SyncManager is disabled, one additional MOSI access will be performed to reset the external serial I/O device. It is configurable if the additional cycle is performed immediately, or at the next MOSI start event (SPI master configuration register 0x0150[3]).

When a SyncManager is deactivated by the master, the buffer handling is immediately disabled. If this would occur during a MOSI access of the SPI master PDI, inconsistent data or outdated data could be sent. To prevent this, the feature “SyncManager deactivation delay” should be used. It is activated in ESC configuration register A5[3], changing the reset value of SyncManager PDI control 0x0807[6]. SyncManager deactivation delay keeps the buffer handling of the PDI read side active even when the SyncManager is deactivated, allowing the SPI master PDI to read the last valid buffer content from a SyncManager.

6.4.9 Data mapping

The SPI master supports up to 8 data lines. They can be connected to the same serial I/O device or split between multiple devices (e.g. four devices with two data lines for each of them). By default, the data of all I/O devices would be mixed across several bytes in the SyncManager buffer, requiring bit-operations for separations. But the SPI master supports limited ordering features for this case, using SPI master extended configuration register 0x0152:0x0153[5:2] / 0x0192:0x0193[5:2] – bit operations are not required in most cases.

The number of MISO/MOSI signals depends on the number of channels and the number of SPI slaves. Supported configurations are listed below:

Table 101: SPI master number of MISO/MOSI signals

Number of MISO/MOSI signals	Number of channels 0x0152[1:0]/ 0x0192[1:0]	Number of slaves 0x0152[3:2]/ 0x0152[3:2]
x1	0	0
x2	0	1
	1	0
x4	0	2
	1	1
	2	0
x8	0	3
	1	2
	2	1
	3	0

The following diagrams show supported connections, the access order on the SPI interface, and the memory view (assuming the SyncManager is configured to start at address 0x1000).

The configuration of 3 bytes/slave is only available for 2 slaves, and the only supported length of the SyncManager buffers for MISO/MOSI is 6 bytes (2 slaves * 3 bytes/slave).

6.4.9.1 Data mapping for 1 slave

Data mapping for 1 slave does not depend on the data width (x1, x2, x4, x8). Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

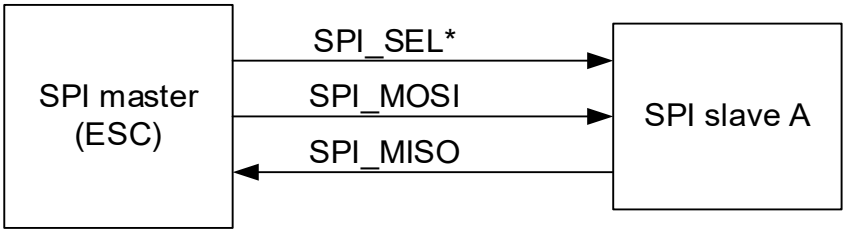

Figure 48: SPI master connection with 1 slave

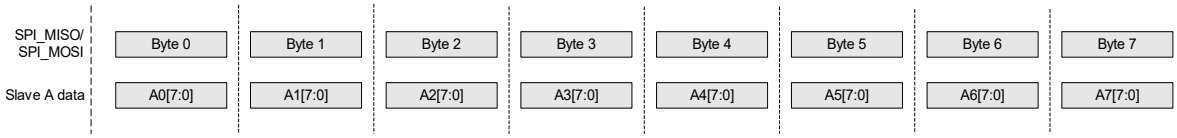

Figure 49: SPI master access to 1 slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	A2[7:0]
0x1003	A3[7:0]
0x1004	A4[7:0]
0x1005	A5[7:0]
0x1006	A6[7:0]
0x1007	A7[7:0]

Figure 50: SPI master data mapping for 1 slave

6.4.9.2 Data mapping for 2 slaves with 1 data channel per slave

Data mapping for 2 slaves depends on the data width. Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

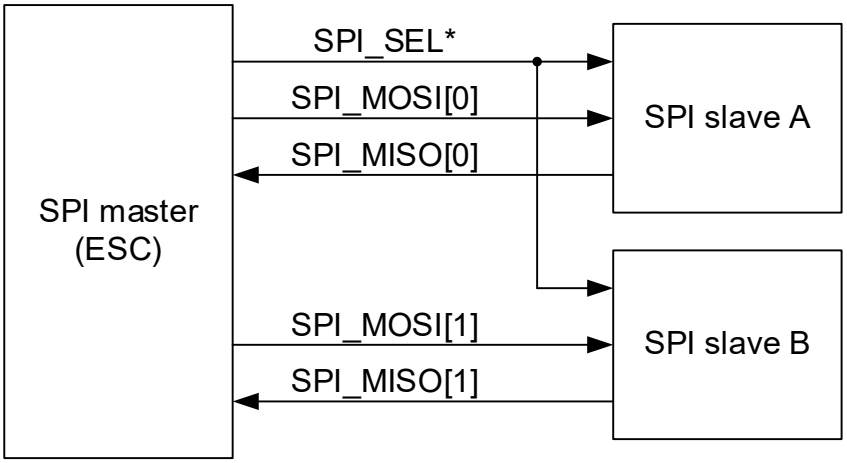

Figure 51: SPI master connection with 2 slaves, 1 channel per slave

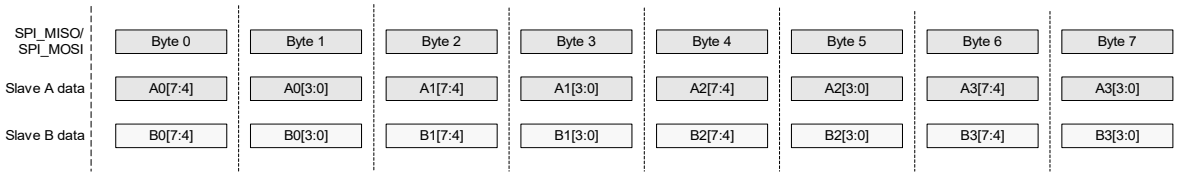

Figure 52: SPI master access to 2 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	B0[7]	A0[7]	B0[6]	A0[6]	B0[5]	A0[5]	B0[4]	A0[4]
0x1001	B0[3]	A0[3]	B0[2]	A0[2]	B0[1]	A0[1]	B0[0]	A0[0]
0x1002	B1[7]	A1[7]	B1[6]	A1[6]	B1[5]	A1[5]	B1[4]	A1[4]
0x1003	B1[3]	A1[3]	B1[2]	A1[2]	B1[1]	A1[1]	B1[0]	A1[0]
0x1004	B2[7]	A2[7]	B2[6]	A2[6]	B2[5]	A2[5]	B2[4]	A2[4]
0x1005	B2[3]	A2[3]	B2[2]	A2[2]	B2[1]	A2[1]	B2[0]	A2[0]
0x1006	B3[7]	A3[7]	B3[6]	A3[6]	B3[5]	A3[5]	B3[4]	A3[4]
0x1007	B3[3]	A3[3]	B3[2]	A3[2]	B3[1]	A3[1]	B3[0]	A3[0]

Figure 53: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

1 byte per slave

0x1000	A0[7:0]
0x1001	B0[7:0]
0x1002	A1[7:0]
0x1003	B1[7:0]
0x1004	A2[7:0]
0x1005	B2[7:0]
0x1006	A3[7:0]
0x1007	B3[7:0]

2 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	B0[7:0]
0x1003	B1[7:0]
0x1004	A2[7:0]
0x1005	A3[7:0]
0x1006	B2[7:0]
0x1007	B3[7:0]

3 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	A2[7:0]
0x1003	B0[7:0]
0x1004	B1[7:0]
0x1005	B2[7:0]
0x1006	-
0x1007	-

4 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	A2[7:0]
0x1003	A3[7:0]
0x1004	B0[7:0]
0x1005	B1[7:0]
0x1006	B2[7:0]
0x1007	B3[7:0]

Figure 54: SPI master data with mapping: configured for 2 slaves, 1 channel per slave

6.4.9.3 Data mapping for 2 slaves with 2 data channels per slave

Data mapping for 2 slaves depends on the data width, unless correct 'bytes per slave' is configured. Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

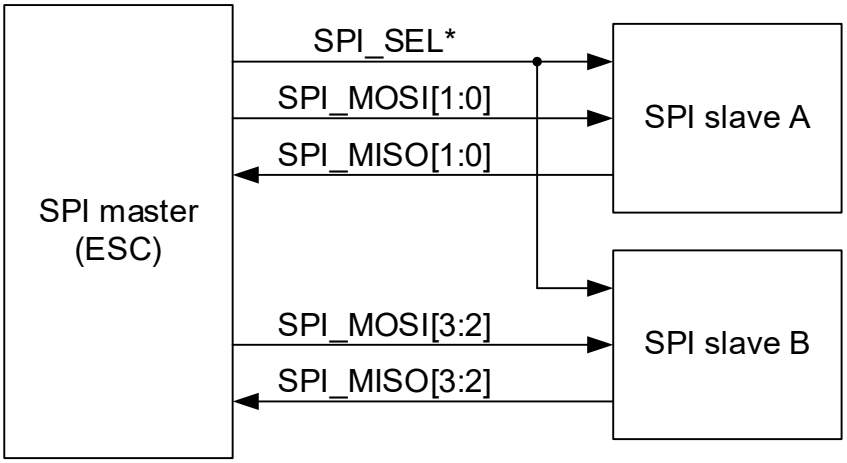

Figure 55: SPI master connection with 2 slaves, 2 channels per slave

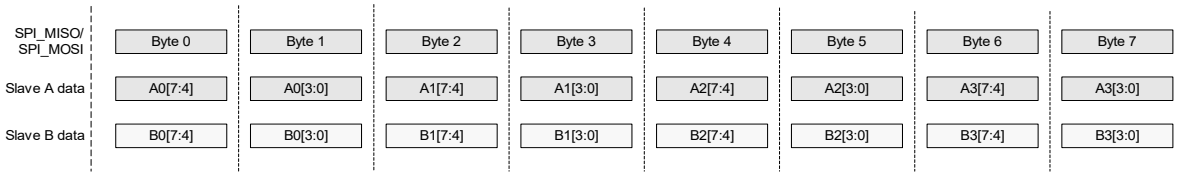

Figure 56: SPI master access to 2 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	B0[7:6]		A0[7:6]		B0[5:4]		A0[5:4]	
0x1001	B0[3:2]		A0[3:2]		B0[1:0]		A0[1:0]	
0x1002	B1[7:6]		A1[7:6]		B1[5:4]		A1[5:4]	
0x1003	B1[3:2]		A1[3:2]		B1[1:0]		A1[1:0]	
0x1004	B2[7:6]		A2[7:6]		B2[5:4]		A2[5:4]	
0x1005	B2[3:2]		A2[3:2]		B2[1:0]		A2[1:0]	
0x1006	B3[7:6]		A3[7:6]		B3[5:4]		A3[5:4]	
0x1007	B3[3:2]		A3[3:2]		B3[1:0]		A3[1:0]	

Figure 57: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

1 byte per slave

0x1000	A0[7:0]
0x1001	B0[7:0]
0x1002	A1[7:0]
0x1003	B1[7:0]
0x1004	A2[7:0]
0x1005	B2[7:0]
0x1006	A3[7:0]
0x1007	B3[7:0]

2 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	B0[7:0]
0x1003	B1[7:0]
0x1004	A2[7:0]
0x1005	A3[7:0]
0x1006	B2[7:0]
0x1007	B3[7:0]

3 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	A2[7:0]
0x1003	B0[7:0]
0x1004	B1[7:0]
0x1005	B2[7:0]
0x1006	-
0x1007	-

4 bytes per slave

0x1000	A0[7:0]
0x1001	A1[7:0]
0x1002	A2[7:0]
0x1003	A3[7:0]
0x1004	B0[7:0]
0x1005	B1[7:0]
0x1006	B2[7:0]
0x1007	B3[7:0]

Figure 58: SPI master data with mapping: configured for 2 slaves, 2 channels per slave

6.4.9.4 Data mapping for 2 slaves with 4 data channels per slave

Data mapping for 2 slaves depends on the data width. Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

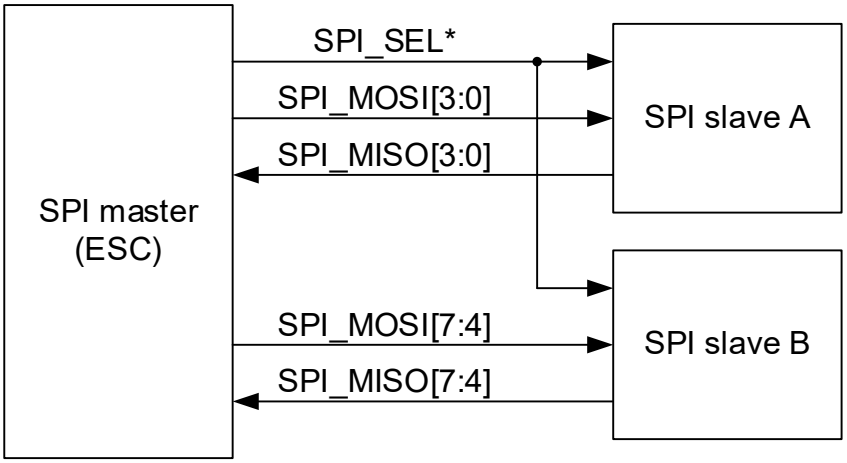

Figure 59: SPI master connection with 2 slaves, 4 channels per slave

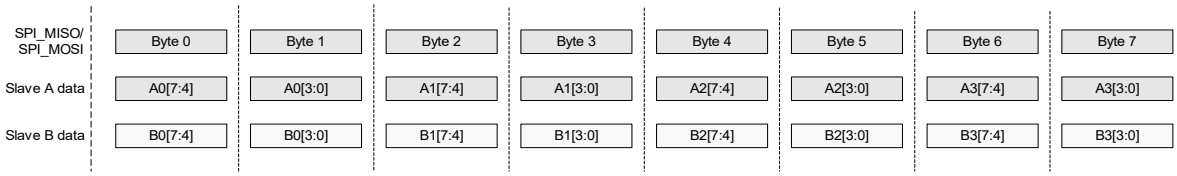

Figure 60: SPI master access to 2 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	B0[7:4]				A0[7:4]			
0x1001	B0[3:0]				A0[3:0]			
0x1002	B1[7:4]				A1[7:4]			
0x1003	B1[3:0]				A1[3:0]			
0x1004	B0[7:4]				A0[7:4]			
0x1005	B0[3:0]				A0[3:0]			
0x1006	B3[7:4]				A3[7:4]			
0x1007	B3[3:0]				A3[3:0]			

Figure 61: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

1 byte per slave		2 bytes per slave		3 bytes per slave		4 bytes per slave	
0x1000	A0[7:0]	0x1000	A0[7:0]	0x1000	A0[7:0]	0x1000	A0[7:0]
0x1001	B0[7:0]	0x1001	A1[7:0]	0x1001	A1[7:0]	0x1001	A1[7:0]
0x1002	A1[7:0]	0x1002	B0[7:0]	0x1002	A2[7:0]	0x1002	A2[7:0]
0x1003	B1[7:0]	0x1003	B1[7:0]	0x1003	B0[7:0]	0x1003	A3[7:0]
0x1004	A2[7:0]	0x1004	A2[7:0]	0x1004	B1[7:0]	0x1004	B0[7:0]
0x1005	B2[7:0]	0x1005	A3[7:0]	0x1005	B2[7:0]	0x1005	B1[7:0]
0x1006	A3[7:0]	0x1006	B2[7:0]	0x1006	-	0x1006	B2[7:0]
0x1007	B3[7:0]	0x1007	B3[7:0]	0x1007	-	0x1007	B3[7:0]

Figure 62: SPI master data with mapping: configured for 2 slaves, 4 channels per slave

6.4.9.5 Data mapping for 4 slaves with 1 data channel per slave

Data mapping for 4 slaves depends on the data width. Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

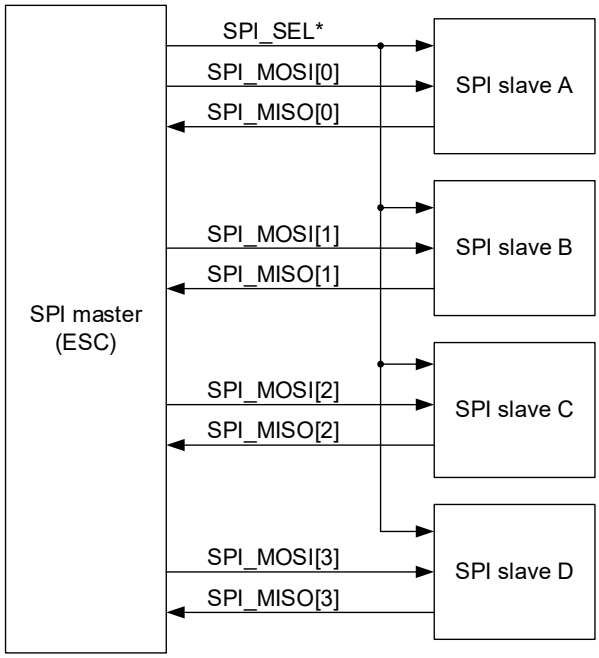

Figure 63: SPI master connection with 4 slaves, 1 channel per slave

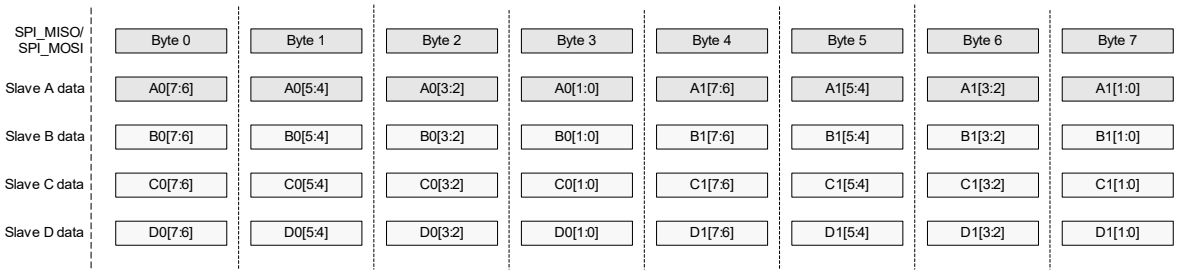

Figure 64: SPI master access to 4 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	D0[7]	C0[7]	B0[7]	A0[7]	D0[6]	C0[6]	B0[6]	A0[6]
0x1001	D0[5]	C0[5]	B0[5]	A0[5]	D0[4]	C0[4]	B0[4]	A0[4]
0x1002	D0[3]	C0[3]	B0[3]	A0[3]	D0[2]	C0[2]	B0[2]	A0[2]
0x1003	D0[1]	C0[1]	B0[1]	A0[1]	D0[0]	C0[0]	B0[0]	A0[0]
0x1004	D1[7]	C1[7]	B1[7]	A1[7]	D1[6]	C1[6]	B1[6]	A1[6]
0x1005	D1[5]	C1[5]	B1[5]	A1[5]	D1[4]	C1[4]	B1[4]	A1[4]
0x1006	D1[3]	C1[3]	B1[3]	A1[3]	D1[2]	C1[2]	B1[2]	A1[2]
0x1007	D1[1]	C1[1]	B1[1]	A1[1]	D1[0]	C1[0]	B1[0]	A1[0]

Figure 65: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

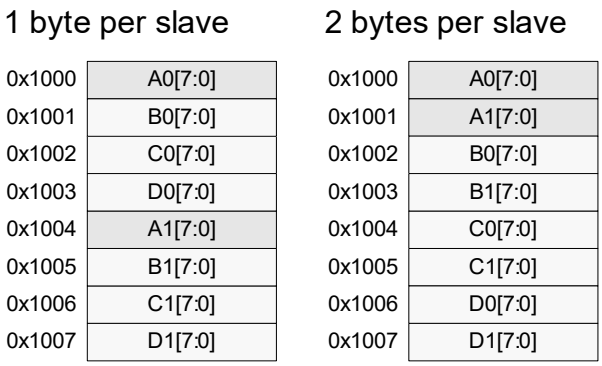

Figure 66: SPI master data with mapping: configured for 4 slaves, 1 channel per slave

6.4.9.6 Data mapping for 4 slaves with 2 data channels per slave

Data mapping for 4 slaves depends on the data width. Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

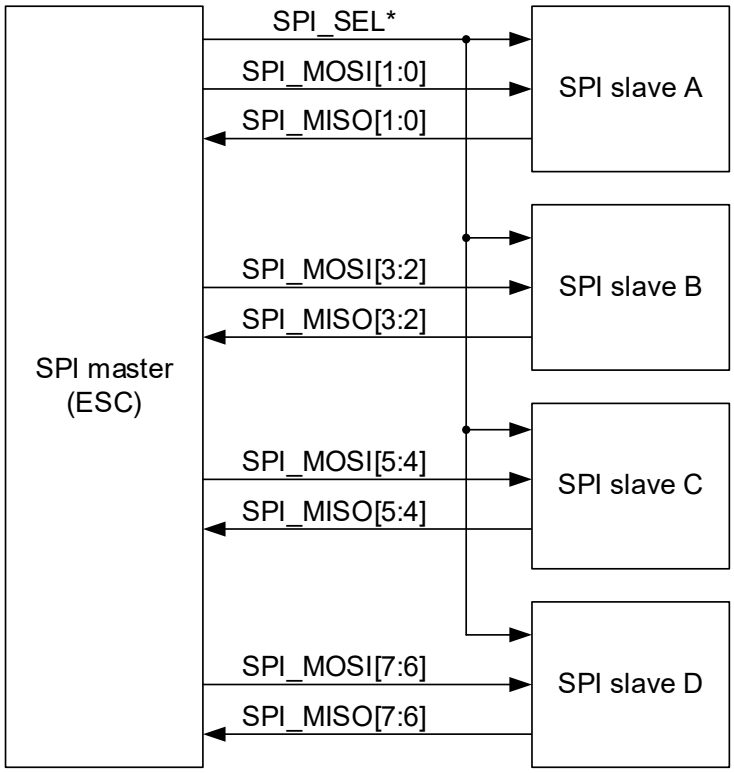

Figure 67: SPI master connection with 4 slaves, 2 channels per slave

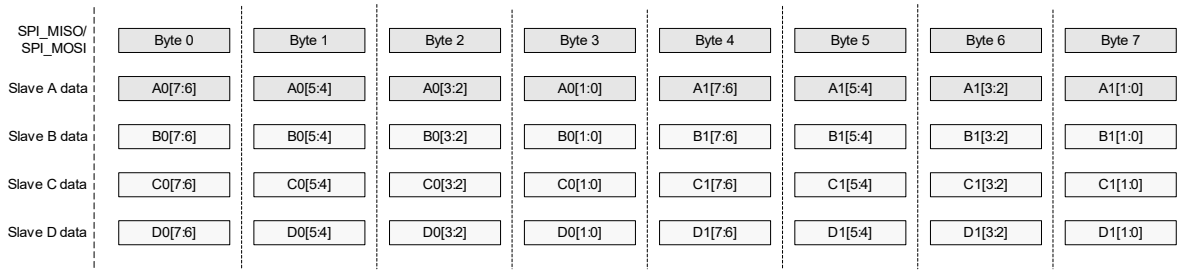

Figure 68: SPI master access to 4 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	D0[7:6]		C0[7:6]		B0[7:6]		A0[7:6]	
0x1001	D0[5:4]		C0[5:4]		B0[5:4]		A0[5:4]	
0x1002	D0[3:2]		C0[3:2]		B0[3:2]		A0[3:2]	
0x1003	D0[1:0]		C0[1:0]		B0[1:0]		A0[1:0]	
0x1004	D1[7:6]		C1[7:6]		B1[7:6]		A1[7:6]	
0x1005	D1[5:4]		C1[5:4]		B1[5:4]		A1[5:4]	
0x1006	D1[3:2]		C1[3:2]		B1[3:2]		A1[3:2]	
0x1007	D1[1:0]		C1[1:0]		B1[1:0]		A1[1:0]	

Figure 69: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

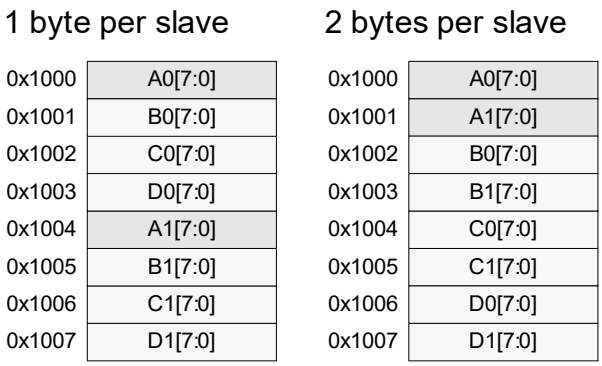

Figure 70: SPI master data with mapping: configured for 4 slaves, 2 channels per slave

6.4.9.7 Data mapping for 8 slaves with 1 data channels per slave

Mapping does not depend on the select signal, so SPI_SEL* is shown as a placeholder.

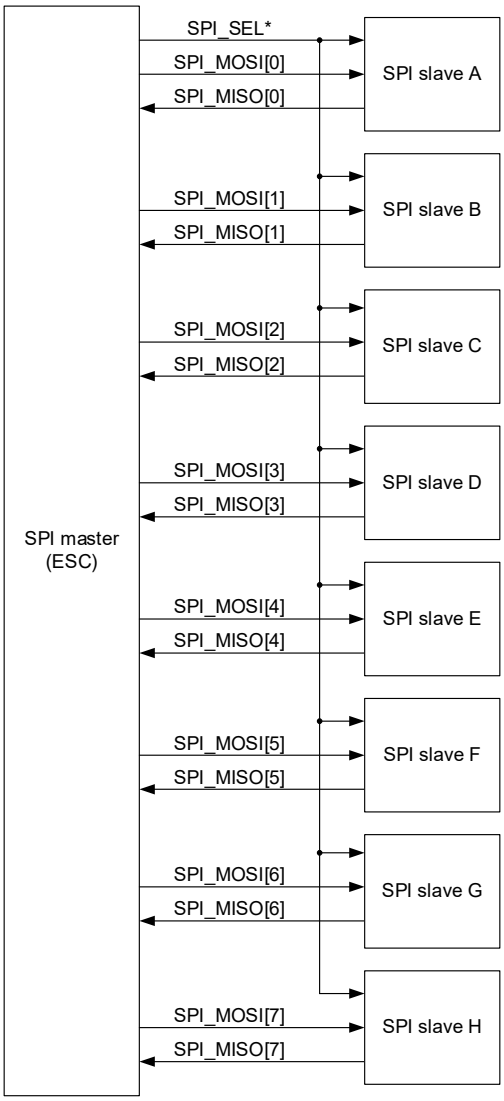

Figure 71: SPI master connection with 8 slaves, 1 channel per slave

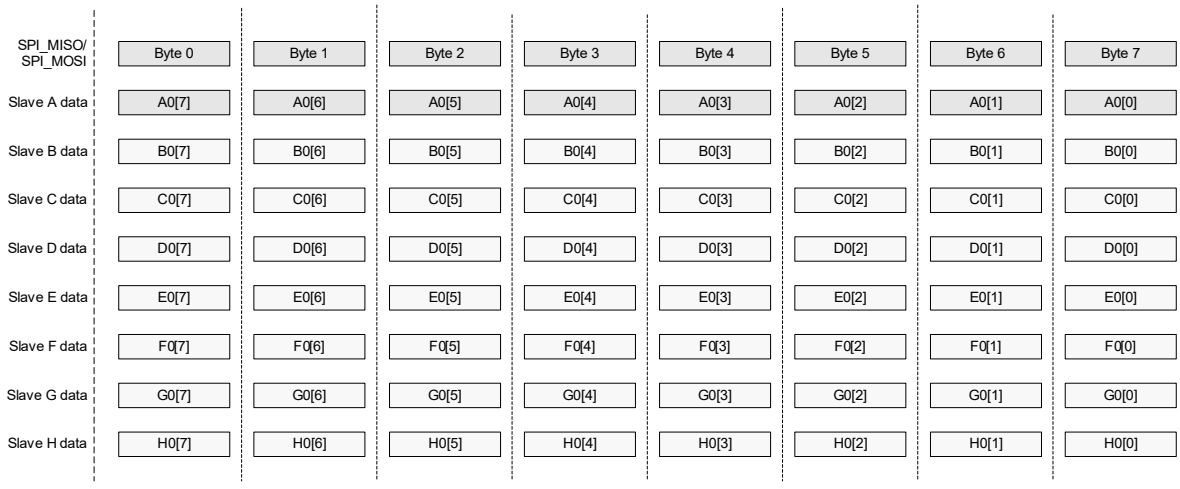

Figure 72: SPI master access to 8 slaves

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0x1000	H0[7]	G0[7]	F0[7]	E0[7]	D0[7]	C0[7]	B0[7]	A0[7]
0x1001	H0[6]	G0[6]	F0[6]	E0[6]	D0[6]	C0[6]	B0[6]	A0[6]
0x1002	H0[5]	G0[5]	F0[5]	E0[5]	D0[5]	C0[5]	B0[5]	A0[5]
0x1003	H0[4]	G0[4]	F0[4]	E0[4]	D0[4]	C0[4]	B0[4]	A0[4]
0x1004	H0[3]	G0[3]	F0[3]	E0[3]	D0[3]	C0[3]	B0[3]	A0[3]
0x1005	H0[2]	G0[2]	F0[2]	E0[2]	D0[2]	C0[2]	B0[2]	A0[2]
0x1006	H0[1]	G0[1]	F0[1]	E0[1]	D0[1]	C0[1]	B0[1]	A0[1]
0x1007	H0[0]	G0[0]	F0[0]	E0[0]	D0[0]	C0[0]	B0[0]	A0[0]

Figure 73: SPI master data without mapping: configured for 1 slave (for reference)

The correct configuration of the number of slaves leads to a better data mapping:

1 byte per slave

0x1000	A0[7:0]
0x1001	B0[7:0]
0x1002	C0[7:0]
0x1003	D0[7:0]
0x1004	E0[7:0]
0x1005	F0[7:0]
0x1006	G0[7:0]
0x1007	H0[7:0]

Figure 74: SPI master data with mapping: configured for 8 slaves, 1 channel per slave

6.4.10 SPI master errors

The following reasons for SPI master errors are detected by the SPI master:

- MISO/MOSI SyncManager configuration is not valid
- Invalid configuration of number of slaves/bytes per slave/number of MISO/MOSI signals
- MISO start event without MISO valid (optionally enabled in SPI master user mode register)
- Start event during MISO/MOSI transfer
- MOSI SyncManager disabled during MOSI transfer

A SPI master error will have these consequences:

- The PDI error counter 0x030D/0x0340 will be incremented. For the first error, a PDI error code will be stored in the PDI error code register 0x030E:0x030F/0x0341:0x0342, indicating details of the error reason.

6.4.11 Timing specifications

Table 102: SPI master timing characteristics ET1150

Parameter	Min	Max	Comment
		PRELIMINARY TIMING	
t_{CLK}	10 ns	400 ns	SPI_CLK frequency configurable (f_{CLK} ≤ 100 MHz)
$t_{SEL\_to\_CLK}$	t_{CLK}/2		First SPI_CLK cycle after SPI_SEL asserted
$t_{CLK\_to\_SEL}$	t_{CLK}/2		De-assertion of SPI_SEL after last SPI_CLK cycle
t_{MISO_setup}	8 ns		SPI_MISO valid before SPI_CLK edge
t_{MISO_hold}	4 ns		SPI_MISO valid after SPI_CLK edge
$t_{CLK\_to\_MOSI\_valid}$		5 ns	SPI_MOSI valid after SPI_CLK edge
$t_{CLK\_to\_MOSI\_invalid}$		5 ns	SPI_MOSI invalid before SPI_CLK edge

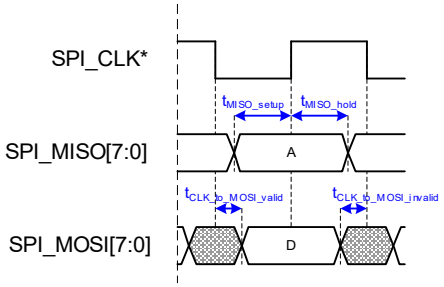

Figure 75: Basic SPI_MOSI/SPI_MISO timing (*refer to access diagrams for relevant edges of SPI_CLK)

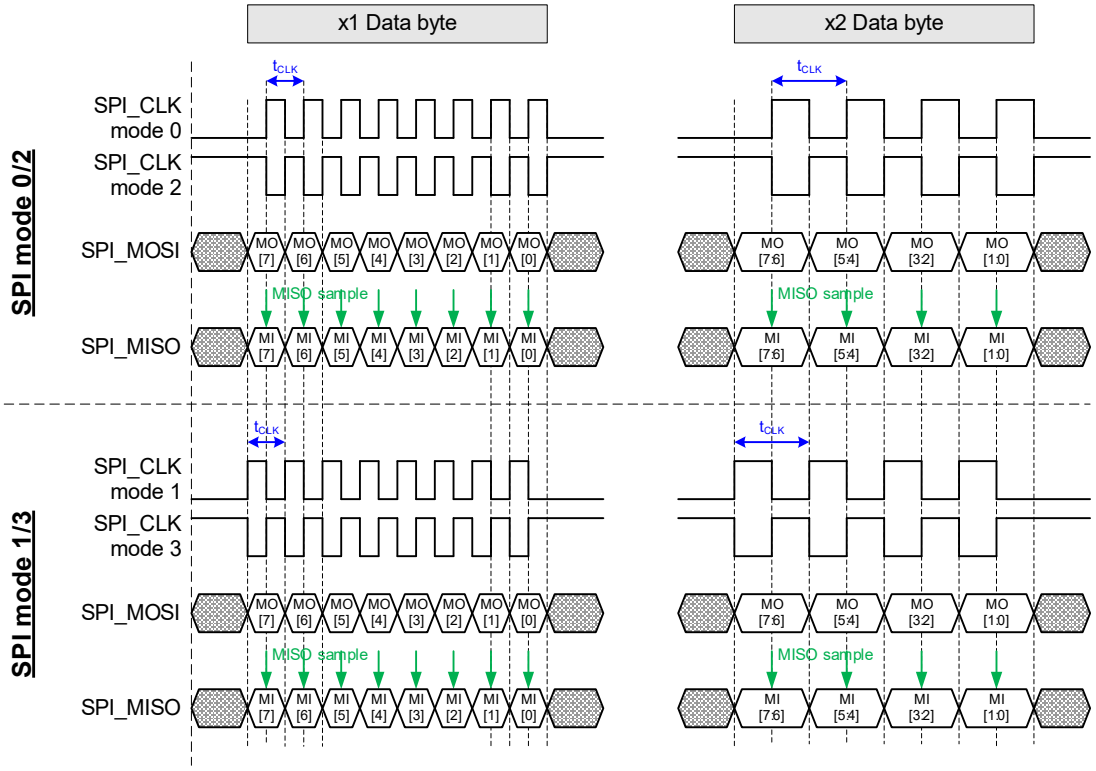

Figure 76: Basic SPI master timing x1/x2 data width (MISO sample delay=0)

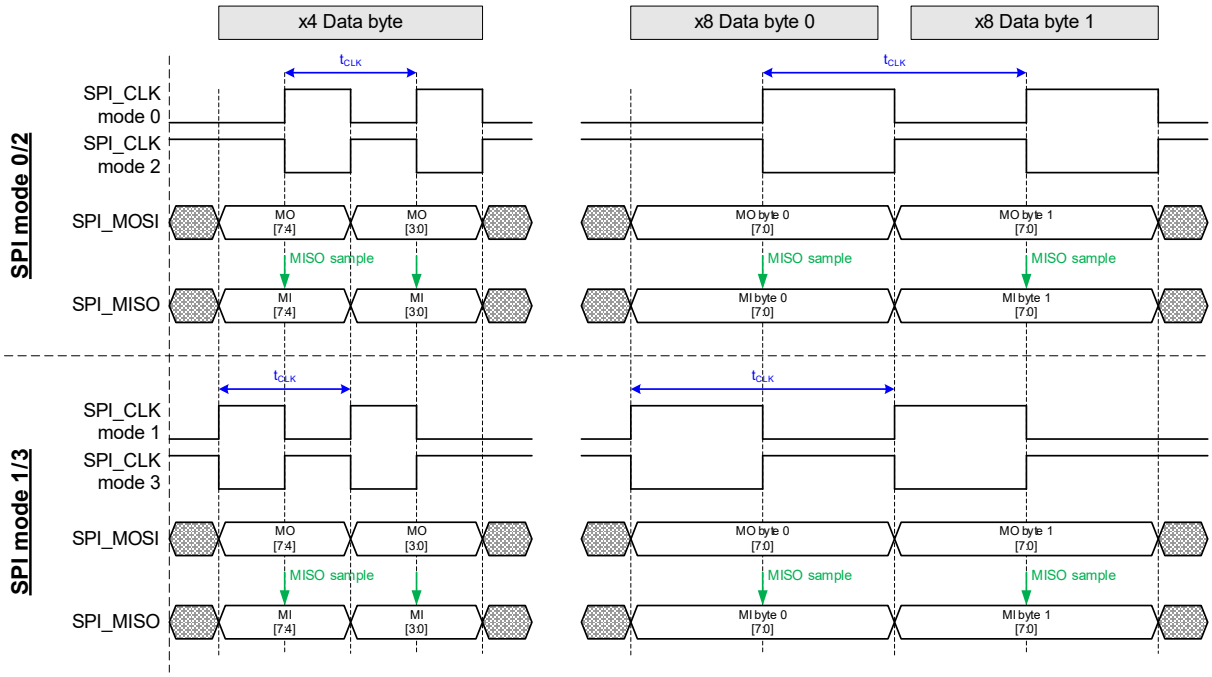

Figure 77: Basic SPI master timing x4/x8 data width (MISO sample delay=0)

6.5 Asynchronous 8/16 bit µController interface

6.5.1 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:

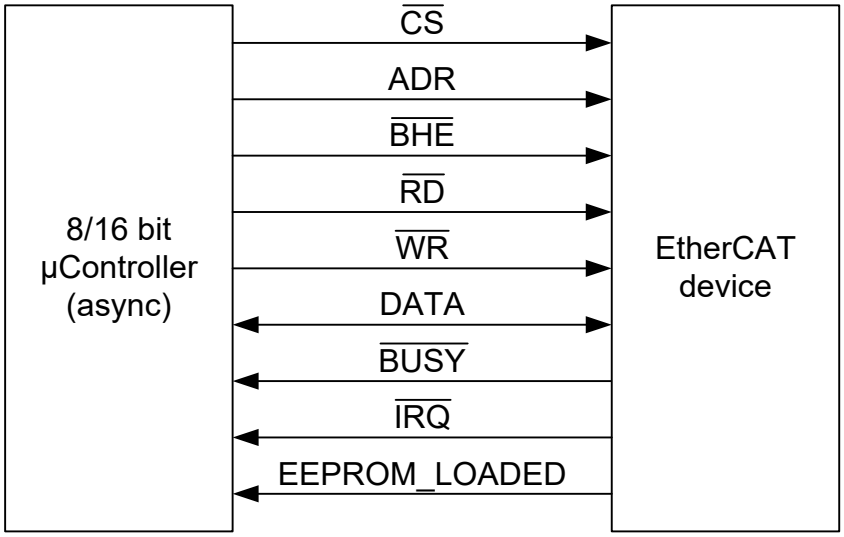

Figure 78: µController interconnection[2]

Table 103: µController signals

Signal	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.

6.5.2 Configuration

The 16 bit asynchronous µController interface is selected with PDI type 0x08 in the PDI0 control register 0x0140, or PDI1 control register 0x0180, the 8 bit asynchronous µController interface has PDI type 0x09. It supports different configurations, which are located in registers 0x0150:0x0153 / 0x0190:0x0193.

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

6.5.3 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 104: 8 bit µController interface access types

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

Table 105: 16 bit µController interface access types

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

6.5.4 Write access

CS and/or WR can be asserted in any order and the access will start when both are asserted. Either CS or WR may be permanently asserted. Address, byte high enable and write data are asserted on the falling edge of WR (active low). Once the µController interface is not BUSY, a rising edge on WR completes the µController access. At the end of a write access, CS and/or WR can be de-asserted in any order. Shortly after the rising edge of WR, the access can be finished by de-asserting 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 de-assertion.

Depending on the configuration (PDI async microcontroller extended configuration register 0x0152:0x0153[1] / 0x0192:0x0193[1]), the internal write access is either performed after the falling edge of WR, or after the rising edge of WR.

Write at falling edge of WR (0x0152:0x0153[1]=1 / 0x0192:0x0193[1]=1)
If the falling edge is selected, the internal write operation begins with the falling edge of WR, and BUSY indicates when the write operation is finished. The internal write operation is performed during the external write access.

Write at rising edge of WR (0x0152:0x0153[1]=0 / 0x0192:0x0193[1]=0)
If the rising edge of WR is selected, the internal operation begins with the rising edge of WR, i.e., after the external write access. Thus, the external write access is very fast, but an access immediately following will be delayed by the preceding write access. The worst-case access time is higher in this case.

6.5.5 Read access

CS and/or RD can be asserted in any order and the access will start when both are asserted. Either CS or RD may be 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. At the end of a read access, CS and/or RD can be de-asserted in any order.

With read busy delay configuration (PDI async microcontroller extended configuration register 0x0152:0x0153[0]=1 / 0x0192:0x0193[0]=1), BUSY de-assertion for read accesses can be additionally delayed by 20 ns, so external DATA setup requirements in respect to BUSY can be met.

Using PDI async microcontroller extended configuration register 0x0152:0x0153[9]=1 / 0x0192:0x0193[9]=1, the BHE signal can be ignored for read accesses, leading to full bus width read access regardless of BHE.

6.5.6 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.
- De-assertion of WR (or de-assertion of CS while WR remains asserted) while the µController interface is BUSY.
- De-assertion of RD (or de-assertion 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/0x0340 will be incremented.
- For A[0]=1 and BHE(act. low)=1 accesses, no access will be performed internally.
- De-assertion 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 de-asserted while the µController interface is BUSY (read has not finished), the access will be terminated internally. Internal byte transfers might still be completed, so special functions (e.g., SyncManager buffer switching) might be performed.
- For debugging, ESC configuration B4[11] can be used to enable a PDI error signal on the LED_RUN/LED_ERR pins, for PDI0/PDI1 respectively.

The reason for the access error can be found in the PDI error code register 0x030E:0x030F/ 0x0341:0x0342.

6.5.7 Default BUSY state

In normal mode (PDI async microcontroller extended configuration register 0x0152:0x0153[4]=0 / 0x0192:0x0193[4]=0), the ESC asserts the BUSY signal after an access is detected (CS and WR asserted/de-asserted, or CS and RD asserted). The BUSY signal remains active until the access is finished, or CS gets de-asserted.

In default busy state operation (PDI async microcontroller extended configuration register 0x0152:0x0153[4]=1 / 0x0192:0x0193[4]=1), the ESC indicates busy directly after CS is asserted – regardless or WR and RD. It remains busy until the access is finished (read data valid, write access finished). If the µController does not assert RD or WR, the ESC indicates busy until CS is de-asserted.

6.5.8 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.

6.5.9 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.

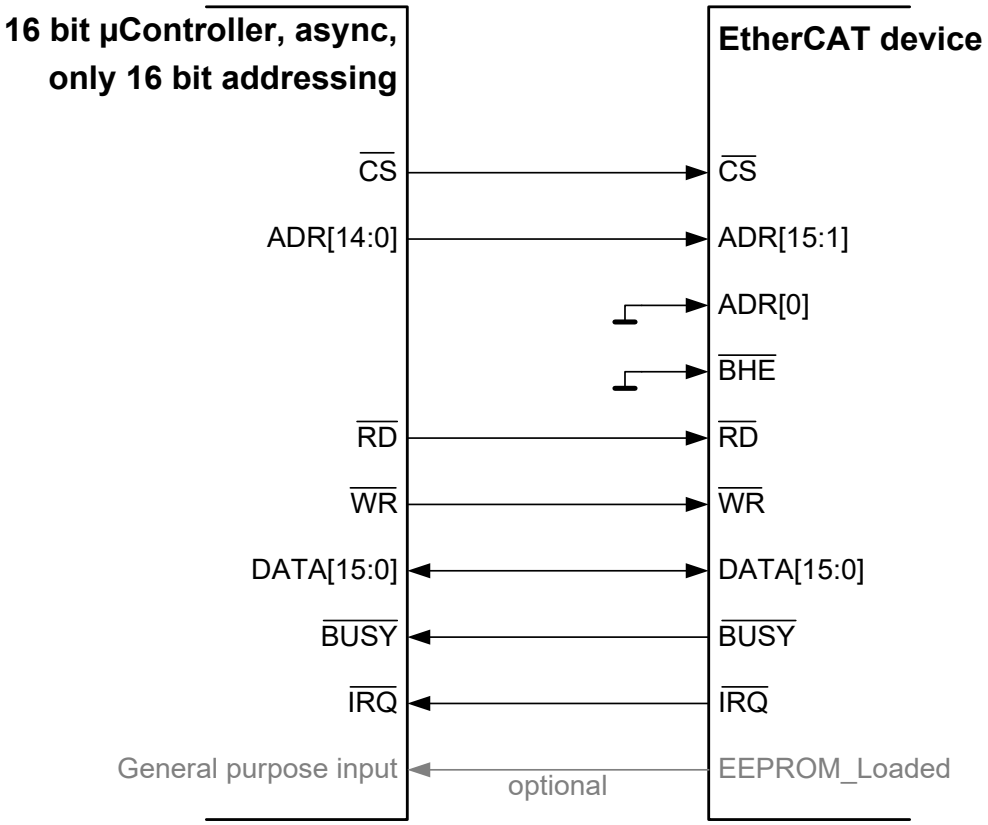

Figure 79: Connection with 16 bit µControllers without byte addressing

6.5.10 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.

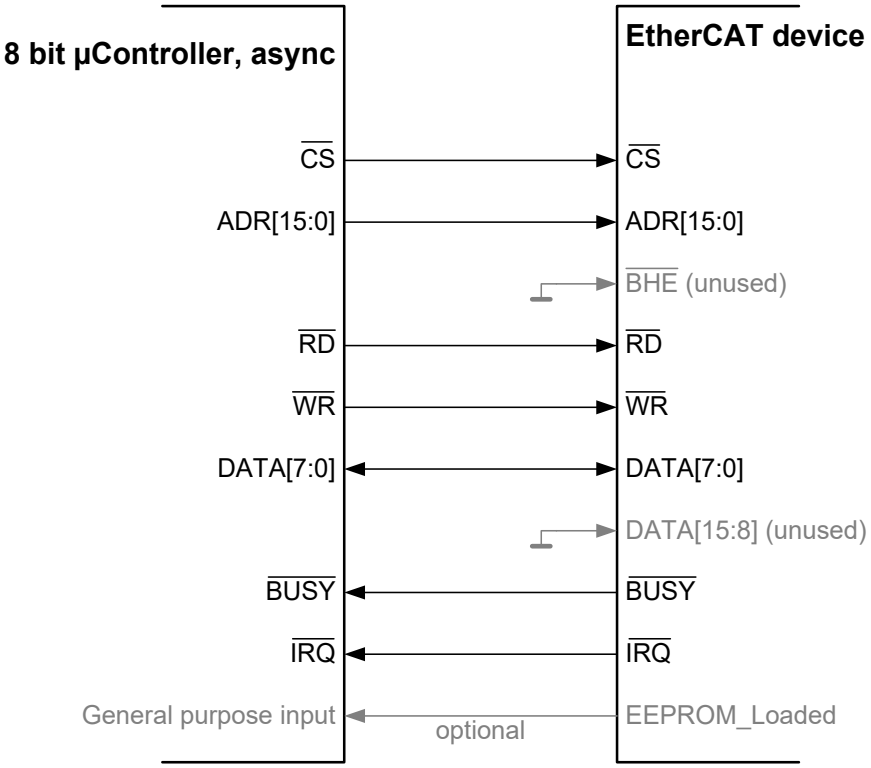

Figure 80: Connection with 8 bit µControllers (BHE and DATA[15:8] should not be left open)

6.5.11 Timing specification

Table 106: µController timing characteristics ET1150

Parameter	Min	Max	Comment
$t_{CS\_to\_BUSY}$	0 ns	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 de-assertion) with normal read busy output (0x0152[0]). Additional 20 ns with delayed read busy output.
		a) 65	a) write after falling edge of WR, or without preceding write access, or $t_{WR\_to\_RD} \geq t_{prec\_write}$
		b) 75 ns	b) worst case with preceding write access
$t_{BUSY\_to\_DATA\_valid}$		a) -2 ns b) -22 ns	DATA bus valid after device BUSY is de-asserted 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 de-assertion or RD de-assertion
$t_{CS\_to\_BUSY\_release}$	0 ns	15 ns	BUSY released after CS de-assertion
t_{CS_delay}	0 ns		Delay between CS de-assertion an assertion
t_{RD_delay}	5 ns		Delay between RD de-assertion and assertion
$t_{ADR\_BHE\_DATA\_setup}$	6 ns		ADR, BHE and write DATA valid before WR de-assertion
$t_{ADR\_BHE\_DATA\_hold}$	0 ns		ADR, BHE and write DATA valid after WR de-assertion
t_{WR_active}	5 ns		WR assertion time
$t_{BUSY\_to\_WR\_CS}$	0 ns		WR or CS de-assertion after BUSY de-assertion
$t_{WR\_to\_BUSY}$		15 ns	BUSY assertion after WR de-assertion
t_{write}	0 ns	50 ns	External write time (WR assertion to BUSY de-assertion)
t_{WR_delay}	5 ns		Delay between WR de-assertion and assertion
$t_{WR\_to\_RD}$	0 ns		Delay between WR de-assertion and RD assertion
$t_{CS\_WR\_overlap}$	5 ns		Time both CS and WR have to be de-asserted simultaneously (only if CS is de-asserted at all)
$t_{CS\_RD\_overlap}$	5 ns		Time both CS and RD have to be de-asserted simultaneously (only if CS is de-asserted 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

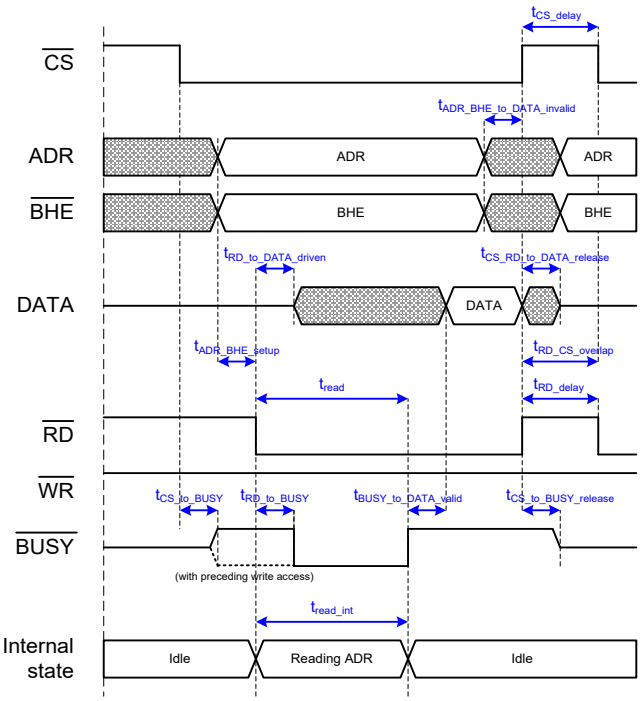

Figure 81: Read access (without preceding write access)

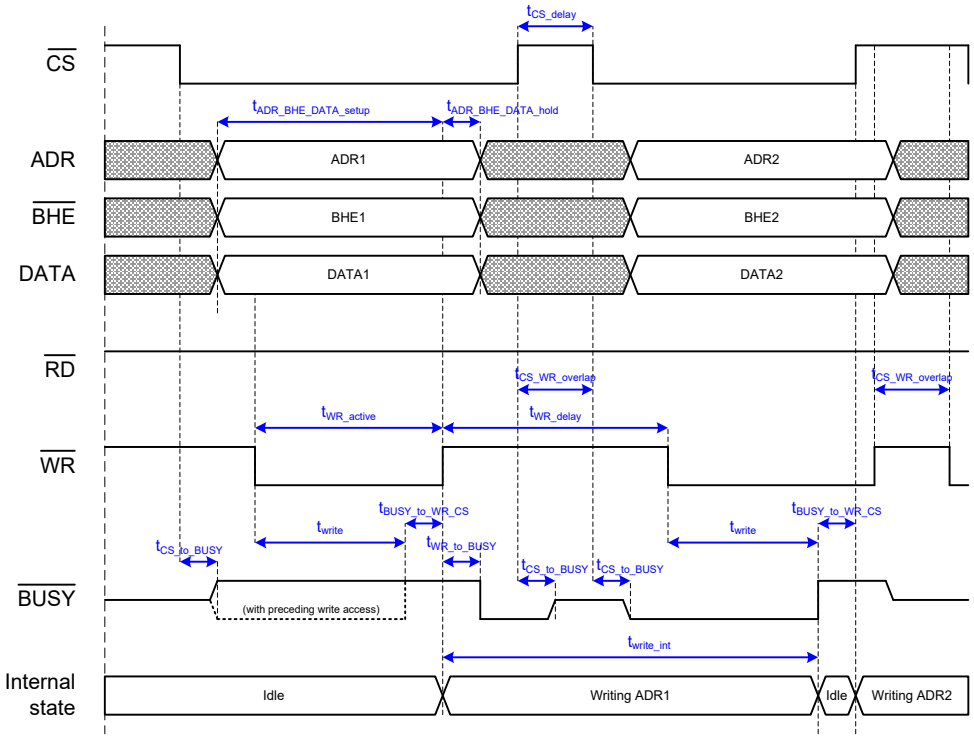

Figure 82: Write access (write after rising edge nWR, without preceding write access)

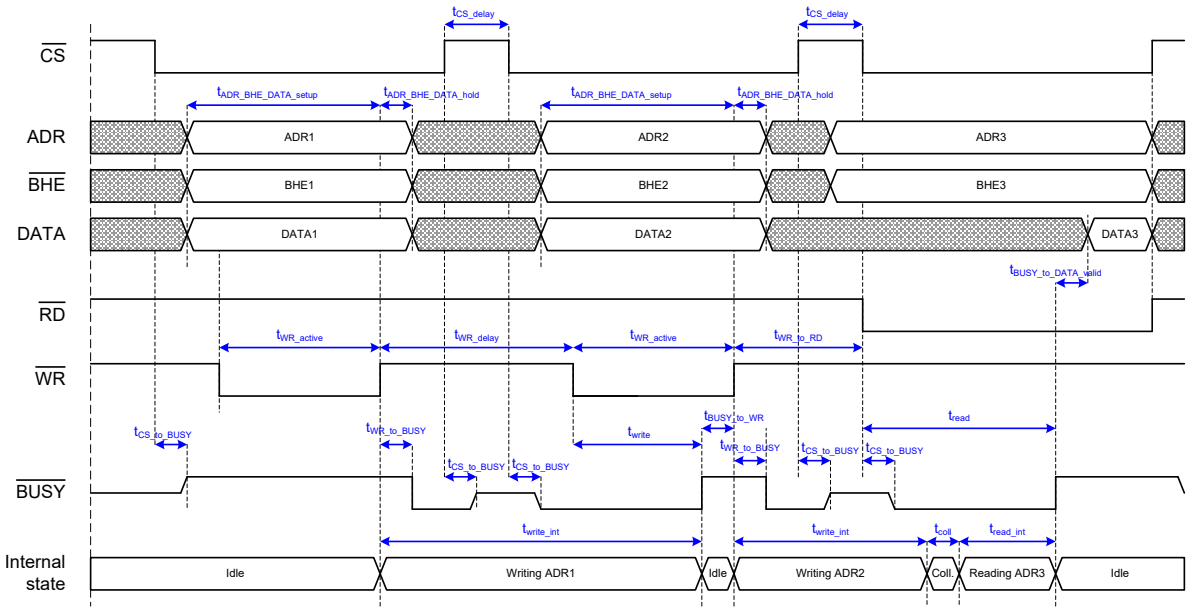

Figure 83: Sequence of two write accesses and a read access

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 de-asserted, BUSY is not driven anymore, nevertheless, the ESC is still writing to ADR1.

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

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

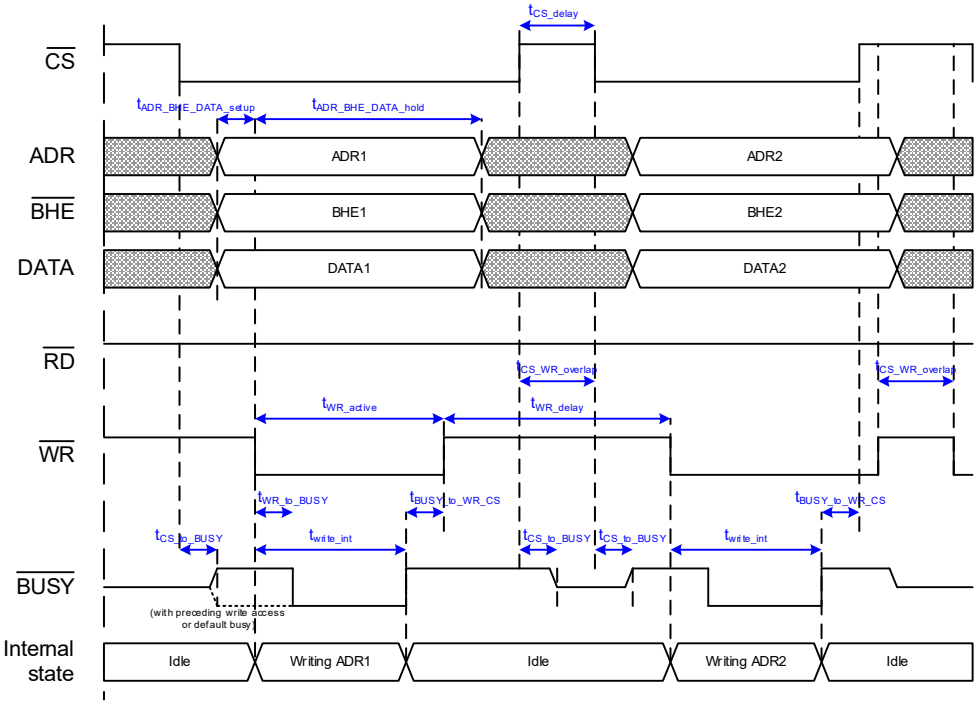

Figure 84: Write access (write after falling edge nWR)

6.6 Synchronous 8/16 bit µController interface

6.6.1 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:

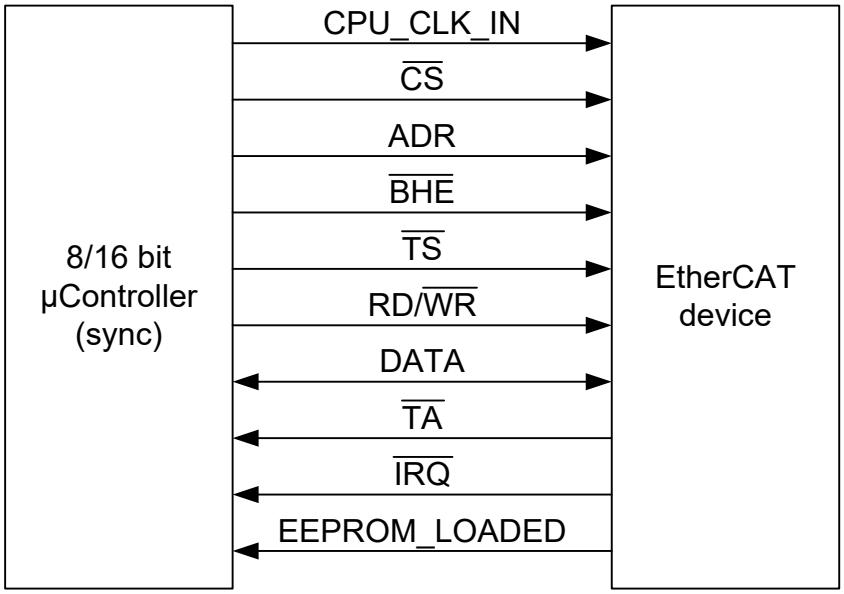

Figure 85: µController interconnection[3]

Table 107: µController signals

Signal	Direction		Description	Signal polarity
CPU_CLK_IN	IN	(µC → ESC)	µController interface clock	
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	Typical: act. low
TS	IN	(µC → ESC)	Transfer start	Typical: act. low
RD/nWR	IN	(µC → ESC)	Read (high) / write (low) access	
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
TA	OUT	(ESC → µC)	Transfer acknowledge	Typical: act. low
IRQ	OUT	(ESC → µC)	Interrupt	Typical: act. low
EEPROM_ LOADED	OUT	(ESC → µC)	PDI is active, EEPROM is loaded	act. high

6.6.2 Configuration

The 16 bit synchronous µController interface is selected with PDI type 0x0A in the PDI0 control register 0x0140, or PDI1 control register 0x0180, the 8 bit synchronous µController interface has PDI type 0x0B. It supports different configurations, which are located registers 0x0150:0x0153 / 0x0190:0x0193.

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

6.6.3 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 108: 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 (PDI multiplexed microcontroller extended configuration register 0x0152:0x0153[6] / 0x0192:0x0193[6]).

EtherCAT devices use Little Endian byte ordering, even with the synchronous µController interface which is typically connected to a µController using Big endian. 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 109: 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 110: 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

6.6.4 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 111: 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:

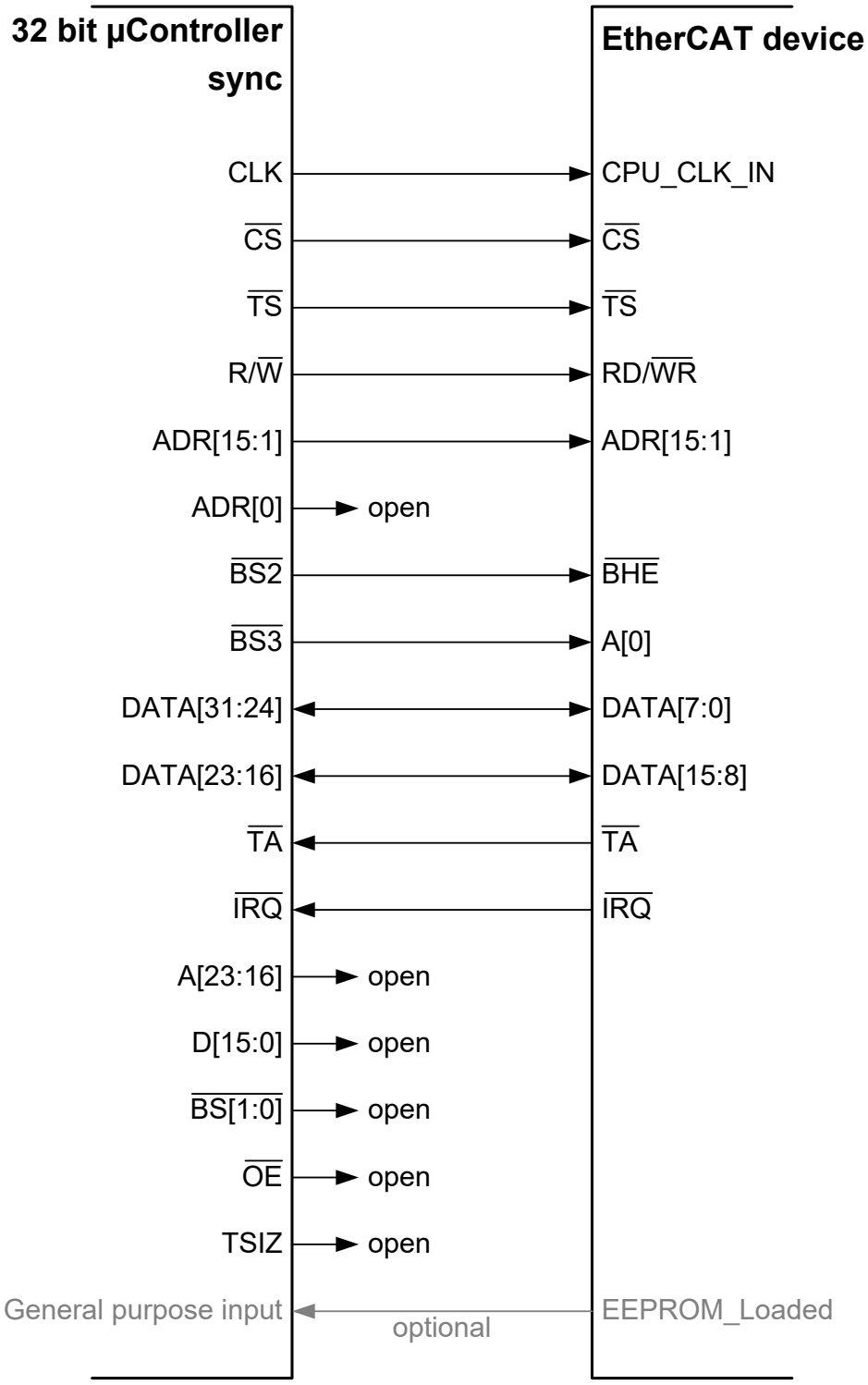

Figure 86: Synchronous 32 bit µController connection using byte select

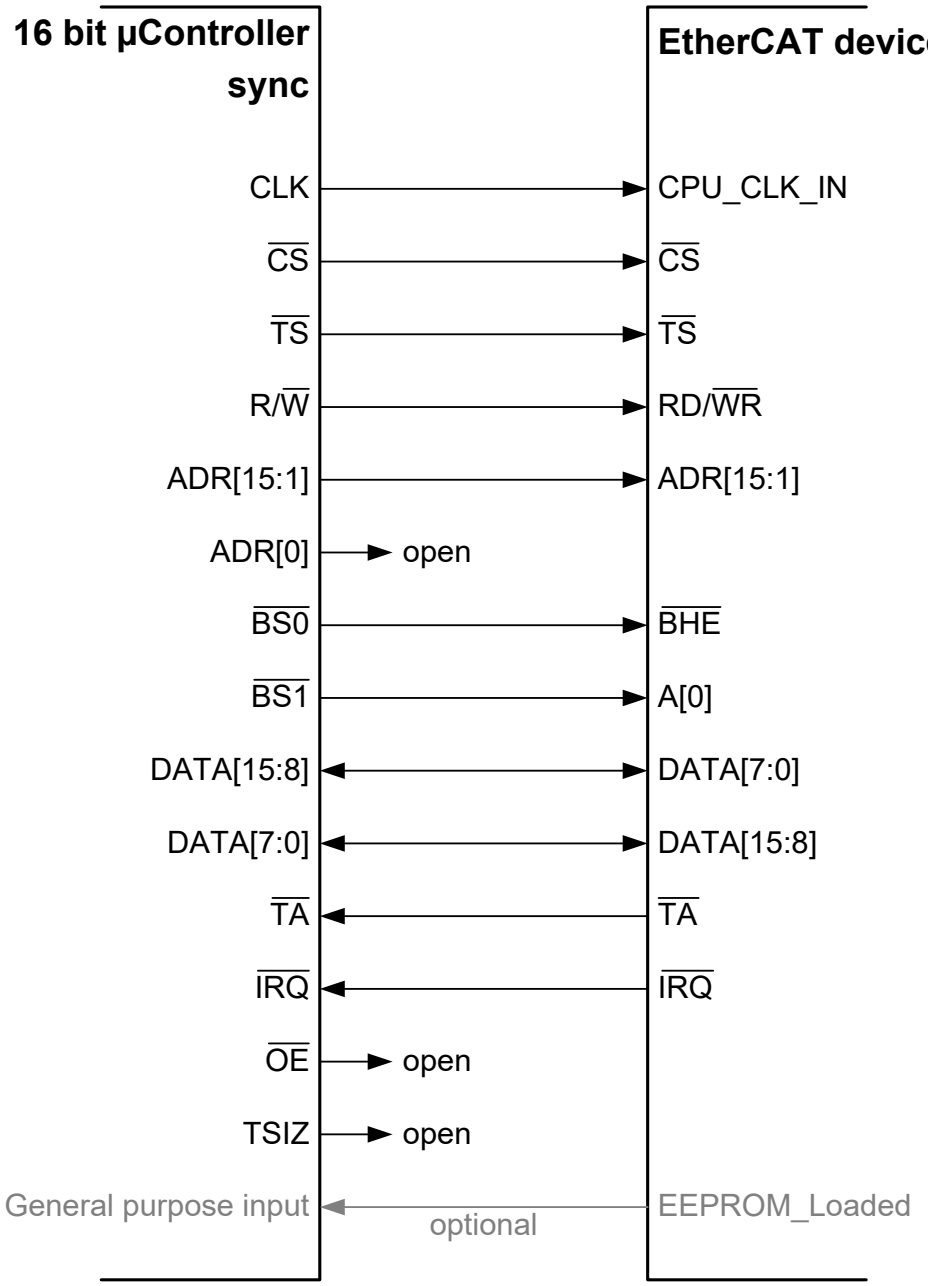

Figure 87: Synchronous 16 bit µController connection using byte select

6.6.5 µController connection using transfer size signals (SIZ)

In case the µController does not provide byte high enable, but 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 transfer size mode in register PDI synchronous microcontroller configuration 0x0150[6]=1 / 0x0190[6]=1.

Table 112: 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 (divided 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:

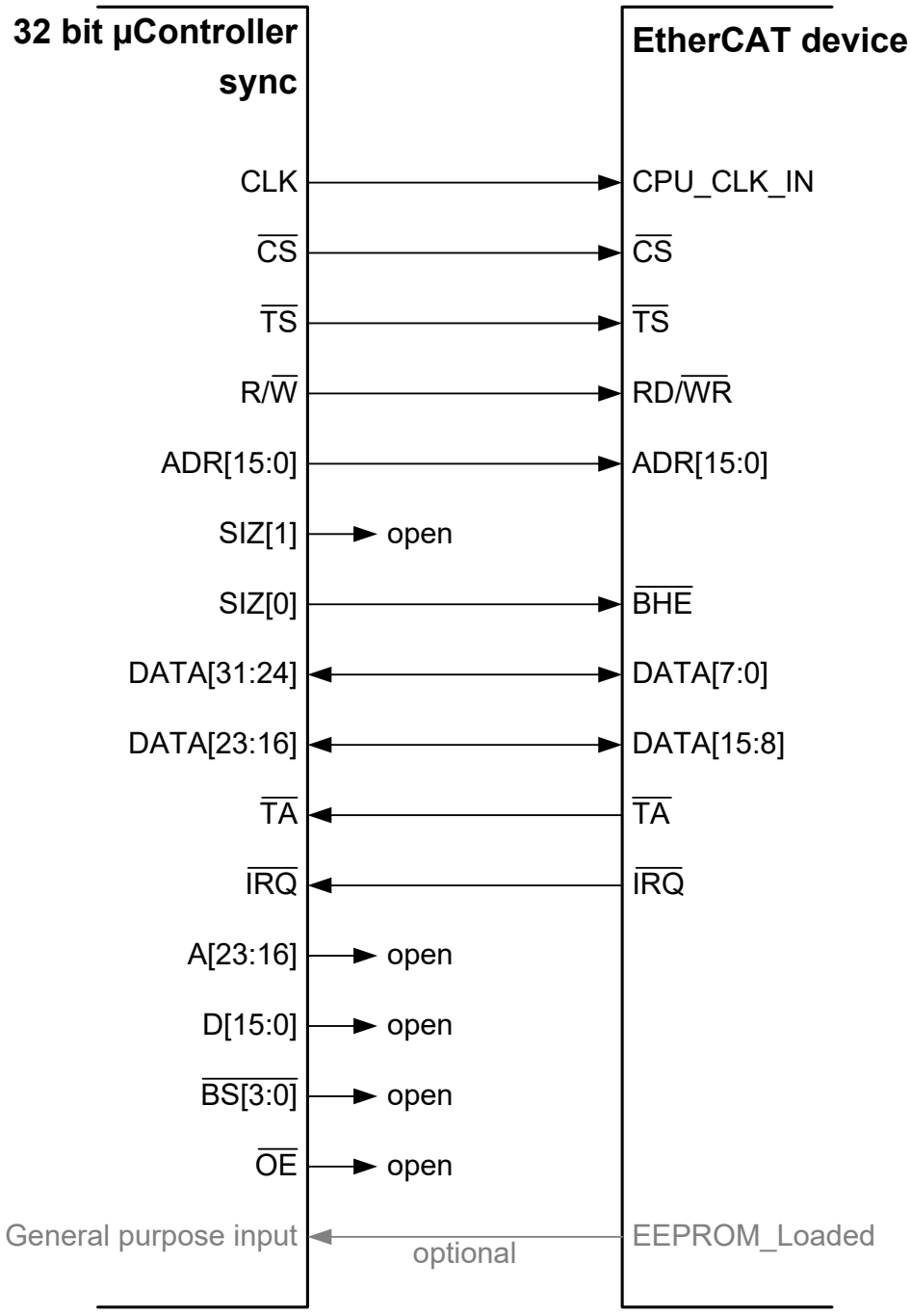

Figure 88: Synchronous 32 bit µController connection using transfer size

6.6.6 Write access

A write access starts with a transfer start (TS). Chip select can be asserted 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/nWR 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 (configurable to rising of falling edge of CPU_CLK_IN).

6.6.7 Read access

A read access starts with a transfer start (TS). Chip select can be asserted 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 RD/nWR 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 is asserted for one clock cycle (configurable to rising of falling edge of CPU_CLK_IN).

Some microcontrollers expect every read access 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.

6.6.8 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/0x0340 will be incremented.
- No access will be performed internally.
- For debugging, ESC configuration B4[11] can be used to enable a PDI error signal on the LED_RUN/LED_ERR pins, for PDI0/PDI1 respectively.

The reason for the access error can be read in the PDI error code register 0x030E:0x030F/ 0x0341:0x0342.

6.6.9 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. EEPROM_LOADED is synchronous to CPU_CLK_IN and will not go high unless CPU_CLK_IN is toggling.

6.6.10 Timing specification

Table 113: µController timing characteristics ET1150

Parameter	Min	Max	Comment
		PRELIMINARY TIMING	
t_{CLK}	20 ns		CPU_CLK_IN period (f_{CLK} ≤ 50 MHz)
t_{IN_setup}	7 ns		Input signals valid before CPU_CLK_IN edge (TS, CS, ADR, BHE, R/nWR, DATA)
t_{IN_hold}	3 ns		Input signals valid after CPU_CLK_IN edge (TS, CS, ADR, BHE, R/nWR, DATA)
$t_{CLK\_to\_OUT\_valid}$		15 ns	Output signals valid after CPU_CLK_IN edge (TA, IRQ, DATA)
$t_{CLK\_to\_OUT\_invald}$	3 ns		Output signals invalid after CPU_CLK_IN edge (TA, IRQ, DATA)
t_{read}		50 ns +3.0*t_{CLK} a) +0.5*t_{CLK} b) +0.5*t_{CLK} c) +t_{CLK}	External read time (TS to TA) Additional delay: a) extra delay if 0x0152.11=1 b) extra delay if 0x0152.10=1 c) extra delay if CS asserted one CPU_CLK_IN cycle after TS
t_{write}		40 ns +3.0*t_{CLK} a) +t_{CLK} b) +0.5*t_{CLK} c) +0.5*t_{CLK} d) +t_{CLK}	External write time (TS to TA) Additional 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
$t_{EEPROM\_LOADED\_to\_access}$	0 ns		Time between EEPROM_LOADED and first access
$t_{EEPROM\_LOADED\_to\_IRQ}$		0 ns	IRQ valid after EEPROM_LOADED

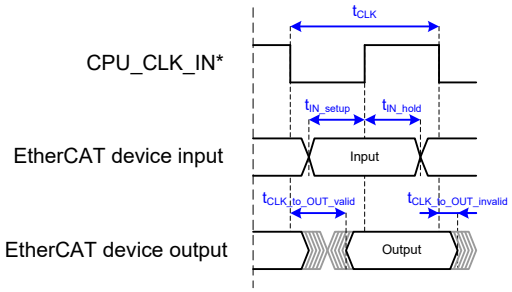

Figure 89: Basic synchronous µController interface timing (*refer to timing diagram for relevant CPU_CLK_IN edges)

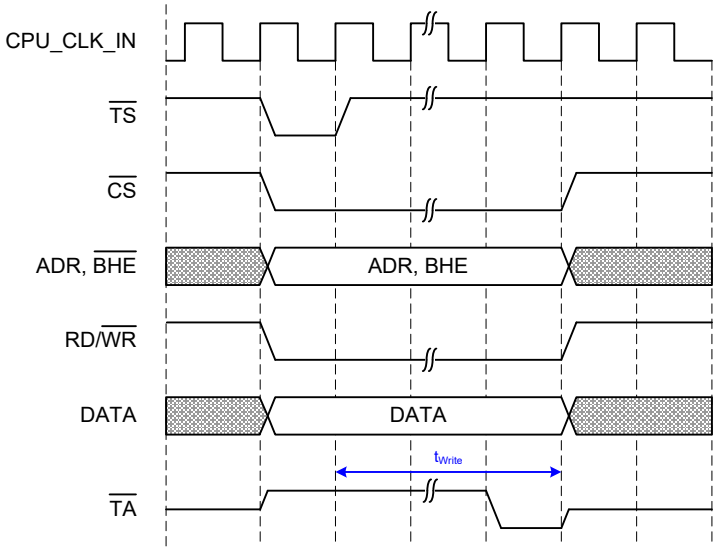

Figure 90: Write access (CS together with TS, write DATA together with CS, CS and TA on rising edge)

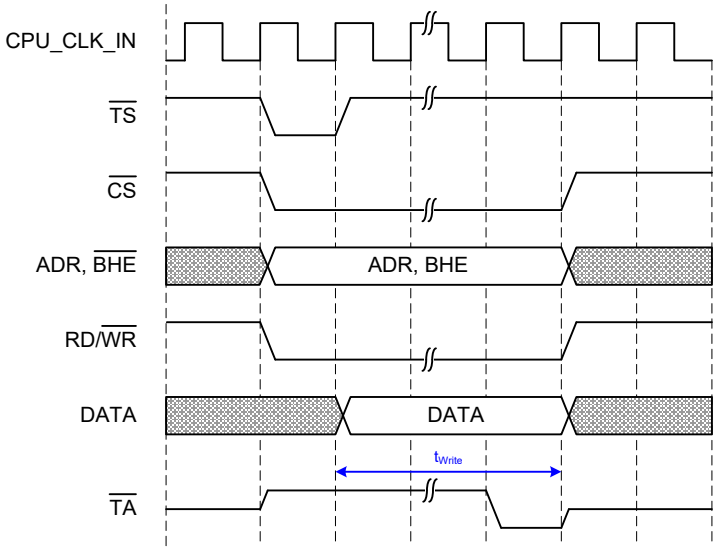

Figure 91: Write access (CS together with TS, write DATA after CS, CS and TA on rising edge)

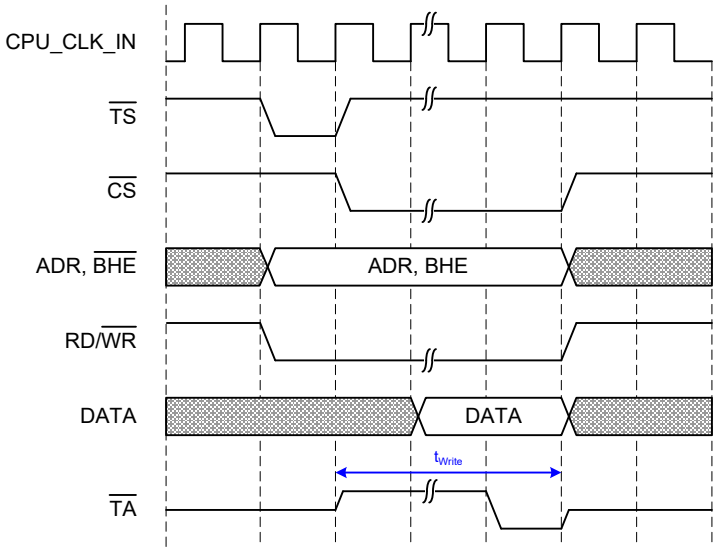

Figure 92: Write access (CS after TS, write DATA after CS, CS and TA on rising edge)

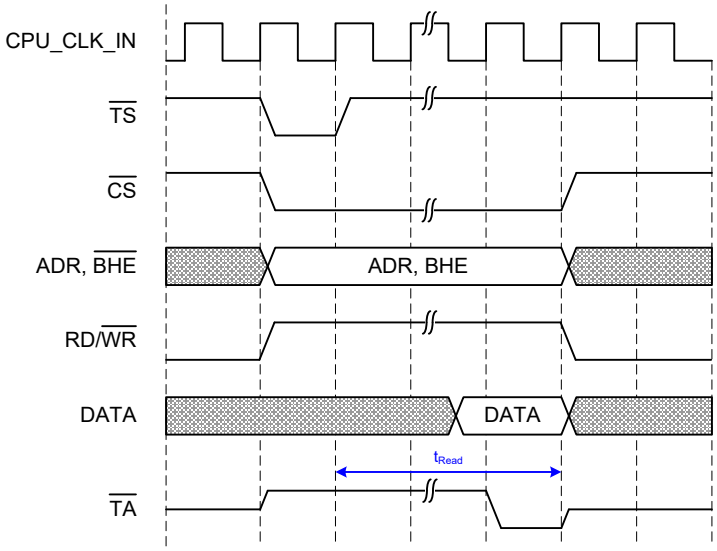

Figure 93: Read access (CS together with TS, CS and TA on rising edge)

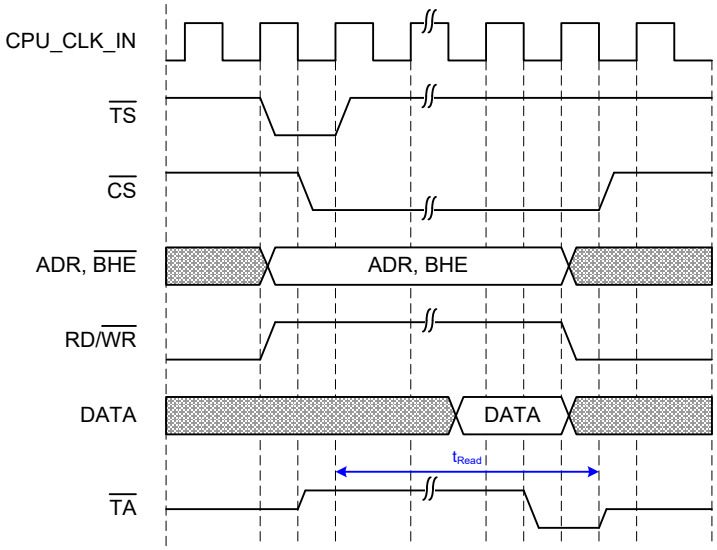

Figure 94: Read access (CS half a clock period after TS, CS and TA on falling edge)

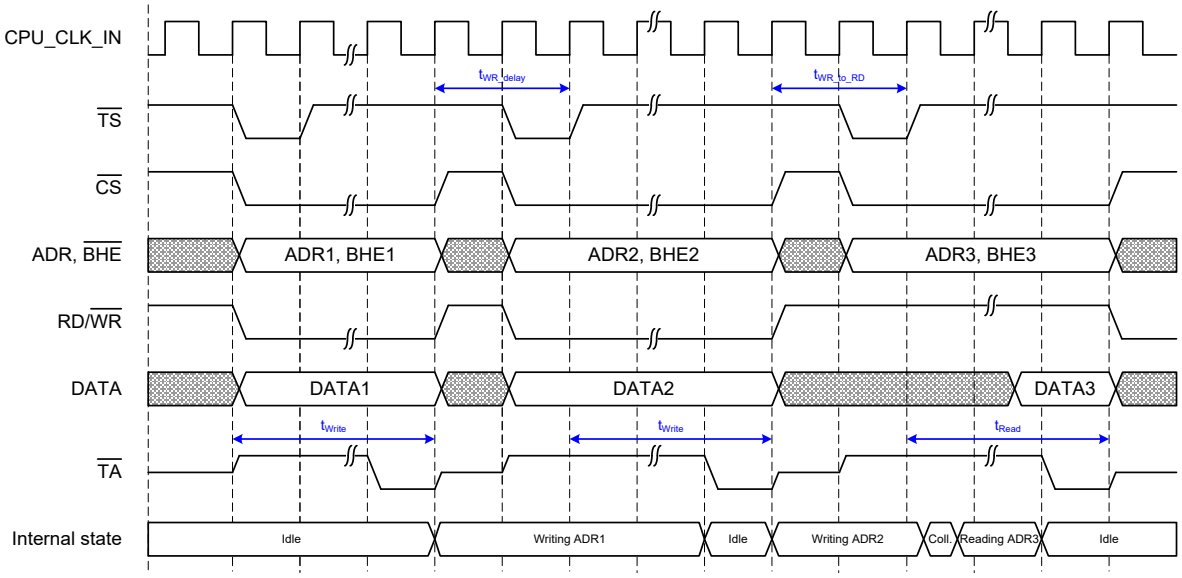

Figure 95: Sequence of two write accesses and a read access

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 de-asserted, TA is not driven anymore, 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.

6.7 Asynchronous multiplexed 8/16/32 bit µController interface

6.7.1 Interface

The asynchronous multiplexed µController interface uses a multiplexed address and data bus. The bidirectional data bus can be either 8 bit, 16 bit or 32 bit wide. The signals of the asynchronous multiplexed µController interface are:

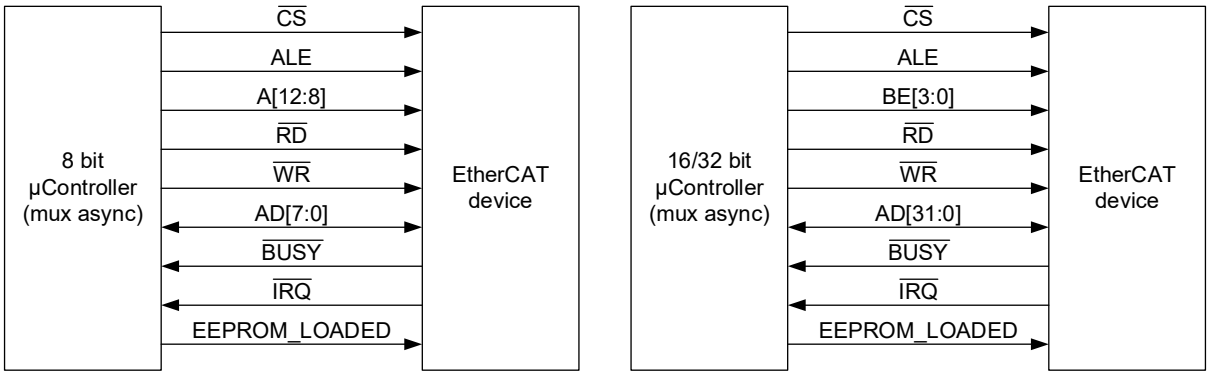

Figure 96: µController interconnection[4]

Table 114: µController signals

Signal	Direction		Description	Signal polarity
CS	IN	(µC → ESC)	Chip select	Typical: act. low
ALE	IN	(µC → ESC)	Adress latch enable	Typical: act. high
A[15:8]	IN	(µC → ESC)	Address bus upper bits for 8 bit interface	Typical: act. high
BE[3:0]	IN	(µC → ESC)	Byte enable or 16/32 bit interface	Typical: act. low
RD	IN	(µC → ESC)	Read command	Typical: act. low
WR	IN	(µC → ESC)	Write command	Typical: act. low
AD[7:0]	BD	(µC ↔ ESC)	Address/data bus for 8 bit interface	act. high
AD[15:0]	BD	(µC ↔ ESC)	Address/data bus for 16 bit interface	act. high
AD[31:0]	BD	(µC ↔ ESC)	Address/data bus for 32 bit 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.

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

6.7.2 Configuration

The 8/16/32 bit synchronous µController interface is selected with PDI type 0x0C/0x0D in the PDI0 control register 0x0140, or PDI1 control register 0x0180. It supports different configurations, located in registers 0x0150:0x0153 / 0x0190:0x0193.

Table 115: Asynchronous multiplexed µController configuration

Bus width	PDI control register PDI0: 0x0140 PDI1: 0x0180	PDI extended configuration register PDI0: 0x0152:0x0153[3] PDI1: 0x0192:0x0193[3]
8	0x0D	0
16	0x0C	0
32	0x0D	1

6.7.3 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, and the 32 bit µController interface supports 8 bit, 16 bit, and 32 bit read/write accesses.

EtherCAT devices use Little Endian byte ordering.

6.7.3.1 µController 16 bit interface access types

The byte enable signals, or alternatively, the least significant address bit A[0] together with byte high enable (BHE, equals BE[1]) are used to distinguish between 8 bit low byte access, 8 bit high byte access and 16 bit access.

Table 116: 8 bit high/low byte and 16 bit access distinction

ADR[0]	BHE/BE[1] (act. low). 0x0152[6]=0 / 0x0192[6]=0	BE[1:0] (act. low). 0x0152[6]=1 / 0x0192[6]=1	Access
0	0	00	16 bit access to ADR[15:0] and ADR[15:0]+1 (low and high byte)
0	1	10	8 bit access to ADR[15:0] (low byte, even address)
1	0	01	8 bit access to ADR[15:0] (high byte, odd address)
1	1	11	invalid access

If byte high enable (BHE/BE[1]) is used, the byte access mode configuration bit has to be set to zero (PDI multiplexed microcontroller extended configuration register 0x0152:0x0153[6] / 0x0192:0x0193[6]).

6.7.3.2 µController 32 bit interface access types

The byte enable signals BE[3:0] are used to select individual bytes for the access. Enabled bytes have to be continuous, selecting either 8 bit, 16 bit, or 32 bit.

6.7.4 Address latching

Any access starts with an address phase. The address is placed on A[12:8] and AD[7:0] for the 8 bit interface, or on AD[15:0] for the larger bus widths, then ALE is asserted. The chip select (CS) status is not evaluated for address latching. After ALE is released, either a write access or a read access can be performed.

6.7.5 Write access

A write access starts with assertion of chip select (CS), if it is not permanently asserted. Write data is 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 by de-assertion of WR or CS. Shortly after the rising edge of WR, the access can be finished by de-asserting 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 de-assertion.

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

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

6.7.6 Read access

A read access starts with assertion of chip select (CS), if it is not permanently asserted. The falling edge of RD starts 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 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 de-assertion for read accesses can be additionally delayed for 20 ns, so external DATA setup requirements in respect to BUSY can be met.

6.7.7 µController access errors

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

- De-assertion of WR (or de-assertion of CS while WR remains asserted) while the µController interface is BUSY.
- De-assertion of RD (or de-assertion of CS while RD remains asserted) while the µController interface is BUSY (read has not finished).
- Read or write access without any byte enable.

A wrong µController access will have these consequences:

- The PDI error counter 0x030D/0x0340 will be incremented.
- De-assertion 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 de-asserted 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.
- For debugging, ESC configuration B4[11] can be used to enable a PDI error signal on the LED_RUN/LED_ERR pins, for PDI0/PDI1 respectively.

The reason for the access error can be read in the PDI error code register 0x030E:0x030F/ 0x0341:0x0342.

6.7.8 Timing specification

Table 117: µController timing characteristics ET1150

Parameter	Min	Max	Comment
PRELIMINARY TIMING			
$t_{CS\_to\_BUSY}$		15 ns	BUSY driven and valid after CS assertion
t_{ADR_setup}	7 ns		ADR valid before ALE assertion
t_{ADR_hold}	0 ns		ADR valid after ALE de-assertion
t_{ALE_delay}	10 ns		Delay between ALE changes
$t_{RD\_to\_DATA\_driven}$	0 ns	15 ns	DATA bus driven after RD assertion
$t_{RD\_to\_BUSY}$	0 ns	15 ns	BUSY asserted after RD assertion
t_{read}		65 ns	External read time (RD assertion to BUSY de-assertion) with normal read busy output (0x0152[0]). Additional 20 ns with delayed read busy output.
$t_{BUSY\_to\_DATA\_valid}$		a) -2 ns b) -22 ns	DATA bus valid after device BUSY is de-asserted a) normal read busy output b) delayed read busy output
$t_{CS\_RD\_to\_DATA\_release}$	0 ns		DATA bus released after CS de-assertion or RD de-assertion
$t_{CS\_to\_BUSY\_release}$	0 ns	15 ns	BUSY released after CS de-assertion
t_{CS_delay}	0 ns		Delay between CS de-assertion an assertion
t_{RD_delay}	5 ns		Delay between RD de-assertion and assertion
$t_{BE\_DATA\_setup}$	6 ns		BE and write DATA valid before WR de-assertion (write at rising edge nWR)
$t_{BE\_DATA\_hold}$	0 ns		BE and write DATA valid after WR de-assertion (write at rising edge nWR)
t_{WR_active}	5 ns		WR assertion time
$t_{BUSY\_to\_WR\_CS}$	0 ns		WR or CS de-assertion after BUSY de-assertion
$t_{WR\_to\_BUSY}$		15 ns	BUSY assertion after WR de-assertion
t_{write}	0 ns	50 ns	External write time (WR assertion to BUSY de-assertion)
t_{WR_delay}	5 ns		Delay between WR de-assertion and assertion
$t_{WR\_to\_RD}$	0 ns		Delay between WR de-assertion and RD assertion
$t_{CS\_WR\_overlap}$	5 ns		Time both CS and WR have to be de-asserted simultaneously (only if CS is de-asserted at all)
$t_{CS\_RD\_overlap}$	5 ns		Time both CS and RD have to be de-asserted simultaneously (only if CS is de-asserted at all)

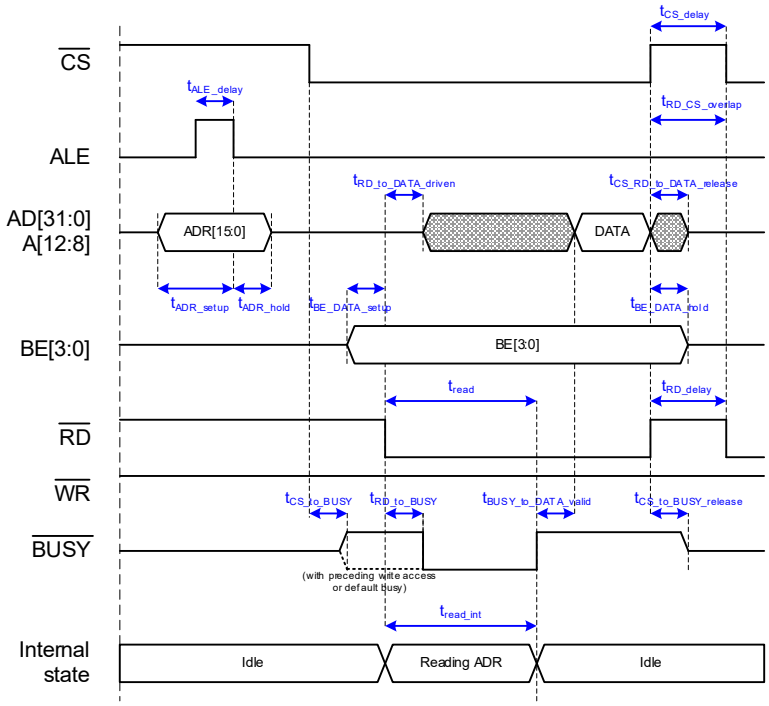

Figure 97: Read access (without preceding write access)

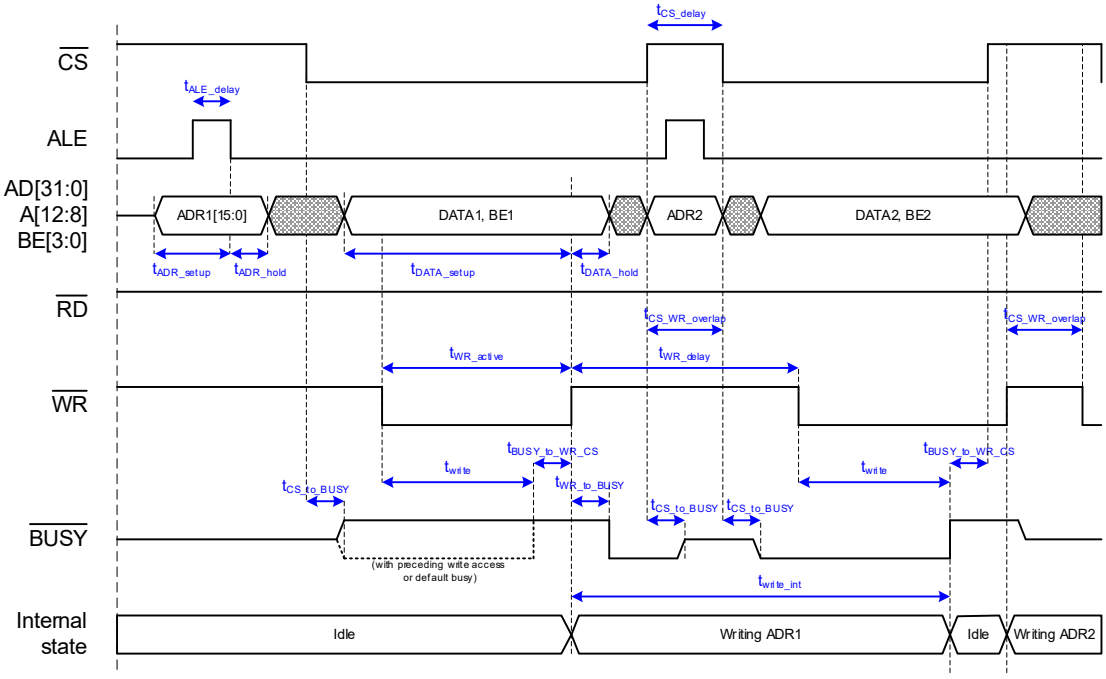

Figure 98: Write access (write after rising edge nWR)

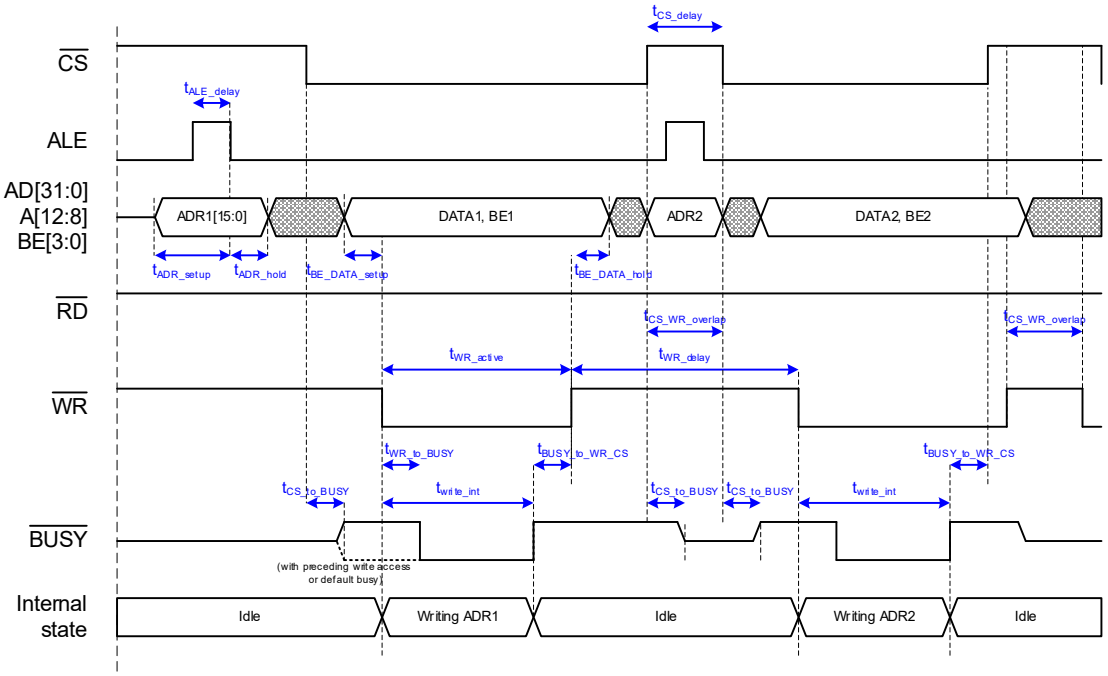

Figure 99: Write access (write after falling edge nWR)

7 Distributed clocks DC_SYNC/LATCH signals

For details about the distributed clocks refer to section I.

7.1 Signals

The distributed clocks unit of the ET1150 has the following external signals:

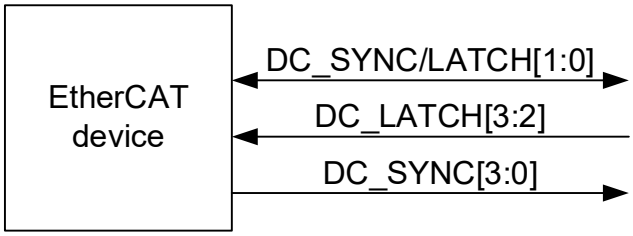

Figure 100: Distributed clocks signals

When the pins DC_SYNC_LATCH[1:0] are configured to provide DC_LATCH[1:0], all 4 DC LatchSignals and all 4 DC SyncSignals can be assigned to ET1150 pins.

DC_SYNC[2] output driver/polarity is same as DC_SYNC[0] output driver/polarity. DC_SYNC[3] output driver/polarity is same as DC_SYNC[1] output driver/polarity.

Table 118: Distributed clocks signals

Signal	Direction	Description
DC_SYNC/LATCH[1:0]	OUT/IN	SyncSignal (OUT) or LatchSignal (IN), direction bitwise configurable via sync/latch configuration register A 0x0151.
DC_LATCH[3:2]	IN	LatchSignal[3:2], part of DC signal block
DC_SYNC[3:0]	OUT	SyncSignal[3:0], part of DC signal block

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

7.2 Timing specification

Table 119: DC_SYNC/DC_LATCH timing characteristics ET1150

Parameter	Min	Max	Comment
t_{DC_LATCH}	15 ns		Time between DC_LATCH[3:0] events (each bit)
$t_{DC\_SYNC\_jitter}$		15 ns	DC_SYNC[3:0] output jitter
$t_{DC\_SYNC\_pulse\_IRQ}$	40 ns		SYNC0/1 pulse width if the SYNC0/1 signal is used as AL event request (PDI interrupt) signal (masked by register 0x0220 ff.), and if acknowledge mode is not used.

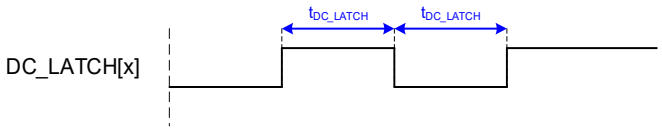

Figure 101: LatchSignal timing

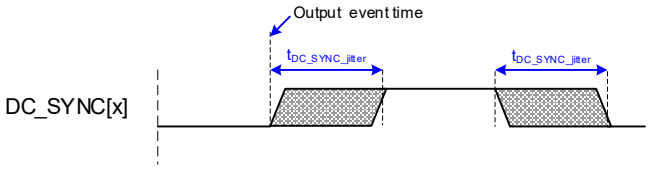

Figure 102: SyncSignal timing

8 SII EEPROM interface (I²C)

For details about the ESC SII EEPROM interface refer to section I. The SII EEPROM interface is intended to be a point-to-point interface between ET1150 and I²C EEPROM. If other I²C masters are required to access the I²C bus, the ET1150 must be held in reset state (e.g. for in-circuit-programming of the EEPROM).

8.1 Signals

The EEPROM interface of the ET1150 has the following signals:

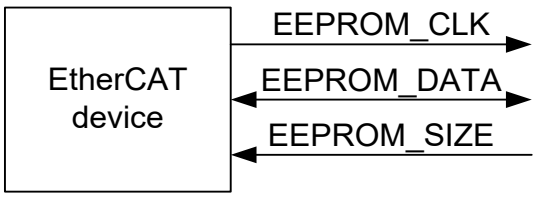

Figure 103: I²C EEPROM signals

Table 120: 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 ET1150.

8.2 Timing specification

Table 121: SII EEPROM timing characteristics

Parameter	Typical		Comment
	1 Kbit-16 Kbit	32 Kbit-4 Mbit	
PRELIMINARY TIMING			
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 c) ~ 2.12 ms	a) ~ 740 µs b) ~ 1.22 ms c) ~ 2.18 ms	Read access time (without errors): a) 4 words b) configuration (8 Words) c) configuration (16 Words)
t_{Delay}	400 ns		Time until configuration loading begins after reset is gone

9 OTP configuration

The ET1150 includes a one-time programmable memory array for device information, configuration, and application usage. This memory array has a total size of 256 byte. The default value for each bit is 0, and it can be programmed to 1 once. A bit that is already 1 cannot be reprogrammed to 0.

The OTP can be programmed at any time in-system, with normal supply voltages, in 3 ways:

- EtherCAT masters can program the OTP via ESC registers 0x0E30:0x0E3B
- µControllers attached to the PDI can also program the OTP via ESC registers 0x0E30:0x0E3B
- The JTAG interface allows programming of the OTP by a dedicated instruction

9.1 OTP sections

The 256 byte OTP memory is divided into 4 sections, intended for different purposes:

Table 122: OTP sections

#	Section name	Size	Description
1	Fab section	32 byte	The fab section is programmed during ESC production. It contains production information, e.g. date code of the test, and trimming values. This section is write protected.
2	Feature section	64 byte	The feature section contains ESC options controlled by Beckhoff. This section is write protected.
3	User section	128 byte	The user section contains ESC configuration options open to implementors, e.g. Vendor ID, serial number, product date code, forcing EEPROM/POR options, etc.. This section is not write protected.
4	Application section	32 byte	The application section is not used by the ESC. The usage of the app section free for implementors or field applications. This section is not write protected.

9.2 Write protection and checksum

Each section has two reserved bytes for write protection and a checksum of the section content. A section with active write protection can never be written again. The checksum status is available in the power-on value register 0xE00[39:36].

The checksum uses the CRC12 polynomial $x^{12}+x^8+x^7+x^6+x^5+x^2+x+1$ for the content of the section, excluding the last two bytes with write protection and CRC. The CRC initialization value is 0x000, the LSB of the content is used first. The checksum of a completely unprogrammed section is zero, and therefore valid.

The write protection of a section is implemented using 4 bits. All bits shall be programmed to 1 for activation of the write protection.

9.3 OTP content

The OTP content is described in a separate html document, which includes CRC calculation.

9.3.1 Fab section

Table 123: Fab section

Byte	Length	Type	Description
0x00	12 byte	ASCII (12 character)	ESC LOT, as printed on package
0x0C	1 byte	uint8	Wafer number
0x0D	2 byte	int16	Wafer x coordinate
0x0F	2 byte	int16	Wafer y coordinate
0x11	2 byte	uint16	ESC Build, ESC register 0x0003:0x0002
0x14	4 byte	ASCII (4 character)	ESC test date code YYWW (year – week), typically close to package date code
0x1E	2 byte		[11:0] CRC for bytes 0x00-0x1D [15:12] Write protect fab section

Note: Other bytes/bits are reserved, they shall not be programmed.

9.3.2 Feature section

Table 124: Feature section

Byte	Length	Type	Description
0x5E	2 byte		[11:0] CRC for bytes 0x20-0x5D [15:12] Write protect fab section

Note: Other bytes/bits are reserved, they shall not be programmed.

9.3.3 User section

Table 125: User section

Byte	Length	Type	Description
0x60	14 byte		ESC configuration area A override values
0x70	14 byte		ESC configuration area A override mask (0=EEPROM, 1 = OTP override value)
0x80	14 byte		ESC configuration area B override values
0x90	14 byte		ESC configuration area B override mask (0=EEPROM, 1 = OTP override value)
0xA0	4 byte	Hex	Vendor ID
0xA4	4 byte	Hex	Serial number
0xA8	4 byte	Hex	Product date code
0xAC	5 byte		POR override values (register 0x0E00:0x0E004)
0xB1	5 byte		POR override mask (0=EEPROM, 1 = OTP override value)
0xB6	1 byte		Port 0 [4:0] PHY address [5] Use OTP PHY address
0xB7	1 byte		Port 1 [4:0] PHY address [5] Use OTP PHY address
0xB8	1 byte		Port 2 [4:0] PHY address [5] Use OTP PHY address
0xB9	1 byte		Port 3 [4:0] PHY address [5] Use OTP PHY address

Byte	Length	Type	Description
0xBA	3 byte		PHY management initialization command 1 [15:0] Write data [20:16] Register [22] Enable PHY initialization [23] Command 2 available
0xBD	3 byte		PHY management initialization command 2 [15:0] Write data [20:16] Register [23] Command 3 available
0xC0	3 byte		PHY management initialization command 3 [15:0] Write data [20:16] Register [23] Command 4 available
0xC3	3 byte		PHY management initialization command 4 [15:0] Write data [20:16] Register
0xC6	1 byte		[0] EEPROM emulation allows any PDI configuration [4] LDO disable (for external supply only)
0xDE	2 byte		[11:0] CRC for bytes 0x60-0xDD [15:12] Write protect fab section

Note: Other bytes/bits are reserved, they shall not be programmed.

ESC configuration area override
Overriding both EEPROM configuration areas by OTP completely, speeds up device startup, as EEPROM will not be loaded then.

Vendor ID, serial number, product date code
Usage and layout of these 12 bytes is user specific, this is a suggestion.

ESC configuration area override
Overriding the POR values completely, speeds up device startup, as the timeout until POR values are settled is skipped.

PHY management initialization commands
The commands are written to PHYs of all MII/RGMII ports once at startup. E.g. used to enable RGMII in-band status.

9.3.4 Application section

Table 126: Application section

Byte	Length	Type	Description
0xE0	30 byte	Any	Free for any use case. This data is never evaluated by the ESC itself, but it could be used by a µController
0xFE	2 byte		[11:0] CRC for bytes 0xE0-0xFE [15:12] Write protect fab section

9.4 OTP access

After ESC power-up, the OTP is activated and read by the ESC. Afterwards, the OTP is deactivated again. For user access to the OTP, it has to be activated first, using OTP commands for OTP control register 0x0E32:0x0E33. Different activation command sequences have to be performed for read access only, or for read/write access:

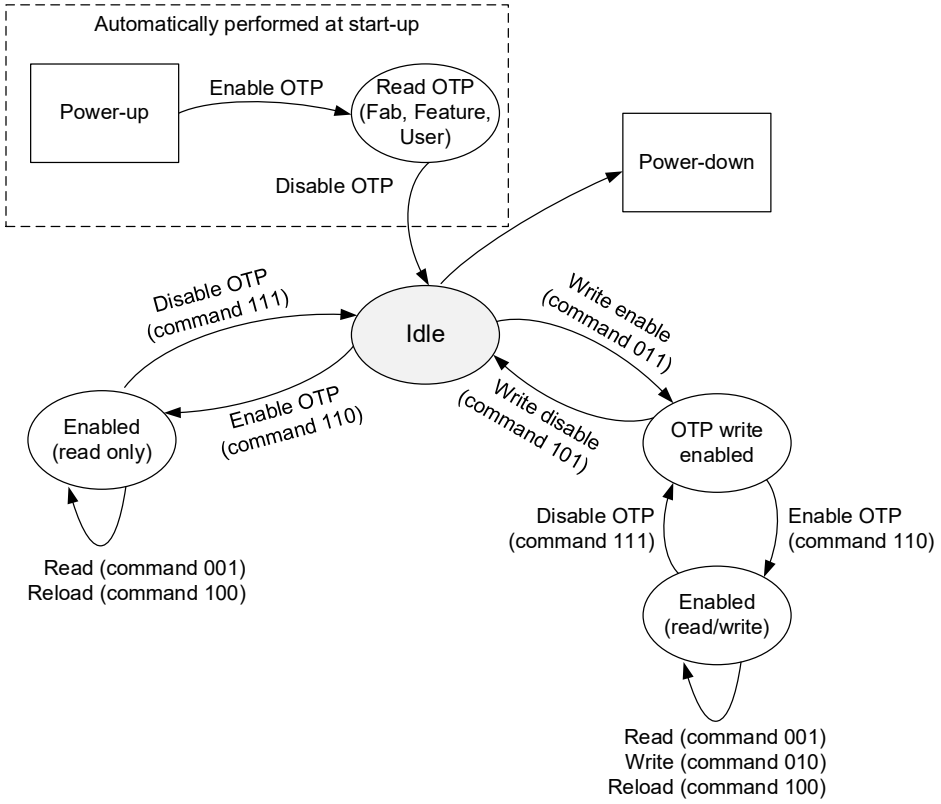

Figure 104: OTP activation/deactivation

After OTP operations, before potential power-down, the OTP should be disabled. If you use the app section for your application, we suggest that you read the relevant data on startup and cache them in memory for further accesses.

9.4.1 OTP read example

The following steps have to be performed for an OTP read access:

1.	Enable OTP:	Write 0x0E32 := 0x0600 (16 bit)
2.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
3.	Check OTP status:	Read 0x0E32, should be 0x0001
4.	Set address:	Write 0x0E34 := <OTP address> (32 bit, 8 bit used, [0:1]=00)
5.	Read command:	Write 0x0E32 := 0x0100 (16 bit)
6.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
7.	Check error bits:	Read 0x0E32, should be 0x0001
8.	Get read data:	Read 0xE38 = <read data> (32 bit)
9.	Disable OTP:	Write 0x0E32 := 0x0700 (16 bit)
10.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step

9.4.2 OTP programming example

The following steps have to be performed for OTP programming:

1.	Enable OTP writing:	Write 0x0E32 := 0x0300 (16 bit)
2.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
3.	Check OTP status:	Read 0x0E32, should be 0x0002
4.	Enable OTP:	Write 0x0E32 := 0x0600 (16 bit)
5.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
6.	Check OTP status:	Read 0x0E32, should be 0x0003
7.	Set address:	Write 0x0E34 := <OTP address> (32 bit, 8 bit used, [0:1]=00)
8.	Set write data:	Write 0x0E38 := <write data> (32 bit)
9.	Write command:	Write 0x0E32 := 0x0200 (16 bit)
10.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
11.	Check error bits:	Read 0x0E32, should be either 0x0003 (write success, no other bits were already set), or 0x0803 (write success, some bits were already set) Any other write error (not programmed, or weakly programmed): retry once, if still fails, discard ESC: OTP malfunction.
12.	Disable OTP:	Write 0x0E32 := 0x0700 (16 bit)
13.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step
14.	Disable OTP writing:	Write 0x0E32 := 0x0500 (16 bit)
15.	Check busy:	Read 0x0E32, if 0x0E32[15]=1 (busy), repeat this step

10 JTAG

The ET1150 supports JTAG boundary scan, as well as OTP programming and device debugging over JTAG.

JTAG pins should be made available on the PCB, even if boundary scan testing is not used, e.g. as test points

10.1 Interface

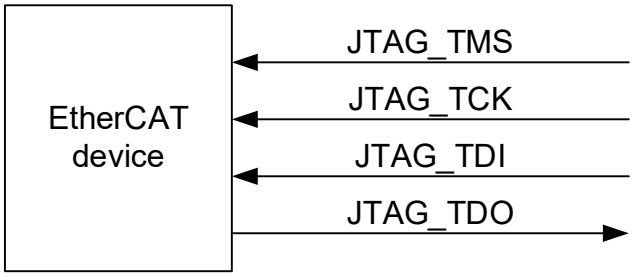

Figure 105: JTAG signals

Table 127: JTAG signals

Signal	Direction	Description	Idle value
JTAG_TMS	IN	Mode select	High
JTAG_TCK	IN	Clock	Low
JTAG_TDI	IN	Data input	High
JTAG_TDO	OUT	Data output	

10.2 JTAG reset

JTAG reset is performed by the power supply supervisors, i.e., JTAG reset is active during power-up, then released.

10.3 Instruction register (IR) and ESC status

The instruction register (8 bits), is used to receive instructions, and it delivers basic ESC status information:

Table 128: JTAG instruction register

Bit	TDI JTAG instruction	TDO ESC status
7	Instruction[7]	0
6	Instruction[6]	OTP checksum ok 0: CRC error or still loading, see [3] 1: CRC ok
5	Instruction[5]	RAM BIST result 0: BIST ok or still running, see [2] 1: BIST error
4	Instruction[4]	POR values status 0: not latched 1: valid
3	Instruction[3]	OTP initial loading status 0: finished 1: busy
2	Instruction[2]	RAM BIST status 0: finished 1: busy
1	Instruction[1]	0
0	Instruction[0]	1

The JTAG interface supports the following instructions. All other instructions are reserved, selecting reserved instructions potentially destroys the ESC.

Table 129: JTAG instructions

Name	Instruction code	Description
Standard instructions		
IDCODE	0x01	Device identification
SAMPLE/PRELOAD	0x02	Boundary scan SAMPLE/PRELOAD
EXTEST	0x20	Boundary scan EXTEST
CLAMP	0x22	Boundary scan CLAMP
HIGHZ	0x23	Disable output drivers
BYPASS	0xFF	Bypass ET1150
Custom instructions		
OTP_SERIAL	0x80	OTP programming
INFO	0x81	Status information
All other instructions		
Reserved	Other	Reserved, do not use (may damage ESC)

10.4 IDCODE instruction 0x01

The IDCODE of the ET1150 is 0x011507A7.

Table 130: IDCODE register

Bit	TDI	TDO	Description
31	-	0	JTAG version [3:0] = 0000
30	-	0	
29	-	0	
28	-	0	
27	-	0	Part number [15:0] = 0x1150
26	-	0	
25	-	0	
24	-	1	
23	-	0	
22	-	0	
21	-	0	
20	-	1	
19	-	0	
18	-	1	
17	-	0	
16	-	1	
15	-	0	
14	-	0	
13	-	0	
12	-	0	
11	-	0	Manufacturer ID [10:0] = 0x3D3
10	-	1	
9	-	1	
8	-	1	
7	-	1	
6	-	0	
5	-	1	
4	-	0	
3	-	0	
2	-	1	
1	-	1	
0	-	1	Static value

10.5 OTP_SERIAL instruction 0x80

This instruction can be used to read and program the OTP. The principle is the same as via EtherCAT or PDI.

Attention: OTP address must be 32 bit aligned, i.e., OTP address [1:0] = 00.

Table 131: OTP_SERIAL register

Bit	TDI	TDO	Description
55:24	Write OTP data register [31:0]	Read OTP data register [31:0]	0x0E38:0x0E3B
23:16	Write OTP address register [7:0]; [1:0] must be 00	Read OTP address register [7:0]	0x0E34:0x0E37
15	0	0: OTP interface idle 1: OTP interface busy	0x0E32[15]
14	0	0: last command successful 1: Command error	0x0E32[14]
13	0	0: write allowed 1: area write protected	0x0E32[13]
12	0	OTP write verification result [1:0] 00: success 01: success, additional bits 10: error 11: error, weakly programmed	TDI: Start OTP access TDO: 0x0E32[12:11]
11	0: No command start 1: OTP command start using command in bits [10:8]		
10:8	Write OTP command register [2:0]	Read OTP command status [2:0]	0x0E32[10:8]
7:2	0	0	Reserved
1	0	0: OTP write disabled 1: OTP write enabled	0x0E32[1]
0	0	0: OTP disabled 1: OTP enabled	0x0E32[0]

10.6 INFO instruction 0x81

Note: status of latch bits [76:68] is undefined after power-on.

Table 132: INFO register

Bit	TDI	TDO	Description
79	-	1	
78	-	0	
77	-	OSC_IN status: 0: OSC_IN not toggling 1: OSC_IN valid	
76	-	PLL reference slip latch: 0: normal function 1: slip detected	0x0E10[13] Cleared by INFO latch reset
75	-	PLL feedback slip latch: 0: normal function 1: slip detected	0x0E10[12] Cleared by INFO latch reset
74	-	V_{CC_CORE} undervoltage fine latch 0: normal 1: undervoltage detected	0x0E10[8] Cleared by INFO latch reset
73	-	V_{CC_PLL} undervoltage fine latch 0: normal 1: undervoltage detected	0x0E10[8] Cleared by INFO latch reset
72	-	V_{CC_CORE} undervoltage coarse latch 0: normal 1: undervoltage detected	0x0E10[7] Cleared by INFO latch reset
71	-	V_{CC_PLL} undervoltage coarse latch 0: normal 1: undervoltage detected	0x0E10[6] Cleared by INFO latch reset
70	-	V_{CC_PDI} undervoltage coarse latch 0: normal 1: undervoltage detected	0x0E10[5] Cleared by INFO latch reset
69	-	V_{CC_COM} undervoltage latch 0: normal 1: undervoltage detected	0x0E10[4] Cleared by INFO latch reset
68	-	V_{CC_REG} undervoltage latch 0: normal 1: undervoltage detected	0x0E10[3] Cleared by INFO latch reset
67	-	V_{CC_PLL} status: 0: undervoltage 1: ok	
66	-	V_{CC_PDI} status: 0: undervoltage 1: ok	
65	-	V_{CC_COM} status: 0: undervoltage 1: ok	
64	-	V_{CC_REG} status: 0: undervoltage 1: ok	
63:48	-	Build number [15:0]	0x0003:0x0002 Valid when OTP not busy
47:1	-	POR values [47:0]	0x0E00:0x0E05 Valid when POR values latched
0	0: no latch reset 1: INFO latch reset		

10.7 Boundary scan ET1150

A BSDL file for boundary scan is available, please refer to that file for cell order.

Bidirectional IO pads have two boundary scan cells:

- output driver enable (nOE)
- output value (data)

The boundary scan chain of the ET1150 is ordered as

TDI → cell 168 (nOE TX_DATA02_x_C25SHI0)
→ cell 167 (data TX_DATA02_x_C25SHI0)
→ … (cells 166 to 2) …
→ cell 1 (nOE PDI44)
→ cell 0 (data PDI44)
→ **TDO**

10.8 Timing specification

Table 133: JTAG timing characteristics ET1150

Parameter	Min	Max	Comment
		PRELIMINARY VALUES	
t_{CLK}	40 ns		JTAG clock period (f_{CLK} = 25 MHz)

11 Electrical and mechanical specifications

11.1 Absolute maximum conditions

Exceeding absolute maximum conditions may cause permanent damage to the device. Functional operation at absolute maximum ratings is not implied, exposure for extended periods may affect device reliability.

Table 134: Absolute maximum conditions

Symbol	Parameter	Condition	Min	Max	Units
PRELIMINARY VALUES					
$V_{CC\_REG\_IN}$	$V_{CC\_REG\_IN}$ - GND		-0.3	3.6	V
$V_{CC\_REG\_OUT}$	$V_{CC\_REG\_IN}$ - $V_{CC\_REG\_OUT}$		-0.3	3.6	V
$V_{CC\_IO\_COM}$	$V_{CC\_IO\_COM}$ - GND		-0.3	3.6	V
$V_{CC\_IO\_PDI}$	$V_{CC\_IO\_PDI}$ - GND		-0.3	3.6	V
V_{CC_CORE}	V_{CC_CORE} - GND		-0.3	1.3	V
V_{CC_PLL}	V_{CC_PLL} - GND		-0.3	1.3	V
V_{pin}	V_{pin} - GND	Non-supply pins	-0.3	V_{Domain} + 0.3	V
I_{REG_OUT}	LDO/DC-DC output current		-2	200	mA
P_{TOT}	Total power dissipation			600	mW
V_{ESD}	ESD protection	Human body model ANSI/ESDA/JEDEC JS001-2014	2		kV
I_{INJ_IO}	Injection current into I/O pins	$V_{pin} > V_{Domain}$ or $V_{pin} < GND$	-2	2	mA

11.2 Operating conditions

11.2.1 Power supply

Table 135: Power supply

Symbol	Parameter	Condition	Min	Typ	Max	Units
PRELIMINARY VALUES						
$V_{CC\_REG\_IN}$	$V_{CC\_REG\_IN}$ - GND		2.25	2.5/3.3	3.6	V
$V_{CC\_REG\_OUT}$	LDO/DC-DC output voltage			1.1		V
$I_{CC\_REG\_OUT}$	LDO/DC-DC output current		0		50	mA
V_{CC_CORE}	Logic power supply		0.99	1.10	1.21	V
V_{CC_PLL}	PLL power supply		0.99	1.10	1.21	V
$V_{CC\_IO\_COM}$	I/O COM power supply	3.3V operation	3.0	3.3	3.6	V
$V_{CC\_IO\_COM}$	I/O COM power supply	2.5V operation	2.25	2.5	2.75	V
$V_{CC\_IO\_COM}$	I/O COM power supply	1.8V operation	1.62	1.8	1.98	V
$V_{CC\_IO\_PDI}$	I/O PDI power supply	3.3V operation	3.0	3.3	3.6	V
$V_{CC\_IO\_PDI}$	I/O PDI power supply	2.5V operation	2.25	2.5	2.75	V
$V_{CC\_IO\_PDI}$	I/O PDI power supply	1.8V operation	1.62	1.8	1.98	V

11.2.2 Electrical characteristics

Table 136: DC characteristics ET1150

Symbol	Parameter	Condition	Min	Typ	Max	Units
		PRELIMINARY VALUES				
Voltage supervisor						
V_{UVD_Reg}	Undervoltage threshold for V_{CC_REG}					V
$V_{UVD\_I/O}$	Undervoltage threshold for $V_{CC\ PDI}$, V_{CC_COM}	$V_{CC\ I/O}$=3.3V a) rising b) falling		 a) 2.71 b) 2.62		V
$V_{UVD\_I/O}$	Undervoltage threshold for $V_{CC\ PDI}$, V_{CC_COM}	$V_{CC\ I/O}$=2.5V a) rising b) falling		 a) 2.03 b) 1.96		V
$V_{UVD\_I/O}$	Undervoltage threshold for $V_{CC\ PDI}$, V_{CC_COM}	$V_{CC\ I/O}$=1.8V a) rising b) falling		 a) 1.52 b) 1.47		V
V_{UVD_Core}	Undervoltage threshold (fine) for V_{CC_CORE}/V_{CC_PLL}	a) rising b) falling		a) 0.90 b) 0.86		V
V_{POR_Core}	Undervoltage threshold (coarse) for V_{CC_CORE}/V_{CC_PLL}	a) rising b) falling		a) 0.64 b) 0.61		V
I/O pins						
V_{IL}	Input low voltage (not OSC_IN)	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	-0.3		a) 0.8 b) 0.7 c) 0.6	V
V_{IH}	Input high voltage (not OSC_IN)	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 2.0 b) 1.7 c) 1.2		a) 3.6 b) 2.8 c) 2.1	V
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
V_{HYST}	Schmitt trigger hysteresis	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 250 b) 200 c) 150			mV
V_{OL}	Output low voltage				0.4	V
V_{OH}	Output high voltage	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 2.9 b) 2.1 c) 1.4			V
I_{OH}	Output high current				4	mA
I_{OL}	Output low current				-4	mA
R_{PU_3K3}	Pull-up resistor 3K3	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 2.4 b) 2.2 c) 2.0	a) 3.3 b) 3.3 c) 3.3	a) 4.7 b) 5.0 c) 5.3	kΩ
R_{WPU}	Weak internal pull-up resistor	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 30 b) 32 c) 32	a) 50 b) 50 c) 50	a) 82 b) 82 c) 82	kΩ
R_{WPD}	Weak internal pull-down resistor	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V c) $V_{CC\ I/O}$=1.8V	a) 34 b) 27 c) 24	a) 50 b) 50 c) 50	a) 77 b) 85 c) 82	kΩ
OSC_IN/OSC_OUT						
$V_{I\ OSC\_IN}$	Input voltage OSC_IN	a) $V_{CC\ I/O}$=3.3V b) $V_{CC\ I/O}$=2.5V			a) 3.6 b) 2.75	V

Symbol	Parameter	Condition	Min	Typ	Max	Units
$V_{IT\ OSC\_IN}$	Input threshold voltage OSC_IN (no Schmitt trigger)		1.0	1.3	1.9	V
C_{OSC}	OSC_IN/OSC_OUT pin capacitance			1.0		pF

NOTE: R_{WPU}/R_{WPD} cannot be used externally, their full effectiveness appears only inside the ET1150 (implemented as transistors).

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

11.2.2.1 Power consumption

A separate tool for estimation of the power consumption in different configurations is available, please refer to that tool for details.

This is just an example with:

- SPI slave PDI
- 3 FMMU
- 4 SyncManager
- Distributed clocks SyncSignal 0
- Internal LDO used

Table 137: Power consumption ET1150

Symbol	Parameter	Condition	Typ	Units
		PRELIMINARY VALUES		
I_{CC_REG}	2x MII	a) $V_{CC\ REG}$=3.3V b) $V_{CC\ REG}$=2.5V	a) 13 b) 12	mA
I_{CC_COM}	2x MII	a) $V_{CC\ COM}$=3.3V b) $V_{CC\ COM}$=2.5V c) $V_{CC\ COM}$=1.8V	a) 2 b) 1 c) 1	mA
I_{CC_PDI}	2x MII	a) $V_{CC\ PDI}$=3.3V b) $V_{CC\ PDI}$=2.5V c) $V_{CC\ PDI}$=1.8V	a) 4 b) 3 c) 2	mA

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

11.2.3 Timing characteristics

Table 138: Timing characteristics

Symbol	Parameter	Min	Typ	Max	Units
	PRELIMINARY TIMING				
f_{CLK25}	Clock source (OSC_IN) with initial accuracy	25 MHz ± 25 ppm			
$t_{CLK25OUT1}$	CLK25OUT1 rising edge after OSC_IN rising edge, V_{CC_PDI} = 3.3V	-1.2		2.4	ns
$t_{CLK25OUT2}$	CLK25OUT2 rising edge after OSC_IN rising edge, V_{CC_PDI} = 3.3V	-1.0		2.3	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 V_{CC_PDI} = V_{CC_COM} = 3.3V	a) -0.3 b) -0.6 c) -0.4		a) 4.7 b) 4.6 c) 4.5	ns
$t_{CPU\_CLK\_OUT1}$	CPU_CLK_OUT1 (25 MHz) rising edge after OSC_IN rising edge, V_{CC_PDI} = 3.3V	-0.5	1	3.8	ns
t_{POR_Sample}	POR value sample time after power good a) normal POR strapping b) without POR strapping (fixed by OTP)		a) 84 b) 14		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}	ET1150 reset output time	80	84		ms
t_{Reset_Func}	ET1150 functional after RESET signal high (EEPROM not loaded, PDI not functional)			50	µs
$t_{Startup}$	Startup time (PDI operational after power good, I²C EEPROM, without SII loading error) a) normal POR strapping b) without POR strapping (fixed by OTP)			a) 340 b) 275	ms

NOTE: If an active clock source is used (external crystal oscillator instead of crystal) and it is not operational at power-good time, startup will be delayed (t_{POR_Sample} will be longer)

The timing characteristics of the PDIs, distributed clocks, EEPROM I²C interface, and MII interface can be found in their respective chapters.

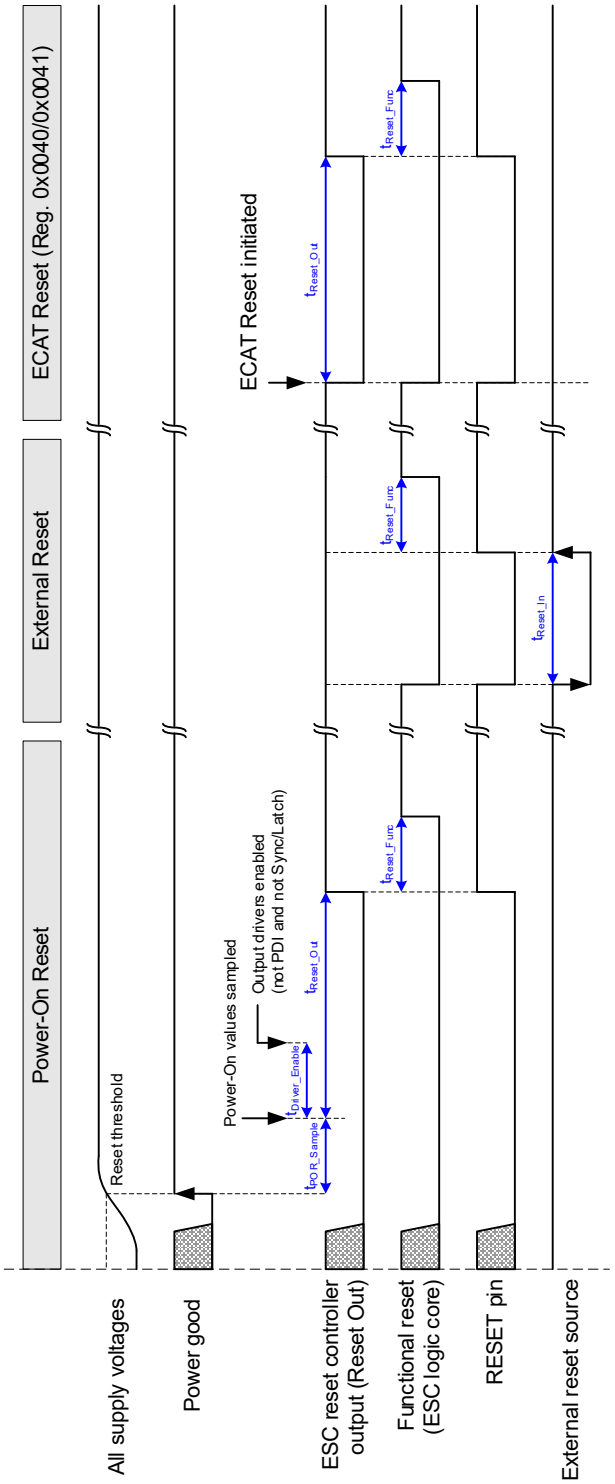

Figure 106: Reset timing

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

Table 139: Forwarding delays

Symbol	Parameter	Min	Average	Max	Units
	PRELIMINARY TIMING				
t_{Diff}	Average difference processing delay minus forwarding delay (without RX FIFO jitter) between any two ports				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) b)	a) b)	a) b)	ns

NOTE: Average timings are used for distributed clocks calculations.

11.2.4 Thermal characteristics

Table 140: Thermal characteristics ET1150

Symbol	Parameter	Min	Typ	Max	Units
	PRELIMINARY VALUES				
ϑ_A	Ambient temperature	-40		85	°C
ϑ_J	Junction temperature	-40		125	°C
Θ_{JA}	Thermal resistance theta junction to ambient		51.3		°C/W
Θ_{JB}	Thermal resistance theta junction to board		37.8		°C/W
Θ_{JC}	Thermal resistance theta junction to case		19.7		°C/W
Ψ_{JT}	Thermal resistance psi junction to top		3.51		°C/W
Ψ_{JB}	Thermal resistance psi junction to bottom		35.4		°C/W

11.3 Mechanical specifications

11.3.1 Package information

A 10mm x 10mm TFBGA (Thin-profile Fine-pitch BGA) with 128 balls is used for the ET1150. The ball pitch is 0.8mm, with 0.43mm ball diameter. The material of the balls is 98.25% Sn / 1.2% Ag / 0.5% Cu / 0.05% Ni (SAC125N).

The ET1150 is compliant to RoHS 2 (2011/65/EU) including amendment "COMMISSION DELEGATED DIRECTIVE (EU) 2015/863".

Cleaning PCB with de-ionized water (DI water) is allowed.

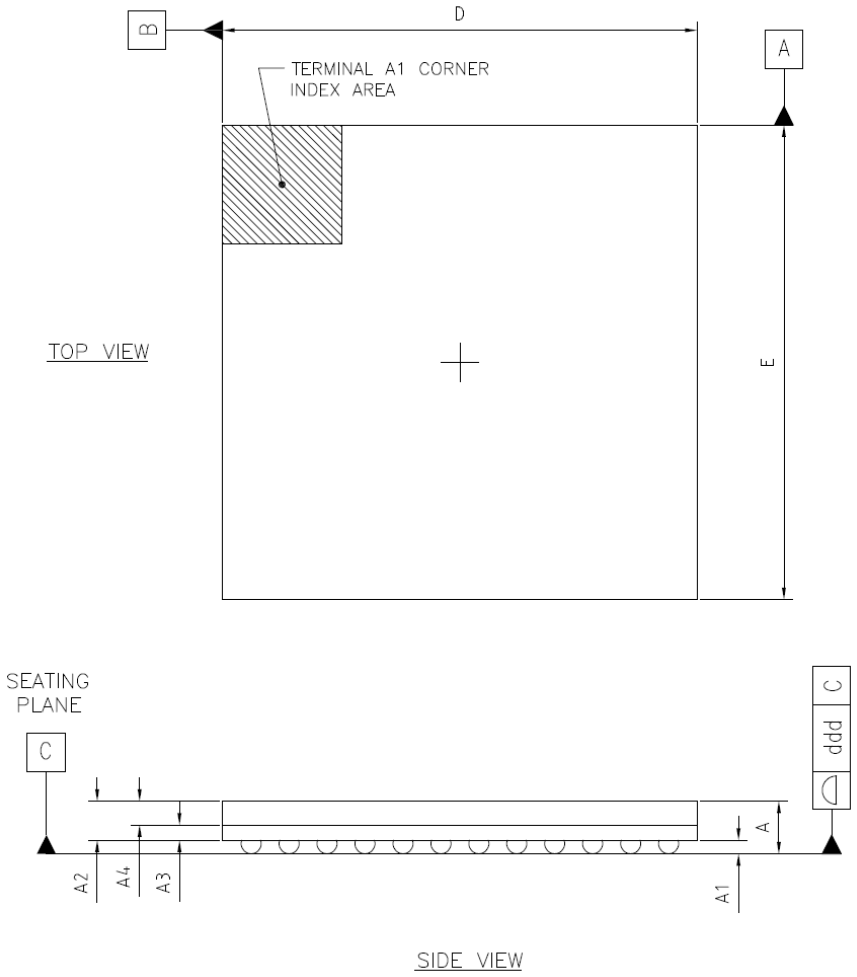

Figure 107: Package top and side view

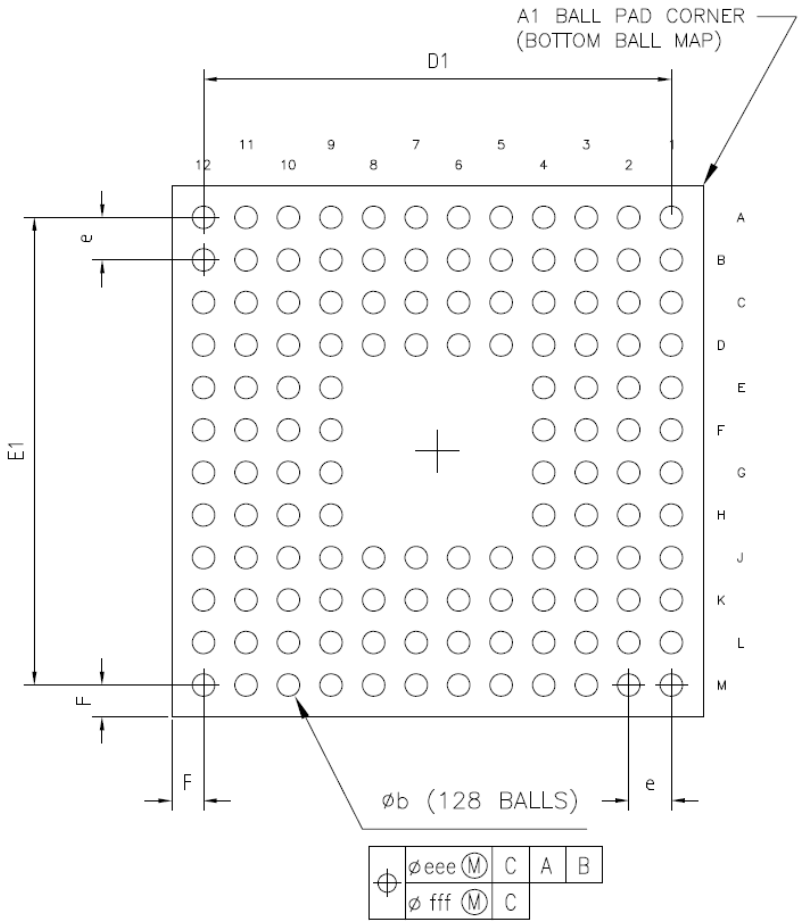

Figure 108: Package bottom view

Table 141: Package dimensions ET1150

Symbol	Min.	Nom.	Max	Unit	Note
PRELIMINARY VALUES					
A	0.94	1.04	1.14	mm	Seating plane to top of component
A1	0.23	0.29		mm	
A2		0.75		mm	
A3		0.25		mm	
A4	0.45	0.50	0.55	mm	
b	0.38	0.43	0.48	mm	
D	9.85	10.00	10.15	mm	
D1		8.80		mm	
e		0.80		mm	
E	9.85	10.00	10.15	mm	
E1		8.80		mm	
F		0.6		mm	
ddd		0.10		mm	Tolerance of ball crowns with respect to datum C
eee		0.15		mm	Ball position tolerance with respect to datums A and B
fff		0.08		mm	Ball position tolerance with respect to each other

The chip label contains chip identification, the production date code, and production lot number:

- Pin A1 marking
- Logo: EtherCAT
- Vendor: BECKHOFF
- ESC: ET1150-000X (X=stepping)
- YYWW: year and week of packaging (YY=year, WW=week)
- LLLLL LL: production lot ID
- AA: additional information (optional)

Figure 109: Chip marking ET1150

11.3.2 Tape and reel information ET1150

The ET1150 is available as tape on reel.

The reel dimensions are:

Table 142: ET1150 reel information

Dimension	Value
Diameter	330 mm

11.3.3 Moisture sensitivity and storage

The ET1150 is shipped in a sealed moisture barrier bag (dry-pack). There is a “caution” label on the dry-pack which contains all necessary information required for handling the devices. Refer to the JEDEC standards J-STD-020 and J-STD-033 for more details (http://www.jedec.org).

The information on the dry-pack takes precedence over information in this chapter.

The moisture sensitivity level of the ET1150 is MSL 3. The maximum shelf-life of the ET1150 packed in a dry-pack is one year after bag seal date. If the ET1150 is stored longer than one year, drying (baking) is required before soldering.

Drying and re-packaging can have negative effects on solderability and conducting surfaces. To minimize issues, the following steps should be taken:

- Visual inspection of the ET1150 devices
- solderability tests with some samples of the ET1150
- final test of the product using the ET1150 with focus on the ET1150 connections

Table 143: Absolute maximum storage conditions of ET1150 devices

Symbol	Parameter	Min	Max	Units
$\vartheta_{Storage}$	Storage temperature	-65	150	°C

11.4 Processing

11.4.1 PCB recommendations

PCB manufacturing technology is complex, please consult your PCB manufacturer and your PCB assembly house for advice. A few recommendations for many use cases are given here.

The pinout of the ET1150 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.

Non-solder mask defined pads (NSMD) with a conductive pad diameter of 300 µm and an actual solder mask opening diameter of 400 µm (after widening) are recommended. Thus, the solder ball covers the whole pad, resulting in a better connection.

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. This results in an equal soldering behavior of the balls.

PCBs without plating (copper) are not recommended, because of brittle solder joints.

11.4.2 Soldering profile

The following soldering profile is used to illustrate minimum and maximum values. For the actual soldering profile many factors have to be taken into consideration, e.g., solder paste characteristics, the PCB, plating, other components, materials, and process type.

Please consult your PCB assembly house for advice.

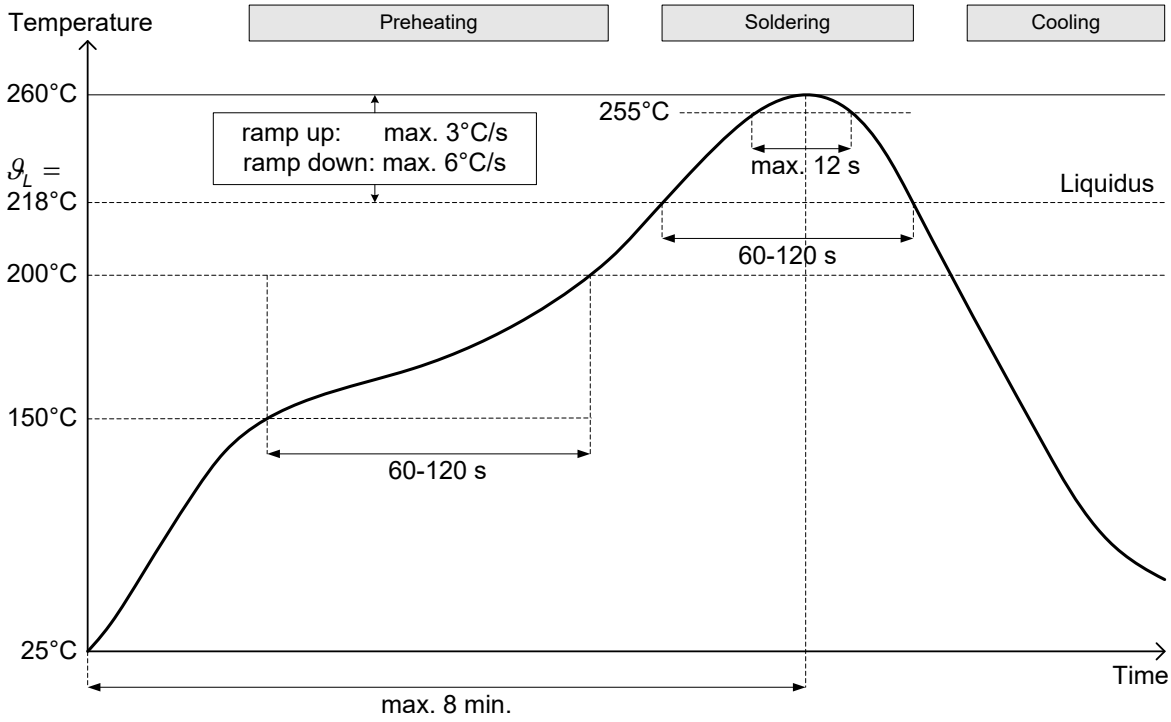

Figure 110: Soldering temperature and time

Table 144: Soldering temperature and time

Symbol	Parameter	Value	Abs. Max.	Units
ϑ_L	Liquidus temperature	218		°C
t_L	Time above ϑ_L (TAL)		120	s
ϑ_P	Peak temperature		260	°C
t_P	Time at ϑ_P		12	s
N_R	Number of reflow cycles		3	

NOTE: Recommended reading: “First Principles of Solder Reflow” by John Vivari.

12 Ordering codes

The ordering codes for the ET1150 devices are composed like this:

ET1150-0000-NNNN

The code part NNNN identifies the size of the packing unit. Do not confuse the ordering codes with the stepping code ET1150-0002. You will always get the latest stepping while the ordering codes are unchanged.

13 Appendix

13.1 Support and service

Beckhoff and our 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.

13.1.1 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.

13.2 Beckhoff headquarters

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

Phone: +49 (0) 5246 963-0

Fax: +49 (0) 5246 963-198

E-mail: info@beckhoff.com

Web: www.beckhoff.com

Beckhoff support

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

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

Hotline: +49 (0) 5246 963-157

Fax: +49 (0) 5246 963-9157

E-mail: support@beckhoff.com

Beckhoff service

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

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

Hotline: +49 (0) 5246 963-460

Fax: +49 (0) 5246 963-479

E-mail: service@beckhoff.com