all items

EEPROM

- Overview
- Meaning of the ESC Configuration Area
- How to read/write EEPROM binary content
- Access
- Programming
- References

top

Overview

EEPROM / SII

EEPROM and SII are used synonymously. SII stands for “Slave Information Interface” and provides all the necessary information of the slave online, i.e. the master can read it from the slave and would not need an ESI file to operate the slave. “SII” does only mean the data structure, but not how this data is stored. The most typical implementation of the SII is that the data is stored on an EEPROM memory device which is connected to the ESC via an I²C interface. Such a NVRAM (typically a serial EEPROM with I²C interface) to store the device configuration and device description is mandatory for any slave. EEPROM sizes from 1 Kbit up to 4 Mbit are supported, depending on the ESC.

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

ESI (Device Description)

ESI (EtherCAT Slave Information) describes detailed an EtherCAT slave device for an offline configuration tool with all its features and settings necessary to know for proper operation. It is stored as an .xml file. For more information please download the ETG.2000 EtherCAT Slave Information (ESI) Specification.

ESI vs. SII

ESI and SII both supply the same information about a slave. So why provide the information twice?

There are some master implementations which run e.g. on embedded systems. These masters often do not use ESI files e.g. due to a missing file system. For them it is necessary to be able to read all the information from the slave directly.

Considering large networks masters with possibility to store and access ESIs are much faster when reading the device information from hard disk instead from each slave online. Especial for this masters ESI files are useful. When configuring an EtherCAT network offline ESI files are necessary as well, to get the information about the slave devices without having them connected to the configuration tool.

top

Meaning of the ESC Configuration Area

The ESC Configuration Area defines the hardware settings of the PDI (Process Data Interface; the hardware interface providing data exchange between the ESC and the local application which is typically implemented in a µC). ESCs typically support several PDIs with multiple hardware options. For each particular slave implementation, a specific PDI and hardware options shall be configured on ESC side according to the µC properties. The settings corresponding to the ESC Configuration Area are saved in the first words of SII EEPROM content and loaded into specific ESC registers after a power-on or reset (Figure 1).

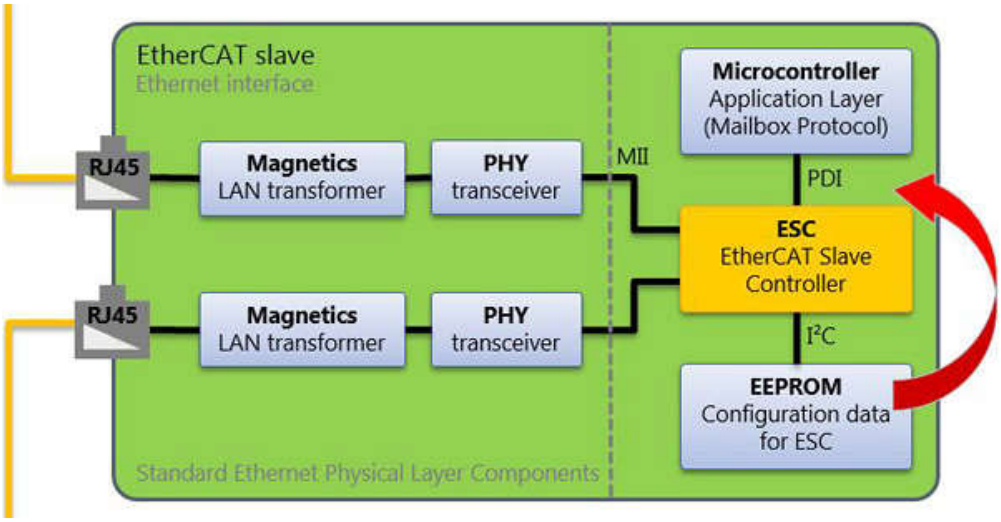

Figure 1: PDI configured with ESC Configuration Area in EEPROM

EEPROM Content

The EEPROM Content consists of the Info Structure and the Categories area, whereby the Info Structure contains the EEPROM Configuration Data and the Categories area describe further features of the slave.

EEPROM Configuration Data

The following explains how the EEPROM configuration data is assembled. This is done on the example of the ET1100 EtherCAT Slave Controller (Beckhoff). In Table 1 the structure of the first 7 words of an EtherCAT Slave Controller EEPROM is given as an example on which all further steps will be based on.

Table 1: SII Area - first 7 words

Parameter	Address	Value/Description
PDI Control	0x0000	Init. value for PDI control register (0x0140-0x0141)
PDI Configuration	0x0001	Init. value for PDI configuration register (0x0150-0x0151)
SyncImpulseLen	0x0002	Sync impulse in multiplies of 10ns
PDI Configuration2	0x0003	Init. value for PDI configuration register (0x0152-0x0153)
Configured Station Alias	0x0004	Alias address
Reserved	0x0005	Set to zero
Reserved	0x0006	Set to zero
Checksum	0x0007	Low byte contains division remainder of word 0 to word 6 as unsigned number divided by the polynomial x^8+x^2+x+1 (initial value 0xFF)

ATTENTION: The PDI Configuration Data is ESC-specific. Check with the individual respective data sheet of the vendor. Example for ET1100 and IP Cores:

				ESC20 [7:1]	ET1100 [7:4]	ET1200 [7:2]	IP Core [7:4] V2.2.0/ V2.02a
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**			
0	Device emulation (control of AL status): 0: AL status register has to be set by PDI 1: AL status register will be set to value written to AL control register	r/-	r/-	IP Core: 1 with Digital I/O PDI, PDI_EMULATION pin with µC/On-chip bus Others: 0, later EEPROM word 0			
1	Enhanced Link detection all ports: 0: disabled (if bits [7:4]=0)	r/-	r/-	1, later EEPROM word 0			

1. Spreadsheet to generate EEPROM Configuration Data

Some Vendors provide a convenient tool for generating EEPROM configuration data. In the case of ET1100, Beckhoff offers a spreadsheet where setting a couple of options in dropdown menus returns the first 7 words of EEPROM configuration data in Byte, Word and XML format.
In the first part PDI selection is performed together with some general configuration settings (Device emulation, Enhanced link detection, DC unit). Depending on the PDI selection, automatic recommendations and restricted options for general configuration settings are shown in the spreadsheet. Further detailed PDI interface settings become available upon the selection of the PDI:

- Digital I/O
- SPI
- µController a-/sync. 8/16bit

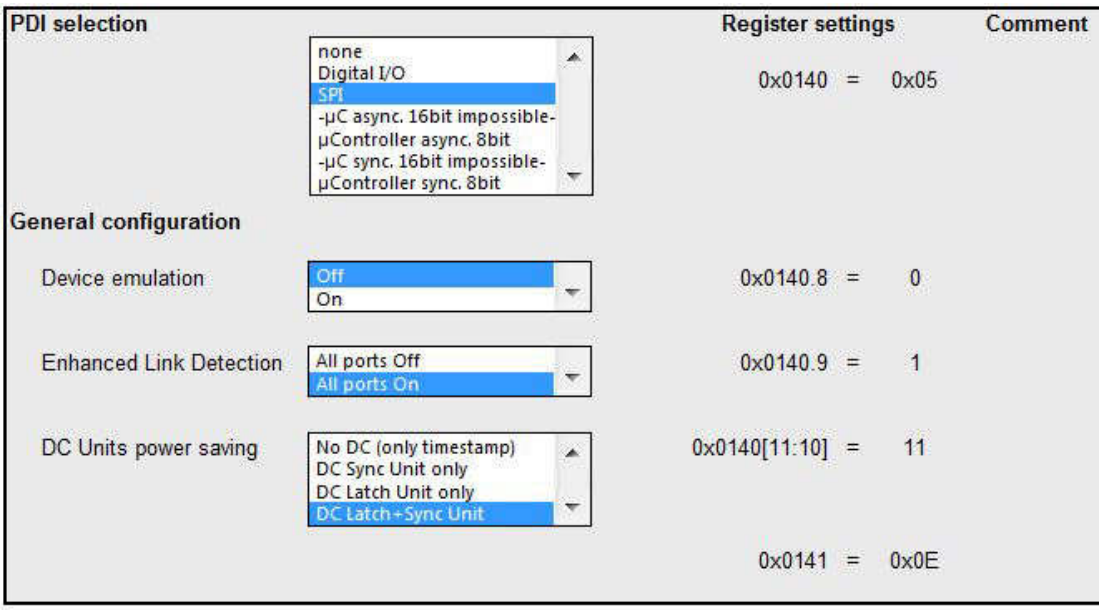

Figure 2: ET1100 ESI EEPROM Configuration - PDI selection

For this example SPI slave with device emulation == OFF, enhanced link detection on all ports and a DC unit with Latch and Sync is chosen. The first word block of EEPROM is as follows in Table 2:

Table 2: PDI Control example value

Parameter	Address	Value/Description
PDI Control	0x0000	0x0E05

Since the chosen PDI in this example is SPI (Serial Peripheral Interface) it needs to be further specified. The options which the tool offers are shown in Figure 3:

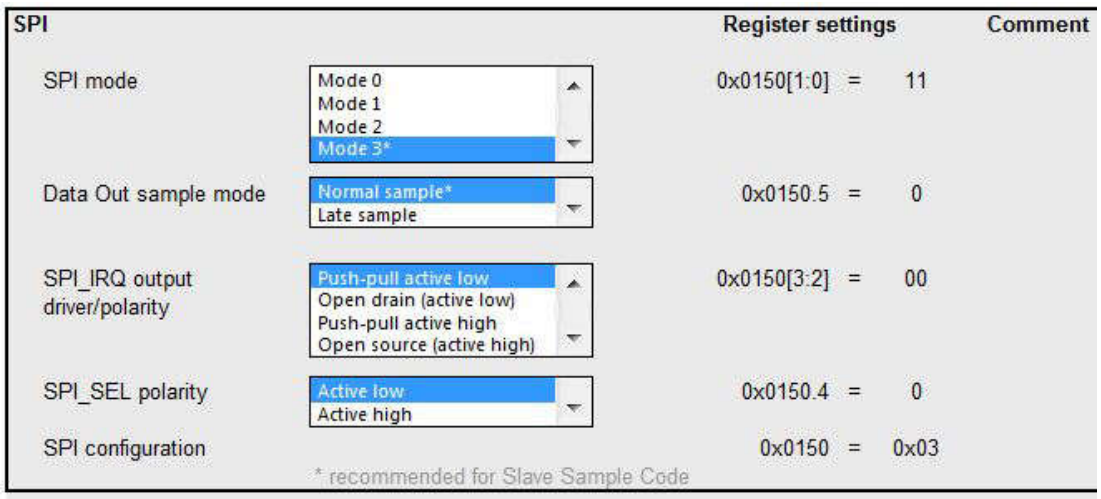

Figure 3: ET1100 ESI EEPROM Configuration - SPI

Thus the low byte of the second word of the EEPROM is determined. The high word defines the SYNC/LATCH configuration as shown in Figure 4:

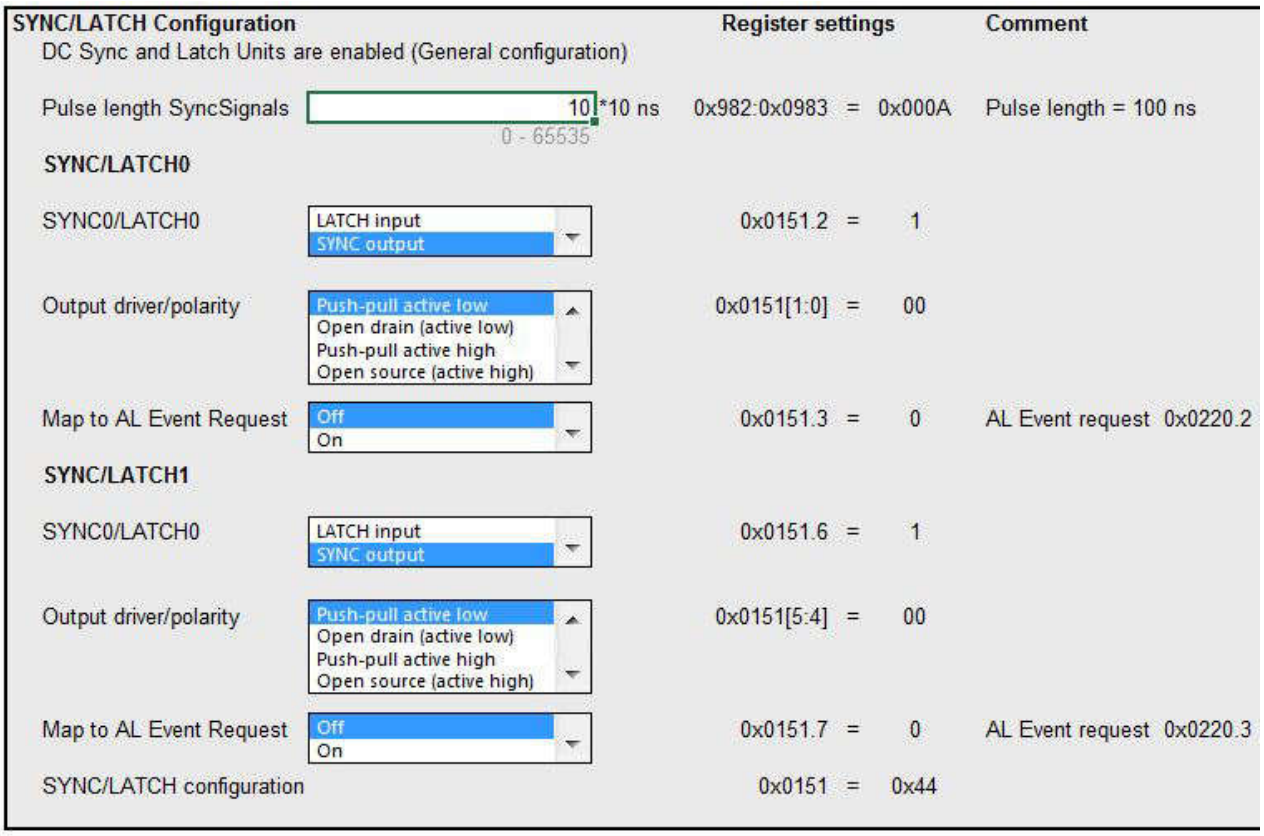

Figure 4: ET1100 ESI EEPROM Configuration - SYNC/LATCH

Together they form the values of the second address in EEPROM:

Table 3: PDI Configuration

Parameter	Address	Value/Description
PDI Configuration	0x0001	0x4403

In this same area of the spreadsheet also the value for the third word of the EEPROM, namely the pulse length of the Sync signals, is set:

Table 4: Sync impulse length

Parameter	Address	Value/Description
SyncImpulseLen	0x0002	0x000A

The remaining values for the fourth and fifth E2PROM address are in this example zero. A µC was not chosen therefore no further extended PDI configurations for the address 0x0003 are not needed. In this configuration the Configured Station Alias address is zero as default:

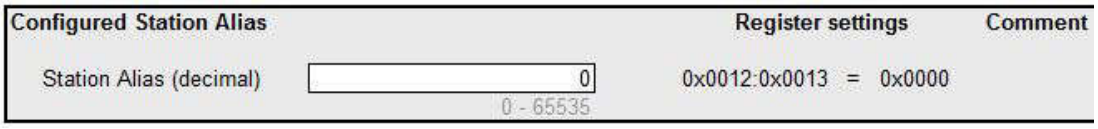

Figure 5: Configured Station Alias

Table 5: Extended PDI configuration and Configured Station Alias address values

Parameter	Address	Value/Description
PDI Configuration2	0x0003	0x0000
Configured Station Alias	0x0004	0x0000

Addresses 0x0005 and 0x0006 are specified as Reserved. The Checksum value of the last E2PROM address is calculated automatically by the tool.

Table 6: eserved addresses and Checksum

Parameter	Address	Value/Description

Parameter	Address	Value/Description
Reserved	0x0005	0x0000
Reserved	0x0006	0x0000
Checksum	0x0007	0x0052

All determined values are updated by the tool automatically and returned in the summary table at the end of the spreadsheet as shown in Figure 6:

ESI EEPROM settings

Byte view (low byte to high byte)
05 0E 03 44 0A 00 00 00 00 00 00 00 00 00 52 00

Word view (low word to high word)
0E05 4403 000A 0000 0000 0000 0000 0052

XML EEPROM ConfigData
050E03440A00

Word	Value	Register description	Reg. Address
0	0x0E05	PDI Control	0x0140:0x0141
1	0x4403	PDI Configuration	0x0150:0x0151
2	0x000A	Pulse Length SyncSignals	0x0982:0x0983
3	0x0000	Extended PDI Configuration	0x0152:0x0153
4	0x0000	Configured Station Alias	0x0012:0x0013
5	0x0000	reserved	-
6	0x0000	reserved	-
7	0x0052	CRC	-

Figure 6: EEPROM Configuration data

2. Determining the EEPROM Configuration manually by use of the ESC Hardware data sheet

Another way to determine the first 7 words of the EEPROM, especially if there are no vendor provided tools, is the manual approach, by checking the documentation of the ESC itself. In the case of ET1100 its documentation is available on the Beckhoff website. The needed data of the first 7 words, as presented in Table 1, can be obtained by checking the ET1100 EtherCAT Registers (Section II) document. Referring to the registers, mentioned in Table 1 the needed values can be traced in the tables of the document. For example the value of PDI control area – register 0x0141 - can be picked out of Table 28, as shown in Figure 7:

2.26 ESC Configuration (0x0141)

Table 28: Register ESC Configuration (0x0141)

		ESC20 [7:1]	ET1100 [7:4]	ET1200 [7:2]	IP Core [7:4] V2.2.0/ V2.02a
Bit	**Description**	**ECAT**	**PDI**	**Reset Value**	
0	Device emulation (control of AL status): 0: AL status register has to be set by PDI 1: AL status register will be set to value written to AL control register	r/-	r/-	IP Core: 1 with Digital I/O PDI, PDI_EMULATION pin with µC/On-chip bus Others: 0, later EEPROM word 0	
1	Enhanced Link detection all ports: 0: disabled (if bits [7:4]=0) 1: enabled at all ports (overrides bits [7:4])	r/-	r/-	1, later EEPROM word 0	
2	Distributed Clocks SYNC Out Unit: 0: disabled (power saving) 1: enabled	r/-	r/-	IP Core: Depends on configuration Others: 0, later EEPROM word 0	
3	Distributed Clocks Latch In Unit: 0: disabled (power saving) 1: enabled	r/-	r/-		
4	Enhanced Link port 0: 0: disabled (if bit 1=0) 1: enabled	r/-	r/-	1, later EEPROM word 0	
5	Enhanced Link port 1: 0: disabled (if bit 1=0) 1: enabled	r/-	r/-		
6	Enhanced Link port 2: 0: disabled (if bit 1=0) 1: enabled	r/-	r/-		
7	Enhanced Link port 3: 0: disabled (if bit 1=0) 1: enabled	r/-	r/-		

NOTE: EEPROM values of bits 1, 4, 5, 6, and 7 are only transferred into this register at first EEPROM load after power-on or reset.

Figure 7: Register 0x0141

In this example the Device Emulation is set to zero, Enhanced Link Detection on all ports == 1, Distributed Clocks SNYC Out Unit and Distributed Clocks Latch In Unit are both enabled, value == 1. Enabling or disabling of the Enhanced Link detection on separate ports is in case of ET1100 not possible. The end EEPROM value for register 0x0140 of the ESC thus equals 0000 1110 in BIN and 0x0E in HEX.

3. ESI (EtherCAT Slave Information)

The EEPROM configuration data block may be replaced in the ESI element “Eeprom:ConfigData” as shown on Figure 8. Using a configuration tool, the ESI file can be used to generate and write the EEPROM binary to the EEPROM of the EtherCAT slave.

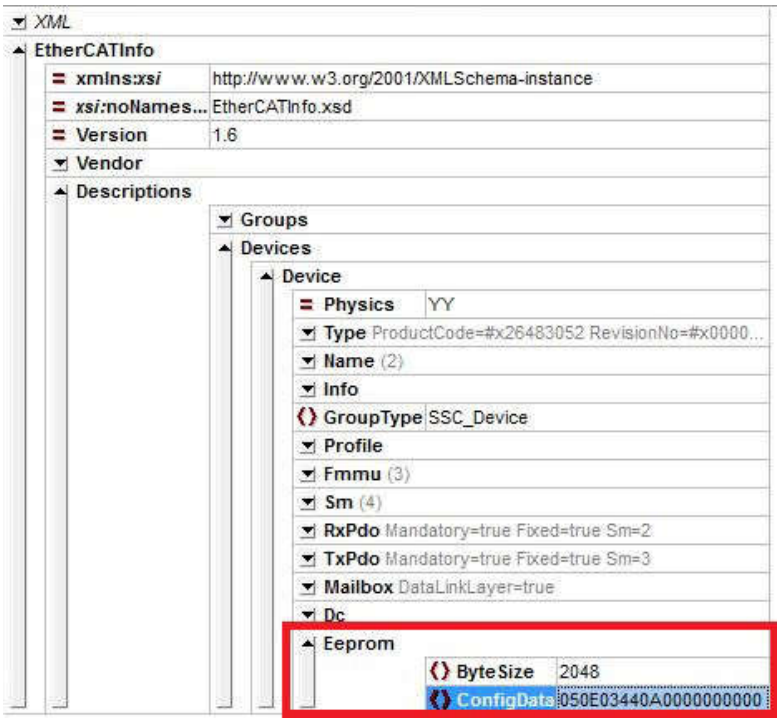

Figure 8: EEPROM element in ESI

Categories

Changing the order can lead to errors while reading the EEPROM if, for example, strings are already referenced before.

top

How to read/write EEPROM binary content

The SII EEPROM content may be verified, or created/overwritten, via the SII EEPROM Logical Interface by using the CTT (Conformance Test Tool). As an alternative, a Master Configuration Tool (e.g. Beckhoff TwinCAT System Manager) can support a utility to read/write the SII EEPROM content. As another alternative to write the EEPROM the SSC (Slave Stack Code) Tool from Beckhoff may be used.

EtherCAT Conformance Test Tool (CTT)

>> see also help file of CTT: F1 > search for “EEPROM Tabbed Documents”

1. Copy the ESI file to the CTT ESI folder (see. "ESI Cache" - window)
2. Scan the connected slaves
3. Open the Slave EEPROM tab (Figure 9):

- It is possible to upload the current EEPROM content (A)
- It is possible to visualize the uploaded EEPROM content in hex format (B)
- It is possible generate the EEPROM content by drag-and-drop of the corresponding ESI description (C1) and then overwrite the EEPROM (C2)
- It is possible to manually edit the hex content (D1) and then download the modified content into the EEPROM (D2)

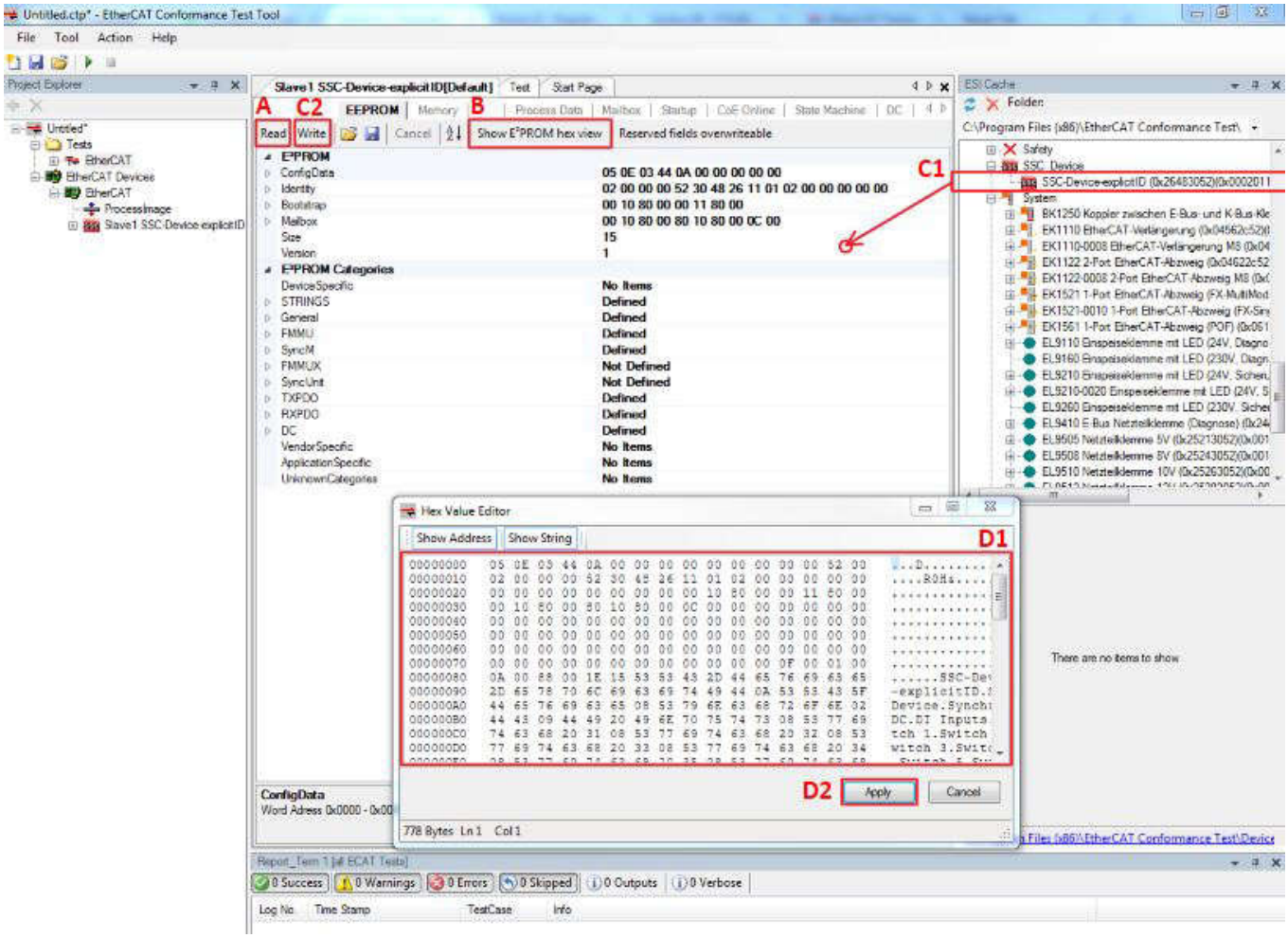

Figure 9: EEPROM Management via CTT

EtherCAT Configuration Tool (TwinCAT)

1. Copy the ESI file to the TwinCAT ESI folder ("C:\TwinCAT\3.1\Config\Io\EtherCAT\" for TwinCAT 3, "C:\TwinCAT\Io\EtherCAT\" for TwinCAT 2).
2. Create a new TwinCAT Project and scan the connected slaves, or add them to the configuration manually (see How To: Set-Up a Network Configuration).
3. Open the EtherCAT slave Advanced Settings, then go to ESC Access → E2PROM→ Hex Editor (Figure 10):

- It is possible to read out the current EEPROM content (A)
- It is possible to save the uploaded EEPROM content into an external text file (B)
- It is possible to edit the EEPROM content manually (C1), or to import it from an external text file (C2), or generate the EEPROM content from a corresponding ESI description (C3)
- It is possible to download the modified content into the EEPROM (D).

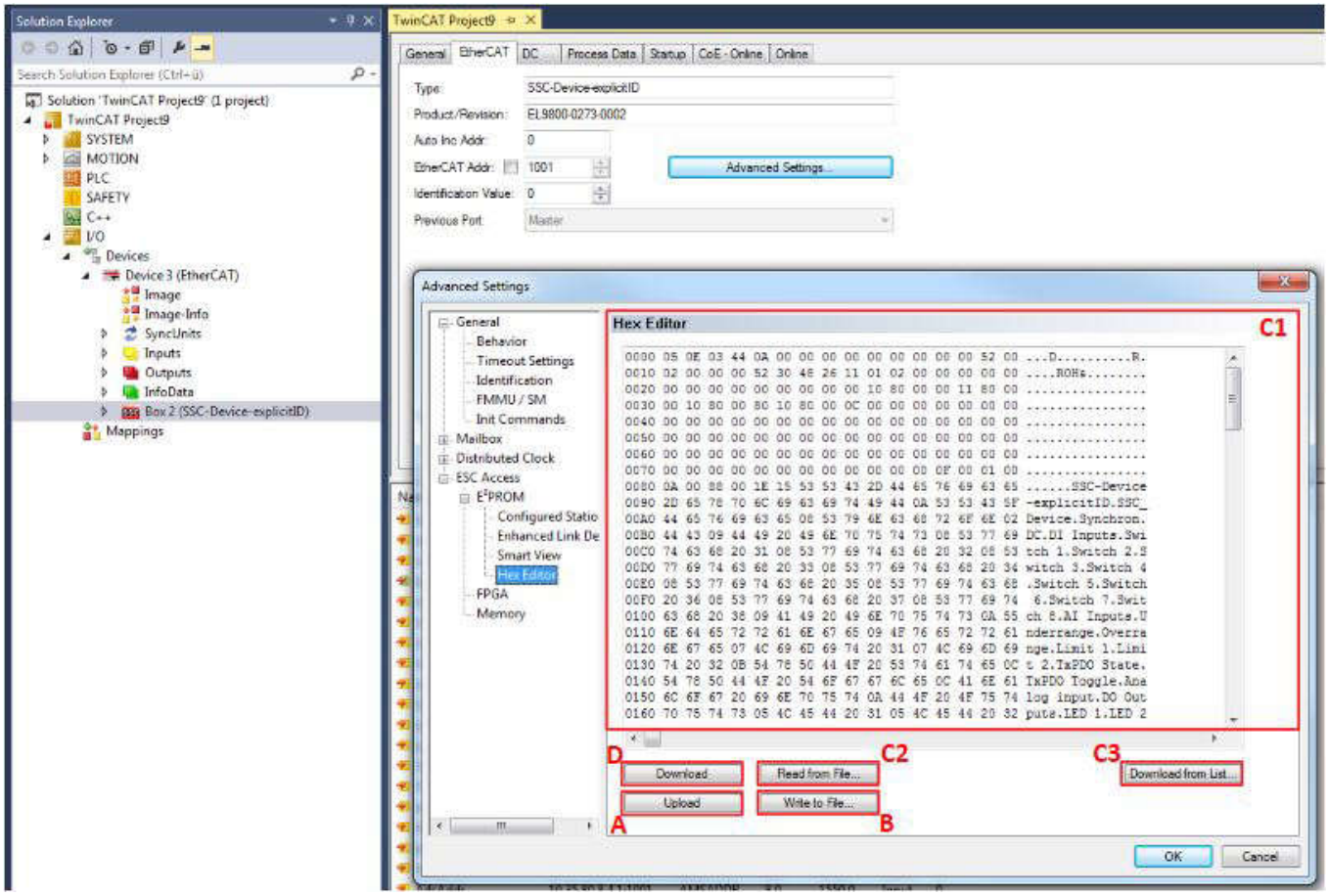

Figure 10: EEPROM Management via TwinCAT

Slave Stack Code Tool (SSC Tool)

For an EEPROM update with a help of SSC internal tool "EEPROM Programmer", please see the Application Note SlaveStackCode (SSC), Chapter 14.

top

Access

The EEPROM access can either be assigned to the PDI or the EtherCAT master.

References

ETG.1020 EtherCAT Protocol Enhancements, clause EEPROM Access

ETG.2000 EtherCAT Slave Information (ESI) Specification, keyword "AssignToPdi"

top

Programming

Write EEPROM during productions

Possibilities to write EEPROM:

- Application Controller on board (connected to I²C bus)
  The PCB shall be designed in a way that allows the application controller access to the EEPROM via the I²C interface. If the application controller accesses the EEPROM it has to ensure that the ESC does not access the EEPROM at the same time by holding the ESC in RESET via the RESET pin. After finishing the EEPROM

programming the application controller shall release the ESC from RESET. It is useful to have a flag within the local application which is set when EEPROM was written the first time. So the application controller will not write EEPROM on every single power-up.

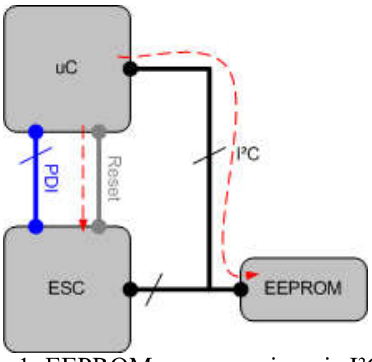

Figure 1: EEPROM programming via I²C bus

- Using an EEPROM programmer
  The PCB shall be designed in a way that leaves the required pads to connect an EEPROM programmer to the board to program it. When writing to EEPROM ESC may not be powered or has to be in reset to grant that ESC will not try to access EEPROM via I²C bus.

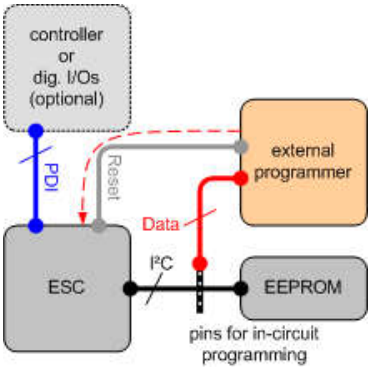

Figure 2: EEPROM programming with external programmer

- "Mini"-Master programs EEPROM
  To avoid the need of a full master license a small master application can be written. This application only sends the data for the EEPROM programming. To find out which data has to be sent a scan of the network traffic during EEPROM content download using a configuration tool might help.

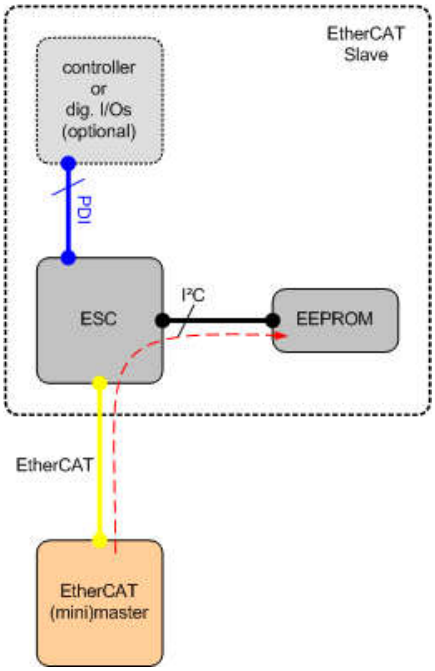

Figure 3: EEPROM programming with a (mini-)master tool

- TwinCAT Master (EtherCAT Configuration Tool)

  Mark the EtherCAT master device in the tree view on the left side. Mark several devices in the device area in the lower area on the right side using the SHIFT key. Make a right click on the marked devices and choose 'EEPROM

Update...'. Choose the corresponding device description from the dialog and press OK two times to download the EEPROM file

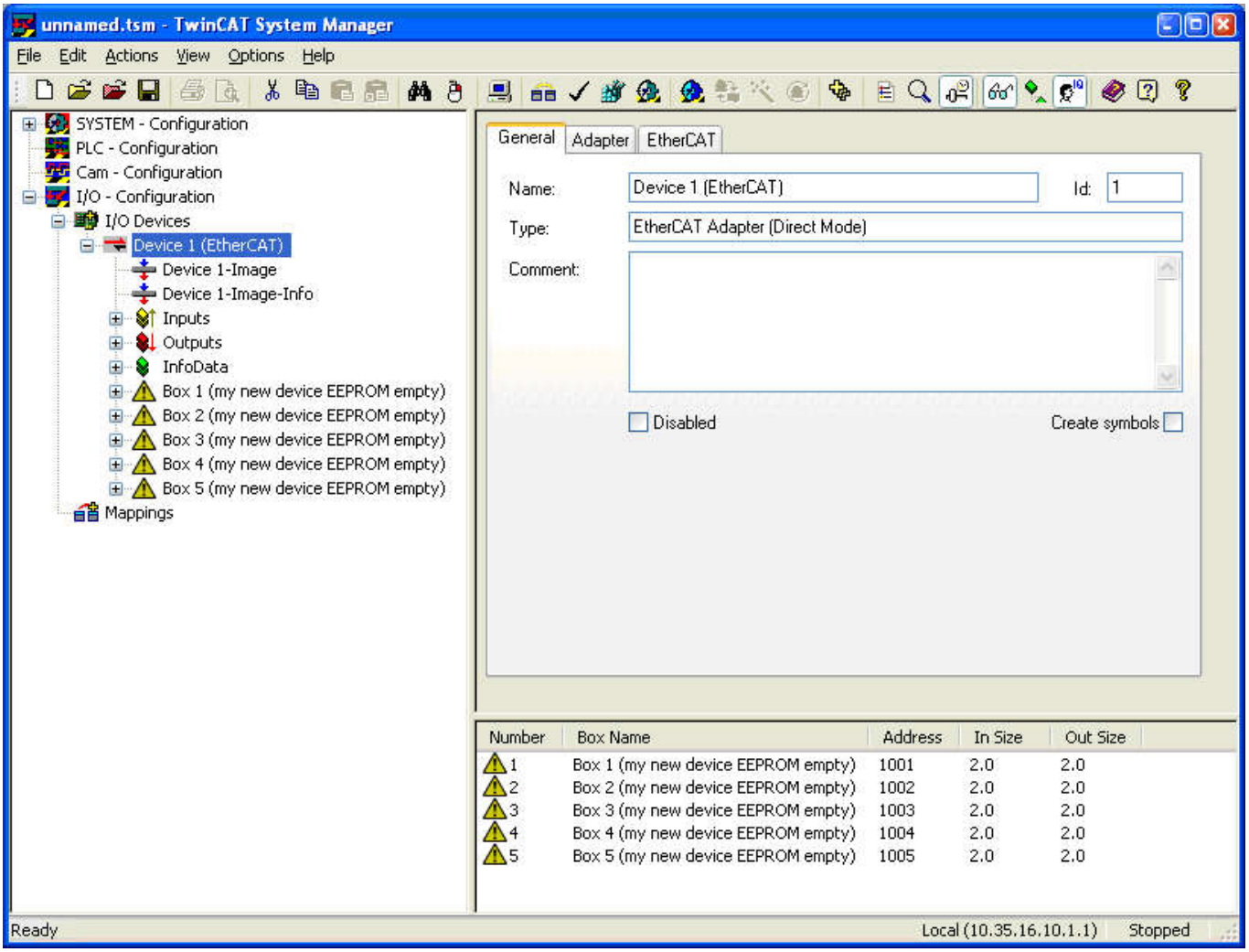

Figure 4: Select EtherCAT master

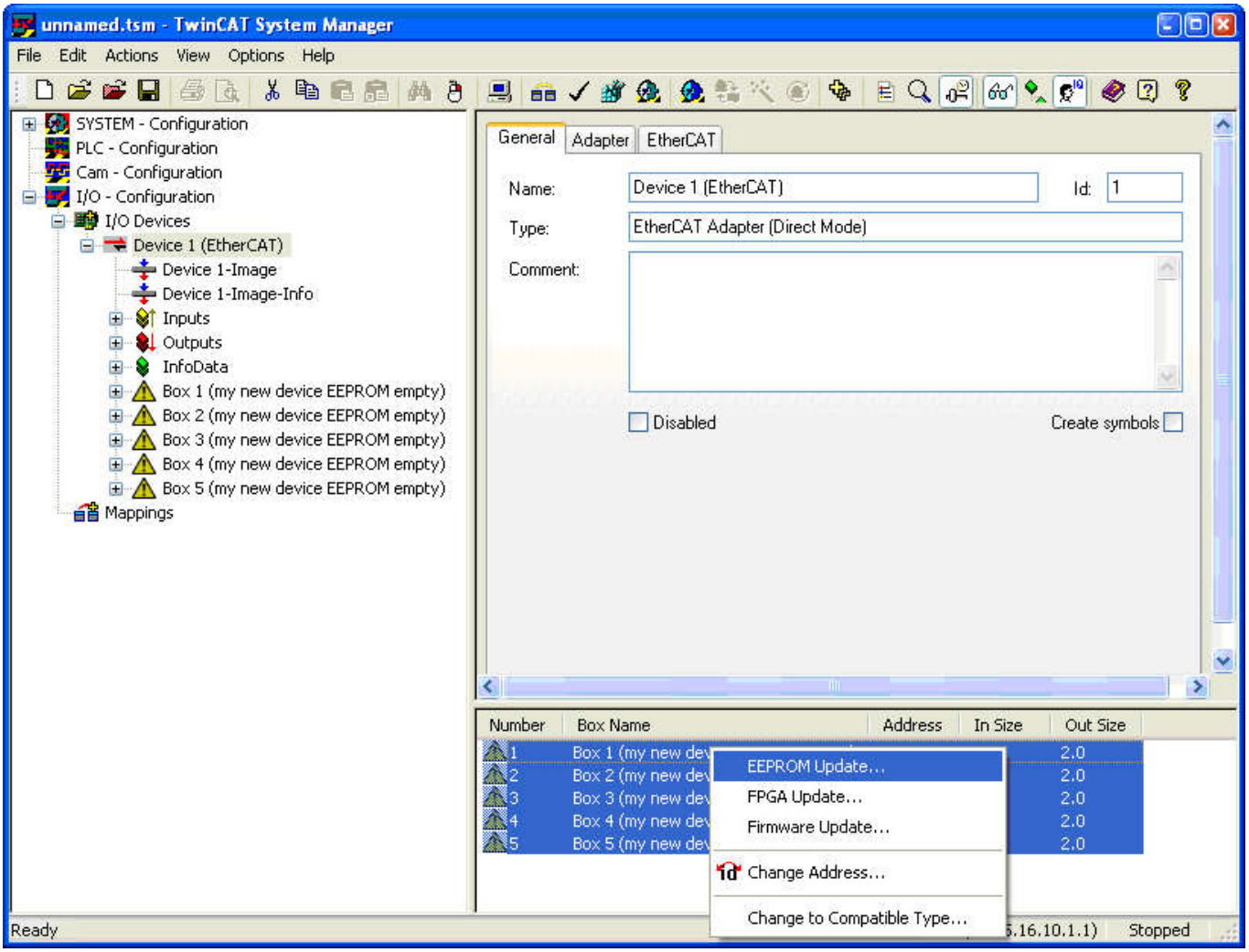

Figure 5: Select the devices and choose 'EEPROM Update...'

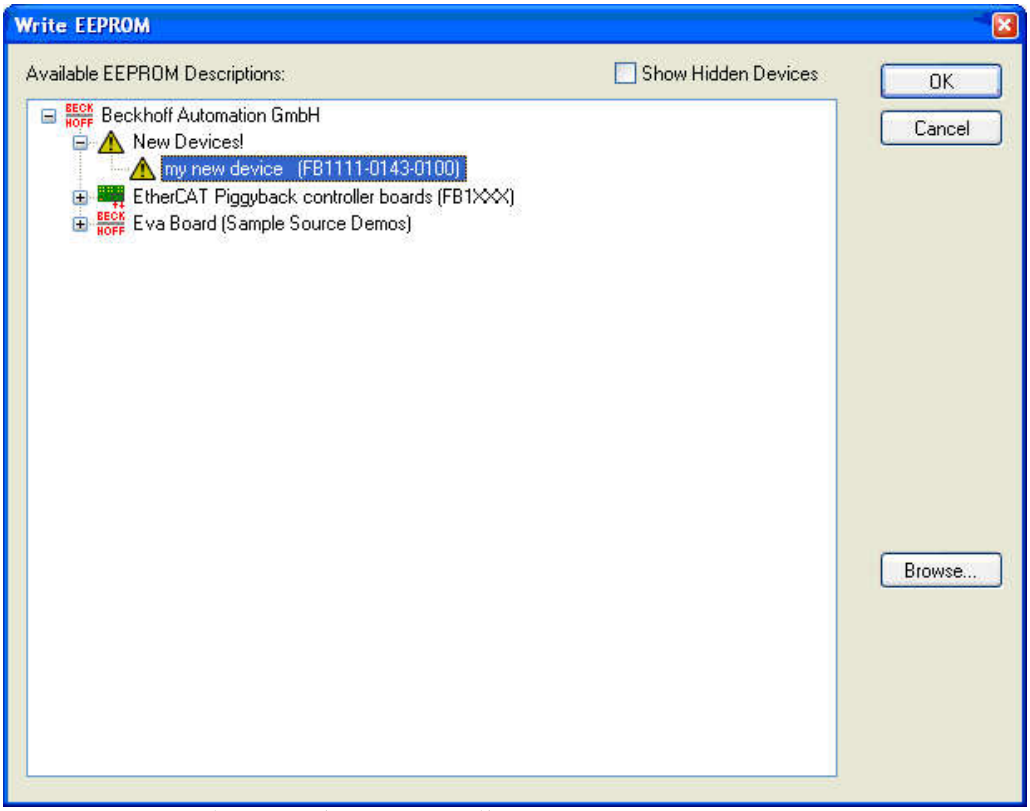

Figure 6: select your ESI file to generate EEPROM Content

Subsequent Access / change of EEPROM

In case the EEPROM content may has to be changed after it was loaded correctly in the first place it is possible to access the EEPROM from PDI side via register communication. However, this requires that the PDI configuration was loaded

sucessfully and the access from host controller side to the ESC is possible.

- Application Controller on board (not connected to I²C bus) - EEPROM access via ESC registers
  The application controller can access the EEPROM via the ESC when the master grants this access during state transition from Init to PreOp.

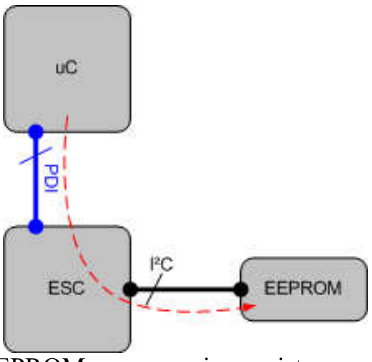

Figure 7: EEPROM programming register communication

top

References

How To: Set-Up a Network Configuration

ETG.1000.6: Application Layer protocol specification, section 5.4 (IEC61158-6-12, section 5.4)

ETG.2010 EtherCAT Slave Information Interface (SII)