

## **EPSG Draft Standard 301**

# **Ethernet POWERLINK**

## **Communication Profile Specification**

**Version 1.3.0**

**© EPSG**

**(Ethernet POWERLINK Standardisation Group)**

**2016**

---

EPSG (Ethernet POWERLINK Standardisation Group)

POWERLINK-Office of the EPSG  
Bonsaiweg 6  
D-15370 Fredersdorf  
Germany

Phone +49 33439 539 270  
Fax +49 33439 539 272  
[info@ethernet-powerlink.org](mailto:info@ethernet-powerlink.org)  
[www.ethernet-powerlink.org](http://www.ethernet-powerlink.org)

## Pre. 1 Disclaimer

Use of this EPSG Standard is wholly voluntary. The EPSG disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other EPSG Standard document.

The EPSG does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. EPSG Standards documents are supplied "AS IS".

The existence of an EPSG Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the EPSG Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Users are cautioned to check to determine that they have the latest edition of any EPSG Standard.

In publishing and making this document available, the EPSG is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the EPSG undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other EPSG Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

**Interpretations:** Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of the EPSG, the group will initiate action to prepare appropriate responses. Since EPSG Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, the EPSG and its members are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of EPSG Standards are welcome from any interested party, regardless of membership affiliation with the EPSG. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

POWERLINK-Office of the EPSG, Bonsaiweg 6, D-15370 Fredersdorf, Germany

### Pre 1.1 Patent notice

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The EPSG shall not be responsible for identifying patents for which a license may be required by an EPSG standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

## Pre. 2 History

| Vers. | State | Date       | Author / Filename  |  | Description   |
|-------|-------|------------|--|--|---|
| 1.0.0 | DS    | 2006-04-18 | Knopke et al.   Lenze et al.<br>EPSG DS 301 V-1-0-0.doc                                  |  | DS version created from working documents   |
| 1.0.1 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-0-1.doc                               |  |   |
| 1.0.2 | WDP   | 2008-06-13 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-0-2.doc                               |  | Changes from TWG meetings in 2/07 and 2/08.   |
| 1.0.3 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-0-3.doc                               |  |   |
| 1.0.4 | WDP   | 2008-09-05 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-0-4.doc                               |  | Objects used in EPSG DS 302-A are marked as reserved.<br>Feedback from certification meeting                          |
| 1.0.4 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG DSP 301 V-1-0-4.doc                               |  |   |
| 1.0.5 | DSP   | 2008-10-17 | Kirchmayer et al.   B&R et al.<br>EPSG DSP 301 V-1-0-5.doc                               |  | Feedback from TWG conference call in 10/08  |
| 1.1.0 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG DS 301 V-1-1-0.doc                                |  |   |
| 1.1.1 | WDP   | 2010-01-13 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-1.doc                               |  | WDP version created from DS 1.1.0<br>Changes from TWG meeting in 6/09<br>Feedback from ITEI (China)                   |
| 1.1.2 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-2.doc                               |  |   |
| 1.1.3 | WDP   | 2011-08-12 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-3.doc                               |  | Changes from TWG meeting in 9/10<br>Refer to IEC instead of IAONA standards   |
| 1.1.4 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-4.doc                               |  |   |
| 1.1.5 | WDP   | 2013-02-27 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-5.docx                              |  | Changes from TWG meeting in 9/12<br>Contribution list removed<br>PDO mapping attribute of 1F8Ch is Opt                |
| 1.1.6 |       |            | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-6.docx                              |  |   |
| 1.1.7 | WDP   | 2013-06-11 | Kirchmayer et al.   B&R et al.<br>EPSG WDP 301 V-1-1-7.docx<br>EPSG WDP 301 V-1-1-7.docx |  | Changes from TWG telco in 5/13,<br>Clarify Tab. 53, Replace Sender/Receiver by<br>Client/Server in Fig. 45 to Fig. 54 |
| 1.1.7 |       |            | Kirchmayer   B&R<br>EPSG DSP 301 V-1-1-7.docx  |  |   |
| 1.2.0 | DS    | 2013-12-18 | Kirchmayer   B&R<br>EPSG DS 301 V-1-2-0.docx   |  | Changes from TWG meeting in 12/13<br>Update of references to other standards<br>Status change to DS                   |
| 1.2.0 |       |            | Kirchmayer   B&R<br>EPSG DS 301 V-1-2-0.docx   |  |   |
| 1.2.1 | WDP   | 2014-09-25 | Kirchmayer   B&R<br>EPSG WDP 301 V-1-2-1.docx  |  | Changes from TWG meeting 06/14  |
| 1.2.2 |       |            | Kirchmayer   B&R<br>EPSG WDP 301 V-1-2-2.docx  |  |   |
| 1.2.3 | DSP   | 2015-12-16 | Kirchmayer   B&R<br>EPSG DSP 301 V-1-2-3.docx  |  | Changes from TWG meeting 12/15<br>Status change to DSP  |
| 1.3.0 |       |            | Kirchmayer   B&R<br>EPSG DSP 301 V-1-3-0.docx  |  |   |

Editorial Remarks:

notes, examples, hints, etc. are marked by *cursive*

## Pre. 3 Change Record

The changes based on the last valid Communication Profile Specification (EPSG DS 301 V1.2.0) are tracked in a separate change record. Hereby the change record provides a detailed history from the last draft standard to the current one.

## Pre. 4 Content

|             |  |    |
|-------------|--|----|
| Pre. 1      | Disclaimer                                       | 3  |
| Pre 1.1     | Patent notice                                    | 3  |
| Pre. 2      | History  | 4  |
| Pre. 3      | Change Record                                    | 5  |
| Pre. 4      | Content  | 6  |
| Pre. 5      | Tables   | 15 |
| Pre. 6      | Figures  | 18 |
| Pre. 7      | Definitions and Abbreviations                    | 20 |
| Pre 7.1     | Definitions                                      | 20 |
| Pre 7.2     | Abbreviations                                    | 22 |
| Pre. 8      | References                                       | 24 |
| 1           | Introduction                                     | 25 |
| 1.1         | Slot Communication Network Management            | 25 |
| 1.2         | POWERLINK key features                           | 25 |
| 1.3         | Integration                                      | 26 |
| 1.4         | Modular Machines                                 | 27 |
| 2           | Modelling  | 28 |
| 2.1         | Reference Model                                  | 28 |
| 2.1.1       | Application Layer                                | 28 |
| 2.1.1.1     | Service Primitives                               | 28 |
| 2.1.1.2     | Application Layer Service Types                  | 29 |
| 2.2         | Device Model                                     | 29 |
| 2.2.1       | General  | 29 |
| 2.2.2       | The Object Dictionary                            | 30 |
| 2.2.2.1     | Index and Sub-Index Usage                        | 32 |
| 2.3         | Communication Model                              | 32 |
| 2.3.1       | Master/Slave relationship                        | 32 |
| 2.3.2       | Client/Server relationship                       | 33 |
| 2.3.3       | Producer/Consumer relationship - Push/Pull model | 33 |
| 2.3.4       | Superimposing of Communication Relationships     | 34 |
| 3           | Physical Layer                                   | 35 |
| 3.1         | Topology   | 35 |
| 3.1.1       | Hubs   | 35 |
| 3.1.2       | Switches   | 35 |
| 3.2         | Network Guidelines                               | 35 |
| 3.2.1       | Jitter   | 36 |
| 3.3         | Ports and Connectors                             | 36 |
| 3.3.1       | RJ-45  | 36 |
| 3.3.2       | M12  | 36 |
| 3.3.3       | Crossover Pin Assignment                         | 37 |
| 3.3.3.1     | RJ45 to RJ45                                     | 37 |
| 3.3.3.2     | M12 to M12                                       | 37 |
| 3.3.3.3     | M12 to RJ45                                      | 38 |
| 3.4         | Cables (recommendation)                          | 38 |
| 4           | Data Link Layer                                  | 39 |
| 4.1         | Modes of Operation                               | 39 |
| 4.2         | POWERLINK Mode                                   | 39 |
| 4.2.1       | Introduction                                     | 39 |
| 4.2.2       | POWERLINK Nodes                                  | 39 |
| 4.2.2.1     | POWERLINK Managing Node                          | 39 |
| 4.2.2.2     | POWERLINK Controlled Node                        | 40 |
| 4.2.2.2.1   | Isochronous CN                                   | 40 |
| 4.2.2.2.2   | Async-only CN                                    | 40 |
| 4.2.3       | Services   | 40 |
| 4.2.4       | POWERLINK Cycle                                  | 40 |
| 4.2.4.1     | Isochronous POWERLINK Cycle                      | 41 |
| 4.2.4.1.1   | Isochronous phase                                | 41 |
| 4.2.4.1.1.1 | Multiplexed Timeslots                            | 42 |
| 4.2.4.1.2   | Asynchronous phase                               | 43 |
| 4.2.4.1.2.1 | Asynchronous Scheduling                          | 44 |
| 4.2.4.1.2.2 | Distribution of the Asynchronous phase           | 44 |
| 4.2.4.1.2.3 | Asynchronous Transmit Priorities                 | 44 |
| 4.2.4.1.3   | Idle Phase                                       | 47 |
| 4.2.4.2     | Reduced POWERLINK Cycle                          | 47 |
| 4.2.4.3     | POWERLINK Cycle Timing                           | 47 |
| 4.2.4.3.1   | POWERLINK Cycle Timing Error Handling            | 55 |
| 4.2.4.4     | Multiplexed Slot Timing                          | 58 |
| 4.2.4.5     | CN Cycle State Machine                           | 58 |

|               |  |    |
|---------------|--|----|
| 4.2.4.5.1     | Overview   | 58 |
| 4.2.4.5.2     | States   | 59 |
| 4.2.4.5.3     | Events   | 59 |
| 4.2.4.5.4     | Dependance of the NMT_CS on the DLL_CS   | 59 |
| 4.2.4.5.4.1   | State NMT_GS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_BASIC_ETHERNET,<br>NMT_CS_PRE_OPERATIONAL_1 | 60 |
| 4.2.4.5.4.1.1 | Transitions in other NMT states  | 60 |
| 4.2.4.5.4.2   | State NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE,<br>NMT_CS_OPERATIONAL, NMT_CS_STOPPED     | 61 |
| 4.2.4.5.4.2.1 | Transitions  | 61 |
| 4.2.4.6       | MN Cycle State Machine   | 63 |
| 4.2.4.6.1     | Overview   | 63 |
| 4.2.4.6.2     | States   | 63 |
| 4.2.4.6.3     | Events   | 63 |
| 4.2.4.6.4     | Usage of the NMT_MS state by the DLL_MS  | 64 |
| 4.2.4.6.4.1   | State NMT_GS_INITIALISATION, NMT_MS_NOT_ACTIVE   | 64 |
| 4.2.4.6.4.2   | NMT_MS_BASIC_ETHERNET  | 64 |
| 4.2.4.6.4.3   | State NMT_MS_PRE_OPERATIONAL_1   | 64 |
| 4.2.4.6.4.3.1 | Transitions  | 65 |
| 4.2.4.6.4.4   | State NMT_MS_OPERATIONAL, NMT_MS_READY_TO_OPERATE and<br>NMT_MS_PRE_OPERATIONAL_2                  | 66 |
| 4.2.4.6.4.4.1 | Transitions  | 66 |
| 4.2.5         | Recognizing Active Nodes   | 68 |
| 4.3           | Basic Ethernet Mode  | 68 |
| 4.4           | MAC Addressing   | 69 |
| 4.4.1         | MAC Unicast  | 69 |
| 4.4.2         | MAC Multicast  | 69 |
| 4.4.3         | MAC Broadcast  | 69 |
| 4.5           | POWERLINK Addressing   | 69 |
| 4.6           | Frame Structures   | 70 |
| 4.6.1         | Integration with Ethernet  | 70 |
| 4.6.1.1       | POWERLINK Frame  | 70 |
| 4.6.1.1.1     | POWERLINK Basic Frame  | 70 |
| 4.6.1.1.2     | Start of Cycle (SoC)   | 72 |
| 4.6.1.1.3     | PollRequest (PReq)   | 73 |
| 4.6.1.1.4     | PollResponse (PRes)  | 74 |
| 4.6.1.1.5     | Start of Asynchronous (SoA)  | 75 |
| 4.6.1.1.5.1   | RequestedServiceID s   | 75 |
| 4.6.1.1.6     | Asynchronous Send (ASnd)   | 76 |
| 4.6.1.1.6.1   | ServiceID values   | 76 |
| 4.6.1.2       | Non-POWERLINK Frames   | 77 |
| 4.6.1.3       | Transfer Protection  | 77 |
| 4.7           | Error Handling Data Link Layer (DLL)   | 77 |
| 4.7.1         | Possible Error Sources and Error Symptoms  | 77 |
| 4.7.2         | Error Handling Table for CN  | 78 |
| 4.7.3         | Error Handling Table for MN  | 79 |
| 4.7.4         | Error Handling Registration  | 80 |
| 4.7.4.1       | Threshold counters   | 81 |
| 4.7.4.2       | Cumulative Counter   | 81 |
| 4.7.5         | Physical Layer Error Sources   | 81 |
| 4.7.5.1       | Loss of Link   | 81 |
| 4.7.5.2       | Incorrect physical Ethernet operating mode   | 82 |
| 4.7.5.3       | Rx MAC buffer overflow / Tx MAC buffer underrun  | 82 |
| 4.7.5.4       | Transmission / CRC Errors  | 83 |
| 4.7.6         | Communication Error Symptoms detected by the MN  | 83 |
| 4.7.6.1       | Timing Violation   | 83 |
| 4.7.6.1.1     | Slot Time Exceeded   | 83 |
| 4.7.6.1.1.1   | Case 1-2 Frame received in time  | 84 |
| 4.7.6.1.1.2   | Case 3 Loss of PRes: Frame not received  | 84 |
| 4.7.6.1.1.3   | Case 4-6 Late PRes: Frame received in foreign slot (also collisions)                               | 84 |
| 4.7.6.2       | Loss of PRes   | 85 |
| 4.7.6.3       | Late PRes  | 86 |
| 4.7.6.4       | Cycle Time Exceeded  | 87 |
| 4.7.6.5       | Collisions   | 88 |
| 4.7.6.6       | Invalid Formats  | 89 |
| 4.7.6.7       | POWERLINK Address Conflicts  | 89 |
| 4.7.6.8       | Multiple MNs on a single POWERLINK Network   | 90 |
| 4.7.6.9       | Loss of StatusResponse   | 90 |
| 4.7.7         | Communication Error Symptoms detected by the CN  | 91 |
| 4.7.7.1       | Collisions   | 91 |
| 4.7.7.2       | Invalid Formats  | 92 |
| 4.7.7.3       | Loss of Frames   | 92 |
| 4.7.7.3.1     | Loss of SoC  | 93 |
| 4.7.7.3.2     | Loss of SoA  | 93 |
| 4.7.7.3.3     | Loss of PReq   | 94 |

|           |  |     |
|-----------|--|-----|
| 4.7.7.3.4 | SoC Jitter out of Range  | 94  |
| 4.7.8     | DLL Error Handling Objects   | 95  |
| 4.7.8.1   | Object 1C00 <sub>h</sub> : DLL_MNCRCError_REC                          | 95  |
| 4.7.8.2   | Object 1C01 <sub>h</sub> : DLL_MNCollision_REC                         | 96  |
| 4.7.8.3   | Object 1C02 <sub>h</sub> : DLL_MNCycTimeExceed_REC                     | 97  |
| 4.7.8.4   | Object 1C03 <sub>h</sub> : DLL_MNLossOfLinkCum_U32                     | 98  |
| 4.7.8.5   | Object 1C04 <sub>h</sub> : DLL_MNCNLatePResCumCnt_AU32                 | 99  |
| 4.7.8.6   | Object 1C05 <sub>h</sub> : DLL_MNCNLatePResThrCnt_AU32                 | 99  |
| 4.7.8.7   | Object 1C06 <sub>h</sub> : DLL_MNCNLatePResThreshold_AU32              | 100 |
| 4.7.8.8   | Object 1C07 <sub>h</sub> : DLL_MNCNLossPResCumCnt_AU32                 | 100 |
| 4.7.8.9   | Object 1C08 <sub>h</sub> : DLL_MNCNLossPResThrCnt_AU32                 | 101 |
| 4.7.8.10  | Object 1C09 <sub>h</sub> : DLL_MNCNLossPResThreshold_AU32              | 101 |
| 4.7.8.11  | Object 1C0A <sub>h</sub> : DLL_CNCollision_REC                         | 102 |
| 4.7.8.12  | Object 1C0B <sub>h</sub> : DLL_CNLossSoC_REC                           | 104 |
| 4.7.8.13  | Object 1C0C <sub>h</sub> : DLL_CNLossSoA_REC                           | 105 |
| 4.7.8.14  | Object 1C0D <sub>h</sub> : DLL_CNLossPReq_REC                          | 106 |
| 4.7.8.15  | Object 1C0E <sub>h</sub> : DLL_CNSoCJitter_REC                         | 107 |
| 4.7.8.16  | Object 1C0F <sub>h</sub> : DLL_CNCRCError_REC                          | 108 |
| 4.7.8.17  | Object 1C10 <sub>h</sub> : DLL_CNLossOfLinkCum_U32                     | 109 |
| 4.7.8.18  | Object 1C12 <sub>h</sub> : DLL_MNCycleSuspendNumber_U32                | 109 |
| 4.7.8.19  | Object 1C13 <sub>h</sub> : DLL_CNSoCJitterRange_U32                    | 109 |
| 4.7.8.20  | Object 1C14 <sub>h</sub> : DLL_CNLossOfSocTolerance_U32                | 109 |
| 4.7.8.21  | Object 1C15 <sub>h</sub> : DLL_MNLossStatusResCumCnt_AU32              | 110 |
| 4.7.8.22  | Object 1C16 <sub>h</sub> : DLL_MNLossStatusResThrCnt_AU32              | 110 |
| 4.7.8.23  | Object 1C17 <sub>h</sub> : DLL_MNLossStatusResThreshold_AU32           | 111 |
| 4.7.8.24  | Object 0424 <sub>h</sub> : DLL_ErrorCntRec_TYPE                        | 111 |
| 5         | Network / Transport Layer  | 112 |
| 5.1       | Internet Protocol (IP)   | 112 |
| 5.1.1     | IP Host Requirements   | 112 |
| 5.1.1.1   | Nodes without IP Communication   | 112 |
| 5.1.1.2   | Minimum Requirements for SDO Communication                             | 112 |
| 5.1.1.2.1 | IP Stack Requirements  | 112 |
| 5.1.1.2.2 | UDP Requirements   | 112 |
| 5.1.1.3   | Minimum Requirements for Standard IP Communication                     | 112 |
| 5.1.1.3.1 | IP Stack Requirements  | 113 |
| 5.1.2     | IP Addressing  | 113 |
| 5.1.3     | Address Resolution   | 113 |
| 5.1.4     | Hostname   | 114 |
| 5.1.5     | Object description   | 115 |
| 5.1.5.1   | Object 1E4A <sub>h</sub> : NWL_IpGroup_REC                             | 115 |
| 5.1.5.2   | Object 1E40 <sub>h</sub> .. 1E49 <sub>h</sub> : NWL_IpAddrTable_Xh_REC | 115 |
| 5.1.5.3   | Object 0425 <sub>h</sub> : NWL_IpGroup_TYPE                            | 117 |
| 5.1.5.4   | Object 0426 <sub>h</sub> : NWL_IpAddrTable_TYPE                        | 117 |
| 5.2       | POWERLINK compliant UDP/IP format                                      | 118 |
| 5.3       | POWERLINK Sequence Layer   | 118 |
| 6         | Application Layer  | 119 |
| 6.1       | Data Types and Encoding Rules  | 119 |
| 6.1.1     | General Description of Data Types and Encoding Rules                   | 119 |
| 6.1.2     | Data Type Definitions  | 119 |
| 6.1.3     | Bit Sequences  | 120 |
| 6.1.3.1   | Definition of Bit Sequences  | 120 |
| 6.1.3.2   | Transfer Syntax for Bit Sequences                                      | 121 |
| 6.1.4     | Basic Data Types   | 121 |
| 6.1.4.1   | NIL  | 121 |
| 6.1.4.2   | Boolean  | 121 |
| 6.1.4.3   | Void   | 121 |
| 6.1.4.4   | Bit  | 121 |
| 6.1.4.5   | Unsigned Integer   | 122 |
| 6.1.4.6   | Signed Integer   | 122 |
| 6.1.4.7   | Floating-Point Numbers   | 123 |
| 6.1.4.8   | MAC Address  | 123 |
| 6.1.4.9   | IP address   | 124 |
| 6.1.5     | Compound Data Types  | 124 |
| 6.1.6     | Extended Data Types  | 125 |
| 6.1.6.1   | Octet String   | 125 |
| 6.1.6.2   | Visible String   | 125 |
| 6.1.6.3   | Unicode String   | 125 |
| 6.1.6.4   | Time of Day  | 125 |
| 6.1.6.5   | Time Difference  | 125 |
| 6.1.6.6   | Domain   | 125 |
| 6.1.6.7   | Net Time   | 126 |
| 6.2       | Object Dictionary  | 126 |
| 6.2.1     | Object Dictionary Entry Definition                                     | 126 |
| 6.2.1.1   | Sub-Index Definition   | 129 |
| 6.2.2     | Data Type Entry Specification  | 130 |

|               |  |     |
|---------------|--|-----|
| 6.2.2.1       | Static Data Types  | 131 |
| 6.2.2.2       | Complex Data Types   | 131 |
| 6.2.2.3       | Extension for Multiple Device Modules  | 131 |
| 6.3           | Service Data (SDO)   | 131 |
| 6.3.1         | SDO Layer Model  | 132 |
| 6.3.1.1       | SDO Hosting in Frames  | 133 |
| 6.3.2         | SDO in Asynchronous Phase  | 133 |
| 6.3.2.1       | SDO via UDP/IP   | 133 |
| 6.3.2.1.1     | UDP Layer  | 134 |
| 6.3.2.2       | SDO via POWERLINK ASnd   | 135 |
| 6.3.2.3       | Asynchronous SDO Sequence Layer  | 136 |
| 6.3.2.3.1     | Connection   | 137 |
| 6.3.2.3.1.1   | Initialisation of Connection   | 137 |
| 6.3.2.3.1.2   | Closing a connection   | 137 |
| 6.3.2.3.1.3   | Data Transfer  | 137 |
| 6.3.2.3.1.4   | Data Transfer with Delay   | 138 |
| 6.3.2.3.1.5   | Sender History Full  | 139 |
| 6.3.2.3.2     | Errors   | 139 |
| 6.3.2.3.2.1   | Error: Loss of Frame with Data   | 140 |
| 6.3.2.3.2.2   | Error: Loss of Acknowledge Frame   | 140 |
| 6.3.2.3.2.3   | Error: Duplication of Frame  | 141 |
| 6.3.2.3.2.4   | Error: Overtaking of Frames  | 141 |
| 6.3.2.3.2.5   | Broken Connection  | 141 |
| 6.3.2.3.2.6   | Error: Flooding with commands  | 142 |
| 6.3.2.4       | Asynchronous SDO Command Layer   | 142 |
| 6.3.2.4.1     | POWERLINK Command Layer Protocol   | 143 |
| 6.3.2.4.1.1   | Download Protocol  | 145 |
| 6.3.2.4.1.2   | Upload Protocol  | 146 |
| 6.3.2.4.1.3   | Abort Transfer   | 147 |
| 6.3.2.4.2     | Commands   | 148 |
| 6.3.2.4.2.1   | SDO Protocol   | 148 |
| 6.3.2.4.2.1.1 | Command: Write by Index  | 148 |
| 6.3.2.4.2.1.2 | Command: Read by Index   | 149 |
| 6.3.2.4.2.1.3 | Command: Write All by Index  | 149 |
| 6.3.2.4.2.1.4 | Command: Read All by Index   | 150 |
| 6.3.2.4.2.1.5 | Command: Write by Name   | 150 |
| 6.3.2.4.2.1.6 | Command: Read by Name  | 151 |
| 6.3.2.4.2.2   | File Transfer  | 151 |
| 6.3.2.4.2.2.1 | Command: File Write  | 151 |
| 6.3.2.4.2.2.2 | Command: File Read   | 152 |
| 6.3.2.4.2.3   | Variable groups  | 152 |
| 6.3.2.4.2.3.1 | Command: Write Multiple Parameter by Index                                       | 152 |
| 6.3.2.4.2.3.2 | Write Multiple Parameter by Index Request  | 153 |
| 6.3.2.4.2.3.3 | Write Multiple Parameter by Index Response                                       | 153 |
| 6.3.2.4.2.3.4 | Command: Read Multiple Parameter by Index  | 154 |
| 6.3.2.4.2.3.5 | Read Multiple Parameter by Index Request   | 154 |
| 6.3.2.4.2.3.6 | Read Multiple Parameter by Index Response  | 155 |
| 6.3.2.4.2.4   | Parameter Services   | 155 |
| 6.3.2.4.2.4.1 | Command: Maximum Segment Size  | 155 |
| 6.3.3         | SDO Embedded in PDO  | 156 |
| 6.3.3.1       | Embedded Sequence Layer for SDO in PDO   | 157 |
| 6.3.3.1.1     | Connection   | 158 |
| 6.3.3.1.1.1   | Initialisation of Connection   | 158 |
| 6.3.3.1.1.2   | Closing a connection   | 159 |
| 6.3.3.1.1.3   | Data Transfer  | 160 |
| 6.3.3.1.2     | Errors   | 161 |
| 6.3.3.1.2.1   | Error: Request Lost  | 161 |
| 6.3.3.1.2.2   | Error: Response Lost   | 162 |
| 6.3.3.1.3     | Handling of Segmented Transfers  | 162 |
| 6.3.3.1.3.1   | Segmented Download from Client to Server   | 162 |
| 6.3.3.1.3.2   | Segmented Upload from Server to Client   | 163 |
| 6.3.3.2       | Embedded Command Layer for SDO in Cyclic Data                                    | 163 |
| 6.3.3.2.1     | Command Write by Index via PDO   | 163 |
| 6.3.3.2.2     | Command Read by Index via PDO  | 164 |
| 6.3.3.3       | Object Description   | 164 |
| 6.3.3.3.1     | Object 1200 <sub>h</sub> .. 127F <sub>h</sub> : SDO_ServerContainerParam_XXh_REC | 164 |
| 6.3.3.3.2     | Object 1280 <sub>h</sub> .. 12FF <sub>h</sub> : SDO_ClientContainerParam_XXh_REC | 165 |
| 6.3.3.3.3     | Object 0422 <sub>h</sub> : SDO_ParameterRecord_TYPE                              | 166 |
| 6.3.4         | SDO Timeouts   | 166 |
| 6.3.4.1       | Object 1300 <sub>h</sub> : SDO_SequLayerTimeout_U32                              | 166 |
| 6.3.4.2       | Object 1301 <sub>h</sub> : SDO_CmdLayerTimeout_U32                               | 166 |
| 6.3.4.3       | Object 1302 <sub>h</sub> : SDO_SequLayerNoAck_U32                                | 167 |
| 6.4           | Process Data Object (PDO)  | 167 |
| 6.4.1         | PDO Mapping Limitations  | 168 |
| 6.4.1.1       | TPDO Mapping Limitations   | 168 |
| 6.4.1.2       | RPDO Mapping Limitations   | 168 |

|           |  |     |
|-----------|--|-----|
| 6.4.1.3   | Further Limitations                                | 169 |
| 6.4.2     | PDO Mapping Version                                | 169 |
| 6.4.3     | SDO via PDO Container                              | 170 |
| 6.4.4     | Transmit PDOs                                      | 170 |
| 6.4.5     | Receive PDOs                                       | 170 |
| 6.4.6     | PDO via PReq                                       | 170 |
| 6.4.7     | PDO via PRes                                       | 171 |
| 6.4.8     | PDO Error Handling                                 | 171 |
| 6.4.8.1   | Dynamic Errors                                     | 171 |
| 6.4.8.1.1 | Incompatible Mapping                               | 171 |
| 6.4.8.1.2 | Unexpected End of PDO                              | 171 |
| 6.4.8.2   | Configuration Errors                               | 172 |
| 6.4.9     | Object Description                                 | 172 |
| 6.4.9.1   | Object 1400h .. 14FFh: PDO_RxCommParam_XXh_REC     | 172 |
| 6.4.9.2   | Object 1600h .. 16FFh PDO_RxMappParam_XXh_AU64     | 173 |
| 6.4.9.3   | Object 1800h .. 18FFh PDO_TxCommParam_XXh_REC      | 174 |
| 6.4.9.4   | Object 1A00h .. 1AFFh PDO_TxMappParam_XXh_AU64     | 175 |
| 6.4.9.5   | Object 1C80h: PDO_ErrMapVers_OSTR                  | 176 |
| 6.4.9.6   | Object 1C81h: PDO_ErrShort_RX_OSTR                 | 177 |
| 6.4.9.7   | Object 0420h: PDO_CommParamRecord_TYPE             | 177 |
| 6.5       | Error Signaling                                    | 177 |
| 6.5.1     | Error Entry  | 179 |
| 6.5.2     | Interface to Error Signaling                       | 180 |
| 6.5.3     | Processing of CN Error Information on the MN       | 180 |
| 6.5.4     | Error Signaling Bits                               | 180 |
| 6.5.5     | Initialisation                                     | 181 |
| 6.5.5.1   | Startup value and behaviour of the EC flag         | 181 |
| 6.5.6     | Error Signaling with PReq and PRes frames          | 182 |
| 6.5.7     | Error Signaling with Async-only CNs                | 183 |
| 6.5.8     | Format of StatusResponse Data                      | 183 |
| 6.5.8.1   | Static Error Bit Field                             | 183 |
| 6.5.8.2   | Status and History Entries                         | 183 |
| 6.5.9     | Examples   | 184 |
| 6.5.9.1   | Case 1 – Only Bit Field, No Status/History Entries | 184 |
| 6.5.9.2   | Case 2 – Status Entries                            | 185 |
| 6.5.9.3   | Case 3 – History Entries                           | 186 |
| 6.5.9.4   | Case 4 – Status and History Entries                | 186 |
| 6.5.10    | Object descriptions                                | 186 |
| 6.5.10.1  | Object 1001h : ERR_ErrorRegister_U8                | 186 |
| 6.5.10.2  | Object 1003h: ERR_History_ADOM                     | 187 |
| 6.6       | Program Download                                   | 187 |
| 6.6.1     | Object Dictionary Entries on the CN                | 188 |
| 6.6.1.1   | Object 1F50h: PDL_DownloadProgData_ADOM            | 188 |
| 6.6.1.2   | Object 1F51h: PDL_ProgCtrl_AU8                     | 188 |
| 6.6.1.3   | Object 1F52h: PDL_LocVerApplSw_REC                 | 189 |
| 6.6.1.4   | Object 0427h: PDL_LocVerApplSw_TYPE                | 190 |
| 6.6.2     | Object Dictionary Entries on the MN                | 190 |
| 6.6.2.1   | Object 1F53h: PDL_MnExpAppSwDateList_AU32          | 191 |
| 6.6.2.2   | Object 1F54h: PDL_MnExpAppSwTimeList_AU32          | 191 |
| 6.7       | Configuration Management                           | 192 |
| 6.7.1     | Device Description                                 | 192 |
| 6.7.1.1   | Local Storage on the Device                        | 192 |
| 6.7.1.2   | Central Storage on the MN                          | 192 |
| 6.7.2     | Device Configuration Storage                       | 192 |
| 6.7.2.1   | Device Configuration File Storage                  | 193 |
| 6.7.2.2   | Concise Configuration Storage                      | 193 |
| 6.7.2.3   | Check Configuration Process                        | 193 |
| 6.7.2.4   | Request Configuration                              | 193 |
| 6.7.3     | Object Dictionary Entries                          | 194 |
| 6.7.3.1   | Object 1020h: CFM_VerifyConfiguration_REC          | 194 |
| 6.7.3.2   | Object 1021h: CFM_StoreDevDescrFile_DOM            | 195 |
| 6.7.3.3   | Object 1022h: CFM_StoreDevDescrFormat_U16          | 195 |
| 6.7.3.4   | Object 1F20h: CFM_StoreDcfList_ADOM                | 196 |
| 6.7.3.5   | Object 1F21h: CFM_DcfStorageFormatList_AU8         | 196 |
| 6.7.3.6   | Object 1F22h: CFM_ConciseDcfList_ADOM              | 197 |
| 6.7.3.7   | Object 1F23h: CFM_StoreDevDescrFileList_ADOM       | 198 |
| 6.7.3.8   | Object 1F24h: CFM_DevDescrFileFormatList_AU8       | 199 |
| 6.7.3.9   | Object 1F25h: CFM_ConfCNRequest_AU32               | 199 |
| 6.7.3.10  | Object 1F26h: CFM_ExpConfDateList_AU32             | 200 |
| 6.7.3.11  | Object 1F27h: CFM_ExpConfTimeList_AU32             | 201 |
| 6.7.3.12  | Object 1F28h: CFM_ExpConfdList_AU32                | 201 |
| 6.7.3.13  | Object 0435h: CFM_VerifyConfiguration_TYPE         | 202 |
| 6.8       | Input from a Programmable Device                   | 202 |
| 6.8.1     | Basics   | 202 |
| 6.8.2     | Dynamic index assignment                           | 203 |

|               |  |     |
|---------------|--|-----|
| 6.8.3         | Object dictionary entries  | 203 |
| 6.8.3.1       | Object 1F70 <sub>h</sub> : INP_ProcessImage_REC                          | 204 |
| 6.8.3.2       | Object 0428 <sub>h</sub> : INP_ProcessImage_TYPE                         | 204 |
| 7             | Network Management (NMT)   | 205 |
| 7.1           | NMT State Machine  | 205 |
| 7.1.1         | Overview   | 205 |
| 7.1.2         | Common Initialisation NMT State Machine                                  | 205 |
| 7.1.2.1       | States   | 205 |
| 7.1.2.1.1     | NMT_GS_POWERED   | 205 |
| 7.1.2.1.1.1   | NMT_GS_INITIALISATION  | 205 |
| 7.1.2.1.1.1.1 | Sub-states   | 206 |
| 7.1.2.1.1.2   | NMT_GS_COMMUNICATING   | 207 |
| 7.1.2.2       | Transitions  | 207 |
| 7.1.3         | MN NMT State Machine   | 209 |
| 7.1.3.1       | Overview   | 209 |
| 7.1.3.2       | States   | 209 |
| 7.1.3.2.1     | NMT_MS_NOT_ACTIVE  | 209 |
| 7.1.3.2.2     | NMT_MS_EPL_MODE  | 210 |
| 7.1.3.2.2.1   | NMT_MS_PRE_OPERATIONAL_1   | 210 |
| 7.1.3.2.2.2   | NMT_MS_PRE_OPERATIONAL_2   | 210 |
| 7.1.3.2.2.3   | NMT_MS_READY_TO_OPERATE  | 210 |
| 7.1.3.2.2.4   | NMT_MS_OPERATIONAL   | 211 |
| 7.1.3.2.3     | NMT_MS_BASIC_ETHERNET  | 211 |
| 7.1.3.3       | Transitions  | 212 |
| 7.1.4         | CN NMT State Machine   | 213 |
| 7.1.4.1       | States   | 213 |
| 7.1.4.1.1     | NMT_CS_NOT_ACTIVE  | 213 |
| 7.1.4.1.2     | NMT_CS_EPL_MODE  | 214 |
| 7.1.4.1.2.1   | NMT_CS_PRE_OPERATIONAL_1   | 214 |
| 7.1.4.1.2.2   | NMT_CS_PRE_OPERATIONAL_2   | 214 |
| 7.1.4.1.2.3   | NMT_CS_READY_TO_OPERATE  | 214 |
| 7.1.4.1.2.4   | NMT_CS_OPERATIONAL   | 215 |
| 7.1.4.1.2.5   | NMT_CS_STOPPED   | 215 |
| 7.1.4.1.3     | NMT_CS_BASIC_ETHERNET  | 215 |
| 7.1.4.2       | Transitions  | 216 |
| 7.1.4.3       | States and Communication Object Relation                                 | 216 |
| 7.1.4.4       | Relationship to other state machines                                     | 217 |
| 7.2           | NMT Object Dictionary Entries  | 218 |
| 7.2.1         | NMT General Objects  | 218 |
| 7.2.1.1       | Identification   | 218 |
| 7.2.1.1.1     | Object 1000 <sub>h</sub> : NMT_DeviceType_U32                            | 218 |
| 7.2.1.1.2     | Object 1008 <sub>h</sub> : NMT_ManufactDevName_VS                        | 218 |
| 7.2.1.1.3     | Object 1009 <sub>h</sub> : NMT_ManufactHwVers_VS                         | 219 |
| 7.2.1.1.4     | Object 100A <sub>h</sub> : NMT_ManufactSwVers_VS                         | 219 |
| 7.2.1.1.5     | Object 1018 <sub>h</sub> : NMT_IdentityObject_REC                        | 219 |
| 7.2.1.1.6     | Object 1F82 <sub>h</sub> : NMT_FeatureFlags_U32                          | 220 |
| 7.2.1.1.7     | Object 1F83 <sub>h</sub> : NMT_EPLVersion_U8                             | 222 |
| 7.2.1.2       | Parameter Storage  | 222 |
| 7.2.1.2.1     | Object 1010 <sub>h</sub> : NMT_StoreParam_REC                            | 222 |
| 7.2.1.2.2     | Object 1011 <sub>h</sub> : NMT_RestoreDefParam_REC                       | 224 |
| 7.2.1.3       | Communication Interface Description                                      | 226 |
| 7.2.1.3.1     | Object 1F93 <sub>h</sub> : NMT_EPLNodeID_REC                             | 226 |
| 7.2.1.3.2     | Object 1030 <sub>h</sub> ..1039 <sub>h</sub> : NMT_InterfaceGroup_Xh_REC | 227 |
| 7.2.1.3.3     | Object 1F9A <sub>h</sub> : NMT_HostName_VSTR                             | 229 |
| 7.2.1.4       | Node List  | 230 |
| 7.2.1.4.1     | Object 1F81 <sub>h</sub> : NMT_NodeAssignment_AU32                       | 230 |
| 7.2.1.5       | Timing   | 232 |
| 7.2.1.5.1     | Object 1006 <sub>h</sub> : NMT_CycleLen_U32                              | 232 |
| 7.2.1.5.2     | Object 1F98 <sub>h</sub> : NMT_CycleTiming_REC                           | 232 |
| 7.2.1.5.3     | Object 1F9B <sub>h</sub> : NMT_MultiplCycleAssign_AU8                    | 235 |
| 7.2.1.5.4     | Object 1016 <sub>h</sub> : NMT_ConsumerHeartbeatTime_AU32                | 236 |
| 7.2.1.5.5     | Object 1F8D <sub>h</sub> : NMT_PResPayloadLimitList_AU16                 | 237 |
| 7.2.1.6       | NMT Service Interface  | 238 |
| 7.2.1.6.1     | Object 1F9E <sub>h</sub> : NMT_ResetCmd_U8                               | 238 |
| 7.2.1.7       | NMT Diagnostics  | 239 |
| 7.2.1.7.1     | Object 1F8C <sub>h</sub> : NMT_CurrNMTState_U8                           | 239 |
| 7.2.2         | NMT MN Objects   | 239 |
| 7.2.2.1       | MN Start Up Behavior   | 239 |
| 7.2.2.1.1     | Object 1F80 <sub>h</sub> : NMT_StartUp_U32                               | 239 |
| 7.2.2.1.2     | Object 1F89 <sub>h</sub> : NMT_BootTime_REC                              | 241 |
| 7.2.2.2       | NMT Master Network Node Lists  | 243 |
| 7.2.2.2.1     | Object 1F84 <sub>h</sub> : NMT_MNDeviceTypeIdList_AU32                   | 243 |
| 7.2.2.2.2     | Object 1F85 <sub>h</sub> : NMT_MNVendorIdList_AU32                       | 244 |
| 7.2.2.2.3     | Object 1F86 <sub>h</sub> : NMT_MNProductCodeList_AU32                    | 245 |
| 7.2.2.2.4     | Object 1F87 <sub>h</sub> : NMT_MNRevisionNoList_AU32                     | 245 |

|             |   |     |
|-------------|---|-----|
| 7.2.2.2.5   | Object 1F88h: NMT_MNSerialNoList_AU32         | 246 |
| 7.2.2.3     | Timing  | 247 |
| 7.2.2.3.1   | Object 1F8Ah: NMT_MNCycleTiming_REC           | 247 |
| 7.2.2.3.2   | Object 1F8Bh: NMT_MNPReqPayloadLimitList_AU16 | 248 |
| 7.2.2.3.3   | Object 1F92h: NMT_MNCNPResTimeout_AU32        | 249 |
| 7.2.2.3.4   | Object 1F9Ch: NMT_IsochrSlotAssign_AU8        | 250 |
| 7.2.2.4     | CN NMT State Surveillance                     | 252 |
| 7.2.2.4.1   | Object 1F8Eh: NMT_MNNodeCurrState_AU8         | 252 |
| 7.2.2.4.2   | Object 1F8Fh: NMT_MNNodeExpState_AU8          | 252 |
| 7.2.2.5     | NMT Service Interface                         | 253 |
| 7.2.2.5.1   | Object 1F9Fh: NMT_RequestCmd_REC              | 253 |
| 7.2.3       | NMT CN Objects                                | 254 |
| 7.2.3.1     | CN StartUp Behaviour                          | 254 |
| 7.2.3.1.1   | Object 1F99h: NMT_CNBASICEthernetTimeout_U32  | 254 |
| 7.2.4       | NMT Object Types                              | 255 |
| 7.2.4.1     | Object 0023h: IDENTITY                        | 255 |
| 7.2.4.2     | Object 0429h: NMT_ParameterStorage_TYPE       | 255 |
| 7.2.4.3     | Object 042Bh: NMT_InterfaceGroup_Xh_TYPE      | 255 |
| 7.2.4.4     | Object 042Ch: NMT_CycleTiming_TYPE            | 255 |
| 7.2.4.5     | Object 042Eh: NMT_BootTime_TYPE               | 256 |
| 7.2.4.6     | Object 042Fh: NMT_MNCycleTiming_TYPE          | 256 |
| 7.2.4.7     | Object 0439h: NMT_EPLNodeID_TYPE              | 256 |
| 7.2.4.8     | Object 043Ah: NMT_RequestCmd_TYPE             | 256 |
| 7.3         | Network Management Services                   | 257 |
| 7.3.1       | NMT State Command Services                    | 257 |
| 7.3.1.1     | Implicit NMT State Command Services           | 257 |
| 7.3.1.1.1   | Implicit NMT State Command Transmission       | 258 |
| 7.3.1.2     | Explicit NMT State Command Services           | 258 |
| 7.3.1.2.1   | Plain NMT State Command                       | 259 |
| 7.3.1.2.1.1 | NMT Reset Commands to the MN                  | 261 |
| 7.3.1.2.2   | Extended NMT State Command                    | 261 |
| 7.3.1.2.3   | POWERLINK Node List Format                    | 261 |
| 7.3.2       | NMT Managing Command Services                 | 262 |
| 7.3.2.1     | Service Descriptions                          | 263 |
| 7.3.2.1.1   | NMTNetHostNameSet                             | 263 |
| 7.3.2.1.2   | NMTFlushArpEntry                              | 264 |
| 7.3.3       | NMT Response Services                         | 264 |
| 7.3.3.1     | NMT State Response                            | 264 |
| 7.3.3.2     | IdentResponse Service                         | 265 |
| 7.3.3.2.1   | IdentResponse Frame                           | 266 |
| 7.3.3.3     | StatusResponse Service                        | 268 |
| 7.3.3.3.1   | StatusResponse Frame                          | 268 |
| 7.3.4       | NMT Info Services                             | 269 |
| 7.3.4.1     | Service Descriptions                          | 270 |
| 7.3.4.1.1   | NMTPublishConfiguredNodes                     | 270 |
| 7.3.4.1.2   | NMTPublishActiveNodes                         | 270 |
| 7.3.4.1.3   | NMTPublishPreOperational1                     | 270 |
| 7.3.4.1.4   | NMTPublishPreOperational2                     | 271 |
| 7.3.4.1.5   | NMTPublishReadyToOperate                      | 271 |
| 7.3.4.1.6   | NMTPublishOperational                         | 271 |
| 7.3.4.1.7   | NMTPublishStopped                             | 271 |
| 7.3.4.1.8   | NMTPublishNodeStates                          | 271 |
| 7.3.4.1.9   | NMTPublishEmergencyNew                        | 272 |
| 7.3.4.1.10  | NMTPublishTime                                | 272 |
| 7.3.5       | NMT Guard Services                            | 272 |
| 7.3.5.1     | Guarding CNs                                  | 272 |
| 7.3.5.1.1   | Guarding Async-Only CNs                       | 272 |
| 7.3.5.2     | Guarding the MN                               | 273 |
| 7.3.6       | Request NMT Services by a CN                  | 273 |
| 7.3.6.1     | NMTRequest Frame                              | 273 |
| 7.3.6.1.1   | Invalid NMTRequests                           | 273 |
| 7.3.7       | NMT Services via Object Dictionary            | 274 |
| 7.3.7.1     | NMT Reset Commands                            | 274 |
| 7.3.7.2     | NMT Requests to the MN                        | 274 |
| 7.3.8       | NMT Services via UDP/IP                       | 274 |
| 7.4         | Boot-up Managing Node                         | 275 |
| 7.4.1       | NMT_MS dependant Network Boot-up              | 275 |
| 7.4.1.1     | Overview                                      | 275 |
| 7.4.1.2     | NMT_MS_NOT_ACTIVE                             | 275 |
| 7.4.1.3     | NMT_MS_PRE_OPERATIONAL_1                      | 276 |
| 7.4.1.4     | NMT_MS_PRE_OPERATIONAL_2                      | 277 |
| 7.4.1.5     | NMT_MS_READY_TO_OPERATE                       | 279 |
| 7.4.1.6     | NMT_MS_OPERATIONAL                            | 280 |
| 7.4.2       | MN Boot-up Procedure on CN Level              | 282 |
| 7.4.2.1     | Overview                                      | 282 |
| 7.4.2.2     | Boot-up of optional and mandatory CNs         | 282 |

|               |  |     |
|---------------|--|-----|
| 7.4.2.2.1     | BOOT_STEP1   | 283 |
| 7.4.2.2.1.1   | CHECK_IDENTIFICATION   | 284 |
| 7.4.2.2.1.2   | CHECK_SOFTWARE   | 285 |
| 7.4.2.2.1.3   | CHECK_CONFIGURATION  | 287 |
| 7.4.2.2.1.3.1 | GET_IDENT  | 288 |
| 7.4.2.2.2     | BOOT_STEP2   | 289 |
| 7.4.2.2.3     | CHECK_COMMUNICATION  | 290 |
| 7.4.2.2.4     | START_CN   | 291 |
| 7.4.2.2.5     | START_ALL  | 292 |
| 7.4.2.2.6     | CHECK_STATE  | 293 |
| 7.4.2.2.7     | CHANGE_NMT_STATE   | 294 |
| 7.4.2.2.8     | OPERATIONAL  | 294 |
| 7.4.2.2.9     | ERROR_TREATMENT  | 294 |
| 7.4.3         | Boot-up Errors   | 295 |
| 7.4.3.1       | Bus activity   | 295 |
| 7.4.3.2       | BOOT_STEP1 failed  | 296 |
| 7.4.3.3       | BOOT_STEP2 failed  | 296 |
| 7.4.3.4       | Boot-up in NMT_MS_READY_TO_OPERATE failed                                  | 296 |
| 7.4.3.5       | Get Ident failed   | 296 |
| 7.4.3.6       | Device Type Invalid  | 297 |
| 7.4.3.7       | Vendor ID invalid  | 297 |
| 7.4.3.8       | Configuration failed   | 297 |
| 7.4.3.9       | Product Code invalid   | 297 |
| 7.4.3.10      | Revision number invalid  | 298 |
| 7.4.3.11      | Serial number invalid  | 298 |
| 7.4.3.12      | NMT state invalid  | 298 |
| 7.4.3.13      | Invalid Software   | 298 |
| 7.4.3.14      | Invalid NMT state for SW update  | 299 |
| 7.4.3.15      | SW update not allowed  | 299 |
| 7.4.3.16      | SW update failed   | 299 |
| 7.4.4         | Minimal Boot-up MN   | 300 |
| 7.4.5         | Example Boot-up Sequence   | 301 |
| 7.4.6         | Application Notes  | 301 |
| 8             | Diagnostics  | 303 |
| 8.1           | Diagnostic Object Dictionary Entries                                       | 303 |
| 8.1.1         | Object 1101 <sub>h</sub> : DIA_NMTTelegrCount_REC                          | 303 |
| 8.1.2         | Object 1102 <sub>h</sub> : DIA_ERRStatistics_REC                           | 305 |
| 8.1.3         | Diagnostics Object Types   | 306 |
| 8.1.3.1       | Object 0437 <sub>h</sub> : DIA_NMTTelegrCount_TYPE                         | 306 |
| 8.1.3.2       | Object 0438 <sub>h</sub> : DIA_ERRStatistics_TYPE                          | 307 |
| 9             | Routing  | 308 |
| 9.1           | Routing Type 1   | 308 |
| 9.1.1         | Core Tasks of a POWERLINK Router   | 308 |
| 9.1.2         | Reference Model  | 309 |
| 9.1.3         | Data Link Layer  | 309 |
| 9.1.3.1       | DLL POWERLINK Interface  | 310 |
| 9.1.3.2       | DLL interface to the external network                                      | 310 |
| 9.1.4         | Network Layer  | 310 |
| 9.1.4.1       | Communication between POWERLINK and the external network                   | 310 |
| 9.1.4.2       | IP Coupling  | 310 |
| 9.1.4.2.1     | IP Routing   | 310 |
| 9.1.4.2.1.1   | Configuration  | 311 |
| 9.1.4.2.1.1.1 | SNMP   | 311 |
| 9.1.4.2.1.1.2 | SDO  | 311 |
| 9.1.4.2.2     | Network Address Translation (NAT)  | 311 |
| 9.1.4.2.2.1   | Configuration  | 313 |
| 9.1.4.2.2.1.1 | SNMP   | 313 |
| 9.1.4.2.2.1.2 | SDO  | 313 |
| 9.1.5         | Security   | 313 |
| 9.1.5.1       | Packet Filter – Firewall   | 314 |
| 9.1.5.1.1     | ACL – Filter Entries   | 315 |
| 9.1.5.1.2     | Filter strategy  | 315 |
| 9.1.5.1.3     | Configuration  | 315 |
| 9.1.5.1.3.1   | SNMP   | 315 |
| 9.1.5.1.3.2   | SDO  | 315 |
| 9.1.6         | Additional Services of a POWERLINK Router                                  | 315 |
| 9.1.7         | Object description   | 316 |
| 9.1.7.1       | Object 1E80 <sub>h</sub> : RT1_EplRouter_REC                               | 316 |
| 9.1.7.2       | Object 1E90 <sub>h</sub> .. 1ECF <sub>h</sub> : RT1_IpRoutingTable_XXh_REC | 316 |
| 9.1.7.3       | Object 1D00 <sub>h</sub> .. 1DFF <sub>h</sub> : RT1_NatTable_XXh_REC       | 318 |
| 9.1.7.4       | Object 1E81 <sub>h</sub> : RT1_SecurityGroup_REC                           | 320 |
| 9.1.7.5       | Object 1B00 <sub>h</sub> .. 1BFF <sub>h</sub> : RT1_AclFwdTable_XXh_REC    | 320 |
| 9.1.7.6       | Object 1ED0 <sub>h</sub> .. 1EDF <sub>h</sub> : RT1_AclInTable_Xh_REC      | 323 |
| 9.1.7.7       | Object 1EE0 <sub>h</sub> .. 1EEF <sub>h</sub> : RT1_AclOutTable_Xh_REC     | 325 |
| 9.1.7.8       | Router Type I Object Types   | 328 |

---

|           |  |     |
|-----------|--|-----|
| 9.1.7.8.1 | Object 0430_h: RT1_EplRouter_TYPE          | 328 |
| 9.1.7.8.2 | Object 0431_h: RT1_IpRoutingTable_TYPE     | 328 |
| 9.1.7.8.3 | Object 0432_h: RT1_NatTable_TYPE           | 328 |
| 9.1.7.8.4 | Object 0433_h: RT1_SecurityGroup_TYPE      | 328 |
| 9.1.7.8.5 | Object 0434_h: RT1_AclTable_TYPE           | 328 |
| 9.1.8     | POWERLINK Router MIB                       | 329 |
| 9.2       | Routing Type 2                             | 329 |
| 10        | Indicators                                 | 330 |
| 10.1      | Indicator states and flash rates           | 330 |
| 10.2      | Indicator Signaling                        | 331 |
| 10.3      | Recommended labelling                      | 332 |
| App. 1    | Summary Object Library (normative)         | 333 |
| App. 1.1  | Object Dictionary Entries, sorted by index | 333 |
| App. 1.2  | Object Dictionary Entries, sorted by name  | 339 |
| App. 2    | Device Description Entries (normative)     | 344 |
| App. 3    | Constant Value Assignments (normative)     | 348 |
| App. 3.1  | POWERLINK Message Type Ids                 | 348 |
| App. 3.2  | AsyncSend Request Priorities               | 348 |
| App. 3.3  | ASnd ServiceIDs                            | 348 |
| App. 3.4  | SoA RequestedServiceIDs                    | 349 |
| App. 3.5  | Object Dictionary Object Types             | 349 |
| App. 3.6  | NMT States                                 | 349 |
| App. 3.7  | NMT Commands                               | 350 |
| App. 3.8  | General Purpose Constants                  | 351 |
| App. 3.9  | Error Code Constants                       | 352 |
| App. 3.10 | SDO Abort Codes                            | 354 |
| App. 4    | Data Sheet Requirements (normative)        | 355 |

## Pre. 5 Tables

|         |  |     |
|---------|--|-----|
| Tab. 1  | Object dictionary structure  | 31  |
| Tab. 2  | Pin assignment RJ45 port   | 36  |
| Tab. 3  | Pin assignment IP67 port   | 37  |
| Tab. 4  | POWERLINK cycle timing parameters  | 54  |
| Tab. 5  | POWERLINK cycle timing verification: Error codes and handling  | 57  |
| Tab. 6  | Transitions for CN cycle state machine, states NMT_GS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PRE_OPERATIONAL_1, NMT_CS_BASIC_ETHERNET | 60  |
| Tab. 7  | Transitions for CN cycle state machine, states NMT_CS_OPERATIONAL, NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE                     | 62  |
| Tab. 8  | Transitions for MN cycle state machine, state NMT_MS_PRE_OPERATIONAL_1   | 65  |
| Tab. 9  | Transitions for MN cycle state machine, states NMT_MS_OPERATIONAL, NMT_MS_READY_TO_OPERATE and NMT_MS_PRE_OPERATIONAL_2                  | 67  |
| Tab. 10 | Assigned multicast addresses   | 69  |
| Tab. 11 | POWERLINK Node ID assignment   | 70  |
| Tab. 12 | Ethernet POWERLINK frame structure   | 71  |
| Tab. 13 | Ethernet POWERLINK frame fields  | 71  |
| Tab. 14 | POWERLINK message types  | 71  |
| Tab. 15 | SoC frame structure  | 72  |
| Tab. 16 | SoC frame data fields  | 72  |
| Tab. 17 | PReq frame structure   | 73  |
| Tab. 18 | PReq frame data fields   | 73  |
| Tab. 19 | PRes frame structure   | 74  |
| Tab. 20 | PRes frame data fields   | 74  |
| Tab. 21 | SoA frame structure  | 75  |
| Tab. 22 | SoA frame data fields  | 75  |
| Tab. 23 | Definition of the RequestedServiceID in the SoA frame  | 76  |
| Tab. 24 | ASnd frame structure   | 76  |
| Tab. 25 | ASnd frame data fields   | 76  |
| Tab. 26 | ServiceID values in the ASnd frame   | 77  |
| Tab. 27 | CN error handling table  | 78  |
| Tab. 28 | MN error handling table  | 79  |
| Tab. 29 | IP parameters of a POWERLINK node  | 113 |
| Tab. 30 | POWERLINK compliant UDP/IP frame structure   | 118 |
| Tab. 31 | Transfer syntax for bit sequences  | 121 |
| Tab. 32 | Transfer syntax for data type UNSIGNEDn  | 122 |
| Tab. 33 | Transfer syntax for data type INTEGERn   | 123 |
| Tab. 34 | Transfer syntax of data type REAL32  | 123 |
| Tab. 35 | MAC address encoding, <i>example 00-0A-86-xx-xx-xx shows a Lenze device</i>  | 124 |
| Tab. 36 | IP address encoding, <i>example shows the IP address of a POWERLINK MN 192.168.100.240</i>   | 124 |
| Tab. 37 | Object type definitions  | 127 |
| Tab. 38 | Access attributes for data objects   | 128 |
| Tab. 39 | PDO mapping attributes for data objects  | 128 |
| Tab. 40 | Static data type object definition example   | 128 |
| Tab. 41 | Complex data type object definition example  | 129 |
| Tab. 42 | NumberOfEntries sub-index description example  | 129 |
| Tab. 43 | Record type object sub-index description example   | 129 |
| Tab. 44 | Array type object sub-index description example  | 130 |
| Tab. 45 | Structure of sub-index FF <sub>n</sub>   | 130 |
| Tab. 46 | Complex data type description example  | 131 |
| Tab. 47 | SDO via UDP/IP   | 134 |
| Tab. 48 | SDO via UDP/IP field interpretation  | 134 |
| Tab. 49 | UDP header   | 134 |
| Tab. 50 | SDO via POWERLINK ASnd   | 135 |
| Tab. 51 | SDO via ASnd field interpretation  | 136 |
| Tab. 52 | POWERLINK sequence layer in asynchronous data frame  | 136 |
| Tab. 53 | Fields of POWERLINK sequence layer in asynchronous data frame  | 136 |
| Tab. 54 | POWERLINK command layer  | 143 |
| Tab. 55 | POWERLINK command layer field interpretation   | 144 |
| Tab. 56 | Abort transfer frame   | 147 |
| Tab. 57 | Abort transfer frame field interpretation  | 147 |
| Tab. 58 | Command services and command ID  | 148 |
| Tab. 59 | Command: Write by Index request  | 149 |
| Tab. 60 | Write by Index request field interpretation  | 149 |
| Tab. 61 | Command: Read by Index Request   | 149 |
| Tab. 62 | Read by Index request field interpretation   | 149 |
| Tab. 63 | Command: Write All by Index request  | 149 |
| Tab. 64 | Write All by Index request field interpretation  | 150 |
| Tab. 65 | Command: Read All by Index Request   | 150 |
| Tab. 66 | Read All by Index request field interpretation   | 150 |
| Tab. 67 | Command: Write by Name request   | 150 |
| Tab. 68 | Write by Name request field interpretation   | 150 |
| Tab. 69 | Command: Read by Name request  | 151 |

|          |  |     |
|----------|--|-----|
| Tab. 70  | Read by Name request field interpretation  | 151 |
| Tab. 71  | Command: File Write request  | 152 |
| Tab. 72  | File Write request field interpretation  | 152 |
| Tab. 73  | Command: File Read request   | 152 |
| Tab. 74  | File Read request field interpretation   | 152 |
| Tab. 75  | Command: Write Multiple Parameter by Index Request   | 153 |
| Tab. 76  | Write Multiple Parameter by Index request field interpretation   | 153 |
| Tab. 77  | Command: Write Multiple Parameter by Index Response  | 153 |
| Tab. 78  | Write Multiple Parameter by Index response field interpretation  | 154 |
| Tab. 79  | Command: Read Multiple Parameter by Index request  | 154 |
| Tab. 80  | Read Multiple Parameter by Index request field interpretation  | 154 |
| Tab. 81  | Command: Read Multiple Parameter by Index response   | 155 |
| Tab. 82  | Read Multiple Parameter by Index response field interpretation   | 155 |
| Tab. 83  | Command: Maximum Segment Size  | 156 |
| Tab. 84  | Maximum Segment Size field interpretation  | 156 |
| Tab. 85  | SDO embedded in PDO  | 156 |
| Tab. 86  | SDO embedded in PDO field interpretation   | 157 |
| Tab. 87  | POWERLINK sequence layer for embedding of SDO in cyclic data   | 157 |
| Tab. 88  | Fields of POWERLINK sequence layer for embedding of SDO in cyclic data   | 158 |
| Tab. 89  | Command: Write by Index Request via PDO  | 163 |
| Tab. 90  | Command: Read by Index Request via PDO   | 164 |
| Tab. 91  | Structure of PDO Mapping version   | 169 |
| Tab. 92  | Structure of PDO Mapping Entry   | 174 |
| Tab. 93  | Internal bit mapping of PDO mapping entry  | 174 |
| Tab. 94  | Format of one entry  | 179 |
| Tab. 95  | Description of one entry   | 179 |
| Tab. 96  | Format of the field entry type   | 179 |
| Tab. 97  | Error signaling bits   | 180 |
| Tab. 98  | Static error bit field   | 183 |
| Tab. 99  | Abbreviations for the following examples   | 184 |
| Tab. 100 | PDL_ProgCtrl_AU8 sub-index value interpretation  | 189 |
| Tab. 101 | Device description file and device configuration storage formats   | 196 |
| Tab. 102 | Concise DCF stream format  | 198 |
| Tab. 103 | CNConfigurationRequest write access signature  | 200 |
| Tab. 104 | Structure of SelectedRange_U32   | 204 |
| Tab. 105 | Common initialisation NMT state transitions  | 208 |
| Tab. 106 | MN specific state transitions  | 212 |
| Tab. 107 | CN specific state transitions  | 216 |
| Tab. 108 | States and communication objects   | 217 |
| Tab. 109 | NMT_DeviceType_U32 value interpretation  | 218 |
| Tab. 110 | Structure of Revision number   | 220 |
| Tab. 111 | NMT_FeatureFlags_U32 interpretation  | 222 |
| Tab. 112 | NMT_EPLVersion_U8 encoding   | 222 |
| Tab. 113 | NMT_StoreParam_REC storage write access signature  | 223 |
| Tab. 114 | NMT_StoreParam_REC storage read access structure   | 223 |
| Tab. 115 | NMT_StoreParam_REC structure of read access  | 224 |
| Tab. 116 | NMT_RestoreDefParam_REC restoring write access signature   | 225 |
| Tab. 117 | NMT_RestoreDefParam_REC restoring default values read access structure   | 225 |
| Tab. 118 | NMT_RestoreDefParam_REC structure of restore read access   | 225 |
| Tab. 119 | NMT_NodeAssignment_AU32 interpretation   | 232 |
| Tab. 120 | HeartbeatDescription value interpretation  | 237 |
| Tab. 121 | NMT_StartUp_U32 interpretation   | 240 |
| Tab. 122 | Implicit NMT state commands  | 258 |
| Tab. 123 | NMT state command service, NMT managing command service and NMT info service structure of the NMT Service Slot field | 259 |
| Tab. 124 | NMT Service Slot fields of explicit NMT state command services   | 259 |
| Tab. 125 | Plain NMT state commands   | 260 |
| Tab. 126 | Extended NMT state commands  | 261 |
| Tab. 127 | POWERLINK node list: Node ID to bit assignment   | 262 |
| Tab. 128 | NMT Service Slot fields of NMT managing command services   | 263 |
| Tab. 129 | NMT managing command services  | 263 |
| Tab. 130 | NMTCommandData structure of NMTNetHostNameSet  | 263 |
| Tab. 131 | NMTCommandData data fields of NMTNetHostNameSet  | 263 |
| Tab. 132 | NMTFlushArpEntry ASnd service slot structure   | 264 |
| Tab. 133 | NMTCommandData data fields of NMTFlushArpEntry   | 264 |
| Tab. 134 | NMT Service Slot structure of IdentResponse  | 266 |
| Tab. 135 | NMT Service Slot data fields of IdentResponse  | 268 |
| Tab. 136 | NMT Service Slot structure of StatusResponse   | 268 |
| Tab. 137 | NMT Service Slot data fields of StatusResponse   | 269 |
| Tab. 138 | NMT Service Slot data fields of NMT managing info services   | 269 |
| Tab. 139 | NMT info services  | 270 |
| Tab. 140 | NMTCommandData structure of NMTPublishNodeStates   | 271 |
| Tab. 141 | NMTCommandData data fields of NMTPublishNodeStates   | 271 |
| Tab. 142 | NMTCommandData structure of NMTPublishTime   | 272 |
| Tab. 143 | NMTCommandData data fields of NMTPublishTime   | 272 |
| Tab. 144 | NMT Service Slot structure of NMTRquest  | 273 |

Tab. 145 NMT Service Slot data fields of an NMTRequest frame

273

## Pre. 6 Figures

|          |  |     |
|----------|--|-----|
| Fig. 1.  | Slot Communication Network Management (SCNM)   | 25  |
| Fig. 2.  | Integration POWERLINK based machines into the IT infrastructure of end customer  | 27  |
| Fig. 3.  | Typical centralized and decentralized controller structures  | 27  |
| Fig. 4.  | Reference model  | 28  |
| Fig. 5.  | Service types  | 29  |
| Fig. 6.  | Device model   | 30  |
| Fig. 7.  | Unconfirmed master slave communication   | 32  |
| Fig. 8.  | Confirmed master slave communication   | 33  |
| Fig. 9.  | Client/Server communication  | 33  |
| Fig. 10. | Push model   | 33  |
| Fig. 11. | Pull model   | 34  |
| Fig. 12. | Star topology and line topology  | 35  |
| Fig. 13. | RJ45 pin assignment (left: connector, right: port)   | 36  |
| Fig. 14. | IP67 port pin assignment.  | 37  |
| Fig. 15. | recommended RJ45 to RJ45 pin assignment  | 37  |
| Fig. 16. | not recommended RJ45 to RJ45 pin assignment  | 37  |
| Fig. 17. | M12 to M12 pin assignment  | 37  |
| Fig. 18. | M12 to RJ45 pin assignment   | 38  |
| Fig. 19. | POWERLINK Cycle  | 41  |
| Fig. 20. | POWERLINK - an isochronous process   | 41  |
| Fig. 21. | Multiplexed POWERLINK cycle  | 42  |
| Fig. 22. | Asynchronous scheduling  | 44  |
| Fig. 23. | Asynchronous transmit priority handling (Priority level PR: 7 = PRIO_NMT_REQUEST, 3 = PRIO_GENERIC_REQUEST)                    | 46  |
| Fig. 24. | Reduced POWERLINK cycle  | 47  |
| Fig. 25. | POWERLINK cycle timing, start phase and isochronous phase  | 48  |
| Fig. 26. | POWERLINK cycle timing, asynchronous phase, AsyncSend transmission by CN   | 49  |
| Fig. 27. | POWERLINK cycle timing, asynchronous phase, AsyncSend transmission by MN   | 49  |
| Fig. 28. | Multiple slot assignment   | 58  |
| Fig. 29. | CN cycle state machine, states NMT_GS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PRE_OPERATIONAL_1, NMT_CS_BASIC_ETHERNET       | 60  |
| Fig. 30. | CN cycle state machine (DLL_CS), valid for NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE, NMT_CS states NMT_CS_OPERATIONAL | 61  |
| Fig. 31. | MN cycle state machine, state NMT_MS_PRE_OPERATIONAL_1   | 65  |
| Fig. 32. | MN cycle state machine, States NMT_MS_OPERATIONAL, NMT_MS_READY_TO_OPERATE and NMT_MS_PRE_OPERATIONAL_2                        | 66  |
| Fig. 33. | Error registration   | 80  |
| Fig. 34. | Threshold counter  | 81  |
| Fig. 35. | Timeouts   | 84  |
| Fig. 36. | Timing violation   | 85  |
| Fig. 37. | Cycle time exceeded  | 87  |
| Fig. 38. | Construction of the IPv4 address   | 113 |
| Fig. 39. | POWERLINK frame structure  | 119 |
| Fig. 40. | POWERLINK compliant UDP/IP frame structure   | 119 |
| Fig. 41. | Legacy Ethernet frame structure  | 119 |
| Fig. 42. | SDO layer model  | 132 |
| Fig. 43. | POWERLINK SDO embedded in UDP/IP frame   | 133 |
| Fig. 44. | UDP socket   | 135 |
| Fig. 45. | Initialisation of a asynchronous connection  | 137 |
| Fig. 46. | Closing of asynchronous connection   | 137 |
| Fig. 47. | Normal asynchronous communication  | 138 |
| Fig. 48. | Delayed asynchronous communication   | 139 |
| Fig. 49. | Asynchronous communication when history buffer gets full   | 139 |
| Fig. 50. | Error loss of asynchronous frame   | 140 |
| Fig. 51. | Error loss of asynchronous acknowledge   | 140 |
| Fig. 52. | Error duplication of asynchronous frame  | 141 |
| Fig. 53. | Error asynchronous communication broken  | 142 |
| Fig. 54. | Error flooding with asynchronous commands  | 142 |
| Fig. 55. | Information structure of POWERLINK command layer   | 143 |
| Fig. 56. | Definition of segment size   | 144 |
| Fig. 57. | POWERLINK command layer: Typical download transfer   | 145 |
| Fig. 58. | POWERLINK command layer: Typical upload transfer   | 146 |
| Fig. 59. | Abort transfer   | 147 |
| Fig. 60. | Initialisation of embedded connection  | 158 |
| Fig. 61. | Closing of connection  | 159 |
| Fig. 62. | Normal embedded communication  | 160 |
| Fig. 63. | Error embedded request lost  | 161 |
| Fig. 64. | Error embedded response lost   | 162 |
| Fig. 65. | Embedded segmented download  | 162 |
| Fig. 66. | Embedded segmented upload  | 163 |
| Fig. 67. | Error signaling – Reference model  | 178 |
| Fig. 68. | Error signaling – Overview   | 178 |

---

|           |  |     |
|-----------|--|-----|
| Fig. 69.  | Error signaling initialisation   | 181 |
| Fig. 70.  | Error signaling with PReq and PRes   | 182 |
| Fig. 71.  | Error signaling for Async-only CNs and CNs in state NMT_CS_PRE_OPERATIONAL_1   | 183 |
| Fig. 72.  | Common initialisation NMT state machine  | 206 |
| Fig. 73.  | NMT state diagram of an MN   | 209 |
| Fig. 74.  | State diagram of a CN  | 213 |
| Fig. 75.  | NMT_RestoreDefParam_REC restore procedure                                      | 226 |
| Fig. 76.  | POWERLINK communication slots  | 250 |
| Fig. 77.  | Implicit NMT state command service protocol                                    | 258 |
| Fig. 78.  | Explicit NMT state command service protocol                                    | 259 |
| Fig. 79.  | NMT managing command service protocol  | 263 |
| Fig. 80.  | NMT state response service protocol (isochronous CN)                           | 264 |
| Fig. 81.  | NMT state response service protocol (async-only CN)                            | 265 |
| Fig. 82.  | IdentResponse service protocol   | 265 |
| Fig. 83.  | StatusResponse service protocol  | 268 |
| Fig. 84.  | NMT info service protocol  | 269 |
| Fig. 85.  | State NMT_MS_NOT_ACTIVE  | 275 |
| Fig. 86.  | Detail state NMT_MS_PRE_OPERATIONAL_1  | 277 |
| Fig. 87.  | Detail state NMT_MS_PRE_OPERATIONAL_2  | 278 |
| Fig. 88.  | Detail state NMT_MS_READY_TO_OPERATE   | 279 |
| Fig. 89.  | Detail state NMT_MS_OPERATIONAL  | 281 |
| Fig. 90.  | Overview of the boot process in NMT super-state NMT_MS                         | 282 |
| Fig. 91.  | Network boot process dependencies to the NMT_MS for optional and mandatory CNs | 283 |
| Fig. 92.  | Sub-state BOOT_STEP1   | 284 |
| Fig. 93.  | Sub-state CHECK_IDENTIFICATION[Node ID]  | 285 |
| Fig. 94.  | Sub-state CHECK_SOFTWARE[Node ID]  | 286 |
| Fig. 95.  | Sub-state CHECK_CONFIGURATION[Node ID]   | 287 |
| Fig. 96.  | Sub-state GET_IDENT[Node ID]   | 288 |
| Fig. 97.  | Sub-state BOOT_STEP2[Node ID]  | 289 |
| Fig. 98.  | Sub-state CHECK_COMMUNICATION[Node ID]   | 290 |
| Fig. 99.  | Sub-state START_CN[Node ID]  | 291 |
| Fig. 100. | Sub-state START_ALL  | 292 |
| Fig. 101. | Sub-state CHECK_STATE  | 293 |
| Fig. 102. | Sub-state CHANGE_NMT_STATE   | 294 |
| Fig. 103. | Sub-state ERROR_TREATMENT  | 295 |
| Fig. 104. | Minimal NMT boot-up process  | 300 |
| Fig. 105. | Boot procedure example for a single CN   | 301 |
| Fig. 106. | POWERLINK router, black box model  | 308 |
| Fig. 107. | Possible communication relations via a POWERLINK router                        | 309 |
| Fig. 108. | POWERLINK router reference model   | 309 |
| Fig. 109. | Symmetrical n-to-n NAT   | 312 |
| Fig. 110. | NAT architecture   | 312 |
| Fig. 111. | Integration of NAT in the POWERLINK router                                     | 313 |
| Fig. 112. | Filter tables of the packet filter   | 314 |
| Fig. 113. | POWERLINK router type 2  | 329 |
| Fig. 114. | ERROR LED state machine  | 331 |

## Pre. 7 Definitions and Abbreviations

### Pre 7.1 Definitions

|                             |   |
|-----------------------------|---|
| Ageing                      | <i>Ageing</i> is a common mechanism to maintain (cache) tables. Entries which are not used or refreshed are removed after a specified time.   |
| Application Process         | The Application Process is the task on the Application Layer  |
| Async-only CN               | An <i>Async-only CN</i> is operated in a way, that it isn't accessed cyclically in the isochronous slot by the MN. It is polled during the asynchronous period by a StatusRequest message.  |
| Asynchronous Data           | Data in a POWERLINK network which is not time critical. Within the POWERLINK cycle there is a specific period reserved for <i>Asynchronous Data</i> which is shared by all nodes. Each node connected to the network can send asynchronous data by requesting it to the Managing Node. The Managing Node keeps a list of all asynchronous data requests and will subsequently grant the network access to one node after the other. |
| Asynchronous Period         | The <i>Asynchronous Period</i> is the second part of the POWERLINK cycle, starting with a <i>Start of Asynchronous</i> (SoA) frame.   |
| Asynchronous Scheduling     | The MN's asynchronous scheduler decides when a requested asynchronous data transfer will happen.  |
| Basic Ethernet Mode         | Basic Ethernet Mode provides the Legacy Ethernet communication.   |
| CANopen                     | CANopen is a network technology optimized for the usage in industrial control environments, in machine internal networks and in embedded systems (any control unit deeply "embedded" in a device with electronics). The lower-layer implementation of CANopen is based upon CAN (Controller Area Network).  |
| Continuous                  | <i>Continuous</i> is a POWERLINK communication class where isochronous communication takes place every cycle (the opposite to <i>multiplexed</i> ).   |
| Controlled Node (CN)        | Node in a POWERLINK network without the ability to manage the SCNM mechanism.   |
| Cycle State Machine         | The <i>Cycle State Machine</i> controls the POWERLINK cycle on the Data Link Layer and is itself controlled by the NMT state machine defining the current operating mode.   |
| Cycle Time                  | The time between two consecutive <i>Start of Cycle</i> (SoC) frames – i.e. repeating – process. The <i>Cycle Time</i> includes the time for data transmission and some idle time before the beginning of the next cycle.  |
| Deterministic Communication | Describes a communication process whose timing behaviour can be predicted exactly. I.e. the time when a message reaches the recipient is predictable.   |
| Device Configuration File   | The configuration parameters of a specific device are stored in the <i>Device Configuration File</i> (XDC).   |
| Device Description File     | All device dependent information is stored in the <i>Device Description File</i> (XDD) of each device.  |
| Destination NAT (D-NAT)     | D-NAT (Destination- Network Address Translation) changes the destination address of the IP / ICMP packet.   |
| Domain                      | In the context of CANopen: A <i>Domain</i> is a data object of arbitrary type and length which can be transferred over a POWERLINK network.<br>In the context of internet protocols: A <i>Domain</i> is a part of the internet name space which is supported by the Domain Name System (DNS).   |
| Ethernet POWERLINK (EPL)    | An extension to <i>Legacy Ethernet</i> on layer 2, to exchange data under hard real-time constraints. It was developed for deterministic data exchange, short cycle time and isochronous operation in industrial automation.  |

|                            |   |
|----------------------------|---|
| IdentRequest               | <i>IdentRequests</i> are POWERLINK frames sent by the MN in order to identify active CNs waiting to be included into the network.   |
| IdentResponse              | The <i>IdentResponse</i> is a special form of an ASnd frame in response to an <i>IdentRequest</i> .   |
| Idle Period                | The <i>Idle Period</i> is time interval remaining between the completed asynchronous period and the beginning of the next cycle.  |
| IEEE 1588                  | This standard defines a protocol enabling synchronisation of clocks in distributed networked devices (e.g. connected via Ethernet).   |
| Isochronous                | Pertains to processes that require timing coordination to be successful. Isochronous data transfer ensures that data flows continuously and at a steady rate in close timing with the ability of connected devices.   |
| Isochronous Data           | Data in a POWERLINK network which is to be transmitted every cycle (or every nth cycle in case of multiplexed isochronous data).  |
| Isochronous Period         | The <i>Isochronous Period</i> of a POWERLINK cycle offers deterministic operation, i.e. it is reserved for the exchange of ( <i>continuous</i> or <i>multiplexed</i> ) isochronous data.                              |
| Legacy Ethernet            | Ethernet as standardised in IEEE 802.3 (non-deterministic operation in non-time-critical environments).   |
| Managing Node (MN)         | A node capable to manage the SCNM mechanism in a POWERLINK network.   |
| Media Access Control (MAC) | One of the sub-layers of the Data Link Layer in the POWERLINK reference model that controls who gets access to the medium to send a message.  |
| Multiplexed                | <i>Multiplexed</i> is a POWERLINK communication class where cyclic communication takes place in such a way that m nodes are served in s cycles (the opposite to <i>continuous</i> ).                                  |
| Multiplexed CN             | A node which is allowed to send isochronous data every n <sup>th</sup> cycle.   |
| Multiplexed Timeslot       | A slot destined to carry multiplexed isochronous data, i.e. the timeslot is shared among multiple nodes.  |
| NetTime                    | The MN's clock time is distributed to all CNs within the SoC frame.   |
| Network Management (NMT)   | <i>Network Management</i> functions and services in the POWERLINK model. It performs initialisation, configuration and error handling in a POWERLINK network.   |
| NMT State Machine          | The state machine controlling the overall operating mode and status of a POWERLINK node.  |
| Object Directory           | The repository of all data objects accessible over POWERLINK communications.  |
| PollRequest                | A PollRequest is a frame, which is used in the isochronous part of the cyclic communication. The MN request with this frame the CN to send its data.  |
| PollResponse               | A PollResponse is a frame, which is used in the isochronous part of the cyclic communication. The CN responses with this frame to a PollRequest frame from an MN.   |
| POWERLINK Command Layer    | The <i>POWERLINK Command Layer</i> defines commands to access parameters of the object dictionary. This layer is on top of the <i>Sequence Layer</i> and distinguishes between an expedited and a segmented transfer. |
| POWERLINK Cycle            | Data exchange within a POWERLINK network is structured in fix intervals, called cycles. The cycle is subdivided into the isochronous and the asynchronous period and is organized by the MN.                          |
| POWERLINK Mode             | The <i>POWERLINK Mode</i> includes all NMT states in which POWERLINK cycles are run.  |
| POWERLINK Node ID          | Each POWERLINK node (MN, CN and Router) is addressed by an 8 bit <i>POWERLINK Node ID</i> on the POWERLINK layer. This ID has only local significance (i.e. it is unique within a POWERLINK segment).                 |

|  |  |
|--|--|
| Precision Time Protocol (PTP)                | IEEE 1588, Standard for a Precision Clock Synchronisation Protocol for Networked Measurement and Control Systems   |
| Process Data Object (PDO)                    | Object for isochronous data exchange between POWERLINK nodes.  |
| Reserved                                     | Reserved bits shall be set 0 by the sender. The receiver shall not interpret such bits. It is not allowed to use reserved bits.<br>Their use is reserved for further development or by extensions of this specification.   |
| Router Type 1                                | A Type 1 POWERLINK Router is a coupling element in a network that allows IP communication between a POWERLINK segment and any other datalink layer protocol carrying IP e.g. legacy Ethernet, POWERLINK etc. It is usually a separate network element acting as Controlled Node within the POWERLINK segment.  |
| Router Type 2                                | A Type 2 POWERLINK Router is a router between a POWERLINK segment and a CANopen network.   |
| Service Data Object (SDO)                    | Peer to peer communication with access to the object dictionary of a device.   |
| Sequence Layer                               | The POWERLINK Sequence Layer provides the service of a reliable bidirectional connection that guarantees that no messages are lost or duplicated and that all messages arrive in the correct order.  |
| Slot Communication Network Management (SCNM) | In a POWERLINK network, the managing node allocates data transfer time for data from each node in a cyclic manner within a guaranteed cycle time. Within each cycle there are slots for <i>Isochronous Data</i> , as well as for <i>Asynchronous Data</i> for ad-hoc communication. The SCNM mechanism ensures that there are no collisions during physical network access of any of the networked nodes thus providing deterministic communication via <i>Legacy Ethernet</i> . |
| Source NAT (S-NAT)                           | S-NAT (Source - Network Address Translation) changes the source address of the IP / ICMP packet.   |
| StatusRequest                                | A StatusRequest frame is a special SoA frame used to poll the status of a node.  |
| StatusResponse                               | A StatusResponse frame is transmitted by a CN upon assignment of the asynchronous slot via the StatusRequest in the SoA frame.   |

## Pre 7.2 Abbreviations

|      |  |
|------|--|
| ACL  | Access Control List                              |
| ARP  | Address Resolution Protocol                      |
| ASnd | Asynchronous Send (POWERLINK frame type)         |
| CAN  | Controller Area Network                          |
| CiA  | CAN in Automation                                |
| CN   | POWERLINK Controlled Node                        |
| DCF  | Device Configuration File                        |
| EIA  | Electronic Industries Association                |
| EMC  | Electro Magnetic Compatibility                   |
| EPL  | Ethernet POWERLINK                               |
| EPSG | Ethernet POWERLINK Standardisation Group         |
| ICMP | Internet Control Message Protocol                |
| ID   | Identifier                                       |
| IEC  | International Electrotechnical Commission        |
| IEEE | Institute of Electrical and Electronic Engineers |
| IP   | Internet Protocol                                |
| MAC  | Media Access Control                             |
| MIB  | Management Information Base                      |

|      |   |
|------|---|
| MN   | POWERLINK Managing Node                         |
| MS   | Multiplexed Slot (flag in POWERLINK frame)      |
| MSS  | Maximum Segment Size                            |
| MTU  | Maximum Transmission Unit                       |
| NAT  | Network Address Translation                     |
| NIL  | Not in List (Basic Data Type)                   |
| NMT  | Network Management                              |
| PDO  | Process Data Object                             |
| PR   | Priority (bit field in POWERLINK frame)         |
| PReq | PollRequest (POWERLINK frame type)              |
| PRes | PollResponse (POWERLINK frame type)             |
| PS   | Prescaled Slot (flag in POWERLINK frame)        |
| PTP  | Precision Time Protocol                         |
| RD   | Ready (flag in POWERLINK frame)                 |
| RFC  | Requests for Comments                           |
| RPDO | Receive Process Data Object                     |
| RS   | Request to Send (flag in POWERLINK frame)       |
| EA   | Exception Acknowledge (flag in POWERLINK frame) |
| SCNM | Slot Communication Network Management           |
| SDO  | Service Data Object                             |
| EN   | Exception New (flag in POWERLINK frame)         |
| SNMP | Simple Network Management Protocol              |
| SoA  | Start of Asynchronous (POWERLINK frame type)    |
| Soc  | Start of Cycle (POWERLINK frame type)           |
| TCP  | Transmission Control Protocol                   |
| TIA  | Telecommunications Industry Association         |
| TPDO | Transmit Process Data Object                    |
| UDP  | User Datagram Protocol                          |
| VPN  | Virtual Private Network                         |
| XDC  | XML device configuration file                   |
| XDD  | XML device description file                     |

## Pre. 8 References

- [1] EPSG Draft Standard 302-A (EPSC DS 302-A), Ethernet POWERLINK, Part A: High Availability, Version 1.1.0
- [2] EPSG Draft Standard 302-B (EPSC DS 302-B), Ethernet POWERLINK, Part B: Multiple-ASnd, Version 1.1.0
- [3] EPSG Draft Standard 302-C (EPSC DS 302-C), Ethernet POWERLINK, Part C: PollResponse Chaining, Version 1.1.0
- [4] EPSG Draft Standard 302-D (EPSC WDP 302-D), Ethernet POWERLINK, Part D: Multiple PReq/PRes, Version 1.0.0
- [5] EPSG Draft Standard 302-E (EPSC WDP 302-E), Ethernet POWERLINK, Part E: Dynamic Node Allocation, Version 1.1.0
- [6] IEC 61918 Industrial communication networks – Installation of communication networks in industrial premises
- [7] IEC 61784-5-13 Industrial communication networks – Profiles – Part 5-13: Installation of fieldbuses – Installation profiles for CPF 13

# 1 Introduction

Ethernet POWERLINK is a communication profile for Real-Time Ethernet (RTE). It extends Ethernet according to the IEEE 802.3 standard with mechanisms to transfer data with predictable timing and precise synchronisation. The communication profile meets timing demands typical for high-performance automation and motion applications. It does not change basic principles of the Fast Ethernet Standard IEEE 802.3 but extends it towards RTE. Thus it is possible to leverage and continue to use any standard Ethernet silicon, infrastructure component or test and measurement equipment like a network analyzer.

## 1.1 Slot Communication Network Management

POWERLINK provides mechanisms to achieve the following:

1. Transmit time-critical data in precise isochronous cycles. Data exchange is based on a publish/subscribe relationship. Isochronous data communication can be used for exchanging position data of motion applications of the automation industry.
2. Synchronise networked nodes with high accuracy.
3. Transmit less time-critical data asynchronously on request. Asynchronous data communication can be used to transfer IP-based protocols like TCP or UDP and higher layer protocols such as HTTP, FTP,....

POWERLINK manages the network traffic in a way that there are dedicated time-slots for isochronous and asynchronous data. It takes care that always only one networked device gains access to the network media. Thus transmission of isochronous and asynchronous data will never interfere and precise communication timing is guaranteed. The mechanism is called Slot Communication Network Management (SCNM). SCNM is managed by one particular networked device – the Managing Node (MN) – which includes the MN functionality. All other nodes are called Controlled Nodes (CN).

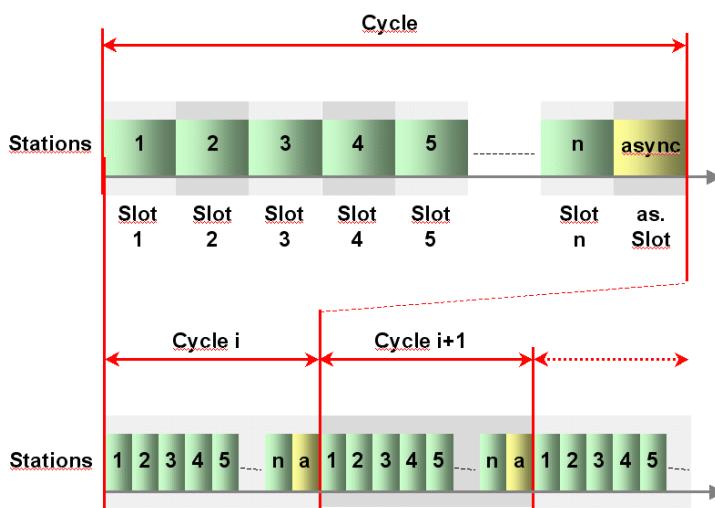


Fig. 1. Slot Communication Network Management (SCNM)

## 1.2 POWERLINK key features

POWERLINK provides the following key features:

- Ease-of-Use to be handled by typical automation engineers without indepth Ethernet network knowledge.
- Up to 240 networked real-time devices in one network segment
- Deterministic Communication Guaranteed
  - Down to 100 µs cycle times
  - Ultra-low jitter (down to <1µs) for precise synchronisation of networked devices
- Standard Compliant

- IEEE 802.3u Fast Ethernet
- IP based protocols supported (TCP, UDP,...)
- Integration with CANopen Profiles EN 50325-4 for device interoperability
- Implementation based on standard Ethernet Chips - No special ASICs necessary
- Direct peer-to-peer communication of all Nodes (Publish/subscribe)
- Hot Plugging
- Seamless IT-Integration – Routing of IP protocols

POWERLINK is based on the ISO/OSI layer model and supports Client/Server and Producer/Consumer communications relationships.

The Ethernet POWERLINK Standardisation Group (EPSG) is working closely with the CiA (CAN in Automation) organisation to integrate CANopen with POWERLINK. CANopen standards define widely deployed communication profiles, device profiles and application profiles. These profiles are in use millions of times all over the world. Integration of POWERLINK with CANopen combines profiles, high performance data exchange and open transparent communication with TCP/UDP/IP protocols.

The POWERLINK communication profile is based on CANopen communication profiles DS301 and DS302. Based on this communication profile, the multitude of CANopen device profiles can be used in a POWERLINK environment without changes.

A main focus of POWERLINK is ease of use. Ethernet technology can be quite complex and confusing for machine and plant manufacturers which are not necessarily networking experts. The following features have been implemented:

- Easy wiring, flexible topologies (line structures, tree structures or star structures). The network is adapting to the needs of the machine.
- Utilization of well known industrial infrastructure components
- Simple address assignment by switch is possible
- Easy replacement of devices in case of failure
- Straightforward network diagnostics
- Basic security features
- Simple engineering separated from end user IT infrastructure
- Easy integration of RTE network with IT infrastructure

## 1.3 Integration

The advantages listed before result from protecting the POWERLINK RTE network segment from regular office and factory networks. This matches typical machine and plant concepts. Hard real time requirements are met within the machine with POWERLINK. Full transparency to the factory network and above is provided, yet it is taken care of protection against hacker attacks on machine level. Modification efforts through machine integration into existing IT infrastructures are minimized. To achieve this POWERLINK provides a private Class-C IP segment solution with fixed IP addresses. A router establishes the connection to factory floor networks or company networks. NAT mechanisms allow the assignment of any IP address to RTE networked nodes.

RTE based on POWERLINK is ideal to support modern modular machine concepts. Producer/Consumer and Client/Server communication relationships, enable centralized master/slave as well as decentral multimaster structures.

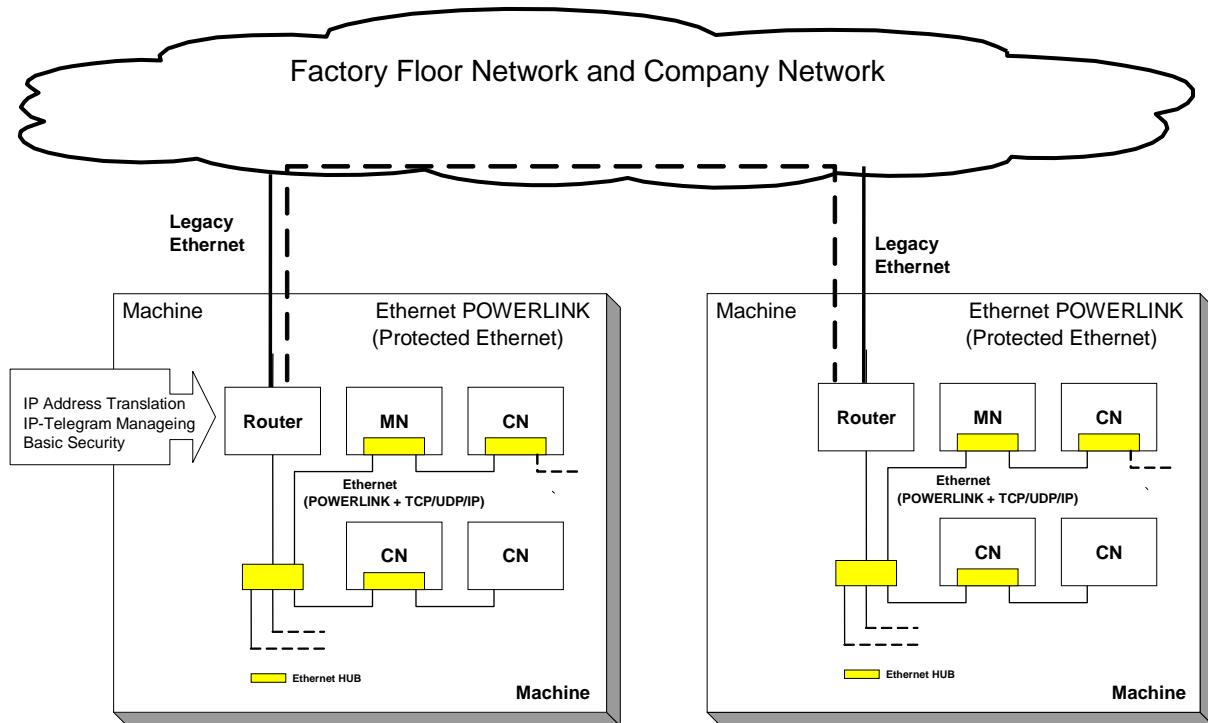


Fig. 2. Integration POWERLINK based machines into the IT infrastructure of end customer

## 1.4 Modular Machines

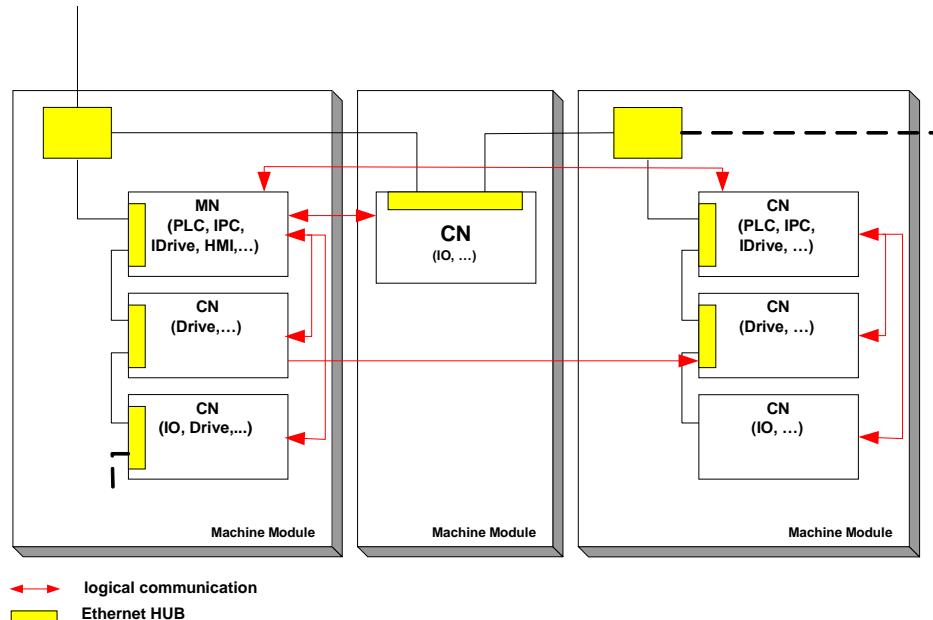


Fig. 3. Typical centralized and decentralized controller structures

A machine concept with autonomous machine modules is illustrated in Fig. 3. Every machine module can be designed separately with its own internal communication relationships. The assembling of the machine can be done in a flexible way by adding additional direct communication relationships between machine modules.

## 2 Modelling

POWERLINK-based networks use the following reference model, device model, and communication model.

### 2.1 Reference Model

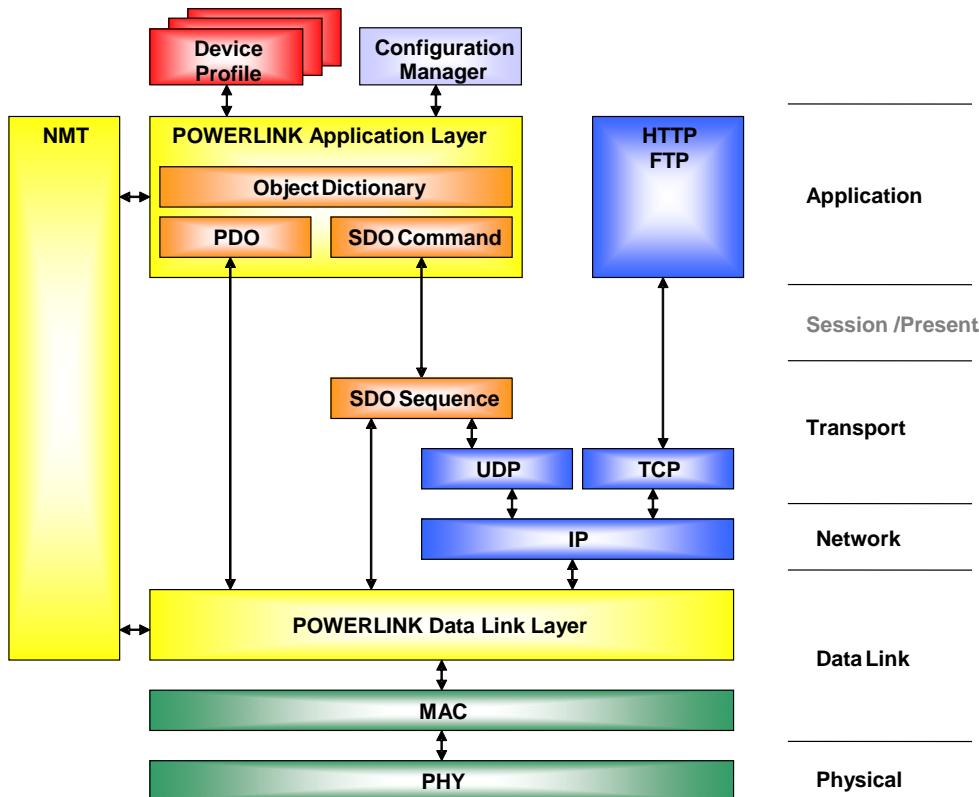


Fig. 4. Reference model

The communication concept can be described with reference to the ISO-OSI Reference Model (right-hand side of Fig. 4).

#### 2.1.1 Application Layer

The Application Layer comprises a concept to configure and communicate real-time-data as well as the mechanisms for synchronisation between devices. The functionality the application layer offers to an application is logically divided over different *service objects* (see SDO) in the application layer. A service object offers a specific functionality and all related services.

Applications interact by invoking services of a service object in the application layer. To realize these services, the service object exchanges data via the Network with (a) peer service object(s) via a protocol. This protocol is described in the *Protocol Specification* of that service object.

##### 2.1.1.1 Service Primitives

Service primitives are the means by which the application and the application layer interact. There are four different primitives:

- a *request* is issued by the application to the application layer to request a service
- an *indication* is issued by the application layer to the application to report an internal event detected by the application layer or indicate that a service is requested

- a *response* is issued by the application to the application layer to respond to a previous received indication
- a *confirmation* is issued by the application layer to the application to report the result of a previously issued request.

### 2.1.1.2 Application Layer Service Types

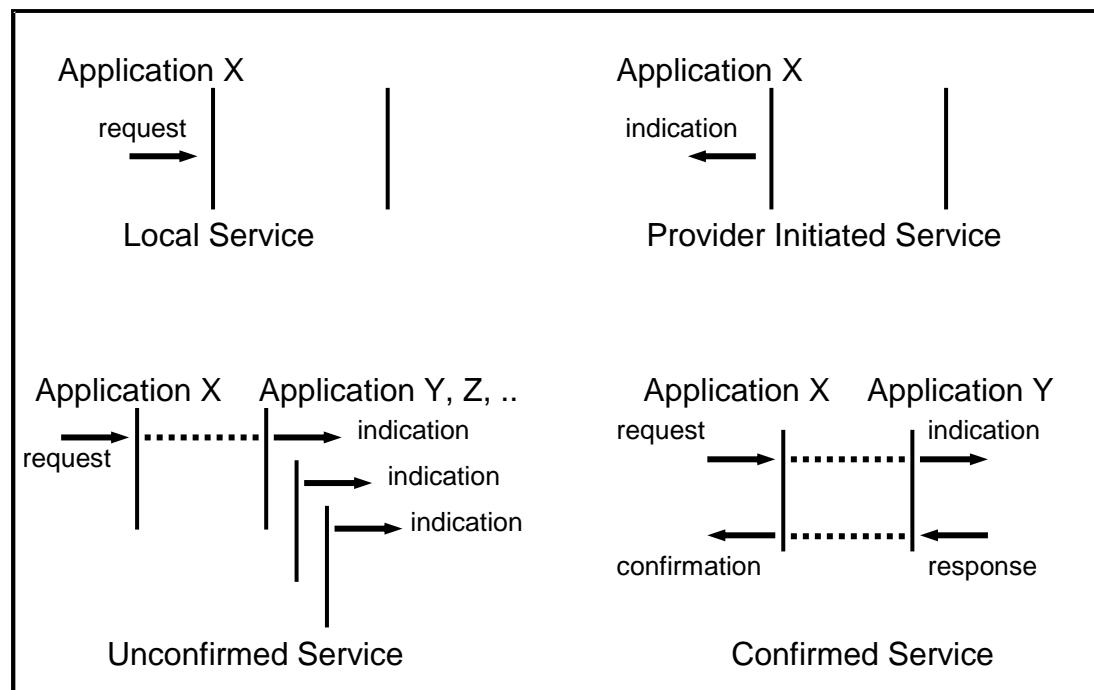


Fig. 5. Service types

A *service type* defines the primitives that are exchanged between the application layer and the co-operating applications for a particular service of a service object.

- A *Local Service* involves only the local service object. The application issues a request to its local service object that executes the requested service without communicating with (a) peer service object(s).
- An *Unconfirmed Service* involves one or more peer service objects. The application issues a request to its local service object. This request is transferred to the peer service object(s) that each pass it to their application as an indication. The result is not confirmed back.
- A *Confirmed Service* can involve only one peer service object. The application issues a request to its local service object. This request is transferred to the peer service object that passes it to the other application as an indication. The other application issues a response that is transferred to the originating service object that passes it as a confirmation to the requesting application.
- A *Provider Initiated service* involves only the local service object. The service object (being the service provider) detects an event not solicited by a requested service. This event is then indicated to the application.

Unconfirmed and confirmed services are collectively called *Remote Services*.

## 2.2 Device Model

### 2.2.1 General

A device is structured as follows (see Fig. 6):

- Communication – This function unit provides the communication objects and the appropriate functionality to transport data items via the underlying network structure.

- Object Dictionary – The Object Dictionary is a collection of all the data items that have an influence on the behaviour of the application objects, the communication objects and the state machine used on this device.
- Application – The application comprises the functionality of the device with respect to the interaction with the process environment.

Thus the Object Dictionary serves as an interface between the communication and the application. The complete description of a device's application with respect to the data items in the Object Dictionary is called the *device profile*.

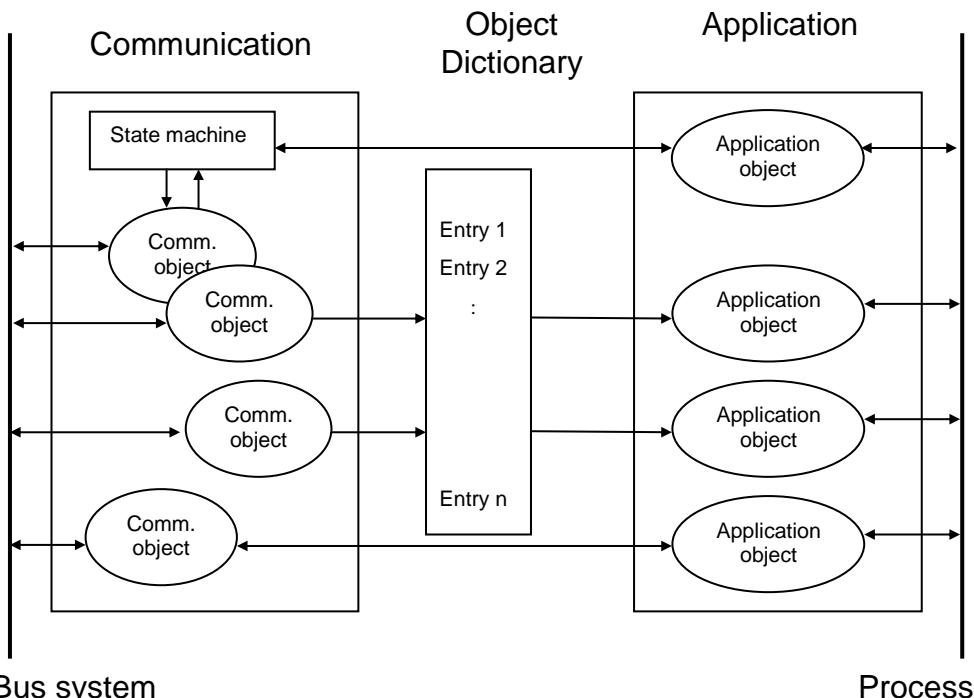


Fig. 6. Device model

## 2.2.2 The Object Dictionary

The most important part of a device profile is the Object Dictionary. The Object Dictionary is essentially a grouping of objects accessible via the network in an ordered, pre-defined fashion. Each object within the dictionary is addressed using a 16-bit index.

The overall layout of the standard Object Dictionary is shown by Tab. 1. This layout closely conforms to other industrial serial bus system concepts.

The Object Dictionary may contain a maximum of 65536 entries which are addressed through a 16-bit index.

The Static Data Types at indices 0001<sub>h</sub> through 001F<sub>h</sub> contain type definitions for standard data types like BOOLEAN, INTEGER, floating point, string, etc. These entries are included for reference only; they cannot be read or written.

| <b>Index</b>                           | <b>Object</b>                              |
|--|--|
| 0000 <sub>h</sub>                      | not used                                   |
| 0001 <sub>h</sub> .. 001F <sub>h</sub> | Static Data Types                          |
| 0020 <sub>h</sub> .. 003F <sub>h</sub> | Complex Data Types                         |
| 0040 <sub>h</sub> .. 005F <sub>h</sub> | Manufacturer Specific Complex Data Types   |
| 0060 <sub>h</sub> .. 007F <sub>h</sub> | Device Profile Specific Static Data Types  |
| 0080 <sub>h</sub> .. 009F <sub>h</sub> | Device Profile Specific Complex Data Types |
| 00A0 <sub>h</sub> .. 03FF <sub>h</sub> | Reserved for further use                   |
| 0400 <sub>h</sub> – 041F <sub>h</sub>  | POWERLINK Specific Static Data Types       |
| 0420 <sub>h</sub> – 04FF <sub>h</sub>  | POWERLINK Specific Complex Data Types      |
| 0500 <sub>h</sub> .. 0FFF <sub>h</sub> | Reserved for further use                   |
| 1000 <sub>h</sub> .. 1FFF <sub>h</sub> | Communication Profile Area                 |
| 2000 <sub>h</sub> .. 5FFF <sub>h</sub> | Manufacturer Specific Profile Area         |
| 6000 <sub>h</sub> .. 9FFF <sub>h</sub> | Standardised Device Profile Area           |
| A000 <sub>h</sub> .. BFFF <sub>h</sub> | Standardised Interface Profile Area        |
| C000 <sub>h</sub> .. FFFF <sub>h</sub> | Reserved for further use                   |

Tab. 1 Object dictionary structure

Complex Data Types at indices 0020<sub>h</sub> through 003F<sub>h</sub> are pre-defined structures that are composed of standard data types and are common to all devices.

Manufacturer Specific Complex Data Types at indices 0040<sub>h</sub> through 005F<sub>h</sub> are structures composed of standard data types but are specific to a particular device.

Device Profiles may define additional data types specific to their device type. The static data types defined by the device profile are listed at indices 0060<sub>h</sub> - 007F<sub>h</sub>, the complex data types at indices 0080<sub>h</sub> - 009F<sub>h</sub>.

A device may optionally provide the structure of the supported complex data types (indices 0020<sub>h</sub> - 005F<sub>h</sub> and 0080<sub>h</sub> - 009F<sub>h</sub>) at read access to the corresponding index. Sub-index 0 provides the number of entries at this index, and the following sub-indices contain the data type encoded as UNSIGNED16 according to 6.1.4.4.

POWERLINK Specific Static Data Types shall be described at indices 0400<sub>h</sub> – 041F<sub>h</sub>. These entries are included for reference only; they cannot be read or written. POWERLINK Specific Complex Data Types shall be described at indices 0420<sub>h</sub> – 04FF<sub>h</sub>

The Communication Profile Area at indices 1000<sub>h</sub> through 1FFF<sub>h</sub> contains the communication specific parameters for the POWERLINK network. These entries are common to all devices.

The standardised device profile area at indices 6000<sub>h</sub> through 9FFF<sub>h</sub> contains all data objects common to a class of devices that can be read or written via the network. The device profiles may use entries from 6000<sub>h</sub> to 9FFF<sub>h</sub> to describe the device parameters and the device functionality. Within this range up to 8 different devices can be described. In such a case the devices are denominated Multiple Device Modules. Multiple Device Modules are composed of up to 8 device profile segments. In this way it is possible to build devices with multiple functionality. The different device profile entries are indexed at increments of 800<sub>h</sub>.

For Multiple Device Modules the object range 6000<sub>h</sub> to 9FFF<sub>h</sub> is sub-divided as follows:

- 6000<sub>h</sub> to 67FF<sub>h</sub> device 0
- 6800<sub>h</sub> to 6FFF<sub>h</sub> device 1
- 7000<sub>h</sub> to 77FF<sub>h</sub> device 2
- 7800<sub>h</sub> to 7FFF<sub>h</sub> device 3
- 8000<sub>h</sub> to 87FF<sub>h</sub> device 4
- 8800<sub>h</sub> to 8FFF<sub>h</sub> device 5
- 9000<sub>h</sub> to 97FF<sub>h</sub> device 6
- 9800<sub>h</sub> to 9FFF<sub>h</sub> device 7

The Object Dictionary concept caters for optional device features: a manufacturer does not have to provide certain extended functionality on his devices but if he wishes to do so he must do it in a pre-defined fashion.

Space is left in the Object Dictionary at indices 2000<sub>h</sub> through 5FFF<sub>h</sub> for truly manufacturer-specific functionality.

### 2.2.2.1 Index and Sub-Index Usage

A 16-bit index is used to address all entries within the Object Dictionary. In the case of a simple variable the index references the value of this variable directly. In the case of records and arrays, however, the index addresses the whole data structure.

To allow individual elements of structures of data to be accessed via the network a sub-index is defined. For single Object Dictionary entries such as an UNSIGNED8, BOOLEAN, INTEGER32 etc. the value for the sub-index is always zero. For complex Object Dictionary entries such as arrays or records with multiple data fields the sub-index references fields within a data-structure pointed to by the main index. The fields accessed by the sub-index can be of differing data types.

## 2.3 Communication Model

The communication model specifies the different communication objects and services and the available modes of frame transmission triggering.

The communication model only specifies the POWERLINK-specific communication objects of the POWERLINK Mode and Basic Ethernet Mode (4.2 resp. 4.3). The mechanism for Legacy Ethernet communication in Basic Ethernet mode is not within the scope of this specification.

The communication model supports the transmission of isochronous and asynchronous frames. Isochronous frames are supported in POWERLINK Mode only, asynchronous frames in POWERLINK Mode and Basic Ethernet Mode.

By means of isochronous frame transmission a network wide coordinated data acquisition and actuation is possible. The isochronous transmission of frames is supported by the POWERLINK Mode cycle structure. The system is synchronised by SoC frames. Asynchronous frames may be transmitted in the asynchronous slot of POWERLINK Mode cycle upon transmission grant by the POWERLINK MN, or at any time in Basic Ethernet Mode.

With respect to their functionality, three types of communication relationships are distinguished

- Master/Slave relationship (Fig. 7 and Fig. 8)
- Client/Server relationship (Fig. 9)
- Producer/Consumer relationship (Fig. 10 and Fig. 11)

### 2.3.1 Master/Slave relationship

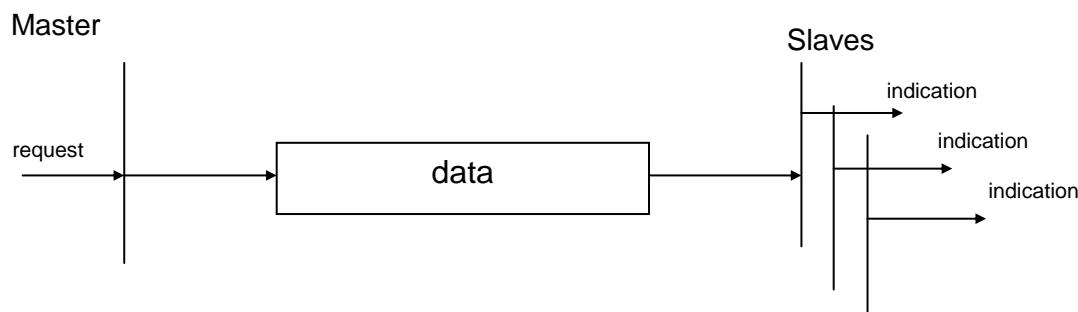


Fig. 7. Unconfirmed master slave communication

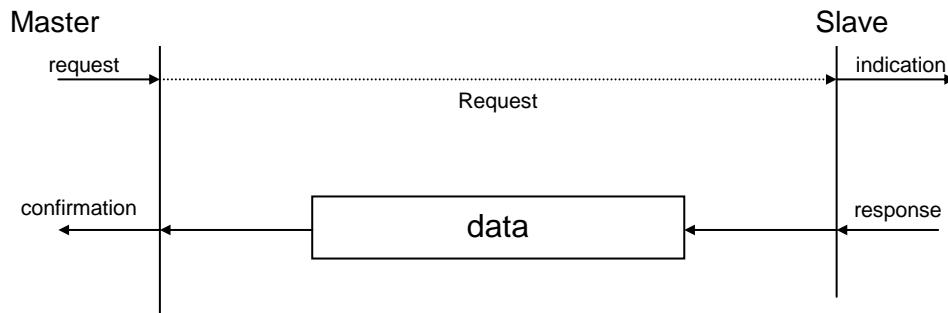


Fig. 8. Confirmed master slave communication

At any time there is exactly one device in the network serving as a master for a specific functionality. All other devices in the network are considered as slaves. The master issues a request and the addressed slave(s) respond(s) if the protocol requires this behaviour.

### 2.3.2 Client/Server relationship

This is a relationship between a single client and a single server. A client issues a request (upload/download) thus triggering the server to perform a certain task. After finishing the task the server answers the request.

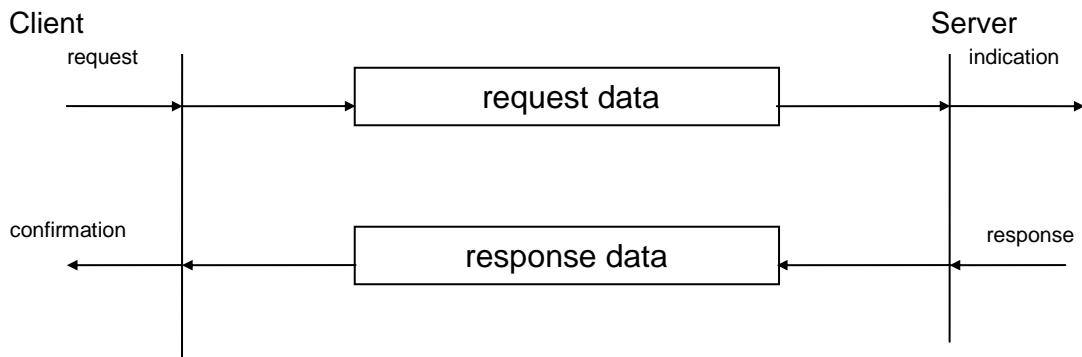


Fig. 9. Client/Server communication

### 2.3.3 Producer/Consumer relationship - Push/Pull model

The producer/consumer relationship model involves a producer and zero or more consumer(s). The push model is characterized by an unconfirmed service requested by the producer. The pull model is characterized by a confirmed service requested by the consumer.

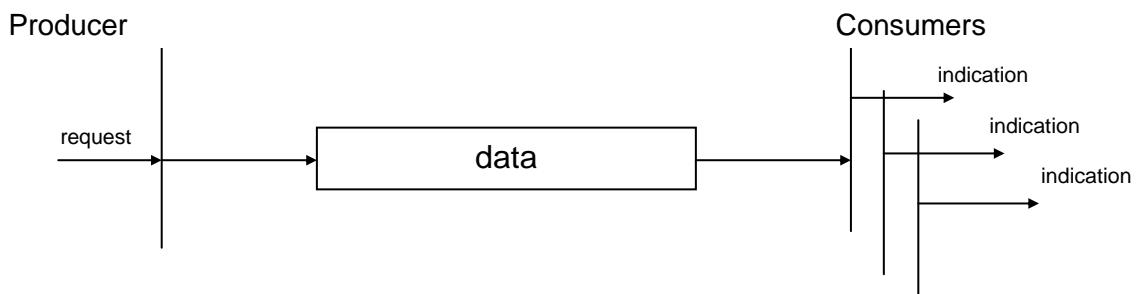


Fig. 10. Push model

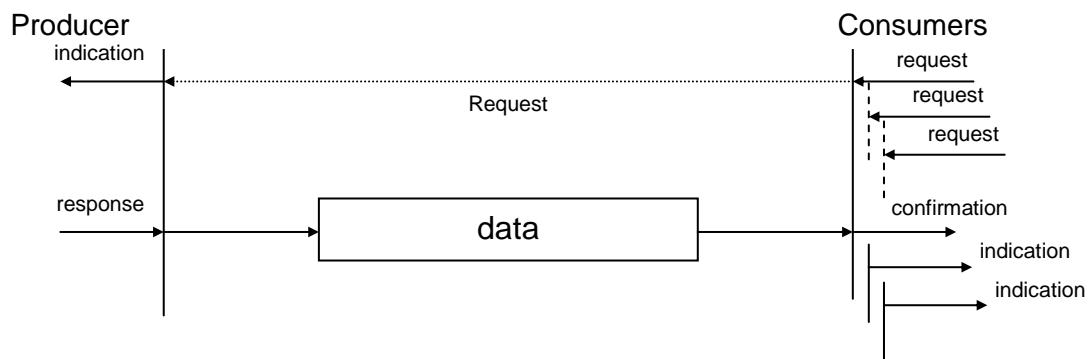


Fig. 11. Pull model

### 2.3.4 Superimposing of Communication Relationships

POWERLINK collects more than one function into one frame (refer 4.6). It is therefore not usually possible to apply a single communication relationship to the complete frame, but only to particulars services inside the frame.

The PollResponse frame for example (refer 4.6.1.1.4) transmitted by the CN includes several services:

- Transmission of the current NMT status of the CN is the response part of a confirmed master/slave relationship triggered by the MN.
- Request of the asynchronous slot is the request part of a client/server relationship.
- Transmission of PDO data occurs in conformance to a push model Producer/Consumer relationship.

## 3 Physical Layer

POWERLINK is a protocol residing on top of the standard IEEE 802.3 MAC layer. The physical layer is 100BASE-X (copper and fiber, see IEEE 802.3). Half-Duplex transmission mode shall be used.

Autonegotiation is not recommended.

POWERLINK uses Ethernet as it is, without any modifications. Hence any advancement in Ethernet Technology can be exploited (e.g. Gigabit Ethernet).

### 3.1 Topology

#### 3.1.1 Hubs

To fit POWERLINK jitter requirements it is recommended to use hubs to build a POWERLINK network. Class 2 Repeaters shall be used in this case.

Hubs have the advantage of reduced path delay value (indicated by D\_PHY\_HubDelay\_U32) and have small frame jitter (indicated by D\_PHY\_HubJitter\_U32).

Hubs may be integrated in the POWERLINK interface cards.

Hub integration shall be indicated by D\_PHY\_HubIntegrated\_BOOL. The number of externally accessible POWERLINK ports provided by a device shall be indicated by D\_PHY\_ExtEPLPorts\_U8.

#### 3.1.2 Switches

Switches may be used to build a POWERLINK network. The additional latency and jitter of switches has to be considered for system configuration.

It has to be considered that any POWERLINK network constructed with anything but Class 2 Repeater Devices does not conform to the POWERLINK standard as defined in this document.

### 3.2 Network Guidelines

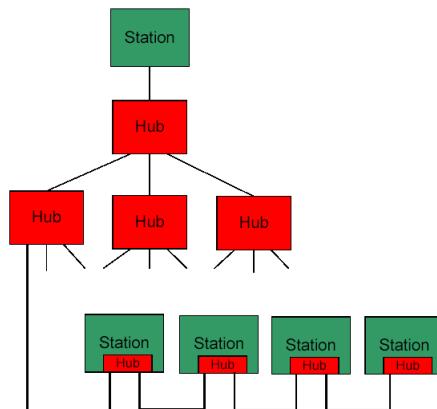


Fig. 12. Star topology and line topology

POWERLINK does not cause collisions. This is why the most extreme topology guideline of the IEEE standard (5120 ns maximum round trip signal runtime) does not apply.

Due to this leniency in the topology, line structures that are required in applications in the field are made possible. Nodes may use integrated hubs, further simplifying construction in the field. A mixed tree and line structure is available when a large number of nodes are being used.

Fiber optic transducers may be used. However, they should be tested to establish whether they cause more jitter and latency than normal hubs. When designing the network infrastructure some timing constraints shall be considered. The MN uses a timeout after sending a PollRequest Frame to detect transmission errors and node failures. The round trip latency between the MN and a CN shall not exceed the timeout value. The timeout value can be set for every single node.

### 3.2.1 Jitter

Every hub level introduces additional Jitter (equal or below 70 ns). Only the number of hub levels between MN and most distanced CN is relevant. If the MN is located in the center of line or ar star topology, the number of hub level between the most distanced CNs is irrelevant for synchronisation jitter.

## 3.3 Ports and Connectors

To connect POWERLINK devices one of two types of connectors shall be used:

4. RJ-45: for light duty environments.
5. M12: for heavy duty environments.

Both types may be mixed on the same cable.

For further information please refer to IEC 61918 and IEC 61784-5-13.

### 3.3.1 RJ-45

Pin assignment as defined by EIA/TIA T568B.

The following is provided for convenience; please refer to the corresponding International Standards.

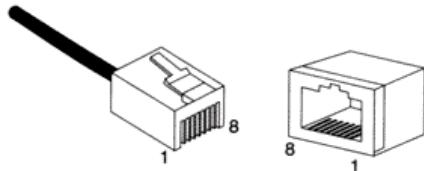


Fig. 13. RJ45 pin assignment (left: connector, right: port)

The pictures shows an RJ45 connector on the cable and a port (on the device or Hub).

The pin assignment on each node shall be that of a hub/switch port. Therefore the port pins are assigned as follows:

| Pin | Wire color code | Assignment 100BASE-TX |
|-----|-----------------|-----------------------|
| 1   | WHT/ORG         | Rx+                   |
| 2   | ORG             | Rx-                   |
| 3   | WHT/GRN         | Tx+                   |
| 4   | BLU             |                       |
| 5   | WHT/BLU         |                       |
| 6   | GRN             | Tx-                   |
| 7   | WHT/BRN         |                       |
| 8   | BRN             |                       |

Tab. 2 Pin assignment RJ45 port

### 3.3.2 M12

For IP67 requirements. 4 pin D-coded as recommended in IEC 61076-2-101.

Male side is fitted on the cable, female on the device or hub.

The following is provided for convinience; please refer to the corresponding International Standard.

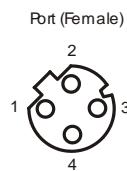


Fig. 14. IP67 port pin assignment.

| Pin | Wire color code | Assignment 100BASE-TX |
|-----|-----------------|-----------------------|
| 1   | BLU/YEL         | Tx+                   |
| 2   | YEL/WHT         | Rx+                   |
| 3   | WHT/ORG         | Tx-                   |
| 4   | ORG/BLU         | Rx-                   |

Tab. 3 Pin assignment IP67 port

### 3.3.3 Crossover Pin Assignment

The pin assignment shall be that of a crossover cable.

Therefore all devices can be interconnected by one type of cable. The pin assignment of a crossover cable is defined as:

- Tx+ to Rx+
- Tx- to Rx-
- Rx+ to Tx+
- Rx- to Tx-

#### 3.3.3.1 RJ45 to RJ45

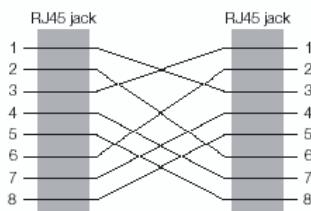


Fig. 15. recommended RJ45 to RJ45 pin assignment

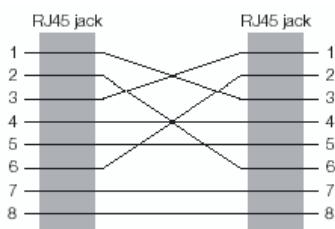


Fig. 16. not recommended RJ45 to RJ45 pin assignment

#### 3.3.3.2 M12 to M12

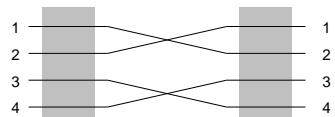


Fig. 17. M12 to M12 pin assignment

### 3.3.3.3 M12 to RJ45

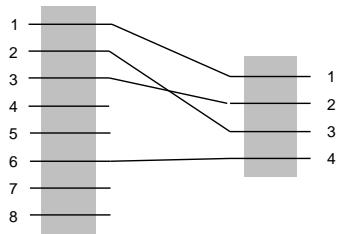


Fig. 18. M12 to RJ45 pin assignment

## 3.4 Cables (recommendation)

Please refer to IEC 61918 and IEC 61784-5-13.

The following is provided for convenience; please refer to the corresponding International Standards. To increase noise immunity only S/FTP or SF/FTP cables should be used (Cat5). The maximum cable length (100 meters) predefined by Ethernet 100Base-TX shall apply.

The pin assignment shall be that of a crossover cable.

Regarding wiring and EMC measures, the IEC 61918 and IEC 61784-5-13 shall be considered.

## 4 Data Link Layer

### 4.1 Modes of Operation

Two operating modes are defined for POWERLINK networks:

- **POWERLINK mode**

In POWERLINK Mode network traffic follows the set of rules given in this standard for Real-time Ethernet communication. Network access is managed by a master, the POWERLINK Managing Node (MN). A node can only be granted the right to send data on the network via the MN. The central access rules preclude collisions, the network is therefore deterministic in POWERLINK Mode.

In POWERLINK Mode most communication transactions are via POWERLINK-specific messages. An asynchronous slot is available for non-POWERLINK frames. UDP/IP is the preferred data exchange mechanism in the asynchronous slot; however, it is possible to use any protocol.

- **Basic Ethernet mode**

In Basic Ethernet Mode network communication follows the rules of Legacy Ethernet (IEEE802.3). Network access is via CSMA/CD. Collisions occur, and network traffic is non-deterministic.

Any protocol on top of Ethernet may be used in Basic Ethernet mode, the preferred mechanisms for data exchange between nodes being UDP/IP and TCP/IP.

### 4.2 POWERLINK Mode

#### 4.2.1 Introduction

POWERLINK Mode is based on the standard Ethernet CSMA/CD technique (IEEE 802.3) and thus works on all Legacy Ethernet hardware.

Determinism is achieved with a pre-planned and organized message exchange: messages are grouped in cycles, which are subdivided into the isochronous and the asynchronous phase.

Each node gets permission for sending its own frames by the POWERLINK MN. Therefore, no collisions should occur and the collision-resolving re-transmission of messages defined by IEEE802.3, which is responsible for the non-deterministic behavior of Legacy Ethernet, is not used.

#### 4.2.2 POWERLINK Nodes

The node managing the permission to send messages to the Ethernet is called the POWERLINK Managing Node (MN).

All other nodes transmit only within communication slots assigned by the MN. They are thus called Controlled Nodes (CN).

##### 4.2.2.1 POWERLINK Managing Node

Only the MN may send messages independently – i.e. not as a response to a received message. Controlled Nodes shall be only allowed to send when requested to by the MN.

The Controlled Nodes shall be accessed cyclically by the MN. Unicast data shall be sent from the MN to each configured CN (frame: PReq), which shall then publish its data via multicast to all other nodes (frame: PRes).

Optionally, the last frame in the isochronous phase may be a multicast PRes frame of the MN (see Fig. 19). With this frame the MN may publish its own data to all other nodes.

All available nodes in the network shall be configured on the MN.

Only one active MN is permitted in a POWERLINK network.

The ability of a node to perform MN functions shall be indicated by the device description entry D\_DLL\_FeatureMN\_BOOL.

## 4.2.2.2 POWERLINK Controlled Node

CNs shall be passive bus nodes. They shall only send when requested by the MN.

The ability of a node to perform CN functions shall be indicated by the device description entry D\_DLL\_FeatureCN\_BOOL.

### 4.2.2.2.1 Isochronous CN

Each isochronous CN shall receive a unicast PReq frame from the MN in the POWERLINK cycle and shall send back a PRes frame to the MN. PReq and PRes frames may transport isochronous data.

CNs may be accessed every cycle or every  $n^{\text{th}}$  cycle (multiplexed nodes,  $n > 1$ ).

PReq can only be received by the specifically addressed CN. However, PRes frames shall be sent by the CN as multicast messages, allowing all other CNs to monitor the data being sent.

Additional data from the MN may be received by a multicast PRes message transmitted by the MN.

Isochronous CNs shall request the right to transmit asynchronous data from the MN, if required.

The ability of a CN to perform isochronous communication shall be indicated by a feature flag in the object dictionary entry NMT\_FeatureFlags\_U32 (1F82<sub>h</sub>) and the device description entry D\_NMT\_Isochronous\_BOOL.

No POWERLINK CN device shall rely on being polled as isochronous CN, when the network performs the isochronous POWERLINK cycle (4.2.4.1). Async-only CN type communication shall be guaranteed. Application level function limitations may occur when a device enabled to perform isochronous functions is operated in Async-only mode.

### 4.2.2.2.2 Async-only CN

CNs may be operated in a way, that they aren't accessed cyclically in the isochronous phase by the MN.

The MN shall cyclically poll each async-only CN during the asynchronous phase with a StatusRequest – a special form of the SoA frame. The CN shall respond with a StatusResponse, special form of Asynchronous Send frame. The poll interval shall be at least C\_NMT\_STATREQ\_CYCLE. It is affected by the asynchronous scheduling and is thus non-deterministic.

Async-only CNs shall request the right to transmit asynchronous data from the MN, if required.

Async-only CNs shall actively communicate during the asynchronous phase only. Nevertheless, they may listen to the multicast network traffic, transmitted by the MN and the isochronous CNs.

## 4.2.3 Services

POWERLINK provides three services:

- Isochronous Data Transfer

One pair of messages per node shall be delivered every cycle, or every  $n^{\text{th}}$  cycle in the case of multiplexed CNs.

Additionally, there may be one multicast PRes message from the MN per cycle.

Isochronous data transfer is typically used for the exchange of time critical data (real-time data).

- Asynchronous Data Transfer

There may be one asynchronous message per cycle. The right to send shall be assigned to a requesting node by the MN via the SoA message.

Asynchronous data transfer is used for the exchange of non time-critical data.

- Synchronisation of all nodes

At the beginning of each isochronous phase, the MN transmits the multicast SoC message very precisely to synchronise all nodes in the network.

## 4.2.4 POWERLINK Cycle

The POWERLINK cycle shall be controlled by the MN.

#### 4.2.4.1 Isochronous POWERLINK Cycle

Isochronous data exchange between nodes shall occur cyclically. It shall be repeated in a fixed interval, called POWERLINK cycle.

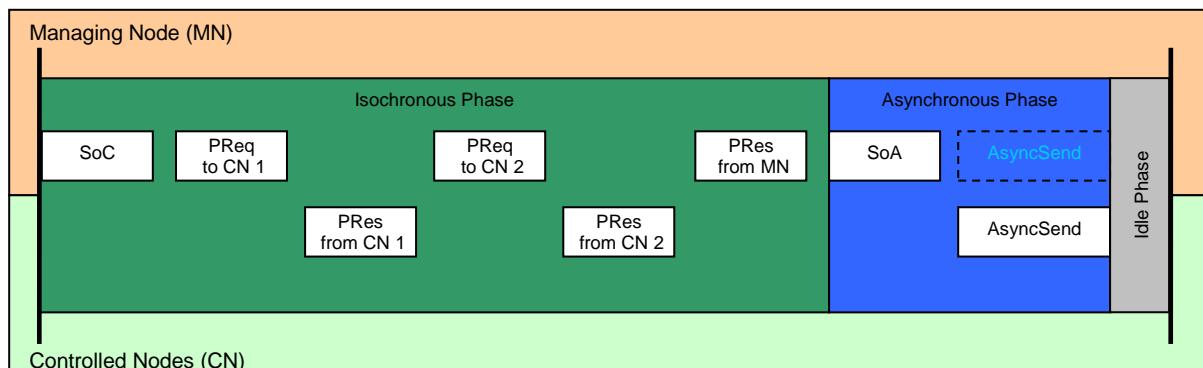


Fig. 19. POWERLINK Cycle

The following time phases exist within one cycle:

- Isochronous phase
- Asynchronous phase
- Idle phase

It is important to keep the start time of a POWERLINK cycle as exact (jitter-free) as possible. The length of individual phases can vary within the preset phase of a POWERLINK cycle.

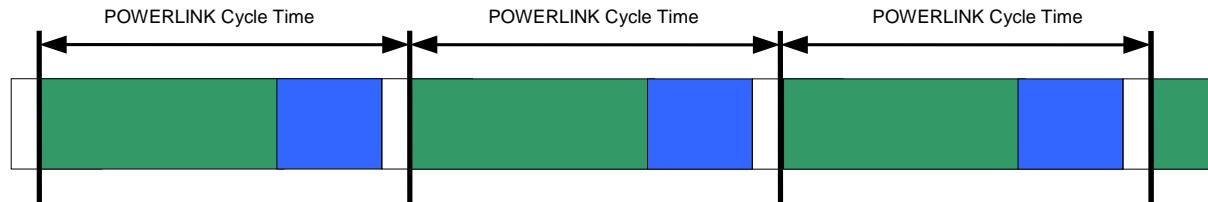


Fig. 20. POWERLINK - an isochronous process

The network shall be configured in a way that the preset cycle time is not exceeded. Adherence to the cycle time shall be monitored by the MN.

All data transfers shall be unconfirmed, i.e. there is no confirmation that sent data has been received. To maintain deterministic behavior, protecting the isochronous data (PReq and PRes) is neither necessary nor desired. Asynchronous data may be protected by higher protocol layers.

##### 4.2.4.1.1 Isochronous phase

At the beginning of a POWERLINK cycle, the MN shall send a SoC frame to all nodes via Ethernet multicast. The send and receive time of this frame shall be the basis for the common timing of all the nodes.

Only the SoC frame shall be generated on a periodic basis. The generation of all other frames shall be event controlled (with additional time monitoring per node).

The MN shall start the isochronous data exchange after the SoC frame has been sent.

A PReq frame shall be sent to every configured and active node. The accessed node shall respond by a PRes frame.

PReq shall be an Ethernet unicast frame. It is received by the target node only. PRes shall be sent as an Ethernet multicast frame.

Both the PReq and the PRes frames may transfer application data. The MN only sends PReq data to one CN per frame. PReq transfer is dedicated to data relevant for the addressed CN only.

In contrast, the PRes frame may be received by all nodes. This makes communication relationships possible according to the producer/consumer model.

The PReq / PRes procedure shall be repeated for each configured and active isochronous CN.

The MN may send a multicast PRes frame to all nodes. This frame is dedicated to transfer data relevant for groups of CNs.

Support of PRes transmission by the MN is optional. The ability of an MN to transmit PRes shall be indicated by the device description entry D\_DLL\_MNFeaturePResTx\_BOOL. If the feature is provided, transmission shall be enabled by NMT\_NodeAssignment\_AU32[C\_ADR\_MN\_DEF\_NODE\_ID ].Bit 12.

The isochronous phase shall be calculated from start of SoC to start of SoA.

The size of the POWERLINK cycle is predominantly affected by the size of the isochronous phase. When configuring the POWERLINK cycle, the sum of the times required by the PReq / PRes accesses to each configured CN shall be taken into account, i.e. the time needed to access all configured nodes in one cycle has to be accounted for. Use of the multiplexed access technology (4.2.4.1.1) may reduce the amount of time.

When operating the isochronous phase, the length of this phase may vary according to the number of active CNs.

The order in which CNs are polled may be implementation specific or controlled by object NMT\_IsochrSlotAssign\_AU8 if supported by the MN. An implementation should pack the performed PReq / PRes packages to the begin of the isochronous phase. It should provide means to rearrange the poll order, to avoid location of the nodes having the worst SoC latency time value (D\_NMT\_CNSoC2PReq\_U32) at the slot following SoC.

Isochronous phases may be counted by the device. If implemented, counting shall be performed on base of transmitted or received SoC frames. The counter value may be accessed via DIA\_NMTTelegrCount\_REC.IsochrCyc\_U32.

#### 4.2.4.1.1 Multiplexed Timeslots

POWERLINK supports CN communication classes, that determine the cycles in which nodes are to be addressed.

- Continuous  
Continuous data shall be exchanged in every POWERLINK cycle.
- Multiplexed  
Multiplexed data to and from one CN shall not be exchanged in every POWERLINK cycle.  
The accesses to the multiplexed CNs shall be dispersed to the multiplexed cycle that consists of a number of POWERLINK cycles.  
The dispersion allows the isochronous access to a large number of CNs without elongating the POWERLINK cycle to an unacceptable amount. However, multiplexed CN access reduces the poll frequency to the particular CN.

The configuration of the multiplexed cycle is shown in 4.2.4.4.

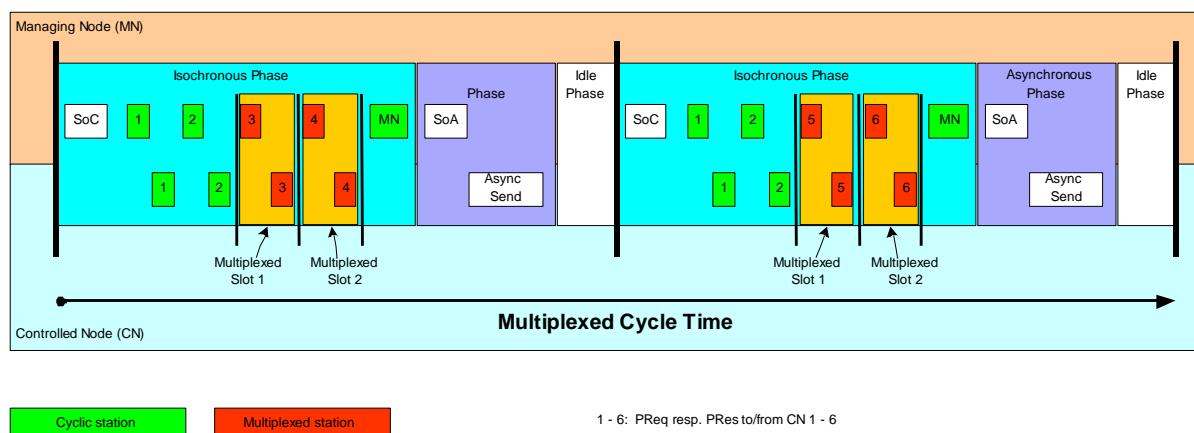


Fig. 21. Multiplexed POWERLINK cycle

Continuous and multiplexed access may be operated in parallel during one POWERLINK cycle. The apportionment of the isochronous phase to continuous and multiplexed slots shall be fixed by configuration (NMT\_MultiplCycleAssign\_AU8, NMT\_IsochrSlotAssign\_AU8).

Although the multiplexed nodes are not processed in each cycle, they can monitor the entire data transfer of the continuous nodes because all PRes frames are sent as multicast frames.

E.g. in Motion Control, multiplexed timeslots can be used for a large number of slave axes to receive position data from few master axes. The master axes are configured to communicate continuously, accesses to the slave axes multiplexed. In this way, the master axes transmit their data to the (monitoring) slave axes in each cycle, while the slave axes also take part in the communication in a slower cycle.

The size of each particular multiplexed slot shall be equal to the maximum time necessary for the PReq / PRes access of the CN assigned to the slot.

In case of MN cycle loss, the multiplexed access sequence shall be continued on a per time base, after the cycle loss error phase is over. E.g. CNs shall be skipped to maintain time equidistance of access to nodes not affected by the cycle loss.

The ability of an MN enabled node to perform control of multiplexed isochronous operation shall be indicated by the device description entry D\_DLL\_MNFeatureMultiplex\_BOOL. The ability of a CN enabled node to be isochronously accessed in a multiplexed way shall be indicated by the device description entry D\_DLL\_CNFeatureMultiplex\_BOOL.

#### 4.2.4.1.2 Asynchronous phase

In the asynchronous phase of the cycle, access to the POWERLINK network may be granted to one CN or to the MN for the transfer of a single asynchronous message only.

There shall be two types of asynchronous frames available:

- The POWERLINK ASnd frame shall use the POWERLINK addressing scheme and shall be sent via unicast or broadcast to any other node.
- A Legacy Ethernet message may be sent.

If no asynchronous message transmission request is pending at the MN scheduling queues (4.2.4.1.2.1), the MN shall issue a SoA without assignment of the right to send to any node. No ASnd frame will follow to the SoA frame in this case.

The MN shall start the asynchronous phase with the SoA. The SoA shall be used to identify CNs, request status information of a CN, to poll async-only CNs and to grant the asynchronous transmit right to one CN.

The SoA frame is the first frame in the asynchronous phase and is a signal to all CNs that all isochronous data have been exchanged during the isochronous phase.

The asynchronous phase shall be calculated from the start of SoA to the end of the asynchronous response. If no asynchronous response is allowed for any node, the asynchronous phase shall be terminated by the end of SoA .

This definition is valid from the network's point view. It may be different from the node's application point of view. Due to the AsyncSend addressing scheme, the asynchronous phase may be terminated by the end of SoA on those nodes not being addressed, whereas it ends at the end of the asynchronous response on the addressed nodes.

Asynchronous frames may be counted by the device. If implemented, received frames shall be indicated by DIA\_NMTTelegrCount\_REC.AsyncRx\_U32 and transmitted frames by DIA\_NMTTelegrCount\_REC.AsyncTx\_U32.

#### 4.2.4.1.2.1 Asynchronous Scheduling

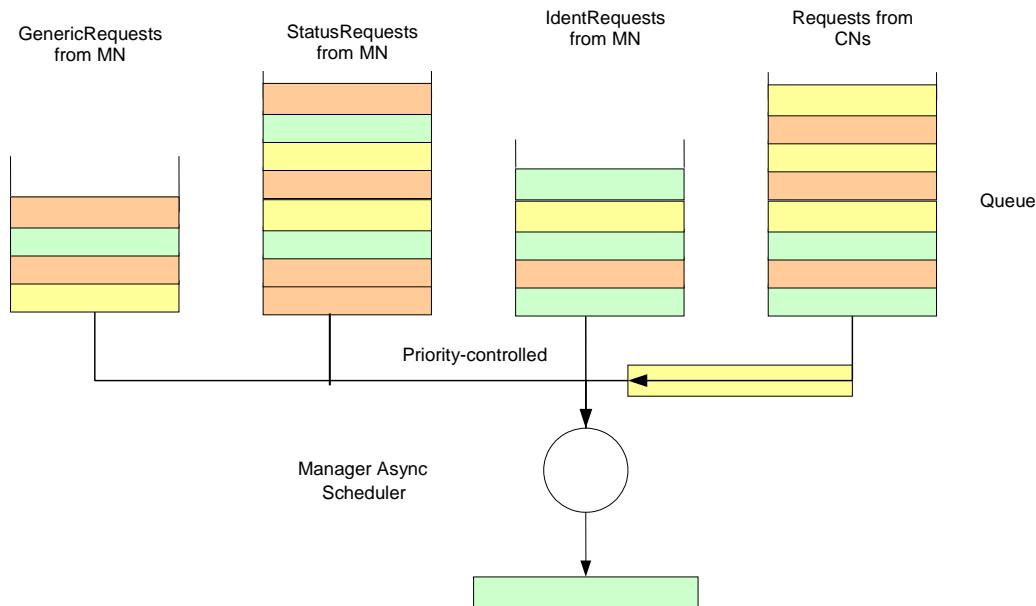


Fig. 22. Asynchronous scheduling

The MN handles scheduling of all asynchronous data transfers.

If a CN wants to send an asynchronous frame, it shall inform the MN via the PRes or the StatusResponse frame.

The asynchronous scheduler of the MN shall determine in which cycle the right to send the asynchronous frame will be granted. It shall guarantee that no send request will be delayed for an indefinite amount of time, even if network load is high.

The MN shall select a node from all queued send requests (including its own). It shall send a SoA frame with a Requested Service Target identifying which node is allowed to send an asynchronous frame.

The MN shall manage the dispatching of the asynchronous phase using different queues:

- Generic transmit requests from the MN.
- IdentRequest frames from the MN to identify CNs
- StatusRequest frames to poll CNs
- Transmit requests from the CNs

#### 4.2.4.1.2.2 Distribution of the Asynchronous phase

With the PRes, IdentResponse or StatusResponse RS flag (3 bits, see 4.6.1.1.4, 7.3.3.2.1, 7.3.3.3.1) the CN shall indicate the number of send-ready packages in its queues.

An RS value of 0 ( $000_b$ ) shall indicate that the queues are empty and an RS value of 7 ( $111_b$ ) shall indicate that 7 or more packages are queued.

The assignment of the asynchronous phase shall decrement the MN-administered number of frames requested by the respective CN. If the MN queue length reached zero, no more further asynchronous phases are assigned.

The algorithm that is used to assign the asynchronous phase when there are multiple requests pending shall be manufacturer-specific.

#### 4.2.4.1.2.3 Asynchronous Transmit Priorities

Asynchronous transmit requests may be prioritized by 3 PR bits in the PRes, the IdentResponse and StatusResponse frame (see 4.6.1.1.4, 7.3.3.2.1, 7.3.3.3.1).

POWERLINK supports eight priority levels. Two of these levels are dedicated to POWERLINK purpose:

- PRIO\_NMT\_REQUEST  
This is the highest priority that shall be exclusively applied if a CN requests an NMT command to be issued by the MN
- PRIO\_GENERIC\_REQUEST  
Medium priority which is the standard priority level for non-NMT command requests. SDO via asynchronous communication (see 6.3.2) requests shall apply this priority level. Application requests may apply PRIO\_GENERIC\_REQUEST

The remaining priority levels above and below PRIO\_GENERIC\_REQUEST are available for application purpose.

Requests with a high priority level shall be preferentially assigned by the MN compared to those with low priority numeric value.

Requests of different priorities shall be handled by independent priority specific queues on the CN..

The PRes PR flags shall indicate the highest priority level containing pending requests. The RS flags shall indicate the number of pending requests at the reported priority level. Lower priority request indication shall be suspended until all high priority requests have been assigned.

Fig. 23 shows an example of priority related asynchronous request handling.

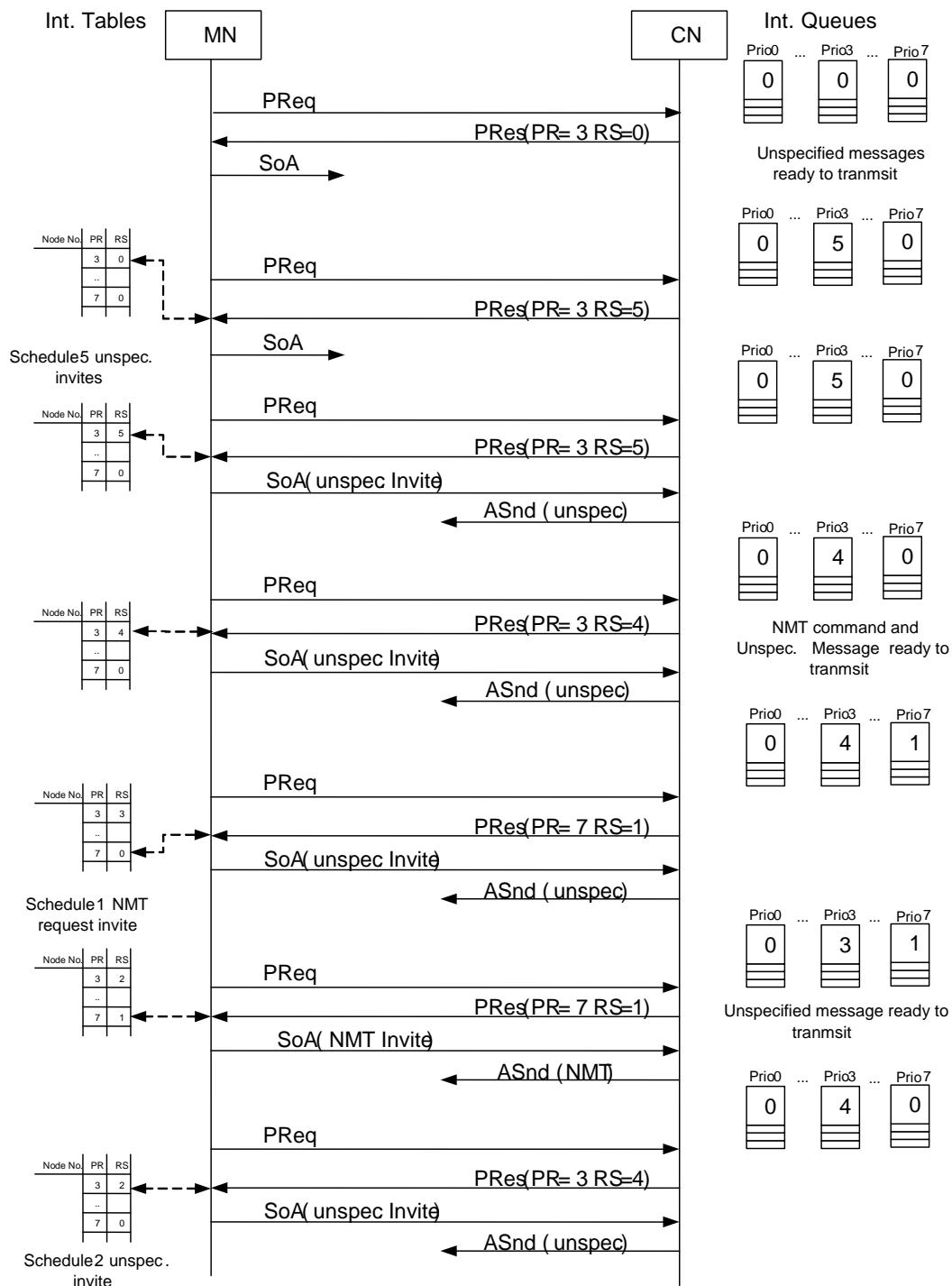


Fig. 23. Asynchronous transmit priority handling  
(Priority level PR: 7 = PRIO\_NMT\_REQUEST, 3 = PRIO\_GENERIC\_REQUEST)

#### 4.2.4.1.3 Idle Phase

The Idle Phase is the remaining time interval between the end of the asynchronous phase and the beginning of the next cycle.

During the Idle Phase, all network components shall "wait" for the beginning of the following cycle. The duration of the Idle Phase may be 0, i.e. an implementation shall not rely on an existing or fixed Idle Phase.

The idle phase shall be calculated from end of SoA or ASnd (see 4.2.4.1.2) to start of SoC.

#### 4.2.4.2 Reduced POWERLINK Cycle

During system startup (NMT state NMT\_MS\_PRE\_OPERATIONAL\_1), a reduced POWERLINK Cycle shall be applied to diminish network load, while the system is being configured via SDO communication.

The Reduced POWERLINK Cycle shall consist of queued asynchronous phases only. The duration of the asynchronous phase and thus the duration of the Reduced POWERLINK Cycle may vary from one cycle to the next.

If a CN is assigned to send and there are no information about the length of the expected AsyncSend frame available at the MN, the next Reduced POWERLINK Cycle shall not start until at least the timeout given by the length of a maximum size ethernet frame

NMT\_CycleTiming\_REC.AsyncMTU\_U16 plus the maximum response time to the SoA invite message required by the CNs (NMT\_CycleTiming\_REC.ASndMaxLatency\_U32) elapsed.

If the MN endues AsyncSend length information, i.e. if the MN assignes the Asynchronous Slot to itself or if the MN is the target node of the asynchronous message, it may reduce the Reduce POWERLINK cycle length (Fig. 24).

If there is no assignment to any node (MN included), the next Reduced POWERLINK Cycle may follow without any timeout.

The assignment mechanism used for the asynchronous phase of the isochronous POWERLINK cycle (4.2.4.1.2) shall also be applied to the Reduced POWERLINK Cycle.

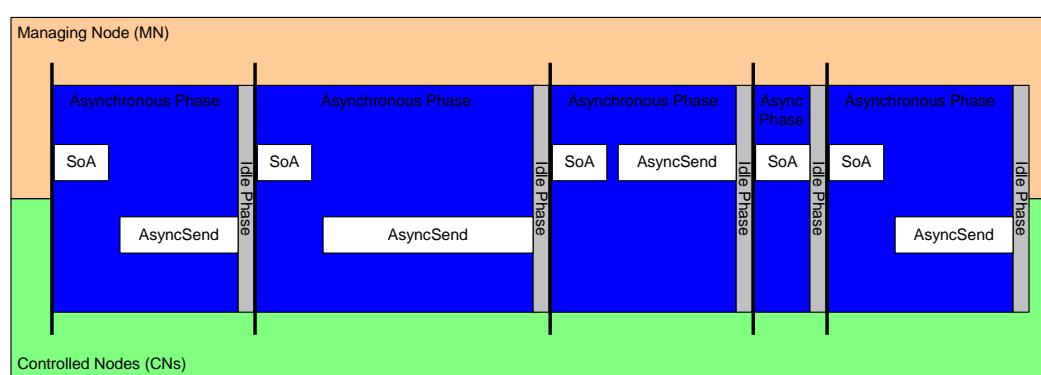


Fig. 24. Reduced POWERLINK cycle

The Reduded POWERLINK Cycle shall be robust to collisions. Collisions shall be resolved by CSMA/CD. In case of collision, the next Reduced POWERLINK Cycle shall start after a timeout given by the double length of a maximum size ethernet frame (1518 Bytes).

#### 4.2.4.3 POWERLINK Cycle Timing

Fig. 25, Fig. 26 and Fig. 27 show the timing of the isochronous POWERLINK cycle. The figures show a system of three nodes. The MN is physically located between two isochronously accessed CNs.

The figures are a schematic approach. They are valid for the NMT states NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_OPERATIONAL or NMT\_CS\_READY\_TO\_OPERATE and NMT\_CS\_OPERATIONAL. The amount of time shown does not reflect the realistic timing relationships of a real system.

Fig. 25 shows the typical sequence of messages in the isochronous phase. The PRes frames from one CN are received by the other CN (cross traffic). Fig. 26 depicts the timing of a transmission of an asynchronous frame by a CN, Fig. 27 the less critical case of asynchronous transmission by the MN.

Tab. 4 provides the description of the timing parameters introduced by the figures, the source of the data, special handling of the data and object dictionary entries containing the parameters.

The node located verifications proposed by Tab. 4 are optional on the node if not mentioned otherwise. A POWERLINK network configuration tool shall perform all the verifications during configuration process.

#### Application hint:

*The timing parameters introduced by this chapter allows a sophisticated fine tuning of the cycle timing, required by notably cycle time sensitive high end applications.*

*The majority of applications will apply default values for most of the parameters. Only a few variables will remain to be setup to configure a typical POWERLINK network.*

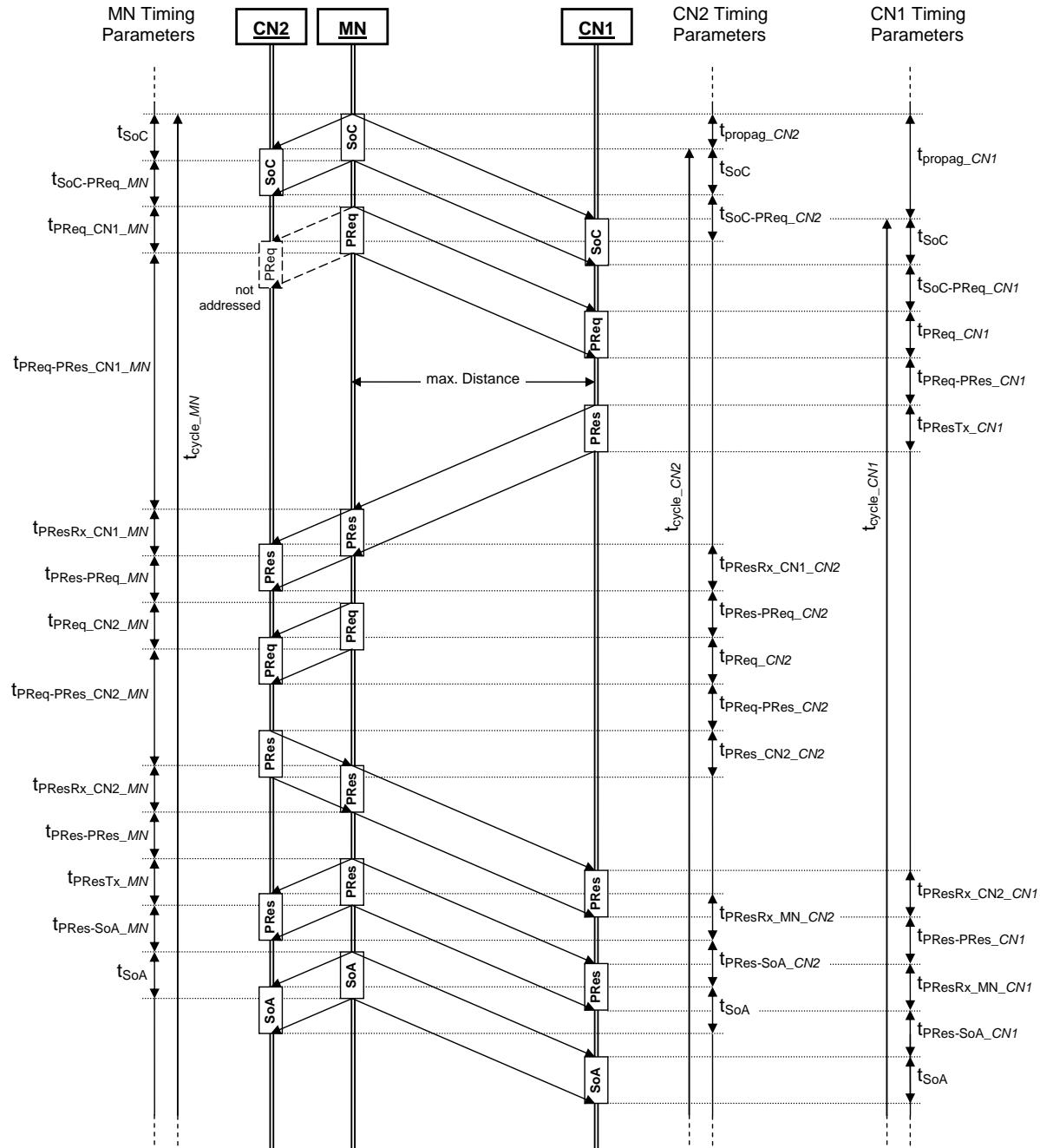
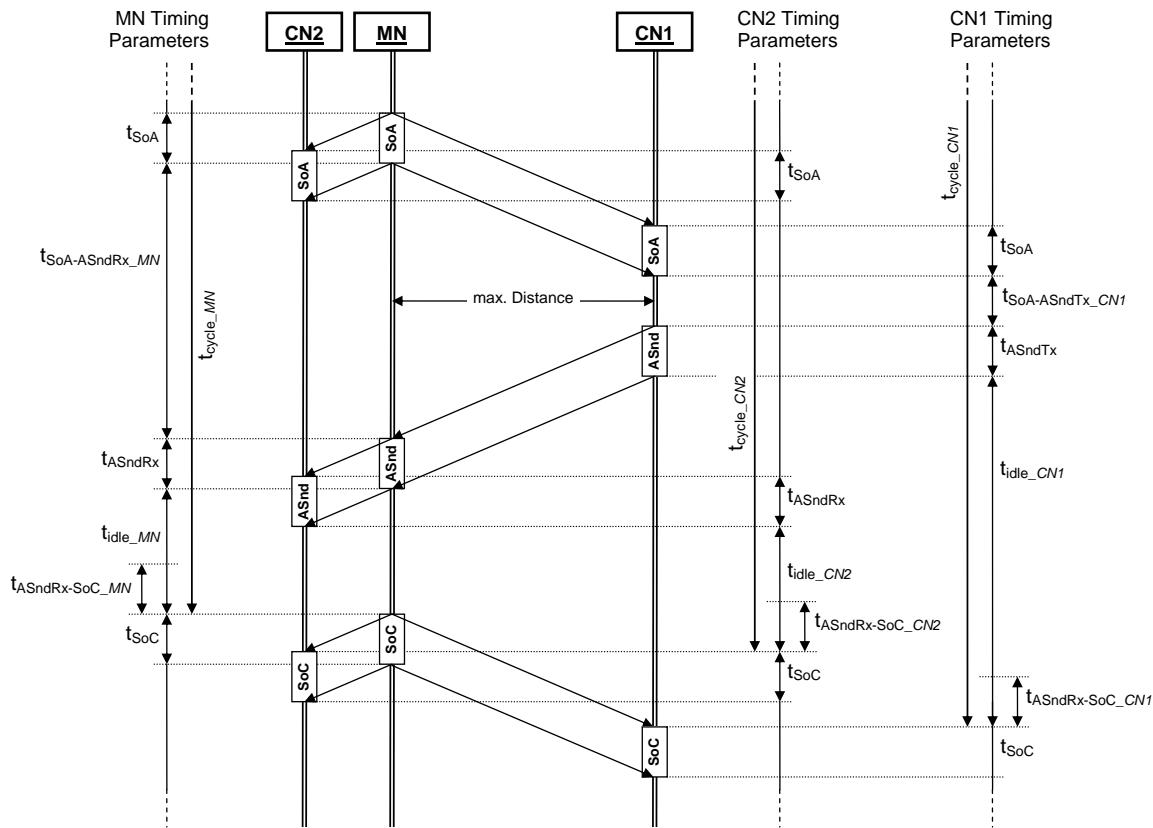
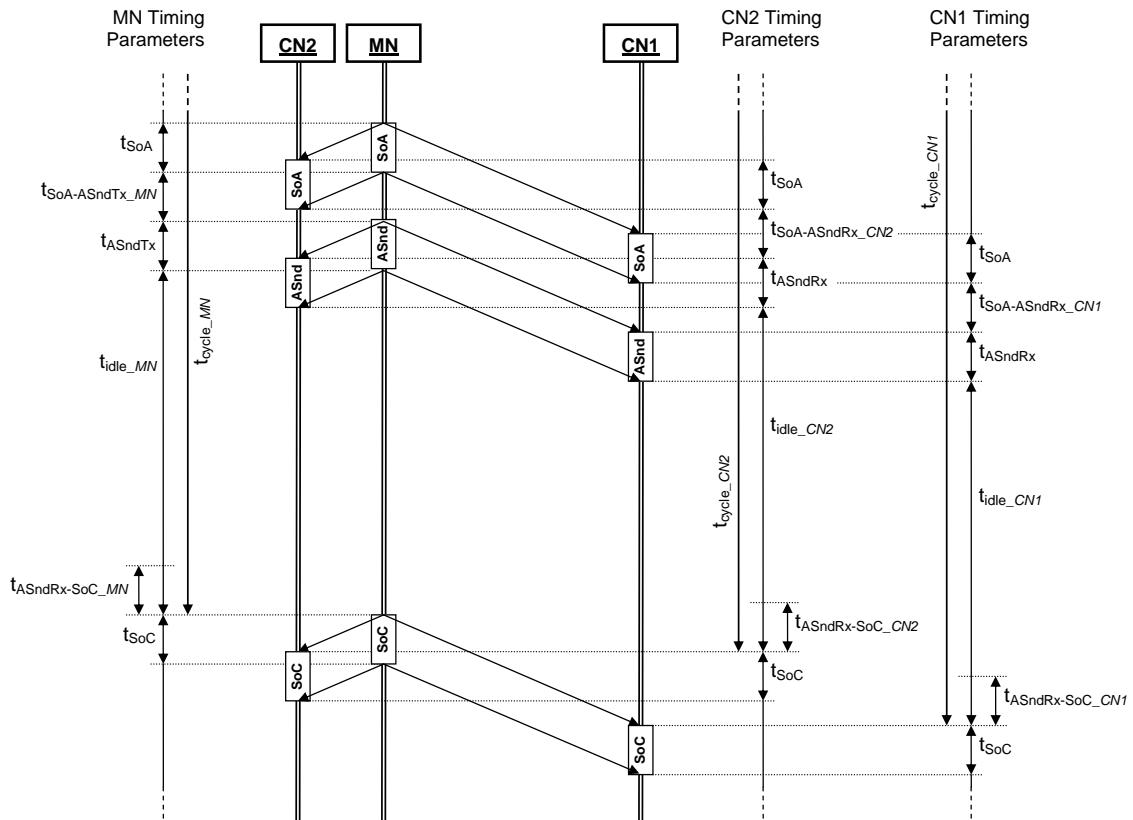


Fig. 25. POWERLINK cycle timing, start phase and isochronous phase



Hint: ASnd refers to POWERLINK ASnd frames and non-POWERLINK frames

Fig. 26. POWERLINK cycle timing, asynchronous phase, AsyncSend transmission by CN



Hint: ASnd refers to POWERLINK ASnd frames and non-POWERLINK frames

Fig. 27. POWERLINK cycle timing, asynchronous phase, AsyncSend transmission by MN

| No | Parameter                              | Description  | Data Source   | Handling <sup>1</sup>   | Index   |
|----|--|--|---|---|---|
| 1  | $t_{cycle\_MN}$ ,<br>$t_{cycle\_CNn}$  | length of POWERLINK isochronous cycle  | system configuration  | MN may verify before isochronous access to CN <sub>n</sub> :<br>$t_{cycle\_MN} = t_{cycle\_CNn}$  | NMT_CycleLen_U32  |
| 2  | $t_{soc}$                              | SoC frame time consumption   | constant<br>$t_{soc} = C\_DLL\_T\_MIN\_FRAME + C\_DLL\_T\_PREAMBLE$   | --  | --  |
| 3  | $t_{propag\_CNn}$                      | signal propagation time MN → CN <sub>n</sub> and vice versa  | system configuration  | $t_{propag\_CNn} = (t_{PReq-PRes\_CNn\_MN} - t_{PReq-PRes\_CNn}) / 2$   | --  |
| 4  | $t_{SoC-PReq\_MN}$                     | delay between end of SoC transmission and start of PReq transmission                               | system configuration  | depends on implementation of NMT_MNCycleTiming_REC. WaitSoCPReq:<br>a. $t_{SoC-PReq\_MN} = \max(t_{SoC-PReq\_CNn}, t_{SoC-PReqMin\_MN})$<br>b. $t_{SoC-PReq\_MN} = t_{SoC-PReqMin\_MN}$ | a. NMT_MNCycleTiming_REC.<br>WaitSoCPReq<br>b. --           |
| 5  | $t_{SoC-PReqMin\_MN}$                  | MN minimum delay between end of SoC transmission and start of PReq transmission                    | MN device description<br>D_NMT_MNSoC2PReq_U32   | $t_{SoC-PReqMin\_MN} \geq C\_DLL\_T\_IFG$   | --  |
| 6  | $t_{SoC-PReq\_CNn}$                    | minimum delay required by CN <sub>n</sub> between end of SoC reception and start of PReq reception | CN <sub>n</sub> device description<br>D_NMT_CNSoC2PReq_U32  | $t_{SoC-PReq\_CNn} \geq C\_DLL\_T\_IFG$   | --  |
| 7  | $t_{PReq\_CNn\_MN}$<br>$t_{PReq\_CNn}$ | time consumption of PReq frame to be transmitted to CN <sub>n</sub>                                | $t_{PReq\_CNn\_MN} = (t_{PReqPL\_CNn\_MN} + C\_DLL\_T\_EPL\_PDO\_HEADER + C\_DLL\_T\_ETH2\_WRAPPER) * 8 * C\_DLL\_T\_BITTIME + C\_DLL\_T\_PREAMBLE$ | --  | --  |
| 8  | $t_{PReqPL\_CNn\_MN}$                  | payload of PReq frame to be transmitted to CN <sub>n</sub>   | system configuration  | MN may verify at start up:<br>$t_{PReqPL\_CNn\_MN} \leq t_{PReqPLMax\_MN}$<br>MN may verify before isochronous access to CN <sub>n</sub> :<br>$t_{PReqPL\_CNn\_MN} = t_{PReqPL\_CNn}$   | NMT_MNPReqPayloadLimitList_AU16<br>[NodeID <sub>CNn</sub> ] |
| 9  | $t_{PReqPLMax\_MN}$                    | size of MN isochronous frame transmit buffer (referring to payload)                                | system configuration  | --  | NMT_CycleTiming_REC.<br>IsochrTxMaxPayload_U16              |
| 10 | $t_{PReqPL\_CNn}$                      | payload of PReq frame expected by CN <sub>n</sub>  | system configuration  | CN <sub>n</sub> may verify at start up:<br>$t_{PReqPL\_CNn} \leq t_{PReqPLMax\_CNn}$  | NMT_CycleTiming_REC.<br>PReqActPayloadLimit_U16             |

<sup>1</sup> The node located verifications are optional if not mentioned otherwise. A configuration tool shall perform the verifications during network configuration process.

| No | Parameter  | Description  | Data Source   | Handling <sup>1</sup>  | Index   |
|----|--|--|---|--|---|
| 11 | $t_{PReqPLMax\_CNn}$   | size of $CN_n$ isochronous frame receive buffer (referring to payload)                                 | system configuration  | --   | NMT_CycleTiming_REC.<br>IsochrRxMaxPayload_U16  |
| 12 | $t_{PReq-PRes\_CNn}$   | delay between end of PReq reception and start of PRes transmission                                     | system configuration  | $t_{PReq-PRes\_CNn} \geq C\_DLL\_T\_IFG$   | NMT_CycleTiming_REC.<br>PResMaxLatency_U32  |
| 13 | $t_{PResTx\_CNn}$<br>$t_{PResRx\_CNn\_MN}$<br>$t_{PResRx\_CNn\_CNm}$ | time consumption of PRes to be transmitted by $CN_n$   | $t_{PResTx\_CNn} = (t_{PResPLTx\_CNn} + C\_DLL\_T\_EPL\_PDO\_HEADER + C\_DLL\_T\_ETH2\_WRAPPER) * 8 * C\_DLL\_T\_BITTIME + C\_DLL\_T\_PREAMBLE$ | --   | --  |
| 14 | $t_{PResPLTx\_CNn}$  | payload of PRes to be transmitted by $CN_n$  | system configuration  | $CN_n$ may verify at start up:<br>$t_{PResPLTx\_CNn} \leq t_{PResPLTxMax\_CNn}$  | NMT_CycleTiming_REC.<br>PResActPayloadLimit_U16   |
| 15 | $t_{PResPLTxMax\_CNn}$   | size of $CN_n$ PRes frame transmit buffer (referring to payload)                                       | system configuration  | --   | NMT_CycleTiming_REC.<br>IsochrTxMaxPayload_U16  |
| 16 | $t_{PReq-PRes\_CNn\_MN}$   | delay timeout between end of transmission of PReq to $CN_n$ and start of reception of PRes from $CN_n$ | system configuration  | <p>if 1<sup>st</sup> device polled after SoC depends on implementation of NMT_MNCycleTiming_REC. WaitSoCPReq (cf. Tab. 4, Line 4):</p> <ul style="list-style-type: none"> <li>a. <math>t_{PReq-PRes\_CNn\_MN} = t_{PReq-PRes\_CNn} + 2 * t_{propag\_CNn}</math></li> <li>b. <math>t_{PReq-PRes\_CNn\_MN} = \max(t_{SoC-PReq\_CNn}) + t_{PReq-PRes\_CNn} + 2 * t_{propag\_CNn}</math></li> </ul> <p>else</p> $t_{PReq-PRes\_CNn\_MN} = t_{PReq-PRes\_CNn} + 2 * t_{propag\_CNn}$ <p>used by CN response supervision of Error Handling DLL</p> | depends on implementation of NMT_MNCycleTiming_REC. WaitSoCPReq (cf. Tab. 4, Line 4) <ul style="list-style-type: none"> <li>a. NMT_MNCNPResTimeout_AU32 [NodeID<math>_{CNn}</math>]</li> <li>b. NMT_MNCNPResTimeout_AU32 [NodeID<math>_{CNn}</math>], NMT_MNCycleTiming_REC. WaitSoCPReq</li> </ul> |
| 17 | $t_{PResPLRx\_CNn\_MN}$  | payload of PRes frame from $CN_n$ expected by MN   | system configuration  | <p>MN may verify at start up:<br/><math>t_{PResPLRx\_CNn\_MN} \leq t_{PResPLRxMax\_MN}</math></p> <p>MN may verify before isochronous access to <math>CN_n</math>:</p> $t_{PResPLRx\_CNn\_MN} = t_{PResPLTx\_CNn}$   | NMT_PResPayloadLimitList_AU16 [NodeID $_{CNn}$ ]  |

| No | Parameter                                 | Description  | Data Source   | Handling <sup>1</sup>   | Index   |
|----|---|--|---|---|---|
| 18 | $t_{PResPLRxMax\_MN}$                     | size of MN PRes frame receive buffer (referring to payload)  | system configuration  | --  | NMT_CycleTiming_REC.<br>IsochrRxMaxPayload_U16      |
| 19 | $t_{PResPLRx\_CNm\_CNn}$                  | payload of PRes frame from $CN_m$ expected by $CN_n$   | system configuration  | $CN_n$ may verify at start up:<br>$t_{PResPLRx\_CNm\_CNn} \leq t_{PResPLRxMax\_CNn}$ <sup>2</sup><br>and<br>$t_{PResPLRx\_CNm\_CNn} \geq t_{PResPLTx\_CNm}$ | NMT_PResPayloadLimitList_AU16<br>[NodeID $_{CNm}$ ] |
| 20 | $t_{PResPLRxMax\_CNn}$                    | size of $CN_n$ PRes frame receive buffer (referring to payload)  | system configuration  | --  | NMT_CycleTiming_REC.<br>IsochrRxMaxPayload_U16      |
| 21 | $t_{PRes-PReq\_MN}$                       | delay between end of PRes reception and start of PReq transmission   | MN device description<br>D_NMT_MNPRes2PReq_U32  | $t_{PRes-PReq\_MN} \geq C\_DLL\_T\_IFG$   | --  |
| 22 | $t_{PRes-PReq\_CNn}$                      | minimum delay between end of PRes reception and start of PReq reception  | $CN_n$ implementation requirement   | every $CN$ shall support<br>$t_{PRes-PReq\_MN} = C\_DLL\_T\_IFG$  | --  |
| 23 | $t_{PRes-PRes\_MN}$                       | delay between end of reception of PRes from $CN_n$ and start of transmission of PRes by MN                     | MN device description<br>D_NMT_MNPRes2PRes_U32  | $t_{PRes-PRes\_MN} \geq C\_DLL\_T\_IFG$   | --  |
| 24 | $t_{PRes-PRes\_CNn}$                      | minimum delay between end of reception of PRes from $CN_n$ and start of reception of PRes from MN <sup>3</sup> | $CN_n$ implementation requirement   | every $CN$ shall support<br>$t_{PRes-PRes\_MN} = C\_DLL\_T\_IFG$  | --  |
| 25 | $t_{PResTx\_MN}$<br>$t_{PResRx\_MN\_CNn}$ | time consumption of PRes frame to be transmitted by MN   | $t_{PResTx\_MN\_MN} = ( t_{PResPLTx\_MN\_MN} + C\_DLL\_T\_EPL\_PDO\_HEADER + C\_DLL\_T\_ETH2\_WRAPPER ) * 8 * C\_DLL\_T\_BITTIME + C\_DLL\_T\_PREAMBLE$ | --  | --  |
| 26 | $t_{PResPLTx\_MN\_MN}$                    | payload of PRes frame to be transmitted by MN  | system configuration  | MN may verify at start up:<br>$t_{PResPLTx\_MN\_MN} \leq t_{PResPLTxMax\_MN}$   | NMT_PResPayloadLimitList_AU16<br>[NodeID $_{MN}$ ]  |
| 27 | $t_{PResPLTxMax\_MN}$                     | size of MN PRes frame transmit buffer (referring to payload)   | system configuration  | --  | NMT_CycleTiming_REC.<br>IsochrTxMaxPayload_U16      |
| 28 | $t_{PResPLRx\_MN\_CNn}$                   | payload of PRes frame from MN expected by $CN_n$   | system configuration  | $CN_n$ may verify at start up:<br>$t_{PResPLRx\_MN\_CNn} \leq t_{PResPLRxMax\_CNn}$   | NMT_PResPayloadLimitList_AU16<br>[NodeID $_{MN}$ ]  |

<sup>2</sup> see 7.2.1.5.5<sup>3</sup> For future development, CNs shall also support the reception of several PRes from other CNs without intermitting time gaps.

| No | Parameter                    | Description  | Data Source   | Handling <sup>1</sup>   | Index                                      |
|----|------------------------------|--|---|---|--|
| 29 | $t_{PRes-SoA\_MN}$           | if PRes transmission by MN:<br>delay between end of PRes<br>transmission by MN and start of<br>transmission of SoA by MN<br><br>else<br>delay between end of reception of<br>PRes from CN <sub>n</sub> and start of<br>transmission of SoA by MN | MN device description<br>D_NMT_MNPResTx2SoA_U32<br>resp.<br>D_NMT_MNPResRx2SoA_U32                        | $t_{PRes-SoA\_MN} \geq C\_DLL\_T\_IFG$  | --   |
| 30 | $t_{PRes-SoA\_CNn}$          | minimum delay between end of<br>reception of PRes and start of<br>reception of SoA   | CN <sub>n</sub> implementation requirement  | every CN shall support<br>$t_{PRes-SoA\_MN} = C\_DLL\_T\_IFG$   | --   |
| 31 | $t_{SoA}$                    | SoA frame time consumption   | constant<br>$t_{SoA} = C\_DLL\_T\_MIN\_FRAME + C\_DLL\_T\_PREAMBLE$                                       | --  | --   |
| 32 | $t_{SoA-AsndTx\_CNn}$        | delay between end of SoA reception<br>and start of AsyncSend transmission  | system configuration  | $t_{SoA-AsndTx\_CNn} \geq C\_DLL\_T\_IFG$   | NMT_CycleTiming_REC.<br>ASndMaxLatency_U32 |
| 33 | $t_{ASndTx}$<br>$t_{ASndRx}$ | time consumption of asynchronous<br>frame to be transmitted  | $t_{ASndTx} = ( t_{ASndPLTx} + C\_DLL\_T\_ETH2\_WRAPPER ) * 8 * C\_DLL\_T\_BITTIME + C\_DLL\_T\_PREAMBLE$ | --  | --   |
| 34 | $t_{ASndPLTx}$               | payload of asynchronous frame to be<br>transmitted   | application   | Node shall verify:<br>$t_{ASndPLTx} \leq t_{ASndMTU\_MN/CNn}$   | --   |
| 35 | $t_{ASndPLRx}$               | payload of received asynchronous<br>frame  | application   | --  | --   |
| 36 | $t_{ASndMTU\_MN/CNn}$        | maximum size of asynchronous frame<br>to be transmitted (referring to payload)   | system configuration  | Node may verify:<br>$t_{ASndMTU\_MN/CNn} \leq t_{ASndPLTxMax\_MN/CNn}$<br>and<br>$t_{ASndMTU\_MN/CNn} \leq t_{ASndPLRxMax\_MN/CNn}$<br>MN may verify:<br>$t_{ASndMTU\_MN} = t_{ASndMTU\_CNn}$ | NMT_CycleTiming_REC.<br>AsyncMTU_U16       |
| 37 | $t_{ASndPLTxMax\_MN/CNn}$    | size of AsyncSend frame transmit<br>buffer (referring to payload)  | system configuration  | --  | NMT_CycleTiming_REC.<br>AsyncMTU_U16       |
| 38 | $t_{ASndPLRxMax\_MN/CNn}$    | size of MN AsyncSend frame receive<br>buffer (referring to payload)  | system configuration  | --  | NMT_CycleTiming_REC.<br>AsyncMTU_U16       |

| No | Parameter                                    | Description  | Data Source  | Handling <sup>1</sup>  | Index                 |
|----|--|--|--|--|-----------------------|
| 39 | $t_{SoA-AsndRx\_MN}$                         | delay timeout between end of transmission of SoA and start of reception of AsyncSend from that CN, that requires the most time for signal travel and response to SoA | system configuration   | $t_{SoA-AsndRx\_MN} = \max(t_{SoA-AsndTx\_CNn} + 2 * t_{propag\_CNn})$<br>used by CN response supervision of Error Handling DLL in case of SoA.StatusRequest or SoA.IdentRequest | NMT_MNCycleTiming_REC |
| 40 | $t_{idle\_MN}, t_{idle\_CNn}$                | idle time before next cycle  | system configuration, time consumption by asynchronous traffic | MN may verify:<br>$t_{idle\_MN} \geq t_{ASndRx-SoC\_MN}$<br>CN <sub>n</sub> may verify:<br>$t_{idle\_CNn} \geq t_{ASndRx-SoC\_CNn}$  | --                    |
| 41 | $t_{ASndRx-SoC\_MN}$                         | minimum delay between end of reception of AsyncSend and start of transmission of SoC   | MN device description D_NMT_MNASnd2SoC_U32                     | $t_{ASndRx-SoC\_MN} \geq C\_DLL\_T\_IFG$   | --                    |
| 42 | $t_{ASndRx-SoC\_CNn}$                        | minimum delay between end of reception of AsyncSend and start of reception of SoC  | CN <sub>n</sub> implementation requirement                     | every CN shall support<br>$t_{ASndRx-SoC\_MN} = C\_DLL\_T\_IFG$  | --                    |
| 43 | $t_{SoA-AsndTx\_MN}$<br>$t_{SoA-AsndRx\_CN}$ | delay between end of transmission of SoA and start of transmission of AsyncSend by MN  | MN device description D_NMT_MNSoA2ASndTx_U32                   | $t_{SoA-AsndTx\_MN} \geq C\_DLL\_T\_IFG$   | --                    |

Tab. 4 POWERLINK cycle timing parameters

*Hint: Tab. 4 does not differentiate between frame size and the time required to transmit the frame. Both quantities shall be regarded to be equivalent. The transmission time shall be calculated from the size by multiply with transmit time per byte.*

*When regarding the supporting indices, the frame type specific protocol overhead in frame size shall be considered by time calculation (see 4.6).*

#### 4.2.4.3.1 POWERLINK Cycle Timing Error Handling

POWERLINK cycle timing may be safeguarded at runtime by the node located verifications claimed by Tab. 4. Tab. 5 provides error codes to be issued, when the respective verification fails and the handling of the errors.

All errors shall be stored at the local error history (see 6.5). Errors identified by the CN shall be entered to the Emergency Queue of the Error Signaling. Mode of all errors shall be  $3_h$  (refer 6.5.1). The errors shall be assigned to the POWERLINK profile ( $002_h$ ).

| Tab. 4<br>No | Verification  | Error Code         | Error History<br>Additional Information | Error Handling  |
|--------------|---|--------------------|---|---|
| 1            | MN may verify before isochronous access to CN <sub>n</sub> :<br>$t_{cycle\_MN} = t_{cycle\_CNn}$  | E_NMT_CYCLE_LEN    | CN <sub>n</sub> Node ID                 | do not access CN <sub>n</sub> isochronously   |
| 8            | MN may verify at start up:<br>$t_{PReqPL\_CNn\_MN} \leq t_{PReqPLMax\_MN}$  | E_NMT_PREQ_LIM     | --                                      | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2                                |
| 8            | MN may verify before isochronous access to CN <sub>n</sub> :<br>$t_{PReqPL\_CNn\_MN} = t_{PReqPL\_CNn}$   | E_NMT_PREQ_CN      | CN <sub>n</sub> Node ID                 | do not access CN <sub>n</sub> isochronously   |
| 10           | CN <sub>n</sub> may verify at start up:<br>$t_{PReqPL\_CNn} \leq t_{PReqPLMax\_CNn}$  | E_NMT_PREQ_LIM     | --                                      | do not change NMT state to<br>NMT_CS_PRE_OPERATIONAL_2                                |
| 14           | CN <sub>n</sub> may verify at start up:<br>$t_{PResPLTx\_CNn} \leq t_{PResPLTxMax\_CNn}$  | E_NMT_PRES_TX_LIM  | --                                      | do not change NMT state to<br>NMT_CS_PRE_OPERATIONAL_2                                |
| 16           | see Error Handling DLL (see 4.7.6.3)  |                    |   |   |
| 17           | MN may verify at start up:<br>$t_{PResPLRx\_CNn\_MN} \leq t_{PResPLRxMax\_MN}$  | E_NMT_PRES_RX_LIM  | CN <sub>n</sub> Node ID                 | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2                                |
| 17           | MN may verify before isochronous access to CN <sub>n</sub> :<br>$t_{PResPLRx\_CNn\_MN} = t_{PResPLTx\_CNn}$   | E_NMT_PRES_CN      | CN <sub>n</sub> Node ID                 | do not access CN <sub>n</sub> isochronously   |
| 19           | CN <sub>n</sub> may verify at start up:<br>$t_{PResPLRx\_CNm\_CNn} \leq t_{PResPLRxMax\_CNn}$<br>and<br>$t_{PResPLRx\_CNm\_CNn} \geq t_{PResPLTx\_CNm}$ | E_NMT_PRES_RX_LIM  | CN <sub>m</sub> Node ID                 | CN <sub>n</sub> : do not read PRes from CN <sub>m</sub>                               |
| 26           | MN may verify at start up:<br>$t_{PResPLTx\_MN\_MN} \leq t_{PResPLTxMax\_MN}$   | E_NMT_PRES_TX_LIM  | --                                      | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2                                |
| 28           | CN <sub>n</sub> may verify at start up:<br>$t_{PResPLRx\_MN\_CNn} \leq t_{PResPLRxMax\_CNn}$  | E_NMT_PRES_RX_LIM  | MN Node ID                              | CN <sub>n</sub> : do not read PRes from MN  |
| 34           | Node shall verify:<br>$t_{ASndPLTx} \leq t_{ASndMTU\_MN/CNn}$   | E_NMT_ASND_TX_LIM  | --                                      | do not transmit frame   |
| 36           | Node may verify:<br>$t_{ASndMTU\_MN/CNn} \leq t_{ASndPLTxMax\_MN/CNn}$<br>and<br>$t_{ASndMTU\_MN/CNn} \leq t_{ASndPLRxMax\_MN/CNn}$                     | E_NMT_ASND_MTU_LIM | --                                      | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2 or<br>NMT_CS_PRE_OPERATIONAL_2 |
| 36           | MN may verify:<br>$t_{ASndMTU\_MN} = t_{ASndMTU\_CNn}$  | E_NMT_ASND_MTU_DIF | CN <sub>n</sub> Node ID                 | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2 or<br>NMT_CS_PRE_OPERATIONAL_2 |
| 39           | see Error Handling DLL (see 4.7.6.4)  |                    |   |   |

| Tab. 4<br>No | Verification  | Error Code     | Error History<br>Additional Information | Error Handling   |
|--------------|---|----------------|---|--|
| 40           | MN may verify:<br>$t_{idle\_MN} \geq t_{ASndRx-SoC\_MN}$                | E_NMT_IDLE_LIM | --                                      | do not change NMT state to<br>NMT_MS_PRE_OPERATIONAL_2 |
| 40           | CN <sub>n</sub> may verify:<br>$t_{idle\_CNn} \geq t_{ASndRx-SoC\_CNn}$ | E_NMT_IDLE_LIM | --                                      | do not change NMT stateto<br>NMT_CS_PRE_OPERATIONAL_2  |

Tab. 5 POWERLINK cycle timing verification: Error codes and handling

#### 4.2.4.4 Multiplexed Slot Timing

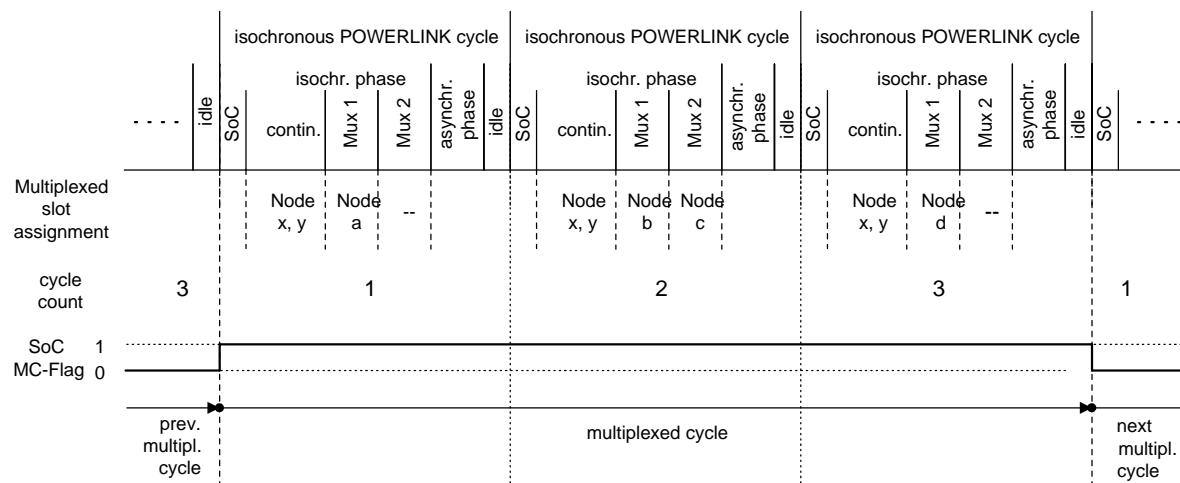


Fig. 28. Multiple slot assignment

Fig. 28 demonstrates the assignment of the multiplexed slots to CNs. The multiplexed slots are identified by "Mux 1" and "Mux 2".

The assignment is controlled by the object dictionary entries

NMT\_CycleTiming\_REC.MultiplCycleCnt\_U8 and NMT\_MultiplCycleAssign\_AU8.

NMT\_CycleTiming\_REC.MultiplCycleCnt\_U8 defines the length of the multiplexed cycle in POWERLINK cycle counts. If NMT\_CycleTiming\_REC.MultiplCycleCnt\_U8 is zero, the multiplexed access method shall not be applied, e.g. all CNs shall be accessed continuously.

The respective sub-index of NMT\_MultiplCycleAssign\_AU8 defines the cycle count inside the multiplexed cycle, when the respective CN shall be polled by the MN. If the sub-index is zero, the CN shall be accessed continuously.

The order in which the CNs are polled by the MN may be set up by object NMT\_IsochrSlotAssign\_AU8.

The number of slots per isochronous cycle can not be programmed directly by the configuration. It is derived from the maximum number of CN assignments that are cumulated to a cycle.

The following table shows the parameter values that control the system in Fig. 28:

|   |   |
|---|---|
| NMT_CycleTiming_REC.MultiplCycleCnt_U8            | 3 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>a</sub> ] | 1 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>b</sub> ] | 2 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>c</sub> ] | 2 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>d</sub> ] | 3 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>x</sub> ] | 0 |
| NMT_MultiplCycleAssign_AU8 [NodeID <sub>y</sub> ] | 0 |

#### 4.2.4.5 CN Cycle State Machine

##### 4.2.4.5.1 Overview

The cycle state machine of the CN (DLL\_CS) handles communication within a POWERLINK Cycle. The DLL\_CS tracks the order of the frames received within a cycle and reacts as described below. The expected order of frame reception is dependant on the NMT\_CS state (see 4.2.4.5.4 )

If an error in the communication is detected by the DLL\_CS, an error event to DLL Error Handling will be generated. The DLL\_CS will attempt to uphold communication regardless of any errors.

#### 4.2.4.5.2 States

- **DLL\_CS\_NON\_CYCLIC**

This state means that the isochronous communication isn't started yet or the connection was lost. It depends on the current state of the NMT\_CS, which events are processed and which will be ignored.

- **DLL\_CS\_WAIT\_SOC**

The state machine waits in this state after receiving the SoA frame until the beginning of the next cycle (triggered by a SoC frame from the MN). Ethernet frames of any type may be received between the SoA and the SoC frames (asynchronous phase).

- **DLL\_CS\_WAIT\_PREQ**

After the beginning of the cycle, the state machine waits in this state for a PReq frame. After PReq reception the CN shall respond with a PRes Frame. The CN may receive and process PRes Frames from other CN whilst in this state.

- **DLL\_CS\_WAIT\_SOA**

After reception of a PReq frame the state machine waits for the reception of a SoA frame.

Reception of a SoA frame confirms the end of the isochronous phase. The CN may receive and process PRes Frames from other nodes whilst in this state.

#### 4.2.4.5.3 Events

- **DLL\_CE\_SOC**

This Event signifies that a POWERLINK SoC frame was received from the MN. It marks the beginning of a new cycle and simultaneously the beginning of the isochronous phase of the cycle.

- **DLL\_CE\_PREQ**

This Event signifies that a POWERLINK PReq frame was received from the MN.

- **DLL\_CE\_PRES**

The CN may be configured to process the PRes frames of other CN's (cross traffic). Every time a PRes frame is received, a DLL\_CE\_PRES event is generated

- **DLL\_CE\_SOA**

This event signifies that a SoA frame was received from the MN. It marks the end of the isochronous phase of the cycle and the beginning of the asynchronous phase.

- **DLL\_CE\_ASND**

This event signifies that an ASnd frame or a non POWERLINK frame has been received. Since the frame types during the asynchronous phase are not limited to POWERLINK types, this event is generated on reception of all legal Ethernet frames.

- **DLL\_CE\_SOC\_TIMEOUT**

This event signifies that a SoC frame of the MN was lost. It occurs, when the SoC timeout supervision detects a missed SoC frame.

#### 4.2.4.5.4 Dependance of the NMT\_CS on the DLL\_CS

The state of the NMT\_CS represents the network state and is used as a qualifier for certain transitions of the DLL\_CS. Because the NMT state influences the behaviour of the DLL\_CS we could filter out the relevant DLL\_CS transitions for a single NMT state, so we see only DLL\_CS transitions which are possible in a distinct NMT state.

*Notation comment:*

*For clarity purposes the NMT\_CS states as conditions for DLL\_CS transitions have been omitted.*

*Because of comprehension and clarity purposes, the relevant transitions of single NMT\_CS states are filtered out and displayed within an own diagram as an "operation mode" of the DLL\_CS. Some*

operation modes are nearly similar, so they are shown within a single figure and the differences are described in the transition table.

#### 4.2.4.5.4.1 State NMT\_GS\_INITIALISATION, NMT\_CS\_NOT\_ACTIVE, NMT\_CS\_BASIC\_ETHERNET, NMT\_CS\_PRE\_OPERATIONAL\_1

In this NMT states the DLL\_CS is in state DLL\_CS\_NON\_CYCLIC.

For description of these NMT states see 7.1.4.

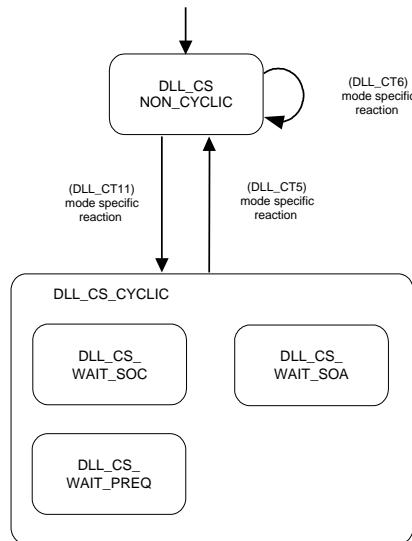


Fig. 29. CN cycle state machine, states NMT\_GS\_INITIALISATION, NMT\_CS\_NOT\_ACTIVE, NMT\_CS\_PRE\_OPERATIONAL\_1, NMT\_CS\_BASIC\_ETHERNET

##### 4.2.4.5.4.1.1 Transitions in other NMT states

|                     |   |
|---------------------|---|
| DLL_CT6,<br>DLL_CT5 | DLL_CE_* [ ] / NMT state specific reaction<br><br>The cycle state machine is not active in the NMT states NMT_GS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PRE_OPERATIONAL_1 and NMT_CS_BASIC_ETHERNET. This means, after the initial transition to DLL_CS_NON_CYCLIC its state does not influence the reaction of the CN. The reactions are defined by the state of the NMT_CS only. (see 7.1.4) |
| DLL_CT11            | DLL_CE_* [ ] / NMT state specific reaction<br><br>This transition is triggered by the NMT state machine when changing from NMT_CS_PRE_OPERATIONAL_1 to NMT_CS_PRE_OPERATIONAL_2 (NMT_CT4)   |

Tab. 6 Transitions for CN cycle state machine, states NMT\_GS\_INITIALISATION, NMT\_CS\_NOT\_ACTIVE, NMT\_CS\_PRE\_OPERATIONAL\_1, NMT\_CS\_BASIC\_ETHERNET

**4.2.4.5.4.2 State NMT\_CS\_PRE\_OPERATIONAL\_2,  
NMT\_CS\_READY\_TO\_OPERATE, NMT\_CS\_OPERATIONAL,  
NMT\_CS\_STOPPED**

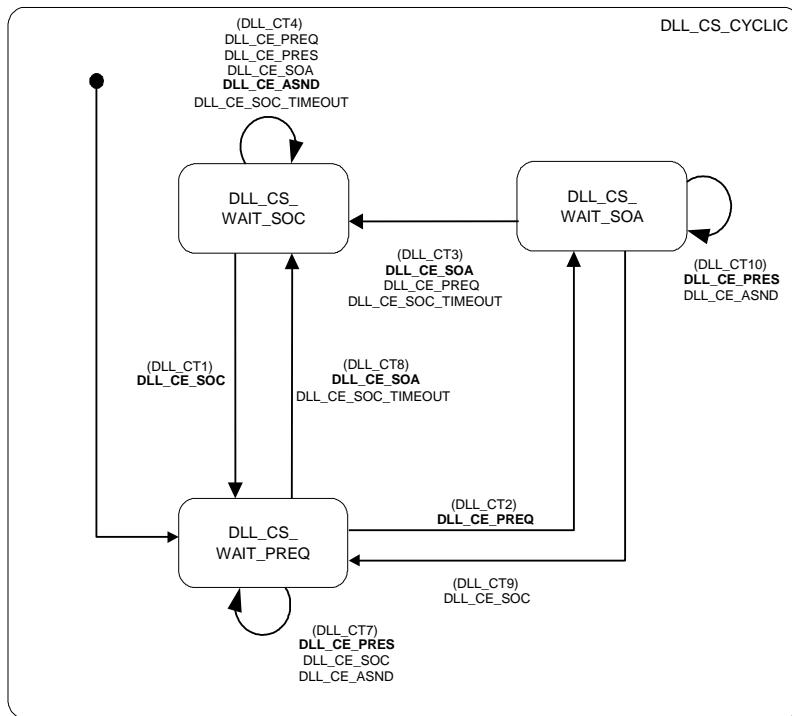


Fig. 30. CN cycle state machine (DLL\_CS), valid for NMT\_CS\_PRE\_OPERATIONAL\_2, NMT\_CS\_READY\_TO\_OPERATE, NMT\_CS states NMT\_CS\_OPERATIONAL

In the NMT\_CS\_OPERATIONAL and NMT\_CS\_READY\_TO\_OPERATE states, there are three mandatory frames for a non-multiplexed node, which shall occur each cycle in the specified order: SoC, PReq and SoA. If a node is accessed multiplexed, only the SoC and SoA frames are mandatory for every cycle. The PReq frame is only mandatory in the multiplexed cycle the node was configured for.

In the NMT\_CS\_PRE\_OPERATIONAL\_2 state, there are two mandatory frames, which shall occur each cycle in the order SoC and SoA. The PReq frame may occur between SoC and SoA.

In the NMT\_CS\_PRE\_OPERATIONAL\_2 state the timeout detection of the SoC is disabled because the node may not be configured yet.

The cycle state machine keeps track of all receive frames to detect frame losses. Receive frames shall be accepted by the CN independent of the current cycle state machine state.

#### 4.2.4.5.4.2.1 Transitions

|         |  |
|---------|--|
| DLL_CT1 | DLL_CE_SOC [ ] / synchronise the start of cycle and generate a SoC trigger to the application<br>The occurrence of the SoC event indicates the beginning of a new POWERLINK cycle. The asynchronous phase of the previous cycle ends and the isochronous phase of the next cycle begins. |
| DLL_CT2 | DLL_CE_PREQ [ ] / Process the PReq frame and send a PRes frame<br>The PReq event occurs within the isochronous phase of communication.   |

|          |   |
|----------|---|
| DLL_CT3  | <p>DLL_CE_SOAs [ ] / process SoA, if allowed send an ASnd frame or a non POWERLINK frame<br/>         DLL_CE_SOC_TIMEOUT [CN NMT state != NMT_CS_PRE_OPERATIONAL_2] / Synchronise to the next SoC, report error DLL_CEV_LOSS_SOC and DLL_CEV_LOSS_SOA<br/>         DLL_CE_PREQ [ ] / accept the PReq frame and send a PRes frame, report error DLL_CEV_LOSS_SOC and DLL_CEV_LOSS_SOA</p> <p>The DLL_CE_SOAs event denotes the end of the isochronous phase and the beginning of the asynchronous phase of the current cycle. If the SoA frame includes an invitation to the CN, the CN may respond with one valid frame.<br/>         The occurrence of a DLL_CE_PREQ signifies that the expected SoA and SoC frames were lost. The DLL Error Handling shall be notified.<br/>         In case of a DLL_CE_SOC_TIMEOUT event happened in NMT_CS_READY_TO_OPERATE or NMT_CS_OPERATIONAL, SoA and SoC frames may have been lost. The DLL Error Handling shall be notified.</p>  |
| DLL_CT4  | <p>DLL_CE_ASND [ ] / process frame<br/>         DLL_CE_PREQ [ ] / respond with PRes frame, report error DLL_CEV_LOSS_SOC<br/>         DLL_CE_PRES [ ] / report error DLL_CEV_LOSS_SOA<br/>         DLL_CE_SOAs [ ] / report error DLL_CEV_LOSS_SOC<br/>         DLL_CE_SOC_TIMEOUT [CN NMT state != NMT_CS_PRE_OPERATIONAL_2] / report error DLL_CEV_LOSS_SOC</p> <p>If an ASnd frame has been received it shall be processed. The state shall not be changed. Although only one asynchronous frame per cycle is allowed, the state machine of the CN does not limit the amount of received frames within the asynchronous phase of the cycle.<br/>         If a SoA, PReq or PRes frame is received, there may be a loss of a SoC frame in between. The DLL Error Handling shall be notified with the error DLL_CEV_LOSS_SOC. This event shall be triggered once per cycle only.<br/>         In case of a DLL_CE_SOC_TIMEOUT event happened in NMT_CS_READY_TO_OPERATE or NMT_CS_OPERATIONAL, SoA and SoC frames may have been lost. The DLL Error Handling shall be notified.<br/>         If a PReq frame was received, the incoming data may be ignored and a PRes frame shall be sent.</p>  |
| DLL_CT7  | <p>DLL_CE_PRES [ ] / process PRes frames (cross traffic)<br/>         DLL_CE_SOC [ ] / synchronise to the cycle begin, report error DLL_CEV_LOSS_SOA<br/>         DLL_CE_ASND [ ] / report error DLL_CEV_LOSS_SOA</p> <p>If a PRes frame of another CN was received (cross traffic), the PRes frame shall be processed (if configured to do so). The CN waits for either a PRes from another CN (cross traffic) or a PReq frame. The reaction to a SoC frame is state independent. The state machine synchronises to the start of a new cycle.<br/>         ASnd frames and non POWERLINK frames shall be processed during the isochronous phase.</p>   |
| DLL_CT8  | <p>DLL_CE_SOAs [CN NMT state = NMT_CS_STOPPED] / process SoA, if invited, transmit a legal Ethernet frame<br/>         DLL_CE_SOAs [CN = multiplexed] / process SoA; if invited, transmit a legal Ethernet frame<br/>         DLL_CE_SOAs [CN != multiplexed] / process SoA; if invited, transmit a legal Ethernet frame, additionally report error DLL_CEV_LOSS_PREQ<br/>         DLL_CE_SOC_TIMEOUT [CN NMT state != NMT_CS_PRE_OPERATIONAL_2] / Synchronise on the next SoC, report error DLL_CEV_LOSS_SOC and DLL_CEV_LOSS_SOA</p> <p>If the CN is in the NMT_CS_OPERATIONAL or NMT_CS_READY_TO_OPERATE the CN will assume a LOSS_OF_PREQ if the number of cycles since the last PReq is greater than that expected. (1 for non multiplexed CNs, n for multiplexed CNs where n is NMT_CycleTiming_REC.MultiplexCycleCnt_U8)<br/>         In case of a DLL_CE_SOC_TIMEOUT event happened in NMT_CS_READY_TO_OPERATE; NMT_CS_OPERATIONAL or NMT_CS_STOPPED, SoA and SoC frames may have been lost. On non-stopped and non-multiplexed nodes or if a multiplexed node should have been requested this cycle, the PRes frame was additionally lost. The DLL Error Handling shall be notified.</p> |
| DLL_CT9  | <p>DLL_CE_SOC [ ] / synchronise on the SoC, report error DLL_CEV_LOSS_SOA</p> <p>The reaction on reception of a SoC is independent of the NMT state, the state machine assumes that an expected frame was lost and (re-)synchronises on the SoC.</p>  |
| DLL_CT10 | <p>DLL_CE_PRES [ ] / process PRes frames (cross traffic)<br/>         DLL_CE_ASND [ ] / report error DLL_CEV_LOSS_SOA</p> <p>The CN may process PRes of other CNs.<br/>         ASnd frames and non POWERLINK frames shall be accepted during the isochronous phase.</p>  |

Tab. 7 Transitions for CN cycle state machine, states NMT\_CS\_OPERATIONAL, NMT\_CS\_PRE\_OPERATIONAL\_2, NMT\_CS\_READY\_TO\_OPERATE

Common issues:

- Loss of frames will be detected when an unexpected frame was received or the DLL\_CE\_SOC\_TIMEOUT occurs.
- The Cycle State machine informs the NMT\_CS of an error, which will then be processed by the NMT\_CS (see 7.1.4).
- DLL\_CEV\_LOSS\_SOA and DLL\_CEV\_LOSS\_PREQ are optional and may be omitted if not supported by the DLL Error Handling
- The DLL error handling shall be notified only once for every real error event although the same error may be detected more often in a cycle.  
e.g. DLL\_CEV\_LOSS\_SOC in DLL\_CT4

## 4.2.4.6 MN Cycle State Machine

### 4.2.4.6.1 Overview

The cycle state machine of the MN (DLL\_MS) shall manage the communication within a POWERLINK cycle.

The DLL\_MS generates the flow of the frames during a POWERLINK cycle and monitors the reaction of the CNs. The flow order is NMT\_MS state dependent (see 4.2.4.6.4 ).

Usually the CNs are synchronised by the reception of the SoC. This means the most significant parameter for the synchronisation of the POWERLINK network is the timing accuracy of the event DLL\_ME\_SOC\_TRIG.

If an error in the communication is detected by the DLL\_MS, an error event to DLL Error Handling will be generated.

### 4.2.4.6.2 States

- **DLL\_MS\_NON\_CYCLIC**

This state means that the cyclic communication is not started yet or was stopped by the NMT\_MS state machine (NMT state NMT\_MS\_PRE\_OPERATIONAL\_1). The state machine waits here until the NMT state changes to NMT\_MS\_PRE\_OPERATIONAL\_2. It depends on the current NMT state, which events will be processed and which will be ignored.

- **DLL\_MS\_WAIT\_SOC\_TRIG**

If the communication of the cycle is finished, the state machine remains in this state until the next cycle begins with a DLL\_ME\_SOC\_TRIG.

- **DLL\_MS\_WAIT\_PRES**

After the sending of the PReq frame the state machine waits in this state for a response. The waiting time is limited by a timeout.

- **DLL\_MS\_WAIT\_ASND**

If a SoA with an Invite is sent, the state machine waits in this state until the asynchronous phase ends with the event DLL\_ME\_SOC\_TRIG.

In DLL\_MS\_NON\_CYCLIC the event DLL\_ME\_SOA\_TRIG shall be generated instead of DLL\_ME\_SOC\_TRIG.

If a ASnd is expected and the timeout NMT\_MNCycleTiming\_REC.AsyncSlotTimeout\_U32 occurs, the error DLL\_MEV\_SOA\_TIMEOUT shall be generated.

- **DLL\_MS\_WAIT\_SOA**

If a SoA with an Invite is sent that is not to be answered, the MN waits in this state until the timeout of the async phase elapsed or any Ethernet frame was received before the next reduced POWERLINK cycle starts.

### 4.2.4.6.3 Events

The DLL\_MS is triggered by events which are generated by an event handler. The DLL\_MS has an interface to:

- the hardware
- the NMT state machine

The event handler should serialize the events (it's possible that a timeout occurs simultaneously with an Ethernet frame receiving). The implementation of the interface to the hardware is out of the scope of this specification.

- **DLL\_ME\_PRES**

This event signifies that a PRes frame was received.

- **DLL\_ME\_PRES\_TIMEOUT**

This event is produced when the PRes frame was not (or not completely) received within a preconfigured time.

- **DLL\_ME\_ASND**

This event means that an ASnd frame or a non POWERLINK frame was received.

- **DLL\_ME\_ASND\_TIMEOUT**

This event is produced when the ASnd frame was not (or not completely) received within a preconfigured time.

- **DLL\_ME\_SOC\_TRIG**

This event triggers emission of the SoC frame and starts a new POWERLINK cycle. The timing accuracy determines the synchronisation accuracy of the POWERLINK network.

- **DLL\_ME\_SOA\_TRIG**

This event means that a new reduced POWERLINK cycle shall start. The event can either be generated cyclically or directly after the reception of a requested ASnd message to continue the reduced POWERLINK cycle as fast as possible.

#### 4.2.4.6.4 Usage of the NMT\_MS state by the DLL\_MS

The state of the NMT\_MS represents the network state and is used as a condition in some transitions of the DLL\_MS. Relevant DLL\_MS transitions for a single NMT state could be filtered out.

A notation comment:

The transitions of DLL\_MS could be displayed within a single diagram where the states of the NMT\_MS are conditions for the transitions. Because of comprehension and clarity purposes, the relevant transitions of single NMT\_MS states are filtered out and displayed within an own diagram as an "operation mode" of the DLL\_MS.

##### 4.2.4.6.4.1 State NMT\_GS\_INITIALISATION, NMT\_MS\_NOT\_ACTIVE

In these states the MN cycle state machine is not active. This means, prior to the initial transition to DLL\_MS\_NON\_CYCLIC its state does not influence the reaction of the MN. The reactions are defined by the state of the NMT\_MS.

##### 4.2.4.6.4.2 NMT\_MS\_BASIC\_ETHERNET

In this state the cycle state machine is not active. This means, prior to the initial transition to DLL\_MS\_NON\_CYCLIC its state does not influence the reaction of the MN. The reactions are defined by the state of the NMT\_MS.

##### 4.2.4.6.4.3 State NMT\_MS\_PRE\_OPERATIONAL\_1

In the NMT\_MS\_PRE\_OPERATIONAL\_1 state, the cycle state machine of the MN generates the Reduced POWERLINK Cycle and observes the behaviour of the CNs. Error handling is described in chapter 4.6. The DLL\_MS is in the DLL\_MS\_NON\_CYCLIC mode.

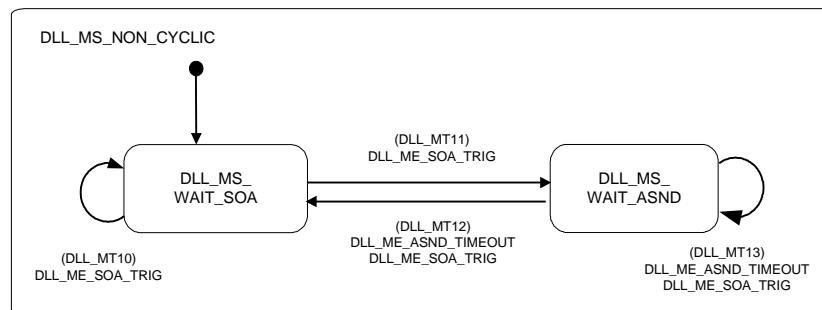


Fig. 31. MN cycle state machine, state NMT\_MS\_PRE\_OPERATIONAL\_1

#### 4.2.4.6.4.3.1 Transitions

|          |  |
|----------|--|
| DLL_MT10 | <p>DLL_ME_SOA_TRIG [async_in != 0 &amp; no resp. expected] / send SoA with Invite<br/>         DLL_ME_SOA_TRIG [async_in = 0 &amp; async_out != 0] / send SoA with Invite to MN and send ASnd or non POWERLINK frame<br/>         DLL_ME_SOA_TRIG [async_in = 0 &amp; async_out = 0] / send SoA</p> <p>If there is an asynchronous transmission request from a CN, the MN sends a SoA message with Invite. The next reduced POWERLINK cycle shall start after the asynchronous slot timeout elapsed or any frame is received.<br/>         If there is outgoing asynchronous MN communication to be done in the current cycle, the MN sends this frame directly after the SoA frame with an Invite to the MN itself. The next reduced POWERLINK cycle shall start after transmission of the ASnd or non POWERLINK frame.<br/>         If there is no Invite for a CN and no ASnd or non POWERLINK frame from MN the next DLL_ME_SOA_TRIG shall be generated after the asynchronous slot timeout.</p> |
| DLL_MT11 | <p>DLL_ME_SOA_TRIG [async_in != 0 &amp; resp. expected] / send SoA with Invite</p> <p>After sending the SoA invite message to the CN, the MN changes to the state DLL_MS_WAIT_ASND to detect transmission timeouts.</p>  |
| DLL_MT12 | <p>DLL_ME_SOA_TRIG   DLL_ME_ASND_TIMEOUT [async_in = 0] / send SoA, ASnd if available<br/>         DLL_ME_SOA_TRIG   DLL_ME_ASND_TIMEOUT [async_in != 0 &amp; no resp. expected] / send SoA with Invite</p> <p>The waiting time ends with either a DLL_ME_SOA_TRIG or a DLL_ME_ASND_TIMEOUT (configurable via NMT_MNCycleTiming_REC.AsyncSlotTimeout_U32). If a certain CN shall be invited, the MN sends a SoA containing an invite for the node. If there is no CN to be invited, the MN sends a SoA. If there is outgoing asynchronous communication to be done in the current cycle, the MN sends this frame after the SoA frame. Afterwards the state changes to DLL_MS_WAIT_SOA.<br/>         The error event DLL_MEV_ASND_TIMEOUT shall be generated.</p>   |
| DLL_MT13 | <p>DLL_ME_SOA_TRIG   DLL_ME_ASND_TIMEOUT [async_in != 0 &amp; resp. expected] / send SoA with Invite, report error DLL_MEV_ASND_TIMEOUT</p> <p>A SoA message containing an Invite for a CN is transmitted. The MN stays in the state MS_WAIT_ASND as an answer of the CN is expected.<br/>         The timeout event DLL_ME_ASND_TIMEOUT (configurable via NMT_MNCycleTiming_REC.AsyncSlotTimeout_U32) shall generate DLL_MEV_ASND_TIMEOUT.</p>  |

Tab. 8 Transitions for MN cycle state machine, state NMT\_MS\_PRE\_OPERATIONAL\_1

Abbreviations used in the transition table:

- „no resp. expected“ means that there is no answer expected upon the async invite of the MN (e.g. unicast message exchange)
- „async\_in != 0“ means that an invite must be sent in this cycle and an ASnd or a non POWERLINK frame could be received.
- „async\_out != 0“ means that an ASnd must be send in this cycle after a SoA was sent.

Common issues:

- DLL\_MEV\_ASND\_TIMEOUT is optional and may be omitted if not supported by the DLL Error Handling
- If a StatusRequest was sent and the asynchronous slot time expires without receiving a StatusResponse frame the DLL Error Handling will be notified with the error DLL\_MEV\_LOSS\_STATUSRESPONSE.

#### 4.2.4.6.4.4 State NMT\_MS\_OPERATIONAL, NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_PRE\_OPERATIONAL\_2

In the NMT\_MS\_OPERATIONAL, NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_PRE\_OPERATIONAL\_2 state, the cycle state machine of the MN generates the POWERLINK Cycle and observes the behaviour of the CNs. Error handling is described in chapter 4.6.

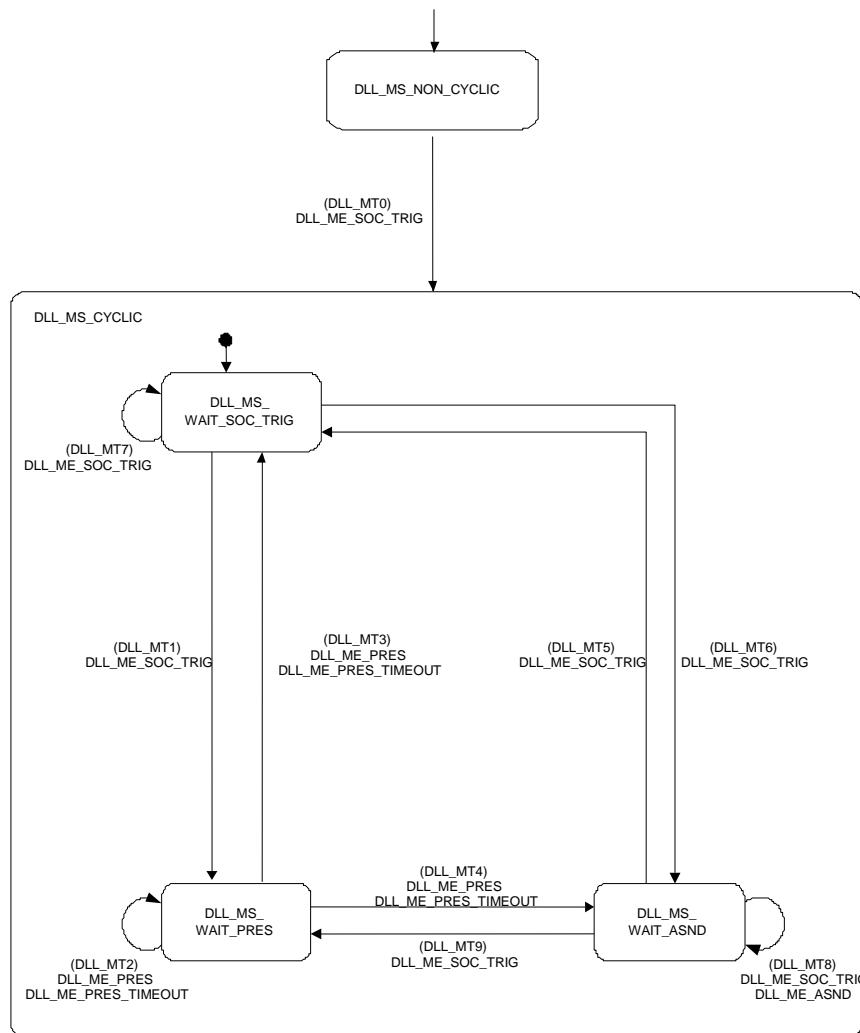


Fig. 32. MN cycle state machine, States NMT\_MS\_OPERATIONAL, NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_PRE\_OPERATIONAL\_2

##### 4.2.4.6.4.4.1 Transitions

|         |  |
|---------|--|
| DLL_MT0 | DLL_ME_SOC_TRIG [ ] / go to state DLL_MS_WAIT_SOC_TRIG<br><br>If the NMT state machine of the MN (NMT_MS) changes the mode to NMT_MS_PRE_OPERATIONAL_2 the DLL_MS shall start the cycle timer (value configurable by NMT_CycleLen_U32), which generates the DLL_ME_SOC_TRIG. The DLL_MS shall prepare the system for the start of the first cycle.                           |
| DLL_MT1 | DLL_ME_SOC_TRIG [ isochr != 0 ] / send SoC, PReq<br><br>Immediately after DLL_ME_SOC_TRIG event occurred a SoC frame is sent, the communication with the NMT state machine will be done. If there are isochronous frames to send, the first PReq will be sent and a timer will be started to observe the response time (configurable via NMT_MNCNPResTimeout_AU32[Node ID]). |
| DLL_MT2 | DLL_ME_PRES [ isochr != 0 ] / send next PReq<br>DLL_ME_PRES_TIMEOUT [ isochr != 0 ] / send next PReq, report error DLL_MEV_LOSS_PRES<br><br>The waiting time ends with either a DLL_ME_PRES or a DLL_ME_PRES_TIMEOUT. The MN sends the next PReq if more frames in the isochronous queue exist. The state does not change.   |

|         |  |
|---------|--|
| DLL_MT3 | <p>DLL_ME_PRES [ isochr = 0 &amp; async_in = 0 ] / send PRes [isochr_out != 0], SoA and ASnd [async_out != 0]<br/>         DLL_ME_PRES_TIMEOUT [ isochr = 0 &amp; async_in = 0 ] / send PRes [isochr_out != 0], SoA and ASnd [async_out != 0], report error DLL_MEV LOSS PRES</p> <p>The isochronous phase ends with either a DLL_ME_PRES or a DLL_ME_PRES_TIMEOUT (configurable via NMT_MNCNPResTimeout_AU32[Node ID]). If there is no more communication to be done (neither isochronous nor asynchronous), the MN sends a SoA frame and changes the state to DLL_MS_WAIT_SOC_TRIG.<br/>         If there is outgoing asynchronous communication to be done in the current cycle, the MN sends this frame after the SoA.</p> |
| DLL_MT4 | <p>DLL_ME_PRES [ isochr = 0 &amp; async_in != 0 ] / send PRes [isochr_out != 0] and SoA with Invite<br/>         DLL_ME_PRES_TIMEOUT [ isochr = 0 &amp; async_in != 0 ] / send PRes [isochr_out != 0] and SoA with Invite, report error DLL_MEV LOSS PRES</p> <p>The isochronous phase ends with either a DLL_ME_PRES or a DLL_ME_PRES_TIMEOUT (configurable via NMT_MNCNPResTimeout_AU32[Node ID]).<br/>         If the MN is configured to send a PRes this frame shall be sent before the SoA.<br/>         If the scheduled asynchronous communication for the current cycle is directed from the CN to the MN or another CN, an invite within the SoA frame will be sent.</p>   |
| DLL_MT5 | <p>DLL_ME_SOC_TRIG [ isochr = 0 &amp; async_in = 0 ] / send SoC, PRes [isochr_out != 0], SoA and ASnd [async_out != 0]</p> <p>Immediately after the DLL_ME_SOC_TRIG event (value configurable via NMT_CycleLen_U32) a SoC frame will be sent, the communication with the NMT state machine will be done. If there is no communication to be done, then a SoA frame is additionally sent. The state does not change.<br/>         If the MN is configured to send a PRes this frame shall be sent before the SoA.<br/>         If there is outgoing asynchronous communication to be done in the current cycle, the MN sends this frame after the SoA frame and changes the state to DLL_MS_WAIT_SOC_TRIG.</p>                  |
| DLL_MT6 | <p>DLL_ME_SOC_TRIG [ isochr = 0 &amp; async_in != 0 ] / send SoC, PRes [isochr_out != 0] and SoA with Invite</p> <p>Immediately after the DLL_ME_SOC_TRIG (value configurable via NMT_CycleLen_U32) a SoC frame will be sent. Then, the communication with the NMT state machine will be done.<br/>         If the MN is configured to send a PRes this frame shall be sent before the SoA.<br/>         If there are only asynchronous frames to send, the SoA frame will be send. If the asynchronous communication is directed to a CN, an ASnd frame will be sent additionally.</p>  |
| DLL_MT7 | <p>DLL_ME_SOC_TRIG [ isochr = 0 &amp; async_in = 0 ] / send SoC, PRes [isochr_out != 0], SoA and ASnd [async_out != 0]</p> <p>Immediately after the DLL_ME_SOC_TRIG event a SoC frame will be sent, the communication with the NMT state machine will be done.<br/>         If the MN is configured to send a PRes this frame shall be sent before the SoA.<br/>         If there is no communication to be done, then a SoA frame is additionally sent. The state does not change. If there is outgoing asynchronous communication to be done in the current cycle, the MN sends this frame after the SoA.</p>  |
| DLL_MT8 | <p>DLL_ME_ASND [ ] / process the frame<br/>         DLL_ME_SOC_TRIG [ isochr = 0 &amp; async_in != 0 ] / send SoC and SoA with Invite</p> <p>Immediately after the DLL_ME_SOC_TRIG a SoC frame will be sent. Then, the communication with the NMT state machine will be done. If there are only asynchronous frames to send, the SoA frame will be send. If the asynchronous communication is directed to a CN, an ASnd frame will be sent additionally.</p>   |
| DLL_MT9 | <p>DLL_ME_SOC_TRIG [ isochr != 0 ] / send SoC and PReq</p> <p>Immediately after DLL_ME_SOC_TRIG event occurred (value configurable via NMT_CycleLen_U32) a SoC frame is sent, the communication with the NMT state machine will be done. If there are isochronous frames to send, the first PReq will be sent and a timer will be started to observe the response time (configurable via NMT_MNCNPResTimeout_AU32[Node ID]).</p>   |

Tab. 9      Transitions for MN cycle state machine, states NMT\_MS\_OPERATIONAL,  
 NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_PRE\_OPERATIONAL\_2

Abbreviations used in the transition table:

- „isochr != 0“ means that there are frames in the isochronous list, which must be send during the current cycle.
- „isochr\_out != 0“ means that the MN is configured to send a PRes
- „async\_in != 0“ means that an invite must be sent in this cycle and an ASnd or a non POWERLINK frame could be received.

- “async\_out != 0” means that an ASnd must be send in this cycle after a SoA was sent. The reaction on unexpected events is not described in the above figures and tables because of clarity purposes. A general statement for these events can be given:
  - The unexpected frame types and unexpected sender shall be accepted. The state does not change. The PRes frames shall be passed to the NMT state machine, which may analyse this frames (and e.g. remove the corresponding CN from the communication). The state machine does not react in any other way to this event.
  - If the DLL\_MS receives frames, which can be sent by another MN only (SoC, SoA, PReq), it shall notify the NMT state machine and the DLL Error Handling.
  - If an unexpected internal event (e.g. timeout) occurs, an internal error will be assumed and the NMT\_MS will be notified.
  - It could happen that the DLL\_ME\_SOC\_TRIG occurs in states where it was not expected or during transmission or reception of Ethernet frames. (e.g. malconfiguration of PRes timeouts). In this case the DLL Error Handling will be notified with the error DLL\_MEV\_CYCLE\_EXCEED

If a StatusRequest was sent and the asynchronous slot time expires without receiving a StatusResponse frame the DLL Error Handling will be notified with the error DLL\_MEV\_LOSS\_STATUSRESPONSE.

## 4.2.5 Recognizing Active Nodes

The MN shall be configured with a list of all nodes on the network.

All configured nodes shall be marked as inactive when the MN boots. Configured but inactive CNs shall be periodically accessed by an IdentRequest, a special form of the SoA frame. When a CN receives an IdentRequest addressed to itself, it shall return an IdentResponse, a special form of an ASnd frame, in the same asynchronous phase.

The CN shall be marked as active if the MN receives an IdentResponse from the CN. An active CN may take part in the isochronous data transfer, e.g. it may be accessed via a PReq.

## 4.3 Basic Ethernet Mode

Network communication behaves according to the rules of the Legacy Ethernet (IEEE 802.3). The network medium is accessed according to CSMA/CD.

The network communication is collision-prone and non-deterministic.

In the Basic Ethernet Mode any protocol on top of Legacy Ethernet can be used. Data between the nodes are preferentially exchanged via UDP/IP and TCP/IP. The large extension of the maximum topology of an Ethernet POWERLINK Network conflicts with the topology rules of IEEE 802.3. Due to this fact, CSMA/CD might work poorly in large POWERLINK networks. Higher layer protocols shall be applied to handle communication errors caused by collisions unresolved by CSMA/CD.

POWERLINK nodes shouldn't operate in Basic Ethernet Mode, when the node is part of a automation system. Basic Ethernet Mode is provided for point to point configurations, to be used for node setup and service purpose only.

## 4.4 MAC Addressing

A POWERLINK node must support unicast, multicast and broadcast Ethernet MAC addressing in accordance with IEEE802.3.

### 4.4.1 MAC Unicast

The high-order bit of the MAC address is 0 for ordinary addresses (unicast). The unicast addresses used for POWERLINK shall be globally unique, or at least unique within the POWERLINK segment.

### 4.4.2 MAC Multicast

For group addresses the high-order bit of the MAC address is 1. Group addresses allow multiple nodes to listen to a single address. When a frame is sent to a group address, all the nodes registered for this group address receive it. Sending to a group of nodes is called multicast.

The following MAC-multicast addresses shall be used:

|   | <b>MAC-Multicast address</b> |
|---|------------------------------|
| Start of Cycle (SoC)  | C_DLL_MULTICAST_SOC          |
| PollResponse (PRes)   | C_DLL_MULTICAST_PRES         |
| Start of Asynchronous (SoA)                                     | C_DLL_MULTICAST_SOA          |
| AsynchronousSend (ASnd)   | C_DLL_MULTICAST_ASND         |
| Active Managing Node Indication (AMNI) used by EPSG DS302-A [1] | C_DLL_MULTICAST_AMNI         |

Tab. 10 Assigned multicast addresses

### 4.4.3 MAC Broadcast

The address consisting of all 1 bits is reserved for broadcast.

## 4.5 POWERLINK Addressing

Each POWERLINK node (MN, CN and Router) has a unique Node ID within a POWERLINK segment. The number 240 is permanently assigned to the MN. A node set to 240 Node ID operates as the MN, if the node has MN functionality. Devices with pure CN function cannot be assigned the Node ID 240. POWERLINK Node IDs 1-239 may be used for the CNs. Tab. 11 shows the POWERLINK Node ID assignment and allowed CN access options for the POWERLINK Node ID intervals.

The POWERLINK Node ID is either configured by the application process or is set on the device (e.g. using address switches).

The terms unicast, multicast and broadcast refer to POWERLINK addresses if not otherwise mentioned. If the POWERLINK broadcast address is used, the frame shall be sent with the dedicated MAC multicast address (see Tab. 10). If no MAC multicast address is assigned to this type of POWERLINK frame the MAC broadcast address shall be used instead.

POWERLINK dummy node shall be used to transmit messages addressing none of the existing node. POWERLINK dummy Node ID shall never be occupied by an existing node. No node shall expect valid data from the POWERLINK dummy node.

The self addressing Node ID shall never be assigned to any node. It may be used only by PDO mapping to indicate reception and evaluation of a PDO transporting PRes frame, transmitted by the receiving node itself.

Diagnostic device shall be a default node, available without any configuration. Its default access options shall be optional and async only. By configuration isochronous operation may be declared.

A device may support less than the maximum number of regular CNs defined by this specification. The number of supported regular CNs may be provided by the device description entry D\_NMT\_MaxCNNumber\_U8. The upper limit of the interval of POWERLINK Node IDs available for regular CNs may be reduced by the device description entry D\_NMT\_MaxCNNodeID\_U8.

If only D\_NMT\_MaxCNNumber\_U8 is provided the Node IDs may be selected from the complete interval 1 .. 239. If D\_NMT\_MaxCNNodeID\_U8 is additionally provided, valid regular CN Node IDs shall be selected from the interval 1 .. D\_NMT\_MaxCNNodeID\_U8.

| <b>POWERLINK Node ID</b> |                        | <b>Description</b>  | <b>access options</b>                                 |
|--------------------------|------------------------|---|---|
| 0                        | C_ADR_INVALID          | Invalid   | no  |
| 1 .. 239                 |                        | regular POWERLINK CNs   | no / mandatory / optional<br>isochronous / async only |
| 240                      | C_ADR_MN_DEF_NODE_ID   | POWERLINK MN  | mandatory<br>isochronous                              |
| 241 .. 250               |                        | Reserved.<br>Used by EPSG DS302-A [1]                           | no  |
| 251                      | C_ADR_SELF_ADR_NODE_ID | POWERLINK pseudo Node ID to be used by a node to address itself | no  |
| 252                      | C_ADR_DUMMY_NODE_ID    | POWERLINK dummy node  | no  |
| 253                      | C_ADR_DIAG_DEF_NODE_ID | Diagnostic device   | optional<br>isochronous / async only                  |
| 254                      | C_ADR_RT1_DEF_NODE_ID  | POWERLINK to legacy Ethernet Router                             | no / mandatory / optional<br>isochronous              |
| 255                      | C_ADR_BROADCAST        | POWERLINK broadcast   | no  |

Tab. 11 POWERLINK Node ID assignment

## 4.6 Frame Structures

### 4.6.1 Integration with Ethernet

POWERLINK is a protocol residing on top of the standard 802.3 MAC layer.

POWERLINK messages shall be encapsulated in Ethernet II frames. The Ethernet Type (EtherType) field shall be set to 88AB<sub>h</sub>.

The length of the frame shall be restricted to the configured size. Otherwise the cycle time could not be guaranteed. Ethernet frames shall not be shorter than the specified minimum of 64 octets.

#### 4.6.1.1 POWERLINK Frame

To be independent of the underlying protocol, POWERLINK defines its own addressing scheme (refer 4.5) and header format.

##### 4.6.1.1.1 POWERLINK Basic Frame

The POWERLINK Basic Frame format shall contain 5 fields:

- Reserved (1 bit)
- MessageType (7 bits)
- Destination node address (1 octet), POWERLINK addressing scheme (See 4.5)
- Source node address (1 octet), POWERLINK addressing scheme (See 4.5)
- Payload (n octets)

The POWERLINK Basic Frame format shall be encapsulated by the Ethernet II wrapper consisting of 14 octets of leading Ethernet header (Destination and Source MAC addresses, EtherType) and 4 octets of terminating CRC32 checksum.

| Octet Offset <sup>4</sup> | Bit Offset              |             |   |   |   |   |   |   | entry defined by      |  |  |  |  |  |  |  |  |
|---------------------------|-------------------------|-------------|---|---|---|---|---|---|-----------------------|--|--|--|--|--|--|--|--|
|                           | 7                       | 6           | 5 | 4 | 3 | 2 | 1 | 0 |                       |  |  |  |  |  |  |  |  |
| 0 .. 5                    | Destination MAC Address |             |   |   |   |   |   |   | Ethernet type II      |  |  |  |  |  |  |  |  |
| 6 .. 11                   | Source MAC Address      |             |   |   |   |   |   |   |                       |  |  |  |  |  |  |  |  |
| 12 .. 13                  | EtherType               |             |   |   |   |   |   |   |                       |  |  |  |  |  |  |  |  |
| 14                        | res                     | MessageType |   |   |   |   |   |   | Ethernet<br>POWERLINK |  |  |  |  |  |  |  |  |
| 15                        | Destination             |             |   |   |   |   |   |   |                       |  |  |  |  |  |  |  |  |
| 16                        | Source                  |             |   |   |   |   |   |   |                       |  |  |  |  |  |  |  |  |
| 17 .. n                   | Data                    |             |   |   |   |   |   |   |                       |  |  |  |  |  |  |  |  |
| n+1 .. n+4                | CRC32                   |             |   |   |   |   |   |   | Ethernet type II      |  |  |  |  |  |  |  |  |

C\_DLL\_MIN\_PAYL\_OFFSET+14 ≤ n ≤ C\_DLL\_MAX\_PAYL\_OFFSET+14

Tab. 12 Ethernet POWERLINK frame structure

The Ethernet POWERLINK defined part of the Ethernet frame shall be regarded to be the POWERLINK frame.

| Field                   | Abbr. | Description   | Value               |
|-------------------------|-------|---|---------------------|
| Destination MAC Address | dmac  | MAC address of the addressed node(s)  | see 4.4             |
| Source MAC Address      | smac  | MAC address of the transmitting node  | see 4.4             |
| EtherType               | etyp  | Ethernet message type   | C_DLL_ETHERTYPE_EPL |
| MessageType             | mtyp  | POWERLINK message type identification   | see Tab. 14         |
| Destination             | dest  | POWERLINK Node ID of the addressed node(s)  | see 4.5             |
| Source                  | src   | POWERLINK Node ID of the transmitting node  | see 4.5             |
| Data                    | data  | Data depending on MessageType<br>shall be made up to C_DLL_MIN_PAYL_OFFSET bytes<br>by lower layer using padding bytes, if data length is<br>below this limit | refer below         |
| CRC32                   | crc   | CRC32 checksum  |                     |

Tab. 13 Ethernet POWERLINK frame fields

The following message types shall be applied:

| Message Type          | ID / Abbr. | MAC Transfer type |
|-----------------------|------------|-------------------|
| Start of Cycle        | SoC        | Multicast         |
| PollRequest           | PReq       | Unicast           |
| PollResponse          | PRes       | Multicast         |
| Start of Asynchronous | SoA        | Multicast         |
| Asynchronous Send     | ASnd       | Multicast         |

Tab. 14 POWERLINK message types

Refer 4.4.2 for multicast addresses to be used by the respective message type.

Reserved values shall be set to 0.

<sup>4</sup> Octet Offset refers to the start of the Ethernet frame.

#### 4.6.1.1.2 Start of Cycle (SoC)

| Octet Offset <sup>5</sup> | Bit Offset              |             |     |     |     |     |     |     |  |  |  |  |  |  |
|---------------------------|-------------------------|-------------|-----|-----|-----|-----|-----|-----|--|--|--|--|--|--|
|                           | 7                       | 6           | 5   | 4   | 3   | 2   | 1   | 0   |  |  |  |  |  |  |
| 0                         | res                     | MessageType |     |     |     |     |     |     |  |  |  |  |  |  |
| 1                         | Destination             |             |     |     |     |     |     |     |  |  |  |  |  |  |
| 2                         | Source                  |             |     |     |     |     |     |     |  |  |  |  |  |  |
| 3                         | reserved                |             |     |     |     |     |     |     |  |  |  |  |  |  |
| 4                         | MC                      | PS          | res | res | res | res | res | res |  |  |  |  |  |  |
| 5                         | res                     | res         | res |     |     | res |     |     |  |  |  |  |  |  |
| 6 .. 13                   | NetTime / reserved      |             |     |     |     |     |     |     |  |  |  |  |  |  |
| 14 .. 21                  | RelativeTime / reserved |             |     |     |     |     |     |     |  |  |  |  |  |  |
| 22 .. 45                  | reserved                |             |     |     |     |     |     |     |  |  |  |  |  |  |

Tab. 15 SoC frame structure

SoC shall be transmitted using a multicast MAC address (See 4.4.2).

| Field                       | Abbr     | Description  | Value                |
|-----------------------------|----------|--|----------------------|
| MessageType                 | mtyp     | POWERLINK message type identification  | SoC                  |
| Destination                 | dest     | POWERLINK Node ID of the addressed node(s)   | C_ADR_BROADCAST      |
| Source                      | src      | POWERLINK Node ID of the transmitting node   | C_ADR_MN_DEF_NODE_ID |
| Multiplexed Cycle Completed | MC       | Flag: Shall be toggled when the final multiplexed cycle has ended  |                      |
| Prescaled Slot              | PS       | Flag: Shall be toggled by the MN every n-th cycle (n is configurable by NMT_CycleTiming_REC.Prescaler_U16). This prescaled signal is useful for “slow” nodes, which can not react every cycle (The SoC reception shall be signalled to the application every n-th cycle).  |                      |
| NetTime                     | time     | MN may distribute the starting time of the POWERLINK cycle. NetTime shall be of data type NETTIME. NetTime transmission is optional. Support is indicated by D_NMT_NetTime_BOOL. IEEE 1588 conform distribution via NetTime is indicated by D_NMT_NetTimeIsRealTime_BOOL.  |                      |
| RelativeTime                | relti me | MN may distribute a relative time, which is incremented by the cycle time (NMT_CycleLen_U32) when a SoC is generated. RelativeTime shall be set to 0 when NMT state equals NMT_GS_INITIALISING. RelativeTime shall be of data type UNSIGNED64. RelativeTime shall be transmitted in $\mu$ s. RelativeTime transmission is optional. Support is indicated by D_NMT_RelativeTime_BOOL. |                      |

Tab. 16 SoC frame data fields

<sup>5</sup> Octet Offset refers to the start of the POWERLINK frame. Offset to the start of the Ethernet frame is 14 Octets.

### 4.6.1.1.3 PollRequest (PReq)

| Octet Offset <sup>6</sup> | Bit Offset       |                  |     |     |     |     |     |    |  |  |  |  |  |  |
|---------------------------|------------------|------------------|-----|-----|-----|-----|-----|----|--|--|--|--|--|--|
|                           | 7                | 6                | 5   | 4   | 3   | 2   | 1   | 0  |  |  |  |  |  |  |
| 0                         | res              | MessageType      |     |     |     |     |     |    |  |  |  |  |  |  |
| 1                         | Destination      |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 2                         | Source           |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 3                         | res              |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 4                         | res              | res              | MS  | res | res | EA  | res | RD |  |  |  |  |  |  |
| 5                         | res <sup>7</sup> | res <sup>8</sup> | res |     |     | res |     |    |  |  |  |  |  |  |
| 6                         | PDOVersion       |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 7                         | res              |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 8 .. 9                    | Size             |                  |     |     |     |     |     |    |  |  |  |  |  |  |
| 10 .. n                   | Payload          |                  |     |     |     |     |     |    |  |  |  |  |  |  |

$n \leq C\_DLL\_MAX\_PAYL\_OFFSET$

Tab. 17 PReq frame structure

PReq shall be transmitted using the unicast MAC address of the CN (See 4.4.1).

| Field                 | Abbr | Description   | Value                      |
|-----------------------|------|---|----------------------------|
| MessageType           | mtyp | POWERLINK message type identification   | PReq                       |
| Destination           | dest | POWERLINK Node ID of the addressed node(s)  | CN Node ID                 |
| Source                | src  | POWERLINK Node ID of the transmitting node  | C_ADR_MN_DEF_NODE_ID       |
| Multiplexed Slot      | MS   | Flag: Shall be set in PReq frames to CNs that are served by a multiplexed timeslot  |                            |
| Exception Acknowledge | EA   | Flag: Error signaling, refer 6.5.2  |                            |
| Ready                 | RD   | Flag: Shall be set if the transferred payload data are valid. It shall be set by the application process of the MN. A CN shall be allowed to accept data only when this bit is set. |                            |
| PDOVersion            | pdov | Shall indicate the version of the PDO encoding used by the payload data, refer 6.4.2  |                            |
| Size                  | size | Shall indicate the number of payload data octets.   | 0 .. C_DLL_ISOCHR_MAX_PAYL |
| Payload               | pl   | Isochronous payload data sent from the MN to the addressed CN.<br>The lower layer shall be responsible for padding.<br>Payload to be used by PDO, refer 6.3.4                       |                            |

Tab. 18 PReq frame data fields

The PReq POWERLINK PDO message header consists of all components of the PReq Frame besides the payload.

<sup>6</sup> Octet Offset refers to the start of the POWERLINK frame. Offset to the start of the Ethernet frame is 14 Octets.

<sup>7</sup> Used by EPSG DS302-A [1]

<sup>8</sup> Used by EPSG DS302-A [1]

#### 4.6.1.1.4 PollResponse (PRes)

| Octet Offset <sup>9</sup> | Bit Offset        |                   |    |    |     |     |     |    |  |  |  |  |  |  |
|---------------------------|-------------------|-------------------|----|----|-----|-----|-----|----|--|--|--|--|--|--|
|                           | 7                 | 6                 | 5  | 4  | 3   | 2   | 1   | 0  |  |  |  |  |  |  |
| 0                         | res               | MessageType       |    |    |     |     |     |    |  |  |  |  |  |  |
| 1                         | Destination       |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 2                         | Source            |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 3                         | NMTStatus         |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 4                         | res               | res               | MS | EN | res | res | res | RD |  |  |  |  |  |  |
| 5                         | res <sup>10</sup> | res <sup>11</sup> | PR |    |     | RS  |     |    |  |  |  |  |  |  |
| 6                         | PDOVersion        |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 7                         | reserved          |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 8 .. 9                    | Size              |                   |    |    |     |     |     |    |  |  |  |  |  |  |
| 10 .. n                   | Payload           |                   |    |    |     |     |     |    |  |  |  |  |  |  |

$n \leq C\_DLL\_MAX\_PAYL\_OFFSET$

Tab. 19 PRes frame structure

PRes shall be transmitted using the multicast MAC address (See 4.4.2).

| Field            | Abbr | Description   | Value   |
|------------------|------|---|---|
| MessageType      | mtyp | POWERLINK message type identification   | PRes  |
| Destination      | dest | POWERLINK Node ID of the addressed nodes  | C_ADR_BROADCAST                                 |
| Source           | src  | POWERLINK Node ID of the transmitting node  | CN Node ID                                      |
| NMTStatus        | stat | Shall report the current status of the CN's NMT state machine   |   |
| Multiplexed Slot | MS   | Flag: Shall be set in PRes frames from CNs that are served by a multiplexed timeslot. Based on this information, other CNs can identify that the transmitting CN is served by a multiplexed slot      |   |
| Exception New    | EN   | Flag: Error signaling, refer 6.5.2  |   |
| Ready            | RD   | Flag: Shall be set if the transferred payload data are valid.<br>It shall be handled by the application process in the CN. All other CNs and the MN shall be allowed to accept data only if RD is set |   |
| Priority         | PR   | Flags: Shall indicate the priority of the frame in the asynchronous send queue with the highest priority. (See 4.2.4.1.2.2)   | C_DLL_ASND_PRIO_NMTRQST,<br>C_DLL_ASND_PRIO_STD |
| RequestToSend    | RS   | Flags: Shall indicate the number of pending frames in asynchronous send queue on the node. The value C_DLL_MAX_RS shall indicate C_DLL_MAX_RS or more requests, 0 shall indicate no pending requests  | 0 - C_DLL_MAX_RS                                |
| PDOVersion       | pdov | Shall indicate the version of the PDO encoding used by the payload data, refer 6.4.2  |   |
| Size             | size | Shall indicate the number of payload data octets  | 0 .. C_DLL_ISOCHR_MAX_PAYL                      |
| Payload          | pl   | Isochronous payload data sent from the node to the POWERLINK network.<br>The lower layer shall be responsible for padding.<br>Payload to be used by PDO, refer 6.3.4                                  |   |

Tab. 20 PRes frame data fields

The PRes POWERLINK PDO message header consists of all components of the PRes Frame besides the payload.

<sup>9</sup> Octet Offset refers to the start of the POWERLINK frame. Offset to the start of the Ethernet frame is 14 Octets.

<sup>10</sup> Used by EPSG DS302-A [1]

<sup>11</sup> Used by EPSG DS302-A [1]

#### 4.6.1.1.5 Start of Asynchronous (SoA)

| Octet Offset <sup>12</sup> | Bit Offset             |             |     |     |     |        |        |     |  |  |  |  |  |  |
|----------------------------|------------------------|-------------|-----|-----|-----|--------|--------|-----|--|--|--|--|--|--|
|                            | 7                      | 6           | 5   | 4   | 3   | 2      | 1      | 0   |  |  |  |  |  |  |
| 0                          | res                    | MessageType |     |     |     |        |        |     |  |  |  |  |  |  |
| 1                          | Destination            |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 2                          | Source                 |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 3                          | NMTStatus              |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 4                          | res                    | res         | res | res | res | EA/res | ER/res | res |  |  |  |  |  |  |
| 5                          | res                    | res         | res |     |     | res    |        |     |  |  |  |  |  |  |
| 6                          | RequestedServiceID     |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 7                          | RequestedServiceTarget |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 8                          | EPLVersion             |             |     |     |     |        |        |     |  |  |  |  |  |  |
| 9 .. 45                    | reserved               |             |     |     |     |        |        |     |  |  |  |  |  |  |

Tab. 21 SoA frame structure

SoA shall be transmitted using the multicast MAC address 3 (See 4.4.2).

| Field                   | Abbr | Description   | Value                |
|-------------------------|------|---|----------------------|
| MessageType             | mtyp | POWERLINK message type identification   | SoA                  |
| Destination             | dest | POWERLINK Node ID of the addressed nodes  | C_ADR_BROADCAST      |
| Source                  | src  | POWERLINK Node ID of the transmitting node  | C_ADR_MN_DEF_NODE_ID |
| NMTStatus               | stat | Shall report the current status of the MN's NMT state machine   |                      |
| Exception Acknowledge   | EA   | Flag: Error signaling, refer 6.5.2 EA bit shall be valid only, if RequestedServiceID equals StatusRequest.  |                      |
| Exception Reset         | ER   | Flag: Error signaling, refer 6.5.2 ER bit shall be valid only, if RequestedServiceID equals StatusRequest.  |                      |
| Requested ServiceID     | svid | Shall indicate the asynchronous service ID dedicated to the SoA and to the following asynchronous slot (refer below). NO_SERVICE shall indicate that the asynchronous slot is not assigned. | see Tab. 23          |
| Requested ServiceTarget | svtg | Shall indicate the POWERLINK address of the node, which is allowed to send.<br>C_ADR_INVALID shall indicate the asynchronous slot is not assigned.  |                      |
| EPLVersion              | eplv | Shall indicate the current POWERLINK Version of the MN (See Tab. 112).  |                      |

Tab. 22 SoA frame data fields

RequestedServiceID and RequestedServiceTarget are combined to a AsynInvite Command.

##### 4.6.1.1.5.1 RequestedServiceID s

The following values shall be used for the RequestedServiceID entry, indicating the granted asynchronous service:

<sup>12</sup> Octet Offset refers to the start of the POWERLINK frame. Offset to the start of the Ethernet frame is 14 Octets.

| Description / ID                       | Comment  |
|--|--|
| NoService / NO_SERVICE                 | Shall be used if the asynchronous slot is not assigned to any node. RequestedServiceTarget shall be C_ADR_INVALID.   |
| IdentRequest / IDENT_REQUEST           | Shall be used to identify inactive CNs and/or to query the identification data of a CN. The addressed CN shall answer immediately after the reception of the SoA with the node specific IdentRequest frame. The IdentResponse frame is based on the ASnd frame.  |
| StatusRequest / STATUS_REQUEST         | Shall be used to request the current status and detailed error information of a node. Async-only CNs shall be cyclically queried by StatusRequest to supervise their status and to query their requests for the asynchronous slot. The addressed node shall answer immediately after the reception of the SoA, with the node specific StatusRequest frame. The StatusResponse frame is based on the ASnd frame.    |
| NMTRequestInvite / NMT_REQUEST_INVITE  | Shall be used to assign the asynchronous slot to a node that has indicated a pending NMTCommand / NMTRequest by a Request to Send (RS bit of PRes, StatusResponse or IdentResponse) with the priority level PRIO_NMT_REQUEST. The addressed node shall answer immediately after the reception of the SoA with the NMTCommand / NMTRequest frame. The NMTCommand and NMTRequest frames are based on the ASnd frame. |
| Manufacturer specific / MANUF_SVC_IDS  | Shall be used for manufacturer specific purposes.  |
| UnspecifiedInvite / UNSPECIFIED_INVITE | Shall be used to assign the asynchronous slot to a node that has indicated a pending transmit request by a Request to Send (RS bit of PRes, StatusResponse or IdentResponse). The addressed node shall answer immediately after the reception of the SoA, with any kind of a POWERLINK ASnd or a Legacy Ethernet frame.  |

Tab. 23 Definition of the RequestedServiceID in the SoA frame

Assignment of the asynchronous slot to the MN itself shall be indicated in the same way as assignments to CNs.

#### 4.6.1.1.6 Asynchronous Send (ASnd)

| Octet Offset <sup>13</sup> | Bit Offset |   |   |   |   |   |   |             |
|----------------------------|------------|---|---|---|---|---|---|-------------|
|                            | 7          | 6 | 5 | 4 | 3 | 2 | 1 | 0           |
| 0                          | res        |   |   |   |   |   |   | MessageType |
| 1                          |            |   |   |   |   |   |   | Destination |
| 2                          |            |   |   |   |   |   |   | Source      |
| 3                          |            |   |   |   |   |   |   | ServiceID   |
| 4 .. n                     |            |   |   |   |   |   |   | Payload     |

$n \leq C_{DLL\_MAX\_PAYL\_OFFSET}$

Tab. 24 ASnd frame structure

The transmission of an ASnd frame by a node shall occur immediately after the transmission / reception of a SoA frame.

ASnd frames shall be transmitted using a unicast or multicast MAC address (See 4.4). Received ASnd frames having a unicast, multicast or broadcast MAC address shall be accepted.

| Field       | Abbr. | Description  | Value       |
|-------------|-------|--|-------------|
| MessageType | mtyp  | POWERLINK message type identification                            | ASnd        |
| Destination | dest  | POWERLINK Node ID of the addressed node(s)                       |             |
| Source      | src   | POWERLINK Node ID of the transmitting node                       |             |
| ServiceID   | svid  | Shall indicate the service ID dedicated to the asynchronous slot | see Tab. 26 |
| Payload     | pl    | Shall contain data, that are specific for the current ServiceID  |             |

Tab. 25 ASnd frame data fields

#### 4.6.1.1.6.1 ServiceID values

The following values shall be used for the ServiceID entry:

<sup>13</sup> Octet Offset refers to the start of the POWERLINK frame. Offset to the start of the Ethernet frame is 14 Octets.

| Description / ID                      | Comment   |
|---------------------------------------|---|
| IdentResponse / IDENT_RESPONSE        | Shall be issued by a node that received an IdentRequest via SoA.  |
| StatusResponse / STATUS_RESPONSE      | Shall be issued by a node that received a StatusRequest via SoA.  |
| NMTRequest / NMT_REQUEST              | Shall be issued by a CN that received a NMTRequestInvite via SoA.                                       |
| NMTCommand / NMT_COMMAND              | Shall be issued by the MN upon an internal request or upon an external request via NMTRequest.          |
| SDO / SDO                             | May be issued by a CN that received an UnspecifiedInvite via SoA to indicate SDO transmission via ASnd. |
| Manufacturer specific / MANUF_SVC_IDS | Shall be used for manufacturer specific purposes.   |

Tab. 26 ServiceID values in the ASnd frame

Service IDs not listed by Tab. 26 are reserved.

### 4.6.1.2 Non-POWERLINK Frames

Non-POWERLINK frames may be transmitted in accordance with the specifications of any Legacy Ethernet protocol. Non-POWERLINK frame transmission is allowed by the MN if the asynchronous slot is requested by a node..

Refer 5.1 for special requirements to IP (non-POWERLINK) frames.

### 4.6.1.3 Transfer Protection

Transfer disturbances shall be detected by the Ethernet CRC32.

## 4.7 Error Handling Data Link Layer (DLL)

The error handling on the data link layer forms the basis for diagnosis. Often the real error source can be detected only by analysing/interpreting of multiple error symptoms on multiple nodes. Depending on the error symptom / error source the nodes have to react on different layers. The error handling should be simple and easy to implement.

### 4.7.1 Possible Error Sources and Error Symptoms

The following error sources are handled by the MN and the CN. Details are explained in the following sections.

- Physical layer error sources
  - Loss of link (no link condition – port of Ethernet controller)
  - Incorrect physical Ethernet operating modes (10 Mbit/s or full duplex)
  - Transmission Errors detected by CRC errors
  - Rx buffer overflow
  - Tx buffer underrun
- POWERLINK Data Link Layer error symptoms
  - Loss of frame
    - SoC-Frame/ SoA-Frame
    - PReq / PRes Frame
  - Collisions
  - Cycle Time exceeded
  - POWERLINK Address Conflict
  - Multiple Managing Nodes
  - Timing Violation (late Response)

Error recognition strongly depends of the device's hardware and software implementation. Device implementation should be close to this specification but some of the optional error classes listed by the following paragraphs may not be supported. Support shall be indicated by the respective device description entry.

## 4.7.2 Error Handling Table for CN

| Error Symptoms detected by the CNs | Cumulative Cnt | Threshold Cnt | Direct Reaction | DLL Local Handling                       | Error Codes          | NMT Local Handling   |
|------------------------------------|----------------|---------------|-----------------|--|----------------------|--|
| Loss of link                       | o              | o             |                 | These are considered to be error sources | E_DLL LOSS OF LINK   | Logging in Error History   |
| Incorrect Physical operating mode  |                |               | o               |  | E_DLL_BAD_PHYS_MODE  | Logging in Error History   |
| Tx/Rx Buffer underrun / overflow   |                |               | o               |  | E_DLL_MAC_BUFFER     | NMT_GT6<br>Internal Communication Error (handling of internal SW errors)<br>Logging in Error History |
| CRC Error                          | m              | o             |                 |  | E_DLL_CRC_TH         | NMT_CT11, Error Condition<br>Logging in Error History  |
| Collision                          | o              | o             |                 |  | E_DLL_COLLISION_TH   | NMT_GT6<br>Internal Communication Error (handling of internal SW errors)<br>Logging in Error History |
| Invalid Format                     |                |               | m               |  | E_DLL_INVALID_FORMAT | NMT_GT6<br>Internal Communication Error (handling of internal SW errors)<br>Logging in Error History |
| SoC jitter out of range            | o              | o             | o               |  | E_DLL_JITTER_TH      | NMT_CT11, Error Condition<br>Logging in Error History  |
| Loss of PReq                       | o              | o             |                 |  | E_DLL LOSS_PREQ_TH   | NMT_CT11, Error Condition<br>Logging in Error History  |
| Loss of SoA                        | o              | o             |                 |  | E_DLL LOSS_SOA_TH    | NMT_CT11, Error Condition<br>Logging in Error History  |
| Loss of SoC                        | m              | m             |                 |  | E_DLL LOSS_SOC_TH    | NMT_CT11, Error Condition<br>Logging in Error History  |

Tab. 27 CN error handling table

Remarks:

- Change of NMT state shall be signalled to all nodes (reason can be read at ERR\_History\_ADOM)
- In ERR\_History\_ADOM, all logging events shall be registered.
- None of the described Error symptoms on the CN shall be signalled via emergency queue to the MN.
- Cumulative Cnt, Threshold Cnt
  - m mandatory (Counters: shall be implemented / Detection: shall be detected )
  - o optionally (Counters: may be implemented / Detection: may be detected )
- Direct Reaction:
  - m a direct Reaction on an error occurrence shall be proceeded either on the DLL state machine or on the NMT state machine
  - o a direct Reaction on an error occurrence may be proceeded either on the DLL state machine or on the NMT state machine

### 4.7.3 Error Handling Table for MN

| Error Symptoms                    | Cumulative Cnt | Threshold Cnt | Direct Reaction | DLL Local Handling  | Error Codes             | NMT Local Handling  |
|-----------------------------------|----------------|---------------|-----------------|---|-------------------------|---|
| Loss of link                      | o              |               | o               | These are considered to be error source   | E_DLL LOSS OF LINK      | Log in Error History  |
| Incorrect Physical operating mode |                |               | o               |   | E_DLL_BAD_PHYS_MODE     | Log in Error History  |
| Tx/Rx Buffer underrun / overflow  |                |               | o               |   | E_DLL_MAC_BUFFER        | NMT_GT6 Internal Communication Error (handling of internal SW errors)<br>Logging in Error History   |
| CRC error                         | m              | o             |                 |   | E_DLL_CRC_TH            | NMT_MT6<br>Logging in Error History   |
| Collision                         | o              | o             |                 |   | E_DLL_COLLISION_TH      | NMT_GT6 Internal Communication Error (handling of internal SW errors)<br>Logging in Error History   |
| Collision                         |                |               | m               | Communication suspends for a configurable number of cycles .<br>Changes its State to:<br>DLL_MS_WAIT_SOC_TIME | E_DLL_COLLISION         | Logging in Error History  |
| Invalid Format                    |                |               | m s             |   | E_DLL_INVALID_FORMAT    | Remove respective CN from configuration, Send NMT State Command "NMTResetNode" to respective CN.<br>Logging in Error History                              |
| Multiple MNs                      |                |               | o               |   | E_DLL_MULTIPLE_MN       | State != NMT_MS_NOT_ACTIVE-> NMT_GT6 Internal Communication Error<br>State = NMT_MS_NOT_ACTIVE -> reside in NMT_MS_NOT_ACTIVE<br>Logging in Error History |
| POWERLINK Address conflict        |                |               | m s             |   | E_DLL_ADDRESS_CONFLICT  | Remove all involved CNs from configuration  |
| Loss of PRes                      | o s            | m s           |                 |   | E_DLL LOSS PRES TH      | Remove respective CN from configuration, Send NMT State Command NMTResetNode to respective CN.<br>Logging in Error History                                |
| Late PRes                         | o s            | o s           |                 |   | E_DLL_LATE_PRES_TH      | Remove respective CN from configuration, Send NMT State Command NMTResetNode to respective CN<br>Logging in Error History                                 |
| Loss of StatusResponse            | o s            | m s           |                 |   | E_DLL LOSS STATUSRES TH | Remove respective CN from configuration, Send NMT State Command NMTResetNode to respective CN.<br>Logging in Error History                                |
| Cycle time exceeded               | o              | o             |                 |   | E_DLL_CYCLE_EXCEED TH   | NMT_MT6<br>Logging in Error History   |
| Cycle time exceeded               |                |               | m               | Skip next cycle   | E_DLL_CYCLE_EXCEED      | Logging in Error History  |

Tab. 28 MN error handling table

Remarks:

- Change of NMT state shall be signalled to all nodes ( See Object ERR\_History\_ADOM)
- In Object ERR\_History\_ADOM, all logging events shall be registered.
- Cumulative Cnt, Threshold Cnt
  - m mandatory (Counters: shall be implemented / Detection: shall be detected )
  - o optionally (Counters: may be implemented / Detection: may be detected )
  - s per CN (Counters: per CN a Counter is used / Detection: Error shall be assigned to a CN)
- Direct Reaction:
  - m a direct reaction on an error occurrence shall be proceeded either on the DLL state machine or on the NMT state machine
  - o a direct reaction on an error occurrence may be proceeded either on the DLL state machine or on the NMT state machine
  - s per CN (Reaction: per CN / Detection: Error shall be assigned to a CN)

## 4.7.4 Error Handling Registration

This section gives an overview of the error registration on the MN and on CNs. The figure below shows all events that can occur and how they may get registered. On each node an Error History exists, where the occurred error symptoms are stored.

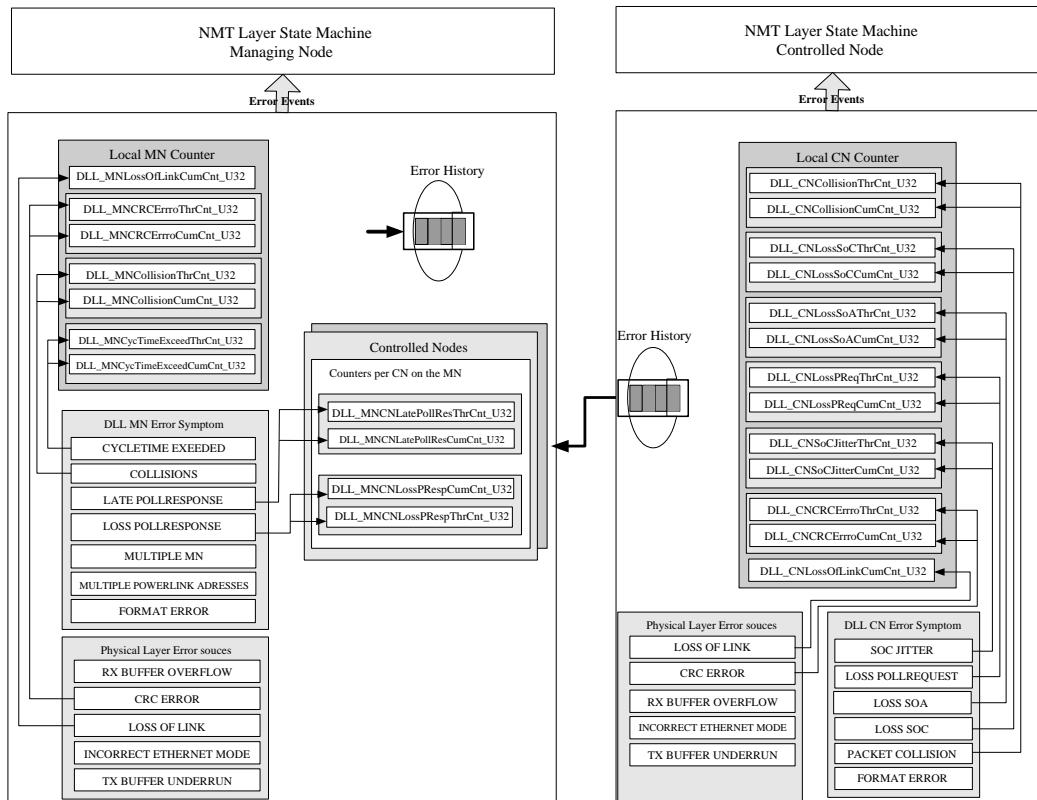


Fig. 33. Error registration

#### 4.7.4.1 Threshold counters

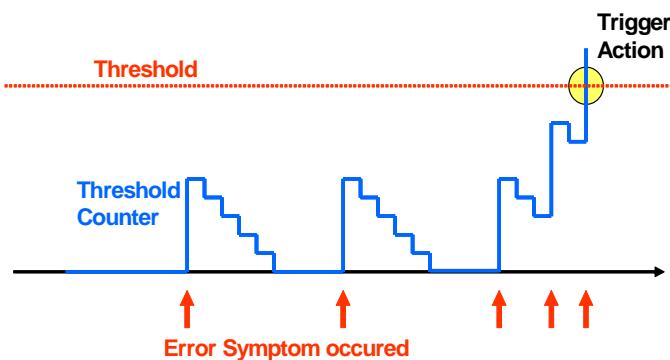


Fig. 34. Threshold counter

Every time an error symptom occurs the threshold counter shall be incremented by 8. After each cycle without reoccurrence of the error the counter shall be decremented by one (Threshold counter 8:1).

When the threshold value is reached (threshold counter  $\geq$  threshold), it shall trigger an action and the threshold counter shall be reset to 0.

All kinds of POWERLINK cycle, e.g. reduced and isochronous cycle, shall decrement threshold counters. Async-only and multiplexed nodes shall decrement at every cycle but not only to cycles addressing them.

The threshold value shall be configurable.

Immediate error reaction shall be commanded by a threshold value of 1.

Threshold counting and error reaction may be deactivated by setting the threshold value to 0.

Threshold counter handling shall be performed on a per error source basis.

#### 4.7.4.2 Cumulative Counter

The Cumulative counter shall be incremented by 1 every time an error symptom occurs. An overflow may occur.

Cumulative counters shall not be reset by reset commands. An application may provide means to reset cumulative counters.

### 4.7.5 Physical Layer Error Sources

The data link layer uses the physical layer error sources for diagnosis of DLL communication error symptoms.

#### 4.7.5.1 Loss of Link

- **Error source**

The Loss of Link can occur if the connection is interrupted, e.g. wire breaks, somebody pulls out the network cable or a hub in the POWERLINK network is defect.

- **Error recognition**

Loss of Link is a late detected error source explaining the primary error detections. Whenever a loss of a frame or a timing violation on the Data Link Layer is detected, the MN resp. CN checks the Physical Layer for a no link condition on the Ethernet MAC controller.

Recognition is optional.

- **Handling**

If the Loss of Link is detected, it is logged in the Error History.

Loss of Link shall be reported regardless any error log triggered by the preceding primary error detection.

- **Registration**

Optionally a cumulative counter (DLL\_MNLossOfLinkCum\_U32, resp. DLL\_CNLossOfLinkCum\_U32) is incremented.

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code         | Timestamp | Additional Information |
|----------------|------------------|--------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL LOSS OF LINK | XXXX      |                        |

#### 4.7.5.2 Incorrect physical Ethernet operating mode

- **Error source**

During the initialisation the new node may check whether it is 100 Mbit/s half duplex Ethernet operating mode. Otherwise timing requirements won't be fulfilled. This situation may occur if auto negotiation is used (not recommended, see Par. 3) and the communication partner is using 10 MBits (e.g. a Hub) or 100 MBit full duplex (e.g. a Switch).

- **Error recognition**

Incorrect physical Ethernet operating mode is a late detected error source explaining the primary error detections. Whenever a loss of a frame or a timing violation on the Data Link Layer is detected, the MN resp. CN checks the Physical Layer for an incorrect physical Ethernet operating mode on the Ethernet MAC controller. Every time, when one of the following error symptoms occurs, it checks if the cause of the error symptom was an incorrect physical ETHERNET operating mode.

Recognition is optional. Support shall be indicated by D\_DLL\_ErrBadPhysMode\_BOOL.

- **Handling**

If an incorrect physical Ethernet operating mode is detected, it is logged in the error history.

Incorrect physical Ethernet operating mode is late detected error source explaining the primary error detections specified by 4.7.6.2, 4.7.7.3.1, 4.7.7.3.2 and 4.7.7.3.3. It shall be reported regardless any error log triggered by the preceding primary error detection.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code          | Timestamp | Additional Information |
|----------------|------------------|---------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_BAD_PHYS_MODE | XXXX      |                        |

#### 4.7.5.3 Rx MAC buffer overflow / Tx MAC buffer underrun

- **Error source**

If the receive MAC buffer of a CN or MN overflows, it cannot receive frames for a while.

The transmit MAC buffer underrun error on the physical layer occurs; when the buffer becomes empty during transmission.

- **Error recognition**

Buffer overflow resp underrun is a late detected error source explaining the primary error detections. Whenever a loss of a frame or a timing violation on the Data Link Layer is detected, the MN resp. CN checks the Physical Layer for an Rx MAC buffer overflow or a TX MAC buffer underrun on the Ethernet MAC controller.

Recognition is optional. Support shall be indicated by D\_DLL\_ErrMacBuffer\_BOOL.

- **Handling**

If a Buffer error is detected, it is logged in the error history and the NMT layer is notified. The CN resp. MN NMT state machine handles this error source as an internal Communication Error (NMT\_GT6) and changes its state to NMT\_GS\_RESET\_COMMUNICATION.

It shall be reported regardless any error log triggered by the preceding primary error detection.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code       | Timestamp | Additional Information |
|----------------|------------------|------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_MAC_BUFFER | XXXX      |                        |

#### 4.7.5.4 Transmission / CRC Errors

- **Error source**

Transmission errors are detected by hardware (CRC-Check) in the Ethernet-Controller. Received frames containing CRC errors are simply discarded.

- **Error recognition**

Every time a frame is lost, the node shall check if a CRC error has occurred. A device may also detect CRC errors on unexpected frames.

- **Handling**

If a CRC error is detected, it shall be logged in the error history.

Error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1).

If the threshold counter DLL\_MNCRCError\_REC.ThresholdCnt\_U32, resp.

DLL\_CNCRCError\_REC.ThresholdCnt\_U32 violates the threshold

DLL\_MNCRCError\_REC.Threshold\_U32 resp. DLL\_CNCRCError\_REC.Threshold\_U32, the CN resp. MN NMT state machine shall handle this error source as an “error condition” (NMT\_CT11 resp. NMT\_MT6) and shall change its state to NMT\_CS\_PRE\_OPERATIONAL\_1 resp. NMT\_MS\_PRE\_OPERATIONAL\_1.

*It's not recommended to enable error reaction if unexpected frame CRC error recognition is active.*

- **Registration**

MN and CN shall operate a cumulative counter (DLL\_MNCRCError\_REC.CumulativeCnt\_U32, resp. DLL\_CNCRCError\_REC.CumulativeCnt\_U32, see. 4.7.4.2.) and a threshold counter (DLL\_MNCRCError\_REC.ThresholdCnt\_U32, resp.

DLL\_CNCRCError\_REC.ThresholdCnt\_U32).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code   | Timestamp | Additional Information |
|----------------|------------------|--------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_CRC_TH | XXXX      |                        |

#### 4.7.6 Communication Error Symptoms detected by the MN

This section describes the error symptoms on the data link layer which are detected and handled by the MN.

##### 4.7.6.1 Timing Violation

###### 4.7.6.1.1 Slot Time Exceeded

Certain timing constellations must be distinguished when a frame is received. The timing behaviour of nodes shall be monitored, otherwise the entire cycle time can exceed.

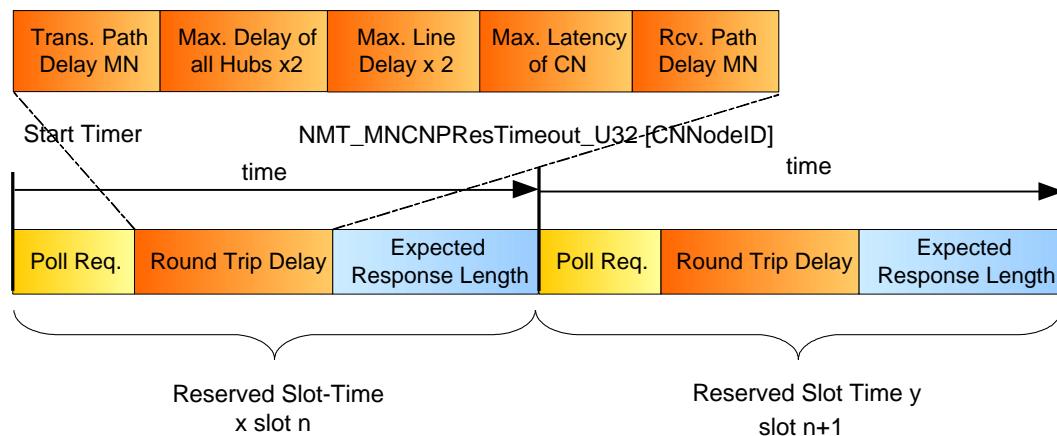


Fig. 35. Timeouts

The PRes frame of the CN must have been received completely before the slot timeout given by NMT\_MNCNPResTimeout\_U32[CN Node ID] expires. The violation of the slot time produces the situations described below.

#### 4.7.6.1.1.1 Case 1-2 Frame received in time

The behaviour is identical. The second case shows the latest acceptable time for the frame receipt. The current slot timeout is switched off immediately after the frame is received.

#### 4.7.6.1.1.2 Case 3 Loss of PRes: Frame not received

The Loss of a PRes frame is detected by the slot timeout.

#### 4.7.6.1.1.3 Case 4-6 Late PRes: Frame received in foreign slot (also collisions)

The cases 4 - 6 have one fact in common: A frame is received, which does not belong to the current slot. The worst case could lead to a violation of the cycle time. This kind of error can disturb the entire power link communication. Only in case 4 and 6 a Late PRes error can be detected. In case 5 a collision occurs as a result of a Late PRes error.

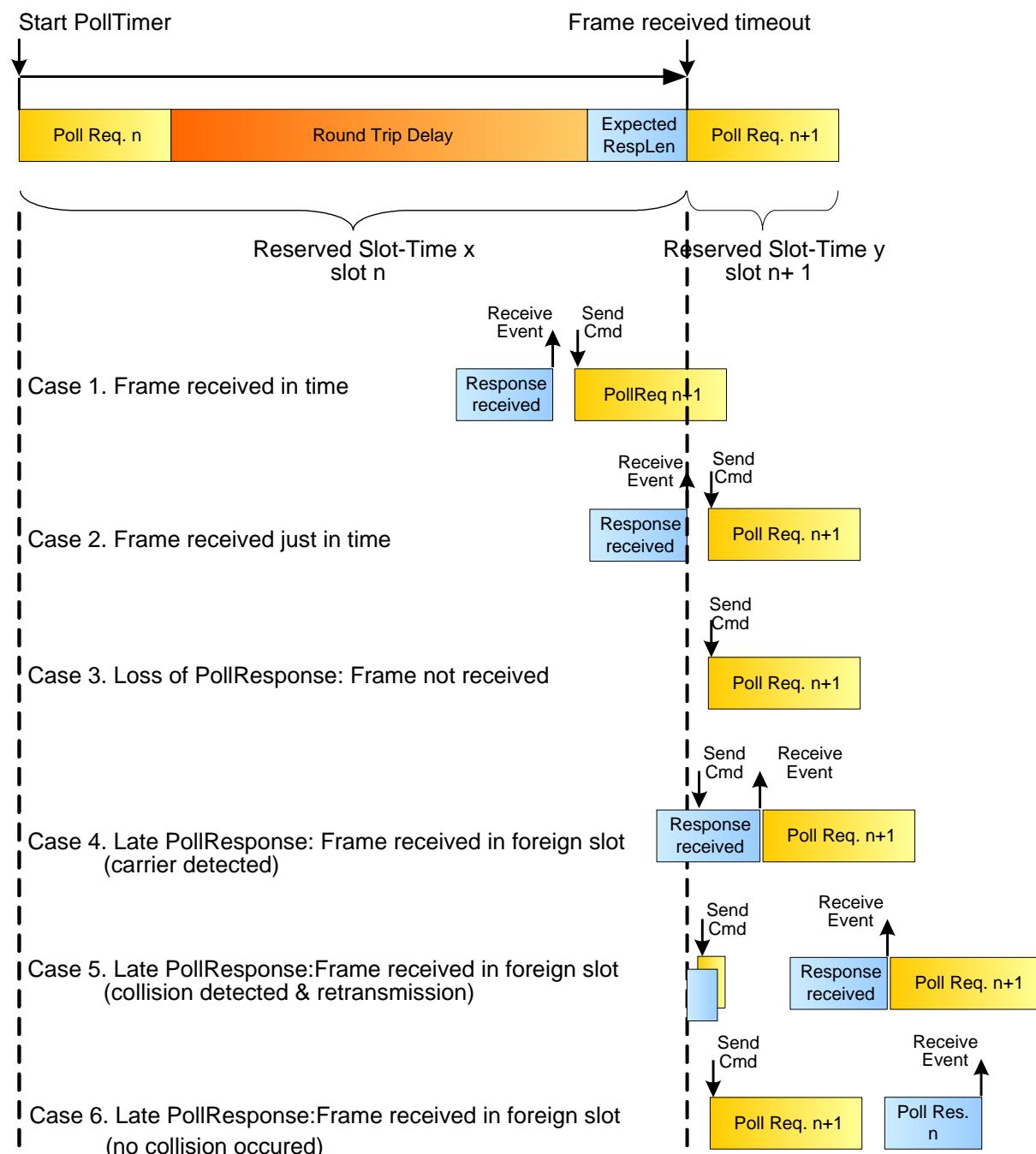


Fig. 36. Timing violation

#### 4.7.6.2 Loss of PRes

- **Error source**

Following possible error sources could cause this error symptom:

- Physical error sources on the MN
  - Transmission Error (CRC Error)
  - Loss of link
  - Rx Buffer overflow
  - Tx Buffer underrun
- Error symptoms on a CN in the POWERLINK network
  - Frame Collision error symptom

- A CN, which response latency is higher than allowed.
  - A component of the network structure is defect.
  - Power failure on a CN
  - Etc.
  - **Error recognition**  
If the slot timer expires, no frame was received during the reserved slot time. (See Slot Time exceeded: Case 3).  
Loss of PRes is detected by the MN cycle state machine and reported via error event DLL\_MEV\_LOSS\_PRES.
  - **Handling**  
After detecting a CN's Loss of PRes by the NMT\_MNCNPResTimeout\_AU32[CN Node ID], the MN shall proceed with the PReq for the next CN (or SoA if the end is reached).  
Error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1).  
If the threshold counter DLL\_MNCNLossPResThrCnt\_AU32[CN Node ID] violates the threshold DLL\_MNCNLossPResThreshold\_AU32[CN Node ID], the MN NMT State machine shall consider the CN inactive and shall remove it from the isochronous processing. Additionally it shall send the command NMTResetNode to the respective CN. On this command the CN will change its state (See 7.1.4). Whenever this error symptom is detected, the MN shall check for a physical layer error source.  
The error symptom shall be logged in the Error History every time the threshold value is reached.
  - **Registration**  
The MN shall operate a threshold counter array DLL\_MNCNLossPResThrCnt\_AU32 and optionally a cumulative counter array DLL\_MNCNLossPResCumCnt\_AU32 (see. 4.7.4.2). CNs are represented by the subindices of the arrays.  
History Entry Object ERR\_History\_ADOM:
- | Mode | Profile | Error Code         | Timestamp | Additional Information |
|------|---------|--------------------|-----------|------------------------|
| 3h   | 002h    | E_DLL_LOSS_PRES_TH | XXXX      |                        |

#### 4.7.6.3 Late PRes

- **Error source**  
Following possible error sources could cause this error symptom:
  - Physical error sources on the MN
    - Transmission Error (CRC Error)
    - Loss of link
    - Rx Buffer overflow
    - Tx Buffer underrun
  - Error symptoms on a CN in the POWERLINK network
    - Frame Collision error symptom
  - A CN, which response latency is higher than allowed.
  - A component of the network structure is defect.
  - Power failure on a CN
  - Etc.
- **Error recognition**  
A Late PRes error symptom may be detected on the MN, when the carrier is busy while trying to send the PReq frame (See Slot Time exceeded: Case 4) or when it receives a PRes frame from an unexpected CN (See Slot Time exceeded: Case 6).

Error recognition shall be performed at NMT states NMT\_MS\_PRE\_OPERATIONAL\_2, NMT\_MS\_READY\_TO\_OPERATE and NMT\_MS\_OPERATIONAL on isochronous CNs.

Recognition is optional.

- **Handling**

After detecting a Late PRes error, the MN proceeds with the PReq for the next CN (or SoA if the end is reached).

Error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1).

If the threshold counter DLL\_MNCNLatePResThrCnt\_AU32[CN Node ID] violates the threshold DLL\_MNCNLatePResThreshold\_AU32[CN Node ID], the MN NMT State machine shall consider the CN inactive and shall remove it from the isochronous processing. Additionally it shall send the command NMTResetNode to the respective CN. On this command the CN will change it's state (See 7.1.4).

If a PRes frame, that does not belong to the current slot, is received, the frame shall be rejected.

Whenever a Late PR error is detected, the MN shall check for a physical layer error source.

The error symptom shall be logged in the Error History every time the threshold value is reached

- **Registration**

The MN shall operate a threshold counter array DLL\_MNCNLatePResThrCnt\_AU32 and optionally a cumulative counter array DLL\_MNCNLatePResCumCnt\_AU32 (see. 4.7.4.2). CNs are represented by the subindices of the arrays.

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code         | Timestamp | Additional Information |
|----------------|------------------|--------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_LATE_PRES_TH | XXXX      |                        |

#### 4.7.6.4 Cycle Time Exceeded

- **Error source**

Following possible error sources could cause this error symptom:

- POWERLINK configuration failure
- A CN, which Response latency is higher than allowed.
- A component of the network structure, which is defect.
- Etc.

- **Error recognition**

A cycle time violation situation is defined as: The POWERLINK network was busy up to a time where a SoC should have been sent. If an ASnd frame or an Ethernet frame at the end of a cycle is delayed, then it may cause a collision with the SoC frame or a delay of the SoC frame.

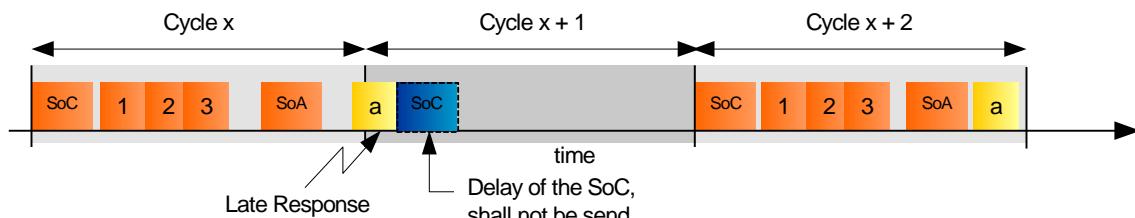


Fig. 37. Cycle time exceeded

Error detection is performed by the MN cycle state machine and reported via error event DLL\_MEV\_CYCLE\_EXCEED.

To prevent cycle time exceed error the cycle shall be dimensioned for the worst case (with max response times / timeouts).

On line check may be done by verifying the remaining cycle time  $\geq$  maximum async slot time (refer. DLL) before sending the SoA frame. No inviting SoA shall be transmitted if verification failed.

- **Handling**

A cycle time violation is considered a configuration error, hence the default behaviour shall be to suspend one cycle and to log the error symptom at the error history.

Optionally a threshold counter DLL\_MNCycTimeExceed\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_MNCycTimeExceed\_REC.CumulativeCnt\_U32 may be operated. Every time DLL\_MNCycTimeExceed\_REC.ThresholdCnt\_U32 reaches the threshold DLL\_MNCycTimeExceed\_REC.Threshold\_U32 or, if no Threshold counter is implemented, every time the error symptom occurs, the MN NMT state machine shall be notified. It shall handle this error source as an “error condition” (NMT\_MT6) and shall change its state to NMT\_MS\_PRE\_OPERATIONAL\_1.

Every error event shall be logged at the error history by E\_DLL\_CYCLE\_EXCEED, threshold violations by E\_DLL\_CYCLE\_EXCEED\_TH.

- **Registration**

The MN may optionally operate a threshold counter

DLL\_MNCycTimeExceed\_REC.ThresholdCnt\_U32 and optionally a cumulative counter array DLL\_MNCycTimeExceed\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code            | Timestamp | Additional Information |
|----------------|------------------|-----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_CYCLE_EXCEED    | XXXX      |                        |
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_CYCLE_EXCEED_TH | XXXX      |                        |

## 4.7.6.5 Collisions

- **Error Source**

The number of hubs in the POWERLINK network may violate the path delay variability specification of IEEE 802.3. Because standard Ethernet controllers according to IEEE 802.3 are used, collisions can be detected only in some cases.

ETHERNET powerlink does not depend on the discovery of collisions.

In NMT\_MS\_PRE\_OPERATIONAL\_1, NMT\_MS\_PRE\_OPERATIONAL\_2, NMT\_MS\_READY\_TO\_OPERATE, and NMT\_MS\_OPERATIONAL, no collisions should occur due to the POWERLINK cycle design. If a node does not follow these requirements, then the determinism and the high precision synchronisation cannot be guaranteed anymore. Nevertheless collisions can occur in case of configuration failures or defect nodes.

- **Error recognition**

If the Ethernet controller discovers a collision in the POWERLINK network, it shall start the standard Ethernet procedure for collisions.

- **Handling**

MN shall log the error symptom in the error history and shall suspends cycle generation for a configurable number (DLL\_MNCycleSuspendNumber\_U32) of cycles, before continuing with the isochronous and asynchronous communication.

The MN data link layer state machine shall change its state to DLL\_MS\_WAIT\_SOC\_TIME.

Optionally a threshold counter DLL\_MNCollision\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_MNCollision\_REC.CumulativeCnt\_U32 may be operated. Every time DLL\_MNCollision\_REC.ThresholdCnt\_U32 reaches the threshold

DLL\_MNCollision\_REC.Threshold\_U32 or, if no Threshold counter is implemented, every time the error symptom occurs, the MN NMT state machine shall be notified. It shall handle this error source as an “internal Communication Error (NMT\_GT6)” and shall change its state to NMT\_GS\_RESET\_COMMUNICATION.

Every error event shall be logged in the error history by E\_DLL\_COLLISION, threshold violations by E\_DLL\_COLLISION\_TH.

- **Registration**

The MN may optionally operate a threshold counter DLL\_MNCollision\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_MNCollision\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code         | Timestamp | Additional Information |
|----------------|------------------|--------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_COLLISION    | XXXX      |                        |
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_COLLISION_TH | XXXX      |                        |

## 4.7.6.6 Invalid Formats

- **Error Source**

Invalid Formats can result from software faults or hardware errors in the nodes. Format Errors shall only be detected on PRes and ASnd frames with correct CRC. Format Errors in the asynchronous communication not using the POWERLINK ASnd frame shall be ignored.

Invalid Format Errors can also occur if there are various firmware versions within the POWERLINK network. In that case nodes may not support frame formats of other nodes.

- **Error recognition**

An invalid Format error symptom shall be recognized if a POWERLINK frame header contains an unsupported value. This may be a false Node ID, etc. An invalid format error shall also be caused, if the received frame size is larger than the predicted buffer input size.

A false resp. unknown message type or service ID does not result in an invalid format error.

*Note: Otherwise the mixing of old and new devices that support extensions of the specification with new message types or service IDs will result in invalid format errors in old CNs.*

- **Handling**

If a CN causes an invalid format error, the MN NMT State machine shall consider the CN inactive and shall remove it from the isochronous processing. Additionally it shall send to the CN the command NMTResetNode. On this command the CN will change its state (See 7.1.4).

The error symptom shall be logged to error history, every time it occurs.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_INVALID_FORMAT | XXXX      |                        |

## 4.7.6.7 POWERLINK Address Conflicts

- **Error source**

Because of the distinct MAC address of each node it is not possible that two or more nodes own the same MAC address but it is still possible that two or more nodes own the same powerlink node address. Only in the asynchronous communication multiple CNs can answer on a SoA frame (service channel: Ident). Since the MN sends a MAC multicast (SoA frame) to a unicast powerlink address, several CNs with the same powerlink address are able to respond.

- **Error recognition**

The MN shall detect multiple used POWERLINK addresses in a network by counting the responses on an IdentRequest SoA frame.

- **Handling**

If the MN detects that multiple CNs cause POWERLINK address conflicts, the MN NMT State machine shall consider the involved CNs inactive and shall remove them from the configuration.

The error symptom shall be logged to error history, every time it occurs.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code             | Timestamp | Additional Information |
|----------------|------------------|------------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_ADDRESS_CONFLICT | XXXX      | Status of the MN       |

### 4.7.6.8 Multiple MNs on a single POWERLINK Network

- **Error source**

In NMT\_MS\_NOT\_ACTIVE the MN shall observe the POWERLINK network whether another MN is already running.

If multiple MNs simultaneously start the communication, a combination of error symptoms will be detected. One of these symptoms will be the receipt of SoC or SoA frames from the other MN.

- **Error recognition**

The MN will receive SoC or SoA frames from the other MN.

Recognition is optional. Support shall be indicated by D\_DLL\_ErrMNMultipleMN\_BOOL.

- **Handling**

If the MN is in the state NMT\_MS\_NOT\_ACTIVE and detects that another MN is running, it shall reside in its state.

If multiple MNs start the communication simultaneously and an MN detects this error symptom it notifies the NMT state machine. It handles this error source as an “internal Communication Error (NMT\_GT6)” and changes its state to NMT\_GS\_RESET\_COMMUNICATION.

The error symptom shall be logged every time it occurs.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code        | Timestamp | Additional Information |
|----------------|------------------|-------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_MULTIPLE_MN | XXXX      |                        |

### 4.7.6.9 Loss of StatusResponse

- **Error source**

Following possible error sources could cause this error symptom:

- Physical error sources on the MN
  - Transmission Error (CRC Error)
  - Loss of link
  - Rx Buffer overflow
  - Tx Buffer underrun
- Error symptoms on a CN in the POWERLINK network
  - Frame Collision error symptom
  - A CN, which response latency is higher than allowed.
  - A component of the network structure is defect.
  - Power failure on a CN
  - Etc.

- **Error recognition**

If a StatusRequest was sent and the asynchronous slot time expires, no StatusResponse frame was received during the asynchronous slot.

The error is detected by the MN cycle state machine and reported via error event DLL\_MEV\_LOSS\_STATUSRESPONSE.

- **Handling**

After detecting a CN's Loss of StatusResponse by the NMT\_MNCycleTiming\_REC. AsyncSlotTimeout\_U32, the MN shall proceed with the SoC of the next cycle.

Error reaction shall be triggered by the threshold counter mechanism (see 4.7.4.1).

If the threshold counter DLL\_MNLossStatusResThrCnt\_AU32[CN Node ID] violates the threshold DLL\_MNLossStatusResThreshold\_AU32[CN Node ID], the MN NMT State machine shall consider the CN inactive and shall remove it from the (isochronous) processing.

Additionally it shall send the command NMTResetNode to the respective CN. On this command the CN will change its state (See 7.1.4). Whenever this error symptom is detected, the MN shall check for a physical layer error source.

The error symptom shall be logged in the Error History every time the threshold value is reached.

- **Registration**

The MN shall operate a threshold counter array DLL\_MNLossStatusResThrCnt\_AU32 and optionally a cumulative counter array DLL\_MNLossStatusResCumCnt\_AU32 (see. 4.7.4.2). CNs are represented by the subindices of the arrays.

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code              | Timestamp | Additional Information |
|----------------|------------------|-------------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_LOSS_STATUSRES_TH | XXXX      |                        |

## 4.7.7 Communication Error Symptoms detected by the CN

This section describes error symptoms on the data link layer detected and handled by CNs.

### 4.7.7.1 Collisions

- **Error Source**

The number of hubs in the POWERLINK network may violate the path delay variability specification of IEEE 802.3. Because standard Ethernet controllers according to IEEE 802.3 are used, collisions can be detected only in some cases.

ETHERNET powerlink does not depend on the discovery of collisions.

In NMT\_CS\_PRE\_OPERATIONAL\_1, NMT\_CS\_PRE\_OPERATIONAL\_2, NMT\_CS\_READY\_TO\_OPERATE, and NMT\_CS\_OPERATIONAL, no collisions should occur because of the POWERLINK cycle design. If a node does not fulfill these requirements, then the determinism and the high precision synchronisation can not be guaranteed anymore. Nevertheless collisions can occur in case of configuration failures or defect nodes.

- **Error recognition**

If the Ethernet controller discovers a collision in the POWERLINK network, it shall start the standard Ethernet procedure for collisions.

- **Handling**

A threshold counter DLL\_CNCollision\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNCollision\_REC.CumulativeCnt\_U32 may be operated optionally. Every time the threshold counter reaches the threshold DLL\_CNCollision\_REC.Threshold\_U32, the CN NMT state machine shall be notified. It shall handle this error source as an "Internal Communication Error (NMT\_GT6)" and shall change its state to NMT\_GS\_RESET\_COMMUNICATION.

Threshold violations shall be logged in the error history.

- **Registration**

The CN may optionally operate a threshold counter DLL\_CNCollision\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNCollision\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code         | Timestamp | Additional Information |
|----------------|------------------|--------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_COLLISION_TH | XXXX      |                        |

### 4.7.7.2 Invalid Formats

- **Error Source**

Invalid Formats can result from software faults or hardware errors in the nodes. Format Errors shall be detected on all POWERLINK frames with correct CRC received by the CN, e.g SoC, PReq, PRes, SoA and ASnd. Format Errors in the asynchronous communication not using the POWERLINK ASnd frame shall be ignored.

Invalid Format Errors can also occur if there are various firmware versions within the POWERLINK network. In that case nodes may not support frame formats of other nodes.

- **Error recognition**

An invalid Format error symptom shall be recognized if a POWERLINK frame header contains an unsupported value. This may be a false Node ID, etc. An invalid format error is also caused, if the received frame size is larger than the predicted buffer input size.

A false resp. unknown message type or service ID does not result in an invalid format error.

*Note: Otherwise the mixing of old and new devices that support extensions of the specification with new message types or service IDs will result in invalid format errors in old CNs.*

- **Handling**

If a CN detects an invalid format error, it shall notify its NMT State machine. The CN NMT State machine handles this error source as an “Internal Communication Error (NMT\_GT6)” and changes its state to NMT\_GS\_RESET\_COMMUNICATION.

The error symptom is logged to error history every time it occurs.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_INVALID_FORMAT | XXXX      |                        |

### 4.7.7.3 Loss of Frames

Loss of Frame errors are detected by the CN cycle state machine and reported via error events. The state machine only detects the loss of frames, which were sent by the MN.

- **Error source**

Following error sources could cause this error symptom:

- Physical error sources on the MN (see 4.7.5)
  - Transmission Error (CRC Error)
  - Loss of link
  - Rx Buffer overflow
  - Tx Buffer underrun
- Error symptoms on a CN in the POWERLINK network
  - Frame Collision (see 4.7.7.1)
- A component of the network structure is defect.
- Power failure on a CN or an MN

- Etc.

### 4.7.7.3.1 Loss of SoC

- **Error recognition**

Loss of SoC error is detected by the CN Cycle State Machine and reported via error event DLL\_CEV\_LOSS\_SOC.

- **Handling**

On detecting a Loss of SoC the cumulative counter DLL\_CNLossSoC\_REC.CumulativeCnt\_U32 shall be incremented. The CN shall reply on any invitation with the data of the previous cycle. The CN shall accept new isochronous or asynchronous data.

Error reaction shall be triggered by the threshold counter mechanism (see 4.7.4.1).

If the threshold counter DLL\_CNLossSoC\_REC.ThresholdCnt\_U32 violates the threshold DLL\_CNLossSoC\_REC.Threshold\_U32, the CN shall notify its NMT state machine and shall log the error symptom to the error history. The CN NMT state machine shall handle this error source as an “error condition (NMT\_CT11)” and shall change it’s state to NMT\_CS\_PRE\_OPERATIONAL\_1.

Whenever this error symptom is detected, the CN shall check for a physical layer error source (see 4.7.5).

- **Registration**

The CN shall operate a threshold counter DLL\_CNLossSoC\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNLossSoC\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode | Profile | Error Code        | Timestamp | Additional Information |
|------|---------|-------------------|-----------|------------------------|
| 3h   | 002h    | E_DLL LOSS SOC TH | XXXX      |                        |

### 4.7.7.3.2 Loss of SoA

- **Error recognition**

Loss of SoA error is detected by the CN Cycle State Machine and reported via error event DLL\_CEV\_LOSS\_SOA.

- **Handling**

On detecting a Loss of SoA the cumulative counter DLL\_CNLossSoA\_REC.CumulativeCnt\_U32 shall be incremented. The CN shall continue operation. It shall accept new isochronous or asynchronous data.

Error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1).

If the threshold counter DLL\_CNLossSoA\_REC.ThresholdCnt\_U32 violates the threshold DLL\_CNLossSoA\_REC.Threshold\_U32, the CN shall notify its NMT state machine and shall log the error symptom to the error history. The CN NMT state machine shall handle this error source as an “error condition (NMT\_CT11)” and shall change it’s state to NMT\_CS\_PRE\_OPERATIONAL\_1.

Whenever this error symptom is detected, the CN checks for a physical layer error source (see 4.7.5).

- **Registration**

The CN shall operate a threshold counter DLL\_CNLossSoA\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNLossSoA\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code        | Timestamp | Additional Information |
|----------------|------------------|-------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL LOSS SOA TH | XXXX      |                        |

#### 4.7.7.3.3 Loss of PReq

- **Error recognition**

Loss of PReq error is detected by the CN Cycle State Machine and reported via error event DLL\_CEV\_LOSS\_PREQ.

- **Handling**

On detecting a Loss of PReq the cumulative counter DLL\_CNLossPReq\_REC.CumulativeCnt\_U32 shall be incremented. The CN shall continue communication and listen to cross traffic. It shall accept new isochronous or asynchronous data. Further error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1). If the threshold counter DLL\_CNLossPReq\_REC.ThresholdCnt\_U32 violates the threshold DLL\_CNLossPReq\_REC.Threshold\_U32, the CN shall notify its NMT state machine and shall log the error symptom to the error history. The CN NMT state machine shall handle this error source as an “error condition (NMT\_CT11)” and shall change it’s state to NMT\_CS\_PRE\_OPERATIONAL\_1.

Whenever this error symptom is detected, the CN checks for a physical layer error source (see 4.7.5).

- **Registration**

The CN shall operate a threshold counter DLL\_CNLossPReq\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNLossPReq\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM

| Mode           | Profile          | Error Code         | Timestamp | Additional Information |
|----------------|------------------|--------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL LOSS_PREQ_TH | XXXX      |                        |

#### 4.7.7.3.4 SoC Jitter out of Range

- **Error source**

This error could have various error sources:

- Jitter on not POWERLINK compliant network components
- Collisions in the POWERLINK network
- Failure during the transmission
- MN failures
- Late response of CN causes a delay of the SoC
- Etc.

- **Error recognition**

A CN may control the SoC cycle time jitter.

Every time it receives the SoC frame, it shall measure the cycle time. A SoC jitter error shall be recognized, when the difference between the nominal cycle time and the measured time is out of a configurable range DLL\_CNSoCJitterRange\_U32.

Recognition shall be performed in NMT\_CS\_READY\_TO\_OPERATE and NMT\_CS\_OPERATIONAL.

- **Handling**

If the CN detects a SoC Jitter range violation the cumulative counter DLL\_CNSoCJitter\_REC.CumulativeCnt\_U32 shall be incremented,

Further error reaction shall be triggered by the threshold counter mechanism ( see 4.7.4.1).

If the threshold counter DLL\_CNSoCJitter\_REC.ThresholdCnt\_U32 violates the threshold DLL\_CNSoCJitter\_REC.Threshold\_U32, the CN shall notify its NMT state machine and shall log the error symptom to the error history. The CN NMT state machine shall handle this error source as an “error condition (NMT\_CT11)” and shall change it’s state to NMT\_CS\_PRE\_OPERATIONAL\_1.

- **Registration**

The CN shall operate a threshold counter DLL\_CNSoCJitter\_REC.ThresholdCnt\_U32 and a cumulative counter DLL\_CNSoCJitter\_REC.CumulativeCnt\_U32 (see. 4.7.4.2).

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code      | Timestamp | Additional Information |
|----------------|------------------|-----------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_DLL_JITTER_TH | XXXX      |                        |

## 4.7.8 DLL Error Handling Objects

In this section the objects used by the DLL error handling are described.

Support of threshold counter objects requires support of the respective threshold value object and vice versa.

### 4.7.8.1 Object 1C00<sub>h</sub>: DLL\_MNCRCError\_REC

The following objects are used to monitor CRC errors. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

|           |                      |             |                |
|-----------|----------------------|-------------|----------------|
| Index     | 1C00 <sub>h</sub>    | Object Type | RECORD         |
| Name      | DLL_MNCRCError_REC   |             |                |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: M<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 1 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | M  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a CRC error occurs. Its value monitors all CRC errors that were detected by the MN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | O  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a CRC error occurs on the MN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the CRC error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Threshold_U32   |             |     |
| Data Type     | UNSIGNED32      | Category    | O   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 15              | PDO Mapping | No  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.1)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.2 Object 1C01<sub>h</sub>: DLL\_MNCollision\_REC

The following objects are used to monitor Collision error symptoms. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

|           |                      |             |                |
|-----------|----------------------|-------------|----------------|
| Index     | 1C01 <sub>h</sub>    | Object Type | RECORD         |
| Name      | DLL_MNCollision_REC  |             |                |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: O<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 0 .. 3          | Access      | rw |
| Default Value | -               | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | O  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a Collision error symptom occurs. Its value monitors all Collision error symptoms that were detected by the MN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | O  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a Collision error symptom occurs on the MN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the Collision error occurrence.

- #### • **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |               |             |     |  |
|---------------|---------------|-------------|-----|--|
| Sub-Index     | 03h           |             |     |  |
| Name          | Threshold_U32 |             |     |  |
| Data Type     | UNSIGNED32    | Category    | O   |  |
| Value Range   | UNSIGNED32    | Access      | rws |  |
| Default Value | 15            | PDO Mapping | No  |  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.1)

Threshold counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.3 Object 1C02<sub>h</sub>: DLL\_MNCycTimeExceed\_REC

The following objects are used to monitor “Cycle time exceeded” error symptoms. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

|           |                         |             |                |
|-----------|-------------------------|-------------|----------------|
| Index     | 1C02 <sub>h</sub>       | Object Type | RECORD         |
| Name      | DLL_MNCycTimeExceed_REC |             |                |
| Data Type | DLL_ErrorCntRec_TYPE    | Category    | MN: O<br>CN: - |

- #### • **Sub-Index 00<sub>b</sub>: NumberOfEntries**

|               |                 |             |       |  |
|---------------|-----------------|-------------|-------|--|
| Sub-Index     | 00 <sub>h</sub> |             |       |  |
| Name          | NumberOfEntries |             |       |  |
| Value Range   | 0 .. 3          | Access      | const |  |
| Default Value | -               | PDO Mapping | No    |  |

- Sub-Index 01\_h: CumulativeCnt U32

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | O  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a “Cycle time Exceeded” error symptom occurs. Its value monitors all “Cycle Time exceeded” error symptom that were detected by the MN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | O  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a “Cycle Time exceeded” error symptom occurs on the MN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Cycle Time exceeded” error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Threshold_U32   |             |     |
| Data Type     | UNSIGNED32      | Category    | O   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 15              | PDO Mapping | No  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.1)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.4 Object 1C03<sub>h</sub>: DLL\_MNLossOfLinkCum\_U32

The following objects are used to monitor the “Loss of Link” error source. The cumulative counter shall be incremented by 1 every time a “Loss of Link” error symptom occurs. Its value monitors all “Loss of Link” error symptoms that were detected by the MN.

The object may be implemented only if the error recognition is provided.

| Index         | 1C03 <sub>h</sub>       | Object Type | VAR               |
|---------------|-------------------------|-------------|-------------------|
| Name          | DLL_MNLossOfLinkCum_U32 |             |                   |
| Data Type     | UNSIGNED32              | Category    | MN: Cond<br>CN: - |
| Value Range   | UNSIGNED32              | Access      | rw                |
| Default Value | 0                       | PDO Mapping | No                |

#### 4.7.8.5 Object 1C04<sub>h</sub>: DLL\_MNCNLatePResCumCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a cumulative counter of Late PRes events. The cumulative counter of the respective CN shall be incremented by 1 every time a “Late PollResponse” error symptom occurs. Its value monitors all “Late PollResponse” error symptoms that were detected by the MN.

The object may be implemented only if the error recognition is provided.

|           |                             |             |                   |
|-----------|-----------------------------|-------------|-------------------|
| Index     | 1C04 <sub>h</sub>           | Object Type | ARRAY             |
| Name      | DLL_MNCNLatePResCumCnt_AU32 |             |                   |
| Data Type | UNSIGNED32                  | Category    | MN: Cond<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00h             |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CumCnt**

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | CumCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | rw |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.6 Object 1C05<sub>h</sub>: DLL\_MNCNLatePResThrCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold counter of late PRes. The threshold counter of the respective CN shall be incremented by 8 every time a “Late PollResponse” error symptom occurs on the MN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Late PollResponse” error occurrence.

The object may be implemented only if the error recognition is provided.

|           |                             |             |                   |
|-----------|-----------------------------|-------------|-------------------|
| Index     | 1C05 <sub>h</sub>           | Object Type | ARRAY             |
| Name      | DLL_MNCNLatePResThrCnt_AU32 |             |                   |
| Data Type | UNSIGNED32                  | Category    | MN: Cond<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> | Access      | ro |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: ThrCnt**

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> | Access      | ro |
| Name          | ThrCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | ro |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.7 Object 1C06<sub>h</sub>: DLL\_MNCNLatePResThreshold\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold of late PRes.

Every time the respective DLL\_MNCNLatePResThrCnt\_AU32 value reaches the threshold, a defined action shall proceed and the respective DLL\_MNCNLatePResThrCnt\_AU32 value shall be reset to 0. (See 4.7.7.1)

Threshold Counting may be deactivated by setting respective DLL\_MNCNLatePResThreshold\_AU32 value to 0. If Threshold Counting is deactivated, no error reaction will occur.

The object may be implemented only if the error recognition is provided.

|           |                                |             |                   |
|-----------|--------------------------------|-------------|-------------------|
| Index     | 1C06 <sub>h</sub>              | Object Type | ARRAY             |
| Name      | DLL_MNCNLatePResThreshold_AU32 |             |                   |
| Data Type | UNSIGNED32                     | Category    | MN: Cond<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> | Access      | ro |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: Threshold**

|               |                                   |             |     |
|---------------|-----------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> | Access      | rws |
| Name          | Threshold                         |             |     |
| --            | --                                | Category    | M   |
| Value Range   | UNSIGNED32                        | Access      | rws |
| Default Value | 15                                | PDO Mapping | No  |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.8 Object 1C07<sub>h</sub>: DLL\_MNCNLossPResCumCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a cumulative counter of Loss of PRes events. The cumulative counter of the respective CN shall be incremented by 1 every time a

“Loss of PollResponse” error symptom occurs. Its value monitors all “Loss PollResponse” error symptoms that were detected by the MN.

|           |                             |             |                |
|-----------|-----------------------------|-------------|----------------|
| Index     | 1C07 <sub>h</sub>           | Object Type | ARRAY          |
| Name      | DLL_MNCNLossPResCumCnt_AU32 |             |                |
| Data Type | UNSIGNED32                  | Category    | MN: O<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CumCnt**

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | CumCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | rw |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.9 Object 1C08<sub>h</sub>: DLL\_MNCNLossPResThrCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold counter of Loss of PRes. The threshold counter of the respective CN shall be incremented by 8 every time a “Loss of PollResponse” error symptom occurs on the MN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Loss of PollResponse” error occurrence.

|           |                             |             |                |
|-----------|-----------------------------|-------------|----------------|
| Index     | 1C08 <sub>h</sub>           | Object Type | ARRAY          |
| Name      | DLL_MNCNLossPResThrCnt_AU32 |             |                |
| Data Type | UNSIGNED32                  | Category    | MN: M<br>CN: - |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: ThrCnt**

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | ThrCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | ro |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.10 Object 1C09<sub>h</sub>: DLL\_MNCNLossPResThreshold\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold of lost PRes.

Every time the respective DLL\_MNCNLossPResThrCnt\_AU32 value reaches the threshold, a defined action shall proceed and the respective DLL\_MNCNLossPResThrCnt\_AU32 value shall be reset to 0. (See 4.7.7.1)

Threshold counting may be deactivated by setting respective DLL\_MNCNLossPResThreshold\_AU32 value to 0. If Threshold Counting is deactivated, no error reaction will occur.

|           |                                |             |                |
|-----------|--------------------------------|-------------|----------------|
| Index     | 1C09 <sub>h</sub>              | Object Type | ARRAY          |
| Name      | DLL_MNCNLossPResThreshold_AU32 |             |                |
| Data Type | UNSIGNED32                     | Category    | MN: M<br>CN: - |

- **Sub-Index 0h: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00h             |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01h – FEh: Threshold**

|               |            |             |     |
|---------------|------------|-------------|-----|
| Sub-Index     | 01h – FEh  |             |     |
| Name          | Threshold  |             |     |
| --            | --         | Category    | M   |
| Value Range   | UNSIGNED32 | Access      | rws |
| Default Value | 15         | PDO Mapping | No  |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

#### 4.7.8.11 Object 1C0A<sub>h</sub>: DLL\_CNCollision\_REC

The following objects are used to monitor “Collision” error symptoms detected by a CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

The object may be implemented if the error recognition is provided.

|           |                      |             |                |
|-----------|----------------------|-------------|----------------|
| Index     | 1C0A <sub>h</sub>    | Object Type | RECORD         |
| Name      | DLL_CNCollision_REC  |             |                |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: O |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 0 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | O  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a “Collision” error symptom occurs. Its value monitors all “Collision” error symptoms that were detected by the CN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | O  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a “Collision” error symptom occurs and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Collision” error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Threshold_U32   |             |     |
| Data Type     | UNSIGNED32      | Category    | O   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 15              | PDO Mapping | No  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.1)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.12 Object 1C0B<sub>h</sub>: DLL\_CNLossSoC\_REC

The following objects are used to monitor “Loss of Soc” error symptoms detected by a CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

|           |                      |             |                |
|-----------|----------------------|-------------|----------------|
| Index     | 1C0B <sub>h</sub>    | Object Type | RECORD         |
| Name      | DLL_CNLossSoC_REC    |             |                |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: M |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 3               | Access      | const |
| Default Value | 3               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | M  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a “Loss of SoC” error symptom occurs. Its value monitors all “Loss of SoC” error symptoms that were detected by the CN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | M  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a “Loss of SoC” error symptom occurs and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Loss of SoC” error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Threshold_U32   |             |     |
| Data Type     | UNSIGNED32      | Category    | M   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 15              | PDO Mapping | No  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.3.1)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.13 Object 1C0Ch: DLL\_CNLossSoA\_REC

The following objects are used to monitor “Loss of SoA” error symptoms detected by a CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

The object may be implemented only if the error recognition is provided.

|           |                      |             |                   |
|-----------|----------------------|-------------|-------------------|
| Index     | 1C0Ch                | Object Type | RECORD            |
| Name      | DLL_CNLossSoA_REC    |             |                   |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: Cond |

- **Sub-Index 00h: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00h             |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 0 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01h: CumulativeCnt\_U32**

|             |                   |          |    |
|-------------|-------------------|----------|----|
| Sub-Index   | 01h               |          |    |
| Name        | CumulativeCnt_U32 |          |    |
| Data Type   | UNSIGNED32        | Category | O  |
| Value Range | UNSIGNED32        | Access   | rw |

The cumulative counter shall be incremented by 1 every time a “Loss of SoA” error symptom occurs. Its value monitors all “Loss of SoA” error symptoms that were detected by the CN.

- **Sub-Index 02h: ThresholdCnt\_U32**

|             |                  |          |    |
|-------------|------------------|----------|----|
| Sub-Index   | 02h              |          |    |
| Name        | ThresholdCnt_U32 |          |    |
| Data Type   | UNSIGNED32       | Category | O  |
| Value Range | UNSIGNED32       | Access   | ro |

The threshold counter shall be incremented by 8 every time a “Loss of SoA” error symptom occurs and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Loss of SoA” error occurrence.

- **Sub-Index 03h: Threshold\_U32**

|             |               |          |     |
|-------------|---------------|----------|-----|
| Sub-Index   | 03h           |          |     |
| Name        | Threshold_U32 |          |     |
| Data Type   | UNSIGNED32    | Category | O   |
| Value Range | UNSIGNED32    | Access   | rws |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.3.2)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.14 Object 1C0D<sub>h</sub>: DLL\_CNLossPReq\_REC

The following objects are used to monitor “Loss of PReq” error symptoms detected by an isochronous CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

The object may be implemented only if the error recognition is provided.

|           |                      |             |                   |
|-----------|----------------------|-------------|-------------------|
| Index     | 1C0D <sub>h</sub>    | Object Type | RECORD            |
| Name      | DLL_CNLossPReq_REC   |             |                   |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: Cond |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 0 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|             |                   |          |    |
|-------------|-------------------|----------|----|
| Sub-Index   | 01 <sub>h</sub>   |          |    |
| Name        | CumulativeCnt_U32 |          |    |
| Data Type   | UNSIGNED32        | Category | O  |
| Value Range | UNSIGNED32        | Access   | rw |

The cumulative counter shall be incremented by 1 every time a “Loss of PReq” error symptom occurs. Its value monitors all “Loss of PReq” error symptoms that were detected by the CN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|             |                  |          |    |
|-------------|------------------|----------|----|
| Sub-Index   | 02 <sub>h</sub>  |          |    |
| Name        | ThresholdCnt_U32 |          |    |
| Data Type   | UNSIGNED32       | Category | O  |
| Value Range | UNSIGNED32       | Access   | ro |

The threshold counter shall be incremented by 8 every time a “Loss of PReq” error symptom occurs and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “Loss of PReq” error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|             |                 |          |     |
|-------------|-----------------|----------|-----|
| Sub-Index   | 03 <sub>h</sub> |          |     |
| Name        | Threshold_U32   |          |     |
| Data Type   | UNSIGNED32      | Category | O   |
| Value Range | UNSIGNED32      | Access   | rws |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.3.3)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.15 Object 1C0E<sub>h</sub>: DLL\_CNSoCJitter\_REC

The following objects are used to monitor “SoC Jitter” error symptoms detected by a CN. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

The object may be implemented only if the error recognition is provided.

|           |                      |             |                   |
|-----------|----------------------|-------------|-------------------|
| Index     | 1C0E <sub>h</sub>    | Object Type | RECORD            |
| Name      | DLL_CNSoCJitter_REC  |             |                   |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: Cond |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 0 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|             |                   |          |    |
|-------------|-------------------|----------|----|
| Sub-Index   | 01 <sub>h</sub>   |          |    |
| Name        | CumulativeCnt_U32 |          |    |
| Data Type   | UNSIGNED32        | Category | O  |
| Value Range | UNSIGNED32        | Access   | rw |

The cumulative counter shall be incremented by 1 every time a “Soc Jitter” error symptom occurs. Its value monitors all “SoC Jitter” error symptoms that were detected by the CN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|             |                  |          |    |
|-------------|------------------|----------|----|
| Sub-Index   | 02 <sub>h</sub>  |          |    |
| Name        | ThresholdCnt_U32 |          |    |
| Data Type   | UNSIGNED32       | Category | O  |
| Value Range | UNSIGNED32       | Access   | ro |

The threshold counter shall be incremented by 8 every time a “SoC Jitter” error symptom occurs and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the “SoC Jitter” error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|             |                 |          |     |
|-------------|-----------------|----------|-----|
| Sub-Index   | 03 <sub>h</sub> |          |     |
| Name        | Threshold_U32   |          |     |
| Data Type   | UNSIGNED32      | Category | O   |
| Value Range | UNSIGNED32      | Access   | rws |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.7.3.1)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.16 Object 1C0F<sub>h</sub>: DLL\_CRCError\_REC

The following objects are used to monitor CRC errors. The record consists of a cumulative counter and a threshold counter data object and its threshold data object.

|           |                      |             |                |
|-----------|----------------------|-------------|----------------|
| Index     | 1C0F <sub>h</sub>    | Object Type | RECORD         |
| Name      | DLL_CRCError_REC     |             |                |
| Data Type | DLL_ErrorCntRec_TYPE | Category    | MN: -<br>CN: M |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 1 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: CumulativeCnt\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | CumulativeCnt_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | M  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

The cumulative counter shall be incremented by 1 every time a CRC error occurs. Its value monitors all CRC errors that were detected by the CN.

- **Sub-Index 02<sub>h</sub>: ThresholdCnt\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>  |             |    |
| Name          | ThresholdCnt_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | O  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | 0                | PDO Mapping | No |

The threshold counter shall be incremented by 8 every time a CRC error occurs on the CN and decremented by 1 at every cycle without reoccurrence of the error. Its value monitors the quality of network in relation to the CRC error occurrence.

- **Sub-Index 03<sub>h</sub>: Threshold\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Threshold_U32   |             |     |
| Data Type     | UNSIGNED32      | Category    | O   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 15              | PDO Mapping | No  |

Every time ThresholdCnt\_U32 reaches the threshold, a defined action shall proceed and ThresholdCnt\_U32 shall be reset to 0. (See 4.7.5.4)

Threshold Counting may be deactivated by setting Threshold\_U32 to 0. If Threshold Counting is deactivated, no error reaction will occur.

#### 4.7.8.17 Object 1C10<sub>h</sub>: DLL\_CNLossOfLinkCum\_U32

The following objects are used to monitor the “Loss of Link” error source. The cumulative counter shall be incremented by 1 every time a “Loss of Link” error symptom occurs. Its value monitors all “Loss of Link” error sources that were detected by the CN.

The object may be implemented only if the error recognition is provided.

|               |                         |             |                   |
|---------------|-------------------------|-------------|-------------------|
| Index         | 1C10 <sub>h</sub>       | Object Type | VAR               |
| Name          | DLL_CNLossOfLinkCum_U32 |             |                   |
| Data Type     | UNSIGNED32              | Category    | MN: -<br>CN: Cond |
| Value Range   | UNSIGNED32              | Access      | rw                |
| Default Value | 0                       | PDO Mapping | No                |

#### 4.7.8.18 Object 1C12<sub>h</sub>: DLL\_MNCycleSuspendNumber\_U32

The DLL\_MNCycleSuspendNumber\_U32 parameter is used to define the number of cycles that will be suspended, when a collision has occurred. (See also Tab. 28 MN error handling table and 4.7.6.5 Collisions)

|               |                              |             |                |
|---------------|------------------------------|-------------|----------------|
| Index         | 1C12 <sub>h</sub>            | Object Type | VAR            |
| Name          | DLL_MNCycleSuspendNumber_U32 |             |                |
| Data Type     | UNSIGNED32                   | Category    | MN: M<br>CN: - |
| Value Range   | UNSIGNED32                   | Access      | rw             |
| Default Value | 1                            | PDO Mapping | No             |

Value 0 means that it will finish the current cycle and continue with the followed cycle; 1 means, that it suspends the followed cycle.

#### 4.7.8.19 Object 1C13<sub>h</sub>: DLL\_CNSoCJitterRange\_U32

The DLL\_CNSoCJitterRange\_U32 parameter is used to define the range in ns within the SoCJitter may vary.

The object may be implemented only if the error recognition is provided.

|               |                          |             |                   |
|---------------|--------------------------|-------------|-------------------|
| Index         | 1C13 <sub>h</sub>        | Object Type | VAR               |
| Name          | DLL_CNSoCJitterRange_U32 |             |                   |
| Data Type     | UNSIGNED32               | Category    | MN: -<br>CN: Cond |
| Value Range   | UNSIGNED32               | Access      | rw                |
| Default Value | 2000                     | PDO Mapping | No                |

#### 4.7.8.20 Object 1C14<sub>h</sub>: DLL\_CNLossOfSoCTolerance\_U32

The object provides a tolerance interval in [ns] to be applied by CN’s Loss of SoC error recognition (see 4.7.7.3.1).

|               |                              |             |                |
|---------------|------------------------------|-------------|----------------|
| Index         | 1C14 <sub>h</sub>            | Object Type | VAR            |
| Name          | DLL_CNLossOfSoCTolerance_U32 |             |                |
| Data Type     | UNSIGNED32                   | Category    | MN: -<br>CN: M |
| Value Range   | UNSIGNED32                   | Access      | rws            |
| Default Value | 100000                       | PDO Mapping | No             |

#### 4.7.8.21 Object 1C15<sub>h</sub>: DLL\_MNLossStatusResCumCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a cumulative counter of Loss of StatusResponse events. The cumulative counter of the respective CN shall be incremented by 1 every time a “Loss of StatusResponse” error symptom occurs. Its value monitors all “Loss StatusResponse” error symptoms that were detected by the MN.

|           |                                |             |                |
|-----------|--------------------------------|-------------|----------------|
| Index     | 1C15 <sub>h</sub>              | Object Type | ARRAY          |
| Name      | DLL_MNLossStatusResCumCnt_AU32 |             |                |
| Data Type | UNSIGNED32                     | Category    | MN: O<br>CN: - |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CumCnt

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | CumCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | rw |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0 and 1.

#### 4.7.8.22 Object 1C16<sub>h</sub>: DLL\_MNLossStatusResThrCnt\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold counter of Loss of StatusResponse. The threshold counter of the respective CN shall be incremented by 8 every time a “Loss of StatusResponse” error symptom occurs on the MN and decremented by 1 at every StatusRequest without reoccurrence of the error. Its value monitors the quality of network in relation to the “Loss of StatusResponse” error occurrence.

|           |                                |             |                |
|-----------|--------------------------------|-------------|----------------|
| Index     | 1C16 <sub>h</sub>              | Object Type | ARRAY          |
| Name      | DLL_MNLossStatusResThrCnt_AU32 |             |                |
| Data Type | UNSIGNED32                     | Category    | MN: M<br>CN: - |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: ThrCnt

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | ThrCnt                            |             |    |
| --            | --                                | Category    | M  |
| Value Range   | UNSIGNED32                        | Access      | ro |
| Default Value | 0                                 | PDO Mapping | No |

The sub-index is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0 and 1.

#### 4.7.8.23 Object 1C17<sub>h</sub>: DLL\_MNLossStatusResThreshold\_AU32

This array on the MN contains for every CN in the POWERLINK network a threshold of lost StatusResponse.

Every time the respective DLL\_MNLossStatusResThrCnt\_AU32 value reaches the threshold, a defined action shall proceed and the respective DLL\_MNLossStatusResThrCnt\_AU32 value shall be reset to 0. (See 4.7.7.1)

Threshold counting may be deactivated by setting respective DLL\_MNLossStatusResThreshold\_AU32 value to 0. If Threshold Counting is deactivated, no error reaction will occur.

|           |                                   |             |                |
|-----------|-----------------------------------|-------------|----------------|
| Index     | 1C17 <sub>h</sub>                 | Object Type | ARRAY          |
| Name      | DLL_MNLossStatusResThreshold_AU32 |             |                |
| Data Type | UNSIGNED32                        | Category    | MN: M<br>CN: - |

- **Sub-Index 0h: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00h             |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

**Sub-Index 01h – FEh: Threshold**

|               |            |             |     |
|---------------|------------|-------------|-----|
| Sub-Index     | 01h – FEh  |             |     |
| Name          | Threshold  |             |     |
| --            | --         | Category    | M   |
| Value Range   | UNSIGNED32 | Access      | rws |
| Default Value | 15         | PDO Mapping | No  |

The sub-index is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0 and 1.

#### 4.7.8.24 Object 0424<sub>h</sub>: DLL\_ErrorCntRec\_TYPE

|                 |                      |                   |            |
|-----------------|----------------------|-------------------|------------|
| Index           | 0424 <sub>h</sub>    | Object Type       | DEFSTRUCT  |
| Name            | DLL_ErrorCntRec_TYPE |                   |            |
| Sub-Index       | Component Name       | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries      | 03 <sub>h</sub>   |            |
| 01 <sub>h</sub> | CumulativeCnt_U32    | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub> | ThresholdCnt_U32     | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub> | Threshold_U32        | 0007 <sub>h</sub> | UNSIGNED32 |

## 5 Network / Transport Layer

### 5.1 Internet Protocol (IP)

The Internet Protocol version 4 (IPv4) and its referred transport layer protocols UDP and TCP are the preferred protocols in the asynchronous phase.

- RFC 791 defines Internet Protocol (IP)
- RFC 768 defines the User Datagram Protocol (UDP)
- RFC 793 defines the Transmission Control Protocol (TCP).

#### 5.1.1 IP Host Requirements

This section discusses the requirements for a POWERLINK node implementation of the Internet Protocol. To use IP transparently in the asynchronous phase, the POWERLINK nodes shall be conformed to RFC1122 “Requirements for Internet Hosts -- Communication Layers”. However, this would prohibit several low-end POWERLINK nodes, communicating with IP in the asynchronous phase. Therefore the following conformance classes are introduced.

##### 5.1.1.1 Nodes without IP Communication

MNs shall support IP communication.

CNs that don't support SDO via UDP/IP do not need an IP stack.

At the CN, support of IP communication may be indicated by the device description entry D\_NWL\_IPSupport\_BOOL.

##### 5.1.1.2 Minimum Requirements for SDO Communication

This conformance class shall be fulfilled, to ensure that a POWERLINK node is able to communicate in the asynchronous phase via SDO. It is not guaranteed that protocols from the Internet Protocol suite will work – e.g. Socket communication, TFTP, FTP HTTP, etc.

###### 5.1.1.2.1 IP Stack Requirements

To communicate via IPv4 in the asynchronous phase, the POWERLINK node shall at least cope with 256 Bytes SDO payload. Therefore, the IP stack shall process at least C\_DLL\_MIN\_ASYNC\_MTU bytes (including IP header and IP payload) that can be received. IP fragmentation and reassembly is not required to fulfil this conformance class. Hence the size of the asynchronous phase shall be equal or bigger than 256 Bytes SDO payload.

###### 5.1.1.2.2 UDP Requirements

A POWERLINK node shall implement the User Datagram Protocol specified in RFC 768 and shall support at least one UDP socket.

###### 5.1.1.3 Minimum Requirements for Standard IP Communication

This conformance class shall be compatible to RFC 1122 and shall cover the entire conformance class for minimum requirements for SDO communication listed above. For convenience the following core requirements are listed.

### 5.1.1.3.1 IP Stack Requirements

- The IPv4 layer shall implement reassembly of IP datagrams – see RFC1122 chapter 3.3.2 Reassembly.
- The IPv4 layer shall implement a mechanism to fragment outgoing datagrams intentionally – see RFC1122 chapter 3.3.3 Fragmentation.
- In general an IPv4 capable POWERLINK node shall at least process IP datagrams up to 576 bytes (including header and data) – see RFC 1122 chapter 3.3.2/3.3.3.
- ICMP (RFC 792) support is optional. It shall be indicated by D\_NWL\_ICMPSupport\_BOOL.

## 5.1.2 IP Addressing

Each IP-capable POWERLINK node possesses an IPv4 address, a subnet mask and default gateway. These attributes are referred to as the IP parameters.

- IPv4 Address**

The private class C Net ID 192.168.100.0 shall be used for a POWERLINK network – see RFC1918. A class C network provides 254 (1-254) IP addresses, which matches the number of valid POWERLINK Node ID's. The Host ID of the private class C Net ID 192.168.100.0 shall be identical to the POWERLINK Node ID. Hence the last byte of the IP address (Host ID) has the same value as the POWERLINK Node ID. The following figure illustrates the construction of the IP address.



Fig. 38. Construction of the IPv4 address

Remarks:

Knowing the Node ID of a POWERLINK node, its IP address and vice versa can be determined easily without any communication overhead.

- Subnet mask**

The subnet mask of a POWERLINK node shall be 255.255.255.0. This is the subnet mask of a class C net.

- Default Gateway**

The Default Gateway preset shall use the IP address 192.168.100.254. The value may be modified to another valid IP address.

Generally the IPv4 Address and Subnet mask parameters of a POWERLINK node shall be fixed. The downside of the fixed IP parameters are compensated by the POWERLINK Router using Network Address Translation (see 9.1.4.2.2).

The following table summarises the default IP parameters.

| IP Parameter    | IP address                        |
|-----------------|-----------------------------------|
| IP address      | 192.168.100.<POWERLINK Node ID>   |
| Subnet mask     | 255.255.255.0                     |
| Default Gateway | 192.168.100.254 (may be modified) |

Tab. 29 IP parameters of a POWERLINK node

### 5.1.3 Address Resolution

The Address Resolution Protocol (ARP) specified in RFC 826 shall be used to obtain the IP to Ethernet MAC relation of a POWERLINK node. Depending on the POWERLINK node state:

- NMT\_CS\_EPL\_MODE and NMT\_MS\_EPL\_MODE state: ARP shall be performed in the asynchronous phase. To reduce the traffic in the asynchronous phase, the MN may determine the IP to MAC address relation from the ident process.

- NMT\_CS\_BASIC\_ETHERNET state: ARP shall be performed like an IEEE802.3 compliant node does, using CSMA/CD.

Optional the MN or CN may send the NMT Managing command NMTFlushArpEntry (see 7.3.2.1.2) if one of them detects that an upcoming node has a new MAC address. This can be done to flush the ARP cache of all nodes in the POWERLINK network. The POWERLINK node may process NMTFlushArpEntry.

Alternativley an unsolicited ARP request frame (containing its IP address) may be broadcasted initiated by the respective POWERLINK node at startup. As a result, the neighbours ARP caches shall be updated.

## 5.1.4 Hostname

Each IP capable POWERLINK node shall have a hostname. The hostname is of type VISIBLE\_STRING32. The hostname can be used to access POWERLINK nodes with its name instead of its IP address.

The admissible values of type VISIBLE\_STRING for the hostname shall be restricted to:

- 30<sub>h</sub> - 39<sub>h</sub> (0 - 9)
- 41<sub>h</sub> - 5A<sub>h</sub> (A - Z)
- 61<sub>h</sub> - 7A<sub>h</sub> (a - z)
- 2D<sub>h</sub> (-)

The data are interpreted as ISO 646-1973(E) 7-bit coded characters.

The default hostname shall be constructed from the POWERLINK Node ID and the Vendor ID parted by the character “-” (<POWERLINK Node ID>-<Vendor ID>). POWERLINK Node ID and the Vendor ID shall be hexadecimally coded.

If no hostname is explicitly assigned, the POWERLINK node shall use the default hostname instead.

The hostname located on the POWERLINK node shall be set with the NMT Managing command NMTNetHostNameSet (refer 7.3.2.1.1). Modification of the hostname value shall not take effect until the POWERLINK node enters the NMT\_GS\_INITIALISATION state. The hostname is read by the ASnd with the Ident Response Service.

A hostname to IP address resolution service may be provided to gather the hostname to IP address association of all POWERLINK nodes within a POWERLINK network. This service configures for example the DNS table of the DNS server located on the POWERLINK to legacy Ethernet Router or a local hostname table on a diagnostics device.

## 5.1.5 Object description

### 5.1.5.1 Object 1E4A<sub>h</sub>: NWL\_IpGroup\_REC

The NWL\_IpGroup\_REC object is a subset of the IP Group RFC1213. The object specifies information about the IP stack.

The Object shall be supported only if IP is supported by the device.

|           |                   |             |        |
|-----------|-------------------|-------------|--------|
| Index     | 1E4A <sub>h</sub> | Object Type | RECORD |
| Name      | NWL_IpGroup_REC   |             |        |
| Data Type | NWL_IpGroup_TYPE  | Category    | Cond   |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2 .. 3          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: Forwarding\_BOOL**

|               |   |             |        |
|---------------|---|-------------|--------|
| Sub-Index     | 01 <sub>h</sub>                             |             |        |
| Name          | Forwarding_BOOL                             |             |        |
| Data Type     | BOOLEAN                                     | Category    | M      |
| Value Range   | FALSE (Not-forwarding)<br>TRUE (forwarding) | Access      | ro/rws |
| Default Value | FALSE (Not-forwarding)                      | PDO Mapping | No     |

The indication whether this entity is acting as an IP router in respect to the forwarding of datagrams received by, but not addressed to this entity. IP routers forward datagrams. IP hosts do not (except those source-routed via the host).

The ability to forward datagrams is indicated by D\_NWL\_Forward\_BOOL. On device not supporting forwarding sub-index 01<sub>h</sub> shall be FALSE, access shall be ro. On device supporting forwarding sub-index 01<sub>h</sub> may be FALSE or TRUE, access shall be rw.

- **Sub-Index 02<sub>h</sub>: DefaultTTL\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | DefaultTTL_U16  |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 64              | PDO Mapping | No  |

The default value inserted into the Time-To-Live field of the IP header of datagrams originated at this entity, whenever a TTL value is not supplied by the transport layer protocol.

- **Sub-Index 03<sub>h</sub>: ForwardDatagrams\_U32**

|               |                      |             |    |
|---------------|----------------------|-------------|----|
| Sub-Index     | 03 <sub>h</sub>      |             |    |
| Name          | ForwardDatagrams_U32 |             |    |
| Data Type     | UNSIGNED32           | Category    | O  |
| Value Range   | UNSIGNED32           | Access      | ro |
| Default Value | -                    | PDO Mapping | No |

The number of input datagrams for which this entity was not their final IP destination, as a result of which an attempt was made to find a route to forward them to the final destination.

### 5.1.5.2 Object 1E40<sub>h</sub> .. 1E49<sub>h</sub>: NWL\_IpAddrTable\_Xh\_REC

The IP address table contains this entity's IP addressing information. The NWL\_IpAddrTable\_Xh\_REC object is a subset of the IP Group RFC1213. It assigns IP parameters to an interface indicated by

NMT\_ItfGroup\_Xh\_REC.ItfIndex\_U16. The IP address table shall have 1 to 10 entries that may be configured via SDO.

The Objects shall be supported only if IP is supported by the device.

POWERLINK interfaces shall be described by the low order objects (e.g. 1E40<sub>h</sub>, 1E41<sub>h</sub>, ...).

|           |  |             |  |
|-----------|--|-------------|--|
| Index     | 1E40 <sub>h</sub> .. 1E49 <sub>h</sub> | Object Type | RECORD   |
| Name      | NWL_IpAddrTable_Xh_REC                 |             |  |
| Data Type | NWL_IpAddrTable_TYPE                   | Category    | 1E40 <sub>h</sub> : Cond<br>1E41 <sub>h</sub> .. 1E49 <sub>h</sub> : O |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 5               | Access      | const |
| Default Value | 5               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: IfIndex\_U16**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> |             |    |
| Name          | IfIndex_U16     |             |    |
| Data Type     | UNSIGNED16      | Category    | M  |
| Value Range   | 1..10           | Access      | ro |
| Default Value | -               | PDO Mapping | No |

The index value which uniquely identifies the interface to which this entry is applicable. The interface identified by a particular value of this index is the same interface as identified by the same value of NMT\_InterfaceGroup\_Xh\_REC.InterfaceIndex\_U16.

- **Sub-Index 02<sub>h</sub>: Addr\_IPAD**

|               |                 |             |        |
|---------------|-----------------|-------------|--------|
| Sub-Index     | 02 <sub>h</sub> |             |        |
| Name          | Addr_IPAD       |             |        |
| Data Type     | IP_ADDRESS      | Category    | M      |
| Value Range   | IP_ADDRESS      | Access      | ro/rws |
| Default Value | -               | PDO Mapping | No     |

The IP address to which this entry's addressing information pertains.

If the object describes an Ethernet POWERLINK interface, access shall be ro. Addr\_IPAD shall be 192.168.100.xxx with xxx = NMT\_EPLNodeID\_REC.Nodeld\_U8 (cf. 5.1.2).

- **Sub-Index 03<sub>h</sub>: NetMask\_IPAD**

|               |                 |             |        |
|---------------|-----------------|-------------|--------|
| Sub-Index     | 03 <sub>h</sub> |             |        |
| Name          | NetMask_IPAD    |             |        |
| Data Type     | IP_ADDRESS      | Category    | M      |
| Value Range   | IP_ADDRESS      | Access      | ro/rws |
| Default Value | -               | PDO Mapping | No     |

The subnet mask associated with the IP address of this entry. The value of the mask is an IP address with all the network bits set to 1 and all the hosts bits set to 0.

If the object describes an Ethernet POWERLINK interface, access shall be ro. NetMask\_IPAD shall be 255.255.255.0 (cf. 5.1.2).

- **Sub-Index 04<sub>h</sub>: ReasmMaxSize\_U16**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 04 <sub>h</sub>  |             |    |
| Name          | ReasmMaxSize_U16 |             |    |
| Data Type     | UNSIGNED16       | Category    | M  |
| Value Range   | UNSIGNED16       | Access      | ro |
| Default Value | -                | PDO Mapping | No |

The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.

- **Sub-Index 05<sub>h</sub>: DefaultGateway\_IPAD**

|               |                     |             |     |
|---------------|---------------------|-------------|-----|
| Sub-Index     | 05 <sub>h</sub>     |             |     |
| Name          | DefaultGateway_IPAD |             |     |
| Data Type     | IP_ADDRESS          | Category    | M   |
| Value Range   | IP_ADDRESS          | Access      | rws |
| Default Value | -                   | PDO Mapping | No  |

The default gateway associated with the IP address of this entry.

If the object describes an Ethernet POWERLINK interface, the entry shall indicate the router type 1 device. Default value shall be C\_ADR\_RT1\_DEF\_NODE\_ID .

### 5.1.5.3 Object 0425<sub>h</sub>: NWL\_IpGroup\_TYPE

| Index           | 0425 <sub>h</sub>    | Object Type     | DEFSTRUCT  |
|-----------------|----------------------|-----------------|------------|
| Name            | NWL_IpGroup_TYPE     |                 |            |
| Sub-Index       | Component Name       | Value           | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries      | 03 <sub>h</sub> |            |
| 01 <sub>h</sub> | Forwarding_BOOL      | 0001h           | BOOLEAN    |
| 02 <sub>h</sub> | DefaultTTL_U16       | 0006h           | UNSIGNED16 |
| 03 <sub>h</sub> | ForwardDatagrams_U32 | 0007h           | UNSIGNED32 |

### 5.1.5.4 Object 0426<sub>h</sub>: NWL\_IpAddrTable\_TYPE

| Index           | 0426 <sub>h</sub>    | Object Type     | DEFSTRUCT  |
|-----------------|----------------------|-----------------|------------|
| Name            | NWL_IpAddrTable_TYPE |                 |            |
| Sub-Index       | Component Name       | Value           | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries      | 05 <sub>h</sub> |            |
| 01 <sub>h</sub> | IfIndex_U16          | 0006h           | UNSIGNED16 |
| 02 <sub>h</sub> | Addr_IPAD            | 0402h           | IP_ADDRESS |
| 03 <sub>h</sub> | NetMask_IPAD         | 0402h           | IP_ADDRESS |
| 04 <sub>h</sub> | ReasmMaxSize_U16     | 0006h           | UNSIGNED16 |
| 05 <sub>h</sub> | DefaultGateway_IPAD  | 0402h           | IP_ADDRESS |

## 5.2 POWERLINK compliant UDP/IP format

In order to enable the transmission of POWERLINK frames encapsulated in UDP/IP frames, the payload portion of the UDP/IP frame shall be leaded by a slightly modified POWERLINK frame header.

The parameter MessageType defined by Ethernet POWERLINK shall be in conformance to the requirements of 4.6.1.1.1. Destination and Source fields of the original POWERLINK header shall be reserved but shall not be supported, when transmission occurs via UDP/IP.

| Octet Offset <sup>14</sup> | Bit Offset              |   |   |   |   |   |   |   | entry defined by   |  |
|----------------------------|-------------------------|---|---|---|---|---|---|---|--------------------|--|
|                            | 7                       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |                    |  |
| 0 - 5                      | Destination MAC Address |   |   |   |   |   |   |   | Ethernet type II   |  |
| 6 - 11                     | Source MAC Address      |   |   |   |   |   |   |   |                    |  |
| 12 - 13                    | EtherType               |   |   |   |   |   |   |   |                    |  |
| 14 - 33                    | IP Header               |   |   |   |   |   |   |   | RFC 791            |  |
| 34 - 41                    | UDP Header              |   |   |   |   |   |   |   | RFC 768            |  |
| 42                         | MessageType             |   |   |   |   |   |   |   | Ethernet POWERLINK |  |
| 43                         | reserved (Destination)  |   |   |   |   |   |   |   |                    |  |
| 44                         | reserved (Source)       |   |   |   |   |   |   |   |                    |  |
| 45                         | ServiceID / reserved    |   |   |   |   |   |   |   |                    |  |
| 46 - n                     | Payload Data            |   |   |   |   |   |   |   | Application        |  |
| n+1 - n+4                  | CRC32                   |   |   |   |   |   |   |   | Ethernet type II   |  |

$n \geq 59$

Tab. 30 POWERLINK compliant UDP/IP frame structure

## 5.3 POWERLINK Sequence Layer

see 6.3.2.3, 6.3.3.1

<sup>14</sup> Octet Offset refers to the start of the Ethernet telegram.

## 6 Application Layer

### 6.1 Data Types and Encoding Rules

This paragraph describes the data formats and encoding rules to be used by frames according to the Ethernet POWERLINK syntax (EtherType = 88AB<sub>h</sub>). The rules apply to the POWERLINK-Content, service specific header and data payload embedded into the Ethernet frame. The encoding of the Ethernet frame follows the rules of IEEE 802.3.

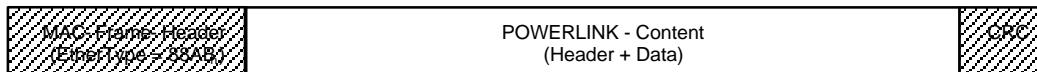


Fig. 39. POWERLINK frame structure

The rules also apply to POWERLINK specific payloads that are embedded into non-POWERLINK frame types (EtherType ≠ 88AB<sub>h</sub>), e.g. SDO-Transfer via UDP/IP etc. The Ethertype specific encoding of these frames, described by RFC 791 (IP) and RFC 768 (UDP), is not the concern of these rules.

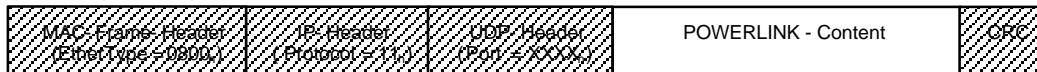


Fig. 40. POWERLINK compliant UDP/IP frame structure

The encoding of non-POWERLINK frames (EtherType ≠ 88AB<sub>h</sub>) without POWERLINK specific payload is not the concern of these rules.

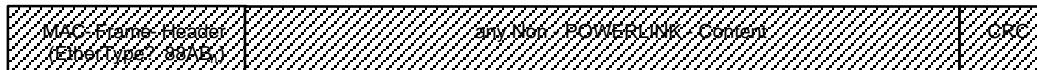


Fig. 41. Legacy Ethernet frame structure

#### 6.1.1 General Description of Data Types and Encoding Rules

In order to exchange meaningful data across a POWERLINK network, the data format and its meaning must be known by the producer and consumer(s). This document specifies data formats and meanings using data types.

The encoding rules define the representation of values of data types and the POWERLINK network transfer syntax for the representations. Values are represented as bit sequences. Bit sequences are transferred in sequences of octets (bytes). For numerical data types the encoding is little endian style as shown in Tab. 31.

Applications often require data types beyond the basic data types. Using the compound data type mechanism the list of available data types can be extended. Some general extended data types are defined as "Visible String" or "Time of Day" (for example see 6.1.6.2 and 6.1.6.4). The compound data types are a means to implement user defined "DEFTYPES" in the terminology of this specification and not "DEFSTRUCTS" (see 6.2.2).

#### 6.1.2 Data Type Definitions

A data type determines a relation between values and encoding for data of that type. Names are assigned to data types in their type definitions. The syntax of data and data type definitions is as follows (see EN 61131-3).

|                 |     |   |
|-----------------|-----|---|
| data_definition | ::= | type_name data_name                         |
| type_definition | ::= | constructor type_name                       |
| constructor     | ::= | compound_constructor  <br>basic_constructor |

|                       |   |
|-----------------------|---|
| compound_constructor  | ::= array_constructor   structure_constructor   |
| array_constructor     | ::= 'ARRAY' '[' length ']' 'OF' type_name   |
| structure_constructor | ::= 'STRUCT' 'OF' component_list  |
| component_list        | ::= component { ',' component }   |
| component             | ::= type_name component_name  |
| basic_constructor     | ::= 'BOOLEAN'  <br>'VOID' bit_size  <br>'BYTE' bit_size  <br>'INTEGER' bit_size  <br>'UNSIGNED' bit_size  <br>'REAL32'  <br>'REAL64'  <br>'NIL' |
| bit_size              | ::= '1'   '2'   <...>   '64'  |
| length                | ::= positive_integer  |
| data_name             | ::= symbolic_name   |
| type_name             | ::= symbolic_name   |
| component_name        | ::= symbolic_name   |
| symbolic_name         | ::= letter { [ '_'] ( letter   digit ) }  |
| positive_integer      | ::= ( '1'   '2'   <...>   '9' ) { digit }   |
| letter                | ::= 'A'   'B'   <...>   'Z'   'a'   'b'   <...>   'z'   |
| digit                 | ::= '0'   '1'   <...>   '9'   |

Recursive definitions are not allowed.

The data type defined by type\_definition is termed either basic, when the constructor is basic\_constructor, or compound, when the constructor is compound\_constructor.

## 6.1.3 Bit Sequences

### 6.1.3.1 Definition of Bit Sequences

A bit can take the values 0 or 1.

A bit sequence b is an ordered set of 0 or more bits.

If a bit sequence b contains more than 0 bits, they are denoted as  $b_j$ ,  $j \geq 0$ .

Let  $b_0, \dots, b_{n-1}$  be bits, n a positive integer. Then

$$b = b_0 b_1 \dots b_{n-1}$$

is called a bit sequence of length  $|b| = n$ .

The empty bit sequence of length 0 is denoted  $\varepsilon$ .

*Examples: 10110100, 1, 101, etc. are bit sequences.*

The inversion operator ( $\neg$ ) on bit sequences assigns to a bit sequence

$$\bar{b} = \bar{b}_0 \bar{b}_1 \dots \bar{b}_{n-1}$$

the bit sequence

$$\neg b = \neg b_0 \neg b_1 \dots \neg b_{n-1}$$

Here  $\neg 0 = 1$  and  $\neg 1 = 0$  on bits.

The basic operation on bit sequences is concatenation.

Let  $a = a_0 \dots a_{m-1}$  and  $b = b_0 \dots b_{n-1}$  be bit sequences. Then the concatenation of a and b, denoted  $ab$ , is

$$a_b = a_0 \dots a_{m-1} b_0 \dots b_{n-1}$$

*Example: (10)(111) = 10111 is the concatenation of 10 and 111.*

The following holds for arbitrary bit sequences a and b:

$$|ab| = |a| + |b|$$

and

$\varepsilon a = a\varepsilon = a$

### 6.1.3.2 Transfer Syntax for Bit Sequences

For transmission across a POWERLINK network a bit sequence is reordered into a sequence of octets. Here and in the following hexadecimal notation is used for octets. Let  $b = b_0 \dots b_{n-1}$  be a bit sequence with  $n \leq 11920_d$  ( $1490_d$  Byte \*  $8_d$  Bit/Byte).

Denote  $k$  a non-negative integer such that  $8(k-1) < n \leq 8k$ . Then  $b$  is transferred in  $k$  octets assembled as shown in Tab. 31. The bits  $b_i$ ,  $i \geq n$  of the highest numbered octet shall be ignored.

| octet number | 1.              | 2.                 | k.                        |
|--------------|-----------------|--------------------|---------------------------|
|              | $b_7 \dots b_0$ | $b_{15} \dots b_8$ | $b_{8k-1} \dots b_{8k-8}$ |

Tab. 31 Transfer syntax for bit sequences

Octet 1 is transmitted first and octet  $k$  is transmitted last. The bit sequence is transferred as follows across the POWERLINK network:

$b_7, b_6, \dots, b_0, b_{15}, \dots, b_8, \dots$

*Example:*

| Bit 9  | ...      | Bit 0     |
|--------|----------|-----------|
| $10_b$ | $0001_b$ | $1100_b$  |
| $2_h$  | $1_h$    | $C_h$     |
|        |          | $= 21C_h$ |

The bit sequence  $b = b_0 \dots b_9 = 0011\ 1000\ 01_b$  represents an UNSIGNED10 with the value  $21C_h$  and is transferred in two octets:

First  $1C_h$  and then  $02_h$ .

## 6.1.4 Basic Data Types

For basic data types “type\_name” equals the literal string of the associated constructor (cf. symbolic\_name), e.g.,

BOOLEAN      BOOLEAN

is the type definition for the BOOLEAN data type.

### 6.1.4.1 NIL

Data of basic data type NIL is represented by  $\varepsilon$ .

### 6.1.4.2 Boolean

Data of basic data type BOOLEAN attains the values TRUE or FALSE.

The values are represented as bit sequences of length 1. The value TRUE is represented by the bit sequence 1, and FALSE by 0.

A BOOLEAN shall be transferred over the network as UNSIGNED8 of value 1 (TRUE) resp. 0 (FALSE). Sequent BOOLEANS may be packed to one UNSIGNED8. Sequences of BOOLEAN and BIT type items may be also packed to one UNSIGNED8.

### 6.1.4.3 Void

Data of basic data type VOIDn is represented as bit sequences of length n bits.

The value of data of type VOIDn is undefined. The bits in the sequence of data of type VOIDn must either be specified explicitly or else marked "do not care".

Data of type VOIDn is useful for reserved fields and for aligning components of compound values on octet boundaries.

### 6.1.4.4 Bit

Data of basic data type BITn are represented as bit sequences of length n bits.

The interpretation of the value of type BITn data is defined by the context of the variable, that is implemented using BITn data type.

If a BIT value is not member of compound data type, it shall be transferred over the network as UNSIGNED8 of value 1 resp. 0. Sequent BITS may be packed to one UNSIGNED8. Sequences of BOOLEAN and BIT type items may be also packed to one UNSIGNED8.

### 6.1.4.5 Unsigned Integer

Data of basic data type UNSIGNEDn has values in the non-negative integers. The value range is 0, ...,  $2^n - 1$ . The data is represented as bit sequences of length n.

The bit sequence

$$b = b_0 \dots b_{n-1}$$

is assigned the value

$$\text{UNSIGNED}_n(b) = b_{n-1} 2^{n-1} + \dots + b_1 2^1 + b_0 2^0$$

Note that the bit sequence starts on the left with the least significant byte.

*Example: The value  $266_d = 10A_h$  with data type UNSIGNED16 is transferred in two octets across the bus, first  $0A_h$  and then  $01_h$ .*

The following UNSIGNEDn data types are transferred as shown below:

| octet number | 0          | 1             | 2                | 3                | 4                | 5                | 6                | 7                |
|--------------|------------|---------------|------------------|------------------|------------------|------------------|------------------|------------------|
| UNSIGNED8    | $b_7..b_0$ |               |                  |                  |                  |                  |                  |                  |
| UNSIGNED16   | $b_7..b_0$ | $b_{15}..b_8$ |                  |                  |                  |                  |                  |                  |
| UNSIGNED24   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ |                  |                  |                  |                  |                  |
| UNSIGNED32   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ | $b_{31}..b_{24}$ |                  |                  |                  |                  |
| UNSIGNED40   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ | $b_{31}..b_{24}$ | $b_{39}..b_{32}$ |                  |                  |                  |
| UNSIGNED48   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ | $b_{31}..b_{24}$ | $b_{39}..b_{32}$ | $b_{47}..b_{40}$ |                  |                  |
| UNSIGNED56   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ | $b_{31}..b_{24}$ | $b_{39}..b_{32}$ | $b_{47}..b_{40}$ | $b_{55}..b_{48}$ |                  |
| UNSIGNED64   | $b_7..b_0$ | $b_{15}..b_8$ | $b_{23}..b_{16}$ | $b_{31}..b_{24}$ | $b_{39}..b_{32}$ | $b_{47}..b_{40}$ | $b_{55}..b_{48}$ | $b_{63}..b_{56}$ |

Tab. 32 Transfer syntax for data type UNSIGNEDn

The data types UNSIGNED24, UNSIGNED40, UNSIGNED48 and UNSIGNED56 should not be applied by new applications.

UNSIGNEDn data types of length deviating from the values listed by Tab. 32 may be applied by compound data types only.

### 6.1.4.6 Signed Integer

Data of basic data type INTEGERn has values in the integers. The value range is  $-2^{n-1}, \dots, 2^{n-1} - 1$ .

The data is represented as bit sequences of length n.

The bit sequence

$$b = b_0 \dots b_{n-1}$$

is assigned the value

$$\text{INTEGER}_n(b) = b_{n-2} 2^{n-2} + \dots + b_1 2^1 + b_0 2^0 \quad \text{if } b_{n-1} = 0$$

and, performing two's complement arithmetic,

$$\text{INTEGER}_n(b) = -\text{INTEGER}_n(^b) - 1 \quad \text{if } b_{n-1} = 1$$

Note that the bit sequence starts on the left with the least significant bit.

*Example: The value  $-266_d = FEF6_h$  with data type INTEGER16 is transferred in two octets across the bus, first  $F6_h$  and then  $FE_h$ .*

The following INTEGERn data types are transferred as shown below:

| octet number | 0                             | 1                              | 2                               | 3                               | 4                               | 5                               | 6                               | 7                               |
|--------------|-------------------------------|--------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|---------------------------------|
| INTEGER8     | b <sub>7..b<sub>0</sub></sub> |                                |                                 |                                 |                                 |                                 |                                 |                                 |
| INTEGER16    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> |                                 |                                 |                                 |                                 |                                 |                                 |
| INTEGER24    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> |                                 |                                 |                                 |                                 |                                 |
| INTEGER32    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> |                                 |                                 |                                 |                                 |
| INTEGER40    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> | b <sub>39..b<sub>32</sub></sub> |                                 |                                 |                                 |
| INTEGER48    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> | b <sub>39..b<sub>32</sub></sub> | b <sub>47..b<sub>40</sub></sub> |                                 |                                 |
| INTEGER56    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> | b <sub>39..b<sub>32</sub></sub> | b <sub>47..b<sub>40</sub></sub> | b <sub>55..b<sub>48</sub></sub> |                                 |
| INTEGER64    | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> | b <sub>39..b<sub>32</sub></sub> | b <sub>47..b<sub>40</sub></sub> | b <sub>55..b<sub>48</sub></sub> | b <sub>63..b<sub>56</sub></sub> |

Tab. 33 Transfer syntax for data type INTEGERn

The data types INTEGER24, INTEGER40, INTEGER48 and INTEGER56 should not be applied by new applications.

INTEGERn data types of length deviating from the values listed by Tab. 33 may be applied by compound data types only.

### 6.1.4.7 Floating-Point Numbers

Data of basic data types REAL32 and REAL64 have values in the real numbers.

The data type REAL32 is represented as bit sequence of length 32. The encoding of values follows the IEEE 754-1985 Standard for single precision floating-point.

The data type REAL64 is represented as bit sequence of length 64. The encoding of values follows the IEEE 754-1985 Standard for double precision floating-point numbers.

A bit sequence of length 32 either has a value (finite non-zero real number, ±0, ±\_) or is NaN (not-a-number). The bit sequence

$$b = b_0 \dots b_{31}$$

is assigned the value (finite non-zero number)

$$\text{REAL32}(b) = (-1)^S 2^{E-127} (1+F)$$

Here

S = b<sub>31</sub> is the sign.

E = b<sub>30</sub> 2<sup>7</sup> + ... + b<sub>23</sub> 2<sup>0</sup>, 0 < E < 255, is the un-biased exponent.

F = 2<sup>-23</sup> (b<sub>22</sub> 2<sup>22</sup> + ... + b<sub>1</sub> 2<sup>1</sup> + b<sub>0</sub> 2<sup>0</sup>) is the fractional part of the number.

E = 0 is used to represent ±0. E = 255 is used to represent infinities and NaN's.

Note that the bit sequence starts on the left with the least significant bit.

Example:

$$6.25 = 2^{E-127} (1+F) \text{ with}$$

$$E=129=2^7+2^6+2^5+2^4+2^3+2^2+2^1+2^0$$

F = 2<sup>-23</sup> (b<sub>22</sub> 2<sup>22</sup> + ... + b<sub>1</sub> 2<sup>1</sup> + b<sub>0</sub> 2<sup>0</sup>) hence the number is represented as:

| S               | E                                 | F                                |
|-----------------|-----------------------------------|----------------------------------|
| b <sub>31</sub> | b <sub>30 .. b<sub>23</sub></sub> | b <sub>22 .. b<sub>0</sub></sub> |
| 0               | 100 0000 1                        | 100 1000 0000 0000 0000 0000     |

$$6.25 = b_0 \dots b_{31} = 0000 \ 0000 \ 0000 \ 0000 \ 0001 \ 0011 \ 0000 \ 0010$$

It is transferred in the following order:

| octet number | 0                             | 1                              | 2                               | 3                               |
|--------------|-------------------------------|--------------------------------|---------------------------------|---------------------------------|
| REAL32       | 00 <sub>h</sub>               | 00 <sub>h</sub>                | C8 <sub>h</sub>                 | 40 <sub>h</sub>                 |
|              | b <sub>7..b<sub>0</sub></sub> | b <sub>15..b<sub>8</sub></sub> | b <sub>23..b<sub>16</sub></sub> | b <sub>31..b<sub>24</sub></sub> |

Tab. 34 Transfer syntax of data type REAL32

### 6.1.4.8 MAC Address

The data type MAC\_ADDRESS is used to describe the MAC address of an Ethernet adapter. It is represented by 6 octets. It may be interpreted as UNSIGNED48.

The MAC address is divided into two sub-items:

- Organisational Unique Identifier (OUI), describing the adapter's manufacturer
- adapter specific unique identifier

| Octet   | 5               | 4               | 3               | 2                               | 1               | 0               |  |
|---------|-----------------|-----------------|-----------------|---------------------------------|-----------------|-----------------|--|
|         | OUI             |                 |                 | adapter's individual identifier |                 |                 |  |
| Example | 00 <sub>h</sub> | 0A <sub>h</sub> | 86 <sub>h</sub> | xx <sub>h</sub>                 | xx <sub>h</sub> | xx <sub>h</sub> |  |

Tab. 35 MAC address encoding, example 00-0A-86-xx-xx-xx shows a Lenze device

### 6.1.4.9 IP address

The data type IP\_ADDRESS is used to describe the IP address of a network adapter, subnet masks etc. It is represented by 4 octets. It may be interpreted as UNSIGNED32.

The IP address is divided to a Net ID part and a Host ID part. Tab. 36 shows the the coding of Net and Host ID. The bit consumption of these parts depends on the type of the subnet and is therefore not fixed.

| Octet   | 3               | 2               | 1               | 0               |
|---------|-----------------|-----------------|-----------------|-----------------|
|         | NetID           |                 |                 | HostID          |
| Example | C0 <sub>h</sub> | A8 <sub>h</sub> | 64 <sub>h</sub> | F0 <sub>h</sub> |
|         | 192             | 168             | 100             | 240             |

Tab. 36 IP address encoding, example shows the IP address of a POWERLINK MN  
192.168.100.240

### 6.1.5 Compound Data Types

Type definitions of compound data types expand to a unique list of type definitions involving only basic data types. Correspondingly, data of compound type 'type\_name' are ordered lists of component data named 'component\_name\_i' of basic type 'basic\_type\_i'.

Compound data types constructors are ARRAY and STRUCT OF.

STRUCT OF

|              |                   |
|--------------|-------------------|
| basic_type_1 | component_name_1, |
| basic_type_2 | component_name_2, |
| ...          | ...               |
| basic_type_N | component_name_N  |

type\_name

ARRAY [ length ] OF basic\_type      type\_name

The bit sequence representing data of compound type is obtained by concatenating the bit sequences representing the component data.

Assume that the components 'component\_name\_i' are represented by their bit sequences  
b(i), for i = 1,...,N.

Then the compound data is represented by the concatenated sequence.

b<sub>0</sub>(1) .. b<sub>n-1</sub>(1) .. b<sub>n-1</sub>(N).

Example: Consider the data type

STRUCT OF

|           |    |
|-----------|----|
| INTEGER10 | x, |
| UNSIGNED5 | u  |

NewData

Assume x = - 423<sub>d</sub> = 259<sub>h</sub> and u = 30<sub>d</sub> = 1E<sub>h</sub>. Let b(x) and b(u) denote the bit sequences representing the values of x and u, respectively. Then:

|                   |   |                    |
|-------------------|---|--------------------|
| b(x)              | = b <sub>0</sub> (x) .. b <sub>9</sub> (x)    | = 1001101001       |
| b(u)              | = b <sub>0</sub> (u) .. b <sub>4</sub> (u)    | = 01111            |
| b(xu) = b(x) b(u) | = b <sub>0</sub> (xu) .. b <sub>14</sub> (xu) | = 1001101001 01111 |

The value of the structure is transferred with two octets, first 59h and then 7Ah.

## 6.1.6 Extended Data Types

The extended data types consist of the basic data types and the compound data types defined in the following subsections.

### 6.1.6.1 Octet String

The data type OCTET\_STRING $length$  is defined below;  $length$  is the maximum length of the octet string.

ARRAY [  $length$  ] OF UNSIGNED8      OCTET\_STRING $length$

If the actual length of the string is shorter than  $length$  the rest shall be filled up with 0<sub>h</sub>.

### 6.1.6.2 Visible String

The data type VISIBLE\_STRING $length$  is defined below. The data type is VISIBLE\_STRING whether a length is defined in the data type name or not. A VISIBLE\_STRING with a length definition defines the maximum length allowed. Otherwise any length is allowed.

The admissible values of data of type VISIBLE\_CHAR are 0<sub>h</sub> and the range from 20<sub>h</sub> to 7E<sub>h</sub>. The data are interpreted as ISO 646-1973(E) 7-bit coded characters.

UNSIGNED8      VISIBLE\_CHAR

ARRAY [  $length$  ] OF VISIBLE\_CHAR      VISIBLE\_STRING $length$

There is no 0<sub>h</sub> necessary to terminate the string. However, if the actual length of the string is shorter than  $length$  the rest shall be filled up with 0<sub>h</sub>.

### 6.1.6.3 Unicode String

The data type UNICODE\_STRING $length$  is defined below;  $length$  is the maximum length of the unicode string.

ARRAY [  $length$  ] OF UNSIGNED16      UNICODE\_STRING $length$

If the actual length of the string is shorter than  $length$  the rest shall be filled up with 0<sub>h</sub>.

### 6.1.6.4 Time of Day

The data type TIME\_OF\_DAY represents absolute time. It follows from the definition and the encoding rules that TIME\_OF\_DAY is represented as bit sequence of length 48.

Component ms is the time in milliseconds after midnight. Component days is the number of days since January 1, 1984.

STRUCT OF

|            |           |
|------------|-----------|
| UNSIGNED28 | ms,       |
| VOID4      | reserved, |
| UNSIGNED16 | days      |

TIME\_OF\_DAY

### 6.1.6.5 Time Difference

The data type TIME\_DIFFERENCE represents a time difference. It follows from the definition and the encoding rules that TIME\_DIFFERENCE is represented as bit sequence of length 48.

Time differences are sums of numbers of days and milliseconds. Component ms is the number milliseconds. Component days is the number of days.

STRUCT OF

|            |           |
|------------|-----------|
| UNSIGNED28 | ms,       |
| VOID4      | reserved, |
| UNSIGNED16 | days      |

TIME\_DIFFERENCE

### 6.1.6.6 Domain

Domains can be used to transfer an arbitrarily large block of data from a client to a server. The contents of a data block is application specific and does not fall within the scope of this document.

### 6.1.6.7 Net Time

The data type NETTIME represents a high precision time value. The NETTIME data type is identical to an improved version of the TimeRepresentation type of IEEE 1588.

NETTIME is composed as follows:

```
STRUCT OF
    UNSIGNED32 seconds
    UNSIGNED32 nanoseconds
END STRUCT
```

NETTIME

The nanoseconds member is defined such that the most significant bit represents the sign bit, 1 indicating a negative number, and the least significant 31 bits represent the nanoseconds portion of the time being represented. TimeRepresentation thus defines a sign magnitude representation for time stamps and time intervals.

The range of the absolute value of the least significant 31 bits of the nanoseconds portion of the representation shall be restricted to:

$$0 \leq | \text{least significant 31 bits of nanoseconds} | < 10^9$$

The sign of the nanoseconds member shall be interpreted as the sign of the entire representation. For example:

- +2.0 seconds is represented by seconds = 00000002<sub>h</sub> and nanoseconds = 00000000<sub>h</sub>
- -2.0 seconds is represented by seconds = 00000002<sub>h</sub> and nanoseconds = 80000000<sub>h</sub>
- +2.000000001 seconds is represented by seconds = 00000002<sub>h</sub> and nanoseconds = 00000001<sub>h</sub>
- -2.000000001 seconds is represented by seconds = 00000002<sub>h</sub> and nanoseconds = 80000001<sub>h</sub>

Timestamps shall be relative to 01-01-1970 00:00 h. A negative timestamp shall indicate time prior to this point of time. Timestamps using NETTIME will overflow in January 2106.

NETTIME shall be used for timestamps and time increments.

## 6.2 Object Dictionary

This section details the Object Dictionary structure and entries which are common to all devices.

The overall layout of the Object Dictionary is shown in 2.2.2.

### 6.2.1 Object Dictionary Entry Definition

An Object Dictionary entry shall be defined by the following items:

- **Index**

Index denotes the objects position within the Object Dictionary. This acts as a kind of address to reference the desired data field. Index shall be declared as hexadecimal value.

Index shall be used to indicate the accessed object of “by index”-type SDO commands.

Object may be subdivided to sub-indices. The sub-index is used to reference data fields within a complex object such as an array or record. The sub-index is not specified here.

- **Object Type**

Object Type contains an entry according to Tab. 37. It is used to denote what kind of object is at that particular index within the Object Dictionary. The following definitions are used:

| Object Type | Comments   | Code |
|-------------|--|------|
| NULL        | A dictionary entry with no data fields   | 0    |
| DEFTYPE     | Denotes a static data type definition such as a Boolean, UNSIGNED16, float and so on   | 5    |
| DEFSTRUCT   | Defines a record type  | 6    |
| VAR         | A single value such as an UNSIGNED8, Boolean, float, Integer16, visible string etc.  | 7    |
| ARRAY       | A multiple data field object where each data field is a simple variable of the same basic or extended data type e.g. array of UNSIGNED16 etc. Sub-index 0 is of UNSIGNED8 and therefore not part of the ARRAY data | 8    |
| RECORD      | A multiple data field object where the data fields may be any combination of simple variables. Sub-index 0 is of UNSIGNED8 and therefore not part of the RECORD data   | 9    |

Tab. 37 Object type definitions

- **Name**

Name provides a simple textual description of the function of that particular object.

Name shall be in accordance to IEC 61131-3. It consists of:

domain prefix, indicating the association of the object to a functional domain,  
3 uppercase characters followed by underline

name (verbally). composed of words, each word shall be leaded by an uppercase character  
followed by lowercase characters or digits, no underlines or spaces

data type postfix, indicating the data type of the object (underline followed by up to 5 uppercase  
characters or digits)

Total length of name shall be equal or below 32 characters.

Name shall be used to indicate the accessed object of “by name”-type SDO commands.

- **Data Type**

The entry provides information about the data type of the object. These include the following pre-defined static data types: Boolean, floating point number, unsigned integer, signed integer, visible/octet string, time-of-day, time-difference and DOMAIN (see 6.1). It also includes the pre-defined complex data types and may also include types which are either manufacturer or device specific.

It is not allowed to define records of records, arrays of records or records with arrays as fields of that record. In the case where an object is an array or a record the sub-index is used to reference one static data type data field within the object.

- **Category**

Category defines whether the object is Mandatory (M) or Optional (O). A mandatory object shall be implemented on a device. An optional object needs not to be implemented on a device. The support of certain objects or features however may require the implementation of related objects. In this case, the relations are described in the detailed object specification.

Category may be Conditional (Cond) if the M/O category of an object depends on the implementation of another object.

Category may be Not-Relevant (“-”) if the object is of no meaning for MN resp. CN.

The following entries shall be indicated for static data types only. In case of complex data types the respective entries shall be provided for each sub-index individually.

- **Access**

The entry defines the access rights for a particular object. The view point is from the bus into the device. It can be one of the following:

|       |  |
|-------|--|
| rw    | read and write access, value shall not be stored on writing dedicated sub-indices of NMT_StoreParam_REC            |
| rws   | read and write access, value shall be stored on writing dedicated sub-indices of NMT_StoreParam_REC                |
| wo    | write only access, value shall not be stored on writing dedicated sub-indices of NMT_StoreParam_REC                |
| wos   | write only access, value shall be stored on writing dedicated sub-indices of NMT_StoreParam_REC                    |
| ro    | read only access   |
| const | read only access, value is constant  |
| cond  | variable access controlled by the device<br>further information about access is provided by the object description |

Tab. 38 Access attributes for data objects

Optional objects listed in the Object Dictionary with the Attribute rw may be implemented as read only. Exceptions are defined in the detailed object specification.

Access type entries may be supplemented by additional information, e.g. reflecting data validity restrictions.

- **Value Range**

The entry indicates the value range of the respective object. It may consist of one or more distinct values or ranges of values. If the item shows a data type identifier, the complete value range of the mentioned data type shall be allowed.

"-" means that this specification does not predefine a value range. In this case the value range of the device profile resp. device description shall apply.

Default value ranges in the device description file shall be equal to predefined default value ranges in this specification.

- **Default Value**

The entry indicates the initialisation value that shall be assigned by the communication profile implementation.

"-" means that this specification does not predefine a default value. In this case the default value of the device profile resp. device description shall apply.

Default values in the device description file shall be equal to predefined default values in this specification.

- **PDO Mapping**

The entry indicates whether an entry may be mapped to a PDO message. It can be one of the following:

|     |  |
|-----|--|
| Opt | Object shall be mappable into a PDO                        |
| Def | Object is part of the default mapping (see device profile) |
| No  | Object shall not be mappable into a PDO                    |

Tab. 39 PDO mapping attributes for data objects

A complete static data type object definition (Object Type = VAR) example is displayed below:

|               |                           |             |     |
|---------------|---------------------------|-------------|-----|
| Index         | 1234 <sub>h</sub>         | Object Type | VAR |
| Name          | SDO_VarDummyParameter_S16 |             |     |
| Data Type     | INTEGER16                 | Category    | M   |
| Value Range   | -15 000 .. 10 000         | Access      | rw  |
| Default Value | 0                         | PDO Mapping | Opt |

Tab. 40 Static data type object definition example

Complex data type object definitions (Object Type = ARRAY or RECORD) are of reduced form, because Access, Value Range, Default Value, and PDO mapping must be defined individually for the the sub-indices

|           |                              |             |       |
|-----------|------------------------------|-------------|-------|
| Index     | 2345 <sub>h</sub>            | Object Type | ARRAY |
| Name      | SDO_ArrayDummyParameter_AU16 |             |       |
| Data Type | UNSIGNED16                   | Category    | M     |

Tab. 41 Complex data type object definition example

- **Category**

refers to the complex data type object as a whole, but not to the particular sub-index. A mandatory object may be composed of mandatory and optional sub-indices. This means that the object must be supported but some of its' sub-indices are optional. It's also allowed, to define optional object with mandatory sub-indices. This means that these sub-indices must be supported, if the object is implemented.

- **Data Type**

shall contain a static data type in case of ARRAY type objects and a complex data type in case of RECORD type objects.

The definition of data type describing objects (Object Type = DEFTYPE or DEFSTRUCT) is shown by 6.2.2.

### 6.2.1.1 Sub-Index Definition

Complex object types (ARRAY, RECORD) objects are composed of up to 256 data items. Each data item may be addressed by an UNSIGNED8 type sub-index.

Sub-Indices are used in the following way:

- **Sub-Index 00<sub>h</sub>      NumberOfEntries**

NumberOfEntries describes the highest available sub-index that follows, not considering FFh. This entry is encoded as UNSIGNED8, regardless the type of the object. If the object exists, NumberOfEntries is mandatory. Data Type and Category are not denoted at NumberOfEntries descriptions.

NumberOfEntries is described by this specification as displayed below:

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> | Access      | rw |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1..15           | Access      | rw |
| Default Value | 15              | PDO Mapping | No |

Tab. 42 NumberOfEntries sub-index description example

In case of ARRAYS, NumberOfEntries may be modified (Access = rw) to show the umber of occupied items in a list. In case of RECORDs, NumberOfEntries shall be hold constant (Access = ro or Const).

NumberOfEntries may mapped to PDO to indicate the number of occupied items.

- **Sub-Index 01<sub>h</sub> - FE<sub>h</sub>      Object Specific Data**

Sub-Indices between 01<sub>h</sub> and FE<sub>h</sub> hold the object's payload data. The highest accessible index is given by NumberOfEntries.

Sub-Indices of RECORD type objects are defined as follows:

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> | Access      | rw  |
| Name          | Recltem1_U8     |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | 1..255          | Access      | rw  |
| Default Value | 1               | PDO Mapping | Def |

Tab. 43 Record type object sub-index description example

- Name is composed of a describing text followed by a data type postfix. Refer the object name rules for further information, the domain prefix is omitted.  
If a name entry is defined by a data type definition, the sub-index name shall be equal to it.  
The sub-index name shall be used in combination to the object name to access the sub-index via “by name” SDO commands. According to IEC 61131-3 object and sub-index name shall be separated by a dot:  
SDO\_RecordExamle\_REC.Recltem1\_U8.
- Category refers to respective sub-index only. Mandatory (M) means that the sub-index must be implemented when the object is implemented. A mandatory sub-index does not force the hosting object to be mandatory.

Sub-Indices of ARRAY type objects are defined as follows:

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | ArraySubindex1  |             |     |
| --            | --              | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rw  |
| Default Value | 0               | PDO Mapping | Opt |

Tab. 44 Array type object sub-index description example

Name consists of a describing text. Refer the object name rules for further information, the domain prefix and data type postfix are omitted. If a name entry is defined by a data type definition, the sub-index name shall be equal to it.

- The sub-index name is for informational purpose only. Sub-index access via “by name” SDO commands shall occur according to IEC 61131-3 via numerical index:  
SDO\_ArrayExamle\_AU16[1]
- Category refers to respective sub-index only. Mandatory (M) means that the sub-index must be implemented when the object is implemented. A mandatory sub-index does not force the hosting object to be mandatory.
- Despite of the functional equivalence of all sub-indices of an array, there may be defined more than one sub-index entry to a particular object to show differences of the Category, Access and / or PDO Mappings options of the sub-indices.

- **Sub-Index FF<sub>h</sub>      StructureOfObject**

Sub-index FF<sub>h</sub> describes the structure of the entry by providing the data type and the object type of the entry. Regardless the type of the object, it is encoded as UNSIGNED32 and organised as follows:

|         | MSB     |   |             | LSB                             |
|---------|---------|---|-------------|---------------------------------|
| Bit-No. | 31 - 24 | 23 - 16                                 | 15 - 8      | 7 - 0                           |
| Value   | 00h     | Data Type (refer 6.2.2)<br>(High Byte□) | (Low Byte□) | Object Type<br>(refer Tab. 37)□ |

Tab. 45 Structure of sub-index FF<sub>h</sub>

It is optional to support sub-index FF<sub>h</sub>. If it is supported throughout the Object Dictionary and the structure of the complex data types is provided as well, it enables to upload the entire structure of the Object Dictionary. If StructureOfEntry is not supported, sub-index FF<sub>h</sub> shall be reserved.

StructureOfObject is not shown by object definitions of this specification.

## 6.2.2 Data Type Entry Specification

The static data types are placed in the Object Dictionary for definition purposes only. However, indices in the range 0001h - 0007h may be mapped in order to define the appropriate space of the RPDO (Dummy Mapping) as not being used by the device (do not care). The indices 0009h - 000Bh, 000Fh may not be mapped to a PDO.

See App. 1.1, index range 0001<sub>h</sub> to 04FF<sub>h</sub> for data type specifying object dictionary entries.

### 6.2.2.1 Static Data Types

The static data type (Object Type = DEFTYPE) representations used are detailed in 6.1.

A device may optionally provide the length of the static data types encoded as UNSIGNED32 at read access to the index that refers to the data type. E.g. index 000C<sub>h</sub> (Time of Day) contains the value 30<sub>h</sub>=48<sub>d</sub> as the data type „Time of Day“ is encoded using a bit sequence of 48 bit. If the length is variable (e.g. 000F<sub>h</sub> = Domain), the entry contains 0<sub>h</sub>.

### 6.2.2.2 Complex Data Types

The predefined complex data types' (Object Type = DEFSTRUCT) representations are provided by the respective paragraph or by the device profile.

For the supported complex data types a device may optionally provide the structure of that data type at read access to the corresponding data type index. Sub-index 0 then provides the number of entries at this index not counting sub-indices 0 and 255 and the following sub-indices contain the data type according to App. 3.5 encoded as UNSIGNED16.

As an example the entry at Index 0023h describing the structure of the Identity object NMT\_IdentityObject\_REC looks as follows:

| Index           | 0023 <sub>h</sub> | Object Type       | DEFSTRUCT  |
|-----------------|-------------------|-------------------|------------|
| Name            | IDENTITY          |                   |            |
| Sub-Index       | Component Name    | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries   | 04 <sub>h</sub>   |            |
| 01 <sub>h</sub> | VendorId_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub> | ProductCode_U32   | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub> | RevisionNo_U32    | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub> | SerialNo_U32      | 0007 <sub>h</sub> | UNSIGNED32 |

Tab. 46 Complex data type description example

Standard (simple) and complex manufacturer specific data types can be distinguished by attempting to read sub-index 1h: At a complex data type the device returns a value and sub-index 0h contains the number of sub-indices that follow, at a standard data type the device aborts the SDO transfer as no sub-index 1h available.

Note that some entries of data type UNSIGNED32 may have the character of a structure (e.g. NMT\_DeviceType\_U32, see 7.2.1.1.1).

### 6.2.2.3 Extension for Multiple Device Modules

For devices or device profiles that provide Multiple Device Modules like multiple axis controllers e.g. the DEFTYPE / DEFSTRUCT mechanism is enhanced for each virtual device with an offset of 40<sub>h</sub> for up to 7 additional virtual devices.

## 6.3 Service Data (SDO)

To access the entries of the object dictionary of a device via Ethernet POWERLINK a set of command services is specified.

SDO communication attends to the client / server model (see 2.3.2). Support of SDO client functions shall be indicated by D\_SDO\_Client\_BOOL, support of SDO server functions by D\_SDO\_Server\_BOOL.

### 6.3.1 SDO Layer Model

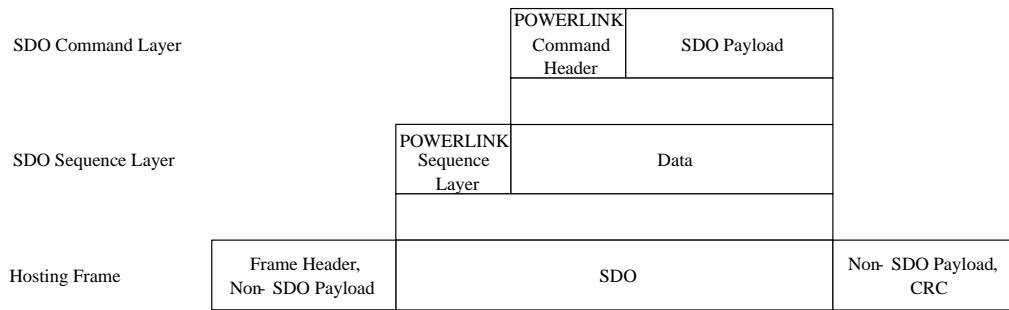


Fig. 42. SDO layer model

Two sublayers are distinguished in the POWERLINK Protocol:

- POWERLINK Sequence Layer (6.3.2.3, 6.3.3.1)  
The Sequence Layer is responsible for sorting the segments of a segmented transfer command so that a correct byte stream is offered to the POWERLINK Command Layer.
- POWERLINK Command Layer (6.3.2.4, 6.3.3.2)  
The Command Layer defines commands to access the parameters of the object dictionary. This layer distinguishes between an expedited and a segmented transfer.

### 6.3.1.1 SDO Hosting in Frames

Ethernet POWERLINK provides three SDO transfer methods:

1. SDO transfer via UDP/IP frames in asynchronous phase
2. SDO transfer via POWERLINK ASnd frames in asynchronous phase
3. SDO embedded in PDO in isochronous phase

The methods 1 (6.3.2.1) and 2 (6.3.2.2) share a common Sequence (6.3.2.3) and Command Layer (6.3.2.4).

At method 3 (6.3.3), SDO data are packed to a compact container to be inserted to the PDO data. Sequence and Command Layer are adapted to the compact layout of the container.

On an MN, support of methods 1 and 2 shall be mandatory.

Support of method 1 shall be indicated at the object dictionary by NMT\_FeatureFlags\_U32 Bit 1 and in the device description by D\_SDO\_SupportUdplp\_BOOL. Support of method 2 shall be indicated by NMT\_FeatureFlags\_U32 Bit 2 and D\_SDO\_SupportASnd\_BOOL.

Support of method 3 is optional at MN and CN. Support of method 3 shall be indicated at the object dictionary by NMT\_FeatureFlags\_U32 Bit 3 and in the device description by D\_SDO\_SupportPDO\_BOOL.

*Remark: The NMT\_FeatureFlags\_U32 are reported to the MN by IdentResponse.*

## 6.3.2 SDO in Asynchronous Phase

### 6.3.2.1 SDO via UDP/IP

The parameter transfer is based on a UDP/IP frame, allowing data transfer via a standard IP-router. Because UDP does not support a reliable connection oriented data transfer, this task must be supported by the sequence and command services.

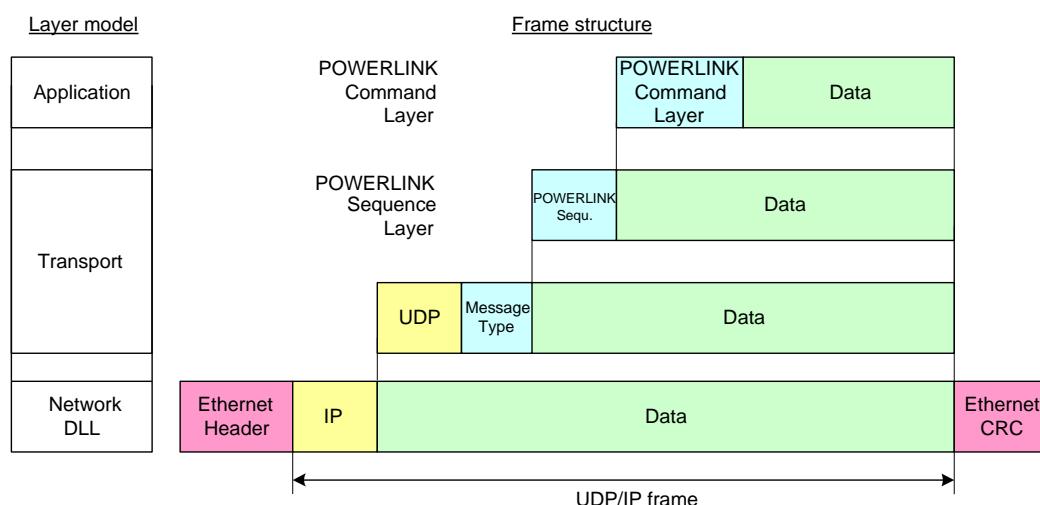


Fig. 43. POWERLINK SDO embedded in UDP/IP frame

SDO via UDP/IP uses the Asynchronous Sequence Layer (6.3.2.1) and the Asynchronous Command Layer (6.3.2.2).

For applications that do not require short cycle times and low jitter, POWERLINK telegrams may be inserted in a UDP/IP datagram, in effect running POWERLINK over UDP/IP. For this reason the message type (defined in the Data Link Layer) is inserted in front of the Sequence Layer.

| Byte Offset | Bit Offset              |   |   |   |   |   |   |   |
|-------------|-------------------------|---|---|---|---|---|---|---|
|             | 7                       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0           | MessageType = ASnd      |   |   |   |   |   |   |   |
| 1 .. 2      | reserved                |   |   |   |   |   |   |   |
| 3           | ServiceID = SDO         |   |   |   |   |   |   |   |
| 4 .. 7      | Sequence Layer Protocol |   |   |   |   |   |   |   |
| 8 .. k-1    | Command Layer Protocol  |   |   |   |   |   |   |   |
| k .. 1471   | SDO Payload Data        |   |   |   |   |   |   |   |

Tab. 47 SDO via UDP/IP

| Field                   | Abbr. | Description   | Value   |
|-------------------------|-------|---|---|
| MessageType             | mt    | POWERLINK Message type (App. 3.1)   | ASnd  |
| reserved                | res   | These fields are reserved<br>They are used for POWERLINK Destination and Source Address when sending SDO without UDP/IP | Zero, when embedded in UDP/IP Frame<br>else mt specific |
| ServiceID               | sid   | ASnd ServiceID (App. 3.3)   | SDO   |
| Sequence Layer Protocol |       | POWERLINK Sequence Layer (6.3.2.1)  |   |
| Command Layer Protocol  |       | POWERLINK Command Layer (6.3.2.2)   |   |

Tab. 48 SDO via UDP/IP field interpretation

### 6.3.2.1.1 UDP Layer

The UDP Header contains:

| Field            | Size   | Description                                     | Used in POWERLINK             |
|------------------|--------|---|-------------------------------|
| Source Port      | 2 Byte | Port Number of the sending process              | Logical channel <sup>15</sup> |
| Destination Port | 2 Byte | Port Number of the receiving process            |                               |
| Length           | 2 Byte | Data length of the whole UDP frame incl. header | not used                      |
| Checksum         | 2 Byte | optional Checksum                               | not used                      |

Tab. 49 UDP header

Parameters are transferred in a communication channel characterized by the IP addresses (source and destination) and UDP Ports (source and destination). This communication channel is known as a *datagram socket*. It establishes a peer-to-peer communication channel between two devices. A device may support more than one channel. One supported server channel is the default case (Default Channel). The default channel uses the UDP port C\_SDO\_EPL\_PORT.

The maximum number of supported channels shall be indicated by D\_SDO\_MaxConnections\_U32.

There may be multiple channels established between one client and one server. The maximum number of such parallel channels shall be indicated by D\_SDO\_MaxParallelConnections\_U32.

The client starts the transfer using the standard destination port C\_SDO\_EPL\_PORT and a client dependent source port (> 1024). The server responds to the request with the source port C\_SDO\_EPL\_PORT and the destination port defined by the client. Therefore up to ( $2^{16}$ –1024) logical channels (sockets) can be opened between a client and a server. Each device is responsible for handling its logical channels.

<sup>15</sup> in combination with the client and server IP address.

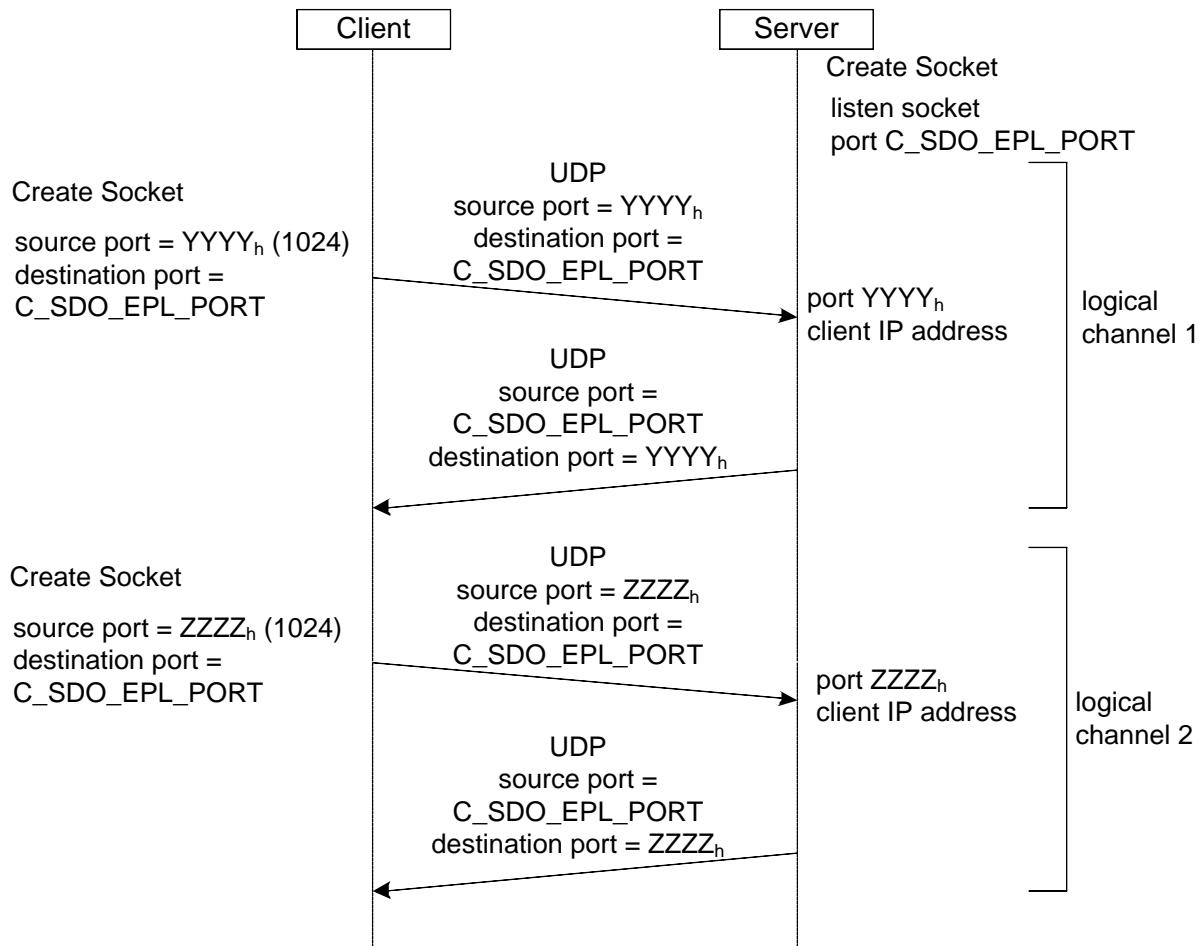


Fig. 44. UDP socket

### 6.3.2.2 SDO via POWERLINK ASnd

SDO via POWERLINK ASnd applies POWERLINK frames for SDO communication.

Since POWERLINK frames are not forwarded by a POWERLINK Router Type 1 (s. 9.1), it is not possible to access nodes exclusively providing SDO via POWERLINK ASnd from outside of the POWERLINK segment.

SDO transfer by POWERLINK ASnd frames is well suited for devices providing reduced resources, because it does not require an IP stack.

| Byte Offset | Bit Offset              |   |   |   |   |   |   |   |
|-------------|-------------------------|---|---|---|---|---|---|---|
|             | 7                       | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0           | MessageType = ASnd      |   |   |   |   |   |   |   |
| 1           | Destination             |   |   |   |   |   |   |   |
| 2           | Source                  |   |   |   |   |   |   |   |
| 3           | ServiceID = SDO         |   |   |   |   |   |   |   |
| 4 .. 7      | Sequence Layer Protocol |   |   |   |   |   |   |   |
| 8 .. k-1    | Command Layer Protocol  |   |   |   |   |   |   |   |
| k .. 1471   | SDO Payload Data        |   |   |   |   |   |   |   |

Tab. 50 SDO via POWERLINK ASnd

| Field                   | Abbr. | Description                                | Value |
|-------------------------|-------|--|-------|
| MessageType             | mt    | POWERLINK Message type (App. 3.1)          | ASnd  |
| Destination             | dest  | POWERLINK Node ID of the addressed node(s) |       |
| Source                  | src   | POWERLINK Node ID of the transmitting node |       |
| ServiceID               | sid   | ASnd ServiceID (App. 3.3)                  | SDO   |
| Sequence Layer Protocol |       | POWERLINK Sequence Layer (6.3.2.3)         |       |
| Command Layer Protocol  |       | POWERLINK Command Layer (6.3.2.4)          |       |

Tab. 51 SDO via ASnd field interpretation

SDO via POWERLINK ASnd applies the Asynchronous Sequence Layer (6.3.2.3) and the Asynchronous Command Layer (6.3.2.4).

### 6.3.2.3 Asynchronous SDO Sequence Layer

The POWERLINK Sequence Layer provides the service of a reliable bidirectional connection that guarantees that no messages are lost or duplicated and that all messages arrive in the correct order. Fragmentation is handled by the SDO Command Layer (6.3.2.4).

The POWERLINK Sequence Layer Header for asynchronous transfer shall consist of 2 bytes.

There shall be a sequence number for each sent frame, and an acknowledgement for the sequence number of the opposite node, as well a connection state and a connection acknowledge.

| Octet Offset | Bit Offset            |   |   |   |   |   |            |   |
|--------------|-----------------------|---|---|---|---|---|------------|---|
|              | 7                     | 6 | 5 | 4 | 3 | 2 | 1          | 0 |
| 0            | ReceiveSequenceNumber |   |   |   |   |   | ReceiveCon |   |
| 1            | SendSequenceNumber    |   |   |   |   |   | SendCon    |   |
| 2 .. 3       | reserved              |   |   |   |   |   |            |   |

Tab. 52 POWERLINK sequence layer in asynchronous data frame

*Remark:*

*Using bits 0 and 1 for connection instead of bits 6 and 7 eases the handling of sequence number. The sequence number easily can be increased by one by increasing the whole byte by four. This increment has no influence to ReceiveCon and SendCon.*

| Field                 | Abbr. | Description  | Value  |
|-----------------------|-------|--|--|
| ReceiveSequenceNumber | rsnr  | Sequence number of the last correctly received frame                           | 0 ... 63   |
| ReceiveCon            | rcon  | Acknowledge of connection code to the opposite node                            | 0: No connection<br>1: Initialisation<br>2: Connection valid<br>3: Error Response (retransmission request)   |
| SendSequenceNumber    | ssnr  | Own sequence number of the frame, shall be increased by 1 with every new frame | 0 ... 63   |
| SendCon               | scon  | Own connection code  | 0: No connection<br>1: Initialisation<br>2: Connection valid<br>3: Connection valid with acknowledge request |

Tab. 53 Fields of POWERLINK sequence layer in asynchronous data frame

### 6.3.2.3.1 Connection

#### 6.3.2.3.1.1 Initialisation of Connection

The client shall request initialisation by setting scon to 1. This shall be responded by the server with the same connection code and the same sequence number.

If the server had already a connection to the client the existing connection shall be shut down, i.e. closed internally without sending an explicit close request before opening the new one.

Additionally the sequence number shall be initialised at the server.

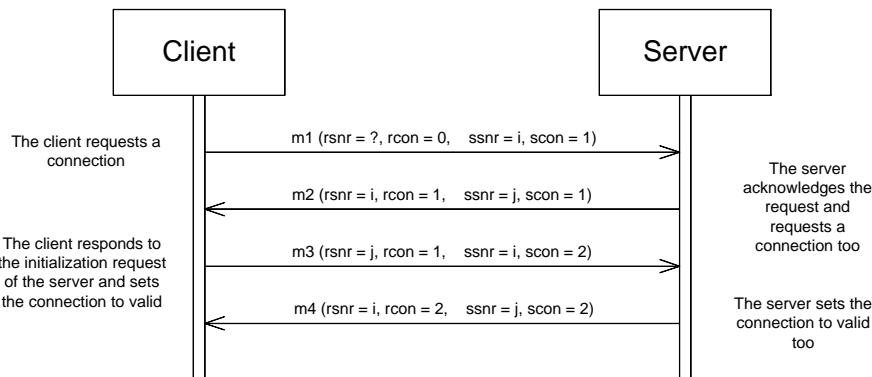


Fig. 45. Initialisation of an asynchronous connection

After this the bidirectional connection is established.

The sequence numbers shall not be incremented until the bidirectional connection initialisation has been completed. It shall be increased by 1 for the first SDO command after the initialisation. No command shall be sent while initialising the bidirectional connection.

#### 6.3.2.3.1.2 Closing a connection

A connection should be shut down, when it is no longer needed.

A node, i.e. client or server, may shut down a connection, if it needs the resources for other reasons.

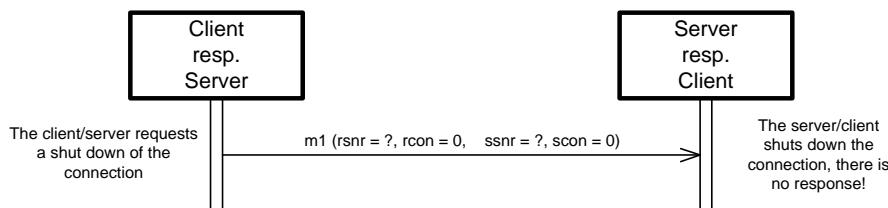


Fig. 46. Closing of asynchronous connection

Closing a connection shall be indicated by rcon = 0 and scon = 0.

There shall be no acknowledge for closing a connection.

#### 6.3.2.3.1.3 Data Transfer

When the connection is established each node is allowed to send frames.

Each node shall keep the sent data in a history buffer until it is acknowledged. Size of the history buffer is indicated by D\_SDO\_SeqLayerTxHistorySize\_U16. If all history buffer slots are occupied by unacknowledged frames, no additional frame may be sent (see 6.3.2.3.1.5).

Each sent frame shall contain the acknowledgement of the last correctly received frame from the opposite side.

When the sender sends an acknowledge request, the receiver shall send an acknowledge.

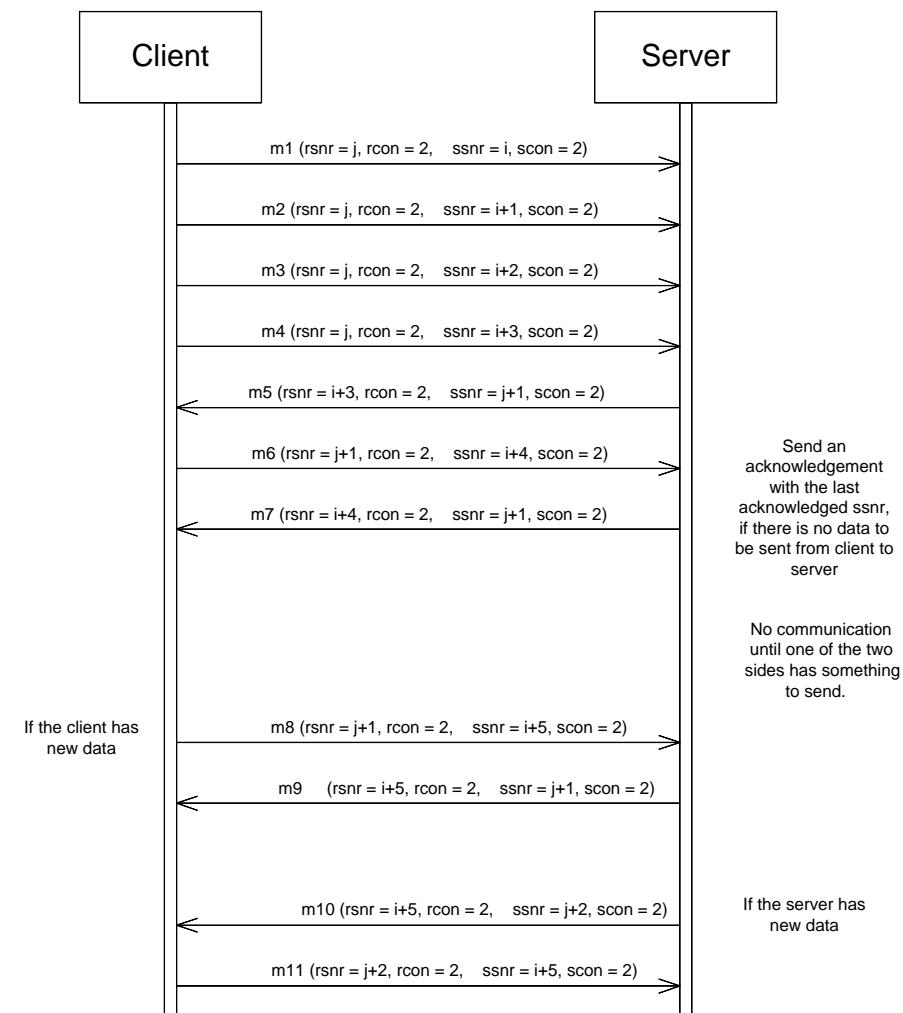


Fig. 47. Normal asynchronous communication

If the receiver has no new data to transmit it shall send an acknowledge frame to the sender, that contains the last sent ssnr. If that ssnr was already acknowledged, the data may be omitted, because the other side will drop the data (as repeated) anyway.

Because of the sliding window protocol only half of the sequence number range may be used. So the maximum possible History Buffer size is 31. Otherwise it is not possible to distinguish between old duplicated frames and new frames.

On the receiving node no buffering of the received frames is required. This may cause flooding of the receiving node with commands (cf. 6.3.2.3.2.6).

#### 6.3.2.3.1.4 Data Transfer with Delay

Due to delays in network, hardware and software layers it may take some cycles until the frame is received by the receiver, and the acknowledge gets back to the sender.

The sender shall not forward more than D\_SDO\_SeqLayerTxHistorySize\_U16 frames before receiving an acknowledgement.

This example shows a configuration where the frames are delayed. A typical situation where frames are delayed is when POWERLINK Networks are connected by means of routers over a legacy ethernet.

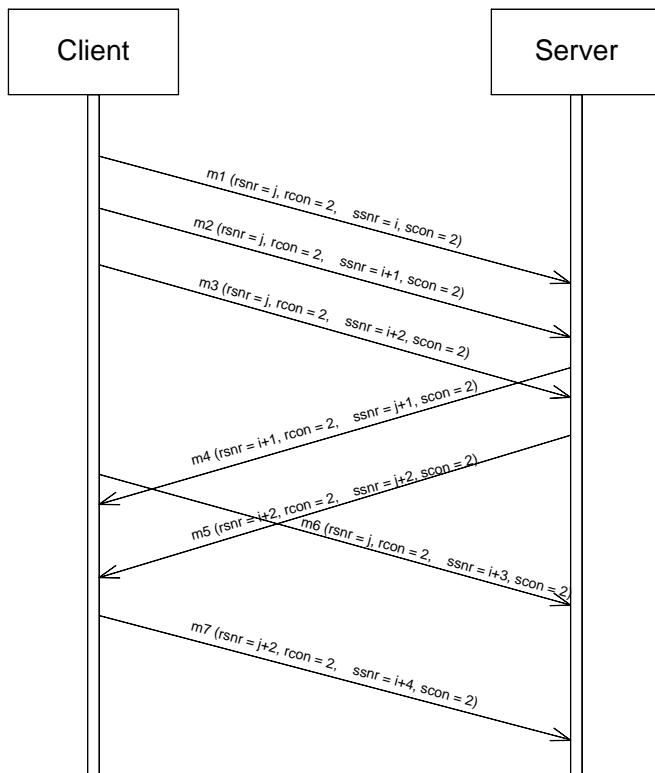


Fig. 48. Delayed asynchronous communication

### 6.3.2.3.1.5 Sender History Full

When the buffer for keeping frames is full (sliding window size exhausted), the sending node may explicitly request an acknowledgement by sending a frame with acknowledge request.

The receiver shall acknowledge this frame with an empty acknowledgement frame, if it has no frames of its own to be sent.

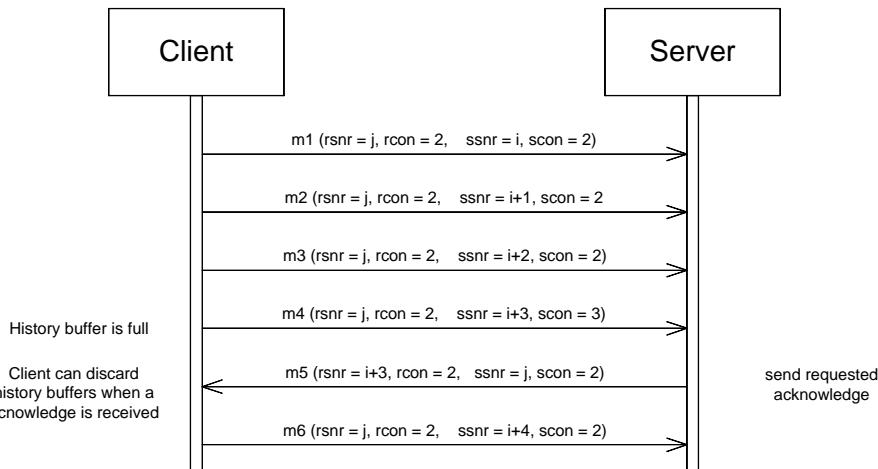


Fig. 49. Asynchronous communication when history buffer gets full

### 6.3.2.3.2 Errors

Errors that may occur in the physical and data link layer:

- Loss of frames
- Duplication of frames
- Overtaking of frames
- Broken connection

Errors that may occur in the sequence layer:

- Flooding with commands

### 6.3.2.3.2.1 Error: Loss of Frame with Data

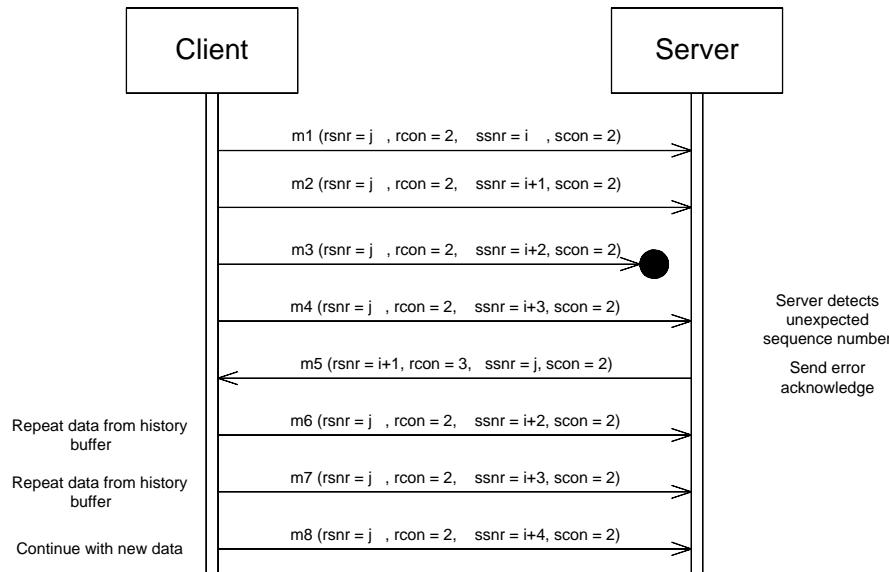


Fig. 50. Error loss of asynchronous frame

If the receiver detects a sequence number that is more than 1 higher than the last correctly received sequence number, it shall respond with rcon=3 and the last correctly received sequence number to indicate this error.

This error response may contain data (from the command layer).

The sender shall repeat all the frames that followed the responded sequence number.

The acknowledge fields in the repeated frames shall be updated.

### 6.3.2.3.2.2 Error: Loss of Acknowledge Frame

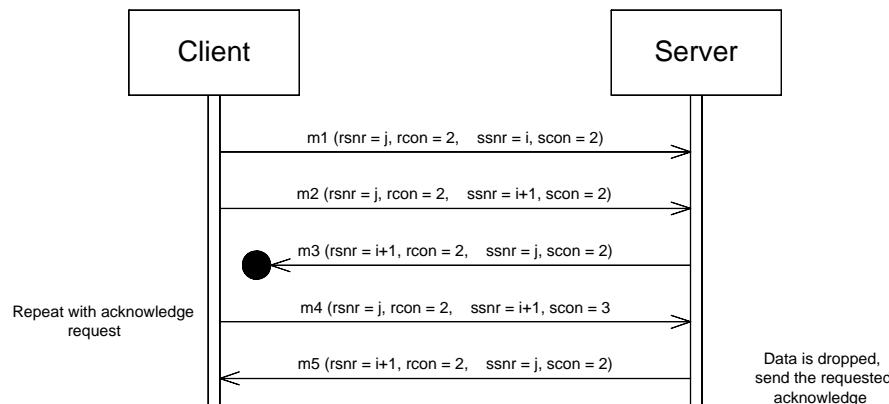


Fig. 51. Error loss of asynchronous acknowledge

If the sender is waiting for an acknowledgement and has no new frames to be sent, it shall repeat the latest message with acknowledgement request after a timeout.

### 6.3.2.3.2.3 Error: Duplication of Frame

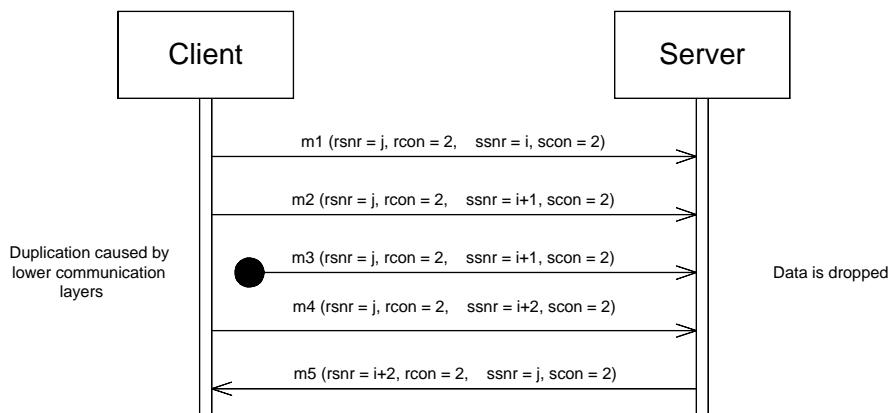


Fig. 52. Error duplication of asynchronous frame

If the receiver detects a sender sequence number that is lower than or equal to the last correctly received sequence, it shall drop that message.

If that message has scon=2 no further action is required.

If that message has scon=3 the receiver shall acknowledge the last correctly received sequence to the sender.

### 6.3.2.3.2.4 Error: Overtaking of Frames

When a frame overtakes, the receiver at first detects a lost frame. The receiver shall send an error response to the sender so that the sender repeats the frame.

Then the overtaken frame arrives at the receiver and is accepted.

After this the repeated frame arrives and is dropped at the receiver, because it is recognized as a duplicated frame.

Remark:

Overtaking of frames will never occur inside the POWERLINK network domain. Only on connections over Internet using Type I routers overtaking can occur on the Legacy Ethernet side.

### 6.3.2.3.2.5 Broken Connection

It shall be detected, that the connection is broken, if the opposite node is shut down or disconnected from the network.

The connection shall be considered broken when no acknowledgement is received within the timeout given by SDO\_SequLayerTimeout\_U32.

Within this timeout multiple acknowledgement requests shall be sent (see 6.3.2.3.2.2). The number of acknowledgement requests to be sent shall be described by SDO\_SequLayerNoAck\_U32. If SDO\_SequLayerNoAck\_U32 is not implemented a minimum value of C\_SDO\_SEQLAYERNOACK shall be used instead.

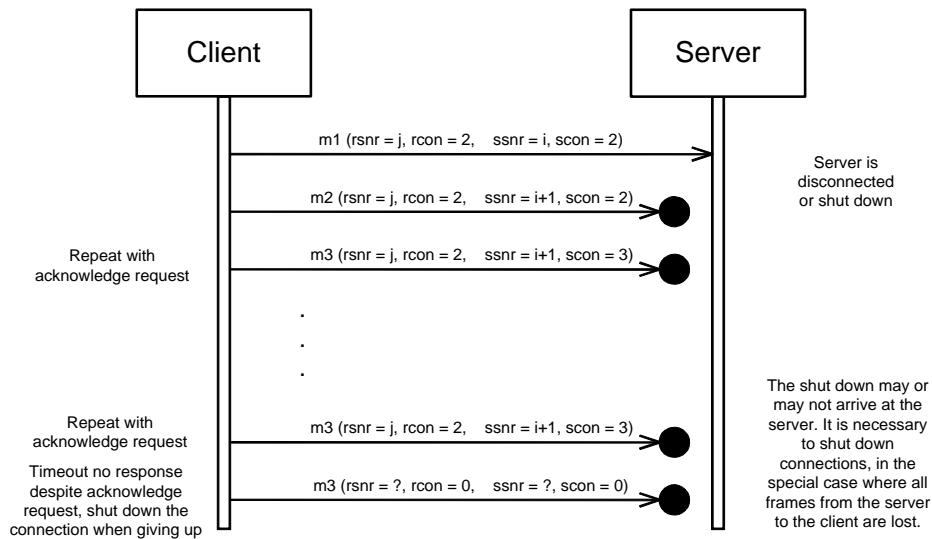


Fig. 53. Error asynchronous communication broken

### 6.3.2.3.2.6 Error: Flooding with commands

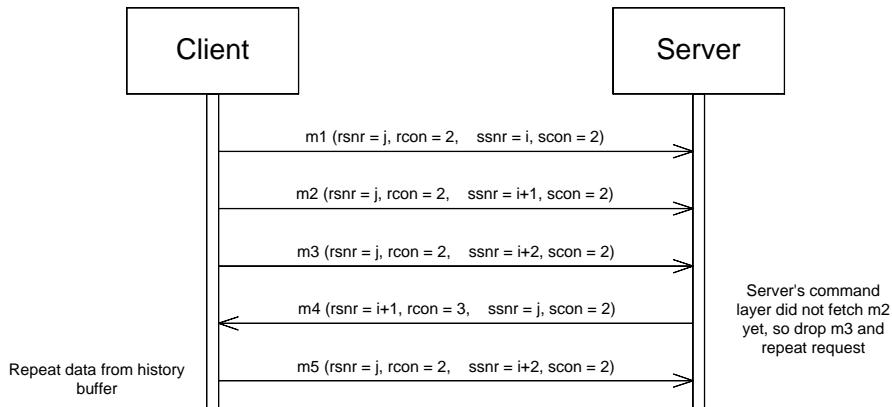


Fig. 54. Error flooding with asynchronous commands

If the sender sends commands at a rate too high, the command layer at the receiver may not be able to process the commands in time. In this case the sequence layer on the receiver side drops newly arriving frames and shall send an acknowledgement of the last correctly handled frame and a rcon=3 back to the sender.

This causes the sender to repeat the dropped frame, and the receiver gains some time to handle the request.

This shall not be misused as a flow control mechanism, flow control shall be done in higher layers.

### 6.3.2.4 Asynchronous SDO Command Layer

Tasks of the POWERLINK Command Layer

1. Addressing of the parameters, e.g. via index/sub-index or via name
2. Provide additional services
3. Distinguish between expedited and segmented transfer

The POWERLINK Command Layer is embedded in the POWERLINK Sequence Layer. If a large block is to be transferred the POWERLINK Command Layer has to decide whether the transfer can be completed in one frame (*expedited transfer*) or if it must be segmented in several frames (*segmented transfer*). Further it has to know whether an Upload or a Download should be initiated.

For all transfer types it is the client that takes the initiative for a transfer. The owner of the accessed object dictionary is the server of the Service Data Object (SDO). Either the client or the server can take the initiative to abort the transfer of a SDO. All commands are confirmed. The remote result

parameter indicates the success of the request. In case of a failure, an Abort Transfer Request must be executed.

Fig. 55 shows the structure of the information in the POWERLINK Command Layer Header.

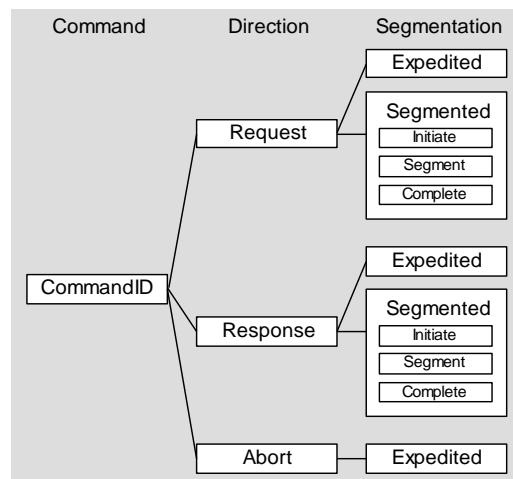


Fig. 55. Information structure of POWERLINK command layer

### 6.3.2.4.1 POWERLINK Command Layer Protocol

This chapter defines the fixed part of the POWERLINK Command Layer protocol.

The POWERLINK Command Layer is structured in the following way:

| Byte Offset <sup>16</sup> | Bit Offset                                  |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |
|---------------------------|---|-------|--------------|---|----------|---|---|---|--------------------------|--|--|--|--|--|--|
|                           | 7   | 6     | 5            | 4 | 3        | 2 | 1 | 0 |                          |  |  |  |  |  |  |
| 0                         | reserved                                    |       |              |   |          |   |   |   | Fixed part               |  |  |  |  |  |  |
| 1                         | Transaction ID                              |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |
| 2                         | Res-  | Abort | Segmentation |   | reserved |   |   |   |                          |  |  |  |  |  |  |
| 3                         | Command ID                                  |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |
| 4 .. 5                    | Segment Size                                |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |
| 6 .. 7                    | reserved                                    |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |
| 8 .. 11                   | Data Size (only if Segmentation = Initiate) |       |              |   |          |   |   |   | Variable part            |  |  |  |  |  |  |
| (8 + 4*d) .. k-1          | Command ID specific header                  |       |              |   |          |   |   |   | Command ID specific part |  |  |  |  |  |  |
| k .. 1463                 | Optional Payload Data                       |       |              |   |          |   |   |   |                          |  |  |  |  |  |  |

*k = Length of Command ID specific header; k < 1464*

*d: if seg = Initiate d = 1  
else d = 0*

Tab. 54 POWERLINK command layer

<sup>16</sup> Offset related to beginning of POWERLINK command layer.

| Field               | Abbr. | Description  | Value   |
|---------------------|-------|--|---|
| reserved            | res   | Reserved<br>This byte is used when embedding the SDO in cyclic data (chapter 6.3.3).   | 0   |
| Transaction ID      | tid   | Unambiguous transaction ID for a command.<br>Changed by the client with every new command.   | 0 – 255   |
| Response            | rsp   | Request / Response   | 0: Request<br>1: Response   |
| Abort               | a     | The requested Transfer could not be completed by the client/server   | 0: transfer ok<br>1: abort transfer   |
| Segmentation        | seg   | Differentiates between expedited and segmented transfer  | 0: Expedited Transfer<br>1: Initiate Segm. Transfer<br>2: Segment<br>3: Segm. Transfer Complete |
| Command ID          | cid   | Specifies the command  | see Tab. 58   |
| Segment Size        | ss    | Length of segment data.<br>Counting from the end of the command header (beginning with Byte Offset 8)  | 256 – 1456  |
| Data Size           | ds    | Contains the number of bytes of the transferred block.<br>Counting from the end of the command header (beginning with Byte Offset 8)<br>Only used for the Initiate Transfer Frame (seg = Initiate)<br>If ds = 0000h, the size is not indicated | 0 – $2^{32}$ -1   |
| Command ID specific |       | Specifies the command referenced by the cid.   | See 6.3.2.4.2   |

Tab. 55 POWERLINK command layer field interpretation

The Transaction ID may be used to support several logic channels parallel via the same UDP socket resp. ASnd channel.

The Segment Size (ss) indicates the length of the segment in the command layer, i.e. the valid data length in the command layer. A minimum size of 256 bytes must be supported by every device. A maximum of 1456 bytes (i.e. 1500 byte payload data for the Ethernet frame) may be supported. The client can use the command “Maximum Segment Size” (see 6.3.2.4.2.4.1) to get the maximum usable size for a communication to a server.

*Note: The maximum segment size is limited to 1456 bytes, because an Ethernet frame can carry a maximum of 1500 bytes. In case of SDO via UDP all the remaining bytes are needed for the protocol overhead (IP (20 bytes), UDP (8 bytes), message type / service ID / etc. (4 bytes), POWERLINK Sequence Layer (4 bytes) and the fixed part of the POWERLINK Command Layer (8 bytes)). In case of SDO via ASnd the maximum possible length of 1500 bytes is not fully used because the protocol overhead is shorter (message type / service ID / etc. (4 bytes), POWERLINK Sequence Layer (4 bytes) and the fixed part of the POWERLINK Command Layer (8 bytes)). However this guarantees an SDO command layer independent of the underlaying transfer method (UDP/IP or ASnd).*

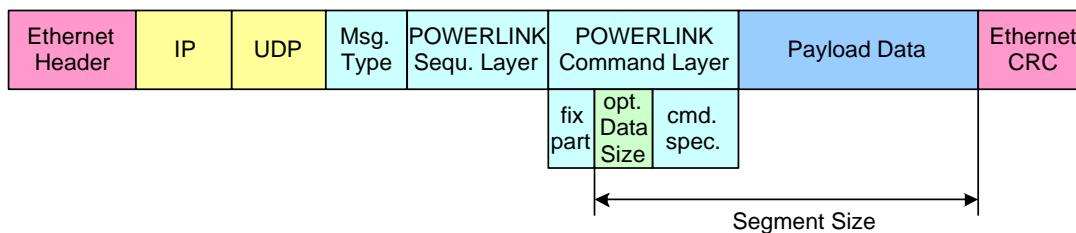


Fig. 56. Definition of segment size

For Initiate Segmented Transfer (seg=1) the number of bytes to be transferred in the command is indicated in the Data Size field. Therefore the offset for the Command ID specific header is 8 (Expedited Transfer) or 12 (Initiate Segmented Transfer) (cf. Tab. 54).

### 6.3.2.4.1.1 Download Protocol

The download service is used by the SDO client to download data to the server (owner of the object dictionary).

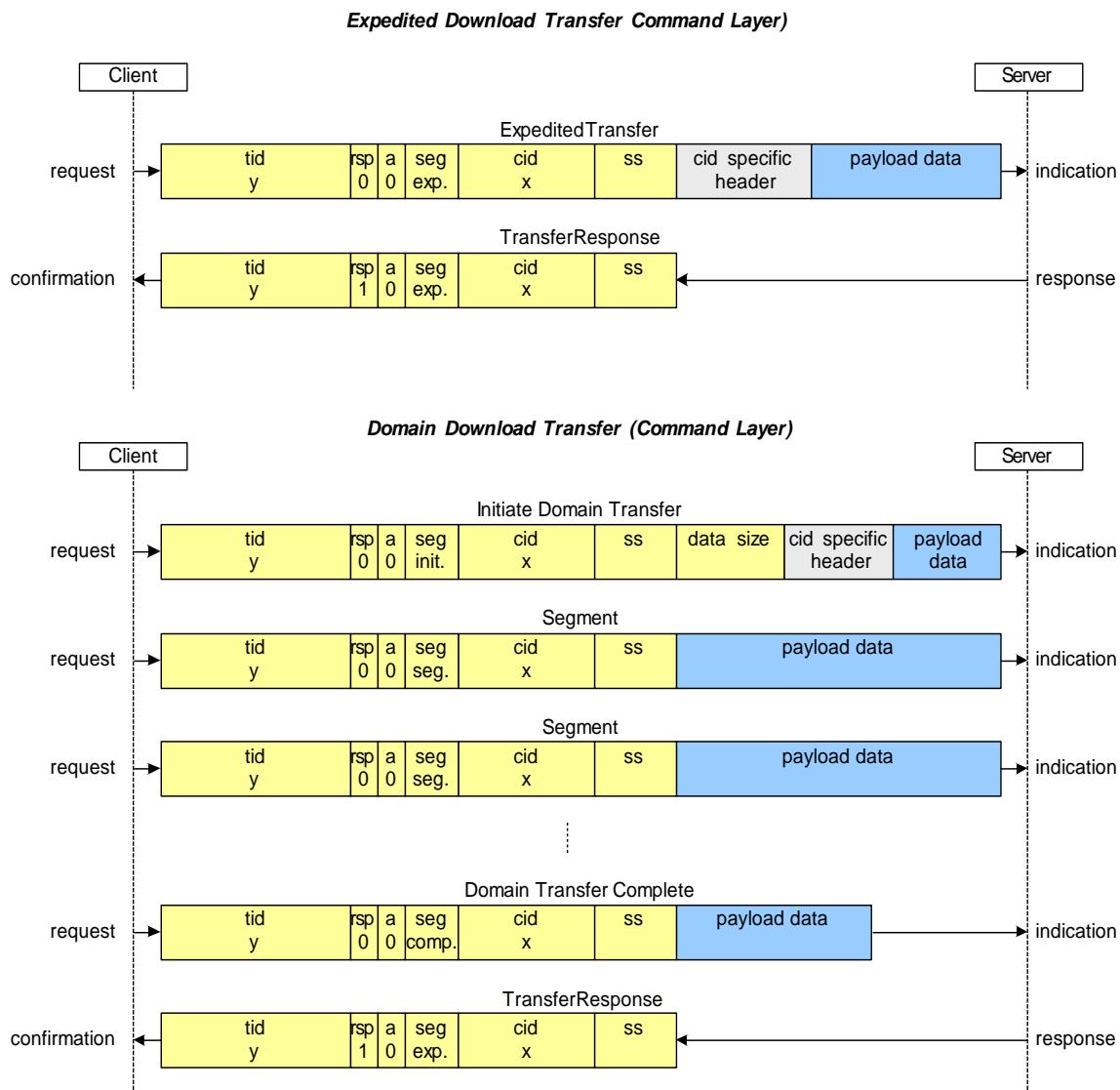


Fig. 57. POWERLINK command layer: Typical download transfer

In an expedited download, the data identified by the cid specific header is transferred to the server.

In a segmented transfer SDOs are downloaded as a sequence of zero or more Download SDO Segment services preceded by an Initiate SDO Download service and followed by a Transfer Complete frame. The sequence is terminated by:

- A response/confirm, indicating the successful completion of a normal download sequence.
- An Abort SDO Transfer request/indication, indicating the unsuccessful completion of the download sequence.

The SDO Sequence Layer is not shown in Fig. 57. There may be more frames involved in the initialisation of the SDO Sequence Layer, see chapter 6.3.2.3 for details.

### 6.3.2.4.1.2 Upload Protocol

The Upload service is used by the SDO client to upload data from the server.

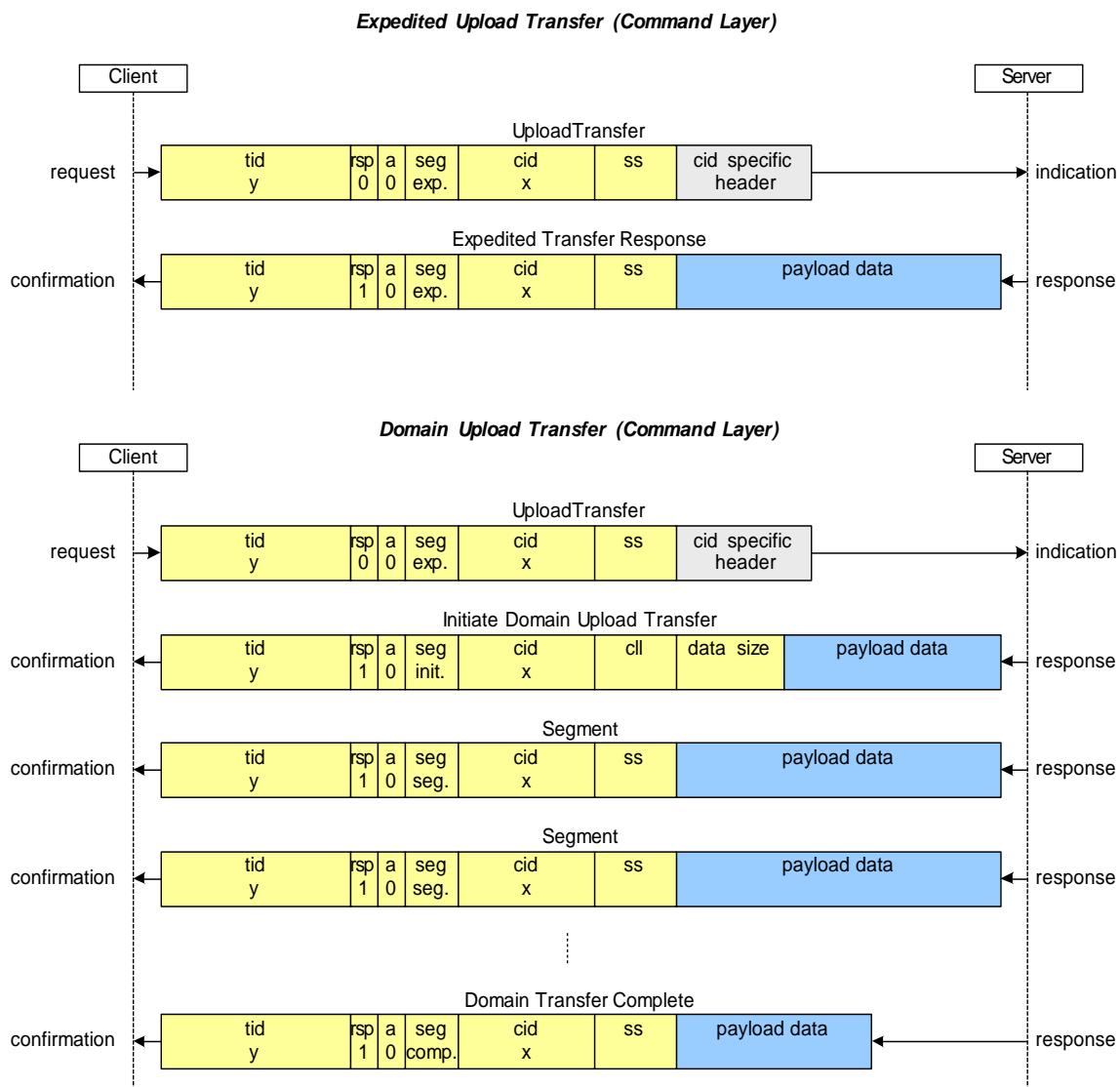


Fig. 58. POWERLINK command layer: Typical upload transfer

If an expedited upload is successful, the service concludes the upload of the data set identified by the cid specific header and the corresponding data is confirmed.

In a segmented transfer, SDOs are uploaded as a sequence of zero or more Upload SDO Segment services preceded by an Initiate SDO Upload service and followed by a Transfer Complete frame. The sequence is terminated by:

- The Transfer Complete frame, indicating the successful completion of a normal upload sequence.
- An Abort SDO Transfer request/indication, indicating the unsuccessful completion of the upload sequence.

### 6.3.2.4.1.3 Abort Transfer

The Abort Transfer service aborts the up- or download referenced by the Transaction ID. The reason is indicated.

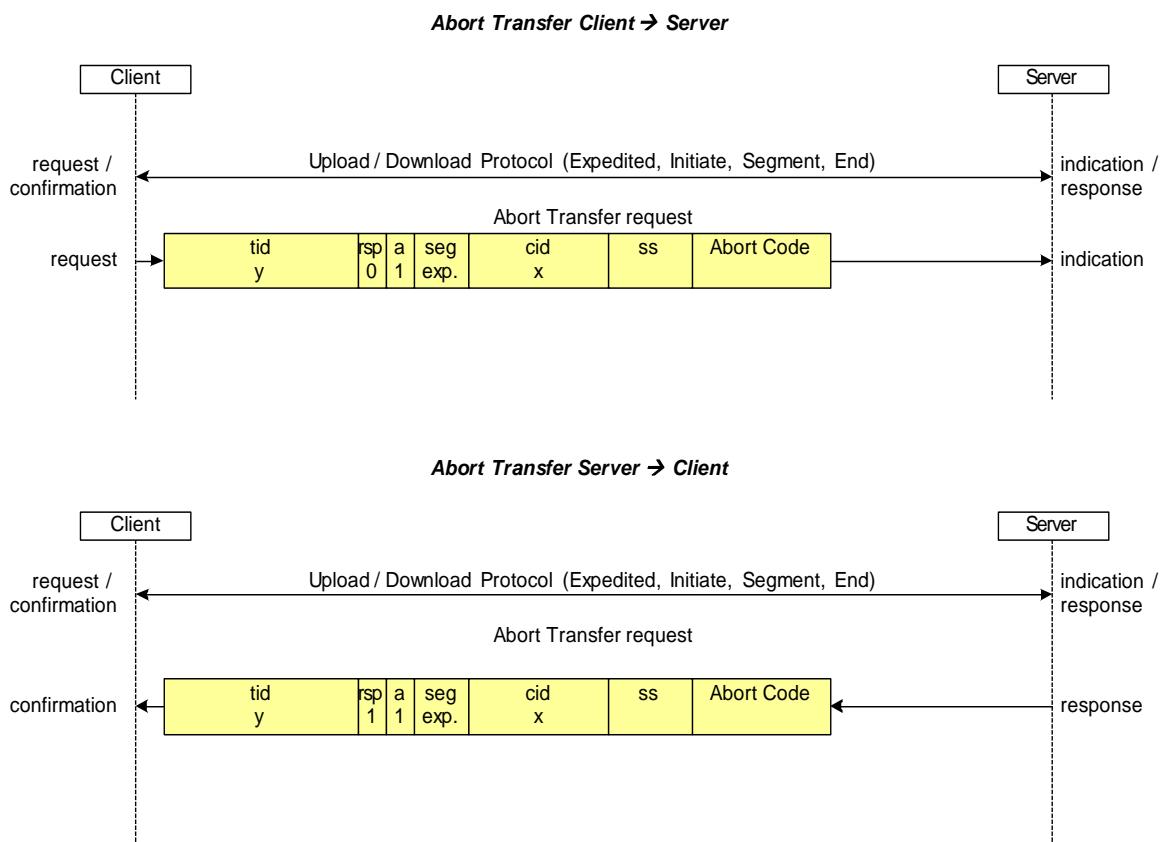


Fig. 59. Abort transfer

The Abort service is unconfirmed. The service may be executed at any time by either the client or the server of a SDO. If the client of a SDO has a confirmed service outstanding, the indication of the abort is taken to be the confirmation of that service.

| Byte Offset | Bit Offset     |       |   |              |          |   |   |   |  |  |  |  |
|-------------|----------------|-------|---|--------------|----------|---|---|---|--|--|--|--|
|             | 7              | 6     | 5 | 4            | 3        | 2 | 1 | 0 |  |  |  |  |
| 0           | reserved       |       |   |              |          |   |   |   |  |  |  |  |
| 1           | Transaction ID |       |   |              |          |   |   |   |  |  |  |  |
| 2           | Res-           | Abort | 1 | Segmentation | reserved |   |   |   |  |  |  |  |
| 3           | Command ID     |       |   |              |          |   |   |   |  |  |  |  |
| 4 .. 5      | Segment Size   |       |   |              |          |   |   |   |  |  |  |  |
| 6 .. 7      | reserved       |       |   |              |          |   |   |   |  |  |  |  |
| 8 .. 11     | Abort Code     |       |   |              |          |   |   |   |  |  |  |  |

Tab. 56 Abort transfer frame

| Field      | Abbr. | Description         | Value         |
|------------|-------|---------------------|---------------|
| Abort Code | ac    | Reason of the abort | see App. 3.10 |

Tab. 57 Abort transfer frame field interpretation

The abort code is encoded as UNSIGNED32. A list of SDO abort codes is provided by App. 3.10.

### 6.3.2.4.2 Commands

This chapter describes the Command ID specific part of the POWERLINK Command Layer.

| Command                           | Command Description // Device Description Entry   | Cmd ID                            | M/O                |
|-----------------------------------|---|-----------------------------------|--------------------|
| NIL <sup>17</sup>                 | Not in List   | 0 <sub>h</sub>                    | M                  |
| <b>SDO Protocol</b>               |   |                                   |                    |
| Write by Index                    | Write a parameter, addressing via index/sub-index   | 1 <sub>h</sub>                    | M                  |
| Read by Index                     | Read a parameter, addressing via index/sub-index  | 2 <sub>h</sub>                    | M                  |
| Write All by Index                | Write a parameter, addressing via index, all sub-indices // D_SDO_CmdWriteAllByIndex_BOOL                   | 3 <sub>h</sub>                    | O                  |
| Read All by Index                 | Read a parameter, addressing via index, all sub-indices // D_SDO_CmdReadAllByIndex_BOOL                     | 4 <sub>h</sub>                    | O                  |
| Write by Name                     | Write a parameter, addressing via name // D_SDO_CmdWriteByName_BOOL   | 5 <sub>h</sub>                    | O                  |
| Read by Name                      | Read a parameter, addressing via name // D_SDO_CmdReadByName_BOOL   | 6 <sub>h</sub>                    | O                  |
| <b>File Transfer</b>              |   |                                   |                    |
| File Write                        | Simple file transfer // D_SDO_CmdFileWrite_BOOL   | 20 <sub>h</sub>                   | O                  |
| File Read                         | Simple file transfer // D_SDO_CmdFileRead_BOOL  | 21 <sub>h</sub>                   | O                  |
| <b>Variable groups</b>            |   |                                   |                    |
| Write Multiple Parameter by Index | Write multiple parameter within one command, addressing via index/sub-index // D_SDO_CmdWriteMultParam_BOOL | 31 <sub>h</sub>                   | O                  |
| Read Multiple Parameter by Index  | Read multiple parameter within one command, addressing via index/sub-index // D_SDO_CmdReadMultParam_BOOL   | 32 <sub>h</sub>                   | O                  |
| <b>Parameter Services</b>         |   |                                   |                    |
| Maximum Segment Size              | Exchange the maximum segment size   | 70 <sub>h</sub>                   | Cond <sup>18</sup> |
| Manufacturer specific             |   | 80 <sub>h</sub> – FF <sub>h</sub> | O                  |

Tab. 58 Command services and command ID

Support of particular commands is optional. Support shall be indicated by device description entries (c.f. Tab. 58).

#### 6.3.2.4.2.1 SDO Protocol

*Note: The Download Protocol (6.3.2.4.1.1, including Fig. 57) is used for write commands and the Upload Protocol (6.3.2.4.1.2, including Fig. 58) is used for read commands.*

##### 6.3.2.4.2.1.1 Command: Write by Index

Using the Write by Index service the client of a SDO downloads data to the server (owner of the object dictionary). The data, the multiplexor (index and sub-index) of the data set that has been downloaded and its size (only for segmented transfer) is indicated to the server.

<sup>17</sup> The NIL command shall be ignored on the command layer but processed at the sequence layer, that is the sequence layer shall send an acknowledge response but the NIL command will otherwise be ignored.

<sup>18</sup> Conditional: Only necessary if a segment size > 256 Byte should be transferred

| Bit Offset         |   |   |   |   |   |   |   |   |
|--------------------|---|---|---|---|---|---|---|---|
| Byte Offset        | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7             | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11            | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 9) + 4*d     | Index                                       |   |   |   |   |   |   |   |
| 10 + 4*d           | Sub-Index                                   |   |   |   |   |   |   |   |
| 11 + 4*d           | reserved                                    |   |   |   |   |   |   |   |
| (12 + 4*d) .. 1463 | Payload Data                                |   |   |   |   |   |   |   |

*d: if seg = Initiate d = 1  
else d = 0*

Tab. 59 Command: Write by Index request

| Field     | Abbr. | Description   | Value      |
|-----------|-------|---|------------|
| Index     | i     | Specifies an entry of the device object dictionary        | 0 .. 65535 |
| Sub-Index | si    | Specifies a component of a device object dictionary entry | 0 .. 254   |

Tab. 60 Write by Index request field interpretation

### 6.3.2.4.2.1.2 Command: Read by Index

Using the Read by Index service the client of a SDO requests that the server upload data to the client. The multiplexor (index and sub-index) of the data set whose upload is initiated is indicated to the server.

| Bit Offset  |                            |   |   |   |   |   |   |   |
|-------------|----------------------------|---|---|---|---|---|---|---|
| Byte Offset | 7                          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7      | Command Layer (fixed part) |   |   |   |   |   |   |   |
| 8 .. 9      | Index                      |   |   |   |   |   |   |   |
| 10          | Sub-Index                  |   |   |   |   |   |   |   |
| 11          | reserved                   |   |   |   |   |   |   |   |

Tab. 61 Command: Read by Index Request

| Field     | Abbr. | Description   | Value      |
|-----------|-------|---|------------|
| Index     | i     | Specifies an entry of the device object dictionary        | 0 .. 65535 |
| Sub-Index | si    | Specifies a component of a device object dictionary entry | 0 .. 254   |

Tab. 62 Read by Index request field interpretation

### 6.3.2.4.2.1.3 Command: Write All by Index

Using the Write All by Index service the client of a SDO downloads data to the server (owner of the object dictionary). The service addresses all sub-indices (except sub-index 0) of the indicated index. The length of the payload data must confirm to the length of data for all sub-indices and all sub-indices must be writable.

| Bit Offset         |   |   |   |   |   |   |   |   |
|--------------------|---|---|---|---|---|---|---|---|
| Byte Offset        | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7             | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11            | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 9) + 4*d     | Index                                       |   |   |   |   |   |   |   |
| 10 + 4*d           | reserved                                    |   |   |   |   |   |   |   |
| 11 + 4*d           | reserved                                    |   |   |   |   |   |   |   |
| (12 + 4*d) .. 1452 | Payload Data                                |   |   |   |   |   |   |   |

*d: if seg = Initiate d = 1  
else d = 0*

Tab. 63 Command: Write All by Index request

| Field | Abbr. | Description  | Value      |
|-------|-------|--|------------|
| Index | i     | Specifies an entry of the device object dictionary | 0 .. 65535 |

Tab. 64 Write All by Index request field interpretation

#### 6.3.2.4.2.1.4 Command: Read All by Index

Using the Read All by Index service the client of a SDO requests that the server upload data to the client. The service addresses all sub-indices (except sub-index 0) of the indicated index. All sub-indices must be readable.

| Byte Offset | Bit Offset                 |   |   |   |   |   |   |   |
|-------------|----------------------------|---|---|---|---|---|---|---|
|             | 7                          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7      | Command Layer (fixed part) |   |   |   |   |   |   |   |
| 8 .. 9      | Index                      |   |   |   |   |   |   |   |
| 10 .. 11    | reserved                   |   |   |   |   |   |   |   |

Tab. 65 Command: Read All by Index Request

| Field | Abbr. | Description  | Value      |
|-------|-------|--|------------|
| Index | i     | Specifies an entry of the device object dictionary | 0 .. 65535 |

Tab. 66 Read All by Index request field interpretation

#### 6.3.2.4.2.1.5 Command: Write by Name

Using the Write by Name service the client of a SDO downloads data to the server. The data, the name of the data set that has been downloaded and its size (only for segmented transfer) are indicated to the server.

| Byte Offset           | Bit Offset                                  |   |   |   |   |   |   |   |
|-----------------------|---|---|---|---|---|---|---|---|
|                       | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7                | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11               | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 9) + 4*d        | Offset Payload Data (k)                     |   |   |   |   |   |   |   |
| (10 + 4*d) .. (k - 1) | Name  |   |   |   |   |   |   |   |
| k .. 1463             | Payload Data                                |   |   |   |   |   |   |   |

*k = Offset of the name length; k < 1464, 4-aligned*

*d: if seg = Initiate d = 1  
else d = 0*

Tab. 67 Command: Write by Name request

| Field               | Abbr. | Description  | Value     |
|---------------------|-------|--|-----------|
| Offset Payload Data | opd   | Specifies the beginning of the payload data (in bytes) in this segment, counting from end of the fixed command header (beginning with Byte Offset 8) | 0 .. 1463 |
| Name                | n     | Specifies an entry of the device object dictionary   |           |

Tab. 68 Write by Name request field interpretation

- The name may not be terminated by a \0.
- The definitions made in IEC 61131-3 for identifiers are adapted to the name conventions<sup>19</sup>. These are:
  - A name is a sequence of characters, dots, digits and underlines, beginning with a character or an underline.
  - Underlines shall be significant in identifiers, e.g. A\_BC and AB\_C are different identifiers.
  - Multiple leading or embedded underlines are not allowed.
  - Identifiers shall not contain embedded space (SP) characters.

<sup>19</sup> The IEC 61131-3 defines several keywords utilized as individual syntax elements. These keywords shall not be used as a parameter name. POWERLINK does not define further keywords.

- At least six unique characters shall be supported in all systems.
  - The payload data shall be 4-byte-aligned. Therefore the name may have to be padded.
- The Write by Name service is defined to access application objects (e.g. global variables) that do not have an index/sub-index.

### 6.3.2.4.2.1.6 Command: Read by Name

Using the Read by Name service the client of a SDO requests that the server upload data to the client. The name of the data set whose upload is initiated is indicated to the server.

| Byte Offset    | Bit Offset                                  |   |   |   |   |   |   |   |
|----------------|---|---|---|---|---|---|---|---|
|                | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 – 7          | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 – 11         | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 + 4*d) .. k | Name  |   |   |   |   |   |   |   |

$k < 1464$

$d: \text{if seg} = \text{Initiate} \quad d = 1$   
 $\text{else} \quad d = 0$

Tab. 69 Command: Read by Name request

| Field | Abbr. | Description  | Value             |
|-------|-------|--|-------------------|
| Name  | N     | Specifies an entry of the device object dictionary | see 6.3.2.4.2.1.5 |

Tab. 70 Read by Name request field interpretation

The payload data shall be 4-byte-aligned. Therefore the name may have to be padded.

The Read by Name service is defined to access application objects (e.g. global variables) that do not have an index/sub-index.

### 6.3.2.4.2.2 File Transfer

A simple File transfer protocol is defined.

For file access, in addition to the naming conventions (see 6.3.2.4.2.1.5) the valid character set is extended with the characters:

- “/” and “\”
- “\*”
- “.”

A file open/close service is not defined because this would cause related services with different Command IDs.

### 6.3.2.4.2.2.1 Command: File Write

The File Write protocol combines several typical operation system commands for file access:

- File open
- File seek
- File write

| Byte Offset           | Bit Offset                                  |   |   |   |   |   |   |   |
|-----------------------|---|---|---|---|---|---|---|---|
|                       | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7                | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11               | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 11) + 4*d       | Address                                     |   |   |   |   |   |   |   |
| (12 .. 13) + 4*d      | Offset Payload Data (k)                     |   |   |   |   |   |   |   |
| (14 .. 15) + 4*d      | reserved                                    |   |   |   |   |   |   |   |
| (16 + 4*d) .. (k – 1) | File Name                                   |   |   |   |   |   |   |   |
| k .. 1463             | Payload Data                                |   |   |   |   |   |   |   |

$k = \text{Offset of the payload data}; k < 1464, 4\text{-aligned}$

$d: \text{if seg} = \text{Initiate} \quad d = 1$   
 $\text{else} \quad d = 0$

Tab. 71 Command: File Write request

| Field               | Abbr. | Description  | Value            |
|---------------------|-------|--|------------------|
| Address             | addr  | Address of the data from the beginning of the file.  | 0 .. $2^{32}$ -1 |
| Offset Payload Data | opd   | Specifies the beginning of the payload data (in bytes) in this segment, counting from the end of the fixed command header (beginning with Byte Offset 8) | 0 .. 1463        |
| File Name           | fn    | File name (complete path)  |                  |

Tab. 72 File Write request field interpretation

The Address field indicates the relative address of the data in the file.

### 6.3.2.4.2.2.2 Command: File Read

| Byte Offset     | Bit Offset                                  |   |   |   |   |   |   |   |
|-----------------|---|---|---|---|---|---|---|---|
|                 | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7          | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11         | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 11) + 4*d | Address                                     |   |   |   |   |   |   |   |
| (12 + 4*d) .. k | File Name                                   |   |   |   |   |   |   |   |

$k < 1464$

d: if seg = Initiate d = 1  
else d = 0

Tab. 73 Command: File Read request

| Field     | Abbr. | Description   | Value            |
|-----------|-------|---|------------------|
| Address   | addr  | Address of the data from the beginning of the file. | 0 .. $2^{32}$ -1 |
| File Name | fn    | File name (complete path)                           |                  |

Tab. 74 File Read request field interpretation

### 6.3.2.4.2.3 Variable groups

#### 6.3.2.4.2.3.1 Command: Write Multiple Parameter by Index

Using the Write Multiple Parameter by Index service the client of a SDO downloads multiple data sets to the server. The data, the multiplexor (index and sub-index) of the data sets that are downloaded and the size of the transfer (only for segmented transfer) are indicated to the server.

A device shall be able to process this command up to a size which can be transferred in a maximum sized Ethernet frame.

### 6.3.2.4.2.3.2 Write Multiple Parameter by Index Request

|                       | Bit Offset                                  |   |   |   |                |   |   |   |  |  |  |  |
|-----------------------|---|---|---|---|----------------|---|---|---|--|--|--|--|
| Byte Offset           | 7   | 6 | 5 | 4 | 3              | 2 | 1 | 0 |  |  |  |  |
| 0 .. 7                | Command Layer (fixed part)                  |   |   |   |                |   |   |   |  |  |  |  |
| 8 .. 11               | Data Size (only if Segmentation = Initiate) |   |   |   |                |   |   |   |  |  |  |  |
| (8 .. 11) + 4*d       | Byte Offset of next Data Set (k)            |   |   |   |                |   |   |   |  |  |  |  |
| (12 .. 13) + 4*d      | Index                                       |   |   |   |                |   |   |   |  |  |  |  |
| 14 + 4*d              | Sub-Index                                   |   |   |   |                |   |   |   |  |  |  |  |
| 15 + 4*d              | reserved                                    |   |   |   | Padding Length |   |   |   |  |  |  |  |
| (16 + 4*d) .. (k - 1) | Payload Data                                |   |   |   |                |   |   |   |  |  |  |  |
| k .. (k + 3)          | Byte Offset of next Data Set (m)            |   |   |   |                |   |   |   |  |  |  |  |
| (k + 4) .. (k + 5)    | Index                                       |   |   |   |                |   |   |   |  |  |  |  |
| k + 6                 | Sub-Index                                   |   |   |   |                |   |   |   |  |  |  |  |
| k + 7                 | reserved                                    |   |   |   | Padding Length |   |   |   |  |  |  |  |
| (k + 8) .. (m - 1)    | Payload Data                                |   |   |   |                |   |   |   |  |  |  |  |
| m ...                 | ... (further write requests)                |   |   |   |                |   |   |   |  |  |  |  |

d: if seg = Initiate d = 1  
else d = 0  
k, m 4-aligned

Tab. 75 Command: Write Multiple Parameter by Index Request

| Field                        | Abbr. | Description   | Value                   |
|------------------------------|-------|---|-------------------------|
| Byte Offset of next Data Set | o2d   | Byte Offset of the next data set. The value is the absolute offset, counting from the beginning of the fixed command header (beginning with byte offset 0)<br>If o2d=ZERO the last data set has been reached. | 0 .. 2 <sup>32</sup> -1 |
| Index                        | i     | Specifies an entry of the device object dictionary  | 0 .. 65535              |
| Sub-Index                    | si    | Specifies a component of a device object dictionary entry   | 0 .. 254                |
| Padding Length               | pl    | Number of padding bytes in the last quadlet (4-byte word) of the payload data   | 0 .. 3                  |
| reserved                     | res   | Reserved for future use.  | 0                       |

Tab. 76 Write Multiple Parameter by Index request field interpretation

### 6.3.2.4.2.3.3 Write Multiple Parameter by Index Response

|                  | Bit Offset                                  |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
|------------------|---|----------|---|---|---|---|---|---|--|--|--|--|--|--|--|
| Byte Offset      | 7   | 6        | 5 | 4 | 3 | 2 | 1 | 0 |  |  |  |  |  |  |  |
| 0 .. 7           | Command Layer (fixed part)                  |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 8 .. 11          | Data Size (only if Segmentation = Initiate) |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| (8 .. 9) + 4*d   | Index                                       |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 10 + 4*d         | Sub-Index                                   |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 11 + 4*d         | SubAbort                                    | reserved |   |   |   |   |   |   |  |  |  |  |  |  |  |
| (12 .. 15) + 4*d | Sub-Abort Code                              |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| (16 .. 17) + 4*d | Index                                       |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 18 + 4*d         | Sub-Index                                   |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 19 + 4*d         | SubAbort                                    | reserved |   |   |   |   |   |   |  |  |  |  |  |  |  |
| (20 .. 23) + 4*d | Sub-Abort Code                              |          |   |   |   |   |   |   |  |  |  |  |  |  |  |
| 24 ...           | ... (further write responses)               |          |   |   |   |   |   |   |  |  |  |  |  |  |  |

d: if seg = Initiate d = 1  
else d = 0

Tab. 77 Command: Write Multiple Parameter by Index Response

| Field          | Abbr. | Description   | Value                               |
|----------------|-------|---|-------------------------------------|
| Index          | i     | Specifies an entry of the device object dictionary          | 0 .. 65.535                         |
| Sub-Index      | si    | Specifies a component of a device object dictionary entry   | 0 .. 254                            |
| Sub-Abort      | a     | The requested transfer could not be processed by the server | 0: transfer ok<br>1: abort transfer |
| reserved       | res   | Reserved for future use                                     | 0                                   |
| Sub-Abort Code | sac   | Reason of the sub-abort                                     | see App. 3.10                       |

Tab. 78 Write Multiple Parameter by Index response field interpretation

In the response a list of all invalid object accesses is transferred.

The list entries consist of the index and sub-index of the object, a sub-abort flag (sa) and the sub-abort code (sac).

The abort flag (a) in the command header is set only if the data contain one single abort code instead of a response list. E.g. "Command ID not valid or unknown".

If all accesses are valid and processed by the server the command specific header is empty, i.e. the abort flag (a) is not set and the list of faulty accesses is empty.

#### 6.3.2.4.2.3.4 Command: Read Multiple Parameter by Index

Using the Read multiple parameter service the client of a SDO requests that the server for upload multiple data sets to the client. The multiplexor (index and sub-index) of the data sets whose upload is initiated is indicated to the server.

A device shall be able to process this command up to a size which can be transferred in a maximum sized Ethernet frame.

#### 6.3.2.4.2.3.5 Read Multiple Parameter by Index Request

| Byte Offset      | Bit Offset                                  |   |   |   |   |   |   |   |
|------------------|---|---|---|---|---|---|---|---|
|                  | 7   | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7           | Command Layer (fixed part)                  |   |   |   |   |   |   |   |
| 8 .. 11          | Data Size (only if Segmentation = Initiate) |   |   |   |   |   |   |   |
| (8 .. 9) + 4*d   | Index                                       |   |   |   |   |   |   |   |
| 10 + 4*d         | Sub-Index                                   |   |   |   |   |   |   |   |
| 11 + 4*d         | reserved                                    |   |   |   |   |   |   |   |
| (12 .. 13) + 4*d | Index                                       |   |   |   |   |   |   |   |
| 14+4*d           | Sub-Index                                   |   |   |   |   |   |   |   |
| 15+4*d           | reserved                                    |   |   |   |   |   |   |   |
| 16 ...           | ... (further read requests)                 |   |   |   |   |   |   |   |

d: if seg = Initiate d = 1  
else d = 0

Tab. 79 Command: Read Multiple Parameter by Index request

| Field     | Abbr. | Description   | Value      |
|-----------|-------|---|------------|
| Index     | i     | Specifies an entry of the device object dictionary        | 0 .. 65535 |
| Sub-Index | si    | Specifies a component of a device object dictionary entry | 0 .. 254   |
| reserved  | res   | Reserved for alignment                                    | 0          |

Tab. 80 Read Multiple Parameter by Index request field interpretation

### 6.3.2.4.2.3.6 Read Multiple Parameter by Index Response

| Byte Offset           | Bit Offset                                  |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
|-----------------------|---|----------|---|----------------|---|---|---|---|--|--|--|--|--|--|--|
|                       | 7   | 6        | 5 | 4              | 3 | 2 | 1 | 0 |  |  |  |  |  |  |  |
| 0 .. 7                | Command Layer (fixed part)                  |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| 8 .. 11               | Data Size (only if Segmentation = Initiate) |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| (8 .. 11) + 4*d       | Byte Offset of next Data Set (k)            |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| (12 .. 13) + 4*d      | Index                                       |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| 14 + 4*d              | Sub-Index                                   |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| 15 + 4*d              | SubAbort                                    | reserved |   | Padding Length |   |   |   |   |  |  |  |  |  |  |  |
| (16 + 4*d) .. (k - 1) | Payload Data / Sub-Abort Code               |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| k .. (k + 3)          | Byte Offset of next Data Set (m)            |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| (k + 4) .. (k + 5)    | Index                                       |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| k + 6                 | Sub-Index                                   |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| k + 7                 | SubAbort                                    | reserved |   | Padding Length |   |   |   |   |  |  |  |  |  |  |  |
| (k + 8) .. (m - 1)    | Payload Data / Sub-Abort Code               |          |   |                |   |   |   |   |  |  |  |  |  |  |  |
| m ...                 | ... (further read responses)                |          |   |                |   |   |   |   |  |  |  |  |  |  |  |

d: if seg = Initiate    d = 1  
     else                d = 0  
     k, m 4-aligned

Tab. 81 Command: Read Multiple Parameter by Index response

| Field                        | Abbr. | Description   | Value                               |
|------------------------------|-------|---|-------------------------------------|
| Byte Offset of next Data Set | o2d   | Byte offset of the next data set. The value is the absolute offset, counting from the beginning of the fixed command header (beginning with byte offset 0)<br>If o2d=ZERO the last data set has been reached. | 0 – $2^{32}$ -1                     |
| Index                        | i     | Specifies an entry of the device object dictionary  | 0 – 65.535                          |
| Sub-Index                    | si    | Specifies a component of a device object dictionary entry   | 0 – 254                             |
| Sub-Abort                    | sa    | The requested transfer could not be served by the server  | 0: transfer ok<br>1: abort transfer |
| Padding Length               | pl    | Number of padding bytes in the last quadlet (4-byte word) of the payload data   | 0 – 3                               |
| reserved                     | res   | Reserved for future use   | 0                                   |
| Sub-Abort Code               | sac   | Reason of the abort   | see App. 3.10                       |

Tab. 82 Read Multiple Parameter by Index response field interpretation

In the response a list of all object accesses is transferred.

The list entries consist of the index and sub-index of the object, a sub-abort flag (sa) and the payload data resp. the sub-abort code (sac).

The abort flag (a) in the command header is set only if the data contain one single abort code instead of a response list. E.g. "Command ID not valid or unknown".

### 6.3.2.4.2.4 Parameter Services

#### 6.3.2.4.2.4.1 Command: Maximum Segment Size

The maximum segment size (MSS) indicates the maximum length of a segment in the command layer.

The minimum segment size that must be supported by every device is 256 bytes. If the client and the server can handle more than 256 bytes the client can use this command to negotiate the maximum segment size.

| Byte Offset | Bit Offset                 |   |   |   |   |   |   |   |
|-------------|----------------------------|---|---|---|---|---|---|---|
|             | 7                          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 7      | Command Layer (fixed part) |   |   |   |   |   |   |   |
| 8 .. 9      | MSS Client                 |   |   |   |   |   |   |   |
| 10 .. 11    | MSS Server                 |   |   |   |   |   |   |   |

Tab. 83 Command: Maximum Segment Size

| Field      | Abbr. | Description   | Value      |
|------------|-------|---|------------|
| MSS Client | mssc  | MSS of the client   | 256 – 1456 |
| MSS Server | msss  | MSS of the server<br>If 0000 <sub>h</sub> the length is not indicated (request from client to server) | 256 – 1456 |

Tab. 84 Maximum Segment Size field interpretation

The maximum segment size is limited to 1456 independent of the transfer method (UDP/IP or ASnd). In the request frame from the client the *mssc* is indicated to the server. The *msss* is set to ZERO and therefore not indicated.

In the response the server repeats the *mssc* of the client and indicates its own *msss*.

Both, client and server, must compare the *mssc* to the indicated *msss* and must calculate the minimum of both. This is the used MSS.

$$\text{Used MSS} = \min \{\text{mssc}; \text{msss}\}$$

### 6.3.3 SDO Embedded in PDO

It is possible to embed the SDO in the cyclic PDO. The embedded SDO is used as a container mapped into the PDO.

The Read/Write by Index command layer protocol is used to access the data. The Header of the container starts with a shortened Sequence Layer (1 Byte). The fixed part of the following POWERLINK Command Layer protocol is adopted in the following points:

- as the container has a fixed length, the valid data length has to be indicated. Therefore the field “valid payload length” is inserted. Up to 255 bytes of payload data may be transferred in a container.

| Byte Offset                   | Bit Offset                                  |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |
|-------------------------------|---|-------|--------------|----------|---|---|---|---|----------------|--|--|--|--|--|--|--|
|                               | 7   | 6     | 5            | 4        | 3 | 2 | 1 | 0 |                |  |  |  |  |  |  |  |
| 0                             | Sequence Layer embedded in PDO              |       |              |          |   |   |   |   | Sequence Layer |  |  |  |  |  |  |  |
| 1                             | Transaction ID                              |       |              |          |   |   |   |   | Command Layer  |  |  |  |  |  |  |  |
| 2                             | Res-<br>ponse                               | Abort | Segmentation | reserved |   |   |   |   |                |  |  |  |  |  |  |  |
| 3                             | Valid Payload Length                        |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |
| 4                             | Command ID                                  |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |
| 5 .. 6                        | Index                                       |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |
| 7                             | Sub-Index                                   |       |              |          |   |   |   |   | Variable Part  |  |  |  |  |  |  |  |
| 8 – 11                        | Data Size (only if Segmentation = Initiate) |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |
| (8 + 4*d) ..<br>k – (8 + 4*d) | Optional Payload Data                       |       |              |          |   |   |   |   |                |  |  |  |  |  |  |  |

*k* = Length of container in byte

*d*: if seg = initiate    *d* = 1  
else                      *d* = 0

Tab. 85 SDO embedded in PDO

| Field                                | Abbr. | Description   | Value   |
|--------------------------------------|-------|---|---|
| Sequence Layer (see chapter 6.3.3.1) |       |   |   |
| Command Layer                        |       |   |   |
| Transaction ID                       | tid   | Unambiguous transaction ID for a command. Must be changed by the client with every new command. | 0 .. 255  |
| Abort                                | a     | The requested Transfer could not be served by the client/server                                 | 0: Transfer ok<br>1: Abort transfer   |
| Response                             | rsp   | Request / Response  | 0 : Request<br>1 : Response   |
| Segmentation                         | seg   | Differentiates between expedited and segmented transfer   | 0: Expedited Transfer<br>1: Initiate Transfer<br>2: Segment<br>3: Transfer Complete |
| Valid Payload Length                 | vpl   | Length of valid payload data in the container in bytes.   | 0 .. 255  |
| Command ID                           | cid   | Specifies the command   | See Tab. 58   |
| Index                                | i     | Specifies an entry of the device object dictionary  | 0 .. 65535  |
| Sub-Index                            | si    | Specifies a component of a device object dictionary   | 0 .. 254  |
| Data Size                            | ds    | see definition in Tab. 57   |   |

Tab. 86 SDO embedded in PDO field interpretation

The container needs a minimum of 8 bytes for header information.

If segmented transfer is supported, the data size field must be inserted for the initiate frame as defined in the POWERLINK Command Layer Protocol so the header will be 12 byte.

The embedded SDO transfer establishes a peer-to-peer communication channel between two devices. This is a unidirectional Client-Server connection. If a device needs to transfer data using this method to several other devices it must establish a SDO communication channels for each.

The client SDO container (CSDO) and the server SDO container (SSDO) parameter are described by the SDO communication parameter objects SDO\_ServerContainerParam\_XXh\_REC resp.

SDO\_ClientContainerParam\_XXh\_REC. For each SDO channel a pair the communication parameters is mandatory.

### 6.3.3.1 Embedded Sequence Layer for SDO in PDO

For embedding of SDO in cyclic data (PollRequest and PollResponse) the first byte within POWERLINK Command Layer is reserved for Embedded Sequence Layer.

| Byte Offset | Bit Offset             |   |   |   |   |   |   |            |
|-------------|------------------------|---|---|---|---|---|---|------------|
|             | 7                      | 6 | 5 | 4 | 3 | 2 | 1 | 0          |
| 0           | SequenceNumber         |   |   |   |   |   |   | Connection |
| 1 ...       | Command Layer Protocol |   |   |   |   |   |   |            |

Tab. 87 POWERLINK sequence layer for embedding of SDO in cyclic data

Remark:

Only one sequence number for both directions is suitable, because the communication is embedded in the cyclic communication. And therefore it is guaranteed that there are messages in both directions.

| Field          | Abbr. | Description  | Value  |
|----------------|-------|--|--|
| SequenceNumber | snr   | Shall be increased by one with each new request frame. | 0 ... 63   |
| Connection     | con   | Shows the different connection states                  | 0: No connection<br>1: Initialisation<br>2: Connection valid<br>3: Error Response (Retransmission Request) |

Tab. 88 Fields of POWERLINK sequence layer for embedding of SDO in cyclic data

### 6.3.3.1.1 Connection

#### 6.3.3.1.1.1 Initialisation of Connection

Connection is not initialised (e.g. after power up). The server has shut down the connection to this client. Now client and server know that the connection is down. “?” is used for counters that shall be ignored.

After this the connection is established.

The sequence number shall not be incremented until the connection initialisation has been completed.  
No command shall be transferred during the initialisation.

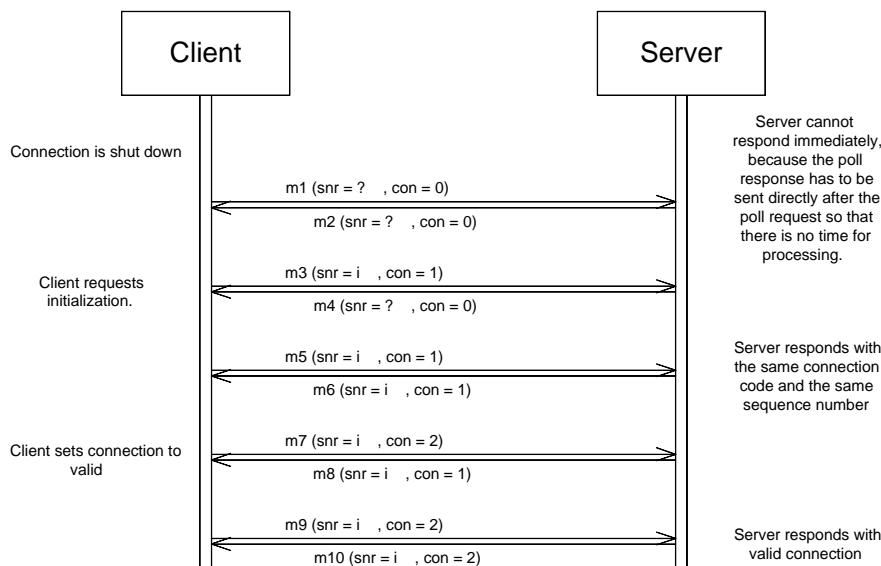


Fig. 60. Initialisation of embedded connection

### 6.3.3.1.1.2 Closing a connection

A connection should be shut down, when it is no longer needed.

A node may shut down a connection if it needs the resources for other reasons.

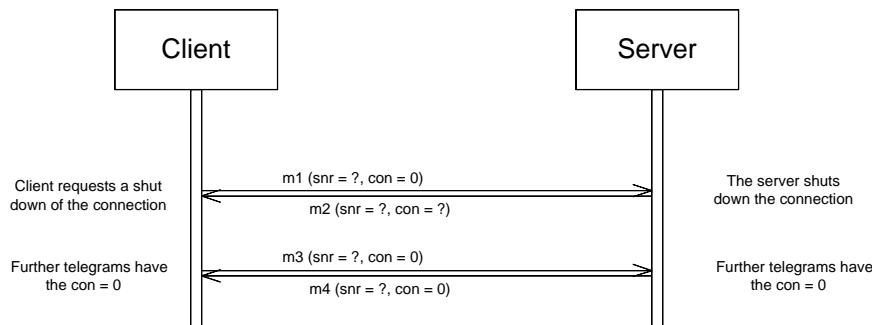


Fig. 61. Closing of connection

Closing a connection shall be indicated by  $\text{con} = 0$ .

### 6.3.3.1.1.3 Data Transfer

When the connection is established the client is allowed to send new request frames. Client and server have to keep the sent frames in some sort of a history buffer.

It is possible to send request frames in advance from client to server in consecutive cyclic frames, even if the responses to the preceding requests have not yet been received. The response frames then are received some cycles later than the corresponding request frames.

If there is nothing to send, the most recently sent packet shall be repeated.

To make the error recovery (see next chapter) for this protocol work, the client has to know how many responses the send history on the server holds. This history size parameter can be read from the object directory, the default value is 1.

The client holds a send history to be able to regain a lost response by repeating the request.

With a send history size of n in the server and a send history size of m in the client, the client shall not forward more than  $\min(m, n)$  requests before receiving the response.

Sample with six new request frames:

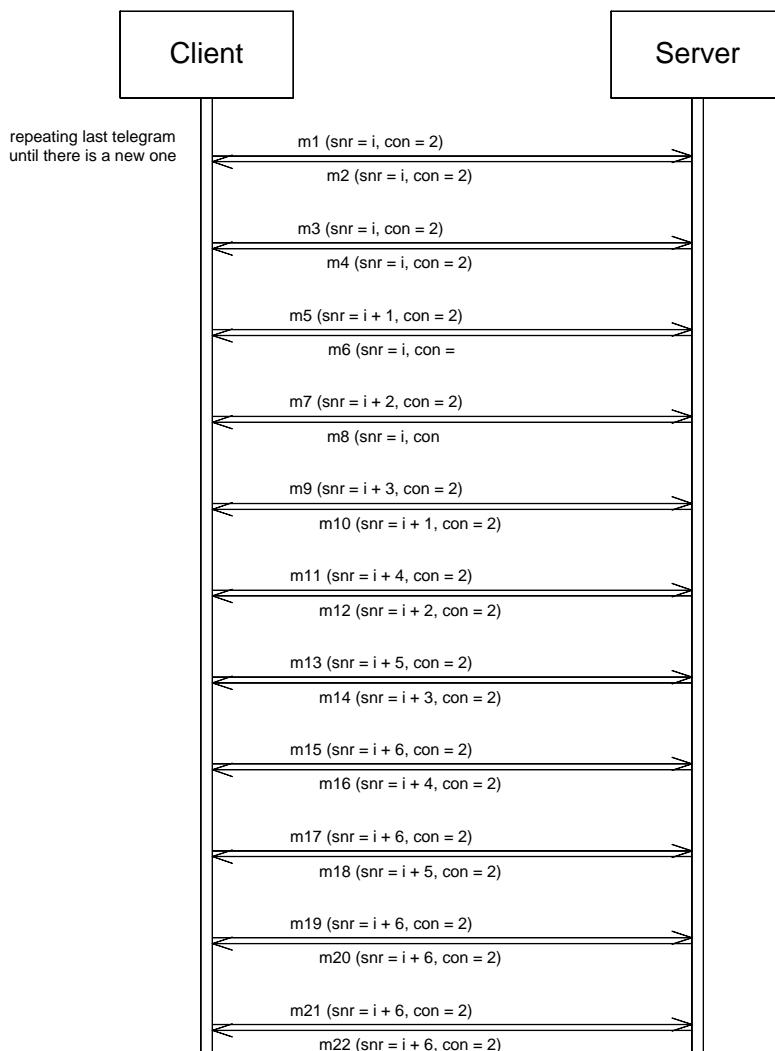


Fig. 62. Normal embedded communication

### 6.3.3.1.2 Errors

#### 6.3.3.1.2.1 Error: Request Lost

If server detects an unexpected sequence number, that is not 1 higher than the last correctly received sequence number, it responds with connection code 3 and the sequence number of the last successful received frame. The client then has to repeat all frames starting after the sequence number of the last successful transferred frame.

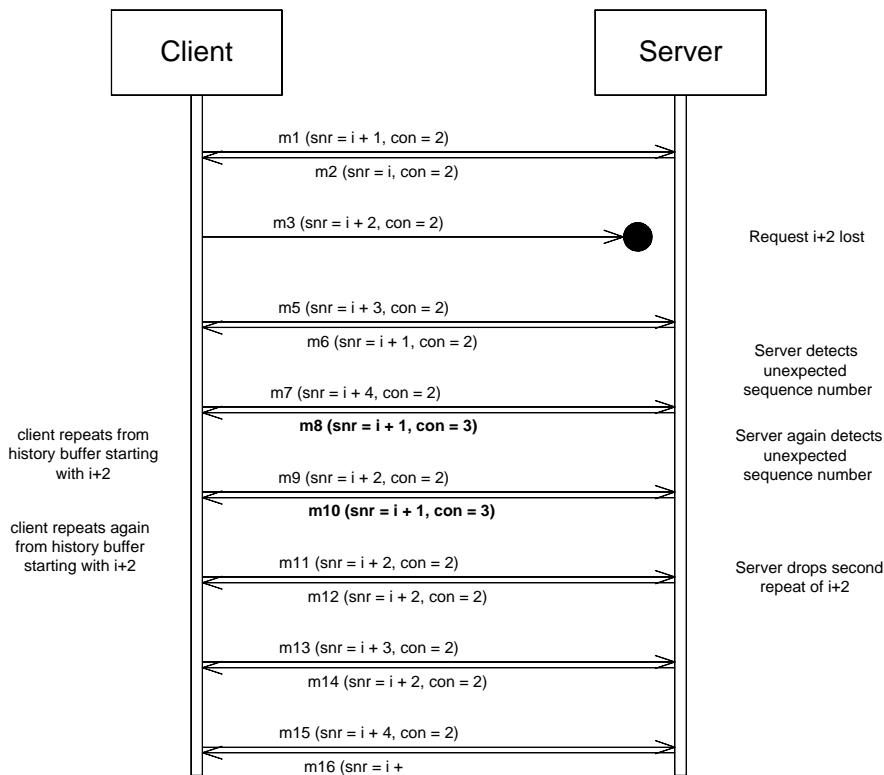


Fig. 63. Error embedded request lost

### 6.3.3.1.2.2 Error: Response Lost

If the client detects an unexpected sequence number that is not 1 higher than the last correctly received sequence number, it has to repeat that frames with connection code 3 for which no response was received.

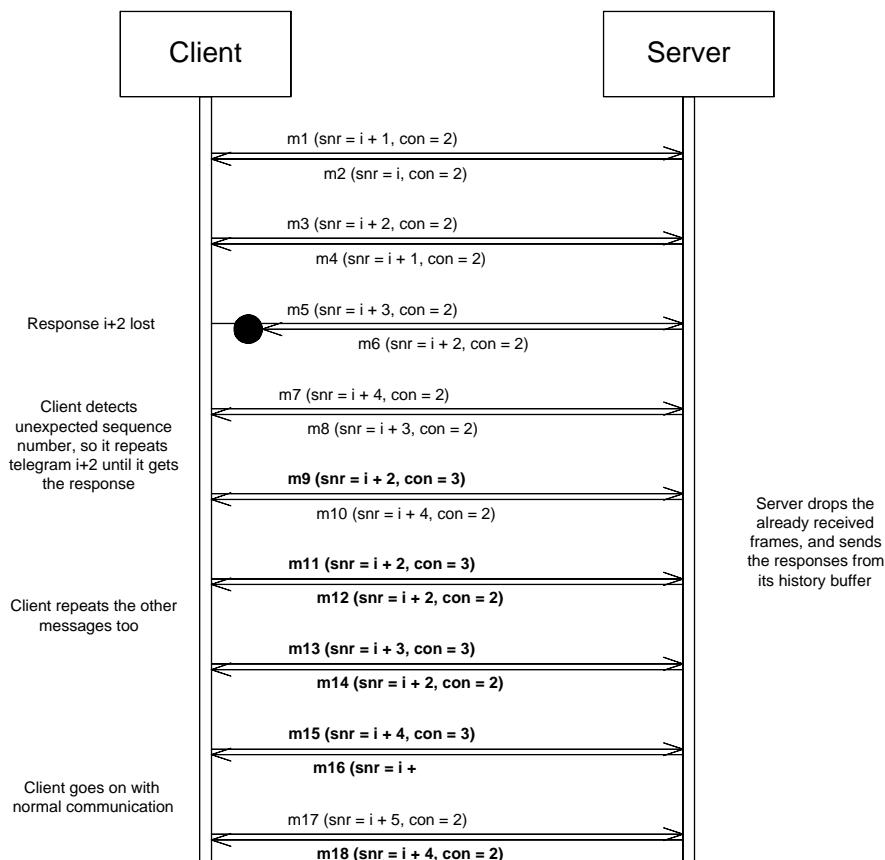


Fig. 64. Error embedded response lost

### 6.3.3.1.3 Handling of Segmented Transfers

#### 6.3.3.1.3.1 Segmented Download from Client to Server

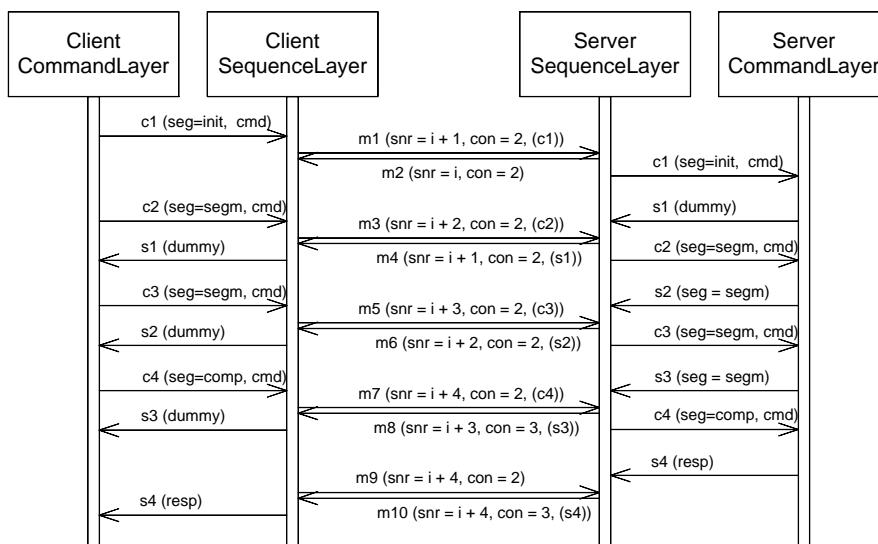


Fig. 65. Embedded segmented download

For SDO embedded in cyclic data each new frame requested by the client shall be responded by the server. In the case of a segmented download from the client to the server, the client will produce more command frames than the server.

So the server shall acknowledge the sequence numbers with dummy frames that contain Command ID "NIL", while the segmented transfer is running.

### 6.3.3.1.3.2 Segmented Upload from Server to Client

In the case of a segmented upload from the server to the client the server will produce more commands than the client. To provide the server with enough sequence numbers the client shall send dummy commands that contain Command ID "NIL", to the server until the upload is complete.

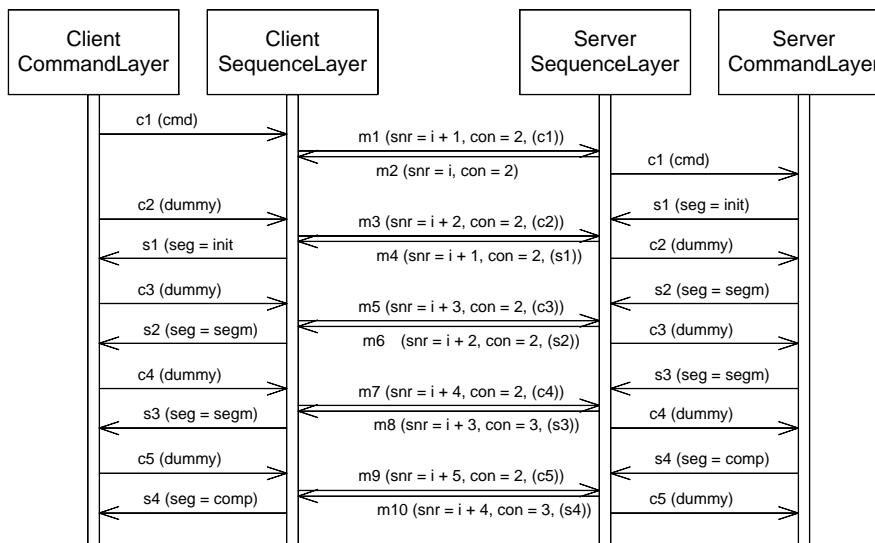


Fig. 66. Embedded segmented upload

### 6.3.3.2 Embedded Command Layer for SDO in Cyclic Data

*Remark: Even though this specification only defines the services Write / Read by Index via SDO in PDO all other commands specified in the SDO Command Layer (chapter 6.3.2.4) can be supported in a similar way.*

#### 6.3.3.2.1 Command Write by Index via PDO

| Byte Offset              | Bit Offset                                  |       |              |   |   |          |   |   |  |  |  |  |  |  |
|--------------------------|---|-------|--------------|---|---|----------|---|---|--|--|--|--|--|--|
|                          | 7   | 6     | 5            | 4 | 3 | 2        | 1 | 0 |  |  |  |  |  |  |
| 0                        | Sequence Layer embedded in PDO              |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 1                        | Transaction ID                              |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 2                        | Res-<br>ponse                               | Abort | Segmentation |   |   | reserved |   |   |  |  |  |  |  |  |
| 3                        | Valid Payload Length                        |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 4                        | Command ID = Write by Index                 |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 5 .. 6                   | Index                                       |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 7                        | Sub-Index                                   |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 8 .. 11                  | Data Size (only if Segmentation = Initiate) |       |              |   |   |          |   |   |  |  |  |  |  |  |
| 8 + 4*d .. k - (8 + 4*d) | Payload Data                                |       |              |   |   |          |   |   |  |  |  |  |  |  |

*k = Length of container in byte*

Tab. 89 Command: Write by Index Request via PDO

### 6.3.3.2.2 Command Read by Index via PDO

| Byte Offset | Bit Offset                     |       |              |   |          |   |   |   |  |  |  |  |  |  |
|-------------|--------------------------------|-------|--------------|---|----------|---|---|---|--|--|--|--|--|--|
|             | 7                              | 6     | 5            | 4 | 3        | 2 | 1 | 0 |  |  |  |  |  |  |
| 0           | Sequence Layer embedded in PDO |       |              |   |          |   |   |   |  |  |  |  |  |  |
| 1           | Transaction ID                 |       |              |   |          |   |   |   |  |  |  |  |  |  |
| 2           | Res-<br>ponse                  | Abort | Segmentation |   | reserved |   |   |   |  |  |  |  |  |  |
| 3           | Valid Payload Length           |       |              |   |          |   |   |   |  |  |  |  |  |  |
| 4           | Command ID = Read by Index     |       |              |   |          |   |   |   |  |  |  |  |  |  |
| 5 – 6       | Index                          |       |              |   |          |   |   |   |  |  |  |  |  |  |
| 7           | Sub-Index                      |       |              |   |          |   |   |   |  |  |  |  |  |  |

Tab. 90 Command: Read by Index Request via PDO

### 6.3.3.3 Object Description

#### 6.3.3.3.1 Object $1200_{\text{h}} \dots 127F_{\text{h}}$ : SDO\_ServerContainerParam\_ $XXh$ \_REC

The SDO\_ServerContainerParam\_ $XXh$ \_REC objects contain the parameters for the SDOs for which the device is the server.

To map the container in the PDO the corresponding index shall be mapped.

To allow access by name " $XXh$ " shall be replaced by a name index. Name index shall be " $00h$ " if object index is  $1200_{\text{h}}$ . It shall be incremented up to " $7Fh$ " corresponding to object index  $127F_{\text{h}}$ .

|           |   |             |        |
|-----------|---|-------------|--------|
| Index     | $1200_{\text{h}} \dots 127F_{\text{h}}$ | Object Type | RECORD |
| Name      | SDO_ServerContainerParam_ $XXh$ _REC    |             |        |
| Data Type | SDO_ParameterRecord_TYPE                | Category    | O      |

- **Sub-Index  $00_{\text{h}}$ : NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | $00_{\text{h}}$ |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 4               | Access      | const |
| Default Value | 4               | PDO Mapping | No    |

- **Sub-Index  $01_{\text{h}}$ : ClientNodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | $01_{\text{h}}$ |             |     |
| Name          | ClientNodeID_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

POWERLINK Node ID of SDO client

- **Sub-Index  $02_{\text{h}}$ : ServerNodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | $02_{\text{h}}$ |             |     |
| Name          | ServerNodeID_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

POWERLINK Node ID of SDO server

- **Sub-Index 03<sub>h</sub>: ContainerLen\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | ContainerLen_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Max. data length of the container (incl. header) in byte

- **Sub-Index 04<sub>h</sub>: HistorySize\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 04 <sub>h</sub> |             |     |
| Name          | HistorySize_U8  |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | 0 .. 63         | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Client Request history size (for sequence layer, see 6.3.3.1)

### 6.3.3.3.2 Object 1280<sub>h</sub> .. 12FF<sub>h</sub>: SDO\_ClientContainerParam\_XXh\_REC

The SDO\_ClientContainerParam\_XXh\_REC objects contain the parameters for the SDOs for which the device is the client. If the entry is supported, all sub-indices must be available.

To map the container in the PDO the corresponding index shall be mapped.

To allow access by name “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1280<sub>h</sub>. It shall be incremented up to “\_7Fh” corresponding to object index 12FF<sub>h</sub>.

|           |  |             |        |
|-----------|--|-------------|--------|
| Index     | 1280 <sub>h</sub> .. 12FF <sub>h</sub> | Object Type | RECORD |
| Name      | SDO_ClientContainerParam_XXh_REC       |             |        |
| Data Type | SDO_ParameterRecord_TYPE               | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00h             |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 5               | Access      | const |
| Default Value | 5               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: ClientNodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | ClientNodeID_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

POWERLINK Node ID of SDO client

- **Sub-Index 02<sub>h</sub>: ServerNodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | ServerNodeID_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

POWERLINK Node ID of SDO server

- Sub-Index 03<sub>h</sub>: ContainerLen\_U8

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | ContainerLen_U8 |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Max. data length of the container (incl. header) in byte

- #### • **Sub-Index 04<sub>h</sub>: HistorySize\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 04 <sub>h</sub> |             |     |
| Name          | HistorySize_U8  |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | 0 .. 63         | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Server Response history size (for sequence layer, see 6.3.3.1)

- Sub-Index 05<sub>h</sub>: reserved

### **6.3.3.3.3 Object 0422<sub>h</sub>: SDO\_ParameterRecord\_TYPE**

|                 |                          |                   |           |
|-----------------|--------------------------|-------------------|-----------|
| Index           | 0422h                    | Object Type       | DEFSTRUCT |
| Name            | SDO_ParameterRecord_TYPE |                   |           |
| Sub-Index       | Component Name           | Value             | Data Type |
| 00 <sub>h</sub> | NumberOfEntries          | 04 <sub>h</sub>   |           |
| 01 <sub>h</sub> | ClientNodeID_U8          | 0005 <sub>h</sub> | UNSIGNED8 |
| 02 <sub>h</sub> | ServerNodeID_U8          | 0005 <sub>h</sub> | UNSIGNED8 |
| 03 <sub>h</sub> | ContainerLen_U8          | 0005 <sub>h</sub> | UNSIGNED8 |
| 04 <sub>h</sub> | HistorySize_U8           | 0005 <sub>h</sub> | UNSIGNED8 |

## 6.3.4 SDO Timeouts

#### **6.3.4.1 Object 1300<sub>h</sub>: SDO\_SequLayerTimeout\_U32**

The object provides a timeout value in [ms] for the connection abort recognition of the SDO sequence layer (see 6.3.2.3.2.5).

|               |                          |             |                     |
|---------------|--------------------------|-------------|---------------------|
| Index         | 1300h                    | Object Type | VAR                 |
| Name          | SDO_SequLayerTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32               | Category    | M                   |
| Value Range   | 100 ms – Max(UNSIGNED32) | Access      | rws, valid on reset |
| Default Value | C_SDO_SEQLAYERTIMEOUT    | PDO Mapping | No                  |

*Note: The default value of C\_SDO\_SEQLAYERTIMEOUT will be high enough even for diagnosing nodes over the internet.*

*Note: Take care when configuring the sequence layer timeout. If the configured value is too low, SDO access to the device is impossible after ResetConfiguration.*

#### **6.3.4.2 Object 1301<sub>h</sub>: SDO\_CmdLayerTimeout\_U32**

The object provides a timeout value in [ms] for the connection abort recognition of the SDO command layer.

If SDO\_CmdLayerTimeout\_U32 is not implemented a value of C\_SDO\_CMDLAYERTIMEOUT shall be used instead for detecting a command layer timeout.

|               |                          |             |                     |
|---------------|--------------------------|-------------|---------------------|
| Index         | 1301h                    | Object Type | VAR                 |
| Name          | SDO_CmdLayerTimeout_U32  |             |                     |
| Data Type     | UNSIGNED32               | Category    | O                   |
| Value Range   | 100 ms – Max(UNSIGNED32) | Access      | rws, valid on reset |
| Default Value | C_SDO_CMDLAYERTIMEOUT    | PDO Mapping | No                  |

*Note: Take care when configuring the command layer timeout.*

*This timeout is valid for each and every SDO command. When configuring its value a long duration for SDO access has to be considered, e.g. for firmware download or storing the OD.*

*If the configured value is too low, SDO access to the device is impossible after ResetConfiguration.*

### 6.3.4.3 Object 1302h: SDO\_SequLayerNoAck\_U32

The object provides the number of acknowledge requests for the connection abort recognition of the SDO sequence layer (see 6.3.2.3.2.5).

|               |                        |             |                     |
|---------------|------------------------|-------------|---------------------|
| Index         | 1302h                  | Object Type | VAR                 |
| Name          | SDO_SequLayerNoAck_U32 |             |                     |
| Data Type     | UNSIGNED32             | Category    | O                   |
| Value Range   | 2 – Max(UNSIGNED32)    | Access      | rws, valid on reset |
| Default Value | C_SDO_SEQLAYERNOACK    | PDO Mapping | No                  |

## 6.4 Process Data Object (PDO)

The real-time data transfer is performed by means of Process Data Objects (PDO).

PDO communication in POWERLINK is always performed isochronously by PReq and/or PRes frames. The PRes frames are sent as broadcasts following the producer/consumer scheme. The PReq frames with unicast addresses comply with the master/slave relationship.

The transmission type of PDO is continuous. There is no “on event” or “on change” transmission type provided.

From the device’s view, there are two types of PDO usage: data transmission and data reception. Transmit PDOs (TPDOs) and Receive PDOs (RPDOs) shall be distinguished. Devices supporting TPDOs are PDO producers or PDO masters and devices which are able to receive PDOs are called PDO consumers or PDO slaves.

The numbers of supported channels shall be provided by the application.

An MN device may support up to 256 RPDO and 256 TPDO channels.

On a CN device, only one TPDO channel may be available while up to 256 RPDO may be supported . An Async-only CN device shall not support any TPDO channel but may provide up to 256 RPDO channels.

The size of PDO channels is application-specific.

A PDO channel shall be described by a pair of objects:

- PDO communication parameter describing communication attributes of the PDO. Mapping version and address information are provided.  
PDO communication parameter are held by the objects PDO\_TxCommParam\_XXh\_REC and PDO\_RxCommParam\_XXh\_REC (Index number 1800<sub>h</sub> .. 18FF<sub>h</sub> resp. 1400<sub>h</sub> .. 14FF<sub>h</sub> ).
- PDO mapping parameter describing the mapping of the objects contained in PDO payload to object dictionary entries.  
PDO mapping parameter are held by the objects PDO\_TxMappParam\_XXh\_AU64 and PDO\_RxMappParam\_XXh\_AU64 (Index number 1A00<sub>h</sub> .. 1AFF<sub>h</sub> resp. 1600<sub>h</sub> .. 16FF<sub>h</sub> ).

Corresponding PDO communication and mapping parameters shall be identified by equal low byte values of the index numbers. TPDO and RPDO object pairs shall be distinguished.

The mapping of application objects into a PDO may be transmitted to a device during the device configuration process by applying the SDO services to the corresponding entries of the Object Dictionary.

Regarding the complete network, TPDO and RPDO mappings applied to a dedicated isochronous frame may be different. Especially in producer/consumer relationships using PRes frames, individual objects mapped by the TPDO mapping of the producer may be targeted to different consumers. The RPDO mapping of a consumer may skip objects not targeted to it.

## 6.4.1 PDO Mapping Limitations

PDO mapping description may require a large amount of memory. Other restriction may derive from the internal structure of a device, e.g. buffer size, interfaces etc.

Devices may limit the amount of PDO data to be transmitted and received per POWERLINK cycle.

A device shall not impose restrictions on the sequence of mapped objects within a PDO.

### 6.4.1.1 TPDO Mapping Limitations

Transmit PDO description amount may be limited in following ways:

- The maximum number of supported TPDO channels may be indicated by D\_PDO\_TPDOChannels\_U16. It limits the number of PDO\_TxCommParam\_XXh\_REC and PDO\_TxMappParam\_XXh\_AU64 objects.
- The maximum number of mapped objects per TPDO channel may be indicated by D\_PDO\_TPDOChannelObjects\_U8. It limits the number of PDO\_TxMappParam\_XXh\_AU64 sub-indices.
- An overall limit to the number of mapped objects regardless their distribution to channels may be indicated by D\_PDO\_TPDOOverallObjects\_U16. It provides an upper limit to the sum of the TPDO objects of all channels.
- D\_PDO\_TPDOChannels\_U16, D\_PDO\_TPDOChannelObjects\_U8 and D\_PDO\_TPDOOverallObjects\_U16 may be combined. If one of the values violates a calculatoric limit given by the other ones it shall be ignored.

*Example: D\_PDO\_TPDOChannels\_U16 and D\_PDO\_TPDOChannelObjects\_U8 together provide an upper limit to the overall number of available mapping object entries. If D\_PDO\_TPDOOverallObjects\_U16 is greater than the product of D\_PDO\_TPDOChannels\_U16 and D\_PDO\_TPDOChannelObjects\_U8, it shall be ignored.*

TPDO limits caused by the internal structure of a device shall indicated as following:

- On a per telegram base, TPDO may be limited by
  - TX buffer size,
  - the bandwidth of an internal interface, that transport complete TPDO payload from application to POWERLINK stack
  - etc.

Such kind of limits shall be indicated by NMT\_CycleTiming\_REC.IsochrTxMaxPayload\_U16.

- On a per cycle base, TPDO data amount may be limited by the bandwidth of an internal interface that transports all TPDO data that shall be transmitted during a cycle. D\_PDO\_TPDOCycleDataLim\_U32 shall indicate such kind of limit.

It is assumed that TPDO telegram generation is performed on the POWERLINK stack side of the interface. TPDO objects mapped to multiple TPDO channels shall be transported once over the interface and distributed to the channels on the POWERLINK stack side. Thus they shall be once taken into account when calculating the bandwidth.

### 6.4.1.2 RPDO Mapping Limitations

Receive PDO description amount may be limited in following ways:

- The maximum number of supported RPDO channels may be indicated by D\_PDO\_RPDOChannels\_U16. It limits the number of PDO\_RxCommParam\_XXh\_REC and PDO\_RxMappParam\_XXh\_AU64 objects.
- The maximum number of mapped objects per RPDO channel may be indicated by D\_PDO\_RPDOChannelObjects\_U8. It limits the number of PDO\_RxMappParam\_XXh\_AU64 sub-indices.

- An overall limit to the number of mapped objects regardless their distribution to channels may be indicated by D PDO\_RPDOOverallObjects\_U16. It provides an upper limit to the sum of the RPDO objects of all channels.
- D PDO\_RPDOChannels\_U16, D PDO\_RPDOChannelObjects\_U8 and D PDO\_RPDOOverallObjects\_U16 may be combined. If one of the values violates a calculatoric limit given by the other ones it shall be ignored.

*Example:* refer 6.4.1.1

RPDO limits caused by the internal structure of a device shall indicated as following:

- On a per telegram base, RPDO may be limited by
    - RX buffer size,
    - the bandwidth of an internal interface, that transport complete RPDO payload from POWERLINK stack to application
    - etc.
- Such kind of limits shall be indicated by NMT\_CycleTiming\_REC.IsochrRxMaxPayload\_U16.
- On a per cycle base, RPDO data amount may be limited by the bandwidth of an internal interface that transports all RPDO data that have been received during a cycle. D PDO\_RPDOCycleDataLim\_U32 shall indicate such kind of limit.

### 6.4.1.3 Further Limitations

The overall memory consumption of the objects defining the mapping (PDO\_TxMappParam\_XXh\_AU64, PDO\_TxCommParam\_XXh\_REC, PDO\_RxMappParam\_XXh\_AU64 and PDO\_RxCommParam\_XXh\_REC) may be limited. The limit is given by D PDO\_MaxDescrMem\_U32. The size is calculated as follows:

- Each object PDO\_TxCommParam\_XXh\_REC whose mapping is activated (PDO\_TxMappParam\_XXh\_AU64.NumberOfEntries is not equal to 0) uses 2 Bytes.
- Each object PDO\_RxCommParam\_XXh\_REC whose mapping is activated (PDO\_RxMappParam\_XXh\_AU64.NumberOfEntries is not equal to 0) uses 2 Bytes.
- Each object PDO\_TxMappParam\_XXh\_AU64 uses PDO\_TxMappParam\_XXh\_AU64.NumberOfEntries \* 8 Bytes.
- Each object PDO\_RxMappParam\_XXh\_AU64 uses PDO\_RxMappParam\_XXh\_AU64.NumberOfEntries \* 8 Bytes.

The sum of these sizes must not exceed D PDO\_MaxDescrMem\_U32.

The minimum size of mapped objects may be indicated by D PDO\_Granularity\_U8. The mapped offset shall be a multiple of the granularity.

## 6.4.2 PDO Mapping Version

Compatibility of TPDO channels and corresponding RPDO channels may be ensured by PDO mapping version handling.

PDO mapping version is held by PDO\_TxCommParam\_XXh\_REC.MappingVersion\_U8 and PDO\_RxCommParam\_XXh\_REC.MappingVersion\_U8. Tab. 91 shows the PDO mapping main and sub version encoding.

| High nibble  | Low nibble  |
|--------------|-------------|
| Main version | Sub version |

Tab. 91 Structure of PDO Mapping version

The assigned value of the PDO Mapping version is application specific.

If the PDO Mapping is changed in a compatible manner e.g. expanding the PDO contents, the sub version shall be incremented.

PDO Mapping may be variable or static. Variable mapping may be dynamically modified by the application, even under operation. Static mapping is pre-defined and may not be modified in any way.

Support of variable mapping shall be indicated by NMT\_FeatureFlags\_U32 Bit 6 and D PDO\_DynamicMapping\_BOOL.

To control compatibility of PDO mapping, PDO\_TxCommParam\_XXh\_REC.MappingVersion shall be set. The version info shall be transmitted by the master or producer with every PDO transporting PReq and PRes frame (s. 4.6.1.1.3 resp. 4.6.1.1.4).

The PDO slave or consumer shall check the mapping version of received PDOs using the corresponding PDO\_RxCommParam\_XXh\_REC.MappingVersion entry. PDOs with differing main version shall be ignored. PDOs with equal main version but differing sub version shall be accepted.

A PDO mapping version value of 0 indicates that there is no mapping version available. This does not disable the version check. If the mapping version in the received PDO is 0 the corresponding PDO\_RxCommParam\_XXh\_REC.MappingVersion entry shall be 0 too and vice versa. Otherwise the received mapping is ignored.

### 6.4.3 SDO via PDO Container

A container may be used by the PDO mapping to enable exchange of SDO data via PDO communication channels.

See 6.3.3 for detailed description of SDO embedded in PDO.

The type of container is defined by a referencing object at index SDO\_ServerContainerParam\_XXh\_REC or SDO\_ClientContainerParam\_XXh\_REC

### 6.4.4 Transmit PDOs

If sub-index 0 of the mapping object (PDO\_TxMappParam\_XXh\_AU64) is 0 the TPDO is invalid. The RD flag of the TPDO transporting Frame shall be reset. (refer Tab. 105 or 7.1.4)

Sending PDO data is implicitly isochronous for a node in the state NMT\_CS\_OPERATIONAL and NMT\_MS\_OPERATIONAL. In NMT\_CS\_READY\_TO\_OPERATE and NMT\_MS\_READY\_TO\_OPERATE, PDO data are sent in the same way, but they are not valid. The RD flag of the TPDO transporting Frame shall be reset..

### 6.4.5 Receive PDOs

RPDO data shall be valid if the RD flag of the RPDO transporting frame is set in the state NMT\_CS\_OPERATIONAL and NMT\_MS\_OPERATIONAL. The data shall be occurred to the objects assigned by the RPDO mapping parameters. The application may access the received PDO data by reading these objects.

If the RD flag is not set in the received message, the mapping shall not be performed, e.g. the data shall not be copied to the mapped objects. The application shall further make use of the old data earlier received. RD flag signaling to the application shall be implementation specific.

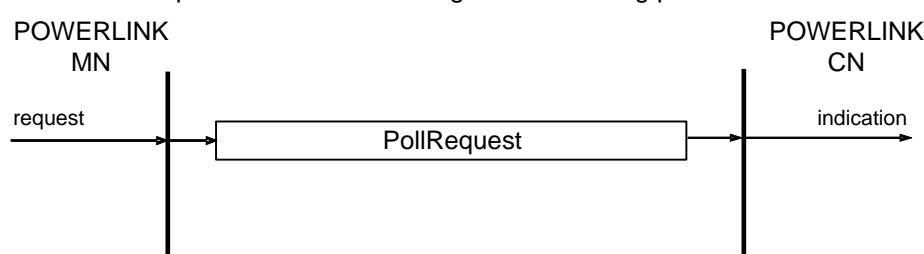
In NMT states not equal NMT\_CS\_OPERATIONAL resp. NMT\_MS\_OPERATIONAL, RPDO data shall be invalid and shall be ignored.

If the length of the data actually received is less than the length of the mapped objects or if the received and the expected PDO mapping versions differ in an incompatible manner the received data has to be ignored and a fault situation occurs.

### 6.4.6 PDO via PReq

PDO via PReq transmission follows the master/slave relationship as described in 2.3.1.

PDO via PReq is carried out according to the following protocol.



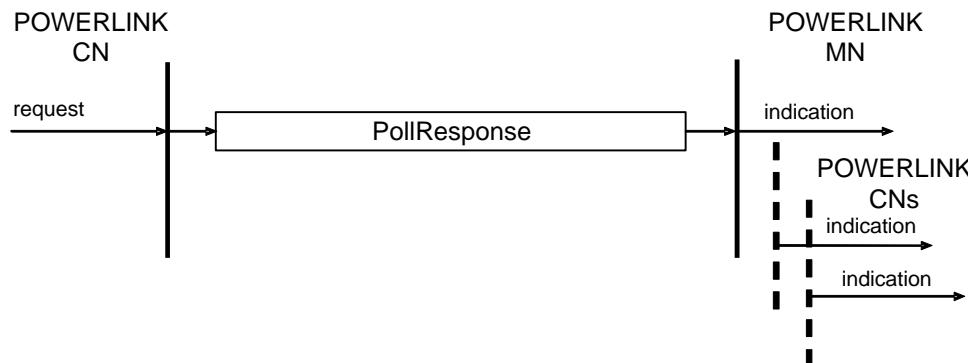
The following data elements in the PReq frame (for the frame structure see 4.6.1.1.3) are relevant for PDO transport:

- The RD flag indicates if the PDO data is valid. If the bit is  $0_b$ , the PDO data is not valid and shall not be interpreted by the POWERLINK CN.
- Size indicates the user data length of the PDO payload data.
- Payload indicates the PDO data.

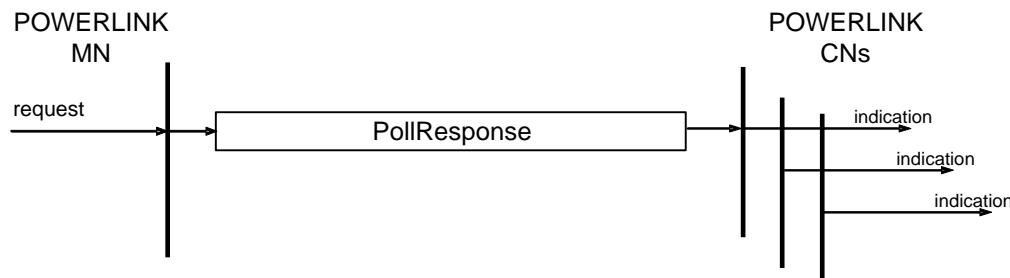
### 6.4.7 PDO via PRes

PDO via PRes transmission follows the producer/consumer relationship as described in 2.3.3.

PDO via PRes from a CN is carried out according to the following protocol.



PDO via PRes from the MN is carried out according to the following protocol.



The following data elements in the PRes frame (for the frame structure see 4.6.1.1.4) are relevant for PDO transport:

- The RD flag indicates if the PDO data is valid. If the bit is  $0_b$ , the PDO data is not valid and shall not be interpreted by the POWERLINK CN.
- Size indicates the user data length of the PDO payload data.
- Payload indicates the PDO data.

### 6.4.8 PDO Error Handling

#### 6.4.8.1 Dynamic Errors

##### 6.4.8.1.1 Incompatible Mapping

If an incompatible PDO Mapping version is received, the PDO shall be ignored.

This error situation shall be logged and signaled to the application. Typically this error occurs many times, so the error shall be logged and signaled once for every received wrong PDO mapping version. The bit corresponding to the error causing node may be set to  $1_b$  in PDO\_ErrMapVers\_OSTR.

| Error code     | Description  |
|----------------|--|
| E_PDO_MAP_VERS | PDO with wrong Mapping version received, PDO ignored |

##### 6.4.8.1.2 Unexpected End of PDO

If a PDO is received which is shorter than the amount of mapped objects, the PDO shall be ignored.

This error situation shall be logged and signaled to the application. Normally this error occurs many times, so the error shall be logged and signaled once for each PDO. The bit corresponding to the error causing Node may be set to  $1_b$  in PDO\_ErrShort\_RX\_OSTR.

| Error code     | Description                          |
|----------------|--------------------------------------|
| E PDO_SHORT_RX | RX PDO length too short, PDO ignored |

### 6.4.8.2 Configuration Errors

If an attempt to change the PDO mapping results in a memory consumption of mapped objects that exceeds the configured payload size limits NMT\_CycleTiming\_REC.IsochrRxMaxPayload\_U16 or NMT\_CycleTiming\_REC.IsochrTxMaxPayload\_U16, this attempt shall be rejected anyway.

If only the limit NMT\_CycleTiming\_REC.PReqActPayloadLimit\_U16 or NMT\_CycleTiming\_REC.PResActPayloadLimit\_U16 is violated the mapping shall be activated. However the RD flag shall be reset resp. the received data shall be ignored.

If any of the limits introduced in 6.4.1 is violated, the attempt shall be rejected too.

The memory size check shall be done when a mapping is enabled by writing the number of mapped objects to PDO\_RxMappParam\_XXh\_AU64.NumberOfEntries or PDO\_TxMappParam\_XXh\_AU64.NumberOfEntries. .

| Error code        | Description  |
|-------------------|--|
| E PDO_MAP_OVERRUN | Mapping exceeds payload size or mapped object number limit |

### 6.4.9 Object Description

#### 6.4.9.1 Object 1400<sub>h</sub> .. 14FF<sub>h</sub>: PDO\_RxCommParam\_XXh\_REC

This description handles a sequence of up to 256 objects. These indices describe the communication attributes of the RPDO channels. Mapping version and address information are provided.

The number of objects may be less than 256 (refer 6.4.1). Objects shall be implemented starting at Index 1400<sub>h</sub>.

The validity of the respective object depends on the NumberOfEntries\_U8 entry of the corresponding (s. 6.3.4) RPDO mapping index PDO\_RxMappParam\_XXh\_AU64.

To change the PDO communication parameter, the PDO shall be deactivated by setting PDO\_RxMappParam\_XXh\_AU64.NumberOfEntries to 0. To enable modifications, PDO\_RxMappParam\_XXh\_AU64.NumberOfEntries shall be set to a value not equal 0.

|           |  |             |        |
|-----------|--|-------------|--------|
| Index     | 1400 <sub>h</sub> .. 14FF <sub>h</sub> | Object Code | RECORD |
| Name      | PDO_RxCommParam_XXh_REC                |             |        |
| Data Type | PDO_CommParamRecord_TYPE               | Category    | Cond   |

To allow access by name, “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1400<sub>h</sub>. It shall be incremented up to “\_FFh” corresponding to object index 14FF<sub>h</sub>.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2               | Access      | const |
| Default Value | 2               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: NodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | NodeID_U8       |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | 0, 1 .. 254     | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Node ID of the node transmitting the corresponding PRes. Valid Node IDs shall be determined by NMT\_NodeAssignment\_AU32 [Node ID] Bits 0 and 8.

Node ID entry 0 is reserved for PReq received by a CN.

Node ID entry C\_ADR\_SELF\_ADR\_NODE\_ID shall indicate mapping of the RPDO to the TPDO describing the locally transmitted PRes frame. This Node ID always refers to the locally transmitted PRes, independent of the local Node ID setting.

The local Node ID may be also used to indicate the reception of self-transmitted PDO (e.g. Node ID 3). But in this case the mapping is fixed to this Node ID. If the local Node ID changes (e.g. from 3 to 4), the mapping still refers to the PRes of Node ID 3

Receipt of self-transmitted PDO is optional. It shall be indicated by the device description entry D(PDO\_SelfReceipt\_BOOL).

- **Sub-Index 02<sub>h</sub>: MappingVersion\_U8**

|               |                   |             |          |
|---------------|-------------------|-------------|----------|
| Sub-Index     | 02 <sub>h</sub>   |             |          |
| Name          | MappingVersion_U8 |             |          |
| Data Type     | UNSIGNED8         | Category    | M        |
| Value Range   | UNSIGNED8         | Access      | rws / ro |
| Default Value | -                 | PDO Mapping | No       |

Access shall be ro if only static mapping is provided by the device.

#### 6.4.9.2 Object 1600<sub>h</sub>..16FF<sub>h</sub> PDO\_RxMappParam\_XXh\_AU64

This description handles a sequence of up to 256 objects. These indices describe the mapping of the objects contained in RPDO payload to object dictionary entries.

The number of objects and subindices may be limited (refer 6.4.1). Objects shall be implemented starting at Index 1600<sub>h</sub>.

To change the PDO mapping, the PDO shall be deactivated by setting NumberOfEntries to 0. The objects may then be remapped. To enable modifications, NumberOfEntries shall be set to the actual number of mapped objects. When the mapping is enabled, it shall be verified, that the cumulative length of all mapped objects does not violate the payload size limit NMT\_CycleTiming\_REC.IsochrRxMaxPayload\_U16 and on the CN additionally the limit NMT\_CycleTiming\_REC.PReqActPayloadLimit\_U16 for the PReq frame. For the appropriate error handling see 6.4.8.2.

On some device, add or remove entries to or from the current list of mapping entries may be performed by modifying NumberOfEntries without effecting the ObjectMapping sub-indices. To do so, NumberOfEntries shall be set to 0 before writing the new NumberOfEntries value.

Default mapping values may be provided by the device profile.

|           |  |             |       |
|-----------|--|-------------|-------|
| Index     | 1600 <sub>h</sub> .. 16FF <sub>h</sub> | Object Code | ARRAY |
| Name      | PDO_RxMappParam_XXh_AU64               |             |       |
| Data Type | UNSIGNED64                             | Category    | Cond  |

To allow access by name, “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1600<sub>h</sub>. It shall be incremented up to “\_FFh” corresponding to object index 16FF<sub>h</sub>.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub> |             |     |
| Name          | NumberOfEntries |             |     |
| Value Range   | 0, 1 .. 254     | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Number of mapped objects. 0 indicates that the mapping index and the corresponding RPDO communication index (6.4.9.1) is deactivated.

On devices providing only static mapping, the value shall be set to the number of entries corresponding to the static mapping or 0.

*Note: By this it is also possible to activate / deactivate (0) a static mapping.*

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: ObjectMapping**

|               |                                    |             |          |
|---------------|------------------------------------|-------------|----------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |          |
| Name          | ObjectMapping                      |             |          |
| --            | --                                 | Category    | O        |
| Value Range   | UNSIGNED64                         | Access      | rws / ro |
| Default Value | -                                  | PDO Mapping | No       |

Access shall be ro if only static mapping is provided by the device.

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub> Value Interpretation**

For every PDO channel up to 254 objects may be mapped.

The offset related to the start address of the PDO payload and the length of data shall be provided for every mapped object.

| Octet Offset | Name      | Description  |
|--------------|-----------|--|
| 0 – 1        | Index     | Index of the object to be mapped                   |
| 2            | Sub-index | Sub-index of the object to be mapped               |
| 3            | reserved  | for alignment purpose                              |
| 4 – 5        | Offset    | Offset related to start of PDO payload (Bit count) |
| 6 – 7        | Length    | Length of the mapped object (Bit count)            |

Tab. 92 Structure of PDO Mapping Entry

Tab. 93 shows the internal bit mapping of an object mapping entry. See 6.1.4.5 for information about data type encoding for transmission purpose.

| UNSIGNED64 |            |            |          |           |            |  |     |
|------------|------------|------------|----------|-----------|------------|--|-----|
|            | MSB        |            |          |           |            |  | LSB |
| Bits       | 63 .. 48   | 47 .. 32   | 31 .. 24 | 23 .. 16  | 15 .. 0    |  |     |
| Name       | Length     | Offset     | reserved | Sub-index | Index      |  |     |
| Encoding   | UNSIGNED16 | UNSIGNED16 | -        | UNSIGNED8 | UNSIGNED16 |  |     |

Tab. 93 Internal bit mapping of PDO mapping entry

Mapped objects of length  $\geq$  8 Bits shall be aligned to a byte boundary offset.

Overlapping of mapped objects should be avoided.

*The following example shows the mapping interpretation:*

*Index 6000<sub>h</sub>, Sub-index 4 with a data length of 8 bit shall be mapped to offset 16 [bit].*

|             |               |               |                 |                  |                   |
|-------------|---------------|---------------|-----------------|------------------|-------------------|
| <i>Bits</i> | 63 .. 48      | 47 .. 32      | 31 .. 24        | 23 .. 16         | 15 .. 0           |
| <i>Name</i> | <i>Length</i> | <i>Offset</i> | <i>reserved</i> | <i>Sub-index</i> | <i>Index</i>      |
| <i>Data</i> | 8             | 16            | 0               | 4                | 6000 <sub>h</sub> |

The object above is mapped with the following value (big endian):

- 0008.0010.00.04.6000<sub>h</sub>

It is transmitted in the following order (little endian):

- 0060.04.00.1000.0800<sub>h</sub>

### 6.4.9.3 Object 1800<sub>h</sub>..18FF<sub>h</sub> PDO\_TxCommParam\_XXh\_REC

This description handles a sequence of up to 256 objects. These indices describe the communication attributes of the TPDO channels. Mapping version and address information are provided.

As a CN has only one TPDO channel, only the first index PDO\_TxCommParam\_XXh\_REC shall be implemented on a CN.

On the MN, the number of objects may be less than 256 (refer 6.4.1).

Objects shall be implemented starting at Index 1800<sub>h</sub>.

The validity of the respective object depends on the NumberOfEntries\_U8 entry of the corresponding (s. 6.3.4) TPDO mapping index PDO\_TxMappParam\_XXh\_AU64.

To change the PDO communication parameter, first the PDO has to be deactivated by means of setting PDO\_TxMappParam\_XXh\_AU64.NumberOfEntries to 0. To enable modifications, PDO\_TxMappParam\_XXh\_AU64.NumberOfEntries shall be set to a value not equal to 0.

|           |  |             |        |
|-----------|--|-------------|--------|
| Index     | 1800 <sub>h</sub> .. 18FF <sub>h</sub> | Object Code | RECORD |
| Name      | PDO_TxCommParam_XXh_REC                |             |        |
| Data Type | PDO_CommParamRecord_TYPE               | Category    | Cond   |

To allow access by name, “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1800<sub>h</sub>. It shall be incremented up to “\_FFh” corresponding to object index 18FF<sub>h</sub>.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2               | Access      | const |
| Default Value | 2               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: NodeID\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | NodeID_U8       |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | 0, 1 .. 254     | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Node ID of the PDO target:

- CN: not used (0)
- MN: Node ID of the PReq target (CN). Valid Node IDs shall be released by NMT\_NodeAssignment\_AU32 [Node ID] Bits 0 and 8.  
Node ID entry 0 shall indicate multicast PRes transmitted by the MN.

- **Sub-Index 02<sub>h</sub>: MappingVersion\_U8**

|               |                   |             |          |
|---------------|-------------------|-------------|----------|
| Sub-Index     | 02 <sub>h</sub>   |             |          |
| Name          | MappingVersion_U8 |             |          |
| Data Type     | UNSIGNED8         | Category    | M        |
| Value Range   | UNSIGNED8         | Access      | rws / ro |
| Default Value | -                 | PDO Mapping | No       |

Access shall be ro if only static mapping is provided by the device.

#### 6.4.9.4 Object 1A00<sub>h</sub> .. 1AFF<sub>h</sub> PDO\_TxMappParam\_XXh\_AU64

This description handles a sequence of up to 256 objects. These indices describe the mapping of the objects contained in TPDO payload to object dictionary entries.

As a CN has only one TPDO channel, only the first index PDO\_TxMappParam\_XXh\_AU64 shall be implemented on a CN.

On the MN, the number of objects may be less than 256 (refer 6.4.1).

Objects shall be implemented starting at Index 1A00<sub>h</sub>.

The number of subindices may be limited (refer 6.4.1).

To change the PDO mapping, the PDO shall be deactivated by setting NumberOfEntries to 0. The objects may then be remapped. To enable modifications, NumberOfEntries shall be set to the actual number of mapped objects. When the mapping is enabled, it shall be verified, that the cumulative length of all mapped objects does not violate the payload size limit NMT\_CycleTiming\_REC.IsochrTxMaxPayload\_U16 and additionally the limit NMT\_CycleTiming\_REC.PResActPayloadLimit\_U16 for the PRes frame. For the appropriate error handling see 6.4.8.2.

On some device, add or remove entries to or from the current list of mapping entries may be performed by modifying NumberOfEntries without effecting the ObjectMapping sub-indices. To do so, NumberOfEntries shall be set to 0 before writing the new NumberOfEntries value.

Default values may be provided by the device profile.

|           |  |             |       |
|-----------|--|-------------|-------|
| Index     | 1A00 <sub>h</sub> .. 1AFF <sub>h</sub> | Object Code | ARRAY |
| Name      | PDO_TxMappParam_XXh_AU64               |             |       |
| Data Type | UNSIGNED64                             | Category    | Cond  |

To allow access by name, “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1A00<sub>h</sub>. It shall be incremented up to “\_FFh” corresponding to object index 1AFF<sub>h</sub>.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub> |             |     |
| Name          | NumberOfEntries |             |     |
| Value Range   | 0, 1 .. 254     | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

0 indicates that the mapping index and the corresponding TPDO communication index (6.4.9.1) is deactivated. The RD flag of the TPDO transporting frame shall be reset (TPDO is invalid).

On devices only providing static mapping, the value shall be set to the number of entries corresponding to the static mapping or 0.

*Note: By this it is also possible to activate / deactivate (0) a static mapping.*

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: ObjectMapping**

|               |                                    |             |          |
|---------------|------------------------------------|-------------|----------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |          |
| Name          | ObjectMapping                      |             |          |
| --            | --                                 | Category    | O        |
| Value Range   | UNSIGNED64                         | Access      | rws / ro |
| Default Value | -                                  | PDO Mapping | No       |

Access shall be ro if only static mapping is provided by the device.

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub> Value Interpretation**

see PDO\_RxMappParam\_XXh\_AU64.ObjectMapping Value Interpretation (6.4.9.2)

#### 6.4.9.5 Object 1C80<sub>h</sub>: PDO\_ErrMapVers\_OSTR

This object contains a list of all the nodes, sending a wrong mapping version (see 6.4.8.1.1).

The bits are arranged in the node list format (refer 7.3.1.2.3). If a wrong mapping version is detected, the bit corresponding to the node transmitting the PDO, shall be set to 1<sub>b</sub>. Reset shall be handled by the application.

*Hint: A configuration tool shall refresh the object to avoid erroneously pending error indication.*

| Index         | 1C80 <sub>h</sub>   | Object Type | VAR |
|---------------|---------------------|-------------|-----|
| Name          | PDO_ErrMapVers_OSTR |             |     |
| Data Type     | OCTET_STRING32      | Category    | O   |
| Value Range   | -                   | Access      | rw  |
| Default Value | 0                   | PDO Mapping | No  |

### 6.4.9.6 Object 1C81<sub>h</sub>: PDO\_ErrShort\_RX\_OSTR

This object contains a list of all the nodes, sending a too short PDO (see 6.4.8.1.2).

The bits are arranged in the node list format (refer 7.3.1.2.3). If a short PDO is detected, the bit corresponding to the node transmitting the PDO, shall be set to 1<sub>b</sub>. Reset shall be handled by the application.

*Hint: A configuration tool shall refresh the object to avoid erroneously pending error indication.*

|               |                      |             |     |
|---------------|----------------------|-------------|-----|
| Index         | 1C81 <sub>h</sub>    | Object Type | VAR |
| Name          | PDO_ErrShort_RX_OSTR |             |     |
| Data Type     | OCTET_STRING32       | Category    | O   |
| Value Range   | -                    | Access      | rw  |
| Default Value | 0                    | PDO Mapping | No  |

### 6.4.9.7 Object 0420<sub>h</sub>: PDO\_CommParamRecord\_TYPE

|                 |                          |                   |           |
|-----------------|--------------------------|-------------------|-----------|
| Index           | 0420 <sub>h</sub>        | Object Type       | DEFSTRUCT |
| Name            | PDO_CommParamRecord_TYPE |                   |           |
| Sub-Index       | Component Name           | Value             | Data Type |
| 00 <sub>h</sub> | NumberOfEntries          | 02 <sub>h</sub>   |           |
| 01 <sub>h</sub> | NodeID_U8                | 0005 <sub>h</sub> | UNSIGNED8 |
| 02 <sub>h</sub> | MappingVersion_U8        | 0005 <sub>h</sub> | UNSIGNED8 |

## 6.5 Error Signaling

This chapter describes how to record errors and events which are generated by a POWERLINK Node and the procedure how a CN shall indicate and transfer errors/events to the MN.

All communication layers as well as the application shall have access to the Error Signaling.

The Data Link Layer of a CN shall query the Error Signaling cyclically for new StatusResponse data.

If there was no change in the Static Error Bit Field and the Status Entries since the last query from the DLL and the Emergency Queue is empty, the Error Signaling shall not provide data to the DLL.

If there was a change or the Emergency Queue is not empty the Error Signaling shall generate new StatusResponse data and pass it to the DLL at the next query.

- Static Error Bit Field :     8 Bytes, see 6.5.8.1
- Error Entry:                 20 Bytes / Entry, see Tab. 94

If the Emergency Queue is used, the space for at least 1 History Entry shall be available in the StatusResponse frame. This History Entry may not contain emergency data to make sure the CN can transmit the Emergency Queue entries to the MN.

Emergency Queue support is optional. D\_NMT\_EmergencyQueueSize\_U32 indicates support (D\_NMT\_EmergencyQueueSize\_U32 > 0) and queue size.

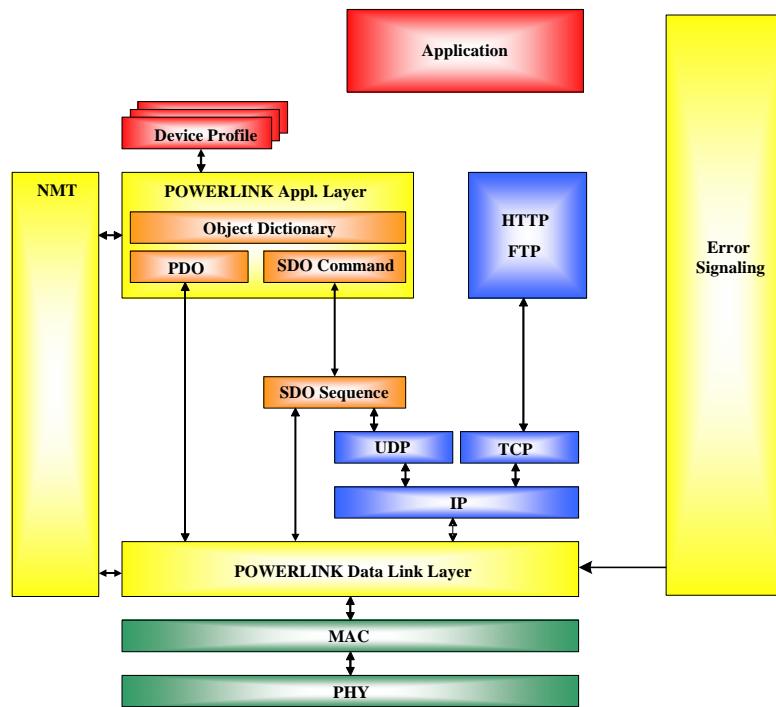


Fig. 67. Error signaling – Reference model

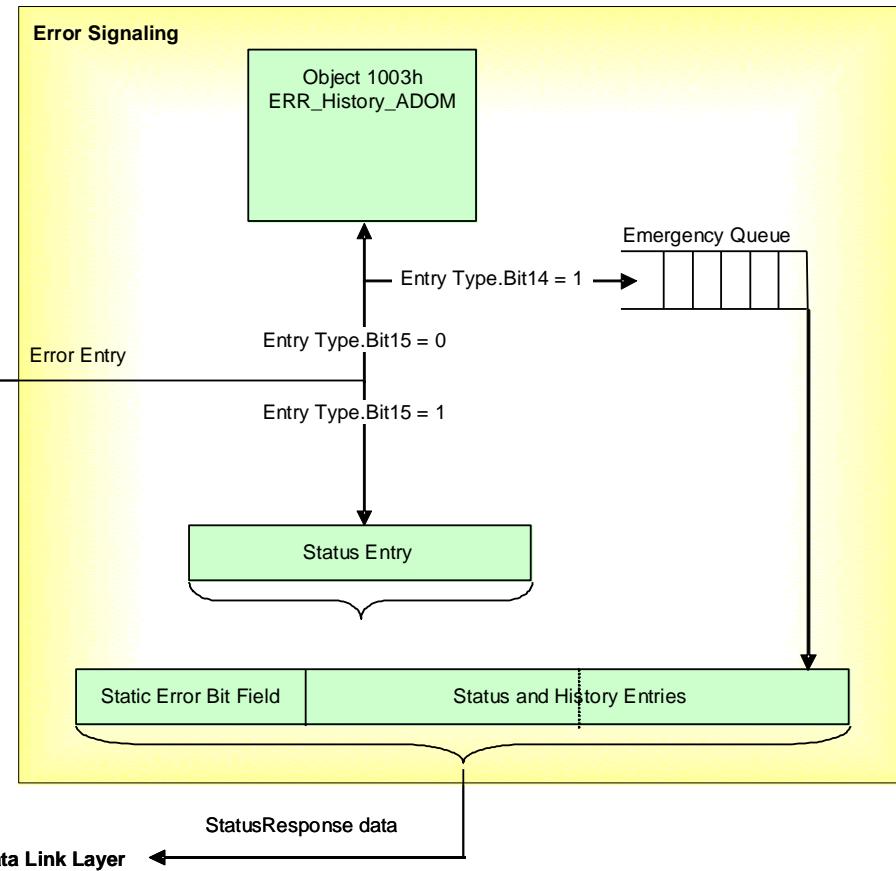


Fig. 68. Error signaling – Overview

## 6.5.1 Error Entry

This chapter describes the function and the data format of an Error Entry listed in the Error History object 1003h ERR\_History\_ADOM or the StatusResponse.

The Error History object holds the errors and events that have occurred on the device..

An ErrorEntry\_DOM has the following format:

| Octet Offset | Description            |
|--------------|------------------------|
| 0 .. 1       | Entry Type             |
| 2 .. 3       | Error Code             |
| 4 .. 11      | Time Stamp             |
| 12 .. 19     | Additional Information |

Tab. 94 Format of one entry

| Field                  | Abbr. | Description  | Value      |
|------------------------|-------|--|------------|
| Entry Type             | type  | see Tab. 96  | UNSIGNED16 |
| Error Code             | code  | Depending on the Entry Type the error codes are described in the device profiles , device descriptions or the communication profile. Communication profile specific error codes are described in App. 3.9. | UNSIGNED16 |
| Time Stamp             | time  | SoC Nettime from the cycle when the error/event was detected.  | UNSIGNED64 |
| Additional Information | add   | This field contains device profile or vendor specific additional error information.  | UNSIGNED64 |

Tab. 95 Description of one entry

All elements of the Error Entry shall be stored and transferred in little endian format.

| Octet  | Bit                  | Value                                | Description   |
|--------|----------------------|--------------------------------------|---|
| 0 .. 1 | 15<br>(status)       | 0 <sub>b</sub>                       | ERR_History_ADOM Entry  |
|        |                      | 1 <sub>b</sub>                       | Status Entry in StatusResponse frame (Bit 14 shall be set to 0 <sub>b</sub> )   |
|        | 14<br>(send)         | 0 <sub>b</sub>                       | ERR_History_ADOM only   |
|        |                      | 1 <sub>b</sub>                       | Additional to the ERR_History_ADOM the entry shall also be entered in to the Emergency Queue of the Error Signaling.  |
|        | 13 .. 12<br>(mode)   | 0 <sub>h</sub>                       | Not allowed in ERR_History_ADOM.<br>Entries with this mode may only be used by the Error Signaling itself to indicate the termination of the History Entries in the StatusResponse frame. |
|        |                      | 1 <sub>h</sub>                       | An error has occurred and is active (e.g. short circuit of output detected)   |
|        |                      | 2 <sub>h</sub>                       | An active error was cleared (e.g. no short circuit anymore)<br>(not allowed for Status Entries)   |
|        |                      | 3 <sub>h</sub>                       | An error / event occurred<br>(not allowed for Status Entries)   |
|        |                      |                                      |   |
|        | 11 .. 0<br>(profile) | 000 <sub>h</sub>                     | Reserved  |
|        |                      | 001 <sub>h</sub>                     | The field Error Code in Tab. 94 contains a vendor specific error code   |
|        |                      | 002 <sub>h</sub>                     | The field Error Code in Tab. 94 contains POWERLINK communication profile specific errors which are listed in App. 3.9 (Network errors, communication errors, data link errors ...)        |
|        |                      | 003 <sub>h</sub> .. FFF <sub>h</sub> | The field Error Code in Tab. 94 contains device profile specific errors.<br>e.g.<br>191h ... CiA 401, Device Profile for Generic IO Modules<br>192h ... CiA 402, Drive Profile            |

Tab. 96 Format of the field entry type

## 6.5.2 Interface to Error Signaling

Any layer of the MN or CN which intends to generate an Error Entry shall provide the following information to the Error Signaling:

- Entry Type (all fields shall be provided, see Tab. 96)
- Error Code (see Tab. 94)
- Additional Information (see Tab. 94)

## 6.5.3 Processing of CN Error Information on the MN

History or Status Entries received from CNs shall be passed to the application layer.

## 6.5.4 Error Signaling Bits

To avoid that the MN has to poll ERR\_History\_ADOM for changes, the following mechanism shall inform the MN when the Static Error Bit Field, the Status Entries or the History Entries of the CN StatusResponse frame have changed.

The following bits shall be used for a confirmed transmission from the CN to the MN:

| Field                 | Abbr. | Description   |
|-----------------------|-------|---|
| Exception Reset       | ER    | <p>Initialisation of the Error Signaling<br/>When a CN receives the value <math>1_b</math> it shall reset its EN bit to <math>0_b</math> and clear the Emergency Queue.<br/>The MN shall send the ER bit with the following frame:<br/>SoA( StatusRequest )</p>   |
| Exception Clear       | EC    | <p>In this bit the CN shall mirror the last received ER from the MN.<br/>This is required to indicate the MN that the initialisation of the Error Signaling was done.<br/>A CN shall send the EC bit with the following frame:<br/>ASnd( StatusResponse )</p>   |
| Exception New         | EN    | <p>By toggling this bit the CN informs the MN that the Static Error Bit Field or the Status Entries have changed or that new History Entries are available in the StatusResponse frame.<br/>A CN shall send the EN bit with the following frames:<br/>PRes<br/>ASnd( StatusResponse )<br/>For isochronous CNs the MN shall evaluate the EN bit of the ASnd(StatusResponse) only in NMT_CS_PRE_OPERATIONAL_1.</p>  |
| Exception Acknowledge | EA    | <p>When the MN detects that the last sent EA bit is different to the last received EN bit it shall send a StatusRequest frame to the CN. After the StatusResonse frame was successfully received by the MN, it shall set EA=EN.<br/>When the bit is transferred to the CN the next time, the CN knows that it may generate a new StatusResponse frame and toggle the EN bit again.<br/>The MN shall send the EA bit with the following frames:<br/>PReq<br/>SoA(StatusRequest )<br/>Isochronous CNs shall evaluate the EA bit of the SoA(...) only in NMT_CS_PRE_OPERATIONAL_1.</p> |

Tab. 97 Error signaling bits

A CN shall only evaluate the ER and EA flags of the respective SoA(...) frames when the RequestedServiceTarget is the CNs POWERLINK Node ID.

## 6.5.5 Initialisation

With this initialisation the MN shall prepare the CN for the Error Signaling.

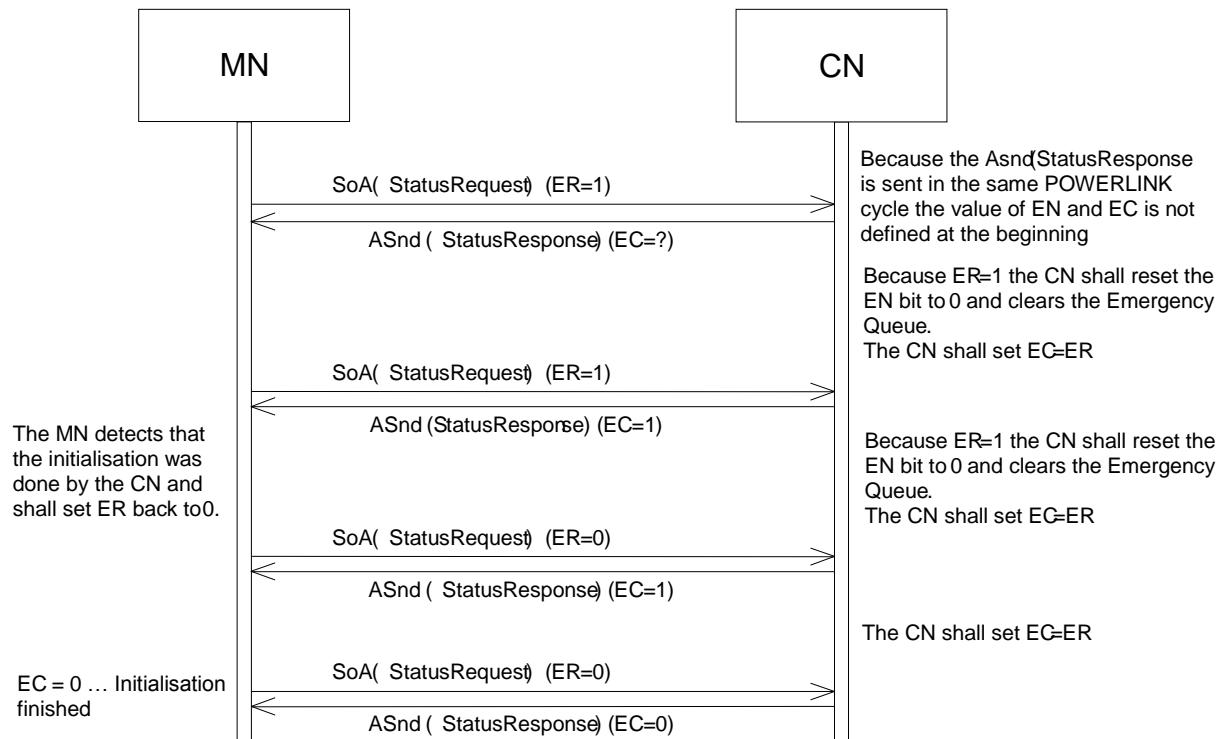


Fig. 69. Error signaling initialisation

### 6.5.5.1 Startup value and behaviour of the EC flag

At startup (NMT\_GT1, NMT\_GT2 or NMT\_GT8) the CN shall reset the Error Signaling and set EC=1.

The CN shall not change the EC flag before at least 1 valid frame with ER=1 was received.

This regulation is necessary for the following case :

- The MN completes the initialisation until ER=0, the CN is still in NMT\_CS\_PRE\_OPERATIONAL\_1
- The CN has a power fail and starts up again. The next received frame from the MN contains ER=0 ... Now the CN shall respond with EC=1 to show the MN that the Error Signaling is not initialised.
- The MN shall restart the initialization of the Error Signaling

Other NMT transitions shall not automatically initialise the Error Signaling in order to allow the MN to receive emergency messages even if the NMT state of the CN changes for e.g. to NMT\_GS\_INITIALISATION because of any reason.

The Error Signaling shall only be initialised by NMT\_GT1, NMT\_GT8, NMT\_GT2 or explicit Error Signaling reset by the MN (ER=1).

## 6.5.6 Error Signaling with PReq and PRes frames

For isochronous CNs only the PReq and PRes frames shall be used for the Error Signaling.

If an isochronous CN is in the state NMT\_CS\_PRE\_OPERATIONAL\_1 the Error Signaling shall behave like an Async-only CN (6.5.7)

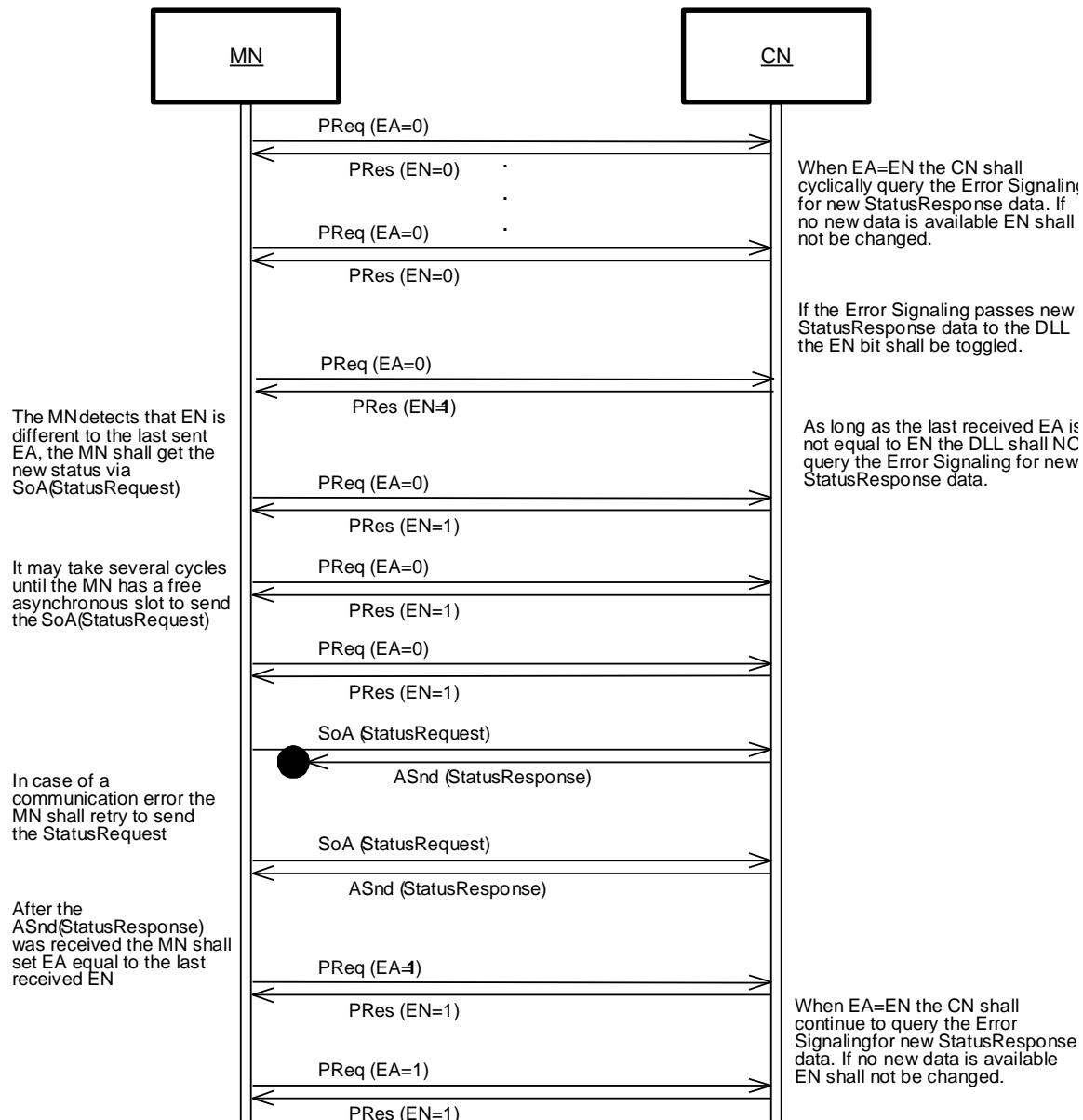


Fig. 70. Error signaling with PReq and PRes

## 6.5.7 Error Signaling with Async-only CNs

Async only CNs are periodically requested by the MN.

This shall be also done for isochronous CNs in NMT\_CS\_PRE\_OPERATIONAL\_1.

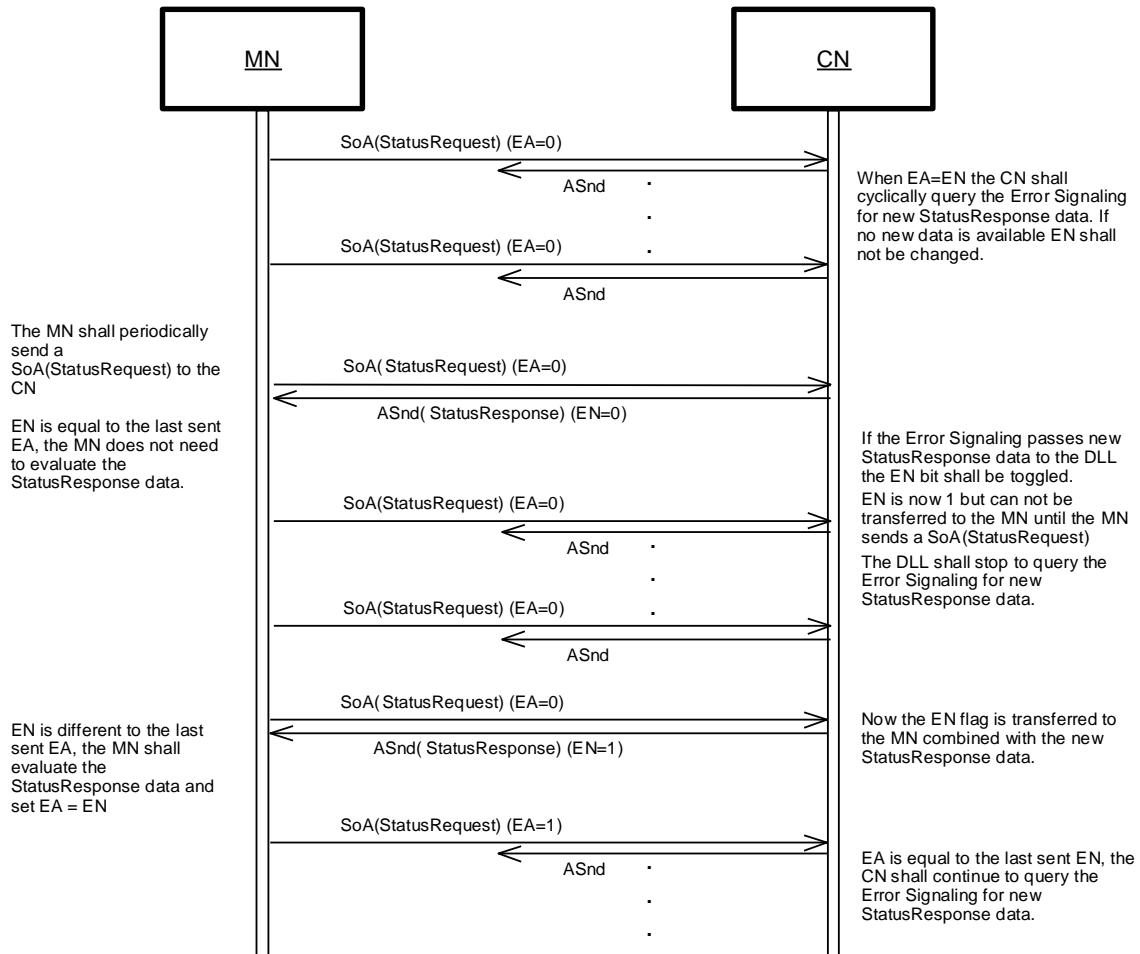


Fig. 71. Error signaling for Async-only CNs and CNs in state NMT\_CS\_PRE\_OPERATIONAL\_1

## 6.5.8 Format of StatusResponse Data

Refer to 7.3.3.3.1 for StatusResponse frame structure.

### 6.5.8.1 Static Error Bit Field

| Octet Offset <sup>20</sup> | Description                              |
|----------------------------|--|
| 6                          | Content of ERR_ErrorRegister_U8          |
| 7                          | Reserved                                 |
| 8 .. 13                    | Device profile or vendor specific errors |

Tab. 98 Static error bit field

### 6.5.8.2 Status and History Entries

If not all available entries of the StatusResponse are used, an entry with Mode 0 (see Tab. 96) shall be used to terminate the History and Status Entry list and declare this and all following entries as unused.

The CN may also change the length of the StatusResponse frame to fit the required Status and History fields. In this case no entry for termination is required.

<sup>20</sup> Byte Offset relates to the beginning of the ASnd service slot (See 7.3.3.3.1)

The number of used Status / History Entries is defined by D\_NMT\_ErrorEntries\_U32 .

Status Entries shall be located in front of the History Entries inside the StatusResponse frame.

## 6.5.9 Examples

The examples in this chapter are an implementation proposal for the creation of the StatusResponse frames on the CN.

Another buffer handling may be implemented in consideration of the following points:

- The CN shall notify the MN about changes of the StatusResponse frame using the EN bit as shown in 6.5.6 or 6.5.7
- After the CN has indicated new StatusResponse data to the MN by toggling the EN flag, the StatusResponse frame on the CN may not be changed anymore as it can be requested by the MN at any time. Only if the reception of the response is acknowledged by the EA flag of the MN, the CN may change the StatusResponse frame data again.

The following examples make use of the following abbreviations:

| Abbreviation | Description  |
|--------------|--|
| BF           | Static Error Bit Field   |
| S            | Bit 15 (status) of the Entry Type (see Tab. 94 Format of one entry)  |
| M            | Bit 13-12 (mode) of the Entry Type (see Tab. 94 Format of one entry) |
| C            | Error Code (see Tab. 94 Format of one entry)                         |

Tab. 99 Abbreviations for the following examples

The Octet Offsets relates to the beginning of the Ethernet frame.

### 6.5.9.1 Case 1 – Only Bit Field, No Status/History Entries

This is the most simple implementation of the Error Signaling. Only the Static Error Bit Field is used to signal errors from the CN to the MN.

The length of the StatusResponse frame is fixed and does not change during runtime:

| Octet Offset | Description            |
|--------------|------------------------|
| 0 .. 23      | Headers                |
| 24 .. 31     | Static Error Bit Field |
| 32 .. 51     | Entry 0 (M=0)          |
| 52 .. 71     | Entry 1 (M=0)          |
| 72 .. 75     | Ethernet CRC           |

| Field                  | Description   |
|------------------------|---|
| Headers                | Ethernet / POWERLINK / ASnd(StatusResponse) Header (see 7.3.3.3.1)  |
| Static Error Bit Field | The bit field containing the status information (see 6.5.8.1)   |
| Entry 0, Entry 1       | For this example no Status Entry is required. Since the frame must have a minimum length the first two entries can not be removed by shortening the frame. To avoid that the MN processes this data, the field Mode of the Entry Type is 0. |

At startup the Error Signaling shall generate 2 identical empty StatusResponse Frames and pass one of them to the Data Link Layer as initial StatusResponse data. A third frame shall be generated for further use.

This frame of the Data Link Layer may be requested by the MN at any time.

- Frame 1: Owner = Error Signaling (May be changed at any time)
- Frame 2: Owner = Data Link Layer
- Frame 3: Owner = Error Signaling (unused until Frame 1 is changed)

If the Static Error Bit Field has changed, the value is copied to frame 3 which will be passed to the Data Link Layer:

- Frame 1: Owner = Error Signaling (May be changed at any time)
- Frame 2: Owner = Data Link Layer (DLL)

- Frame 3: Owner = Error Signaling (Copy of the changed Frame 1, ready to be taken from the DLL)

Now the Error Signaling is not allowed to change this Frame 3 because it can be taken from the Data Link Layer at any time.

The next time the DLL queries for new data the Error Signaling passes Frame 3 to the DLL which shall set the EN flag to signal the MN that new data is available. Frame 2 shall be passed back to the Error Signaling for further use:

- Frame 1: Owner = Error Signaling (May be changed at any time)
- Frame 2: Owner = Error Signaling (unused until Frame 1 is changed)
- Frame 3: Owner = Data Link Layer (DLL)

If the Error Signaling has changed Frame 1 before Frame 2 was passed back it shall immediately create a copy of Frame 1, otherwise it shall wait for the next change of Frame 1 and then create the copy.

- Frame 1: Owner = Error Signaling (May be changed at any time)
- Frame 2: Owner = Error Signaling (Copy of the changed Frame 1, ready to be taken from the DLL)
- Frame 3: Owner = Data Link Layer (DLL)

The sequence continues in this manner and Frame 2/Frame 3 are passed alternatively to the DLL for transmission to the MN. The DLL shall only query the Error Signaling for the next frame if EN=EA, that means the MN has successfully picked up the last changed frame.

### 6.5.9.2 Case 2 – Status Entries

In this example the CNs are allowed to signal errors using the Static Error Bit Field and/or the Status Entries of the StatusResponse frame.

The frame sequence is the same as in Example 1. Every time the Error Signaling makes changes of the StatusResponse frame content (Static Error Bit Field or in one of the Status Entries) it shall provide a new frame for the DLL which will be 1 as soon as EN=EA.

Example with 4 Status Entries:

| Octet Offset | Description            |
|--------------|------------------------|
| 0 .. 23      | Headers                |
| 24 .. 31     | Static Error Bit Field |
| 32 .. 51     | Status Entry 0         |
| 52 .. 71     | Status Entry 1         |
| 72 .. 91     | Status Entry 2         |
| 92 .. 111    | Status Entry 3         |
| 112 .. 115   | Ethernet CRC           |

| Field               | Description  |
|---------------------|--|
| Status Entry 0 .. 3 | <p>S=X,M=0<br/>This entry and all following entries are not valid</p> <p>S=1,M=1,C=0<br/>The entry does not contain status information but the next entry is valid</p> <p>S=1,M=1,C&gt;0<br/>The entry contains valid status information</p> |

If all 4 status entries contain status information and the CN wants to remove Status Entry 1 and 2 it has two ways to do so.

- Set C of Entry 1 and Entry 2 to 0 to invalidate the entries
- Copy Entry 3 to Entry 1 and set M of Entry 2 to 0 to terminate the list
- Copy Entry 3 to Entry 1 and shorten the frame size

### 6.5.9.3 Case 3 – History Entries

The CN sends History Entries or static errors to the MN. The MN shall only process the received History Entries if EN is not equal EA so each entry will be processed only once.

The initial frame shall not contain valid Status or History Entries.

| Octet Offset | Description            |
|--------------|------------------------|
| 0 .. 23      | Headers                |
| 24 .. 31     | Static Error Bit Field |
| 32 .. 51     | Entry 0 (M=0)          |
| 52 .. 59     | Padding                |
| 60 .. 63     | Ethernet CRC           |

Initial frames at startup:

- Frame 1: Owner = Error Signaling (May be changed at any time)
- Frame 2: Owner = Data Link Layer
- Frame 3: Owner = Error Signaling (unused until Frame 1 is changed)

As in Example 1, the CN may change the Static Error Bit Field and perform the same sequence as in Example 1 to signal the change to the MN.

Additionally the Error Signaling shall check periodically if the Emergency Queue (see Fig. 68 Error signaling – Overview) contains any entries. In this case the Error Signaling also shall create a copy of Frame 1 and append as many History Entries from the Emergency Queue to the frame as possible and remove these entries from the queue.

Frame 1 never contains History Entries. They shall be only added to the frame passed to the DLL !

Assume the Emergency Queue contains 10 History Entries and the length of the Status Response is limited to a maximum of 150 byte, the next frame prepared for the DLL will be the following:

| Octet Offset | Description            |
|--------------|------------------------|
| 0 .. 23      | Headers                |
| 24 .. 31     | Static Error Bit Field |
| 32 .. 51     | History Entry 0        |
| 52 .. 71     | History Entry 1        |
| 72 .. 91     | History Entry 2        |
| 92 .. 111    | History Entry 3        |
| 112 .. 131   | History Entry 4        |
| 132 .. 135   | Ethernet CRC           |

The frame will transport 5 of the 10 queue entries to the MN. The next time the Error Signaling shall transfer the rest of the entries in the queue (but always maximum 5 entries in this case). The sequence will continue in this manner until the Emergency Queue is empty.

### 6.5.9.4 Case 4 – Status and History Entries

The CN may also send StatusResponse frames containing both Status and History Entries. In this case the History Entries shall be located in the frame after the Status Entries. As shown in the examples above the Status Entries contain static information and the History Entries are taken from the Emergency Queue.

The MN shall be aware that the number of Status Entries and History Entries as well as the frame length of the StatusResponse can vary at every frame.

## 6.5.10 Object descriptions

### 6.5.10.1 Object 1001h : ERR\_ErrorRegister\_U8

The object ERR\_ErrorRegister\_U8 is compatible to the object ‘error register’ of the standard communication profile CiA DS 301.

|               |                      |             |     |
|---------------|----------------------|-------------|-----|
| Index         | 1001 <sub>h</sub>    | Object Type | VAR |
| Name          | ERR_ErrorRegister_U8 |             |     |
| Data Type     | UNSIGNED8            | Category    | M   |
| Value Range   | 0..255               | Access      | ro  |
| Default Value | 0                    | PDO Mapping | Opt |

- **Value Interpretation**

| Bit | M/O | Description   |
|-----|-----|---|
| 0   | M   | Generic error<br>This bit shall be set to 1 <sub>b</sub> if the Static Error Bit Field or the Status Entries in the StatusResponse frame show one or more errors.<br>If this bit is 0 <sub>b</sub> the MN only needs to evaluate the History Entries of the StatusResponse frame. |
| 1   | O   | Current   |
| 2   | O   | Voltage   |
| 3   | O   | Temperature   |
| 4   | O   | Communication error   |
| 5   | O   | Device profile specific   |
| 6   | O   | Reserved (always 0)   |
| 7   | O   | Manufacturer specific   |

### 6.5.10.2 Object 1003<sub>h</sub>: ERR\_History\_ADOM

Sub-index 0 contains the number of actual errors/events that are recorded in the array starting at sub-index 1. Every new error/event is stored at sub-index 1, the older ones move down the list.

Object ERR\_History\_ADOM shall not be cleared upon initialisation during an NMTResetConfiguration, NMTResetCommunication or NMTResetNode.

*Note: This facilitates reading the error history after the CN performed an NMTReset following an error.*

|           |                   |             |       |
|-----------|-------------------|-------------|-------|
| Index     | 1003 <sub>h</sub> | Object Type | ARRAY |
| Name      | ERR_History_ADOM  |             |       |
| Data Type | DOMAIN            | Category    | O     |

- **Sub-index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub> |             |     |
| Name          | NumberOfEntries |             |     |
| Value Range   | 0 .. 254        | Access      | rw  |
| Default Value | 0               | PDO Mapping | Opt |

When writing to this sub-index only value 0 is allowed (clear history). All other values will be rejected with an error code.

- **Sub-index 01<sub>h</sub> – FE<sub>h</sub> : ErrorEntry\_DOM**

|               |                                   |             |    |
|---------------|-----------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |    |
| Name          | ErrorEntry_DOM                    |             |    |
| --            | --                                | Category    | O  |
| Value Range   | see Tab. 94                       | Access      | ro |
| Default Value | -                                 | PDO Mapping | No |

## 6.6 Program Download

In this chapter, a common method for downloading a program to a device via its object dictionary is specified. Here only the mechanism for performing the program download is specified, not the structure of program data or the data structure. The specified mechanism can be used for downloading complete programs to devices (e.g. if a device only provides a kind of POWERLINK

bootstrap loader) or only parts of a program (e.g. specific tasks of real-time systems). The data structure of the transferred program data must be specified by the manufacturer (e.g. INTEL-HEX format or binary format).

Further specifications for the program download are given in specific device profiles (e.g. in CiA DS-405 for the download of PLC programs).

Programmability of a CN is indicated by NMT\_NodeAssignment\_AU32[sub-index] Bit 6.

## 6.6.1 Object Dictionary Entries on the CN

### 6.6.1.1 Object 1F50<sub>h</sub>: PDL\_DownloadProgData\_ADOM

PDL\_DownloadProgData\_ADOM holds downloaded programs. It allows the access to up to 254 programs.

|           |                           |             |       |
|-----------|---------------------------|-------------|-------|
| Index     | 1F50 <sub>h</sub>         | Object Type | ARRAY |
| Name      | PDL_DownloadProgData_ADOM |             |       |
| Data Type | DOMAIN                    | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                    |             |    |
|---------------|------------------------------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub>                    |             |    |
| Name          | NumberOfEntries                    |             |    |
| Value Range   | 00 <sub>h</sub> .. FE <sub>h</sub> | Access      | rw |
| Default Value | -                                  | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub>: Program**

|               |                 |             |      |
|---------------|-----------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> |             |      |
| Name          | Program         |             |      |
| --            | --              | Category    | Cond |
| Value Range   | DOMAIN          | Access      | cond |
| Default Value | -               | PDO Mapping | No   |

Sub-index 1 is the program access point of a device that is reserved for the firmware of the device holding the device communication profile. Therefore no common program download to this sub-index is allowed.

If the download fails, the device responds with an Abort SDO Transfer (error code 0606 0000<sub>h</sub>).

If the device does not support firmware download via SDO Write by Index this sub-index shall not be present.

*Note: The firmware may also be downloaded to a device by FTP or SDO File Write transfer.*

If the device supports reading of the program the access shall be rw otherwise wo.

- **Sub-Index 02<sub>h</sub> .. FE<sub>h</sub>: Program**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 02 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | Program                            |             |      |
| --            | --                                 | Category    | O    |
| Value Range   | DOMAIN                             | Access      | cond |
| Default Value | -                                  | PDO Mapping | No   |

Program sub-indices provide access to additional programs in a device.

If the device supports reading of the program the access shall be rw otherwise wo.

### 6.6.1.2 Object 1F51<sub>h</sub>: PDL\_ProgCtrl\_AU8

PDL\_ProgCtrl\_AU8 is specified for controlling the execution of stored programs:

|           |                   |             |       |
|-----------|-------------------|-------------|-------|
| Index     | 1F51 <sub>h</sub> | Object Type | ARRAY |
| Name      | PDL_ProgCtrl_AU8  |             |       |
| Data Type | UNSIGNED8         | Category    | Cond  |

PDL\_ProgCtrl\_AU8 shall be implemented if PDL\_DownloadProgData\_ADOM is supported by the device.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                    |             |    |
|---------------|------------------------------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub>                    |             |    |
| Name          | NumberOfEntries                    |             |    |
| Value Range   | 01 <sub>h</sub> .. FE <sub>h</sub> | Access      | rw |
| Default Value | 01 <sub>h</sub>                    | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub>: ProgCtrl**

|               |                         |             |    |
|---------------|-------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>         |             |    |
| Name          | ProgCtrl                |             |    |
| --            | --                      | Category    | M  |
| Value Range   | 0 .. 2 (refer Tab. 100) | Access      | rw |
| Default Value | 0                       | PDO Mapping | No |

ProgCtrl controls the program holding the device communication profile to be accessed by PDL\_DownloadProgData\_ADOM[1].

On write access program execution will be commanded, on read access the current execution state of the program may be queried. The following values are defined:

|   | Write Access  | Read Access     |
|---|---------------|-----------------|
| 0 | stop program  | program stopped |
| 1 | start program | program running |
| 2 | reset program | program stopped |

Tab. 100 PDL\_ProgCtrl\_AU8 sub-index value interpretation

If the action is not possible, the device shall respond with an Abort SDO Transfer (error code 0800 0024<sub>h</sub>).

- **Sub-Index 02<sub>h</sub> .. FE<sub>h</sub>: ProgCtrl**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 02 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | ProgCtrl                           |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | 0 .. 2 (refer Tab. 100)            | Access      | rw   |
| Default Value | -                                  | PDO Mapping | No   |

ProgCtrl sub-indices control the additional programs to be accessed via the PDL\_DownloadProgData\_ADOM sub-indices 02<sub>h</sub> .. FE<sub>h</sub>. The respective sub-index shall be implemented if the corresponding sub-index of PDL\_DownloadProgData\_ADOM is provided by the device.

### 6.6.1.3 Object 1F52<sub>h</sub>: PDL\_LocVerAppISw\_REC

PDL\_LocVerAppISw\_REC is defined to support verification of the version of the program holding the device communication profile to be accessed via PDL\_DownloadProgData\_ADOM[1]<sup>21</sup>.

<sup>21</sup> Note that the object NMT\_ManufactDevName\_VS can be regarded as a version number of a fixed program of a non-programmable node or as a firmware (like boot block and operating system) version number of a programmable node. Hence, a separate object for re-programmable application software is defined.

|           |                       |             |        |
|-----------|-----------------------|-------------|--------|
| Index     | 1F52 <sub>h</sub>     | Object Type | RECORD |
| Name      | PDL_LocVerAppISw_REC  |             |        |
| Data Type | PDL_LocVerAppISw_TYPE | Category    | Cond   |

PDL\_LocVerAppISw\_REC shall be implemented if PDL\_DownloadProgData\_ADOM is supported by the device.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2               | Access      | const |
| Default Value | 2               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: ApplSwDate\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> |             |    |
| Name          | ApplSwDate_U32  |             |    |
| Data Type     | UNSIGNED32      | Category    | M  |
| Value Range   | UNSIGNED32      | Access      | rw |
| Default Value | -               | PDO Mapping | No |

ApplSwDate\_U32 shall contain the number of days since January 1, 1984.

- **Sub-Index 02<sub>h</sub>: ApplSwTime\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub> |             |    |
| Name          | ApplSwTime_U32  |             |    |
| Data Type     | UNSIGNED32      | Category    | M  |
| Value Range   | UNSIGNED32      | Access      | rw |
| Default Value | -               | PDO Mapping | No |

ApplSwTime\_U32 shall contain the number of ms since midnight.

*Note that the date and time are only supported for the default application program. Dates and times of additional programs 2 to 254 are not supported. In case of two or more programs, one possibility could be to store the latest update time and date of any of the programs 2 to 254 in the objects PDL\_MnExpAppSwDateList\_AU32 and PDL\_MnExpAppSwTimeList\_AU32.*

#### 6.6.1.4 Object 0427<sub>h</sub>: PDL\_LocVerAppISw\_TYPE

|           |                       |             |            |
|-----------|-----------------------|-------------|------------|
| Index     | 0427 <sub>h</sub>     | Object Type | DEFSTRUCT  |
| Name      | PDL_LocVerAppISw_TYPE |             |            |
| Sub-Index | Component Name        | Value       | Data Type  |
| 00h       | NumberOfEntries       | 02h         |            |
| 01h       | ApplSwDate_U32        | 0007h       | UNSIGNED32 |
| 02h       | ApplSwTime_U32        | 0007h       | UNSIGNED32 |

#### 6.6.2 Object Dictionary Entries on the MN

Two MN located objects are specified for verification of the version of the application software at the CNs.

In programmable devices the objects shall be compared to PDL\_LocVerAppISw\_REC reflecting the version of the downloaded program holding the device communication profile accessible via PDL\_DownloadProgData\_ADOM[1].

In non-programmable devices the objects may be used to store verification data to be compared to NMT\_ManufactSwVers\_VS.

### 6.6.2.1 Object 1F53<sub>h</sub>: PDL\_MnExpAppSwDateList\_AU32

PDL\_MnExpAppSwDateList\_AU32 holds the expected Application SW date to be compared to the actual value on the CN in order to verify the Application SW.

|           |                             |             |       |
|-----------|-----------------------------|-------------|-------|
| Index     | 1F53 <sub>h</sub>           | Object Type | ARRAY |
| Name      | PDL_MnExpAppSwDateList_AU32 |             |       |
| Data Type | UNSIGNED32                  | Category    | Cond  |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub>                    |             |     |
| Name          | NumberOfEntries                    |             |     |
| Value Range   | 01 <sub>h</sub> .. FE <sub>h</sub> | Access      | rws |
| Default Value | FE <sub>h</sub>                    | PDO Mapping | No  |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: AppSwDate**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |     |
| Name          | AppSwDate                          |             |     |
| --            | --                                 | Category    | O   |
| Value Range   | UNSIGNED32                         | Access      | rws |
| Default Value | 0                                  | PDO Mapping | No  |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 5.

AppSwDate contains the number of days since January 1, 1984.

PDL\_MnExpAppSwDateList\_AU32 shall be implemented by MNs if the MN supports program verification.

PDL\_MnExpAppSwDateList\_AU32 may be implemented by CNs to hold latest update date of additional programs to be accessed via PDL\_DownloadProgData\_ADOM sub-indices 02<sub>h</sub> .. FE<sub>h</sub>.

### 6.6.2.2 Object 1F54<sub>h</sub>: PDL\_MnExpAppSwTimeList\_AU32

PDL\_MnExpAppSwTimeList\_AU32 holds the expected Application SW time to be compared to the actual value on the CN in order to verify the Application SW.

|           |                             |             |       |
|-----------|-----------------------------|-------------|-------|
| Index     | 1F54 <sub>h</sub>           | Object Type | ARRAY |
| Name      | PDL_MnExpAppSwTimeList_AU32 |             |       |
| Data Type | UNSIGNED32                  | Category    | Cond  |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub>                    |             |     |
| Name          | NumberOfEntries                    |             |     |
| Value Range   | 01 <sub>h</sub> .. FE <sub>h</sub> | Access      | rws |
| Default Value | FE <sub>h</sub>                    | PDO Mapping | No  |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: AppSwTime**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |     |
| Name          | AppSwTime                          |             |     |
| --            | --                                 | Category    | O   |
| Value Range   | UNSIGNED32                         | Access      | rws |
| Default Value | 0                                  | PDO Mapping | No  |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 5.

AppSwTime contains the number of milliseconds after midnight (00:00).

PDL\_MnExpAppSwTimeList\_AU32 shall be implemented by MNs if the MN supports program verification.

PDL\_MnExpAppSwTimeList\_AU32 may be implemented by CNs to hold latest update time of additional programs to be accessed via PDL\_DownloadProgData\_ADOM sub-indices 02<sub>h</sub> .. FE<sub>h</sub>.

## 6.7 Configuration Management

The configuration of a device may be stored locally on the device in a non-volatile memory or centrally on the MN performing the Configuration Manager function.

The Configuration Manager is an optional application residing on the MN. It may configure network devices at network boot-up. Support of Configuration Manager functions shall be indicated by device description entry D\_CFM\_ConfigManager\_BOOL.

For this it has to know the (application dependent) parameter values. This information may be provided by a Device Configuration File for each device. This file is stored on the MN. In order to reduce memory consumption Concise Configuration Storage may be applied.

POWERLINK defines a XML based Device Description (XDD) and Device Configuration (XDC) file format. The file format is described in detail in a separate paper (POWERLINK XML Specification).

### 6.7.1 Device Description

The Device Description File provides information about the device's basic communication and functional properties.

POWERLINK employs XDD Files for Device Description. These files shall be provided by the manufacturer of the device.

XDD files may be stored locally on the device or centrally on the MN.

#### 6.7.1.1 Local Storage on the Device

With some devices it may be possible to store the Device Description File (XDD) locally on the device. Local XDD File storage has some advantages:

- The manufacturer does not have the problem of distributing the XDD File via data carrier.
- Management of different XDD File versions for different software versions is less error prone, if they are stored together.
- The complete network settings may be stored on the network. This makes the task of analyzing or reconfiguring a network easier for tools and more transparent for the users.

The object CFM\_StoreDevDescrFile\_DOM contains the XDD File. The format of the file is described by object CFM\_StoreDevDescrFormat\_U16.

The XDD File is downloaded from a Configuration Tool to the device by writing the file as a domain to CFM\_StoreDevDescrFile\_DOM. The file is uploaded by reading CFM\_StoreDevDescrFile\_DOM.

#### 6.7.1.2 Central Storage on the MN

For those devices which are not able to store their XDD Files, the Configuration Manager on the MN may take over this task. The objects CFM\_StoreDevDescrFileList\_ADOM and CFM\_DevDescrFileFormatList\_AU8 are defined at the Configuration Manager to hold the file information.

The device XDD File is downloaded from a Configuration Tool to the Configuration Manager by writing the file as a domain to CFM\_StoreDevDescrFileList\_ADOM with the sub-index equal to the device's Node ID.

The file is uploaded to a tool by reading CFM\_StoreDevDescrFileList\_ADOM with the sub-index equal to the device's Node ID.

### 6.7.2 Device Configuration Storage

Device Configuration provides application specific setup values.

### 6.7.2.1 Device Configuration File Storage

Centralized storage of device configuration is performed by the Configuration Manager on the MN. Data is stored in the form of a Device Configuration File (XDC).

XDC Storage is managed by the objects CFM\_StoreDcfList\_ADOM and CFM\_DcfStorageFormatList\_AU8. CFM\_StoreDcfList\_ADOM holds the XDC Files and CFM\_DcfStorageFormatList\_AU8 provides information about the files' storage format. Each sub-index of the ARRAYS points to the Node ID of the device to which the XDC belongs.

The XDC is downloaded from a Configuration Tool to the Configuration Manager by writing the XDC File as a domain to CFM\_StoreDcfList\_ADOM with the sub-index equal to the device's Node ID.

The XDC is uploaded from the Configuration Manager to a Tool by reading the object CFM\_StoreDcfList\_ADOM with the sub-index equal to the device's Node ID.

### 6.7.2.2 Concise Configuration Storage

The Concise Device Configuration does not contain all of the information contained in the XDC. The information to be stored consists of the parameter values of the object dictionary entries. The information is stored by a stream assigning data values to the particular objects and sub-indices. (See Tab. 102)

It is recommended that Concise Configuration Storage is used when complete XDC storage is not possible.

The concise configuration of a device is downloaded from a Configuration Tool to the Configuration Manager by writing the stream as a domain to CFM\_ConciseDcfList\_ADOM with the sub-index equal to the device's Node ID.

The concise DCF of a device is uploaded from the Configuration Manager to a Tool by reading the object CFM\_ConciseDcfList\_ADOM with the sub-index equal to the device's Node ID.

### 6.7.2.3 Check Configuration Process

POWERLINK defines the object CFM\_VerifyConfiguration\_REC. If a device supports saving of parameters in non-volatile memory, the MN or a configuration tool may use this object to verify the configuration after a device is reset and to check whether a reconfiguration is necessary. The configuration tool shall store configuration date, time and ID in CFM\_VerifyConfiguration\_REC and shall store the same values in the respective sub-indices of CFM\_ExpConfDateList\_AU32, CFM\_ExpConfTimeList\_AU32 and CFM\_ExpConfIdList\_AU32.

Now the configuration tool orders the device to save its configuration by writing the signature "save" to NMT\_StoreParam\_REC.AllParam\_U32.

After a reset the device restores the last configuration and the signature automatically or by request.

The Configuration Manager compares the signature in CFM\_VerifyConfiguration\_REC to the respective sub-indices of CFM\_ExpConfDateList\_AU32, CFM\_ExpConfTimeList\_AU32 and/or CFM\_ExpConfIdList\_AU32 and decides if a reconfiguration is necessary or not.

### 6.7.2.4 Request Configuration

In some applications there might be situations where it is necessary to configure CNs during run-time – for example, if a CN fails and re-boots. The NMT master will recognize this and inform the application (see chapter 7.4.2.2.1.2). With the object CFM\_ConfCNRequest\_AU32 the NMT master application is able to tell the Configuration Manager to configure that CN.

Another example is the connection of a new machine part with several devices. The application needs the ability to start the Configuration Manager at least for the new nodes.

Configuration may be also requested by the target CN itself or by a tool residing on another CN.

To initiate the configuration process of a single CN, the requester writes the signature "conf" to the sub-index equal to the CN's Node ID.

A SDO Write Request to sub-index FF<sub>h</sub> is used to start an overall re-configuration of all CNs on the system. This function enables simple applications to request the Configuration Management without knowing the actual project configuration.

## 6.7.3 Object Dictionary Entries

### 6.7.3.1 Object 1020<sub>h</sub>: CFM\_VerifyConfiguration\_REC

CFM\_VerifyConfiguration\_REC holds device local Configuration date and time.

|           |                              |             |        |
|-----------|------------------------------|-------------|--------|
| Index     | 1020 <sub>h</sub>            | Object Type | RECORD |
| Name      | CFM_VerifyConfiguration_REC  |             |        |
| Data Type | CFM_VerifyConfiguration_TYPE | Category    | M      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2 .. 4          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: ConfDate\_U32**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> |             |                     |
| Name          | ConfDate_U32    |             |                     |
| Data Type     | UNSIGNED32      | Category    | M                   |
| Value Range   | UNSIGNED32      | Access      | rws, valid on reset |
| Default Value | 0               | PDO Mapping | No                  |

ConfDate\_U32 holds the local configuration date. It contains the number of days since January 1, 1984.

- **Sub-Index 02<sub>h</sub>: ConfTime\_U32**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 02 <sub>h</sub> |             |                     |
| Name          | ConfTime_U32    |             |                     |
| Data Type     | UNSIGNED32      | Category    | M                   |
| Value Range   | UNSIGNED32      | Access      | rws, valid on reset |
| Default Value | 0               | PDO Mapping | No                  |

ConfTime\_U32 holds the local configuration time. It contains the number of ms since midnight.

- **Sub-Index 03<sub>h</sub>: Confld\_U32**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Confld_U32      |             |     |
| Data Type     | UNSIGNED32      | Category    | O   |
| Value Range   | UNSIGNED32      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

Confld\_U32 holds a configuration ID value.

The value shall be created by a configuration tool in a manufacturer specific way.

In a POWERLINK network it should be identical only on those nodes, that have an identical HW and configuration besides some node specific objects like POWERLINK Node ID (NMT\_EPLNodeID\_REC.NodeID\_U8), SerialNumber (NMT\_IdentityObject\_REC.SerialNo\_U32) etc. Otherwise Confld\_U32 should be unique for each node in an Ethernet POWERLINK network segment.

- **Sub-Index 04<sub>h</sub>: VerifyConflInvalid\_BOOL**

|               |                         |             |    |
|---------------|-------------------------|-------------|----|
| Sub-Index     | 04 <sub>h</sub>         |             |    |
| Name          | VerifyConflInvalid_BOOL |             |    |
| Data Type     | BOOL                    | Category    | O  |
| Value Range   | BOOL                    | Access      | ro |
| Default Value | TRUE                    | PDO Mapping | No |

VerifyConflInvalid\_BOOL enables temporary local modifications of configuration parameters for test purpose while maintaining the bootability of the network.

VerifyConflInvalid\_BOOL = FALSE indicates that the configuration was not modified since the last storage of Confld\_U32 (sub-index 03<sub>h</sub>).

A change of a parameter defined in this specification which is stored in permanent memory shall set VerifyConflInvalid\_BOOL to TRUE. It may be set also by the application.

VerifyConflInvalid\_BOOL shall be set to FALSE upon writing a value > 0 to Confld\_U32.

A configuration tool or an application may use this information to display a warning if the configuration of a node was modified.

### 6.7.3.2 Object 1021<sub>h</sub>: CFM\_StoreDevDescrFile\_DOM

CFM\_StoreDevDescrFile\_DOM holds the device local Device Description File.

|               |                           |             |           |
|---------------|---------------------------|-------------|-----------|
| Index         | 1021 <sub>h</sub>         | Object Type | VAR       |
| Name          | CFM_StoreDevDescrFile_DOM |             |           |
| Data Type     | DOMAIN                    | Category    | O         |
| Value Range   | DOMAIN                    | Access      | ro or rws |
| Default Value | -                         | PDO Mapping | No        |

The object is read-only if the Device Description File stored by CFM\_StoreDevDescrFile\_DOM is unchangable.

Otherwise a Device Description File may be downloaded from a configuration tool by writing the file as a domain to CFM\_StoreDevDescrFile\_DOM. Reading the object uploads to the tool.

The availability of a Device Description File at CFM\_StoreDevDescrFile\_DOM is indicated by CFM\_StoreDevDescrFormat\_U16. If no data had been stored

(CFM\_StoreDevDescrFormat\_U16 = FF<sub>h</sub>), a SDO Read Request to

CFM\_StoreDevDescrFile\_DOM is aborted with the error code E\_CFM\_DATA\_SET\_EMPTY.

### 6.7.3.3 Object 1022<sub>h</sub>: CFM\_StoreDevDescrFormat\_U16

CFM\_StoreDevDescrFormat\_U16 holds the format of the Device Description File stored by CFM\_StoreDevDescrFile\_DOM.

|               |                             |             |           |
|---------------|-----------------------------|-------------|-----------|
| Index         | 1022 <sub>h</sub>           | Object Type | VAR       |
| Name          | CFM_StoreDevDescrFormat_U16 |             |           |
| Data Type     | UNSIGNED16                  | Category    | Cond      |
| Value Range   | refer Tab. 101              | Access      | ro or rws |
| Default Value | FF <sub>h</sub>             | PDO Mapping | No        |

The object shall be implemented if CFM\_StoreDevDescrFile\_DOM is implemented.

The object is read-only if the Device Description File stored by CFM\_StoreDevDescrFile\_DOM is unchangable.

CFM\_StoreDevDescrFormat\_U16 describes the format of the storage. This allows the use of compressed formats. The device may always store the file in a compressed format internally.

The CFM\_StoreDevDescrFormat\_U16 object describes the external behavior.

Valid values are:

| Value                              | Format  |
|------------------------------------|---|
| 00h                                | XML XDD format as defined in separate paper, not compressed |
| 01 <sub>h</sub> to FE <sub>h</sub> | reserved  |
| FF <sub>h</sub>                    | no Device Description File available                        |

Tab. 101 Device description file and device configuration storage formats

### 6.7.3.4 Object 1F20<sub>h</sub>: CFM\_StoreDcfList\_ADOM

CFM\_StoreDcfList\_ADOM holds XDC files for configured CNs. The object may be implemented on the MN only, if object CFM\_DcfStorageFormatList\_AU8 is implemented.

|           |                       |             |       |
|-----------|-----------------------|-------------|-------|
| Index     | 1F20 <sub>h</sub>     | Object Type | ARRAY |
| Name      | CFM_StoreDcfList_ADOM |             |       |
| Data Type | DOMAIN                | Category    | Cond  |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNDcf**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | CNDcf                              |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | DOMAIN                             | Access      | rws  |
| Default Value | -                                  | PDO Mapping | No   |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

A XDC is downloaded from a Configuration Tool to the Configuration Manager by writing the XDC File to the respective CFM\_StoreDcfList\_ADOM sub-index. The tool uploads the XDC by reading the sub-index.

The availability of XDC data at a CFM\_StoreDcfList\_ADOM sub-index is indicated by the corresponding CFM\_DcfStorageFormatList\_AU8 sub-index. If no data is stored (CFM\_DcfStorageFormatList\_AU8.CNDcfFormat[sub-index] = FF<sub>h</sub>), a SDO Read Request to the CFM\_StoreDcfList\_ADOM sub-index is aborted with the error code E\_CFM\_DATA\_SET\_EMPTY.

### 6.7.3.5 Object 1F21<sub>h</sub>: CFM\_DcfStorageFormatList\_AU8

CFM\_DcfStorageFormatList\_AU8 holds the format of the XDC files for configured CNs. The object may be implemented on the MN only.

|           |                              |             |       |
|-----------|------------------------------|-------------|-------|
| Index     | 1F21 <sub>h</sub>            | Object Type | ARRAY |
| Name      | CFM_DcfStorageFormatList_AU8 |             |       |
| Data Type | UNSIGNED8                    | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNDcfFormat**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | CNDcfFormat                        |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | refer Tab. 101                     | Access      | rws  |
| Default Value | FF <sub>h</sub>                    | PDO Mapping | No   |

CNDcfFormat describes the format of the storage. This allows the use of compressed formats. The device may always store the file in a compressed format internally. The CNDcfFormat object describes the external behavior.

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0 and 1.

If no data is stored (CNDcfFormat[sub-index] = FF<sub>h</sub>), a SDO Read Request to the CFM\_StoreDcfList\_ADOM sub-index is aborted with the error code E\_CFM\_DATA\_SET\_EMPTY.

### 6.7.3.6 Object 1F22<sub>h</sub>: CFM\_ConciseDcfList\_ADOM

CFM\_ConciseDcfList\_ADOM holds Concise Configuration data for configured CNs. The object may be implemented on the MN only.

|           |                         |             |       |
|-----------|-------------------------|-------------|-------|
| Index     | 1F22 <sub>h</sub>       | Object Type | ARRAY |
| Name      | CFM_ConciseDcfList_ADOM |             |       |
| Data Type | DOMAIN                  | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNConciseDcfData**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |     |
| Name          | CNConciseDcfData                   |             |     |
| --            | --                                 | Category    | O   |
| Value Range   | DOMAIN                             | Access      | rws |
| Default Value | -                                  | PDO Mapping | No  |

The content of CNConciseDcfData is a stream with the following structure:

|        |                             |            |
|--------|-----------------------------|------------|
|        | Number of supported entries | UNSIGNED32 |
| Item 1 | Index 1                     | UNSIGNED16 |
|        | Sub-index 1                 | UNSIGNED8  |
|        | Data size of parameter 1    | UNSIGNED32 |
|        | Data of parameter 1         | DOMAIN     |
| Item 2 | Index 2                     | UNSIGNED16 |
|        | Sub-index 2                 | UNSIGNED8  |
|        | Data size of parameter 2    | UNSIGNED32 |
|        | Data of parameter 2         | DOMAIN     |
| .....  |                             |            |
| Item n | Index n                     | UNSIGNED16 |
|        | Sub-index n                 | UNSIGNED8  |
|        | Data size of parameter n    | UNSIGNED32 |
|        | Data of parameter n         | DOMAIN     |

Tab. 102 Concise DCF stream format

The Data Size is measured in octets (i.e. UNSIGNED16 has size 2; size of BOOL is given as 1).

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index.

The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

Device configuration is uploaded from a Configuration Tool to the Configuration Manager by writing the stream as domain to the respective CNConciseDcfData sub-index. Reading the stream uploads to a tool.

An empty data set is written by the following concise stream:

|                             |                |
|-----------------------------|----------------|
| Number of supported entries | 0 (UNSIGNED32) |
|-----------------------------|----------------|

If no data has been stored, a SDO Read Request will return a valid concise stream with the following content:

|                             |                |
|-----------------------------|----------------|
| Number of supported entries | 0 (UNSIGNED32) |
|-----------------------------|----------------|

### 6.7.3.7 Object 1F23<sub>h</sub>: CFM\_StoreDevDescrFileList\_ADOM

CFM\_StoreDevDescrFileList\_ADOM holds Device Description Files of configured CNs. The object may be implemented on the MN only, if object CFM\_DevDescrFileFormatList\_AU8 is implemented.

|           |                                |             |       |
|-----------|--------------------------------|-------------|-------|
| Index     | 1F23 <sub>h</sub>              | Object Type | ARRAY |
| Name      | CFM_StoreDevDescrFileList_ADOM |             |       |
| Data Type | DOMAIN                         | Category    | Cond  |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNDevDescrFile**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |     |
| Name          | CNDevDescrFile                     |             |     |
| --            | --                                 | Category    | O   |
| Value Range   | DOMAIN                             | Access      | rws |
| Default Value | -                                  | PDO Mapping | No  |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

A Device Description File is downloaded from a Configuration Tool to the Configuration Manager by writing the file to the respective CFM\_StoreDevDescrFileList\_ADOM sub-index. Reading the the sub-index uploads to the tool.

The availability of a Device Description File at a CFM\_StoreDevDescrFileList\_ADOM sub-index is indicated by the corresponding CFM\_DevDescrFileFormatList\_AU8 sub-index. If no data had been stored (CFM\_DevDescrFileFormatList\_AU8. CNDevDescrFileFormat [sub-index] = FF<sub>h</sub>), a SDO Read Request to the CFM\_StoreDevDescrFileList\_ADOM sub-index is aborted with the error code E\_CFM\_DATA\_SET\_EMPTY.

### 6.7.3.8 Object 1F24<sub>h</sub>: CFM\_DevDescrFileFormatList\_AU8

CFM\_DevDescrFileFormatList\_AU8 holds the format of the Device Description Files for configured CNs. The object may be implemented on the MN only.

|           |                                |             |       |
|-----------|--------------------------------|-------------|-------|
| Index     | 1F24 <sub>h</sub>              | Object Type | ARRAY |
| Name      | CFM_DevDescrFileFormatList_AU8 |             |       |
| Data Type | UNSIGNED8                      | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNDevDescrFileFormat**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | CNDevDescrFileFormat               |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | refer Tab. 101                     | Access      | rws  |
| Default Value | FF <sub>h</sub>                    | PDO Mapping | No   |

CNDevDescrFileFormat describes the format of the storage. This allows the usage of compressed formats. The device may always store the file in a compressed format internally. The object describes the external behavior.

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

A sub-index is implemented if the corresponding sub-index of CFM\_StoreDevDescrFileList\_ADOM is implemented.

### 6.7.3.9 Object 1F25<sub>h</sub>: CFM\_ConfCNRequest\_AU32

CFM\_ConfCNRequest\_AU32 is used to request configuration data by a CN from the Configuration Manager under runtime conditions. The object may be implemented on the MN only.

|           |                        |             |       |
|-----------|------------------------|-------------|-------|
| Index     | 1F25 <sub>h</sub>      | Object Type | ARRAY |
| Name      | CFM_ConfCNRequest_AU32 |             |       |
| Data Type | UNSIGNED32             | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNConfigurationRequest**

|               |                                    |             |    |
|---------------|------------------------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |    |
| Name          | CNConfigurationRequest             |             |    |
| --            | --                                 | Category    | O  |
| Value Range   | UNSIGNED32                         | Access      | wo |
| Default Value | 0                                  | PDO Mapping | No |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index.

The sub-index value is valid only if there is a CN assigned to the Node ID by index

NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

To initiate the configuration process, the CN writes the signature “conf” to the respective sub-index.

| Signature          | MSB             |                 |                 | LSB             |
|--------------------|-----------------|-----------------|-----------------|-----------------|
| ISO 8859 (“ASCII”) | f               | n               | o               | c               |
| hex                | 66 <sub>h</sub> | 6E <sub>h</sub> | 6F <sub>h</sub> | 63 <sub>h</sub> |

Tab. 103 CNConfigurationRequest write access signature

If no data had been stored, a SDO Write Request to 01<sub>h</sub> to FE<sub>h</sub> is aborted with the error code E\_CFM\_DATA\_SET\_EMPTY.

A SDO Write Request to sub-index FF<sub>h</sub> returns without error.

### 6.7.3.10 Object 1F26<sub>h</sub>: CFM\_ExpConfDateList\_AU32

CFM\_ExpConfDateList\_AU32 holds a list of expected Configuration dates of configured CNs. The object may be implemented on the MN only.

|           |                          |             |       |
|-----------|--------------------------|-------------|-------|
| Index     | 1F26 <sub>h</sub>        | Object Type | ARRAY |
| Name      | CFM_ExpConfDateList_AU32 |             |       |
| Data Type | UNSIGNED32               | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNConfigurationDate**

|               |                                    |             |     |
|---------------|------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |     |
| Name          | CNConfigurationDate                |             |     |
| --            | --                                 | Category    | O   |
| Value Range   | UNSIGNED32                         | Access      | rws |
| Default Value | 0                                  | PDO Mapping | No  |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

Each sub-index holds the configuration date of the respective CN. It contains the number of days since January 1, 1984. It is evaluated in conjunction with the corresponding CFM\_ExpConfTimeList\_AU32 sub-index.

### 6.7.3.11 Object 1F27<sub>h</sub>: CFM\_ExpConfTimeList\_AU32

CFM\_ExpConfTimeList\_AU32 holds a list of expected Configuration times of configured CNs. The object may be implemented on the MN only.

|           |                          |             |       |
|-----------|--------------------------|-------------|-------|
| Index     | 1F27 <sub>h</sub>        | Object Type | ARRAY |
| Name      | CFM_ExpConfTimeList_AU32 |             |       |
| Data Type | UNSIGNED32               | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNConfigurationTime**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | CNConfigurationTime                |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | UNSIGNED32                         | Access      | rws  |
| Default Value | 0                                  | PDO Mapping | No   |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

Each Sub-index holds the configuration time of the respective CN. It contains the number of ms since midnight. It is evaluated in conjunction with the corresponding CFM\_ExpConfDateList\_AU32 sub-index.

A sub-index is implemented if the corresponding sub-index of CFM\_ExpConfDateList\_AU32 is implemented.

### 6.7.3.12 Object 1F28<sub>h</sub>: CFM\_ExpConfdList\_AU32

CFM\_ExpConfdList\_AU32 holds a list of expected Configuration IDs of configured CNs. The object may be implemented on the MN only.

|           |                       |             |       |
|-----------|-----------------------|-------------|-------|
| Index     | 1F28 <sub>h</sub>     | Object Type | ARRAY |
| Name      | CFM_ExpConfdList_AU32 |             |       |
| Data Type | UNSIGNED32            | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 254        | Access      | ro |
| Default Value | 254             | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNConfigurationId**

|               |                                    |             |      |
|---------------|------------------------------------|-------------|------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |      |
| Name          | CNConfigurationId                  |             |      |
| --            | --                                 | Category    | Cond |
| Value Range   | UNSIGNED32                         | Access      | rws  |
| Default Value | 0                                  | PDO Mapping | No   |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is a CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] bits 0 and 1.

Each Sub-index holds the configuration ID of the respective CN.

### 6.7.3.13 Object 0435<sub>h</sub>: CFM\_VerifyConfiguration\_TYPE

| Index           | 0435 <sub>h</sub>            | Object Type       | DEFSTRUCT  |
|-----------------|------------------------------|-------------------|------------|
| Name            | CFM_VerifyConfiguration_TYPE |                   |            |
| Sub-Index       | Component Name               | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries              | 04 <sub>h</sub>   |            |
| 01 <sub>h</sub> | ConfDate_U32                 | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub> | ConfTime_U32                 | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub> | Confld_U32                   | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub> | VerifyConflnvalid_BOOL       | 0001 <sub>h</sub> | BOOLEAN    |

## 6.8 Input from a Programmable Device

### 6.8.1 Basics

In a network a programmable node can be characterised as a process having input and output variables. The set of variables will be arguments of the program and hence their values will not be determined until after the program is written. The arguments must be handled as variables located in the object dictionary.

The definition of such parameters depends on the programming system (e.g. IEC61131-3) and can not be standardised here. It can be assumed, however, that there exists a set of network variables with the logic attribute EXTERN.

Compiling/Linking (or interpreting) a program including EXTERN variables requires relocation information. Within POWERLINK devices this information is the index (and sub-index) of the variable. Most programming systems use the mechanism of a resource definition, used to assign the POWERLINK attributes (index, sub-index, rw, assignment of POWERLINK data type to local data type, etc.) to the corresponding symbolic names (variable name in the program). The resource definition may be created by the user with a simple editor, or, more conveniently using a configuration tool. On systems with a disk-based file system a direct exchange via the DCF format is possible.

Variable names must meet the rules of the underlying programming system. POWERLINK imposes no system-specific restrictions on variable names: correct naming is the responsibility of the programmer/manufacturer.

Defining EXTERN variables requires a rule for distributing the indices, called “dynamic index assignment”.

## 6.8.2 Dynamic index assignment

The index area used for dynamic index assignment is dependent on the device. Each data type and direction (Input/Output) has its own area, called a segment. Segments must not overlap. Variables of the same data type are gathered in an array. If all elements of an array are defined (sub-index 01<sub>h</sub> to FE<sub>h</sub>), the next free object of the area is allocated.

In order to allow programmable devices the use of a process image, they may implement a conversion formula to calculate the offset of a variable in the process image directly from the index and sub-index.

Definition of the abstract object segment (also called dynamic channel):

A segment is a range of indices in the object dictionary with the following attributes:

- **Data type**

The data type of the objects which can be defined in this segment.

- **Direction**

The direction flag distinguishes between inputs and outputs. The values are 'wo' for outputs and 'ro' for inputs. The distinction is important to enable the correct mapping of the variable into a receive PDO (wo) or transmit PDO (ro). This does not affect the access possibilities via SDO.

- **Index range**

The range of indices with start index and end index.

- **PIOffset**

The offset in the process image, where the first object of this segment is allocated.

For byte and multi-byte variables the PIOffset is a 32-bit unsigned offset value.

For Boolean variables it is the offset and additionally the address difference between two Boolean variables counted in bits. If Boolean variables are packed in bytes one bit after the other, the value is 1. If Booleans are each stored in a byte cell, the value is 8.

- **Maximum count**

The maximum number of variables in this segment.

Some devices distinguish strictly between different segments in the process image for different data types. For these devices the PIOffset of the first segment will be 0, the PIOffset of the second segment will be the maximum count of the first segment multiplied by the data type size of the first segment and so on. If the result does not exactly meet the physical configuration, the device software is free to map the segments into the object dictionary in a logical fashion using internal segment descriptors/offsets.

Some devices mix different data types in the same segment. For these devices all PIOffset attributes will have the value 0. Configuration Tools that allocate space in the process image by assigning indices must ensure that under these circumstances indices are omitted to avoid overlapping. (For special applications there may be a feature to explicitly overlap variables – for example, to aid debuggers interpreting memory cells as different data types.)

Any mixed form of these two device types is possible.

## 6.8.3 Object dictionary entries

The network variables are accessed via the entries described by the segments. In some applications it is desirable to read or write the complete process image as one block:

### 6.8.3.1 Object 1F70<sub>h</sub>: INP\_ProcessImage\_REC

INP\_ProcessImage\_REC provides access to process image segments.

|           |                       |             |        |
|-----------|-----------------------|-------------|--------|
| Index     | 1F70 <sub>h</sub>     | Object Type | RECORD |
| Name      | INP_ProcessImage_REC  |             |        |
| Data Type | INP_ProcessImage_TYPE | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 2               | Access      | const |
| Default Value | 2               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: SelectedRange\_U32**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>   |             |    |
| Name          | SelectedRange_U32 |             |    |
| Data Type     | UNSIGNED32        | Category    | M  |
| Value Range   | UNSIGNED32        | Access      | rw |
| Default Value | 0                 | PDO Mapping | No |

Defines the process image segment to be accessed. After writing to SelectedRange\_U32 the corresponding data can be read from or written to the addressed area with ProcessImageDomain\_DOM as an unstructured stream of bytes.

The structure of SelectedRange\_U32 is as follows:

| MSB         |    |    | LSB            |
|-------------|----|----|----------------|
| 31          | 16 | 15 | 0              |
| Data length |    |    | Object segment |

Tab. 104 Structure of SelectedRange\_U32

The Object Segment to be addressed is given by the Index. If several Segments are overlapping, the same memory area can be addressed with each of the indices.

The Data Length gives the maximum size of the transfer in bytes. If the value is 0, the complete segment is to be accessed.

- **Sub-Index 02<sub>h</sub>: ProcessImageDomain\_DOM**

|               |                        |             |    |
|---------------|------------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>        |             |    |
| Name          | ProcessImageDomain_DOM |             |    |
| Data Type     | DOMAIN                 | Category    | M  |
| Value Range   | DOMAIN                 | Access      | rw |
| Default Value | 0                      | PDO Mapping | No |

Holds the data of the segment addressed by SelectedRange\_U32 as an unstructured stream of bytes.

### 6.8.3.2 Object 0428<sub>h</sub>: INP\_ProcessImage\_TYPE

|                 |                        |                   |            |
|-----------------|------------------------|-------------------|------------|
| Index           | 0428 <sub>h</sub>      | Object Type       | DEFSTRUCT  |
| Name            | INP_ProcessImage_TYPE  |                   |            |
| Sub-Index       | Component Name         | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries        | 02 <sub>h</sub>   |            |
| 01 <sub>h</sub> | SelectedRange_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub> | ProcessImageDomain_DOM | 000F <sub>h</sub> | DOMAIN     |

## 7 Network Management (NMT)

### 7.1 NMT State Machine

#### 7.1.1 Overview

The NMT state machine determines the behaviour of the communication function unit (see 2.2.1). The coupling of the application state machine to the NMT state machine is device dependent and falls into the scope of device profiles.

Both MN and CN start up by common initialisation process. At the end of this process, the node specific POWERLINK Node ID is evaluated in order to decide, if the node is setup to be an MN or a CN. The further process differentiates between an MN specific branch and a CN specific one.

The common initialisation process is described by 7.1.2. The paragraph also handles PowerOn, PowerOff and reset levels common to MN and CN.

The MN specific branch is described by Tab. 105, the CN specific one by 7.1.4. Only one of these branches shall be executed on a node.

#### 7.1.2 Common Initialisation NMT State Machine

In Fig. 72 the initialisation of the NMT state machine, common to MN and CN is shown. Fig. 72 also displays PowerOn, PowerOff and common reset levels that affect both MN and CN.

The common initialisation NMT state machine is the nodes upper layer NMT state machine. The MN and CN specific NMT state machines are nested into this state machine. Only one of these nested state machines shall be executed on a node. PowerOff and Reset displayed by the upper layer machine affect each of the nested state machines.

##### 7.1.2.1 States

###### 7.1.2.1.1 NMT\_GS\_POWERED

All the states handled by this paragraph are states that are valid when the device is powered, e.g. they shall be regarded to be sub-states of the super-state NMT\_GS\_POWERED.

NMT\_GS\_POWERED shall be entered on PowerOn (NMT\_GT1). It shall be left on PowerOff (NMT\_GT3).

NMT\_GS\_POWERED is a super-state that won't be signalled over the network by an individual NMTStatus value.

###### 7.1.2.1.1.1 NMT\_GS\_INITIALIZATION

After system start, the node attains the state NMT\_GS\_INITIALIZATION. The node automatically shall enter this state, an NMT command shall not be necessary. In the state NMT\_GS\_INITIALIZATION, the network functionality shall be initialised.

NMT\_GS\_INITIALIZATION and its sub-states are node internal states only. They won't be signalled over the network by NMTStatus.

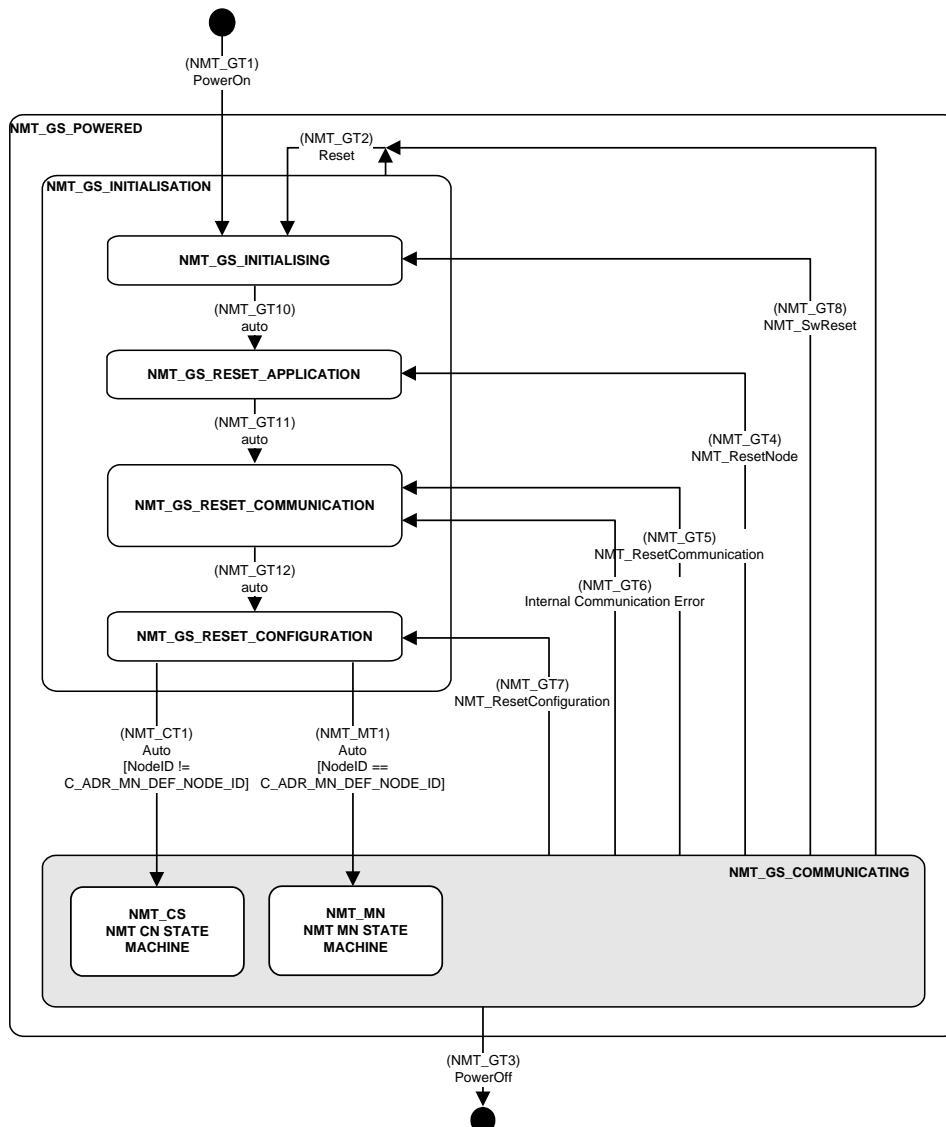


Fig. 72. Common initialisation NMT state machine

### 7.1.2.1.1.1 Sub-states

The state NMT\_GS\_INITIALISATION is divided into four sub-states in order to enable a complete or partial reset of a node (see Fig. 72).

- **NMT\_GS\_INITIALISING**

This is the first sub-state the POWERLINK node shall enter after Power On (NMT\_GT1), hardware resp. software Reset (NMT\_GT2) or the reception of an NMTSwReset (NMT\_GT8) command. After finishing the basic node initialisation, the POWERLINK node shall autonomously enter the sub-state NMT\_GS\_RESET\_APPLICATION (NMT\_GT10).

- **NMT\_GS\_RESET\_APPLICATION**

In this sub-state, the parameters of the manufacturer-specific profile area and of the standardised device profile area shall be set to their PowerOn values. After setting the PowerOn values, the sub-state NMT\_GS\_RESET\_COMMUNICATION shall be entered autonomously (NMT\_GT11).

NMT\_GS\_RESET\_APPLICATION shall be entered upon the reception of an NMTResetNode command from all sub-states of NMT\_GS\_COMMUNICATING, e.g. the NMT MN resp. CN state machine.

- **NMT\_GS\_RESET\_COMMUNICATION**

In this sub-state the parameters of the communication profile area (except ERR\_History\_ADOM) shall be set to their PowerOn values.

NMT\_GS\_RESET\_COMMUNICATION shall be entered upon the recognition of an internal communication error or the reception of an NMTResetCommunication command from all sub-states of NMT\_GS\_COMMUNICATING, e.g. the MN resp. CN NMT state machine.

PowerOn values are the last stored parameters. If no stored configuration is available or if the Reset was preceded by a restore default command (object NMT\_RestoreDefParam\_REC), the PowerOn values shall be set to the default values according to the communication and device profile specifications.

- **NMT\_GS\_RESET\_CONFIGURATION**

In this sub-state the configuration parameters set in the object dictionary are used to generate the active device configuration. The node shall examine its Node ID in order to decide if it's configured to be an MN or a CN. If the node is equal to C\_ADR\_MN\_DEF\_NODE\_ID, the node shall enter the MN NMT state machine (NMT\_MT1), otherwise the CN NMT state machine shall be entered (NMT\_CT1).

NMT\_GS\_RESET\_CONFIGURATION shall be entered upon the reception of an NMTResetConfiguration command from all substates of NMT\_GS\_COMMUNICATING.

This sub-state is used to re-configure devices which do not support storing of communication parameters.

### 7.1.2.1.1.2 NMT\_GS\_COMMUNICATING

When leaving the state NMT\_GS\_INITIALISATION (NMT\_MT1 resp. NMT\_CT1) the super-state NMT\_GS\_COMMUNICATING will be entered. NMT\_GS\_COMMUNICATING includes the MN NMT state machine (refer 7.1.3) as well as the CN NMT state machine (refer 7.1.4).

There shall be a transition from NMT\_GS\_COMMUNICATING to NMT\_GS\_INITIALISATION if an NMTSwReset (NMT\_GT8), NMTRestNode (NMT\_GT4), NMTRestCommunication (NMT\_GT5) or NMTRestConfiguration (NMT\_GT7) command is received or an internal communication error occurs (NMT\_GT6).

NMT\_GS\_COMMUNICATING is a super-state that won't be signalled over the network by an individual NMTStatus value.

### 7.1.2.2 Transitions

|           |   |
|-----------|---|
| (NMT_GT1) | PowerOn [ ] / start basic node initialisation<br>On PowerOn, NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING shall be entered autonomously.<br>NMT_GT1, NMT_GT2 and NMT_GT8 are equivalent transitions, triggered by different reset sources.                           |
| (NMT_GT2) | Reset [ ] / start basic node initialisation<br>After Hardware or local software Reset, NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING shall be entered autonomously.<br>NMT_GT1, NMT_GT2 and NMT_GT8 are equivalent transitions, triggered by different reset sources. |
| (NMT_GT3) | PowerOff [ ]<br>POWERLINK node was powered off in NMT_GS_POWERED  |
| (NMT_GT4) | NMTRestNode [ ] / start application initialisation<br>If an NMTRestNode command is received in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_APPLICATION shall be entered.   |
| (NMT_GT5) | NMTRestCommunication [ ] / start communication initialisation<br>If an NMTRestCommunication command is received in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION shall be entered.   |

|            |  |
|------------|--|
| (NMT_GT6)  | Internal Communication Error [ ] / start communication initialisation<br>If an Internal Communication Error is recognized in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION shall be entered.  |
| (NMT_GT7)  | NMTResetConfiguration [ ] / activate device configuration<br>If an NMTResetConfiguration command is received in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_CONFIGURATION shall be entered.<br>If parameters of the object dictionary concerning the device's cycle configuration (node assignment and timing) are changed, the modification shall take effect after the NMTResetConfiguration command is received. |
| (NMT_GT8)  | NMTSwReset [ ] / start basic node initialisation<br>If an NMTSwReset command is received in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING shall be entered.<br>NMT_GT1, NMT_GT2 and NMT_GT8 are equivalent transitions, triggered by different reset sources.  |
| (NMT_GT10) | Auto [basic node initialisation completed] / start application initialisation<br>NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING completed, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_APPLICATION shall be entered autonomously.  |
| (NMT_GT11) | Auto [application initialisation completed] / start communication initialisation<br>NMT_GS_INITIALISATION sub-state NMT_GS_RESET_APPLICATION completed, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION shall be entered autonomously.  |
| (NMT_GT12) | Auto [communication initialisation completed] / start configuration setup<br>NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION completed, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_CONFIGURATION shall be entered autonomously.   |
| (NMT_MT1)  | Auto [configuration setup completed, Node ID = C_ADR_MN_DEF_NODE_ID ] / start observing network traffic<br>If NMT_GS_INITIALISATION sub-state NMT_GS_RESET_CONFIGURATION is completed and Node ID configuration is equal to the default MN address (C_ADR_MN_DEF_NODE_ID ), state NMT_MS_NOT_ACTIVE shall be entered autonomously, e.g. the MN NMT state machine shall be entered.   |
| (NMT_CT1)  | Auto [communication initialisation completed, Node ID not equal to C_ADR_MN_DEF_NODE_ID ] / start observing network traffic<br>If NMT_GS_INITIALISATION sub-state NMT_GS_RESET_CONFIGURATION is completed and Node ID configuration is not equal to the default MN address (C_ADR_MN_DEF_NODE_ID ), state NMT_CS_NOT_ACTIVE shall be entered autonomously, e.g. the CN NMT state machine shall be entered.                                 |

Tab. 105 Common initialisation NMT state transitions

## 7.1.3 MN NMT State Machine

### 7.1.3.1 Overview

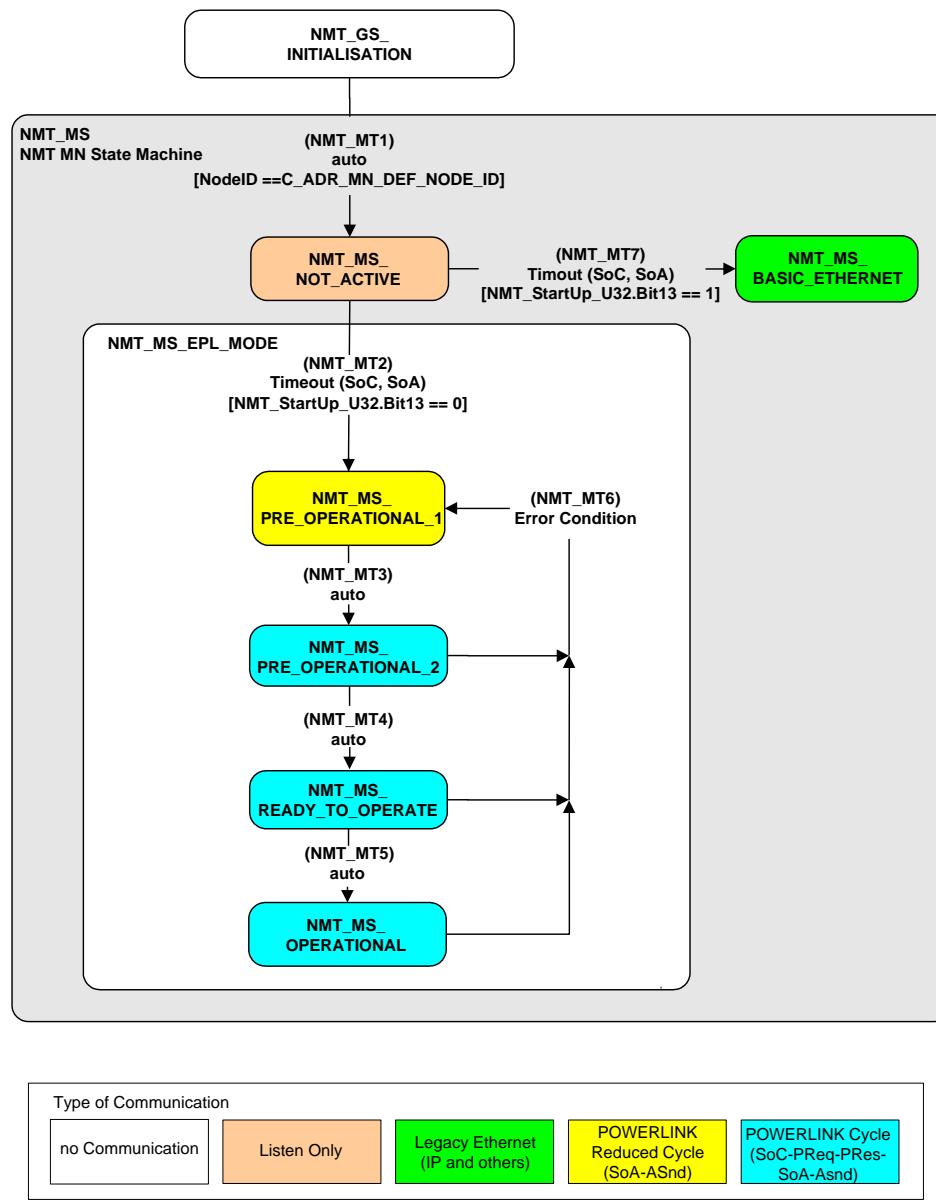


Fig. 73. NMT state diagram of an MN

The MN NMT state machine shall be regarded to be hosted by the common initialisation NMT state machine (7.1.2). The MN NMT state machine represents a sub-state of the super-states NMT\_GS\_POWERED (7.1.2.1.1) and NMT\_GS\_COMMUNICATING (7.1.2.1.1.2). The transitions defined by these states shall be valid at the MN NMT state machine.

### 7.1.3.2 States

The current state of the MN shall define the current state of the POWERLINK network.

#### 7.1.3.2.1 NMT\_MS\_NOT\_ACTIVE

The MN shall observe network traffic in NMT\_MS\_NOT\_ACTIVE in order to ensure, that there is no other MN active on the network.

Reception of a SoC or a SoA frame indicates that there is another MN active. On SoC or SoA, the node shall freeze boot-up. An error shall be signalled to the MN application and the current MN state shall be maintained.

The node shall be not authorised to send any frame in NMT\_MS\_NOT\_ACTIVE.

Depending on NMT\_StartUp\_U32.Bit13 the transition to NMT\_MS\_PRE\_OPERATIONAL\_1 (NMT\_MT2) or NMT\_MS\_BASIC\_ETHERNET (NMT\_MT7) shall be triggered, if there are no SoA or SoC frames received inside the time interval defined by index NMT\_BootTime\_REC.MNWaitNoAct\_U32.

A node that does not support MN mode shall stay in state NMT\_MS\_NOT\_ACTIVE. An error message E\_NMT\_BA1\_NO\_MN\_SUPPORT shall be issued to the application.

### **7.1.3.2.2 NMT\_MS\_EPL\_MODE**

NMT\_MS\_EPL\_MODE is a super-state that is not signalled over the network by an individual NMTStatus value.

#### **7.1.3.2.2.1 NMT\_MS\_PRE\_OPERATIONAL\_1**

In the state NMT\_MS\_PRE\_OPERATIONAL\_1, the MN shall start executing the reduced POWERLINK cycle.

Communication in NMT\_MS\_PRE\_OPERATIONAL\_1 shall be robust to collisions. Collisions shall be resolved by CSMA/CD.

After entering NMT\_MS\_PRE\_OPERATIONAL\_1, the MN shall transmit a sequence of C\_DLL\_PREOP1\_START\_CYCLES SoA frames, not assigning the asynchronous slot to any node including the MN. This start sequence allows collisions pending from pre-NMT\_MS\_EPL\_MODE phase to be resolved.

It shall identify the configured nodes (index NMT\_NodeAssignment\_AU32[Node ID].Bit1) and may check their identification. Identification check completion may be delayed if application SW or configuration data have to be downloaded by the node.

Identified nodes shall be cyclically accessed via StatusRequest.

There is no PDO exchange in NMT\_MS\_PRE\_OPERATIONAL\_1.

The transition from NMT\_MS\_PRE\_OPERATIONAL\_1 to NMT\_MS\_PRE\_OPERATIONAL\_2 (NMT\_MT3) may be triggered if all mandatory CNs have been successfully identified.

It's recommended, that the MN has completed its configuration in NMT\_MS\_PRE\_OPERATIONAL\_1.

Refer 7.4.1.3 for further information about NMT\_MS\_PRE\_OPERATIONAL\_1.

#### **7.1.3.2.2.2 NMT\_MS\_PRE\_OPERATIONAL\_2**

In the state NMT\_MS\_PRE\_OPERATIONAL\_2, the MN shall start executing the isochronous POWERLINK cycle.

It shall start polling the identified CNs, that are in state NMT\_CS\_PRE\_OPERATIONAL\_2 and that are not marked as AsyncOnly CNs, by PReq frames in order to start PDO transfer, synchronisation and heartbeat. The transmitted PReq frames may differ to the PDO mapping requirements. The data shall be declared invalid by not setting the RD flag.

The received PRes frames from the polled CNs shall be ignored.

Identified async-only CNs (index NMT\_NodeAssignment\_AU32[Node ID].Bit8) shall be cyclically accessed via StatusRequest.

Configured but unidentified CNs shall be searched for via SoA IdentRequest frames.

The MN state transition from NMT\_MS\_PRE\_OPERATIONAL\_2 to

NMT\_MS\_READY\_TO\_OPERATE (NMT\_MT4) shall be triggered when all mandatory CNs have signalled to be in state NMT\_CS\_READY\_TO\_OPERATE and the MN has completed its configuration.

Refer 7.4.1.4 for further information about NMT\_MS\_PRE\_OPERATIONAL\_2.

#### **7.1.3.2.2.3 NMT\_MS\_READY\_TO\_OPERATE**

In NMT\_MS\_READY\_TO\_OPERATE, the MN shall execute the isochronous POWERLINK cycle.

When entering NMT\_MS\_READY\_TO\_OPERATE, the MN shall start transmitting PDO data to the identified isochronous CNs according to the requirements of the PDO mapping. The transmitted data shall be declared invalid by resetting the RD flag. The length of the PReq frames shall be equal to the configured PReq payload size of the respective CN (index NMT\_MNPReqPayloadLimitList\_AU16[Node ID]).

PDO data received from the CNs shall be ignored.

Identified async-only CNs shall be cyclically accessed via SoA StatusRequest frames.

Configured but unidentified CNs shall be searched via SoA IdentRequest frames.

The MN state transition from NMT\_MS\_READY\_TO\_OPERATE to NMT\_MS\_OPERATIONAL (NMT\_MT5) shall be triggered if all mandatory CNs transmit their PRes frames with correct frame length and timing.

Refer 7.4.1.5 for further information about NMT\_MS\_READY\_TO\_OPERATE.

#### 7.1.3.2.2.4 NMT\_MS\_OPERATIONAL

NMT\_MS\_OPERATIONAL is the normal operating state of the POWERLINK MN. The MN shall execute the isochronous POWERLINK cycle.

The MN shall transmit PDO data to the identified isochronous CNs according to the requirements of the PDO mapping. The transmitted data may be declared valid by setting the RD flag, if requested by the application. The length of the PReq frames shall be equal to the configured PReq payload size of the respective CN.

The MN may transmit NMTStartNode commands to force CNs state transition from NMT\_CS\_READY\_TO\_OPERATE to NMT\_CS\_OPERATIONAL. The NMTStartNode transmission is controlled by index NMT\_StartUp\_U32.Bits1 and NMT\_StartUp\_U32.Bit3.

Identified Async-only CNs shall be cyclically accessed via SoA StatusRequest frames.

Configured but unidentified CNs shall be searched for via SoA IdentRequest frames.

All mandatory CNs shall be in NMT\_CS\_OPERATIONAL and free of POWERLINK protocol relevant errors. If a mandatory CN is lost, if it has a state not equal to NMT\_CS\_OPERATIONAL or if it has signalled an error, the MN shall change over to NMT\_MS\_PRE\_OPERATIONAL\_1 (NMT\_MT6).

The error reaction of the MN is controlled by index NMT\_StartUp\_U32.Bit4 and NMT\_StartUp\_U32.Bit6.

Refer 7.4.1.6 for further information about NMT\_MS\_OPERATIONAL.

#### 7.1.3.2.3 NMT\_MS\_BASIC\_ETHERNET

In NMT\_MS\_BASIC\_ETHERNET the MN may perform Legacy Ethernet communication according to IEEE 802.3. There is no POWERLINK specific network traffic control. The CSMA/CD collision handling shall control the network access. The node is allowed to transmit autonomously.

Any Legacy Ethernet protocol may be applied. ASnd frames may be transmitted by the MN in state NMT\_MS\_BASIC\_ETHERNET.

To leave NMT\_MS\_BASIC\_ETHERNET, NMT\_MS\_EPL\_MODE has to be enabled by setting NMT\_StartUp\_U32.Bit13 to 0<sub>b</sub> and the device has to be reset by NMTResetCommunication or another reset command.

Support of NMT\_MS\_BASIC\_ETHERNET is optional. Support shall be indicated by D\_NMT\_MNBasicEthernet\_BOOL.

### 7.1.3.3 Transitions

|                         |   |
|-------------------------|---|
| (NMT_MT1)               | Refer Tab. 105  |
| (NMT_MT2)               | <p>Timeout (SoC, SoA) [NMT_StartUp_U32.Bit13 = 0<sub>b</sub>] / enable POWERLINK reduced cycle communication</p> <p>If NMT_MS_EPL_MODE is enabled by NMT_StartUp_U32.Bit13 = 0<sub>b</sub> and the node does not receive any SoA or SoC frame during a definable timeout interval after entering the NMT_MS_NOT_ACTIVE state, the node shall change over to NMT_MS_PRE_OPERATIONAL_1. Timeout defined by NMT_BootTime_REC.MNWaitNoAct_U32</p> |
| (NMT_MT3)               | <p>Auto [All mandatory CNs identified] / enable isochronous POWERLINK cycle communication, invalid PDO, dummy PReq allowed</p> <p>If the node has identified all mandatory CNs, the node shall change to NMT_MS_PRE_OPERATIONAL_2.</p> <p>The state transition may be delayed by the application.</p>   |
| (NMT_MT4)               | <p>Auto [MN configuration completed, all mandatory CNs in NMT_CS_READY_TO_OPERATE] / invalid PDO, configured PReq, PRes</p> <p>If the MN has completed its configuration and if all mandatory CNs are in state NMT_CS_READY_TO_OPERATE, the node shall change to the state NMT_MS_READY_TO_OPERATE.</p> <p>The state transition may be delayed by the application.</p>  |
| (NMT_MT5)               | <p>Auto [The isochronous communication is error free] / enable configured PReq and PRes, start operation</p> <p>If all mandatory CNs transmit their PRes frames with correct frame length and timing, the MN shall change to the state NMT_MS_OPERATIONAL.</p> <p>The state transition may be delayed by the application</p>  |
| (NMT_MT6)               | <p>Auto [Mandatory CN lost, mandatory CN not in NMT_CS_OPERATIONAL, error] / enable POWERLINK reduced cycle communication</p> <p>If the MN detects errors of a mandatory CN, it shall change over to NMT_MS_PRE_OPERATIONAL_1</p>   |
| (NMT_MT7) <sup>22</sup> | <p>Timeout (SoC, SoA) [NMT_StartUp_U32.Bit13 = 1<sub>b</sub>] / enable Legacy Ethernet communication</p> <p>If NMT_StartUp_U32.Bit13 is set, the MN shall change over to NMT_MS_BASIC_ETHERNET</p>  |

Tab. 106 MN specific state transitions

Refer Fig. 72 and Tab. 105 for state transitions defined by the common Initialisation NMT state, which have to be applied to the MN NMT state machine.

<sup>22</sup> optional transition (cf. 7.1.3.2.3)

## 7.1.4 CN NMT State Machine

The CN NMT state machine shall be regarded to be hosted by the common initialisation NMT state machine (7.1.2). The CN NMT state machine represents a sub-state of the super-states NMT\_GS\_POWERED (7.1.2.1.1) and NMT\_GS\_COMMUNICATING (7.1.2.1.1.2). The transitions defined by these states shall be valid at the CN NMT state machine.

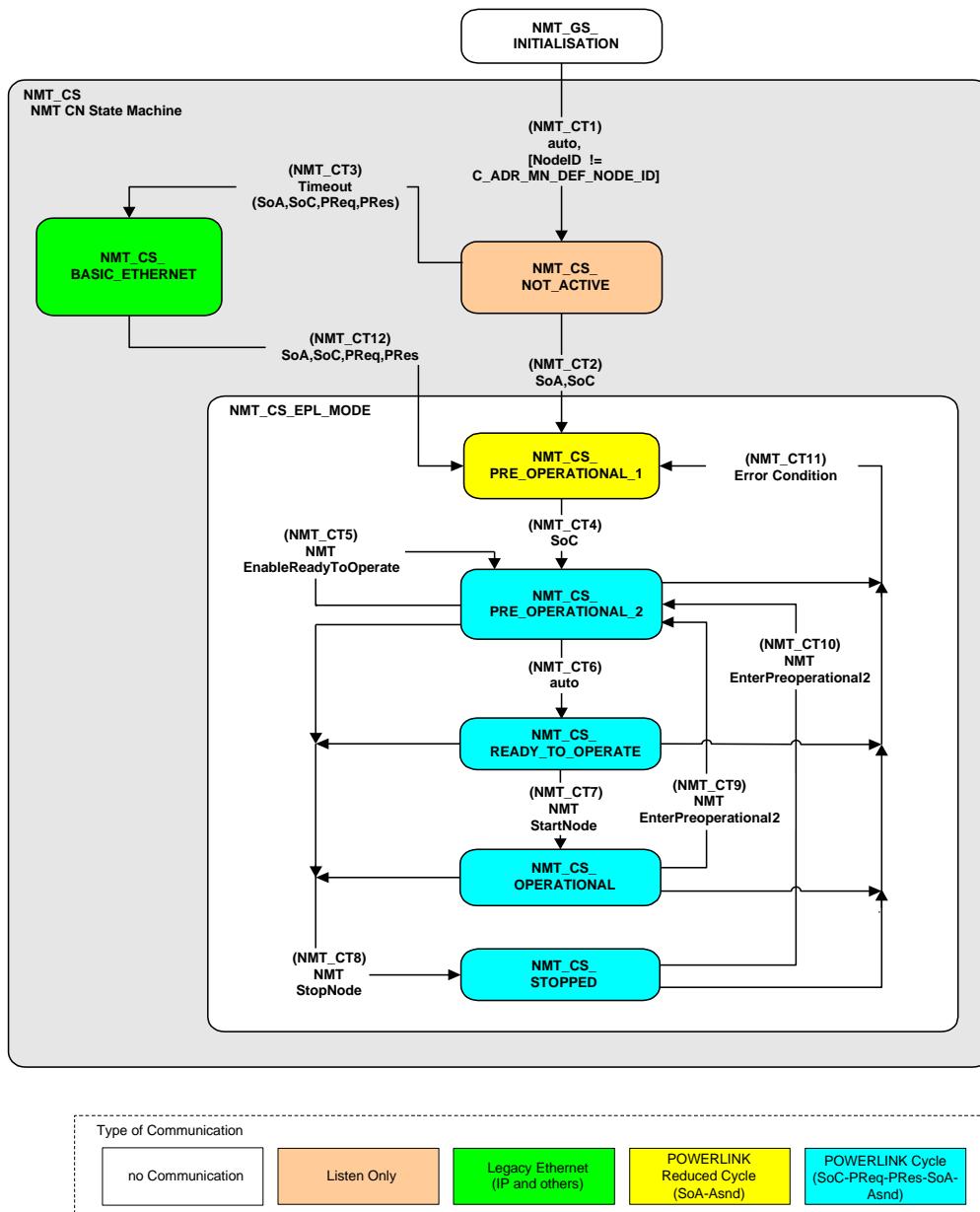


Fig. 74. State diagram of a CN

### 7.1.4.1 States

#### 7.1.4.1.1 NMT\_CS\_NOT\_ACTIVE

NMT\_CS\_NOT\_ACTIVE is a non-permanent state which allows a starting node to recognize the current network state.

The CN shall observe network traffic. The node shall be not authorised to send frames autonomously. There shall be no Legacy Ethernet frame transmission allowed at NMT\_CS\_NOT\_ACTIVE. The node shall be able to recognize NMTReset commands sent via ASnd.

The transition from NMT\_CS\_NOT\_ACTIVE to NMT\_CS\_PRE\_OPERATIONAL\_1 shall be triggered by a SoA or SoC frame being received.

The transition from NMT\_CS\_NOT\_ACTIVE to NMT\_CS\_BASIC\_ETHERNET shall be triggered by timeout for SoC, PReq, PRes and SoA frames.

### **7.1.4.1.2 NMT\_CS\_EPL\_MODE**

NMT\_CS\_EPL\_MODE is a super-state that won't be signalled over the network by an individual NMTStatus value.

#### **7.1.4.1.2.1 NMT\_CS\_PRE\_OPERATIONAL\_1**

In the state NMT\_CS\_PRE\_OPERATIONAL\_1, the CN shall send a frame only if the MN has authorised it to do so by a SoA Asynclnvoke command.

In NMT\_CS\_PRE\_OPERATIONAL\_1 the node shall be identified by the MN via IdentRequest (see 7.3.3.2). If required the CN shall download its configuration data from a configuration server. Both processes may be completely or partially shifted to NMT\_CS\_PRE\_OPERATIONAL\_2, if the MN is not in NMT\_MS\_PRE\_OPERATIONAL\_1 resp. leaves NMT\_MS\_PRE\_OPERATIONAL\_1 before the CN has completed its configuration.

Communication in NMT\_CS\_PRE\_OPERATIONAL\_1 shall be robust to collisions. Collisions shall be resolved by CSMA/CD.

The transition from NMT\_CS\_PRE\_OPERATIONAL\_1 to the following state shall be triggered by a SoC frame being received (see Fig. 74).

There is no PDO communication in NMT\_CS\_PRE\_OPERATIONAL\_1.

#### **7.1.4.1.2.2 NMT\_CS\_PRE\_OPERATIONAL\_2**

In the state NMT\_CS\_PRE\_OPERATIONAL\_2, the CN shall wait for the configuration to be completed.

The node is queried by the MN via PReq. The received PDO data may be invalid, they may differ to the PDO mapping requirements.

The PDO data received from the MN via PReq and from other CNs and the MN via PRes shall be ignored by the CN.

The transmitted PRes frames may differ to the PDO mapping requirements. The data shall be declared invalid by not setting the RD flag.

Async-only CNs shall not be queried by the MN via PReq and thus shall not respond via PRes.

Both types of CN shall respond to Asynclnvoke commands via SoA. If not invited by the MN, there shall be no Ethernet frame transmission allowed at the NMT\_CS\_PRE\_OPERATIONAL\_2 state.

Precondition for the transition from NMT\_CS\_PRE\_OPERATIONAL\_2 to NMT\_CS\_READY\_TO\_OPERATE shall be the reception of an NMTEnableReadyToOperate command. The transition shall be triggered if the application is ready for operation. The maximum transition time from reception of NMTEnableReadyToOperate until the CN is in NMT\_CS\_READY\_TO\_OPERATE may be given by D\_NMT\_CNPReOp2ToReady2Op\_U32.

The transition from NMT\_CS\_PRE\_OPERATIONAL\_2 to NMT\_CS\_PRE\_OPERATIONAL\_1 shall be triggered by an error recognition (see 4.7.7).

The transition from NMT\_CS\_PRE\_OPERATIONAL\_2 to NMT\_CS\_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.3.1.2.1).

#### **7.1.4.1.2.3 NMT\_CS\_READY\_TO\_OPERATE**

With the state NMT\_CS\_READY\_TO\_OPERATE, the CN shall signal its readiness to operation to the MN.

The node may participate in cyclic frame exchange. Cyclic nodes shall respond via PRes when queried via PReq by the MN.

Async-only CNs shall not be queried by the MN via PReq and thus shall not respond via PRes.

Both types of CN shall respond to Asynclnvoke commands via SoA. If not invited by the MN, there shall be no Ethernet frame transmission allowed at the NMT\_CS\_READY\_TO\_OPERATE state.

The RD flag shall be set to 0, regardless if valid process data is available.

The length of the PRes payload shall be less or equal to the configured limit (Object NMT\_CycleTiming\_REC.PResActPayloadLimit\_U16). The transmitted data shall correspond to the requirements defined by the PDO mapping (see 6.4.4).

The transition from NMT\_CS\_READY\_TO\_OPERATE to NMT\_CS\_OPERATIONAL shall be triggered by the reception of NMT state command NMTStartNode (see 7.3.1.2.1)

The transition from NMT\_CS\_READY\_TO\_OPERATE to NMT\_CS\_PRE\_OPERATIONAL\_1 shall be triggered by an error recognition (see 4.7.7).

The transition from NMT\_CS\_READY\_TO\_OPERATE to NMT\_CS\_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.3.1.2.1).

#### **7.1.4.1.2.4 NMT\_CS\_OPERATIONAL**

NMT\_CS\_OPERATIONAL is the normal operating state of a CN.

The CN may participate in cyclic frame exchange. A cyclic CN shall respond via PRes when queried via PReq by the MN.

An Async-only CN isn't queried by the MN via PReq and thus does not respond via PRes.

Both types of CN shall respond to AsyncInvite commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the NMT\_CS\_OPERATIONAL state.

The CN may perform surveillance of other nodes using the NMT guarding mechanism (7.3.5).

The PDO data received from the MN via PReq and from other CNs and the MN via PRes shall be interpreted if selected by the CN application.

The RD flag is controlled by the application. Temporary clearing the RD flag is allowed if PDO data are not valid. The length of the PRes payload shall be less or equal to the configured limit (Object NMT\_CycleTiming\_REC.PResActPayloadLimit\_U16). The transmitted data shall correspond to the requirements defined by the PDO mapping (see 6.4.4).

The transition from NMT\_CS\_OPERATIONAL to NMT\_CS\_PRE\_OPERATIONAL\_2 shall be triggered by the reception of NMT state command NMTEnterPreOperational2 (see 7.3.1.2.1).

The transition from NMT\_CS\_OPERATIONAL to NMT\_CS\_PRE\_OPERATIONAL\_1 shall be triggered by an error recognition(see 4.7.7).

The transition from NMT\_CS\_OPERATIONAL to NMT\_CS\_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.3.1.2.1).

#### **7.1.4.1.2.5 NMT\_CS\_STOPPED**

In the NMT\_CS\_STOPPED state, the node shall be largely passive. NMT\_CS\_STOPPED shall be used for controlled shutdown of a selected CN while the system is still running.

The node shall not participate in cyclic frame exchange, but still observes SoA frames.

It shall not be queried by the MN via PReq.

The node shall not respond via PRes when queried by the MN via PReq.

The node shall respond to AsyncInvite commands via SoA. If not invited by the MN, there is no Ethernet frame transmission allowed at the NMT\_CS\_STOPPED state.

The transition from NMT\_CS\_STOPPED to NMT\_CS\_PRE\_OPERATIONAL\_2 shall be triggered by the reception of NMT state command NMTEnterPreOperational2 (see 7.3.1.2.1)

The transition from NMT\_CS\_STOPPED to NMT\_CS\_PRE\_OPERATIONAL\_1 shall be triggered by an error recognition (see 4.7.7).

#### **7.1.4.1.3 NMT\_CS\_BASIC\_ETHERNET**

In the NMT\_CS\_BASIC\_ETHERNET state the node may perform Legacy Ethernet communication according to IEEE 802.3. There is no POWERLINK specific network traffic control. The CSMA/CD collision handling shall control the network access. The node is allowed to transmit autonomously.

Any Legacy Ethernet protocol may be applied.

ASnd frames may be transmitted by a CN in state NMT\_CS\_BASIC\_ETHERNET.

To avoid disturbance of POWERLINK network traffic when the node is in NMT\_CS\_BASIC\_ETHERNET , the node shall recognize SoC, PReq, PRes and SoA frames. On the reception of such a frame, the CN shall immediately stall any autonomous frame transmission and change over to. NMT\_CS\_PRE\_OPERATIONAL\_1.

### 7.1.4.2 Transitions

|            |   |
|------------|---|
| (NMT_CT1)  | Refer Tab. 105  |
| (NMT_CT2)  | SoA, SoC [ ] / enable POWERLINK reduced cycle communication<br><br>If a SoA or SoC frame is received in NMT_CS_NOT_ACTIVE, the node shall change over to the state NMT_CS_PRE_OPERATIONAL_1.  |
| (NMT_CT3)  | Timeout (SoC, PReq, PRes and SoA) [ ] / enable Legacy Ethernet communication<br><br>If the node does not receive any SoC, PReq, PRes or SoA frame during a definable timeout interval after entering the NMT_CS_NOT_ACTIVE state, the node shall change over to NMT_CS_BASIC_ETHERNET.<br><br>The timeout interval shall be defined by Object NMT_CNBASICETHERNETTIMEOUT_U32. |
| (NMT_CT4)  | SoC [ ] / enable POWERLINK cycle communication, not valid, dummy PRes allowed<br><br>If the node receives a SoC frame in NMT_CS_PRE_OPERATIONAL_1, the node shall change over to NMT_CS_PRE_OPERATIONAL_2.  |
| (NMT_CT5)  | NMTEnableReadyToOperate [] / enable transition to NMT_CS_READY_TO_OPERATE<br><br>The CN is free to change over to NMT_CS_READY_TO_OPERATE after configuration and synchronisation is completed  |
| (NMT_CT6)  | Auto [application is ready and NMTEnableReadyToOperate was received] / enable configured PRes, not valid<br><br>The CN shall automatically change over to NMT_CS_READY_TO_OPERATE   |
| (NMT_CT7)  | NMTStartNode [configuration valid] / enable configured PRes, start operation<br><br>If the CN receives the NMTStartNode command in NMT_CS_READY_TO_OPERATE, it shall change over to NMT_CS_OPERATIONAL  |
| (NMT_CT8)  | NMTStopNode [ ] / freeze cyclic communication<br><br>If the node receives an NMTStopNode command in NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE or NMT_CS_OPERATIONAL, it shall change over to NMT_CS_STOPPED.  |
| (NMT_CT9)  | NMTEEnterPreoperational2 [ ] / reset RD flag in PRes, dummy PRes only<br><br>If the node receives the NMTEEnterPreoperational2 command in NMT_CS_OPERATIONAL, it shall change over to NMT_CS_PRE_OPERATIONAL_2  |
| (NMT_CT10) | NMTEEnterPreoperational2 [ ] / re-enable POWERLINK cycle communication, dummy PRes only<br><br>If the node receives the NMTEEnterPreoperational2 command in NMT_CS_STOPPED, it shall change over to NMT_CS_PRE_OPERATIONAL_2.   |
| (NMT_CT11) | Error condition [ ] / enable POWERLINK reduced cycle communication<br><br>If the node recognizes an error condition (refer 4.7.7) in NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE, NMT_CS_OPERATIONAL or NMT_CS_STOPPED, the node shall change over to NMT_CS_PRE_OPERATIONAL_1  |
| (NMT_CT12) | SoC, PReq, PRes or SoA [ ] / stall autonomous frame transmission<br><br>If a SoC, PReq, PRes or SoA frame is received in NMT_CS_BASIC_ETHERNET, the node shall change over to NMT_CS_PRE_OPERATIONAL_1.<br><br><b><i>It's extremely important that the node immediately stops any autonomous frame transmission, when it recognizes a SoC, PReq, PRes or SoA frame.</i></b>   |

Tab. 107 CN specific state transitions

Refer Fig. 72 and Tab. 105 for state transitions defined by the common Initialisation NMT state, that have to applied to the CN NMT state machine.

### 7.1.4.3 States and Communication Object Relation

Tab. 108 shows the relation between communication states and communication objects. Services on the listed communication objects may only be executed if the devices involved in the communication are in the appropriate communication states.

|  |   | NMT_GS_INITIALISATION | NMT_CS_NOT_ACTIVE | NMT_CS_PRE_OPERATIONAL_1 | NMT_CS_PRE_OPERATIONAL_2 | NMT_CS_READY_TO_OPERATE | NMT_CS_OPERATIONAL | NMT_CS_STOPPED   | NMT_CS_BASIC_ETHERNET |
|--|---|-----------------------|-------------------|--------------------------|--------------------------|-------------------------|--------------------|------------------|-----------------------|
| <b>POWERLINK controlled network traffic</b>        |   |                       |                   |                          |                          |                         |                    |                  |                       |
| SoC  | - | R/S                   | R/S               | R                        | R                        | R                       | -                  | R/S              |                       |
| PReq   | - | -                     | -                 | R                        | R                        | R                       | -                  | R/S              |                       |
| PDO reception                                      | - | -                     | -                 | -                        | (x) <sup>1</sup>         | x                       | -                  | -                |                       |
| PRes receive                                       | - | -                     | -                 | -                        | R                        | R                       | -                  | R/S              |                       |
| PRes transmit                                      | - | -                     | -                 | (T)                      | T                        | T                       | -                  | -                |                       |
| PDO transmission                                   | - | -                     | -                 | -                        | (x) <sup>2</sup>         | x                       | -                  | -                |                       |
| SoA  | - | R/S                   | R                 | R                        | R                        | R                       | R                  | R/S              |                       |
| IdentRequest                                       | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| StatusRequest                                      | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| NMTRequestInvite                                   | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| UnspecifiedInvite                                  | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| Reception of asynchronous frames                   | - | R                     | R                 | R                        | R                        | R                       | R                  | R                |                       |
| SDO reception                                      | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| NMT Command  | - | (x) <sup>3</sup>      | (x) <sup>4</sup>  | (x) <sup>4</sup>         | (x) <sup>4</sup>         | (x) <sup>4</sup>        | (x) <sup>4</sup>   | (x) <sup>4</sup> | (x) <sup>3</sup>      |
| other protocols                                    | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| Transmission, assigned by SoA                      | - | -                     | T                 | T                        | T                        | T                       | T                  | T                | -                     |
| SDO transmission                                   | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| NMTRequest transmission                            | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| IdentResponse                                      | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| StatusResponse                                     | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| other protocols                                    | - | -                     | x                 | x                        | x                        | x                       | x                  | -                |                       |
| <b>Network traffic not controlled by POWERLINK</b> |   |                       |                   |                          |                          |                         |                    |                  |                       |
| Legacy Ethernet reception                          | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | R                     |
| UDP/IP reception                                   | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| SDO reception (UDP/IP)                             | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| POWERLINK-ASnd reception                           | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| SDO reception (POWERLINK-ASnd)                     | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| Legacy Ethernet transmission                       | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | T                     |
| UDP/IP, autonomously sent                          | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| SDO transmission (UDP/IP)                          | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| POWERLINK-ASnd, autonomously sent                  | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |
| SDO transmission (POWERLINK-ASnd)                  | - | -                     | -                 | -                        | -                        | -                       | -                  | -                | (x) <sup>5</sup>      |

- R frame accepted  
 R/S frame accepted, triggers state transition  
 T frame transmitted  
 (T) dummy PRes only  
 x frame data interpreted resp. transmitted  
 (x)<sup>1</sup> frame data may be interpreted  
 (x)<sup>2</sup> data invalidated by resetting the RD flag  
 (x)<sup>3</sup> only selected NMT commands accepted, shall cause state transition, refer 7.3.1.2.1  
 (x)<sup>4</sup> may cause state transition, refer 7.3.1.2.1  
 (x)<sup>5</sup> depends on protocol support  
 - no frame handling

Tab. 108 States and communication objects

#### 7.1.4.4 Relationship to other state machines

The CN NMT state machine is commanded by the MN NMT state machine via NMT commands.

The NMT state machines are operating in close relationship to the cycle state machines (refer 4.2.4.5 resp. 4.2.4.6).

## 7.2 NMT Object Dictionary Entries

NMT Object Dictionary hosts entries defining node internal parameters that control the isochronous POWERLINK cycle. These internal parameters shall not be changed when the isochronous POWERLINK cycle is in operation.

Modifications of the respective OD entries shall be restricted to the OD data handling but shall not immediately influence the internal parameter set controlling the current POWERLINK cycle. To make POWERLINK cycle relevant OD entries valid, the device shall be set to NMT\_GS\_RESET\_CONFIGURATION by NMTResetConfiguration, a more powerful NMT reset command or a HW reset.

OD entries that require a such type handling are indicated by the access type supplement "valid on reset".

### 7.2.1 NMT General Objects

Most of the objects described by this paragraph apply to all nodes.

Some sub-indices of objects otherwise mandatory to the MN may not be implemented on the MN. Refer to the Category entry to identify unsupported items on the MN (Category = No).

#### 7.2.1.1 Identification

##### 7.2.1.1.1 Object 1000<sub>h</sub>: NMT\_DeviceType\_U32

Contains information about the device type. The object describes the type of device and its functionality.

The value shall be setup by the device firmware during system initialisation.

|               |                    |             |       |
|---------------|--------------------|-------------|-------|
| Index         | 1000 <sub>h</sub>  | Object Type | VAR   |
| Name          | NMT_DeviceType_U32 |             |       |
| Data Type     | UNSIGNED32         | Category    | M     |
| Value Range   | UNSIGNED32         | Access      | const |
| Default Value | -                  | PDO Mapping | No    |

- **Value Interpretation**

| Byte: | MSB                    |  |  | LSB                   |
|-------|------------------------|--|--|-----------------------|
|       | Additional Information |  |  | Device Profile Number |

Tab. 109 NMT\_DeviceType\_U32 value interpretation

NMT\_DeviceType\_U32 is composed of a 16-bit field which describes the device profile that is used and a second 16-bit field which gives additional information about optional functionality of the device.

The Additional Information parameter is device profile specific. Its specification does not fall within the scope of this document, it is defined in the appropriate device profile. The value 0000<sub>h</sub> indicates a device that does not follow a standardised device profile.

For multiple device modules the Additional Information parameter contains FFFF<sub>h</sub> and the device profile number referenced by object 1000<sub>h</sub> is the device profile of the first device in the Object Dictionary. All other devices of a multiple device module identify their profiles at objects 67FF<sub>h</sub> + x \* 800<sub>h</sub> with x = internal number of the device (0 – 7). These entries describe the device type of the preceding device.

##### 7.2.1.1.2 Object 1008<sub>h</sub>: NMT\_ManufactDevName\_VS

Contains the manufacturer device name.

If implemented, it shall be setup by the device firmware during system initialisation.

*Remark: In the device description file (see separate paper) the sub element "productName" of element "DeviceIdentity" must be equal to NMT\_ManufactDevName\_VS.*

|               |                        |             |       |
|---------------|------------------------|-------------|-------|
| Index         | 1008 <sub>h</sub>      | Object Type | VAR   |
| Name          | NMT_ManufactDevName_VS |             |       |
| Data Type     | VISIBLE_STRING         | Category    | O     |
| Value Range   | -                      | Access      | const |
| Default Value | -                      | PDO Mapping | No    |

### 7.2.1.1.3 Object 1009<sub>h</sub>: NMT\_ManufactHwVers\_VS

Contains the manufacturer hardware version description.

*Remark: In the device description file (see separate paper) the sub element “version” (with attribute “versionType” set to “HW”) of element “DeviceIdentity” must be equal to NMT\_ManufactHwVers\_VS.*

|               |                       |             |       |
|---------------|-----------------------|-------------|-------|
| Index         | 1009 <sub>h</sub>     | Object Type | VAR   |
| Name          | NMT_ManufactHwVers_VS |             |       |
| Data Type     | VISIBLE_STRING        | Category    | O     |
| Value Range   | -                     | Access      | const |
| Default Value | -                     | PDO Mapping | No    |

### 7.2.1.1.4 Object 100A<sub>h</sub>: NMT\_ManufactSwVers\_VS

Contains the manufacturer software version description.

*Remark: In the device description file (see separate paper) the sub element “version” (with attribute “versionType” set to “FW”) of element “DeviceIdentity” must be equal to NMT\_ManufactSwVers\_VS.*

|               |                       |             |       |
|---------------|-----------------------|-------------|-------|
| Index         | 100A <sub>h</sub>     | Object Type | VAR   |
| Name          | NMT_ManufactSwVers_VS |             |       |
| Data Type     | VISIBLE_STRING        | Category    | O     |
| Value Range   | -                     | Access      | const |
| Default Value | -                     | PDO Mapping | No    |

### 7.2.1.1.5 Object 1018<sub>h</sub>: NMT\_IdentityObject\_REC

The object at index 1018<sub>h</sub> contains general information about the device.

The values shall be setup by the device firmware during system initialisation.

|           |                        |             |        |
|-----------|------------------------|-------------|--------|
| Index     | 1018 <sub>h</sub>      | Object Type | RECORD |
| Name      | NMT_IdentityObject_REC |             |        |
| Data Type | IDENTITY               | Category    | M      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 1 .. 4          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: VendorId\_U32**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 01 <sub>h</sub> |             |       |
| Name          | VendorId_U32    |             |       |
| Data Type     | UNSIGNED32      | Category    | M     |
| Value Range   | UNSIGNED32      | Access      | const |
| Default Value | -               | PDO Mapping | No    |

The sub-index provides the manufacturer-specific vendor ID.

The POWERLINK vendor ID is equal to the CANopen vendor ID.

*Note: A CANopen vendor ID can be obtained from CAN in Automation (CiA).*

- **Sub-Index 02<sub>h</sub>: ProductCode\_U32**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 02 <sub>h</sub> |             |       |
| Name          | ProductCode_U32 |             |       |
| Data Type     | UNSIGNED32      | Category    | O     |
| Value Range   | UNSIGNED32      | Access      | const |
| Default Value | -               | PDO Mapping | No    |

The manufacturer-specific Product code identifies a specific device version. The value shall be equal to the device description entry D\_NMT\_ProductCode\_U32.

- **Sub-Index 03<sub>h</sub>: RevisionNo\_U32**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 03 <sub>h</sub> |             |       |
| Name          | RevisionNo_U32  |             |       |
| Data Type     | UNSIGNED32      | Category    | O     |
| Value Range   | UNSIGNED32      | Access      | const |
| Default Value | -               | PDO Mapping | No    |

The manufacturer-specific revision number consists of a major revision number and a minor revision number. The major revision number identifies a specific device behaviour. If the device functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions with the same device behaviour. The value shall be equal to the device description entry D\_NMT\_RevisionNo\_U32.

|                       |    |                       |     |
|-----------------------|----|-----------------------|-----|
| 31                    | 16 | 15                    | 0   |
| major revision number |    | minor revision number |     |
| MSB                   |    |                       | LSB |

Tab. 110 Structure of Revision number

- **Sub-Index 04<sub>h</sub>: SerialNo\_U32**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 04 <sub>h</sub> |             |       |
| Name          | SerialNo_U32    |             |       |
| Data Type     | UNSIGNED32      | Category    | O     |
| Value Range   | UNSIGNED32      | Access      | const |
| Default Value | -               | PDO Mapping | No    |

The sub-index provides the Serial Number of the device.

### 7.2.1.1.6 Object 1F82<sub>h</sub>: NMT\_FeatureFlags\_U32

Feature Flags indicate communication profile specific properties of the device given by its design.

The object shall be setup by the device firmware during system initialisation.

|               |                      |             |       |
|---------------|----------------------|-------------|-------|
| Index         | 1F82 <sub>h</sub>    | Object Type | VAR   |
| Name          | NMT_FeatureFlags_U32 |             |       |
| Data Type     | UNSIGNED32           | Category    | M     |
| Value Range   | -                    | Access      | const |
| Default Value | -                    | PDO Mapping | No    |

- Value Interpretation

| Octet | Bit | Name                                       | TRUE  | FALSE  |
|-------|-----|--|---|--|
| 0     | 0   | Isochronous                                | device may be isochronously accessed via PReq, it may be operated as isochronous CN (see 4.2.2.2.1) | device does not support isochronous access via PReq, it may be exclusively used as async-only CN (see 4.2.2.2.2)<br><br>D_NMT_Isochronous_BOOL         |
|       | 1   | SDO by UDP/IP                              | device supports SDO communication via UDP/IP frames (see 6.3.3.1)                                   | device does not support SDO communication via UDP/IP frames (see 6.3.3.1)<br><br>D_SDO_SupportUdpIp_BOOL   |
|       | 2   | SDO by ASnd                                | device supports SDO communication via POWERLINK ASnd frames (see 6.3.3.1)                           | device does not support SDO communication via POWERLINK ASnd frames (see 6.3.3.1)<br><br>D_SDO_SupportASnd_BOOL  |
|       | 3   | SDO by PDO                                 | device supports SDO communication via container embedded in PDO communication (see 6.3.3.1)         | device does not support SDO communication via container embedded in PDO communication (see 6.3.3.1)<br><br>D_SDO_SupportPDO_BOOL                       |
|       | 4   | NMT Info Services                          | device supports NMT Info Services (see 7.3.4)   | device does not support NMT Info Services (see 7.3.4)<br><i>Note: Device description entries are defined for each service.</i>                         |
|       | 5   | Extended NMT State Commands                | device supports reception of Extended NMT State Commands (see 7.3.1.2.2)                            | device does not support reception of Extended NMT State Commands (see 7.3.1.2.2)<br><br>D_NMT_ExtNmtCmnds_BOOL   |
|       | 6   | Dynamic PDO Mapping                        | device supports dynamic PDO mapping (see 6.4.2)   | device does not support dynamic PDO mapping (see 6.4.2)<br><br>D_PDO_DynamicMapping_BOOL   |
|       | 7   | NMT Services by UDP/IP                     | device supports NMT Services by UDP/IP (see 7.3.8)  | device does not support NMT Services by UDP/IP (see 7.3.8)<br><br>D_NMT_ServiceUdpIp_BOOL  |
|       | 8   | Configuration Manager                      | device supports Configuration manager functions (see 6.7)   | device does not support Configuration manager functions (see 6.7)<br><br>D_CFM_ConfigManager_BOOL  |
|       | 9   | Multiplexed Access                         | CN device supports Multiplexed isochronous access (see 4.2.4.1.1.1)                                 | CN device does not support Multiplexed isochronous access (see 4.2.4.1.1.1)<br><br>D_DLL_CNFeatureMultiplex_BOOL                                       |
|       | 10  | Node ID setup by SW                        | device supports Node ID setup by SW (see 7.2.1.3.1)   | device does not support Node ID setup by SW (see 7.2.1.3.1)<br><br>D_NMT_NodeIDBySW_BOOL   |
|       | 11  | MN Basic Ethernet Mode                     | MN device supports Basic Ethernet Mode (see 7.1.3.2.3)  | MN device does not support Basic Ethernet Mode (see 7.1.3.2.3)<br><br>D_NMT_MNBasicEthernet_BOOL   |
|       | 12  | Routing Type 1 Support                     | device supports Routing Type 1 functions (see 9.1)  | device does not support Routing Type 1 functions (see 9.1)<br><br>D_RT1_RT1Support_BOOL  |
|       | 13  | Routing Type 2 Support                     | device supports Routing Type 2 functions (see 9.2)  | device does not support Routing Type 2 functions (see 9.2)<br><br>D_RT2_RT2Support_BOOL  |
|       | 14  | SDO Read/Write All by Index                | device supports SDO commands Read and Write All by Index (see 6.3.2.4.2)                            | device does not support SDO commands Read and Write All by Index (see 6.3.2.4.2)<br><br>D_SDO_CmdReadAllByIndex_BOOL,<br>D_SDO_CmdWriteAllByIndex_BOOL |
|       | 15  | SDO Read/Write Multiple Parameter by Index | device supports SDO commands Read and Write Multiple Parameter by Index (see 6.3.2.4.2)             | device does not support SDO commands Read and Write Multiple Parameter by Index (see 6.3.2.4.2)<br><br>D_SDO_CmdReadMultParam_BOOL,                    |

| Octet | Bit      | Name     | TRUE                         | FALSE |
|-------|----------|----------|------------------------------|-------|
|       |          |          | D_SDO_CmdWriteMultParam_BOOL |       |
| 2     | 16       | reserved | Used by EPSG DS302-B [2]     |       |
|       | 17       | reserved | Used by EPSG DS302-A [1]     |       |
|       | 18       | reserved | Used by EPSG DS302-C [3]     |       |
|       | 19       | reserved | Used by EPSG DS302-D [4]     |       |
|       | 20       | reserved | Used by EPSG DS302-E [5]     |       |
|       | 21 .. 23 | reserved | --                           | --    |
|       | 31       | reserved | --                           | --    |

Tab. 111 NMT\_FeatureFlags\_U32 interpretation

### 7.2.1.1.7 Object 1F83<sub>h</sub>: NMT\_EPLVersion\_U8

The index holds the POWERLINK communication profile version that is implemented by device.

The value shall be setup by the device firmware during system initialisation.

|               |                   |             |       |
|---------------|-------------------|-------------|-------|
| Index         | 1F83 <sub>h</sub> | Object Type | VAR   |
| Name          | NMT_EPLVersion_U8 |             |       |
| Data Type     | UNSIGNED8         | Category    | M     |
| Value Range   | -                 | Access      | const |
| Default Value | -                 | PDO Mapping | No    |

- **Value Interpretation**

| High Nibble            | Low Nibble            |
|------------------------|-----------------------|
| POWERLINK Main Version | POWERLINK Sub Version |

Tab. 112 NMT\_EPLVersion\_U8 encoding

### 7.2.1.2 Parameter Storage

#### 7.2.1.2.1 Object 1010<sub>h</sub>: NMT\_StoreParam\_REC

This object supports the saving of parameters in non volatile memory. By read access the device provides information about its saving capabilities.

|           |                           |             |        |
|-----------|---------------------------|-------------|--------|
| Index     | 1010 <sub>h</sub>         | Object Type | RECORD |
| Name      | NMT_StoreParam_REC        |             |        |
| Data Type | NMT_ParameterStorage_TYPE | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                   |             |       |
|---------------|-----------------------------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub>                   |             |       |
| Name          | NumberOfEntries                   |             |       |
| Value Range   | 1 <sub>h</sub> .. 7F <sub>h</sub> | Access      | const |
| Default Value | -                                 | PDO Mapping | No    |

NumberOfEntries is implementation specific.

- **Sub-Index 01<sub>h</sub>: AllParam\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> |             |    |
| Name          | AllParam_U32    |             |    |
| Data Type     | UNSIGNED32      | Category    | M  |
| Value Range   | UNSIGNED32      | Access      | rw |
| Default Value | -               | PDO Mapping | No |

Refers to all parameters that can be stored on the device.

- **Sub-Index 02<sub>h</sub>: CommunicationParam\_U32**

|               |                        |             |    |
|---------------|------------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>        |             |    |
| Name          | CommunicationParam_U32 |             |    |
| Data Type     | UNSIGNED32             | Category    | O  |
| Value Range   | UNSIGNED32             | Access      | rw |
| Default Value | -                      | PDO Mapping | No |

Refers to communication related parameters (Index 1000<sub>h</sub> .. 1FFF<sub>h</sub>: manufacturer specific communication parameters).

- **Sub-Index 03<sub>h</sub>: ApplicationParam\_U32**

|               |                      |             |    |
|---------------|----------------------|-------------|----|
| Sub-Index     | 03 <sub>h</sub>      |             |    |
| Name          | ApplicationParam_U32 |             |    |
| Data Type     | UNSIGNED32           | Category    | O  |
| Value Range   | UNSIGNED32           | Access      | rw |
| Default Value | -                    | PDO Mapping | No |

Refers to application related parameters (Index 6000<sub>h</sub> .. 9FFF<sub>h</sub>: manufacturer specific application parameters).

- **Sub-Index 04<sub>h</sub> .. 7F<sub>h</sub>: ManufacturerParam\_XXh\_U32**

|               |                           |             |    |
|---------------|---------------------------|-------------|----|
| Sub-Index     | 04h .. 7Fh                |             |    |
| Name          | ManufacturerParam_XXh_U32 |             |    |
| Data Type     | UNSIGNED32                | Category    | O  |
| Value Range   | UNSIGNED32                | Access      | rw |
| Default Value | -                         | PDO Mapping | No |

ManufacturerParam\_XXh\_U32 provide means to store manufacturer specific lists of data.

To allow access by name “\_XXh” is replaced with a name index. For example, the name index is “\_04h” if the sub-index is 04<sub>h</sub>. The name index is incremented up to “\_7Fh” corresponding to sub-index 7F<sub>h</sub>.

- **Value Interpretation of Sub-Index 01<sub>h</sub> .. 7F<sub>h</sub>**

In order to avoid storage of parameters by mistake, storage is only executed when a specific signature is written to the appropriate sub-index. The signature is „save“.

| Signature          | MSB             |                 |                 | LSB             |
|--------------------|-----------------|-----------------|-----------------|-----------------|
| ISO 8859 (“ASCII”) | e               | v               | a               | s               |
| hex                | 65 <sub>h</sub> | 76 <sub>h</sub> | 61 <sub>h</sub> | 73 <sub>h</sub> |

Tab. 113 NMT\_StoreParam\_REC storage write access signature

On reception of the correct signature in the appropriate sub-index the device stores the parameter and then confirms the SDO transmission. If the storing fails, the device responds with an Abort SDO Transfer.

If a wrong signature is written, the device refuses to store it and responds with Abort SDO Transfer.

On read access to the appropriate sub-index the device provides information about its storage functionality with the following format:

|      |               |     |     |
|------|---------------|-----|-----|
|      | UNSIGNED32    |     |     |
|      | MSB           |     | LSB |
| bits | 31 .. 2       | 1   | 0   |
|      | reserved (=0) | 0/1 | 0/1 |

Tab. 114 NMT\_StoreParam\_REC storage read access structure

| bit     | value | meaning                                      |
|---------|-------|--|
| 31 .. 2 | 0     | Reserved (=0 <sub>b</sub> )                  |
| 1       | 0     | Device does not save parameters autonomously |
|         | 1     | Device saves parameters autonomously         |
| 0       | 0     | Device does not save parameters on command   |
|         | 1     | Device saves parameters on command           |

Tab. 115 NMT\_StoreParam\_REC structure of read access

Autonomous saving means that a device stores the storables parameters in a non-volatile memory without user request.

### 7.2.1.2.2 Object 1011<sub>h</sub>: NMT\_RestoreDefParam\_REC

With this object the default values of parameters according to the communication or device profile are restored. By read access the device provides information about its capabilities to restore these values. Several parameter groups are distinguished:

|           |                           |             |        |
|-----------|---------------------------|-------------|--------|
| Index     | 1011 <sub>h</sub>         | Object Type | RECORD |
| Name      | NMT_RestoreDefParam_REC   |             |        |
| Data Type | NMT_ParameterStorage_TYPE | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                                   |             |       |
|---------------|-----------------------------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub>                   |             |       |
| Name          | NumberOfEntries                   |             |       |
| Value Range   | 1 <sub>h</sub> .. 7F <sub>h</sub> | Access      | const |
| Default Value | -                                 | PDO Mapping | No    |

NumberOfEntries is implementation specific.

- **Sub-Index 01<sub>h</sub>: AllParam\_U32**

|               |                 |             |                                |
|---------------|-----------------|-------------|--------------------------------|
| Sub-Index     | 01 <sub>h</sub> |             |                                |
| Name          | AllParam_U32    |             |                                |
| Data Type     | UNSIGNED32      | Category    | M                              |
| Value Range   | UNSIGNED32      | Access      | rw, cf. Validation information |
| Default Value | -               | PDO Mapping | No                             |

Refers to all parameters that can be stored on the device.

- **Sub-Index 02<sub>h</sub>: CommunicationParam\_U32**

|               |                        |             |                                |
|---------------|------------------------|-------------|--------------------------------|
| Sub-Index     | 02 <sub>h</sub>        |             |                                |
| Name          | CommunicationParam_U32 |             |                                |
| Data Type     | UNSIGNED32             | Category    | O                              |
| Value Range   | UNSIGNED32             | Access      | rw, cf. Validation information |
| Default Value | -                      | PDO Mapping | No                             |

Restore communication default parameters, refers to communication related parameters (Index 1000<sub>h</sub> .. 1FFF<sub>h</sub>: manufacturer specific communication parameters).

- **Sub-Index 3h: ApplicationParam\_U32**

|               |                      |             |                                |
|---------------|----------------------|-------------|--------------------------------|
| Sub-Index     | 03 <sub>h</sub>      |             |                                |
| Name          | ApplicationParam_U32 |             |                                |
| Data Type     | UNSIGNED32           | Category    | O                              |
| Value Range   | UNSIGNED32           | Access      | rw, cf. Validation information |
| Default Value | -                    | PDO Mapping | No                             |

Restore application default parameters, refers to application related parameters (Index 6000<sub>h</sub> – 9FFF<sub>h</sub> manufacturer specific application parameters).

- **Sub-Index 04<sub>h</sub> .. 7F<sub>h</sub>: ManufacturerParam\_XXh\_U32**

|               |                                    |             |                                |
|---------------|------------------------------------|-------------|--------------------------------|
| Sub-Index     | 04 <sub>h</sub> .. 7F <sub>h</sub> |             |                                |
| Name          | ManufacturerParam_XXh_U32          |             |                                |
| Data Type     | UNSIGNED32                         | Category    | O                              |
| Value Range   | UNSIGNED32                         | Access      | rw, cf. Validation information |
| Default Value | -                                  | PDO Mapping | No                             |

Restore manufacturer defined default parameters. Manufacturers may restore their individual choice of parameters.

To allow access by name “\_XXh” is replaced with a name index. For example, the name index is “\_04h” if the sub-index is 04<sub>h</sub>. The name index is incremented up to “\_7Fh” corresponding to sub-index 7F<sub>h</sub>.

- **Sub-Index 01<sub>h</sub> – 7F<sub>h</sub> Value Interpretation**

In order to avoid the restoring of default parameters by mistake, restoring is only executed when a specific signature is written to the appropriate sub-index. The signature is „load“.

| Signature | MSB             |                 |                 | LSB             |
|-----------|-----------------|-----------------|-----------------|-----------------|
| ASCII     | d               | a               | o               | l               |
| hex       | 64 <sub>h</sub> | 61 <sub>h</sub> | 6F <sub>h</sub> | 6C <sub>h</sub> |

Tab. 116 NMT\_RestoreDefParam\_REC restoring write access signature

On reception of the correct signature in the appropriate sub-index the device restores the default parameters and then confirms the SDO transmission. If the restoring failed, the device responds with an Abort SDO Transfer. If a wrong signature is written, the device refuses to restore the defaults and responds with an Abort SDO Transfer.

On read access to the appropriate sub-index the device provides information about its default parameter restoring capability with the following format:

|      |               |     |
|------|---------------|-----|
|      | UNSIGNED32    |     |
|      | MSB           | LSB |
| bits | 31 .. 1       | 0   |
|      | reserved (=0) | 0/1 |

Tab. 117 NMT\_RestoreDefParam\_REC restoring default values read access structure

| bit number | value | meaning                                    |
|------------|-------|--|
| 31 .. 1    | 0     | reserved (=0)                              |
| 0          | 0     | Device does not restore default parameters |
|            | 1     | Device restores parameters                 |

Tab. 118 NMT\_RestoreDefParam\_REC structure of restore read access

- **Validation Information**

Following a restore-default-parameter SDO command the objects initially keep their current values and are set to their default values after the device is reset (NMTResetNode for sub-index 01<sub>h</sub>, 03<sub>h</sub> .. 7F<sub>h</sub>, NMTResetCommunicationsufficient for sub-index 2<sub>h</sub>) or power cycled.

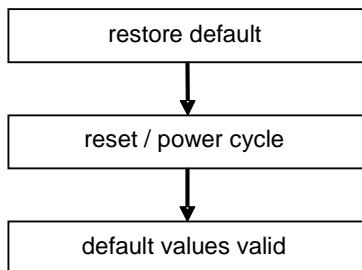


Fig. 75. NMT\_RestoreDefParam\_REC restore procedure

## 7.2.1.3 Communication Interface Description

### 7.2.1.3.1 Object 1F93<sub>h</sub>: NMT\_EPLNodeID\_REC

The object stores the device's POWERLINK Node ID.

|           |                    |             |        |
|-----------|--------------------|-------------|--------|
| Index     | 1F93 <sub>h</sub>  | Object Type | RECORD |
| Name      | NMT_EPLNodeID_REC  |             |        |
| Data Type | NMT_EPLNodeID_TYPE | Category    | M      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |  |             |       |
|---------------|--|-------------|-------|
| Sub-Index     | 00 <sub>h</sub>                                  |             |       |
| Name          | NumberOfEntries                                  |             |       |
| Value Range   | 2,3  | Access      | const |
| Default Value | depends on presence of Sub-Index 03 <sub>h</sub> | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: NodeID\_U8**

|               |                    |             |    |
|---------------|--------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>    |             |    |
| Name          | NodeID_U8          |             |    |
| Data Type     | UNSIGNED8          | Category    | M  |
| Value Range   | 1 .. 240, 253, 254 | Access      | ro |
| Default Value | 1                  | PDO Mapping | No |

The sub-index holds the device's actual Node ID. NodeID\_U8 may be provided by hardware settings (dip switch etc.) or set up by software.

- **Sub-Index 02<sub>h</sub>: NodeIDByHW\_BOOL**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub> |             |    |
| Name          | NodeIDByHW_BOOL |             |    |
| Data Type     | BOOLEAN         | Category    | M  |
| Value Range   | BOOLEAN         | Access      | ro |
| Default Value | -               | PDO Mapping | No |

The sub-index displays the Node ID setup mode of the device. It shall be setup during system initialisation.

- On devices, that setup the POWERLINK Node ID exclusively by HW, the object sub-index be set to TRUE.
- On devices, that setup the POWERLINK Node ID exclusively by SW, the object sub-index be set to FALSE.

- On devices, that enable SW POWERLINK Node ID setup by a special HW setup, the sub-index shall be set to FALSE, if POWERLINK Node ID setup by SW is enabled.  
If SW POWERLINK Node ID setup is enabled by a Node ID HW switch, it's recommended to use Node ID setup = 0.

The ability to define the POWERLINK Node ID by SW shall be indicated in the object dictionary entry NMT\_FeatureFlags\_U32 Bit 10 and in the device description by D\_NMT\_NodeIDBySW\_BOOL. HW setup ability is indicated in the device description by D\_NMT\_NodeIDByHW\_BOOL .

- Sub-Index 03<sub>h</sub>: SWNodeID\_U8**

|               |                    |             |                         |
|---------------|--------------------|-------------|-------------------------|
| Sub-Index     | 03 <sub>h</sub>    |             |                         |
| Name          | SWNodeID_U8        |             |                         |
| Data Type     | UNSIGNED8          | Category    | Cond                    |
| Value Range   | 1 .. 240, 253, 254 | Access      | cond,<br>valid on reset |
| Default Value | 1                  | PDO Mapping | No                      |

The sub-index may be used to setup the Node ID by SW.

If the device supports Node ID setup by SW (NMT\_FeatureFlags\_U32 Bit 10 is set and D\_NMT\_NodeIDBySW\_BOOL is true), the sub-index shall be mandatory. Access shall be rws. Activation of the setting shall be indicated by sub\_index 02<sub>h</sub>.

If the device does not support Node ID setup by SW, the sub-index shall be optional. If implemented, access shall be ro.

### 7.2.1.3.2 Object 1030<sub>h</sub> .. 1039<sub>h</sub> : NMT\_InterfaceGroup\_Xh\_REC

The following objects are used to configure and retrieve parameters of the network interfaces (physical or virtual) via SDO. Each interface has one entry. The InterfaceGroup\_REC object is a subset of the Interface Group RFC1213.

POWERLINK interfaces shall be described by the low order objects (e.g. 1030<sub>h</sub>, 1031<sub>h</sub>, ...).

|           |  |             |   |
|-----------|--|-------------|---|
| Index     | 1030 <sub>h</sub> .. 1039 <sub>h</sub> | Object Type | RECORD  |
| Name      | NMT_InterfaceGroup_Xh_REC              |             |   |
| Data Type | NMT_InterfaceGroup_Xh_TYPE             | Category    | 1030 <sub>h</sub> : M<br>1031 <sub>h</sub> .. 1039 <sub>h</sub> : O |

To allow access by name “\_Xh” is replaced with a name index. For example, the name index is “\_0h” if the object index is 1030<sub>h</sub>. The name index is incremented up to “\_9h” corresponding to object index 1039<sub>h</sub>.

- Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 9               | Access      | const |
| Default Value | 9               | PDO Mapping | No    |

- Sub-Index 01<sub>h</sub>: InterfaceIndex\_U16**

|               |                    |             |    |
|---------------|--------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>    |             |    |
| Name          | InterfaceIndex_U16 |             |    |
| Data Type     | UNSIGNED16         | Category    | M  |
| Value Range   | 1..10              | Access      | ro |
| Default Value | -                  | PDO Mapping | No |

Interface index of the physical interface. This number is the index number subtracted by 102F<sub>h</sub>. The POWERLINK node that adds an interface generates the respective value.

The interface identified by a particular value of this index is the same interface as identified by the same value of NWL\_IpAddrTable\_Xh\_REC.IfIndex\_U16.

- **Sub-Index 02<sub>h</sub>: InterfaceDescription\_VSTR**

|               |                           |             |       |
|---------------|---------------------------|-------------|-------|
| Sub-Index     | 02 <sub>h</sub>           |             |       |
| Name          | InterfaceDescription_VSTR |             |       |
| Data Type     | VISIBLE_STRING            | Category    | M     |
| Value Range   | -                         | Access      | const |
| Default Value | -                         | PDO Mapping | No    |

A textual string containing information about the interface. This string should include the name of the manufacturer, the product name and the version of the hardware interface.

The value shall be setup by the device firmware during system initialisation.

- **Sub-Index 03<sub>h</sub>: InterfaceType\_U8**

|               |   |             |       |
|---------------|---|-------------|-------|
| Sub-Index     | 03 <sub>h</sub>   |             |       |
| Name          | InterfaceType_U8  |             |       |
| Data Type     | UNSIGNED8   | Category    | M     |
| Value Range   | 1 – other<br>6 – ethernet-csmacd<br>7 – iso88023-csmacd<br>see RFC1213 Interface Group object if Type for further numbers | Access      | const |
| Default Value | 6 – ethernet-csmacd   | PDO Mapping | No    |

The type of interface, distinguished according to the physical/link protocol(s) immediately 'below' the network layer in the protocol stack.

The value shall be setup by the device firmware during system initialisation.

- **Sub-Index 04<sub>h</sub>: InterfaceMtu\_U16**

|               |                  |             |       |
|---------------|------------------|-------------|-------|
| Sub-Index     | 04 <sub>h</sub>  |             |       |
| Name          | InterfaceMtu_U16 |             |       |
| Data Type     | UNSIGNED16       | Category    | M     |
| Value Range   | UNSIGNED16       | Access      | const |
| Default Value | -                | PDO Mapping | No    |

The size of the largest datagram which can be sent/received on the interface, specified in octets. For interfaces that are used for transmitting network datagrams, this is the size of the largest network datagram that can be sent on the interface.

The value shall be setup by the device firmware during system initialisation.

- **Sub-Index 05<sub>h</sub>: InterfacePhysAddress\_OSTR**

|               |                           |             |       |
|---------------|---------------------------|-------------|-------|
| Sub-Index     | 05 <sub>h</sub>           |             |       |
| Name          | InterfacePhysAddress_OSTR |             |       |
| Data Type     | OCTET_STRING              | Category    | M     |
| Value Range   | -                         | Access      | const |
| Default Value | -                         | PDO Mapping | No    |

The interface's address at the protocol layer immediately 'below' the network layer in the protocol stack, e.g. the MAC address of an Ethernet interface. For interfaces which do not have such an address (e.g. a serial line), this object should contain an octet string of zero length.

The value shall be setup by the device firmware during system initialisation. It shall be read from the hardware.

- **Sub-Index 06<sub>h</sub>: InterfaceName\_VSTR**

|               |                    |             |    |
|---------------|--------------------|-------------|----|
| Sub-Index     | 06 <sub>h</sub>    |             |    |
| Name          | InterfaceName_VSTR |             |    |
| Data Type     | VISIBLE_STRING     | Category    | M  |
| Value Range   | -                  | Access      | ro |
| Default Value | -                  | PDO Mapping | No |

A user reference name for the interface. This name shall be the name used by the device driver to access the interface (e.g. for Linux “eth0”).

- **Sub-Index 07<sub>h</sub>: InterfaceOperStatus\_U8**

|               |                        |             |    |
|---------------|------------------------|-------------|----|
| Sub-Index     | 07 <sub>h</sub>        |             |    |
| Name          | InterfaceOperStatus_U8 |             |    |
| Data Type     | UNSIGNED8              | Category    | M  |
| Value Range   | 0 – Down<br>1 – Up     | Access      | ro |
| Default Value | -                      | PDO Mapping | No |

The current operational state of the interface.

- **Sub-Index 08<sub>h</sub>: InterfaceAdminState\_U8**

|               |                        |             |     |
|---------------|------------------------|-------------|-----|
| Sub-Index     | 08 <sub>h</sub>        |             |     |
| Name          | InterfaceAdminState_U8 |             |     |
| Data Type     | UNSIGNED8              | Category    | M   |
| Value Range   | 0 – Down<br>1 – Up     | Access      | rws |
| Default Value | Up                     | PDO Mapping | No  |

The current administration state (Down/Up) of the interface.

The value shall not be set to “Down” if this is the only interface that is “Up”.

- **Sub-Index 09<sub>h</sub>: Valid\_BOOL**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 09 <sub>h</sub> |             |     |
| Name          | Valid_BOOL      |             |     |
| Data Type     | BOOLEAN         | Category    | M   |
| Value Range   | BOOLEAN         | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

Specifies whether or not the data of this object is valid. If the value is TRUE the data of this object is valid. If the value is FALSE the data of this object is invalid.

- **Sub-Index 0A<sub>h</sub>:**

*Used by EPSG DS302-E [5]*

### 7.2.1.3.3 Object 1F9A<sub>h</sub>: NMT\_HostName\_VSTR

Provides the node’s DNS hostname.

The Object shall be supported only if IP is supported by the device. (refer 5.1)

|               |                   |             |      |
|---------------|-------------------|-------------|------|
| Index         | 1F9A <sub>h</sub> | Object Type | VAR  |
| Name          | NMT_HostName_VSTR |             |      |
| Data Type     | VISIBLE_STRING32  | Category    | Cond |
| Value Range   | see 5.1.4         | Access      | rws  |
| Default Value | -                 | PDO Mapping | No   |

## 7.2.1.4 Node List

### 7.2.1.4.1 Object 1F81<sub>h</sub>: NMT\_NodeAssignment\_AU32

This object assigns nodes to the NMT Master (MN).

On the CN the object is conditional. It shall be supplied if one of the following the ARRAY type objects are implemented by the CN: NMT\_MultiCycleAssign\_AU8, NMT\_ConsumerHeartbeatTime\_AU32, or NMT\_PresPayloadLimitList\_AU16.

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index.

Available Node IDs may be restricted by the device description entries D\_NMT\_MaxCNNNumber\_U8 and D\_NMT\_MaxCNNodeID\_U8 (cf 4.5). Sub-Indices corresponding to invalid Node IDs shall be set to 0.

The sub-index equal to the MN's Node ID C\_ADR\_MN\_DEF\_NODE\_ID shall represent the MN.

The object should be set by the system configuration. It should be equal on all nodes providing this object.

On the MN, the object controls the identification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1, the identification process shall be restarted.

|           |                         |             |                   |
|-----------|-------------------------|-------------|-------------------|
| Index     | 1F81 <sub>h</sub>       | Object Code | ARRAY             |
| Name      | NMT_NodeAssignment_AU32 |             |                   |
| Data Type | UNSIGNED32              | Category    | MN: M<br>CN: Cond |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: NodeAssignment**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | NodeAssignment                     |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | Bit field, see below               | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

- NodeAssignment Value Interpretation

| Octet | Bit <sup>23</sup> | Value          | Description   | Property | Evaluate |
|-------|-------------------|----------------|---|----------|----------|
| 0     | 0                 | 0 <sub>b</sub> | Node with this ID does not exist, Bits 1 to 30 above are not used.  | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | Node with this ID exists.   |          |          |
| 1     | 0 <sub>b</sub>    | 0 <sub>b</sub> | Node with this ID is not a CN, Bits 2 .. 7, 9, 13 .. 30 are not used.   | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | Node with this ID is a CN. After configuration (with Configuration Manager) the Node will be set to state NMT_CS_OPERATIONAL.   |          |          |
| 2     | 0 <sub>b</sub>    | 0 <sub>b</sub> | On detection of a booting CN inform the application but do NOT automatically configure and start the node.  | CN       | MN       |
|       |                   | 1 <sub>b</sub> | On detection of a booting CN inform the application and continue the process "START_CN" (see 7.4.2.2.4)   |          |          |
| 3     | 0 <sub>b</sub>    | 0 <sub>b</sub> | Optional CN.  | CN       | MN, CN   |
|       |                   | 1 <sub>b</sub> | Mandatory CN.   |          |          |
| 4     | 0 <sub>b</sub>    | 0 <sub>b</sub> | The CN node may be reset by the NMTSwReset, NMTResetNode, NMTResetCommunication or NMTResetConfiguration command independent of its state. Hence no checking of its state needs to be performed prior to NMT Reset Communication. | CN       | MN       |
|       |                   | 1 <sub>b</sub> | MN must not send any of the reset commands listed above to this node if it notices that the CN is in NMT_CS_OPERATIONAL state.  |          |          |
| 5     | 0 <sub>b</sub>    | 0 <sub>b</sub> | Application software version verification for this node is not required.  | CN       | MN       |
|       |                   | 1 <sub>b</sub> | Application software version verification for this node is required.  |          |          |
| 6     | 0 <sub>b</sub>    | 0 <sub>b</sub> | Automatic application software update (download) is not allowed.  | CN       | MN       |
|       |                   | 1 <sub>b</sub> | Automatic application software update (download) is allowed.  |          |          |
| 7     | ---               | ---            | Reserved (0 <sub>b</sub> ).   | ---      | ---      |
| 1     | 8                 | 0 <sub>b</sub> | Isochronously accessed node   | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | AsyncOnly node  |          |          |
| 9     | 0 <sub>b</sub>    | 0 <sub>b</sub> | continuously accessed CN  | CN       | MN, CN   |
|       |                   | 1 <sub>b</sub> | multiplexed CN  |          |          |
| 10    | 0 <sub>b</sub>    | 0 <sub>b</sub> | device is not a Router Type 1   | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | device is a Router Type 1   |          |          |
| 11    | 0 <sub>b</sub>    | 0 <sub>b</sub> | device is not a Router Type 2   | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | device is a Router Type 2   |          |          |
| 12    | 0 <sub>b</sub>    | 0 <sub>b</sub> | MN does not transmit PRes   | MN       | MN, CN   |
|       |                   | 1 <sub>b</sub> | MN transmits PRes   |          |          |
| 13    | ---               | ---            | Reserved (0 <sub>b</sub> ), used by EPSG DS302-B [2]  | ---      | ---      |
| 14    | ---               | ---            | Reserved (0 <sub>b</sub> ), used by EPSG DS302-C [3]  | ---      | ---      |
| 15    | ---               | ---            | Reserved (0 <sub>b</sub> ), used by EPSG DS302-E [5]  | ---      | ---      |
| 2     | 16 .. 23          | ---            | Reserved (00 <sub>h</sub> ).  | ---      | ---      |
| 3     | 24 .. 30          | ---            | Reserved (000 0000 <sub>b</sub> ).  | ---      | ---      |

<sup>23</sup> Bit 0 and further bits may be used to control allocation of memory for depending OD entries

| Octet | Bit <sup>23</sup> | Value          | Description           | Property | Evaluate |
|-------|-------------------|----------------|-----------------------|----------|----------|
|       | 31                | 0 <sub>b</sub> | Bit 0 .. 30 not valid | MN, CN   | MN, CN   |
|       |                   | 1 <sub>b</sub> | Bit 0 .. 30 valid     |          |          |

Tab. 119 NMT\_NodeAssignment\_AU32 interpretation

Bits that control a feature that is not supported by the implementation are ro.

Bit 31 may be used to control memory allocation for Node list array sub-indices located on the MN and the CN.

## 7.2.1.5 Timing

The indices described by this paragraph control the timing behavior of the POWERLINK network traffic. They are common for MN and CN as well. Additional values, that are provided by the MN implementation only, are listed in 7.2.2.3.

### 7.2.1.5.1 Object 1006<sub>h</sub>: NMT\_CycleLen\_U32

This object defines the communication cycle time interval in  $\mu$ s. This period defines the SYNC interval. The object should be set by the system configuration.

|               |                   |             |                     |
|---------------|-------------------|-------------|---------------------|
| Index         | 1006 <sub>h</sub> | Object Type | VAR                 |
| Name          | NMT_CycleLen_U32  |             |                     |
| Data Type     | UNSIGNED32        | Category    | M                   |
| Value Range   | refer below       | Access      | rws, valid on reset |
| Default Value | -                 | PDO Mapping | No                  |

Communication cycle period setup values shall be limited by the device description entries D\_NMT\_CycleTimeMin\_U32 and D\_NMT\_CycleTimeMax\_U32. Both limits shall be multiples of D\_NMT\_CycleTimeGranularity\_U32. Between the limits, values may be taken from the continuum (D\_NMT\_CycleTimeGranularity\_U32 = 1) or stepwise setup may be applied (D\_NMT\_CycleTimeGranularity\_U32 > 1).

To avoid incompatible step sizes that lead to a huge cycle time, D\_NMT\_CycleTimeGranularity\_U32 should be a multiple of the two base granularities 100  $\mu$ s or 125  $\mu$ s.

### 7.2.1.5.2 Object 1F98<sub>h</sub>: NMT\_CycleTiming\_REC

NMT\_CycleTiming\_REC provides node specific timing parameters, that influence the POWERLINK cycle timing.

|           |                      |             |        |
|-----------|----------------------|-------------|--------|
| Index     | 1F98 <sub>h</sub>    | Object Type | RECORD |
| Name      | NMT_CycleTiming_REC  |             |        |
| Data Type | NMT_CycleTiming_TYPE | Category    | M      |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                   |             |       |
|---------------|-------------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub>   |             |       |
| Name          | NumberOfEntries   |             |       |
| Value Range   | MN: 9; CN: 8 .. 9 | Access      | const |
| Default Value | MN: 9, CN: -      | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: IsochrTxMaxPayload\_U16**

|               |                             |             |       |
|---------------|-----------------------------|-------------|-------|
| Sub-Index     | 01 <sub>h</sub>             |             |       |
| Name          | IsochrTxMaxPayload_U16      |             |       |
| Data Type     | UNSIGNED16                  | Category    | M     |
| Value Range   | 36 .. C_DLL_ISOCHR_MAX_PAYL | Access      | const |
| Default Value | -                           | PDO Mapping | No    |

Provides the device specific upper limit for payload data size in octets of isochronous messages to be transmitted by the device.

On all nodes, the sub-index limits the size of the PRes frame issued by the node (sub-index PResActPayloadLimit\_U16, refer below). Additionally on the MN, the size of transmitted PReq messages (object NMT\_MNPReqPayloadLimitList\_AU16) is affected.

The limit shall be setup by the device firmware during system initialisation.

- **Sub-Index 02<sub>h</sub>: IsochrRxMaxPayload\_U16**

|               |                             |             |       |
|---------------|-----------------------------|-------------|-------|
| Sub-Index     | 02 <sub>h</sub>             |             |       |
| Name          | IsochrRxMaxPayload_U16      |             |       |
| Data Type     | UNSIGNED16                  | Category    | M     |
| Value Range   | 36 .. C_DLL_ISOCHR_MAX_PAYL | Access      | const |
| Default Value | -                           | PDO Mapping | No    |

Provides the device specific upper limit for payload data size in octets of isochronous messages to be received by the device.

On all nodes, the sub-index limits the size of the PRes frames received by the node (object NMT\_PResPayloadLimitList\_AU16, see 7.2.1.5.5). Additionally on the CN, the size of the received PReq message (sub-index PReqActPayloadLimit\_U16, refer below) is affected.

The limit shall be setup by the device firmware during system initialisation.

- **Sub-Index 03<sub>h</sub>: PResMaxLatency\_U32**

|               |                    |             |                |
|---------------|--------------------|-------------|----------------|
| Sub-Index     | 03 <sub>h</sub>    |             |                |
| Name          | PResMaxLatency_U32 |             |                |
| Data Type     | UNSIGNED32         | Category    | CN: M<br>MN: - |
| Value Range   | UNSIGNED32         | Access      | const          |
| Default Value | -                  | PDO Mapping | No             |

Provides the maximum time in ns, that is required by the CN to respond to PReq.

The value shall be setup by the device firmware during system initialisation.

- **Sub-Index 04<sub>h</sub>: PReqActPayloadLimit\_U16**

|               |                                 |             |                        |
|---------------|---------------------------------|-------------|------------------------|
| Sub-Index     | 04 <sub>h</sub>                 |             |                        |
| Name          | PReqActPayloadLimit_U16         |             |                        |
| Data Type     | UNSIGNED16                      | Category    | CN: M<br>MN: -         |
| Value Range   | 36 .. sub-index 02 <sub>h</sub> | Access      | rws, valid on reset *) |
| Default Value | -                               | PDO Mapping | No                     |

Provides the configured PReq payload data slot size in octets expected by the CN. The payload data slot size plus headers gives the size of the PReq frame. The data slot may be filled by PDO data up to this limit.

*Note: This results in a fixed frame size regardless of the size of PDO data used.*

\*) The current value of PReqActPayloadLimit\_U16 shall be used for checking the mapping at activation.

*Note: The new mapping is active immediately. If the mapping is active but the mapped data are larger than frame size the received frame shall be ignored. See 6.4.8.2*

*However the frame size shall change to the current version of PReqActPayloadLimit\_U16 after a reset configuration only.*

The MN holds a list of node specific PReq payload data slot values to be transmitted in object NMT\_MNPReqPayloadLimitList\_AU16.

- **Sub-Index 05<sub>h</sub>: PResActPayloadLimit\_U16**

|               |                                 |             |                        |
|---------------|---------------------------------|-------------|------------------------|
| Sub-Index     | 05 <sub>h</sub>                 |             |                        |
| Name          | PResActPayloadLimit_U16         |             |                        |
| Data Type     | UNSIGNED16                      | Category    | CN: M<br>MN: O         |
| Value Range   | 36 .. sub-index 01 <sub>h</sub> | Access      | rws, valid on reset *) |
| Default Value | -                               | PDO Mapping | No                     |

Provides the configured PRes payload data slot size in octets sent by the CN. The payload data slot size plus headers gives the size of the PRes frame. The data slot may be filled by PDO data up to this limit.

*Note: This results in a fixed frame size regardless of the size of PDO data used*

\*) The current value of PResActPayloadLimit\_U16 shall be used for checking the mapping at activation.

*Note: The new mapping is active immediately. If the mapping is active but the mapped data are larger than the frame size the RD flag shall be reset. See 6.4.8.2*

*However the frame size shall change to the current version of PResActPayloadLimit\_U16 after a reset configuration only.*

The PRes payload values expected to be received by the node are listed in object NMT\_PResPayloadLimitList\_AU16.

- **Sub-Index 06<sub>h</sub>: ASndMaxLatency\_U32**

|               |                    |             |                |
|---------------|--------------------|-------------|----------------|
| Sub-Index     | 06 <sub>h</sub>    |             |                |
| Name          | ASndMaxLatency_U32 |             |                |
| Data Type     | UNSIGNED32         | Category    | CN: M<br>MN: - |
| Value Range   | UNSIGNED32         | Access      | const          |
| Default Value | -                  | PDO Mapping | No             |

Provides the maximum time in ns, that is required by the CN to respond to SoA.

The value shall be setup by the device firmware during system initialisation.

- **Sub-Index 07<sub>h</sub>: MultiplCycleCnt\_U8**

|               |                    |             |                     |
|---------------|--------------------|-------------|---------------------|
| Sub-Index     | 07 <sub>h</sub>    |             |                     |
| Name          | MultiplCycleCnt_U8 |             |                     |
| Data Type     | UNSIGNED8          | Category    | M                   |
| Value Range   | UNSIGNED8          | Access      | rws, valid on reset |
| Default Value | 0                  | PDO Mapping | No                  |

This sub-index describes the length of the multiplexed cycle in multiples of the POWERLINK cycle.

The MultiplCycleCnt\_U8 value shall be upper limited by the MN's device description entry D\_NMT\_MNMultiplCycMax\_U8. It shall be equal in all nodes of the segment.

If MultiplCycleCnt\_U8 is zero, there is no support of multiplexed cycle on the network.

*Note: A value of one is a valid value despite the fact that this does not result in a multiplexing.*

- **Sub-Index 08<sub>h</sub>: AsyncMTU\_U16**

|               |                                  |             |                     |
|---------------|----------------------------------|-------------|---------------------|
| Sub-Index     | 08 <sub>h</sub>                  |             |                     |
| Name          | AsyncMTU_U16                     |             |                     |
| Data Type     | UNSIGNED16                       | Category    | M                   |
| Value Range   | C_DLL_MIN_ASYNC_MTU .. see below | Access      | rws, valid on reset |
| Default Value | C_DLL_MIN_ASYNC_MTU              | PDO Mapping | No                  |

This sub-index describes the maximum asynchronous frame size in octets. The value applies to ASnd frames as well as to UDP/IP and other legacy Ethernet type frames. For this reason the value describes the length of the complete Ethernet frame minus 14 octets Ethernet header and 4 octets checksum.

AsyncMTU\_U16 is upper limited by the NMT\_InterfaceGroup\_Xh\_REC.InterfaceMTU\_U16 values of all devices in the segment. This limit shall be 18 octets less than the minimum InterfaceMTU\_U16 value provided by any node in the segment. AsyncMTU\_U16 may grow up to C\_DLL\_MAX\_ASYNC\_MTU.

AsyncMTU\_U16 shall be equal in all nodes of the segment.

Sub-Index 08<sub>h</sub> shall be valid in all NMT states.

- **Sub-Index 09<sub>h</sub>: Prescaler\_U16**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 09 <sub>h</sub> |             |                     |
| Name          | Prescaler_U16   |             |                     |
| Data Type     | UNSIGNED16      | Category    | MN: M, CN: O        |
| Value Range   | 0, 1 .. 1000    | Access      | rws, valid on reset |
| Default Value | 2               | PDO Mapping | No                  |

This sub-index configures the toggle rate of the SoC PS flag. The value provides the number of cycles that have to be completed to toggle the flag by the MN.

If Prescaler\_U16 is 0, there shall be no toggling of the SoC PS flag.

If Prescaler\_U16 is 1 the flag shall be toggled every cycle, if its value is 2 every 2<sup>nd</sup> cycle and so on.

Prescaler\_U16 shall be equal in all nodes of the segment.

- **Sub-Index 0A<sub>h</sub>:**

*Used by EPSG DS302-C [3]*

- **Sub-Index 0B<sub>h</sub>:**

*Used by EPSG DS302-C [3]*

- **Sub-Index 0C<sub>h</sub>:**  
*Used by EPSG DS302-C [3]*
- **Sub-Index 0D<sub>h</sub>:**  
*Used by EPSG DS302-C [3]*
- **Sub-Index 0E<sub>h</sub>:**  
*Used by EPSG DS302-C [3]*
- **Sub-Index 0F<sub>h</sub>:**  
*Used by EPSG DS302-E [5]*

### 7.2.1.5.3 Object 1F9B<sub>h</sub>: NMT\_MultiplCycleAssign\_AU8

This object assigns nodes to the particular POWERLINK cycles of the multiplexed cycle period defined by NMT\_CycleTiming\_REC.MultiplCycleCnt\_U8. The value shall be equal in all nodes of the segment.

|           |                            |             |       |
|-----------|----------------------------|-------------|-------|
| Index     | 1F9B <sub>h</sub>          | Object Code | ARRAY |
| Name      | NMT_MultiplCycleAssign_AU8 |             |       |
| Data Type | UNSIGNED8                  | Category    | Cond  |

The index shall be supported if the node supports Multiplexing (see NMT\_FeatureFlags\_U32, D\_DLL\_CNFeatureMultiplex\_BOOL resp. D\_DLL\_MNFeatureMultiplex\_BOOL) and is valid only if NMT\_CycleTiming\_REC.MultiplCycleCnt\_U8 is not zero.

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 0 .. 254        | Access      | rws, valid on reset |
| Default Value | -               | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CycleNo**

|               |   |             |                     |
|---------------|---|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub>               |             |                     |
| Name          | CycleNo   |             |                     |
| --            | --  | Category    | O                   |
| Value Range   | 0 .. NMT_CycleTiming_REC.<br>MultiplCycleCnt_U8 | Access      | rws, valid on reset |
| Default Value | 0   | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a multiplexed node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 9.

CycleNo defines the POWERLINK cycle index in the multiplexed cycle, when the respective node shall be accessed. If CycleNo is zero, the node shall be accessed continuously (see 4.2.4.4).

### 7.2.1.5.4 Object 1016<sub>h</sub>: NMT\_ConsumerHeartbeatTime\_AU32

The consumer heartbeat time defines the expected heartbeat cycle time (see 7.3.5). Monitoring starts after the reception of the first heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The time must be a multiple of 1ms.

|           |                                |             |       |
|-----------|--------------------------------|-------------|-------|
| Index     | 1016 <sub>h</sub>              | Object Type | ARRAY |
| Name      | NMT_ConsumerHeartbeatTime_AU32 |             |       |
| Data Type | UNSIGNED32                     | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                             |             |     |
|---------------|-----------------------------|-------------|-----|
| Sub-Index     | 00 <sub>h</sub>             |             |     |
| Name          | NumberOfEntries             |             |     |
| Value Range   | 1 .. D_NMT_MaxHeartbeats_U8 | Access      | rws |
| Default Value | -                           | PDO Mapping | No  |

Number of Entries, e.g. number of guard channels, may be limited by the device description entry D\_NMT\_MaxHeartbeats\_U8.

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: HeartbeatDescription**

|               |                                   |             |     |
|---------------|-----------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> – FE <sub>h</sub> |             |     |
| Name          | HeartbeatDescription              |             |     |
| --            | --                                | Category    | O   |
| Value Range   | UNSIGNED32                        | Access      | rws |
| Default Value | 0                                 | PDO Mapping | No  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub> Value Description**

|                   |                       |           |                |
|-------------------|-----------------------|-----------|----------------|
|                   | <b>UNSIGNED32</b>     |           |                |
|                   | <b>MSB</b>            |           | <b>LSB</b>     |
| <b>Bits</b>       | 31-24                 | 23-16     | 15-0           |
| <b>Value</b>      | reserved (value: 00h) | Node ID   | heartbeat time |
| <b>Encoded as</b> | -                     | UNSIGNED8 | UNSIGNED16     |

Tab. 120 HeartbeatDescription value interpretation

If an attempt is made to configure several non-zero consumer heartbeat times for the same Node ID the device aborts the SDO download with abort code E\_NMT\_INVALID\_HEARTBEAT.

### 7.2.1.5.5 Object 1F8D<sub>h</sub>: NMT\_PResPayloadLimitList\_AU16

This object holds a list of the expected PRes payload data slot size in octets for each configured node that is isochronously accessed, e.g. via PReq / PRes frame exchange. The payload data slot size is a measure for the configured size of the PRes frame. The data slot may be filled by PDO data up to this limit.

|           |                               |             |                |
|-----------|-------------------------------|-------------|----------------|
| Index     | 1F8D <sub>h</sub>             | Object Code | ARRAY          |
| Name      | NMT_PResPayloadLimitList_AU16 |             |                |
| Data Type | UNSIGNED16                    | Category    | MN: M<br>CN: O |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: PResPayloadLimit**

|               |  |                |                           |
|---------------|--|----------------|---------------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub>                   |                |                           |
| Name          | PResPayloadLimit                                     |                |                           |
| --            | --   | Category       | M                         |
| Value Range   | 36 .. C_DLL_ISOCHR_MAX_PAYL,<br>0, FFFF <sub>h</sub> | Access         | rws,<br>valid on<br>reset |
| Default Value | 36   | PDO<br>Mapping | No                        |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is an isochronous node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0 and 8.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID indicates the payload size of the PRes frame issued by the MN.

A CN shall support the object only if it listens to PRes messages issued by another node (cross traffic). If the value of NMT\_PResPayloadLimitList\_AU16 [Node ID] equals 0, the node does not listen to the PRes of that Node ID. If the value equals FFFF<sub>h</sub>, the PRes payload limit of this node equals NMT\_CycleTiming\_REC.IsochrRxMaxPayload\_U16.

Values should be equal on all nodes of the segment.

## 7.2.1.6 NMT Service Interface

### 7.2.1.6.1 Object 1F9E<sub>h</sub>: NMT\_ResetCmd\_U8

NMT\_ResetCmd\_U8 may be used to initiate the reset of a node.

|               |  |                |     |
|---------------|--|----------------|-----|
| Index         | 1F9E <sub>h</sub>  | Object Code    | VAR |
| Name          | NMT_ResetCmd_U8  |                |     |
| Data Type     | UNSIGNED8  | Category       | M   |
| Value Range   | NMTInvalidService, NMTRest.getNode,<br>NMTRestConfiguration, NMTRestCommunication,<br>NMTSwReset | Access         | rw  |
| Default Value | NMTInvalidService  | PDO<br>Mapping | No  |

Setting NMT\_ResetCmd\_U8 to the NMT Command ID NMTRestNode, NMTRestConfiguration, NMTRestCommunication or NMTSwReset (see App. 3.7) shall trigger the node internal generation of a respective NMT command to itself.

NMT\_ResetCmd\_U8 shall be automatically reset to NMTInvalidService by the node, when the reset has been completed.

On read access, NMT\_ResetCmd\_U8 will always show NMTInvalidService.

If applied in NMT\_CS\_EPL\_MODE or NMT\_MS\_EPL\_MODE, resets by NMT\_ResetCmd\_U8 may violate the NMT rules and stimulate DLL and NMT Guarding errors.

## 7.2.1.7 NMT Diagnostics

### 7.2.1.7.1 Object 1F8Ch: NMT\_CurrNMTState\_U8

The index holds the node's current NMT state.

| Index         | 1F8Ch  | Object Type | VAR |
|---------------|--|-------------|-----|
| Name          | NMT_CurrNMTState_U8                          |             |     |
| Data Type     | UNSIGNED8                                    | Category    | M   |
| Value Range   | see App. 3.6                                 | Access      | ro  |
| Default Value | NMT_CS_NOT_ACTIVE resp.<br>NMT_MS_NOT_ACTIVE | PDO Mapping | No  |

An overview list containing the NMT states of all nodes in the segment is provided by MN object NMT\_MNNodeCurrState\_AU8.

## 7.2.2 NMT MN Objects

The NMT Master provides services for controlling the network behavior of nodes. Only one NMT Master can exist in an Ethernet POWERLINK Network. In POWERLINK the NMT Master is located in the MN.

The NMT Master Control Settings activate the MN functions and define the boot behavior and error reactions.

### 7.2.2.1 MN Start Up Behavior

Hint: MN and CN startup timing should be well balanced. System power up sequence should be considered.

#### 7.2.2.1.1 Object 1F80h: NMT\_StartUp\_U32

This object configures the boot behavior of a device that is able to become the MN.

Object NMT\_StartUp\_U32 is a configuration object. Internal state transitions must not change this object.

The object should be set by the system configuration.

The object controls the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted.

| Index         | 1F80h                | Object Type | VAR                 |
|---------------|----------------------|-------------|---------------------|
| Name          | NMT_StartUp_U32      |             |                     |
| Data Type     | UNSIGNED32           | Category    | M                   |
| Value Range   | Bit field, see below | Access      | rws, valid on reset |
| Default Value | -                    | PDO Mapping | No                  |

- NMT\_StartUp\_U32 Value Interpretation**

| Octet | Bit     | Value | Description  |
|-------|---------|-------|--|
| 0     | 0       | ---   | Reserved ( $0_b$ ).  |
|       | 1       | $0_b$ | Start only explicitly assigned CNs (if Bit 3 = $0_b$ ).  |
|       |         | $1_b$ | Perform the service NMTStartNode with broadcast addressing (if Bit 3 = $0_b$ ).  |
|       | 2       | $0_b$ | Automatically enter state NMT_MS_OPERATIONAL.  |
|       |         | $1_b$ | Do not automatically enter state NMT_MS_OPERATIONAL. Application will decide when to enter the state.  |
|       | 3       | $0_b$ | Allow to start up the CNs (i.e. to send NMTStartNode).   |
|       |         | $1_b$ | Do not allow to send NMTStartNode; the application may start the CNs.  |
|       | 4       | $0_b$ | On error event from guarding a mandatory CN deal with the CN individually.   |
|       |         | $1_b$ | On error event from guarding a mandatory CN perform NMTRestNode with broadcast addressing. Refer to Bit 6 and NMT_NodeAssignment_AU32 Bit 3.         |
|       | 5       | ---   | Reserved ( $0_b$ ).  |
|       | 6       | $0_b$ | On error event from guarding a mandatory CN deal with the CN according to Bit 4.   |
|       |         | $1_b$ | On error event from guarding a mandatory CN send NMTRestNode with broadcast addressing. Ignore Bit 4.  |
|       | 7       | $0_b$ | Automatically enter state NMT_MS_PRE_OPERATIONAL_2.  |
|       |         | $1_b$ | Do not automatically enter NMT_MS_PRE_OPERATIONAL_2. Application will decide when to enter the state.  |
| 1     | 8       | $0_b$ | Automatically enter state NMT_MS_READY_TO_OPERATE.   |
|       |         | $1_b$ | Do not automatically enter NMT_MS_READY_TO_OPERATE. Application will decide when to enter the state.   |
|       | 9       | $0_b$ | The identification of the CNs shall be limited to verification of the respective NMT_MNDeviceTypeList_AU32 sub-index.                                |
|       |         | $1_b$ | The identification of the CNs shall be completely checked.   |
|       | 10      | $0_b$ | The SW-Version of the CNs shall not be checked.  |
|       |         | $1_b$ | The SW-Version of the CNs shall be checked. If the check fails, the CN's SW has to be updated.   |
|       | 11      | $0_b$ | The Configuration of the CNs shall not be checked.   |
|       |         | $1_b$ | The Configuration of the CNs shall be checked. If the check fails, the CN's configuration has to be updated.   |
|       | 12      | $0_b$ | In case of error event return automatically from NMT_MS_OPERATIONAL to NMT_MS_PRE_OPERATIONAL_1.   |
|       |         | $1_b$ | Do not return to NMT_MS_PRE_OPERATIONAL_1. Application will decide whether to enter the state.   |
|       | 13      | $0_b$ | NMT_MS_EPL_MODE activation:<br>in NMT_MS_NOT_ACTIVE observe the network and change over to NMT_MS_PRE_OPERATIONAL_1 if there is no other MN detected |
|       |         | $1_b$ | NMT_MS_BASIC_ETHERNET released:<br>from NMT_MS_NOT_ACTIVE change over to NMT_MS_BASIC_ETHERNET   |
|       | 14      | ---   | Reserved ( $0_b$ ), used by EPSG DS302-A [1]   |
|       | 15      | ---   | Reserved ( $0_b$ )   |
| 2 – 3 | 16 – 31 | ---   | Reserved ( $00\ 00_h$ )  |

Tab. 121 NMT\_StartUp\_U32 interpretation

### 7.2.2.1.2 Object 1F89<sub>h</sub>: NMT\_BootTime\_REC

This object describes time interval values to be used by the MN when it starts the network.

It gives the maximum time, in  $\mu$ s, the master will wait for all mandatory CNs before signaling an error. If the time is zero (0), it will wait indefinitely.

*Hint: MN and CN startup timing should be well balanced. System power up sequence should be considered.*

|           |                   |             |        |
|-----------|-------------------|-------------|--------|
| Index     | 1F89 <sub>h</sub> | Object Code | RECORD |
| Name      | NMT_BootTime_REC  |             |        |
| Data Type | NMT_BootTime_TYPE | Category    | M      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 9               | Access      | const |
| Default Value | 9               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: MNWaitNotAct\_U32**

|               |                  |             |                     |
|---------------|------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub>  |             |                     |
| Name          | MNWaitNotAct_U32 |             |                     |
| Data Type     | UNSIGNED32       | Category    | M                   |
| Value Range   | >=250            | Access      | rws, valid on reset |
| Default Value | 1 000 000        | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall remain in state NMT\_MS\_NOT\_ACTIVE and listen for POWERLINK frames on the network before it changes over to NMT\_MS\_PRE\_OPERATIONAL\_1.

- **Sub-Index 02h: MNTtimeoutPreOp1\_U32**

|               |                       |             |                     |
|---------------|-----------------------|-------------|---------------------|
| Sub-Index     | 02 <sub>h</sub>       |             |                     |
| Name          | MNTtimeoutPreOp1_U32  |             |                     |
| Data Type     | UNSIGNED32            | Category    | M                   |
| Value Range   | 0, 50 000 – 5 000 000 | Access      | rws, valid on reset |
| Default Value | 500 000               | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall wait in state NMT\_MS\_PRE\_OPERATIONAL\_1 for all mandatory CNs to be identified via the IdentRequest / IdentResponse mechanism before it signals an error to the application.

If the timeout value is zero (0), there shall be no timeout for CN identification.

- **Sub-Index 03<sub>h</sub>: MNWaitPreOp1\_U32**

|               |                       |             |                     |
|---------------|-----------------------|-------------|---------------------|
| Sub-Index     | 03 <sub>h</sub>       |             |                     |
| Name          | MNWaitPreOp1_U32      |             |                     |
| Data Type     | UNSIGNED32            | Category    | O                   |
| Value Range   | 0, 50 000 – 5 000 000 | Access      | rws, valid on reset |
| Default Value | 500 000               | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall remain in state NMT\_MS\_PRE\_OPERATIONAL\_1.

If the wait value is zero (0), NMT\_MS\_PRE\_OPERATIONAL\_1 shall be left as soon as all mandatory CNs have been identified.

- **Sub-Index 04<sub>h</sub>: MNTimeoutPreOp2\_U32**

|               |                     |             |                     |
|---------------|---------------------|-------------|---------------------|
| Sub-Index     | 04 <sub>h</sub>     |             |                     |
| Name          | MNTimeoutPreOp2_U32 |             |                     |
| Data Type     | UNSIGNED32          | Category    | M                   |
| Value Range   | UNSIGNED32          | Access      | rws, valid on reset |
| Default Value | 500 000             | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall wait in state NMT\_MS\_PRE\_OPERATIONAL\_2 for all mandatory CNs to complete the initialisation of the error signaling and to be in state NMT\_CS\_READY\_TO\_OPERATE before it signals an error to the application.

For all optional CNs this sub-index describes the time interval in  $\mu$ s that the MN shall wait after sending the NMT command NMTEnableReadyToOperate to a CN before BOOT\_STEP2 returns with an error.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- **Sub-Index 05<sub>h</sub>: MNTimeoutReadyToOp\_U32**

|               |                        |             |                     |
|---------------|------------------------|-------------|---------------------|
| Sub-Index     | 05 <sub>h</sub>        |             |                     |
| Name          | MNTimeoutReadyToOp_U32 |             |                     |
| Data Type     | UNSIGNED32             | Category    | M                   |
| Value Range   | 0, 50 000 – 5 000 000  | Access      | rws, valid on reset |
| Default Value | 500 000                | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall wait in state NMT\_MS\_READY\_TO\_OPERATE for all mandatory CNs to be in state NMT\_CS\_OPERATIONAL before it signals an error to the application.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- **Sub-Index 06<sub>h</sub>: MNIdentificationTimeout\_U32**

|               |                             |             |                     |
|---------------|-----------------------------|-------------|---------------------|
| Sub-Index     | 06 <sub>h</sub>             |             |                     |
| Name          | MNIdentificationTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32                  | Category    | O                   |
| Value Range   | UNSIGNED32                  | Access      | rws, valid on reset |
| Default Value | 500 000                     | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall wait in the CHECK\_IDENTIFICATION state of the boot process until a device must be able to reply to an Ident Request message before an error is signaled to the application.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- **Sub-Index 07<sub>h</sub>: MNSoftwareTimeout\_U32**

|               |                       |             |                     |
|---------------|-----------------------|-------------|---------------------|
| Sub-Index     | 07 <sub>h</sub>       |             |                     |
| Name          | MNSoftwareTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32            | Category    | O                   |
| Value Range   | UNSIGNED32            | Access      | rws, valid on reset |
| Default Value | 500 000               | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu$ s that the MN shall wait after the software update in the CHECK\_SOFTWARE state until a device must be able to reply to an Ident Request message before an error is signaled to the application.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- Sub-Index 08<sub>h</sub>: MNConfigurationTimeout\_U32

|               |                            |             |                     |
|---------------|----------------------------|-------------|---------------------|
| Sub-Index     | 08h                        |             |                     |
| Name          | MNConfigurationTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32                 | Category    | O                   |
| Value Range   | UNSIGNED32                 | Access      | rws, valid on reset |
| Default Value | 500 000                    | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu\text{s}$  that the MN shall wait after the configuration of a CN was updated in the CHECK\_CONFIGURATION state until a device must be able to reply to an Ident Request message before an error is signaled to the application.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- **Sub-Index 09<sub>h</sub>: MNStartCNTTimeout\_U32**

|               |                       |             |                     |
|---------------|-----------------------|-------------|---------------------|
| Sub-Index     | 09 <sub>h</sub>       |             |                     |
| Name          | MNStartCNTTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32            | Category    | O                   |
| Value Range   | UNSIGNED32            | Access      | rws, valid on reset |
| Default Value | 500 000               | PDO Mapping | No                  |

This sub-index describes the time interval in  $\mu\text{s}$  that the MN shall wait for a CN in the START\_CN state to switch over to NMT\_CS\_OPERATIONAL before an error is signaled to the application.

If the timeout value is zero (0), there shall be no timeout, i.e. the MN will wait indefinitely.

- ### • Sub-Index 0A<sub>h</sub>

Used by EPSG DS302-A [1]

- ### • Sub-Index 0B<sub>h</sub>

Used by EPSG DS302-A [1]

- Sub-Index 0CA<sub>b</sub>

Used by EPSG DS302-A [1]

### **7.2.2.2 NMT Master Network Node Lists**

The Network List consists of objects that give information about which CNs must be managed, how they should be booted and information concerning requested actions on Error events.

### 7.2.2.2.1 Object 1F84<sub>b</sub>: NMT MNDeviceTypeList AU32

This object holds a list of the expected NMT DeviceType Id U32 value for each configured CN.

The object should be set by the system configuration. It shall be filled with the NMT\_DeviceType\_U32 object dictionary entry of the respective device.

It may be used by the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted.

|           |                             |             |       |
|-----------|-----------------------------|-------------|-------|
| Index     | 1F84 <sub>h</sub>           | Object Code | ARRAY |
| Name      | NMT_MNDeviceTypeIdList_AU32 |             |       |
| Data Type | UNSIGNED32                  | Category    | M     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |  |
|---------------|-----------------|-------------|---------------------|--|
| Sub-Index     | 00 <sub>h</sub> |             |                     |  |
| Name          | NumberOfEntries |             |                     |  |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |  |
| Default Value | 254             | PDO Mapping | No                  |  |

- **Sub-Index 01<sub>h</sub> .. FE<sub>h</sub>: CNDeviceTypeld**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNDeviceTypeld                     |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID may represent the MN.

On Boot-Up, the CN's NMT\_DeviceType\_U32 value is reported to the MN via IdentResponse. The MN shall compare the received value with the respective CNDeviceTypeld sub-index value. The Boot-Up for that device is only continued when the two values are equal.

If the value in CNDeviceTypeld is 0, this read access only gives information about the mere existence of a device with this Node ID. There is no comparison to the reported NMT\_DeviceType\_U32 value.

For multi-device-modules the application may perform additional checks.

### 7.2.2.2.2 Object 1F85<sub>h</sub>: NMT\_MNVendorIdList\_AU32

This object holds a list of the expected NMT\_IdentityObject\_REC.VendorId\_U32 value for each configured CN.

The object should be set by the system configuration. It shall be filled with the NMT\_IdentityObject\_REC.VendorId\_U32 object dictionary entry of the respective device.

It may be used by the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted.

|           |                         |             |       |
|-----------|-------------------------|-------------|-------|
| Index     | 1F85 <sub>h</sub>       | Object Code | ARRAY |
| Name      | NMT_MNVendorIdList_AU32 |             |       |
| Data Type | UNSIGNED32              | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNVendorId**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNVendorId                         |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID may represent the MN.

On Boot-Up, the CN's NMT\_IdentityObject\_REC.VendorId\_U32 value is reported to the MN via IdentResponse. The MN compares the received value with the respective CNVendorId sub-index value. The Boot-Up for that device is only continued when the two values are equal.

If the value in CNVendorId is 0, this read access only gives information about the mere existence of a device with this Node ID. There is no comparison with the reported NMT\_IdentityObject\_REC.VendorId\_U32 value.

For multi-device-modules the application may perform additional checks.

### 7.2.2.3 Object 1F86<sub>h</sub>: NMT\_MNProductCodeList\_AU32

This object holds a list of the expected NMT\_IdentityObject\_REC.ProductCode\_U32 value for each configured CN.

The object should be set by the system configuration. It shall be filled with the D\_NMT\_ProductCode\_U32 device description entry of the respective device.

It may be used by the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted

|           |                            |             |       |
|-----------|----------------------------|-------------|-------|
| Index     | 1F86 <sub>h</sub>          | Object Code | ARRAY |
| Name      | NMT_MNProductCodeList_AU32 |             |       |
| Data Type | UNSIGNED32                 | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNProductCode**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNProductCode                      |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID may represent the MN.

On Boot-Up, the CN's NMT\_IdentityObject\_REC.ProductCode\_U32 value is reported to the MN via IdentResponse. The MN compares the received value with the respective CNProductCode sub-index value. The Boot-Up for that device is only continued when the two values are equal.

If the value in CNProductCode is 0, this read access only gives information about the mere existence of a device with this Node ID. There is no comparison with the reported NMT\_IdentityObject\_REC.ProductCode\_U32 value.

For multi-device-modules the application may perform additional checks.

### 7.2.2.4 Object 1F87<sub>h</sub>: NMT\_MNRevisionNoList\_AU32

This object holds a list of the expected NMT\_IdentityObject\_REC.RevisionNo\_U32 value for each configured CN.

The object should be set by the system configuration. It shall be filled with the D\_NMT\_RevisionNo\_U32 device description entry of the respective device.

It may be used by the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted

|           |                           |             |       |
|-----------|---------------------------|-------------|-------|
| Index     | 1F87 <sub>h</sub>         | Object Code | ARRAY |
| Name      | NMT_MNRevisionNoList_AU32 |             |       |
| Data Type | UNSIGNED32                | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNRevisionNo**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNRevisionNo                       |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID may represent the MN.

On Boot-Up, the CN's NMT\_IdentityObject\_REC.RevisionNo\_U32 is reported to the MN via IdentResponse. The MN compares the received value with the respective CNRevisionNo sub-index value. The Boot-Up for that device is only continued when the two values are equal.

If the value in CNRevisionNo is 0, this read access only gives information about the mere existence of a device with this Node ID. There is no comparison with the reported NMT\_IdentityObject\_REC.RevisionNo\_U32 value.

For multi-device-modules the application may perform additional checks.

### 7.2.2.2.5 Object 1F88<sub>h</sub>: NMT\_MNSerialNoList\_AU32

This object holds a list of the expected NMT\_IdentityObject\_REC.SerialNo\_U32 value for each configured CN.

The object should be set by the system configuration.

It may be used by the verification of CNs. If the object is modified in state NMT\_MS\_PRE\_OPERATIONAL\_1 after start of verification, verification process shall be restarted

|           |                         |             |       |
|-----------|-------------------------|-------------|-------|
| Index     | 1F88 <sub>h</sub>       | Object Code | ARRAY |
| Name      | NMT_MNSerialNoList_AU32 |             |       |
| Data Type | UNSIGNED32              | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNSerialNo**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNSerialNo                         |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 0                                  | PDO Mapping | No                  |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID may represent the MN.

On Boot-Up, the CN's NMT\_IdentityObject\_REC.SerialNo\_U32 is reported to the MN via IdentResponse. The MN compares the received value with the respective CNSerialNo sub-index value. If the values are different a warning is issued.

If the value in CNSerialNo is 0, this read access only gives information about the mere existence of a device with this Node ID. There is no comparison with the reported NMT\_IdentityObject\_REC.SerialNo\_U32 value.

For multi-device-modules the application may perform additional checks.

### 7.2.2.3 Timing

The indices described by this paragraph shall be implemented by the MN only. They control the timing behavior of the POWERLINK network traffic. They are supplemental to the objects described by 7.2.1.4.

#### 7.2.2.3.1 Object 1F8A<sub>h</sub>: NMT\_MNCycleTiming\_REC

This object holds timing parameter use by the MN only to control the POWERLINK cycle.

|           |                        |             |        |
|-----------|------------------------|-------------|--------|
| Index     | 1F8A <sub>h</sub>      | Object Code | RECORD |
| Name      | NMT_MNCycleTiming_REC  |             |        |
| Data Type | NMT_MNCycleTiming_TYPE | Category    | M      |

- **Sub-Index 00h: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 1 .. 4          | Access      | ro |
| Default Value | -               | PDO Mapping | No |

- **Sub-Index 01h: WaitSoCPReq\_U32**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> |             |                     |
| Name          | WaitSoCPReq_U32 |             |                     |
| Data Type     | UNSIGNED32      | Category    | M                   |
| Value Range   | UNSIGNED32      | Access      | rws, valid on reset |
| Default Value | 1000            | PDO Mapping | No                  |

The sub-index provides a time interval in ns between end of SoC transmission and begin of the next frame transmission (PReq or PResMN). This wait interval enables the CN first accessed after SoC to complete its SoC handling.

WaitSoCPReq\_U32 shall be set to the maximum of D\_NMT\_MNSoC2PReq\_U32D\_NMT\_MNSoC2PReq\_U32 and the D\_NMT\_CNSoC2PReq\_U32 values of all configured isochronous CNs (cf. NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8).

WaitSoCPReq\_U32 handling may be implemented in 2 ways (see 4.2.4.3):

- a. delayed transmission of 1<sup>st</sup> PReq following SoC by MN, sub-index provides time interval between the end of the SoC transmission and the start of the first PReq
  - b. Transmission of the 1<sup>st</sup> PReq to the non existing node addressed by C\_ADR\_DUMMY\_NODE\_ID. The PRes frame receive timeout shall be set to WaitSoCPReq\_U32. Timeout error handling (see 4.7.6.2, 4.7.6.3) shall be disabled for this dummy message.
- **Sub-Index 02h: AsyncSlotTimeout\_U32**

|               |                      |             |                     |
|---------------|----------------------|-------------|---------------------|
| Sub-Index     | 02 <sub>h</sub>      |             |                     |
| Name          | AsyncSlotTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32           | Category    | O                   |
| Value Range   | >=250                | Access      | rws, valid on reset |
| Default Value | 100 000              | PDO Mapping | No                  |

The sub-index describes the worst case time interval in ns between the end of the SoA transmission and the begin of the reception of an ASnd frame issued by a CN.

- **Sub-Index 03h:**

*Used by EPSG DS302-B [2]*

- **Sub-Index 04h: MinRedCycleTime\_U32**

|               |                     |             |                     |
|---------------|---------------------|-------------|---------------------|
| Sub-Index     | 04 <sub>h</sub>     |             |                     |
| Name          | MinRedCycleTime_U32 |             |                     |
| Data Type     | UNSIGNED32          | Category    | O                   |
| Value Range   | UNSIGNED32          | Access      | rws, valid on reset |
| Default Value | -                   | PDO Mapping | No                  |

The minimum reduced cycle time (MinRedCycleTime\_U32) holds the minimum time between SoA frames in the reduced cycle.

MinRedCycleTime\_U32 shall be set to the maximum of D\_NMT\_MinRedCycleTime\_U32 of the CNs.

### 7.2.2.3.2 Object 1F8B<sub>h</sub>: NMT\_MNPReqPayloadLimitList\_AU16

This object holds a list of the PReq payload data slot size in octets for each configured node that is isochronously accessed, e.g. via PReq / PRes frame exchange. The payload data slot size is a measure for the configured size of the PReq frame. The data slot may be filled by PDO data up to this limit.

|           |                                 |             |       |
|-----------|---------------------------------|-------------|-------|
| Index     | 1F8B <sub>h</sub>               | Object Code | ARRAY |
| Name      | NMT_MNPReqPayloadLimitList_AU16 |             |       |
| Data Type | UNSIGNED16                      | Category    | M     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNPReqPayload**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNPReqPayload                      |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | 36 .. C_DLL_ISOCHR_MAX_PAYL        | Access      | rws, valid on reset |
| Default Value | 36                                 | PDO Mapping | No                  |

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

### 7.2.2.3.3 Object 1F92<sub>h</sub>: NMT\_MNCNPResTimeout\_AU32

This object holds a list of all configured CNs in [ns] with PollRequest to PollResponse timeouts (see 4.7.6.2, 4.7.6.3).

The object should be set by the system configuration.

|           |                          |             |       |
|-----------|--------------------------|-------------|-------|
| Index     | 1F92 <sub>h</sub>        | Object Code | ARRAY |
| Name      | NMT_MNCNPResTimeout_AU32 |             |       |
| Data Type | UNSIGNED32               | Category    | M     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: CNPResTimeout**

|               |                                    |             |                     |
|---------------|------------------------------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub> |             |                     |
| Name          | CNPResTimeout                      |             |                     |
| --            | --                                 | Category    | M                   |
| Value Range   | UNSIGNED32                         | Access      | rws, valid on reset |
| Default Value | 25 000                             | PDO Mapping | No                  |

Each sub-index in the array corresponds to a CN with the Node ID equal to the sub-index. The sub-index value is valid only if there is an isochronous CN assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1 and 8.

This parameter describes the POWERLINK node specific timeout values in ns. Whenever a PollRequest frame is sent to a CN this timer is started. (refer 4.7.6.1.1)

### 7.2.2.3.4 Object 1F9C<sub>h</sub>: NMT\_IsochrSlotAssign\_AU8

This object assigns nodes to a particular isochronous slot. The isochronous POWERLINK cycle can be divided into communication slots each consisting of PReq and PRes message for a particular node (see slot 1 to n in Fig. 76).

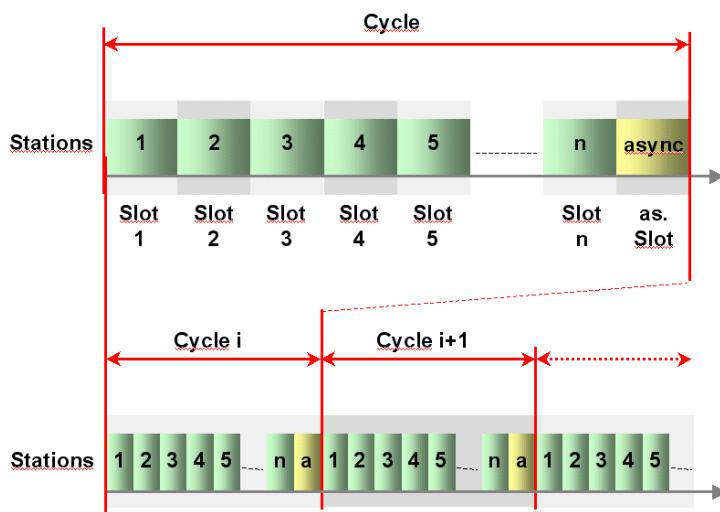


Fig. 76. POWERLINK communication slots

The object 1F9C<sub>h</sub> can be used to request fast processing nodes first and give slower nodes enough time for the SoC processing for example.

|           |                          |             |          |
|-----------|--------------------------|-------------|----------|
| Index     | 1F9C <sub>h</sub>        | Object Code | ARRAY    |
| Name      | NMT_IsochrSlotAssign_AU8 |             |          |
| Data Type | UNSIGNED8                | Category    | Optional |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 00 <sub>h</sub> |             |                     |
| Name          | NumberOfEntries |             |                     |
| Value Range   | 1 .. 254        | Access      | rws, valid on reset |
| Default Value | 254             | PDO Mapping | No                  |

Sub-index 0 may be used to limit the number of isochronous slots per cycle to be checked by the cycle producing in the DLL\_MS.

- **Sub-Index 01h – FEh: Nodeld**

|               |                 |             |                     |
|---------------|-----------------|-------------|---------------------|
| Sub-Index     | 01 <sub>h</sub> |             |                     |
| Name          | Nodeld          |             |                     |
| --            | --              | Category    | O                   |
| Value Range   | 0 .. 254        | Access      | rws, valid on reset |
| Default Value | 0               | PDO Mapping | No                  |

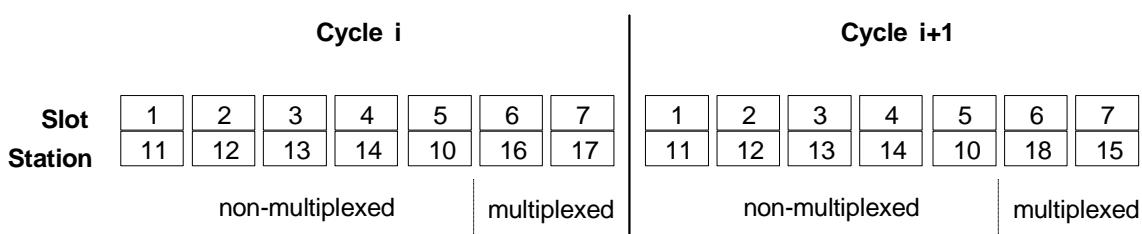
Each sub-index in the array corresponds to an individual communication slot which is equal to the sub-index. The slot shall only be used if there is an isochronous node with the Node ID assigned by index NMT\_NodeAssignment\_AU32[sub-index] Bits 0, 1, 8 and 9. Value 0 indicates that there is no particular node assigned to the communication slot.

The sub-indices can also be used for the slot assignment of multiplexed nodes. If multiplexed and non-multiplexed nodes shall be assigned to particular communication slots, it must be ensured that all the isochronous non-multiplexed stations are configured on the lower sub-indices of object 1F9Ch. Care has also to be taken that the multiplex slots are mapped to the communication slots in ascending order. That means the first multiplexed node assigned to object 1F9Ch must be configured in the first multiplex slot in object 1F9Bh and so on.

Gaps in the NMT\_IsochrSlotAssign\_AU8 are allowed as unused communication slots are skipped.

*Example:*

*Let's assume nodes 10 to 14 are non-multiplexed nodes and nodes 15-18 are multiplexed nodes. Furthermore node 10 is a slow processing node which must be shifted away from the cycle beginning. The slot diagram for the isochronous cycle will look as follows:*



The above example results in the following setup of object NMT\_IsochrSlotAssign\_AU8:

|           |                   |    |    |    |    |    |    |    |    |    |  |
|-----------|-------------------|----|----|----|----|----|----|----|----|----|--|
| Index     | 1F9C <sub>h</sub> |    |    |    |    |    |    |    |    |    |  |
| sub-index | 0                 | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |  |
| value     | 9                 | 11 | 12 | 13 | 14 | 10 | 16 | 17 | 18 | 15 |  |

The setup of object  $1F9B_h$  NMT\_MultipICycleAssign\_AU8 for the multiplexed nodes 15 to 18 must look as follows:

|           |          |    |    |    |    |    |    |    |    |
|-----------|----------|----|----|----|----|----|----|----|----|
| Index     | $1F9B_h$ |    |    |    |    |    |    |    |    |
| sub-index | 10       | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| value     | 0        | 0  | 0  | 0  | 0  | 2  | 1  | 1  | 2  |

## 7.2.2.4 CN NMT State Surveillance

The objects described by this paragraph are used by the MN surveillance of the CN NMT states as described at 7.1.4.

### 7.2.2.4.1 Object $1F8E_h$ : NMT\_MNNodeCurrState\_AU8

This object holds a list of the current NMT states of the configured nodes.

|           |                         |             |       |
|-----------|-------------------------|-------------|-------|
| Index     | $1F8E_h$                | Object Code | ARRAY |
| Name      | NMT_MNNodeCurrState_AU8 |             |       |
| Data Type | UNSIGNED8               | Category    | M     |

- **Sub-Index  $00_h$ : NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | $00_h$          |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 254             | Access      | const |
| Default Value | 254             | PDO Mapping | No    |

- **Sub-Index  $01_h$  –  $FE_h$ : CurrState**

|               |  |             |    |
|---------------|--|-------------|----|
| Sub-Index     | 01h .. Feh                                   |             |    |
| Name          | CurrState                                    |             |    |
| --            | --   | Category    | M  |
| Value Range   | see App. 3.6                                 | Access      | ro |
| Default Value | NMT_CS_NOT_ACTIVE resp.<br>NMT_MS_NOT_ACTIVE | PDO Mapping | No |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID indicates the current state of the MN state machine. It holds values described by Tab. 105.

The individual states of the nodes may be locally accessed via NMT\_CurrNMTState\_U8.

### 7.2.2.4.2 Object $1F8F_h$ : NMT\_MNNodeExpState\_AU8

This object holds a list of the expected NMT states of the configured nodes in accordance with the CNs' boot-up behavior and the NMT state commands transmitted by the MN. See 7.1.4.

The sub-indices of NMT\_MNNodeExpState\_AU8 should be equal to those of NMT\_MNNodeCurrState\_AU8 expect for an interval of C\_NMT\_STATE\_TOLERANCE after an NMT state command to the respective CN.

|           |                        |             |       |
|-----------|------------------------|-------------|-------|
| Index     | $1F8F_h$               | Object Code | ARRAY |
| Name      | NMT_MNNodeExpState_AU8 |             |       |
| Data Type | UNSIGNED8              | Category    | O     |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 254             | Access      | const |
| Default Value | 254             | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub> – FE<sub>h</sub>: ExpState**

|               |  |             |    |
|---------------|--|-------------|----|
| Sub-Index     | 01 <sub>h</sub> .. FE <sub>h</sub>           |             |    |
| Name          | ExpState                                     |             |    |
| --            | --   | Category    | M  |
| Value Range   | see App. 3.6                                 | Access      | ro |
| Default Value | NMT_CS_NOT_ACTIVE resp.<br>NMT_MS_NOT_ACTIVE | PDO Mapping | No |

Each sub-index in the array corresponds to the node with the Node ID equal to the sub-index. The sub-index value is valid only if there is a node assigned to the Node ID by index NMT\_NodeAssignment\_AU32[sub-index] Bit 0.

Sub-index C\_ADR\_MN\_DEF\_NODE\_ID indicates the current state of the MN state machine. It holds values described by Tab. 105.

## 7.2.2.5 NMT Service Interface

### 7.2.2.5.1 Object 1F9F<sub>h</sub>: NMT\_RequestCmd\_REC

NMT\_RequestCmd\_REC may be used by a diagnostic node outside of the POWERLINK segment connected via a router to initiate an NMT command by the MN.

|           |                     |             |        |
|-----------|---------------------|-------------|--------|
| Index     | 1F9F <sub>h</sub>   | Object Code | RECORD |
| Name      | NMT_RequestCmd_REC  |             |        |
| Data Type | NMT_RequestCmd_TYPE | Category    | M      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 3 .. 4          | Access      | const |
| Default Value | -               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: Release\_BOOL**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub> |             |    |
| Name          | Release_BOOL    |             |    |
| Data Type     | BOOLEAN         | Category    | M  |
| Value Range   | BOOLEAN         | Access      | rw |
| Default Value | FALSE           | PDO Mapping | No |

Writing TRUE to Release\_BOOL shall trigger the positioning of the NMT command described by sub-indices 02<sub>h</sub> to 04<sub>h</sub> to the transmission queues of the MN.

Release\_BOOL shall be automatically reset to FALSE by the MN, when queuing of the NMT services has been completed.

- **Sub-Index 02<sub>h</sub>: CmdID\_U8**

|               |                   |             |    |
|---------------|-------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>   |             |    |
| Name          | CmdID_U8          |             |    |
| Data Type     | UNSIGNED8         | Category    | M  |
| Value Range   | see.App. 3.7      | Access      | rw |
| Default Value | NMTInvalidService | PDO Mapping | No |

CmdID\_U8 indicates the requested NMT service. It is equivalent to the NMTRRequestedCommandID entry of the NMTRRequest Frame NMT Service Slot (see 7.3.6.1)

- **Sub-Index 03<sub>h</sub>: CmdTarget\_U8**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 03 <sub>h</sub> |             |    |
| Name          | CmdTarget_U8    |             |    |
| Data Type     | UNSIGNED8       | Category    | M  |
| Value Range   | UNSIGNED8       | Access      | rw |
| Default Value | C_ADR_INVALID   | PDO Mapping | No |

CmdTarget\_U8 indicates POWERLINK address of the target node of the requested NMT service. It is equivalent to the NMTRRequestedCommandTarget entry of the NMTRRequest Frame NMT Service Slot (see 7.3.6.1)

- **Sub-Index 04<sub>h</sub>: CmdData\_DOM**

|               |                 |             |      |
|---------------|-----------------|-------------|------|
| Sub-Index     | 04 <sub>h</sub> |             |      |
| Name          | CmdData_DOM     |             |      |
| Data Type     | DOMAIN          | Category    | Cond |
| Value Range   | DOMAIN          | Access      | rw   |
| Default Value | -               | PDO Mapping | No   |

CmdData\_DOM provides data specific to the requested NMT service. It is equivalent to the NMTRRequestedCommandData entry of the NMTRRequest Frame NMT Service Slot (see 7.3.6.1)

CmdData\_DOM shall be supported if extended NMT State Command Services are supported by the MN, support shall be indicated by the object dictionary index NMT\_FeatureFlags\_U32 Bit 5.

## 7.2.3 NMT CN Objects

### 7.2.3.1 CN StartUp Behaviour

Hint: MN and CN startup timing should be well balanced. System power up sequence should be considered.

#### 7.2.3.1.1 Object 1F99<sub>h</sub>: NMT\_CNBASICEthernetTimeout\_U32

Provide the time in  $\mu$ s to be applied before changing from NMT\_CS\_NOT\_ACTIVE to NMT\_CS\_BASIC\_ETHERNET.

|               |                                |             |                     |
|---------------|--------------------------------|-------------|---------------------|
| Index         | 1F99 <sub>h</sub>              | Object Type | VAR                 |
| Name          | NMT_CNBASICEthernetTimeout_U32 |             |                     |
| Data Type     | UNSIGNED32                     | Category    | M                   |
| Value Range   | UNSIGNED32                     | Access      | rws, valid on reset |
| Default Value | 5 000 000                      | PDO Mapping | No                  |

Value 0 shall mean, never change to NMT\_CS\_BASIC\_ETHERNET. If not 0, the value shall be greater than NMT\_CycleLen\_U32.

To avoid erroneous change over to NMT\_CS\_BASIC\_ETHERNET at system startup, NMT\_CNBASICEthernetTimeout\_U32 should be greater than the MN's NMT\_BootTime\_REC.MNWaitNotAct\_U32.

*Note: It is the responsibility of the user resp. configuration tool to set an appropriate value.*

## 7.2.4 NMT Object Types

### 7.2.4.1 Object 0023<sub>h</sub>: IDENTITY

|           |                   |                   |            |
|-----------|-------------------|-------------------|------------|
| Index     | 0023 <sub>h</sub> | Object Type       | DEFSTRUCT  |
| Name      | IDENTITY          |                   |            |
| Sub-Index | Component Name    | Value             | Data Type  |
| 00h       | NumberOfEntries   | 04 <sub>h</sub>   |            |
| 01h       | VendorId_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 02h       | ProductCode_U32   | 0007 <sub>h</sub> | UNSIGNED32 |
| 03h       | RevisionNo_U32    | 0007 <sub>h</sub> | UNSIGNED32 |
| 04h       | SerialNo_U32      | 0007 <sub>h</sub> | UNSIGNED32 |

### 7.2.4.2 Object 0429<sub>h</sub>: NMT\_ParameterStorage\_TYPE

|                                    |                           |                   |            |
|------------------------------------|---------------------------|-------------------|------------|
| Index                              | 0429 <sub>h</sub>         | Object Type       | DEFSTRUCT  |
| Name                               | NMT_ParameterStorage_TYPE |                   |            |
| Sub-Index                          | Component Name            | Value             | Data Type  |
| 00 <sub>h</sub>                    | NumberOfEntries           | 7F <sub>h</sub>   |            |
| 01 <sub>h</sub>                    | AllParam_U32              | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub>                    | CommunicationParam_U32    | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub>                    | ApplicationParam_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub> .. 7F <sub>h</sub> | ManufacturerParam_XXh_U32 | 0007 <sub>h</sub> | UNSIGNED32 |

### 7.2.4.3 Object 042B<sub>h</sub>: NMT\_InterfaceGroup\_Xh\_TYPE

|                 |                            |                   |                |
|-----------------|----------------------------|-------------------|----------------|
| Index           | 042B <sub>h</sub>          | Object Type       | DEFSTRUCT      |
| Name            | NMT_InterfaceGroup_Xh_TYPE |                   |                |
| Sub-Index       | Component Name             | Value             | Data Type      |
| 00 <sub>h</sub> | NumberOfEntries            | 09 <sub>h</sub>   |                |
| 01 <sub>h</sub> | InterfaceIndex_U16         | 0006 <sub>h</sub> | UNSIGNED16     |
| 02 <sub>h</sub> | InterfaceDescription_VSTR  | 0009 <sub>h</sub> | VISIBLE_STRING |
| 03 <sub>h</sub> | InterfaceType_U8           | 0005 <sub>h</sub> | UNSIGNED8      |
| 04 <sub>h</sub> | InterfaceMtu_U16           | 0006 <sub>h</sub> | UNSIGNED16     |
| 05 <sub>h</sub> | InterfacePhysAddress_OSTR  | 000A <sub>h</sub> | OCTET_STRING   |
| 06 <sub>h</sub> | InterfaceName_VSTR         | 0009 <sub>h</sub> | VISIBLE_STRING |
| 07 <sub>h</sub> | InterfaceOperStatus_U8     | 0005 <sub>h</sub> | UNSIGNED8      |
| 08 <sub>h</sub> | InterfaceAdminState_U8     | 0005 <sub>h</sub> | UNSIGNED8      |
| 09 <sub>h</sub> | Valid_BOOL                 | 0001 <sub>h</sub> | BOOLEAN        |

### 7.2.4.4 Object 042C<sub>h</sub>: NMT\_CycleTiming\_TYPE

|                 |                         |                   |            |
|-----------------|-------------------------|-------------------|------------|
| Index           | 042C <sub>h</sub>       | Object Type       | DEFSTRUCT  |
| Name            | NMT_CycleTiming_TYPE    |                   |            |
| Sub-Index       | Component Name          | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries         | 09 <sub>h</sub>   |            |
| 01 <sub>h</sub> | IsochrTxMaxPayload_U16  | 0006 <sub>h</sub> | UNSIGNED16 |
| 02 <sub>h</sub> | IsochrRxMaxPayload_U16  | 0006 <sub>h</sub> | UNSIGNED16 |
| 03 <sub>h</sub> | PResMaxLatency_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub> | PReqActPayloadLimit_U16 | 0006 <sub>h</sub> | UNSIGNED16 |
| 05h             | PResActPayloadLimit_U16 | 0006 <sub>h</sub> | UNSIGNED16 |

|                 |                    |                   |            |
|-----------------|--------------------|-------------------|------------|
| 06 <sub>h</sub> | ASndMaxLatency_U32 | 0007 <sub>h</sub> | UNSIGNED32 |
| 07 <sub>h</sub> | MultiplCycleCnt_U8 | 0005 <sub>h</sub> | UNSIGNED8  |
| 08 <sub>h</sub> | AsyncMTU_U16       | 0006 <sub>h</sub> | UNSIGNED16 |
| 09 <sub>h</sub> | Prescaler_U16      | 0006 <sub>h</sub> | UNSIGNED16 |

#### 7.2.4.5 Object 042E<sub>h</sub>: NMT\_BootTime\_TYPE

| Index           | Object Code                 | DEFSTRUCT         |
|-----------------|-----------------------------|-------------------|
| Name            | NMT_BootTime_TYPE           |                   |
| Sub-Index       | Component Name              | Value             |
| 00 <sub>h</sub> | NumberOfEntries             | 05 <sub>h</sub>   |
| 01 <sub>h</sub> | MNWaitNotAct_U32            | 0007 <sub>h</sub> |
| 02 <sub>h</sub> | MNTtimeoutPreOp1_U32        | 0007 <sub>h</sub> |
| 03 <sub>h</sub> | MNWaitPreOp1_U32            | 0007 <sub>h</sub> |
| 04 <sub>h</sub> | MNTtimeoutPreOp2_U32        | 0007 <sub>h</sub> |
| 05 <sub>h</sub> | MNTtimeoutReadyToOp_U32     | 0007 <sub>h</sub> |
| 06 <sub>h</sub> | MNIdentificationTimeout_U32 | 0007 <sub>h</sub> |
| 07 <sub>h</sub> | MNSoftwareTimeout_U32       | 0007 <sub>h</sub> |
| 08 <sub>h</sub> | MNConfigurationTimeout_U32  | 0007 <sub>h</sub> |
| 09 <sub>h</sub> | MNStartCNTTimeout_U32       | 0007 <sub>h</sub> |

#### 7.2.4.6 Object 042F<sub>h</sub>: NMT\_MNCycleTiming\_TYPE

| Index           | Object Code            | DEFSTRUCT         |
|-----------------|------------------------|-------------------|
| Name            | NMT_MNCycleTiming_TYPE |                   |
| Sub-Index       | Component Name         | Value             |
| 00 <sub>h</sub> | NumberOfEntries        | 02 <sub>h</sub>   |
| 01 <sub>h</sub> | WaitSoCPReq_U32        | 0007 <sub>h</sub> |
| 02 <sub>h</sub> | AsyncSlotTimeout_U32   | 0007 <sub>h</sub> |

#### 7.2.4.7 Object 0439<sub>h</sub>: NMT\_EPLNodeID\_TYPE

| Index           | Object Code        | DEFSTRUCT         |
|-----------------|--------------------|-------------------|
| Name            | NMT_EPLNodeID_TYPE |                   |
| Sub-Index       | Component Name     | Value             |
| 00 <sub>h</sub> | NumberOfEntries    | 02 <sub>h</sub>   |
| 01 <sub>h</sub> | NodeID_U8          | 0005 <sub>h</sub> |
| 02 <sub>h</sub> | NodeIDByHW_BOOL    | 0001 <sub>h</sub> |

#### 7.2.4.8 Object 043A<sub>h</sub>: NMT\_RequestCmd\_TYPE

| Index           | Object Code         | DEFSTRUCT         |
|-----------------|---------------------|-------------------|
| Name            | NMT_RequestCmd_TYPE |                   |
| Sub-Index       | Component Name      | Value             |
| 00 <sub>h</sub> | NumberOfEntries     | 04 <sub>h</sub>   |
| 01 <sub>h</sub> | Release_BOOL        | 0001 <sub>h</sub> |
| 02 <sub>h</sub> | CmdID_U8            | 0005 <sub>h</sub> |
| 03 <sub>h</sub> | CmdTarget_U8        | 0005 <sub>h</sub> |
| 04 <sub>h</sub> | CmdData_DOM         | 000F <sub>h</sub> |

## 7.3 Network Management Services

POWERLINK Network Management (NMT) is node-oriented and follows a master/slave relationship.

The function of the NMT master is carried out by the MN.

CNs are administrated as NMT slaves by the master. An NMT slave is uniquely identified in the network by its POWERLINK Node ID.

According to the definition, Network Management is directed from the NMT master (MN) to the NMT slaves (CNs).

POWERLINK defines five categories of NMT services:

- **NMT State Command Services**

The MN uses NMT State Command Services to control the CN state machine(s).

- **NMT Managing Command Services**

The MN uses NMT Managing Command Services to access NMT data items of the CN(s) in a fast coordinated way.

- **NMT Response Services**

NMT Response Services indicate the current NMT state of a CN to the MN.

- **NMT Info Services**

NMT Info Services are used to transmit NMT information from the MN to a CN.

- **NMT Guard services**

NMT Guard Services are used by the MN and CNs to detect failures in a POWERLINK network.

A CN may request NMT command and info services to be issued by the MN (NMTRRequest, see 7.3.6).

In NMT\_CS\_EPL\_MODE, a CN shall ignore NMT command services that are not issued by the MN. In NMT\_CS\_BASIC\_ETHERNET NMT command services shall be accepted regardless their source node.

### 7.3.1 NMT State Command Services

The MN controls the state of the CN via NMT State Command Services. The state transitions are defined by the CN state machine (see 7.1.4).

POWERLINK distinguishes between implicit and explicit NMT State Commands.

#### 7.3.1.1 Implicit NMT State Command Services

At system startup, the reception or the timeout of SoA, PReq, PRes or SoC frames trigger CN state machine transitions from the state NMT\_CS\_NOT\_ACTIVE to the next states. In NMT\_CS\_PRE\_OPERATIONAL\_1 the reception of SoC triggers the transition to NMT\_CS\_PRE\_OPERATIONAL\_2. SoA, PReq, PRes or SoC are used to synchronise a CN with the current network mode after system start or reset. (see 7.1.4).

In Basic Ethernet Mode, the reception of POWERLINK SoA, PReq, PRes or SoC will trigger a change over to NMT\_CS\_EPL\_MODE. (see 7.1.4).

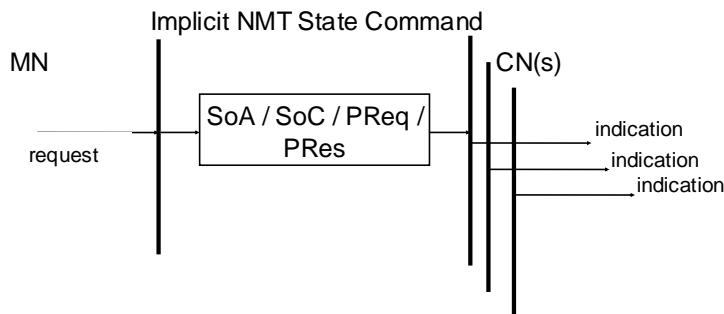


Fig. 77. Implicit NMT state command service protocol

SoA, PReq, PRes or SoC acting as shown above are termed implicit NMT state commands. They are valid regardless of their data content and without further extensions.

Tab. 122 displays implicit NMT state commands in relation to the current CN state when the command is received. SoA and SoC reception or timeouts not mentioned in the table do not trigger state transitions.

| Current State            | Implicit NMT State Command     | Destination State        |
|--------------------------|--------------------------------|--------------------------|
| NMT_CS_NOT_ACTIVE        | SoA, SoC                       | NMT_CS_PRE_OPERATIONAL_1 |
|                          | Timeout (SoA, PReq, PRes, SoC) | NMT_CS_BASIC_ETHERNET    |
| NMT_CS_PRE_OPERATIONAL_1 | Soc                            | NMT_CS_PRE_OPERATIONAL_2 |
| NMT_CS_BASIC_ETHERNET    | SoA, PReq, PRes, SoC           | NMT_CS_PRE_OPERATIONAL_1 |

Tab. 122 Implicit NMT state commands

An NMT Response Service (see 7.3.3) requested by the implicit NMT State Command shall indicate the current state.

### 7.3.1.1 Implicit NMT State Command Transmission

Implicit NMT State Command services (7.3.1.1) do not require explicit NMT frame transmission by the MN. Regular SoA, PReq, PRes and SoC frames (refer 4.6.1.1) can be valid Implicit NMT State Commands on their own.

SoA, PRes and SoC frames cannot be sent directly to a single node – they are multicast. The CN decides according to its own current state whether a received implicit NMT State command is active (i.e. is to be carried out).

### 7.3.1.2 Explicit NMT State Command Services

An explicit NMT State Command shall be transmitted by the MN in the ASnd frame.

The target CN shall be addressed via the ASnd Destination field. The ASnd frame shall be either unicast to one CN or broadcast to all CNs (using the POWERLINK broadcast address C\_ADR\_BROADCAST). A command sent via broadcast shall be ignored at the transmitting MN.

Special broadcast NMT commands may be sent to selected groups of CNs (see 7.3.1.2.2).

The explicit NMT State Command services shall be identified by ASnd ServiceID = NMT\_COMMAND.

Fig. 78 shows the protocol used to implement the explicit NMT State Command Services.

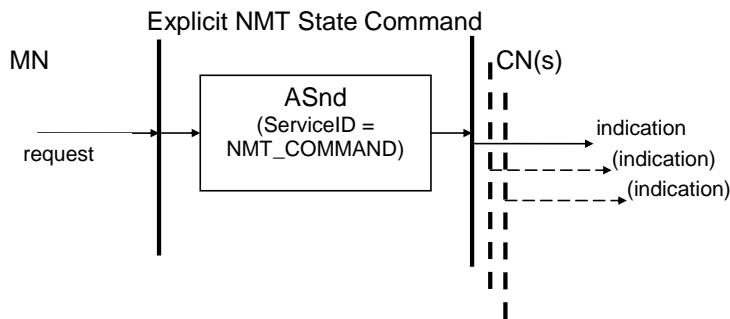


Fig. 78. Explicit NMT state command service protocol

The ASnd ServiceID value of explicit NMT State Command Services shall be NMT\_COMMAND. The specific NMT State Command identification and its data shall be located in the ASnd Payload field, called NMT Service Slot in the following.

| Octet offset <sup>24</sup> | Bit Offset     |   |   |   |   |   |   |   |
|----------------------------|----------------|---|---|---|---|---|---|---|
|                            | 7              | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0                          | NMTCommandID   |   |   |   |   |   |   |   |
| 1                          | Reserved       |   |   |   |   |   |   |   |
| 2 .. n                     | NMTCommandData |   |   |   |   |   |   |   |

$n \leq C\_DLL\_MAX\_PAYL\_OFFSET - 4$

Tab. 123 NMT state command service, NMT managing command service and NMT info service structure of the NMT Service Slot field

| Field          | Abbr. | Description   | Value              |
|----------------|-------|---|--------------------|
| NMTCommandID   | cid   | Qualifies the NMT state command.  | Tab. 125, Tab. 126 |
| NMTCommandData | cdat  | 0..C_DLL_MAX_PAYL_OFFSET – 6 octets of NMT command-specific data to be issued by the MN.<br>The lower layer is responsible for padding. |                    |

Tab. 124 NMT Service Slot fields of explicit NMT state command services

### 7.3.1.2.1 Plain NMT State Command

The Plain NMT State Command shall be unicast to a specific CN or broadcast to all CNs.

Plain NMT State Commands shall ignore NMTCommandData.

The following Plain NMT State Commands are defined:

<sup>24</sup> Octet offset is measured from the begin of the ASnd Payload field. The offset relative to the beginning of the Ethernet frame is 18 octets.

| NMTCommandID            | Initial state  | Destination state  |
|-------------------------|--|--|
| NMTStartNode            | NMT_CS_READY_TO_OPERATE                                | NMT_CS_OPERATIONAL   |
| NMTStopNode             | NMT_CS_PRE_OPERATIONAL_2                               | NMT_CS_STOPPED   |
|                         | NMT_CS_READY_TO_OPERATE                                |  |
|                         | NMT_CS_OPERATIONAL                                     |  |
| NMTEnterPreOperational2 | NMT_CS_OPERATIONAL                                     | NMT_CS_PRE_OPERATIONAL_2   |
|                         | NMT_CS_STOPPED   |  |
| NMTEnableReadyToOperate | NMT_CS_PRE_OPERATIONAL_2                               | NMT_CS_PRE_OPERATIONAL_2   |
| NMTRestNode             | NMT_CS_NOT_ACTIVE /<br>NMT_MS_NOT_ACTIVE               | NMT_GS_INITIALISATION<br>sub-state<br>NMT_GS_RESET_APPLICATION   |
|                         | NMT_CS_PRE_OPERATIONAL_1 /<br>NMT_MS_PRE_OPERATIONAL_1 |  |
|                         | NMT_CS_PRE_OPERATIONAL_2 /<br>NMT_MS_PRE_OPERATIONAL_2 |  |
|                         | NMT_CS_READY_TO_OPERATE /<br>NMT_MS_READY_TO_OPERATE   |  |
|                         | NMT_CS_OPERATIONAL /<br>NMT_MS_OPERATIONAL             |  |
|                         | NMT_CS_STOPPED   |  |
|                         | NMT_CS_BASIC_ETHERNET                                  |  |
| NMTRestCommunication    | NMT_CS_NOT_ACTIVE /<br>NMT_MS_NOT_ACTIVE               | NMT_GS_INITIALISATION<br>sub-state<br>NMT_GS_RESET_COMMUNICATION |
|                         | NMT_CS_PRE_OPERATIONAL_1 /<br>NMT_MS_PRE_OPERATIONAL_1 |  |
|                         | NMT_CS_PRE_OPERATIONAL_2 /<br>NMT_MS_PRE_OPERATIONAL_2 |  |
|                         | NMT_CS_READY_TO_OPERATE /<br>NMT_MS_READY_TO_OPERATE   |  |
|                         | NMT_CS_OPERATIONAL /<br>NMT_MS_OPERATIONAL             |  |
|                         | NMT_CS_STOPPED   |  |
|                         | NMT_CS_BASIC_ETHERNET                                  |  |
| NMTRestConfiguration    | NMT_CS_NOT_ACTIVE /<br>NMT_MS_NOT_ACTIVE               | NMT_GS_INITIALISATION<br>sub-state<br>NMT_GS_RESET_CONFIGURATION |
|                         | NMT_CS_PRE_OPERATIONAL_1 /<br>NMT_MS_PRE_OPERATIONAL_1 |  |
|                         | NMT_CS_PRE_OPERATIONAL_2 /<br>NMT_MS_PRE_OPERATIONAL_2 |  |
|                         | NMT_CS_READY_TO_OPERATE /<br>NMT_MS_READY_TO_OPERATE   |  |
|                         | NMT_CS_OPERATIONAL /<br>NMT_MS_OPERATIONAL             |  |
|                         | NMT_CS_STOPPED   |  |
|                         | NMT_CS_BASIC_ETHERNET                                  |  |
| NMTSwReset              | NMT_CS_NOT_ACTIVE /<br>NMT_MS_NOT_ACTIVE               | NMT_GS_INITIALISATION<br>sub-state NMT_GS_INITIALISING           |
|                         | NMT_CS_PRE_OPERATIONAL_1 /<br>NMT_MS_PRE_OPERATIONAL_1 |  |
|                         | NMT_CS_PRE_OPERATIONAL_2 /<br>NMT_MS_PRE_OPERATIONAL_2 |  |
|                         | NMT_CS_READY_TO_OPERATE /<br>NMT_MS_READY_TO_OPERATE   |  |
|                         | NMT_CS_OPERATIONAL /<br>NMT_MS_OPERATIONAL             |  |
|                         | NMT_CS_STOPPED   |  |
|                         | NMT_CS_BASIC_ETHERNET                                  |  |

Tab. 125 Plain NMT state commands

All commands listed in Tab. 125 are mandatory. The commands shall be only acted on if the CN is in the appropriate initial state (see Tab. 125); otherwise the command is ignored and an entry made in the CN error log.

### 7.3.1.2.1.1 NMT Reset Commands to the MN

NMTSwReset, NMTRestNode, NMTRestConfiguration and NMTRestCommunication addressed to the MN as unicast may be requested by a diagnostic node (refer 7.3.6). Since MN resets will affect the complete network they shall be forwarded to all nodes. The requested command shall be performed by the MN in the following way:

1. route the command to all CNs.
2. perform the requested reset.

### 7.3.1.2.2 Extended NMT State Command

Extended NMT State Commands are used to access groups of CNs.

The ASnd frame transporting the command is broadcast to all CNs.

The “NMTCommand Specific Data” field contains a POWERLINK Node List according to the POWERLINK Node List Format (see 7.3.1.2.3). The POWERLINK Node List indicates the validity of the command for the individual nodes (i.e. whether the node acts on the command). Node IDs to which the command is addressed are indicated by 1<sub>b</sub>. Nodes that have to ignore the command are indicated by 0<sub>b</sub>.

Tab. 126 lists the Extended NMT State Commands. Initial State, Destination State and the validity of the commands at the initial states are identical to the respective plain commands (see Tab. 125).

| NMTCommandID               | M/O |
|----------------------------|-----|
| NMTStartNodeEx             | O   |
| NMTStopNodeEx              | O   |
| NMTEEnterPreOperational2Ex | O   |
| NMTEEnableReadyToOperateEx | O   |
| NMTRestNodeEx              | O   |
| NMTRestCommunicationEx     | O   |
| NMTRestConfigurationEx     | O   |
| NMTSwResetEx               | O   |

Tab. 126 Extended NMT state commands

Support of Extended NMT State Commands is optional. Support shall be indicated by the object dictionary index NMT\_FeatureFlags\_U32 Bit 5 and D\_NMT\_ExtNmtCmds\_BOOL. The Flags don't differentiate particular Extended NMT State Commands. They indicate support of all Extended NMT State Commands defined by this specification.

### 7.3.1.2.3 POWERLINK Node List Format

The POWERLINK Node List is transmitted by the NMTCommandData field.

The POWERLINK Node List format assigns one bit for each POWERLINK Node ID. The Node ID assignment is given in Tab. 127.

| Octet offset <sup>25</sup> | Bit offset |     |     |     |     |     |     |     |  |
|----------------------------|------------|-----|-----|-----|-----|-----|-----|-----|--|
|                            | 7          | 6   | 5   | 4   | 3   | 2   | 1   | 0   |  |
| 0                          | 7          | 6   | 5   | 4   | 3   | 2   | 1   | -   |  |
| 1                          | 15         | 14  | 13  | 12  | 11  | 10  | 9   | 8   |  |
| 2                          | 23         | 22  | 21  | 20  | 19  | 18  | 17  | 16  |  |
| 3                          | 31         | 30  | 29  | 28  | 27  | 26  | 25  | 24  |  |
| 4                          | 39         | 38  | 37  | 36  | 35  | 34  | 33  | 32  |  |
| 5                          | 47         | 46  | 45  | 44  | 43  | 42  | 41  | 40  |  |
| 6                          | 55         | 54  | 53  | 52  | 51  | 50  | 49  | 48  |  |
| 7                          | 63         | 62  | 61  | 60  | 59  | 58  | 57  | 56  |  |
| 8                          | 71         | 70  | 69  | 68  | 67  | 66  | 65  | 64  |  |
| 9                          | 79         | 78  | 77  | 76  | 75  | 74  | 73  | 72  |  |
| 10                         | 87         | 86  | 85  | 84  | 83  | 82  | 81  | 80  |  |
| 11                         | 95         | 94  | 93  | 92  | 91  | 90  | 89  | 88  |  |
| 12                         | 103        | 102 | 101 | 100 | 99  | 98  | 97  | 96  |  |
| 13                         | 111        | 110 | 109 | 108 | 107 | 106 | 105 | 104 |  |
| 14                         | 119        | 118 | 117 | 116 | 115 | 114 | 113 | 112 |  |
| 15                         | 127        | 126 | 125 | 124 | 123 | 122 | 121 | 120 |  |
| 16                         | 135        | 135 | 133 | 132 | 131 | 130 | 129 | 128 |  |
| 17                         | 143        | 142 | 141 | 140 | 139 | 138 | 137 | 136 |  |
| 18                         | 151        | 150 | 149 | 148 | 147 | 146 | 145 | 144 |  |
| 19                         | 159        | 158 | 157 | 156 | 155 | 154 | 153 | 152 |  |
| 20                         | 167        | 166 | 165 | 164 | 163 | 162 | 161 | 160 |  |
| 21                         | 175        | 174 | 173 | 172 | 171 | 170 | 169 | 168 |  |
| 22                         | 183        | 182 | 181 | 180 | 179 | 178 | 177 | 176 |  |
| 23                         | 191        | 190 | 189 | 188 | 187 | 186 | 185 | 184 |  |
| 24                         | 199        | 198 | 197 | 196 | 195 | 194 | 193 | 192 |  |
| 25                         | 207        | 206 | 205 | 204 | 203 | 202 | 201 | 200 |  |
| 26                         | 215        | 214 | 213 | 212 | 211 | 210 | 209 | 208 |  |
| 27                         | 223        | 222 | 221 | 220 | 219 | 218 | 217 | 216 |  |
| 28                         | 231        | 230 | 229 | 228 | 227 | 226 | 225 | 224 |  |
| 29                         | 239        | 238 | 237 | 236 | 235 | 234 | 233 | 232 |  |
| 30                         | 247        | 246 | 245 | 244 | 243 | 242 | 241 | 240 |  |
| 31                         | -          | 254 | 253 | 252 | 251 | 250 | 249 | 248 |  |

Tab. 127 POWERLINK node list: Node ID to bit assignment

### 7.3.2 NMT Managing Command Services

The MN uses NMT Managing Command Services to configure NMT-relevant entries in the database of the CN. These commands do not directly influence the state machine of the CN.

NMT Managing Commands are transmitted in an ASnd frame by the MN. They are unicast to a single CN or broadcast to all CNs (using the POWERLINK broadcast address C\_ADR\_BROADCAST).

The services are identified by ASnd ServiceID set to NMT\_COMMAND.

Fig. 79 shows the protocol used to implement the NMT Managing Command Services.

<sup>25</sup> Octet offset is measured from the beginning of NMTCommandData. The offset relative to the Ethernet frame is 20 octets.

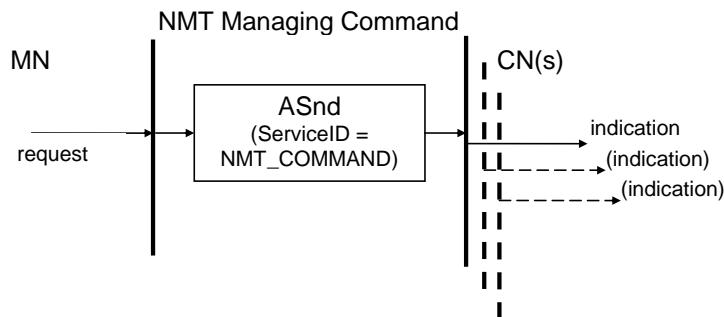


Fig. 79. NMT managing command service protocol

The ASnd ServiceID value of NMT Managing Command Services is NMT\_COMMAND. The specific NMT Managing Command identification and its data are located in the ASnd Payload field.

| Field          | Abbr. | Description   | Value    |
|----------------|-------|---|----------|
| NMTCommandID   | cid   | qualifies the NMT Managing Command.   | Tab. 129 |
| NMTCommandData | cdat  | 0..C_DLL_MAX_PAYL_OFFSET – 6 octets of NMT Managing command-specific data to be issued by the MN. The lower layers are responsible for padding. |          |

Tab. 128 NMT Service Slot fields of NMT managing command services

The following NMT Managing Command Services are defined:

| NMTCommandID      | M/O | Short description                                |
|-------------------|-----|--|
| NMTNetHostNameSet | O   | Sets hostname of an individual CN.               |
| NMTFlushArpEntry  | O   | Clears local MAC and IP address list at all CNs. |

Tab. 129 NMT managing command services

Support of particular NMT Managing Command Services shall be indicated by specific device description entries (see 7.3.2.1.1, 7.3.2.1.2)

### 7.3.2.1 Service Descriptions

#### 7.3.2.1.1 NMTNetHostNameSet

NMTNetHostNameSet sets the hostname of the CN.

NMTCommandID is NMTNetHostNameSet.

| Octet offset <sup>26</sup> | Bit Offset |   |   |   |   |   |   |   |
|----------------------------|------------|---|---|---|---|---|---|---|
|                            | 7          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 31                    | HostName   |   |   |   |   |   |   |   |

Tab. 130 NMTCommandData structure of NMTNetHostNameSet

| Field    | Abbr. | Description                                   | Value |
|----------|-------|---|-------|
| HostName | hn    | May be used to modify CN's local DNS hostname |       |

Tab. 131 NMTCommandData data fields of NMTNetHostNameSet

The NMTNetHostNameSet command is addressed unicast to an individual CN.

After execution of NMTNetSetHostName command, the modified hostname is published by the CN. Publishing is carried out in the following manner:

1. The CN indicates an ASnd Transmission Request using the PRes bits of the PRes frame or a StatusResponse ASnd frame using the priority level C\_DLL\_ASND\_PRIO\_NMTRQST.
2. The MN assigns the asynchronous phase to the CN via an SoA frame. (RequestedServiceID = NMT\_REQUEST\_INVITE).

<sup>26</sup> Octet offset is measured from the beginning of NMTCommandData. The offset relative to the Ethernet frame is 20 octets.

3. The CN requests an IdentRequest to itself using the NMTRequest ASnd frame (see 7.3.6).
4. The MN transmits an SoA (RequestedServiceID = IDENT\_REQUEST) to the requesting CN.
5. The CN publishes its modified HostName via an IdentResponse ASnd frame.

Support shall be indicated by D\_NMT\_NetHostNameSet\_BOOL.

### 7.3.2.1.2 NMTFlushArpEntry

NMTFlushArpEntry removes the entries for a CN from the address tables of all other CNs. The entry to be eliminated is indicated by the Node ID of the CN to be removed.

NMTCommandID is NMTFlushArpEntry.

| Octet offset <sup>27</sup> | Bit Offset |   |   |   |   |   |   |   |
|----------------------------|------------|---|---|---|---|---|---|---|
|                            | 7          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0                          | Node ID    |   |   |   |   |   |   |   |

Tab. 132 NMTFlushArpEntry ASnd service slot structure

| Field   | Abbr. | Description  | Value |
|---------|-------|--|-------|
| Node ID | nid   | Identifies the node whose ARP entry is to be deleted. Using C_ADR_BROADCAST flushes all ARP entries. |       |

Tab. 133 NMTCommandData data fields of NMTFlushArpEntry

The NMTFlushArpEntry command is broadcast to all CNs.

Support shall be indicated by D\_NMT\_FlushArpEntry\_BOOL.

## 7.3.3 NMT Response Services

NMT Response Services are used by the MN to query NMT information from the CN, e.g. current state, error and setup data.

### 7.3.3.1 NMT State Response

The CNs shall signal their state to the MN via NMT State Response services.

CNs that communicate isochronously via PReq / PRes shall use the PRes frame to indicate their current state.

Fig. 80 shows the protocol used to implement the NMT State Response Service from isochronously communicating CNs. The service is mandatory on the MN and every CN device, that is able to communicate isochronously.

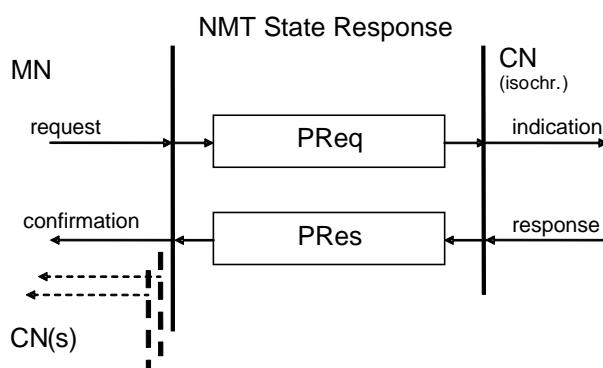


Fig. 80. NMT state response service protocol (isochronous CN)

The MN shall receive the NMT State Response. CNs may receive NMT State Response if configured to do so.

CNs which do not communicate isochronously (i.e. Async-only CNs or isochronous CNs in state NMT\_CS\_PRE\_OPERATIONAL\_1) shall signal their state via StatusResponse and IdentResponse

<sup>27</sup> Octet offset is measured from the beginning of NMTCommandData. The offset relative to the Ethernet frame is 20 octets.

ASnd frames. Isochronous CNs in NMT\_CS\_PRE\_OPERATIONAL\_2, NMT\_CS\_READY\_TO\_OPERATE or NMT\_CS\_OPERATIONAL may be triggered by the MN to respond via StatusResponse and IdentResponse ASnd frames.

Fig. 81 shows the protocol used to implement the NMT Response Service from CNs that are not communicating isochronously. The service is mandatory on the MN and every CN device.

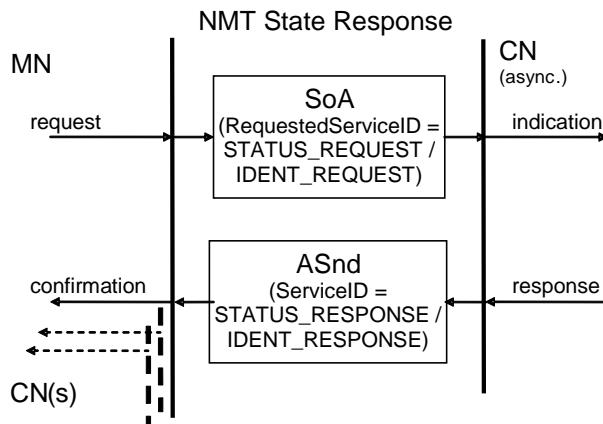


Fig. 81. NMT state response service protocol (async-only CN)

The MN shall receive the NMT State Response. CNs may receive the NMT State Response if configured to do so and if the NMT Status Response is transmitted via an IdentResponse ASnd frame.

NMT State Response shall use the following data fields of the PRes, StatusResponse or IdentResponse frames:

- NMTStatus  
NMTStatus shall report the current NMT status of the CN.  
If the NMT state response requesting message triggers a CN NMT state transition, the pre-transition state shall be reported. No response shall be issued if the CN is in state NMT\_CS\_NOT\_ACTIVE when receiving the request.
- Exception New (EN)  
EN shall report unread entries at the CN emergency queue (see 6.5.2).
- Exception Clear (EC)  
EC shall be used by Error Signaling management (see 6.5.2). EC shall be operated by StatusResponse and IdentResponse only.

Refer to 4.6.1.1.4, 7.3.3.2.1 and 7.3.3.3.1 for detailed information about frame format.

### 7.3.3.2 IdentResponse Service

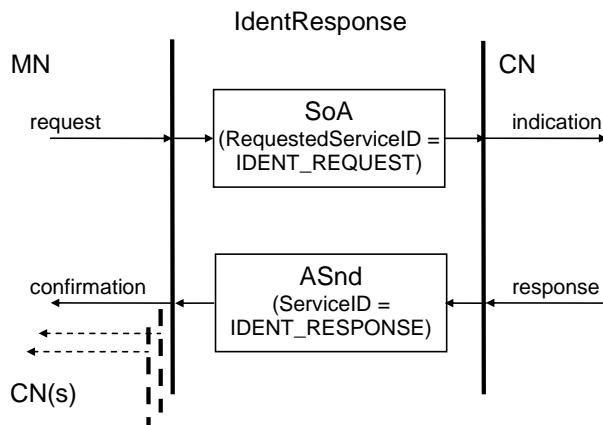


Fig. 82. IdentResponse service protocol

The IdentResponse service is used by the MN to identify configured but unrecognized CNs at system startup or after loss of communication. The service may be used after start-up to query a CN's setup information.

Fig. 82 shows the protocol that is used to implement the IdentResponse service. The service is mandatory on the MN and every CN device.

ASnd frames transporting IdentResponses are broadcast. The MN shall receive IdentResponses; CNs may receive IdentResponses if configured to do so.

The IdentResponse service may be initiated by a CN via the NMT Request mechanism (see 7.3.6). The NMTRequestedCommandID field of the NMT requesting ASnd frame shall be set to IDENT\_REQUEST.

### 7.3.3.2.1 IdentResponse Frame

The IdentResponse service frames are identified by ASnd ServiceID set to IDENT\_RESPONSE.

| Octet Offset <sup>28</sup> | Bit Offset               |                   |     |     |     |     |     |     |
|----------------------------|--------------------------|-------------------|-----|-----|-----|-----|-----|-----|
|                            | 7                        |                   | 5   | 4   | 3   | 2   | 1   | 0   |
| 0                          | res                      | res               | res | res | res | res | res | res |
| 1                          | res <sup>29</sup>        | res <sup>30</sup> |     | PR  |     |     | RS  |     |
| 2                          | NMTStatus                |                   |     |     |     |     |     |     |
| 3                          | Reserved                 |                   |     |     |     |     |     |     |
| 4                          | EPLVersion               |                   |     |     |     |     |     |     |
| 5                          | Reserved                 |                   |     |     |     |     |     |     |
| 6 .. 9                     | FeatureFlags             |                   |     |     |     |     |     |     |
| 10 .. 11                   | MTU                      |                   |     |     |     |     |     |     |
| 12 .. 13                   | PollInSize               |                   |     |     |     |     |     |     |
| 14 .. 15                   | PollOutSize              |                   |     |     |     |     |     |     |
| 16 .. 19                   | ResponseTime             |                   |     |     |     |     |     |     |
| 20 .. 21                   | Reserved                 |                   |     |     |     |     |     |     |
| 22 .. 25                   | DeviceType               |                   |     |     |     |     |     |     |
| 26 .. 29                   | VendorID                 |                   |     |     |     |     |     |     |
| 30 .. 33                   | ProductCode              |                   |     |     |     |     |     |     |
| 34 .. 37                   | RevisionNumber           |                   |     |     |     |     |     |     |
| 38 .. 41                   | SerialNumber             |                   |     |     |     |     |     |     |
| 42 .. 49                   | VendorSpecificExtension1 |                   |     |     |     |     |     |     |
| 50 .. 53                   | VerifyConfigurationDate  |                   |     |     |     |     |     |     |
| 54 .. 57                   | VerifyConfigurationTime  |                   |     |     |     |     |     |     |
| 58 .. 61                   | ApplicationSwDate        |                   |     |     |     |     |     |     |
| 62 .. 65                   | ApplicationSwTime        |                   |     |     |     |     |     |     |
| 66 .. 69                   | IPAddress                |                   |     |     |     |     |     |     |
| 70 .. 73                   | SubnetMask               |                   |     |     |     |     |     |     |
| 74 .. 77                   | DefaultGateway           |                   |     |     |     |     |     |     |
| 78 .. 109                  | HostName                 |                   |     |     |     |     |     |     |
| 110 .. 157                 | VendorSpecificExtension2 |                   |     |     |     |     |     |     |

Tab. 134 NMT Service Slot structure of IdentResponse

<sup>28</sup> Octet Offset is measured from the begin of the ASnd Payload field. The offset relative to the beginning of the Ethernet frame is 18 octets.

<sup>29</sup> Used by EPSG DS302-A [1]

<sup>30</sup> Used by EPSG DS302-A [1]

| Field                     | Abbr | Description   | Value                                     |
|---------------------------|------|---|---|
| Priority                  | PR   | Flags: Indicates the priority of the requested asynchronous frame.<br>(see 4.2.4.1.2.3)   |   |
| RequestToSend             | RS   | Flags: Indicates the number of pending requests to send at the CN. The value C_DLL_MAX_RS indicates C_DLL_MAX_RS or more requests, 0 indicates no pending request.        | 0 – C_DLL_MAX_RS                          |
| NMTStatus                 | stat | Reports the current status of the CN's NMT state machine.   |   |
| EPLVersion                | eplv | Indicates the POWERLINK Version to which the CN conforms.<br>coding of entry (see Tab. 112)   |   |
| FeatureFlags              | feat | Reports the device's feature flags<br>(NMT_FeatureFlags_U32)  |   |
| MTU                       | mtu  | Reports the size of the largest IP frame that can be transmitted over the network, including the size of the transport header.  | C_DLL_MIN_ASYNC_MTU – C_DLL_MAX_ASYNC_MTU |
| PollInSize                | pis  | Reports the actual CN setting for PReq data block size<br>(NMT_CycleTiming_REC.PReqActPayloadLimit_U16).  |   |
| PollOutSize               | pos  | Reports the actual CN setting for PRes data block size<br>(NMT_CycleTiming_REC.PResActPayloadLimit_U16).  |   |
| ResponseTime              | rst  | Reports the time required by the CN to respond to PReq<br>(NMT_CycleTiming_REC.PResMaxLatency_U32).   |   |
| DeviceType                | dt   | Reports the CN's Device type<br>(NMT_DeviceType_U32).   |   |
| VendorID                  | vid  | Reports the CN's Vendor ID, index<br>(NMT_IdentityObject_REC.VendorId_U32)  |   |
| ProductCode               | prdc | Reports the CN's Product Code, index<br>(NMT_IdentityObject_REC.ProductCode_U32)  |   |
| RevisionNumber            | rno  | Reports the CN's Revision Number,<br>(NMT_IdentityObject_REC.RevisionNo_U32).   |   |
| SerialNumber              | sno  | Reports the CN's Serial Number,<br>(NMT_IdentityObject_REC.SerialNo_U32).   |   |
| VendorSpecific-Extension1 | vex1 | May be used for vendor specific purpose, to be filled with zeros if not in use.   |   |
| Verify-ConfigurationDate  | vcd  | Reports the CN's Configuration date<br>(CFM_VerifyConfiguration_REC.ConfDate_U32).  |   |
| Verify-ConfigurationTime  | vct  | Reports the CN's Configuration time<br>(CFM_VerifyConfiguration_REC.ConfTime_U32).  |   |
| ApplicationSW-Date        | ad   | Reports the CN's Application SW date<br>(PDL_LocVerApplSw_REC.ApplSwDate_U32 on programmable device or date portion of NMT_ManufactSwVers_VS on non-programmable device). |   |
| ApplicationSW-Time        | at   | Reports the CN's Application SW time<br>(PDL_LocVerApplSw_REC.ApplSwTime_U32 on programmable device or time portion of NMT_ManufactSwVers_VS on non-programmable device). |   |
| IPAddress                 | ipa  | Reports the current IP address value of the CN<br>(NWL_IpAddrTable_Xh_REC.Addr_IPAD).   |   |
| SubnetMask                | snm  | Reports the current IP subnet mask value of the CN<br>(NWL_IpAddrTable_Xh_REC.NetMask_IPAD).  |   |
| DefaultGateway            | gtw  | Reports the current IP default gateway value of the CN (NWL_IpAddrTable_Xh_REC.DefGateway_IPAD).  |   |
| HostName                  | hn   | Reports the current DNS hostname of the CN<br>(NMT_HostName_VSTR).<br>Unused bytes of the hostname shall be set to 0h in the IdentResponse.                               |   |

| Field                     | Abbr | Description   | Value |
|---------------------------|------|---|-------|
| VendorSpecific-Extension2 | vex2 | May be used for vendor specific purpose, to be filled with zeros if not in use. |       |

Tab. 135 NMT Service Slot data fields of IdentResponse

If the respective object dictionary entry is not implemented by the device, the data field shall be set to 0 resp. empty string.

### 7.3.3.3 StatusResponse Service

The StatusResponse service is used by the MN to query the current status of CNs that are not communicating isochronously.

Fig. 83 shows the protocol used to implement the StatusResponse Service. The service is mandatory on the MN and every CN device.

ASnd frames transporting StatusResponses are broadcast. The MN shall receive StatusResponses; CNs may receive StatusResponses if configured to do so.

The StatusResponse service can be initiated by a CN via the NMT Request mechanism (7.3.6). The NMTRequestedCommandID entry of the NMT requesting ASnd frame is set to STATUS\_REQUEST.

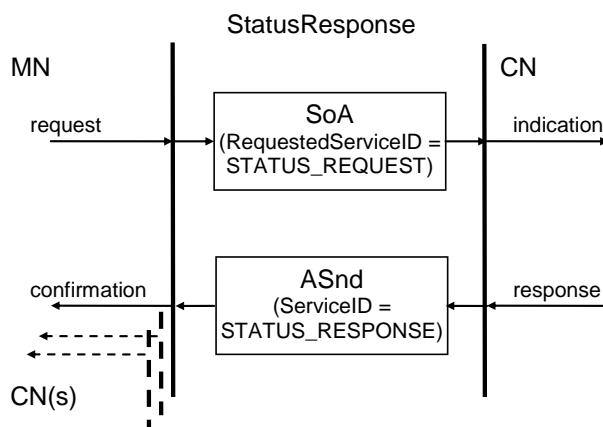


Fig. 83. StatusResponse service protocol

#### 7.3.3.3.1 StatusResponse Frame

The StatusResponse service frames are identified by ASnd ServiceID set to STATUS\_RESPONSE.

| Octet offset <sup>31</sup> | Bit Offset   |                   |     |    |    |     |     |     |  |  |  |  |  |
|----------------------------|--|-------------------|-----|----|----|-----|-----|-----|--|--|--|--|--|
|                            | 7  | 6                 | 5   | 4  | 3  | 2   | 1   | 0   |  |  |  |  |  |
| 0                          | res  | res               | res | EN | EC | res | res | res |  |  |  |  |  |
| 1                          | res <sup>32</sup>  | res <sup>33</sup> |     | PR |    | RS  |     |     |  |  |  |  |  |
| 2                          | NMT Status   |                   |     |    |    |     |     |     |  |  |  |  |  |
| 3 .. 5                     | Reserved   |                   |     |    |    |     |     |     |  |  |  |  |  |
| 6 .. 13                    | StaticErrorBitField  |                   |     |    |    |     |     |     |  |  |  |  |  |
| 14 .. 14+n*20              | List of Errors / Events (see 6.5.8)<br>20 Byte / Entry , minimum n=2 |                   |     |    |    |     |     |     |  |  |  |  |  |

*n (2-14) : Number of error/event entries*

Tab. 136 NMT Service Slot structure of StatusResponse

<sup>31</sup> Octet Offset is measured from the begin of the ASnd Payload field. The offset relative to the beginning of the Ethernet frame is 18 octets.

<sup>32</sup> Used by EPSG DS302-A [1]

<sup>33</sup> Used by EPSG DS302-A [1]

| Field               | Abbr | Description  | Value            |
|---------------------|------|--|------------------|
| ExceptionNew        | EN   | Flag: Error signaling (see 6.5.2).   |                  |
| ExceptionClear      | EC   | Flag: Error signaling (see 6.5.2).   |                  |
| Priority            | PR   | Flags: indicates the priority of the requested asynchronous frame. (see 4.2.4.1.2.3).  |                  |
| RequestToSend       | RS   | Flags: indicates the number of pending requests to send at the CN. The value C_DLL_MAX_RS indicates C_DLL_MAX_RS or more requests, 0 indicates no pending request.   | 0 – C_DLL_MAX_RS |
| NMTStatus           | stat | Reports the current status of the CN's NMT state machine.  |                  |
| StaticErrorBitfield | seb  | Specific bits are set to indicate pending errors at the CN (see 6.5.8.1 for encoding of errors in the bit field).  |                  |
| ErrorCodeList       | el   | Contains a list of errors, that have occurred at the CN. Each Error Code has a size of 20 octets (see 6.5.8.2). The lower layer is responsible for padding. Maximum size of ErrorCodeList is device specific. It is indicated by D_NMT_ErrorEntries_U32. |                  |

Tab. 137 NMT Service Slot data fields of StatusResponse

### 7.3.4 NMT Info Services

NMT Info Services are used to transmit complex status information in the form of bundles as well as to distribute system-relevant setup information from the MN to the CNs.

All NMT Info services are transmitted via ASnd by the MN.

The target CNs are unicast via the ASnd Destination field to one CN or broadcast to all CNs.

NMT Info Services are identified by ASnd ServiceID = NMT\_COMMAND.

Fig. 84 shows the protocol used to implement the NMT Info services.

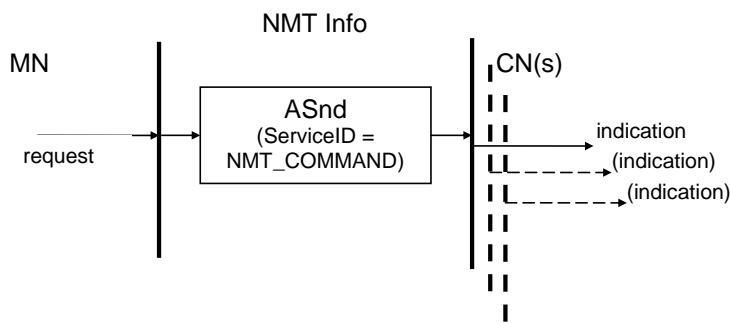


Fig. 84. NMT info service protocol

The particular NMT Info services including their data are located in the NMT Service Slot.

The NMT Info Services use the following parameters of the NMT Service slot:

| Field          | Abbr. | Description   | Value    |
|----------------|-------|---|----------|
| NMTCommandID   | cid   | qualifies the NMT Info service.   | Tab. 139 |
| NMTCommandData | cdat  | 0..C_DLL_MAX_PAYL_OFFSET – 4 octets of NMT Info Service-specific data to be issued by the MN. The lower layers are responsible for padding. |          |

Tab. 138 NMT Service Slot data fields of NMT managing info services

The following NMT Info services are defined:

| NMTCommandID              | Producer |    | Consumer |    | Short description  |
|---------------------------|----------|----|----------|----|--|
|                           | MN       | CN | MN       | CN |  |
| NMTPublishConfiguredNodes | Y        | Y  | Y        | Y  | Provides CNs declared in the configuration of the MN.      |
| NMTPublishActiveNodes     | Y        | N  | Y        | Y  | Provides CNs that have been identified by the MN.          |
| NMTPublishPreOperational1 | Y        | N  | Y        | Y  | Provides active CNs in the state NMT_CS_PRE_OPERATIONAL_1. |
| NMTPublishPreOperational2 | Y        | N  | Y        | Y  | Provides active CNs in the state NMT_CS_PRE_OPERATIONAL_2. |
| NMTPublishReadyToOperate  | Y        | N  | Y        | Y  | Provides CNs in the state NMT_CS_READY_TO_OPERATE.         |
| NMTPublishOperational     | Y        | N  | Y        | Y  | Provides CNs in the state NMT_CS_OPERATIONAL.              |
| NMTPublishStopped         | Y        | N  | Y        | Y  | Provides CNs in the state NMT_CS_STOPPED.                  |
| NMTPublishNodeStates      | Y        | N  | Y        | Y  | Provides current NMT states                                |
| NMTPublishEmergencyNew    | Y        | N  | Y        | Y  | Provides active CNs with the exception new flag (EN) set.  |
| NMTPublishTime            | Y        | Y  | Y        | Y  | Provides the system time.                                  |

Tab. 139 NMT info services

In the object dictionary support shall be indicated by Index NMT\_FeatureFlags\_U32 Bit 4. The flags do not differentiate particular NMT Info Services. They indicate support of at least one NMT Info Services defined by this specification.

Support of particular NMT Info Services shall be indicated by specific device description entries (see 7.3.4.1.1 to 7.3.4.1.10).

### 7.3.4.1 Service Descriptions

#### 7.3.4.1.1 NMTPublishConfiguredNodes

Using the NMTPublishConfiguredNodes service, the MN or a CN may publish a list of nodes configured in its configuration.

The NMTPublishConfiguredNodes service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to configured CNs are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_NodeAssignment\_AU32 sub-index Bit 1.

Support shall be indicated by D\_NMT\_PublishConfigNodes\_BOOL.

#### 7.3.4.1.2 NMTPublishActiveNodes

Using the NMTPublishActiveNodes service, the MN may publish a list of the active nodes.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices. CNs in the states NMT\_CS\_PRE\_OPERATIONAL\_1, NMT\_CS\_PRE\_OPERATIONAL\_2, NMT\_CS\_READY\_TO\_OPERATE, NMT\_CS\_OPERATIONAL and NMT\_CS\_STOPPED are regarded active.

The NMTPublishActiveNodes service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active CNs are indicated by 1<sub>b</sub>.

The MN itself is regarded to be ever active.

Support shall be indicated by D\_NMT\_PublishActiveNodes\_BOOL.

#### 7.3.4.1.3 NMTPublishPreOperational1

Using the NMTPublishPreOperational1 service, the MN may publish a list of nodes in the state NMT\_CS\_PRE\_OPERATIONAL\_1 resp. NMT\_MS\_PRE\_OPERATIONAL\_1.

The NMTPublishPreOperational1 service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active nodes in the state NMT\_CS\_PRE\_OPERATIONAL\_1 or NMT\_MS\_PRE\_OPERATIONAL\_1 are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices.

Support shall be indicated by D\_NMT\_PublishPreOp1\_BOOL.

#### 7.3.4.1.4 NMTPublishPreOperational2

Using the NMTPublishPreOperational2 service, the MN may publish a list of nodes in the state NMT\_CS\_PRE\_OPERATIONAL\_2 resp. NMT\_MS\_PRE\_OPERATIONAL\_2.

The NMTPublishPreOperational2 service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active nodes in the state NMT\_CS\_PRE\_OPERATIONAL\_2 or NMT\_MS\_PRE\_OPERATIONAL\_2 are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices.

Support shall be indicated by D\_NMT\_PublishPreOp2\_BOOL.

#### 7.3.4.1.5 NMTPublishReadyToOperate

Using the NMTPublishReadyToOperate service, the MN may publish a list of nodes in the state NMT\_CS\_READY\_TO\_OPERATE resp. NMT\_MS\_READY\_TO\_OPERATE.

The NMTPublishReadyToOperate service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active nodes in the state NMT\_CS\_READY\_TO\_OPERATE or NMT\_MS\_READY\_TO\_OPERATE are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices.

Support shall be indicated by D\_NMT\_PublishReadyToOp\_BOOL.

#### 7.3.4.1.6 NMTPublishOperational

Using the NMTPublishOperational service, the MN may publish a list of nodes in the state NMT\_CS\_OPERATIONAL resp. NMT\_MS\_OPERATIONAL.

The NMTPublishOperational service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active CNs in the state NMT\_CS\_OPERATIONAL or NMT\_MS\_OPERATIONAL are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices.

Support shall be indicated by D\_NMT\_PublishOperational\_BOOL.

#### 7.3.4.1.7 NMTPublishStopped

Using the NMTPublishStopped service, the MN may publish a list of CNs in the state NMT\_CS\_STOPPED.

The NMTPublishStopped service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to active CNs in the state NMT\_CS\_STOPPED are indicated by 1<sub>b</sub>.

Information to be published is obtained from NMT\_MNNodeCurrState\_AU8 sub-indices.

Support shall be indicated by D\_NMT\_PublishStopped\_BOOL.

#### 7.3.4.1.8 NMTPublishNodeStates

Using the NMTPublishNodeStates service, the MN may publish the current NMT states of the POWERLINK nodes as listed in NMT\_MNNodeCurrState\_AU8 (7.2.2.4.1).

|                            | Bit Offset |  |   |   |   |   |   |   |
|----------------------------|------------|--|---|---|---|---|---|---|
| Octet Offset <sup>34</sup> | 7          |  | 5 | 4 | 3 | 2 | 1 | 0 |
| 0                          | reserved   |  |   |   |   |   |   |   |
| 1 .. 254                   | Node State |  |   |   |   |   |   |   |

Tab. 140 NMTCommandData structure of NMTPublishNodeStates

| Field      | Abbr. | Description  | Value |
|------------|-------|--|-------|
| Node State | dt    | Current NMT state of POWERLINK Nodes<br>Octet Offset = POWERLINK Node ID |       |

Tab. 141 NMTCommandData data fields of NMTPublishNodeStates

<sup>34</sup> Octet offset is measured from the beginning of NMTCommandData. The offset relative to the Ethernet frame is 20 octets.

The octet offset not corresponding to a node configured by index NMT\_NodeAssignment\_AU32[sub-index] Bits 1 and 2 shall be set to NMT\_GS\_OFF.

Support shall be indicated by D\_NMT\_PublishNodeState\_BOOL.

### 7.3.4.1.9 NMTPublishEmergencyNew

Using the NMTPublishEmergencyNew service, the MN may publish a list of nodes with the Exception New flag (EN) set in the NMTStatusResponse feedback.

The NMTPublishEmergencyNew service uses the POWERLINK Node List format (see 7.3.1.2.3). Node IDs that correspond to nodes with the EN flag set are indicated by 1<sub>b</sub>.

Support shall be indicated by D\_NMT\_PublishEmergencyNew\_BOOL.

### 7.3.4.1.10 NMTPublishTime

Using the NMTPublishTime service, the MN or a CN publishes date and time.

|                            |  | Bit Offset |   |   |   |   |   |   |   |
|----------------------------|--|------------|---|---|---|---|---|---|---|
| Octet Offset <sup>35</sup> |  | 7          | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 .. 5                     |  | DateTime   |   |   |   |   |   |   |   |

Tab. 142 NMTCmdData structure of NMTPublishTime

| Field    | Abbr. | Description  | Value |
|----------|-------|--|-------|
| DateTime | dt    | Current RTC time value of the MN coded as TIME_OF_DAY. |       |

Tab. 143 NMTCmdData data fields of NMTPublishTime

Support shall be indicated by D\_NMT\_PublishTime\_BOOL

## 7.3.5 NMT Guard Services

MN and CNs may detect failures in an Ethernet POWERLINK-based network using NMT Guard Services. NMT Guard services are optional.

Local errors in a node may for example lead to a reset or change of state. The definition of such local errors is not within the scope of this specification.

### 7.3.5.1 Guarding CNs

Ethernet POWERLINK may use a guard mechanism to monitor nodes. The MN queries the CNs and receives their replies.

In order to query isochronously-addressed CNs in the POWERLINK Mode, the guard mechanism uses the isochronous PReq / PRes message exchange.

The supervising node monitors the reception of PRes frames from the monitored node. If the PRes frames do not arrive in time, the supervising node informs its application.

Due to the multicast transmission of the PRes frame, CNs are able to monitor other CNs.

If the MN transmits PRes, it may be monitored by the CNs too.

The monitoring time is configurable via the object dictionary entry

NMT\_ConsumerHeartbeatTime\_AU32.

#### 7.3.5.1.1 Guarding Async-Only CNs

An asynchronous channel is allocated to nodes which are not addressed isochronously. They are queried by the MN via a StatusRequest SoA telegram and respond with a StatusResponse ASnd message. If no StatusResponse is received within the monitoring time, the MN informs its application.

The guarded CN is the only one that receives the StatusRequest, other CNs shall ignore it. They shall not rely on a StatusRequest frequency satisfying their guard time requirements regarding the respective CN. That's why, no guarding of async-only CNs by other CNs is possible.

Hint: StatusRequest and StatusResponse transmission is affected by the asynchronous scheduling mechanism. The timing severely influenced by the sum of asynchronous transmission requests in the

<sup>35</sup> Octet offset is measured from the beginning of NMTCmdData. The offset relative to the Ethernet frame is 20 octets.

system. That's why, NMT\_ConsumerHeartbeatTime\_AU32 of async-only CNs shall be set to large values.

### 7.3.5.2 Guarding the MN

The CNs may control the function of the MN by timeout-monitoring the SoC frames (see 7.3.1.1). If a CN does not receive any frames, it changes to the state NMT\_CS\_PRE\_OPERATIONAL\_1. This transition is signalled to the CN's application.

## 7.3.6 Request NMT Services by a CN

A CN may request the execution of explicit NMT State Commands, NMT Management Commands, NMT Info Services, IdentResponse and StatusResponse services at the MN.

1. The CN indicates an NMTRRequest using the RS bits of the PRes frame or a StatusResponse ASnd Frame using the priority level C\_DLL\_ASND\_PRIO\_NMTRQST.
2. The MN assigns the asynchronous phase to the CN via SoA (RequestedServiceID = NMT\_REQUEST\_INVITE).
3. The CN requests the desired service using the ASnd NMTRRequest frame (ServiceID = NMT\_REQUEST).

The service is mandatory on the MN.

### 7.3.6.1 NMTRRequest Frame

NMTRRequests are transmitted by a CN via ASnd upon assignment of the asynchronous phase via an NMTRRequestInvite SoA frame.

The NMTRRequest frames are identified by ASnd ServiceID = NMT\_REQUEST.

| Octet offset <sup>36</sup> | Bit Offset                 |  |   |   |   |   |   |   |
|----------------------------|----------------------------|--|---|---|---|---|---|---|
|                            | 7                          |  | 5 | 4 | 3 | 2 | 1 | 0 |
| 0                          | NMTRRequestedCommandID     |  |   |   |   |   |   |   |
| 1                          | NMTRRequestedCommandTarget |  |   |   |   |   |   |   |
| 2 .. n                     | NMTRRequestedCommandData   |  |   |   |   |   |   |   |

$n \leq C\_DLL\_MAX\_PAYL\_OFFSET - 4$

Tab. 144 NMT Service Slot structure of NMTRRequest

| Field                       | Abbr. | Description   | Value |
|-----------------------------|-------|---|-------|
| NMTRRequested-CommandID     | rclid | The NMT service to be issued by the MN by its NMTRRequestedCommandID value.<br>StatusResponse and IdentResponse services are indicated by the SoA RequestedServiceID values STATUS_REQUEST and IDENT_REQUEST. |       |
| NMTRRequested-CommandTarget | rct   | Indicates the target node of the requested NMT command.   |       |
| NMTRRequested-CommandData   | rcd   | 0..C_DLL_MAX_PAYL_OFFSET – 6 octets of NMT command-specific data to be issued by the MN.<br>The lower layers are responsible for padding.   |       |

Tab. 145 NMT Service Slot data fields of an NMTRRequest frame

The NMTRRequest ASnd frame is unicast to the MN.

### 7.3.6.1.1 Invalid NMTRRequests

If the CN requests an NMT service not supported by the MN, the MN responds with a unicast ASnd frame with ServiceID = NMT\_COMMAND to the requesting CN.

The NMTRRequestedCommandID NMTInvalidService is used:

<sup>36</sup> Octet offset is measured from the begin of the ASnd Payload field. The offset relative to the beginning of the Ethernet frame is 18 octets.

This special service is mandatory. It must not influence the state machine of the CNs and must not transport any data to the CN.

## 7.3.7 NMT Services via Object Dictionary

### 7.3.7.1 NMT Reset Commands

The NMT Command Services NMTSwReset, NMTResetNode, NMTResetConfiguration, and NMTResetCommunication may be initiated by writing the respective NMTServiceID to NMT\_ResetCmd\_U8.

*NMT Reset via NMT\_ResetCmd\_U8 should be applied to CNs in NMT\_CS\_BASIC\_ETHERNET only. If applied to nodes in NMT\_CS\_EPL\_MODE or NMT\_MS\_EPL\_MODE, resets by NMT\_ResetCmd\_U8 may violate the NMT rules and stimulate DLL and NMT Guarding errors. Use NMT Request via Object Dictionary (see 7.3.7.2) instead.*

### 7.3.7.2 NMT Requests to the MN

NMT Requests may be posted to the MN by writing to NMT\_RequestCmd\_REC.

The objects sub-indices shall be mapped to the NMT Service Slot of a virtual NMTReset Frame (see 7.3.6.1):

|                                 |   |                            |
|---------------------------------|---|----------------------------|
| NMT_RequestCmd_REC.CmdID_U8     | → | NMTRequested-CommandID     |
| NMT_RequestCmd_REC.CmdTarget_U8 | → | NMTRequested-CommandTarget |
| NMT_RequestCmd_REC.CmdData_DOM  | → | NMTRequested-CommandData   |

The NMT Request shall be triggered by writing TRUE to NMT\_RequestCmd\_REC.Release\_BOOL.

The NMT Services via Object Dictionary mechanism does not provide a response channel. Therefore, only Explicit NMT State Command Services (see 7.3.1.2) may be requested via Object Dictionary.

## 7.3.8 NMT Services via UDP/IP

NMT requests coming from a diagnostic node outside of the POWERLINK segment via a router may be hosted by UDP/IP frames. Hosting shall be in accordance to the POWERLINK compliant UDP/IP format (see 4.6.1.2).

The NMT Services via UDP/IP mechanism does not provide a response channel. Therefore, only Explicit NMT State Command Services (see 7.3.1.2) may be requested via UDP/IP.

Support of NMT Services via UDP/IP by an MN is optional. Support shall be indicated by object dictionary entry NMT\_FeatureFlags\_U32 Bit 7 and D\_NMT\_ServiceUdplp\_BOOL.

## 7.4 Boot-up Managing Node

### 7.4.1 NMT\_MS dependant Network Boot-up

#### 7.4.1.1 Overview

When a POWERLINK Node starts after PowerOn (NMT\_GT1), Reset (NMT\_GT2) or NMTSwReset (NMT\_GT8), the state machine will begin to execute according to the NMT state diagram of a POWERLINK node (see 7.1.2) and will attain the NMT\_GS\_INITIALISATION state automatically. After completion of the initialisation, the node will enter the super state NMT\_GS\_COMMUNICATING. In this super state the node will perform as either a Managing Node (MN) or Controlled Node (CN) depending on the configured Node ID. The state NMT\_GS\_INITIALISATION is reachable from every state within the super state after processing the NMT command service NMTResetNode (NMT\_GT4), NMTResetCommunication (NMT\_GT5), NMTResetConfiguration (NMT\_GT7), NMTSwReset (NMT\_GT8) or after internal errors (NMT\_GT6).

#### 7.4.1.2 NMT\_MS\_NOT\_ACTIVE

There shall be only one active MN on a POWERLINK network. The purpose of the state NMT\_MS\_NOT\_ACTIVE is to enable a potential MN to detect whether the bus is already managed by another MN.

Detecting SoC or SoA during this state indicates to the potential MN that the network already has an active MN.

The state NMT\_MS\_NOT\_ACTIVE is entered from the NMT\_GS\_INITIALISATION state (NMT\_MT1).

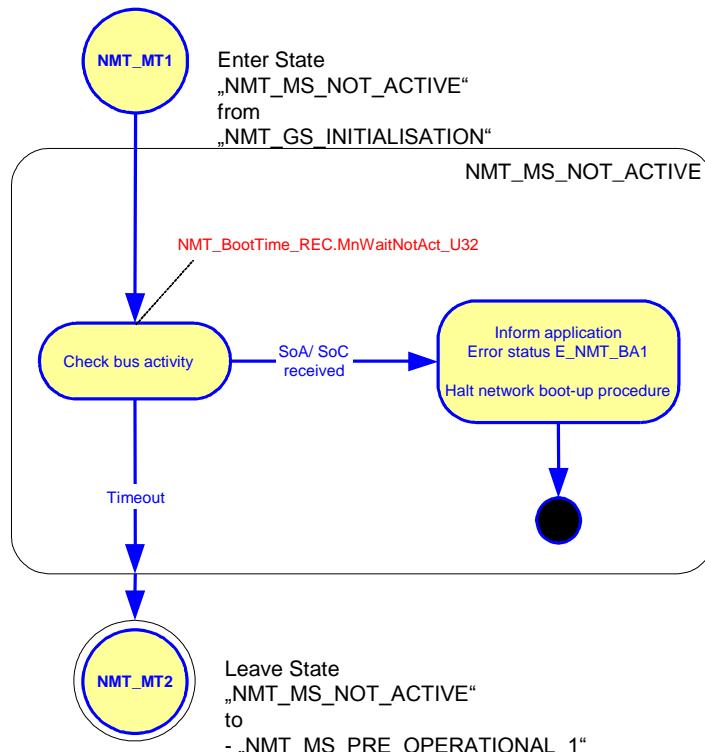


Fig. 85. State NMT\_MS\_NOT\_ACTIVE

In the state NMT\_MS\_NOT\_ACTIVE the MN proceeds as follows:

- Check for bus activity. Reception of SoC or SoA frames indicates that another MN is currently managing the network. In case of the reception of SoA or SoC frames, the error
  - E\_NMT\_BA1 (see 7.4.3.1)
 shall be signalled to the MN application.  
 The current MN state shall be maintained. The MN shall halt network boot-up procedure.
- If the potential MN does not detect either SoC or SoA frames within a time interval defined by index NMT\_BootTime\_REC.MNWaitNoAct\_U32, it shall transition from NMT\_MS\_NOT\_ACTIVE to NMT\_MS\_PRE\_OPERATIONAL\_1.

### 7.4.1.3 NMT\_MS\_PRE\_OPERATIONAL\_1

In the state NMT\_MS\_PRE\_OPERATIONAL\_1 the MN assumes bus master-ship. It will attempt to reach all configured CNs and examine their configuration versions. The MN communicates with the CN using the Reduced POWERLINK Cycle (SoA and ASnd frames) (see 4.2.4.2). In case of communication errors such as frame losses, the MN signals the error to the application. If the error was caused by a mandatory node, the MN shall remain in this state.

This state is entered from the NMT\_MS\_NOT\_ACTIVE (NMT\_MT2) or the NMT\_MS\_OPERATIONAL (NMT\_MT6) state.

The steps of the network boot process in NMT\_MS\_PRE\_OPERATIONAL\_1 are as follows:

- If the state was entered from NMT\_MS\_NOT\_ACTIVE (NMT\_MT2), all nodes of the network will be reset using the NMT command service NMTResetNode.
- Initiate boot process BOOT\_STEP1 for all configured CNs.  
 Configured CNs are identified by:
  - NMT\_NodeAssignment\_AU32[1..254].Bit0
  - NMT\_NodeAssignment\_AU32[1..254].Bit1
- If mandatory CNs are booted sequentially, the boot process shall not be halted in case of an error associated with the particular CN. The MN shall continue to attempt to contact all configured but not responding CNs.
- If an application SW update of one or more nodes was done, the MN must send out the NMT command
  - NMTSwReset
 to all CNs and restart the BOOT\_STEP1 for all configured CNs.
- When the BOOT\_STEP1 process has been successfully terminated for all mandatory CNs, the MN may switch to the state NMT\_MS\_PRE\_OPERATIONAL\_2  
 Mandatory CNs are identified by :
  - NMT\_NodeAssignment\_AU32[1..254].Bit3
- The MN can switch to the state NMT\_MS\_PRE\_OPERATIONAL\_2 either automatically or under control of the application. The transition policy is defined by:
  - NMT\_StartUp\_U32.Bit7.
 If the waiting time interval
  - NMT\_BootTime\_REC.MnWaitPreOp1\_U32
 has not expired, the MN waits in this state until the waiting time expires.
- If BOOT\_STEP1 has not been processed successfully for all mandatory CNs, the MN shall signal the error
  - E\_NMT\_BPO1 (see 7.4.3.2)
 to the MN application.  
 In this case the MN shall halt the network boot-up procedure and maintain its current state.
- If optional CNs answered with IdentResponse successfully, but could not finish BOOT\_STEP1 without errors, the application will be notified. In this case the network boot procedure for these particular CNs shall be restarted.

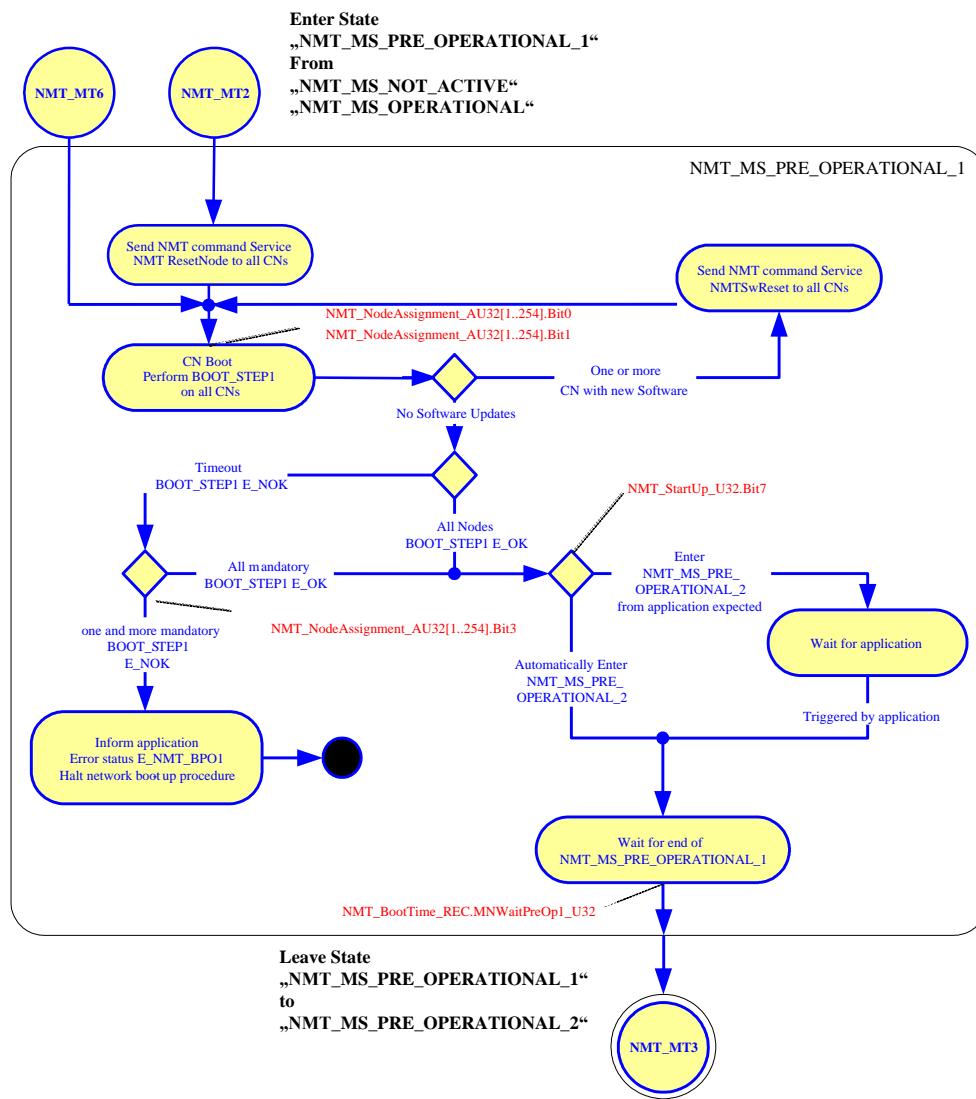


Fig. 86. Detail state NMT\_MS\_PRE\_OPERATIONAL\_1

#### 7.4.1.4 NMT\_MS\_PRE\_OPERATIONAL\_2

In the state NMT\_MS\_PRE\_OPERATIONAL\_2 the MN will initiate isochronous communication with the CNs and insure their transition from NMT\_CS\_PRE\_OPERATIONAL\_2 to NMT\_CS\_READY\_TO\_OPERATE.

When all mandatory CNs have reached the NMT\_CS\_READY\_TO\_OPERATE state, the MN will change to the state NMT\_MS\_READY\_TO\_OPERATE.

In case of an error associated with the mandatory CNs, the MN will signal the error to the application. In this case the MN shall halt the network boot-up procedure and maintain its current state.

The NMT\_MS\_PRE\_OPERATIONAL\_2 state is entered from the NMT\_MS\_PRE\_OPERATIONAL\_1 (NMT\_MT3) state.

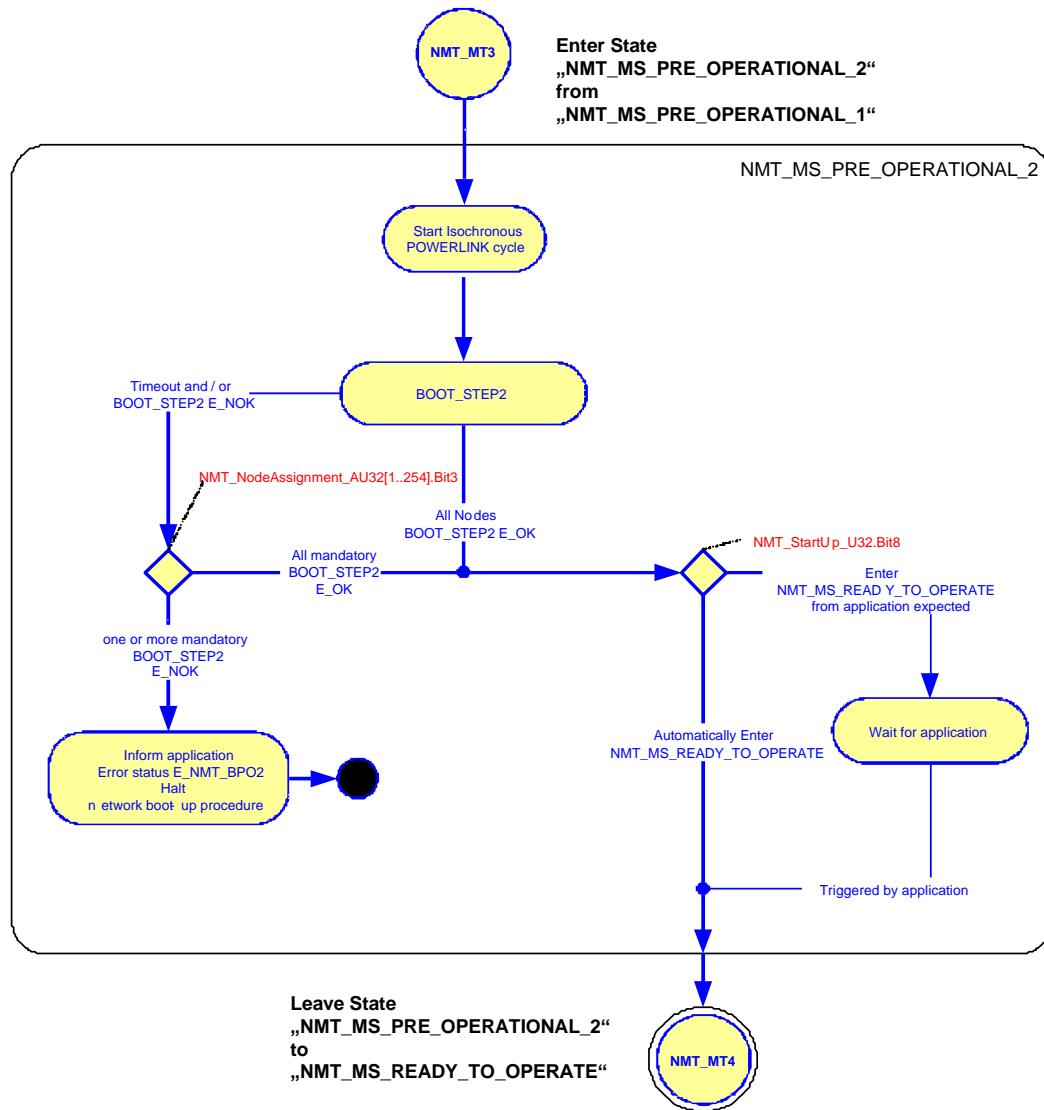


Fig. 87. Detail state NMT\_MS\_PRE\_OPERATIONAL\_2

The network boot steps and tasks in the NMT\_MS\_PRE\_OPERATIONAL\_2 state are as follows:

- Start isochronous POWERLINK Cycle with SoC /SoA frames to synchronise the CNs to the MN POWERLINK Cycle.
  - Start process BOOT\_STEP2 for all configured CNs. Configured CNs are identified by:
    - NMT\_NodeAssignment\_AU32[1..254].Bit0
    - NMT\_NodeAssignment\_AU32[1..254].Bit1
  - After the BOOT\_STEP2 process is successfully terminated for all mandatory CNs (all mandatory CNs are in the NMT\_CS\_READY\_TO\_OPERATE state), the MN itself shall switch to the state NMT\_MS\_READY\_TO\_OPERATE direct or after application trigger, depending on the following object:
    - NMT\_StartUp\_U32.Bit8
- Mandatory CNs are identified by:
- NMT\_NodeAssignment\_AU32[1..254].Bit3

In case of errors or timeout condition during the BOOT\_STEP2 process of one or more mandatory CN, the error

- E\_NMT\_BPO2 (see 7.4.3.3)

shall be signaled to the application.

- Errors of optional CNs in BOOT\_STEP2 shall be signaled to the application. The network boot-up process for these CNs shall be restarted.

- Keep trying to engage previous boot-up processes for all configured optional CNs.

### 7.4.1.5 NMT\_MS\_READY\_TO\_OPERATE

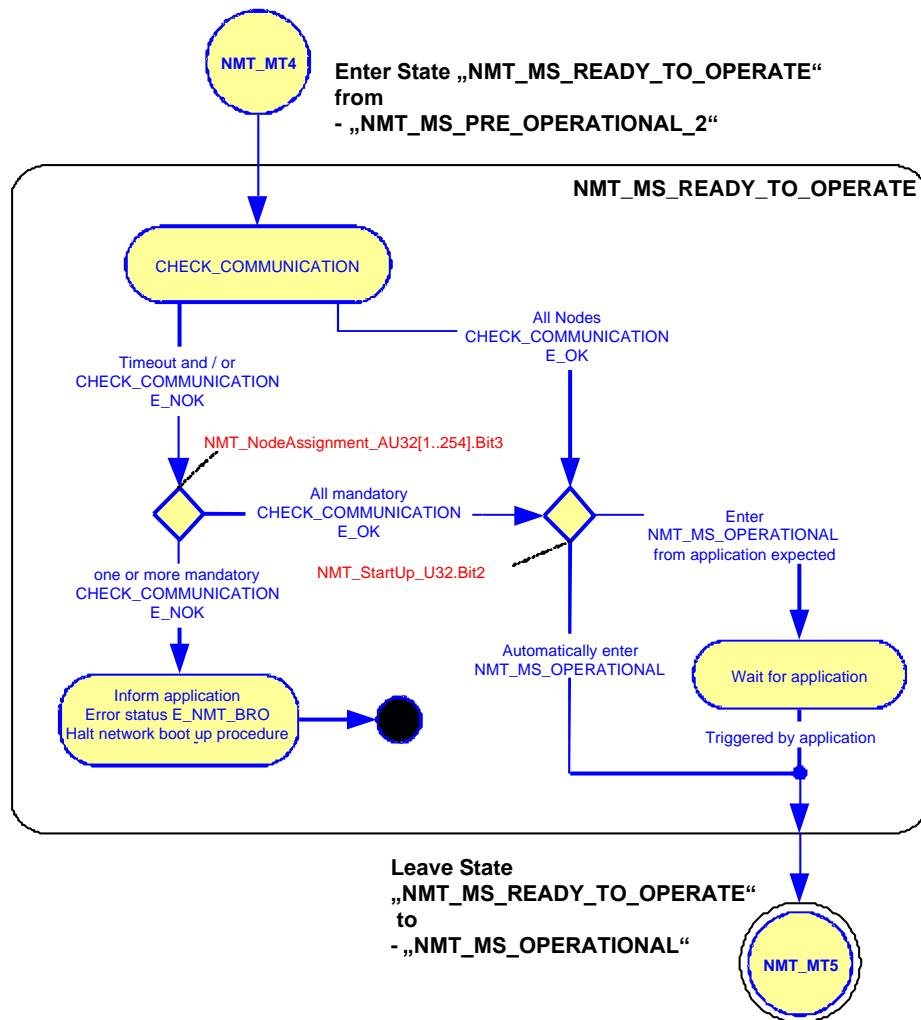


Fig. 88. Detail state NMT\_MS\_READY\_TO\_OPERATE

The purpose of the network boot process in NMT\_MS\_READY\_TO\_OPERATE state is to start and check the isochronous communication with all identified isochronous CNs. The length of the isochronous frames shall be checked against the configured values of NMT\_PResPayloadLimitList\_AU16.

If isochronous communication with all mandatory CNs operates correctly, the MN may enter the NMT\_MS\_OPERATIONAL state.

In case of communication or configuration errors within one or more mandatory CNs, the MN signals the error to the application and shall remain in this state.

This NMT state is entered from the NMT\_MS\_PRE\_OPERATIONAL\_2 state (NMT\_MT4).

In the state NMT\_MS\_READY\_TO\_OPERATE the MN proceeds as follows:

- Start process CHECK\_COMMUNICATION for all configured CN. Configured CNs are identified by:
    - NMT\_NodeAssignment\_AU32[1..254].Bit0
    - NMT\_NodeAssignment\_AU32[1..254].Bit1
  - After the CHECK\_COMMUNICATION process is terminated successfully for all mandatory CNs, the MN shall switch to the state NMT\_MS\_OPERATIONAL direct or after the application trigger. The transition policy is defined by:
    - NMT\_StartUp\_U32.Bit2
- Mandatory CNs are identified by:

- NMT\_NodeAssignment\_AU32[1..254].Bit3

If the CHECK\_COMMUNICATION process of one or more mandatory CN failed, the error:

- E\_NMT\_BRO (see 7.4.3.4)

shall be signaled to the application.

The current MN state shall be maintained. The MN shall halt the network boot-up procedure.

- Errors of optional CNs during the CHECK\_COMMUNICATION process shall be signaled to the application. The boot process for these CN shall be halted. The restart of the boot-up process for these CNs may be triggered by the application.
- Keep trying to engage previous boot processes for all configured optional CNs.

#### 7.4.1.6 NMT\_MS\_OPERATIONAL

In the state NMT\_MS\_OPERATIONAL, the MN forces all CNs that are in the state NMT\_CS\_READY\_TO\_OPERATE to the state NMT\_CS\_OPERATIONAL.

In the NMT\_MS\_OPERATIONAL state the MN continues isochronous communication with all identified isochronous CNs.

The MN can force each CN to NMT\_CS\_OPERATIONAL either individually, with the NMTStartNode unicast command, or all CNs, with the NMTStartNode broadcast command. After all identified CNs are in the state NMT\_CS\_OPERATIONAL, the MN shall start the process OPERATIONAL for all identified CNs.

In case of communication errors of mandatory nodes, the sub-state ERROR\_TREATMENT of the network boot-up procedure is entered. Finally the MN enters the state NMT\_MS\_PRE\_OPERATIONAL\_1.

The NMT\_MS\_OPERATIONAL state is entered from the NMT\_MS\_READY\_TO\_OPERATE state (NMT\_MT5).

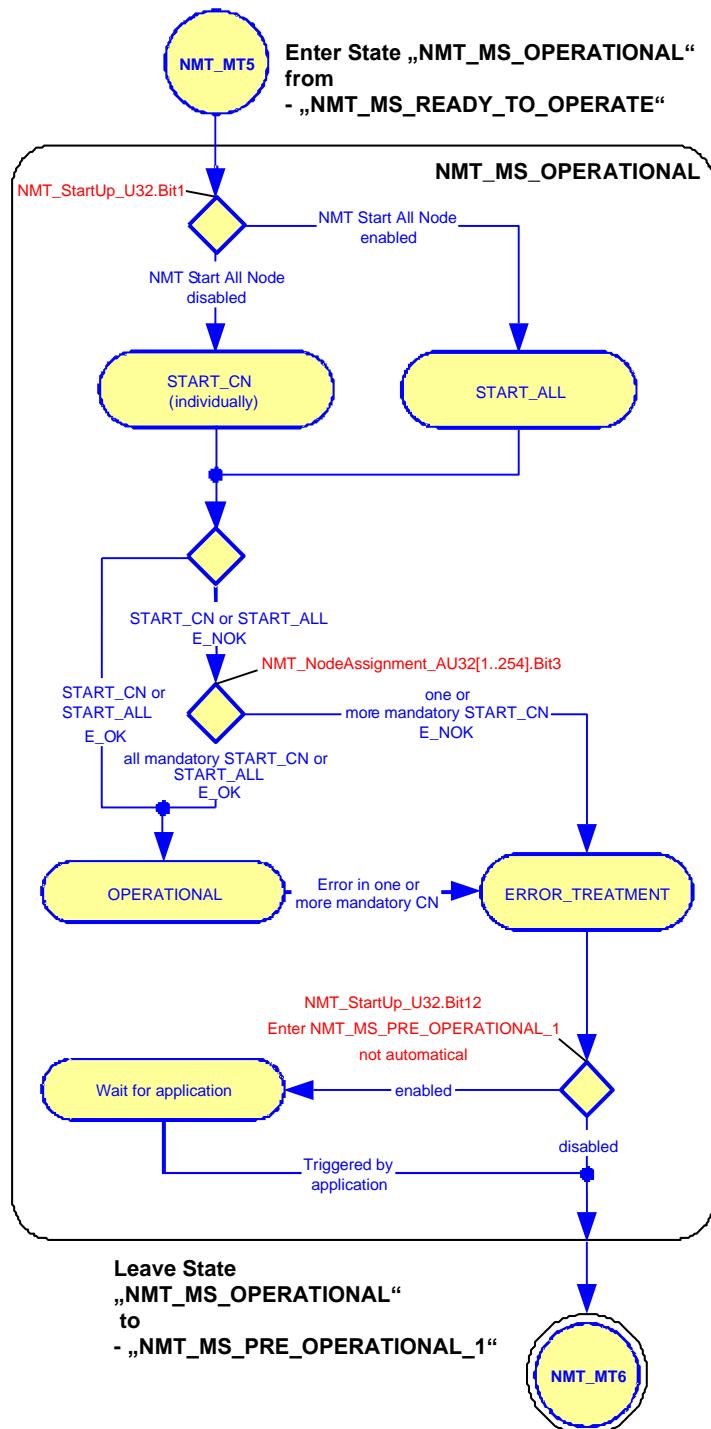


Fig. 89. Detail state NMT\_MS\_OPERATIONAL

The network boot steps and tasks of the NMT\_MS\_OPERATIONAL state are as follows:

- Start process START\_ALL or START\_CN either for all CNs or individually depending on the following object:
  - NMT\_StartUp\_U32.Bit1
 Configured CNs are identified by:
  - NMT\_NodeAssignment\_AU32[1..254].Bit0
  - NMT\_NodeAssignment\_AU32[1..254].Bit1,
- In case of errors of mandatory nodes during START\_CN or START\_ALL, the MN shall start the ERROR\_TREATMENT process. Finally the MN enters state NMT\_MS\_PRE\_OPERATIONAL\_1 direct or after application trigger, depending on the following object:

- NMT\_StartUp\_U32.Bit12

If START\_ALL or START\_CN for all mandatory nodes finished successfully, the MN shall enter the boot-up sub-state OPERATIONAL.

- In case of communication errors during OPERATIONAL of one or more mandatory node, the MN shall enter the ERROR\_TREATMENT sub-state. Finally the MN shall enter NMT\_MS\_PRE\_OPERATIONAL\_1 direct or after the application trigger, depending on the following object:
  - NMT\_StartUp\_U32.Bit12
- Errors of optional CNs during the START\_CN, START\_ALL or OPERATIONAL process shall be signaled to the application. The boot process for these CN shall be halted. The restart of the boot-up process for these CNs may be triggered by the application.
- Keep trying to engage previous boot processes for all configured optional CNs.

## 7.4.2 MN Boot-up Procedure on CN Level

### 7.4.2.1 Overview

The previous chapter describes how the MN performs the network boot-up. This chapter describes how the MN manages the boot-up procedure for a single CN.

On network level, optional and mandatory CNs are handled differently. On CN level the handling is the same.

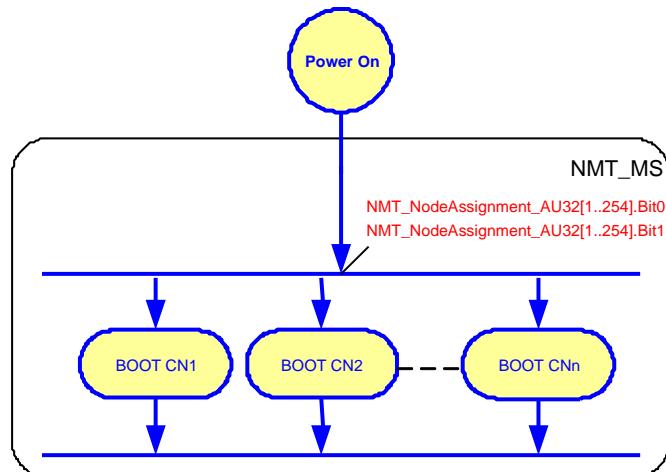


Fig. 90. Overview of the boot process in NMT super-state NMT\_MS

### 7.4.2.2 Boot-up of optional and mandatory CNs

Optional CNs are able to link in the network everytime. For the consistency and errorless working of the network it is important to run the complete boot process also for the optional nodes. Therefore the optional CNs must be booted-up without any impact on the MN NMT state. Errors during the boot-up process of optional CNs do not cause an NMT state change of the MN. The application shall be informed about the errors.

The mandatory CN boot process correlates directly to the MN NMT states. Only if all mandatory CNs reach the same target NMT state within a sub-state of the boot process, the MN will proceed with the boot process.

The MN force all CNs into a specific NMT state. Fig. 91 shows the dependencies between the MN NMT state and the boot process actions. Optional CNs are able to be linked to the network later, even if the MN is already in the NMT\_MS\_OPERATIONAL state .

It is possible to carry out the boot steps of one NMT state in parallel for all nodes or in serial i.e. one CN after the other. For optional nodes, even boot steps of different NMT states can be carried out in parallel for different CNs as there must not be a common NMT state for all optional nodes in the network.

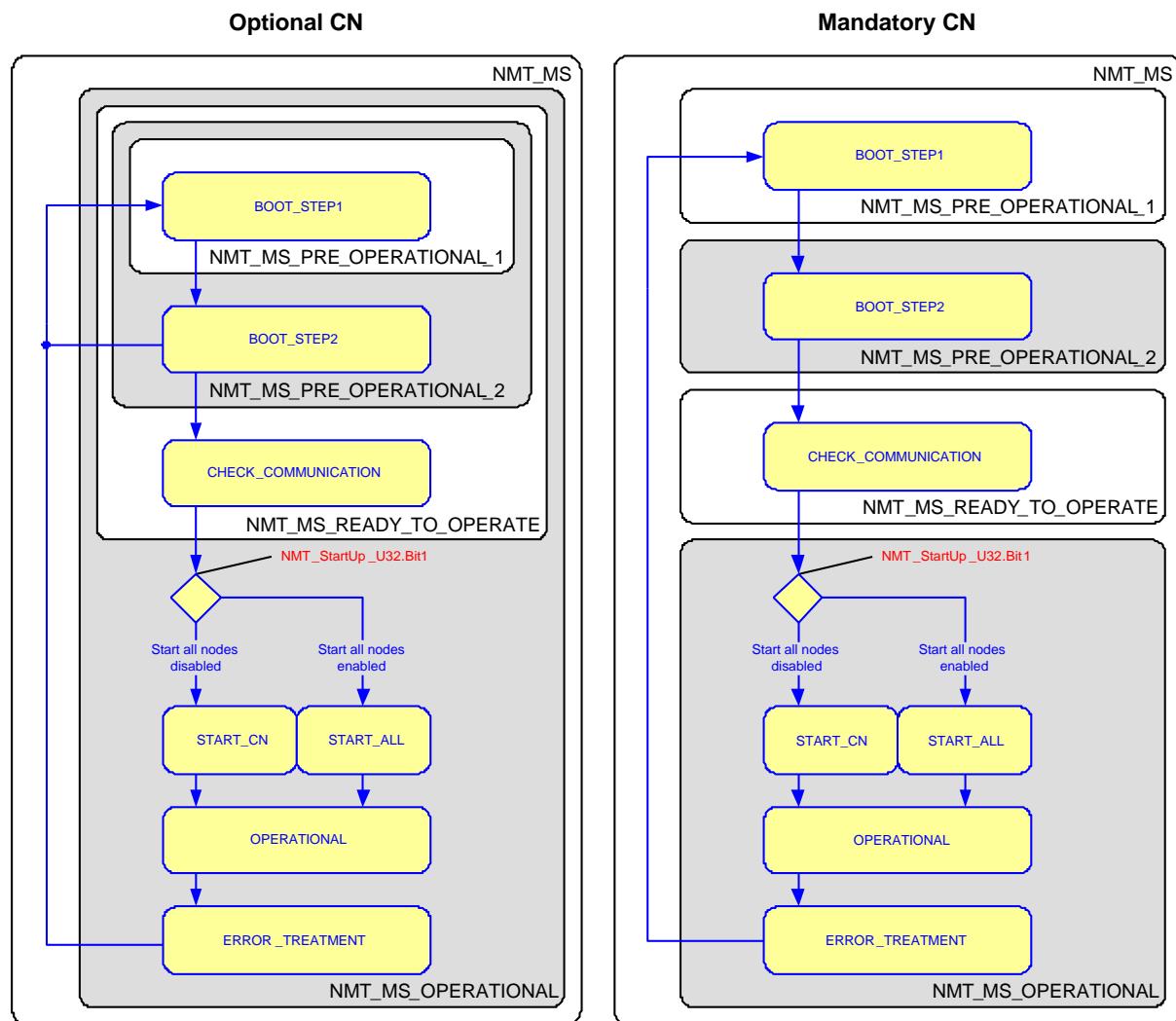


Fig. 91. Network boot process dependencies to the NMT\_MS for optional and mandatory CNs

### 7.4.2.2.1 BOOT\_STEP1

The BOOT\_STEP1 process is started for all configured CNs. The purpose of the BOOT\_STEP1 process is to check the identification of the configured CNs and to check the CN's software and configuration.

The BOOT\_STEP1[Node ID] returns E\_OK after all checks or updates are terminated successfully. If one of the checks or updates fails, BOOT\_STEP1[Node ID] returns E\_NOK.

The following steps are performed in BOOT\_STEP1:

- Check Identification (see chapter 7.4.2.2.1.1)
- Check Software (see chapter 7.4.2.2.1.2)
- Check Configuration (see chapter 7.4.2.2.1.3)

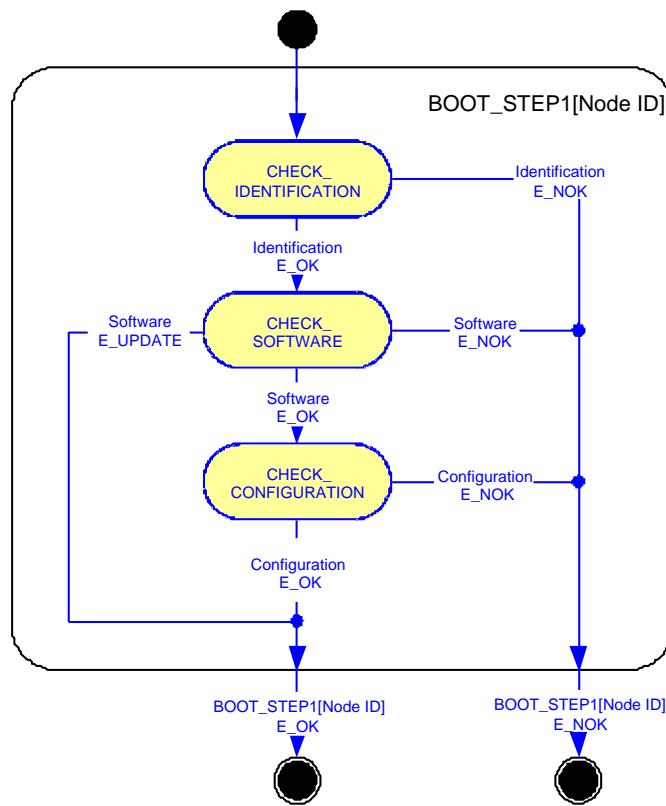


Fig. 92. Sub-state BOOT\_STEP1

#### 7.4.2.2.1.1 CHECK\_IDENTIFICATION

The purpose of the CHECK\_IDENTIFICATION state is to check the identification of a CN. CHECK\_IDENTIFICATION [Node ID] returns E\_OK after all identifications are finished successfully. If one of the identifications fails, CHECK\_IDENTIFICATION [Node ID] returns E\_NOK.

In the sub-state CHECK\_IDENTIFICATION the MN proceeds as follows:

- Request IdentResponse from the CN (see 7.3.3.2). In case of a timeout, the error E\_NMT\_BPO1\_GET\_IDENT is set and CHECK\_IDENTIFICATION[Node ID] is finished with E\_NOK.
- Check the device type of the CN against the following object:
  - NMT\_MNDeviceTypeldList\_AU32[Node ID].
 If the device type checking fails, the Error Status E\_NMT\_BPO1\_DEVICE\_TYPE is set and CHECK\_IDENTIFICATION[Node ID] finished with E\_NOK, otherwise the next step is performed.
- If the boot-up process of a CN shall not continue automatically, the boot process waits for an application trigger to continue. Otherwise the boot process is continued immediately. The switching policy depends on
  - NMT\_NodeAssignment\_AU32[1..254].Bit2
  - If identification check is not required depending on NMT\_StartUp\_U32.Bit9, CHECK\_IDENTIFICATION[Node ID] returns E\_OK.

If the Identification check is required, it is based on the following objects:

- NMT\_MNVendorIdList\_AU32[Node ID]
- NMT\_MNProductCodeList\_AU32[Node ID]
- NMT\_MNRevisionNoList\_AU32[Node ID]

If the identification check fails, CHECK\_IDENTIFICATION[Node ID] returns E\_NOK and error E\_NMT\_BPO1\_VENDOR\_ID, E\_NMT\_BPO1\_PRODUCT\_CODE or E\_NMT\_BPO1\_REVISION\_NO is set, otherwise the boot process continues.

- The CN serial number is checked based on the following object:

- NMT\_MNSerialNoList\_AU32[Node ID].

If the identification check fails, the error E\_NMT\_BPO1\_SERIAL\_NO is set and CHECK\_IDENTIFICATION[Node ID] returns E\_NOK. Otherwise CHECK\_IDENTIFICATION[Node ID] returns E\_OK.

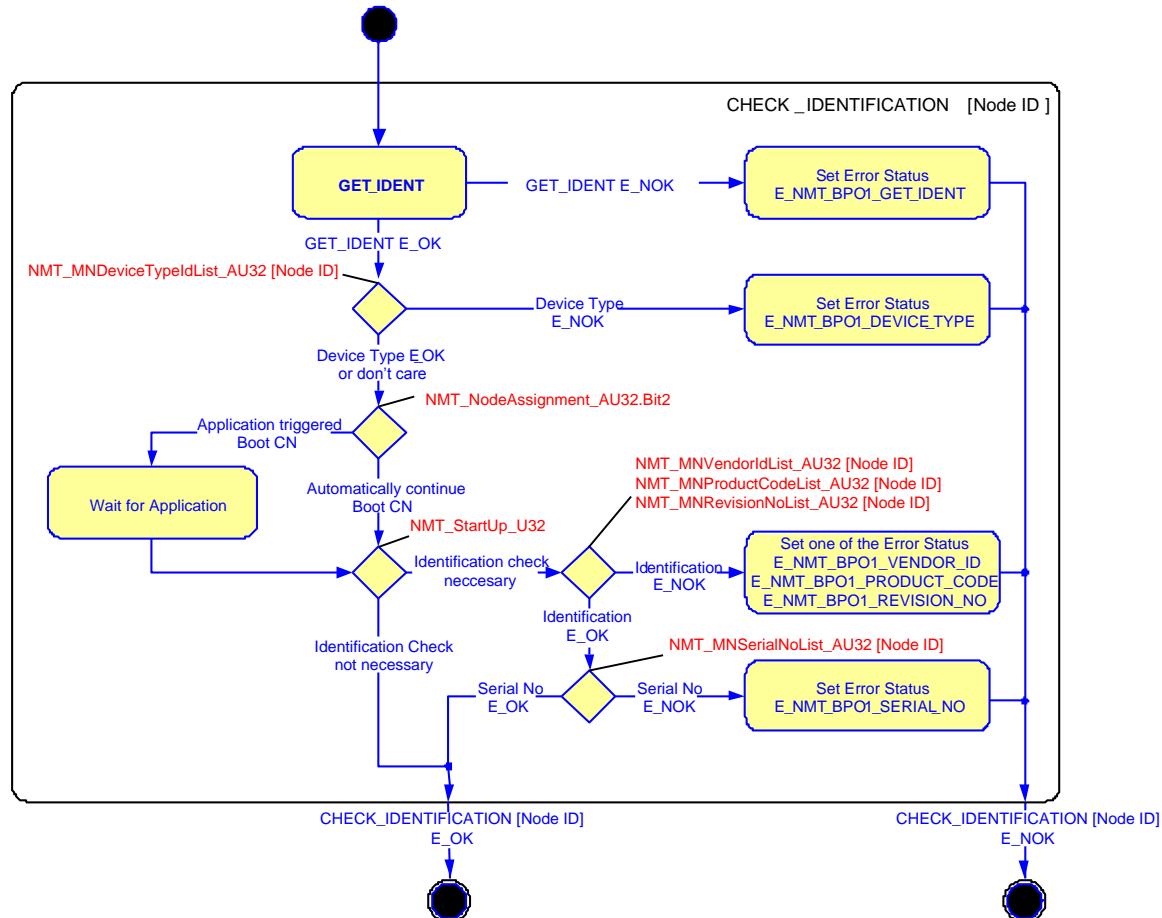


Fig. 93. Sub-state CHECK\_IDENTIFICATION[Node ID]

#### **7.4.2.2.1.2 CHECK\_SOFTWARE**

The purpose of CHECK\_SOFTWARE is to check and update the CN software, if required. The update process itself shall be done by the application. The application signals the result of the update process to the boot-up process.

**CHECK\_SOFTWARE [Node ID]** returns E\_OK after all software checks and updates are finished successfully. If an error occurred, **CHECK\_SOFTWARE [Node ID]** returns E\_NOK.

In the process CHECK SOFTWARE[Node ID] the MN proceeds as follows:

- If software version check is not required (depending on the following object:
    - NMT\_StartUp\_U32.Bit10,
    - NMT\_NodeAssignment\_AU32[Node ID].Bit5),

**CHECK\_SOFTWARE**[Node ID] returns E\_OK, otherwise the reported CN software version is checked against objects

- PDL\_MnExpAppSwDateList\_AU32[Node ID],
  - PDL\_MnExpAppSwTimeList\_AU32[Node ID]

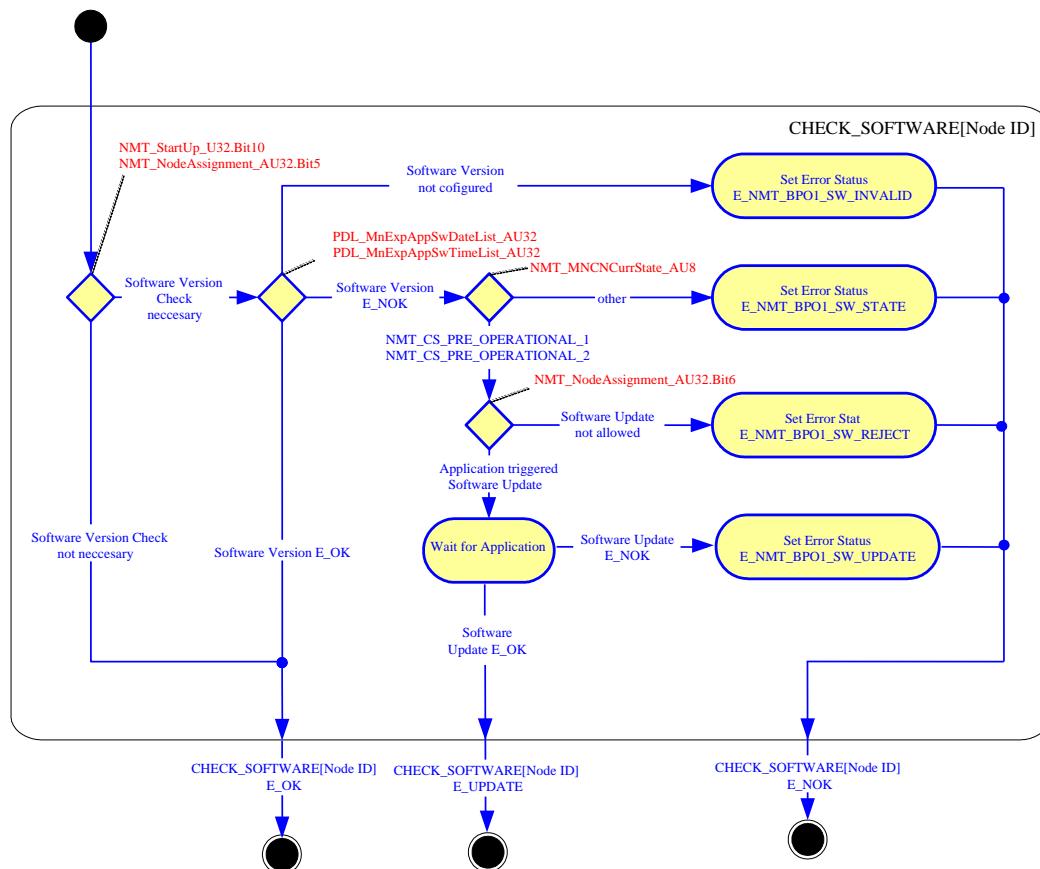


Fig. 94. Sub-state CHECK\_SOFTWARE[Node ID]

- If the checking of the software version is not possible because the software version information on the MN is not configured (corresponding node sub-index of PDL\_MnExpAppSwDateList\_AU32 or PDL\_MnExpAppSwTimeList\_AU32 not found), the error E\_NMT\_BPO1\_SW\_INVALID is set and CHECK\_SOFTWARE[Node ID] returns E\_NOK. If the received software version matches the expected one, CHECK\_SOFTWARE[Node ID] returns E\_OK, otherwise the process continues with the NMT state check of the CN.
- If the current NMT state of the CN is not equal to NMT\_CS\_PREOPERATIONAL\_1 or NMT\_CS\_PREOPERATIONAL\_2 (check based on object
  - NMT\_MNNodeCurrState\_AU8),
 The error E\_NMT\_BPO1\_SW\_STATE is set and CHECK\_SOFTWARE[Node ID] returns E\_NOK, otherwise the next step is proceeded.
- If software update is not allowed (depending on the setting of object
  - NMT\_NodeAssignment\_AU32[Node ID].Bit6),
 the error E\_NMT\_BPO1\_SW\_REJECT is set and CHECK\_SOFTWARE[Node ID] returns E\_NOK. If the software update is allowed, the application shall be informed. Continue of the boot-up process shall be triggered by the application update return code.
- The software update itself is always part of the application. Any protocol (SDO, FTP, ...) is allowed to be used for the application software update. A successful completion of the update is signalled by the application with the event E\_OK.
- If there was an error during the application software update, the application will signal this to the boot-up process by returning E\_NOK. In case of a software update error, the boot step returns E\_NOK.
- Running BOOT\_STEP1 for an optional CN while the MN is already in NMT\_MS\_PRE\_OPERATIONAL\_2 or higher, the NMTSwReset command shall be sent to the CNs individually after leaving CHECK\_SOFTWARE with E\_UPDATE.

### 7.4.2.2.1.3 CHECK\_CONFIGURATION

The purpose of CHECK\_CONFIGURATION is to check the configuration of a CN and to update the configuration if necessary. CHECK\_CONFIGURATION [Node ID] returns E\_OK, if the configuration fits, otherwise CHECK\_CONFIGURATION [Node ID] returns E\_NOK.

The steps of CHECK\_CONFIGURATION[Node ID] are as follows:

- If configuration verification is not required (configured by object

- NMT\_StartUp\_U32.Bit11),

CHECK\_CONFIGURATION returns E\_OK, otherwise the configuration is checked against objects

- CFM\_ExpConfDateList\_AU32[Node ID]

- CFM\_ExpConfTimeList\_AU32[Node ID]

If the configuration date and time received within the IdentResponse of the CN is correct, CHECK\_CONFIGURATION[Node ID] returns E\_OK. Otherwise the boot-up process continues with the next step.

- After the configuration update is finished successfully, the MN shall request an IdentResponse from the updated CN (see 7.3.3.2). If a timeout occurs, the error E\_NMT\_BPO1\_GET\_IDENT is set and CHECK\_CONFIGURATION[Node ID] returns E\_NOK, otherwise the boot-up process continues with the next step.

- Configuration date and time is checked against objects

- CFM\_ExpConfDateList\_AU32[Node ID]

- CFM\_ExpConfTimeList\_AU32[Node ID]

If the configuration check fails, the error E\_NMT\_BPO1\_CF\_VERIFY is set and CHECK\_CONFIGURATION[Node ID] returns E\_NOK, otherwise it returns E\_OK.

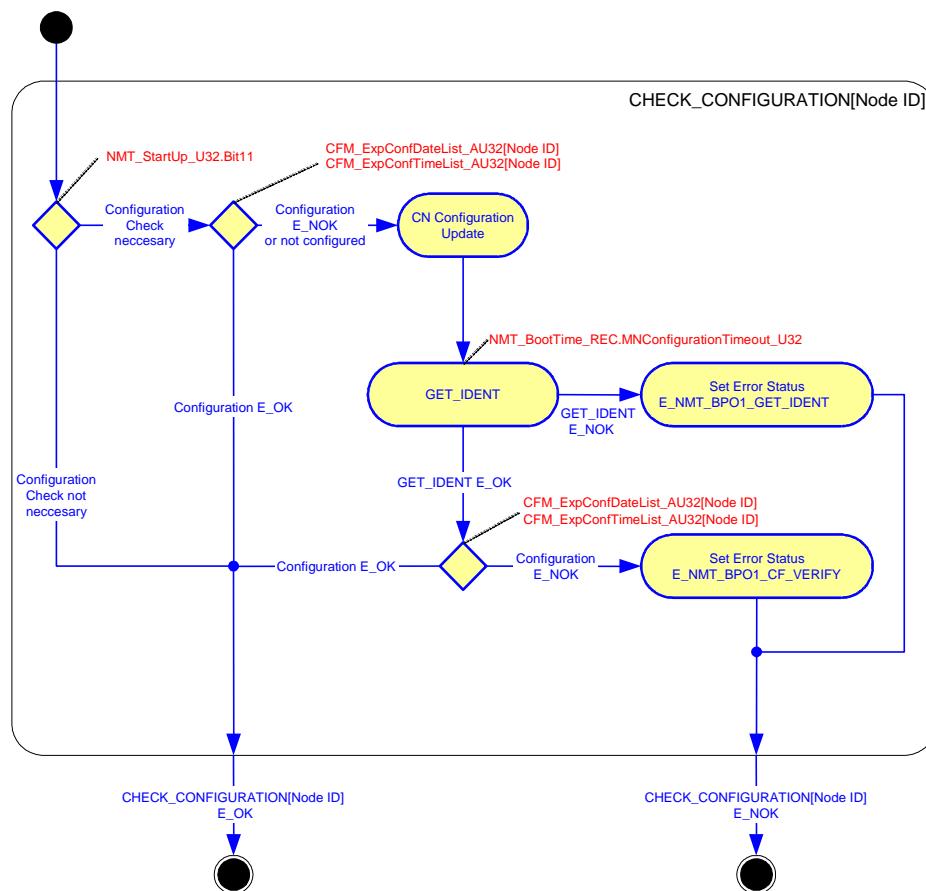


Fig. 95. Sub-state CHECK\_CONFIGURATION[Node ID]

### 7.4.2.2.1.3.1 GET\_IDENT

The purpose of the sub-state GET\_IDENT is to request the IdentResponse from a CN. GET\_IDENT [Node ID] returns E\_OK, if the CN answers within a timeout interval, otherwise GET\_IDENT [Node ID] returns E\_NOK. The timeout interval depends on the actual boot-up state.

GET\_IDENT[Node ID] proceeds as follows:

- Request IdentResponse from the CN.
- The IdentRequest will be repeated until the CN responds with its Ident Response or a timeout occurs (timeout value passed by the caller of this sub state).
- If the CN answers with its IdentResponse, GET\_IDENT[Node ID] returns E\_OK.
- If the CN does not respond within the configured timeout interval, GET\_IDENT[Node ID] returns E\_NOK.

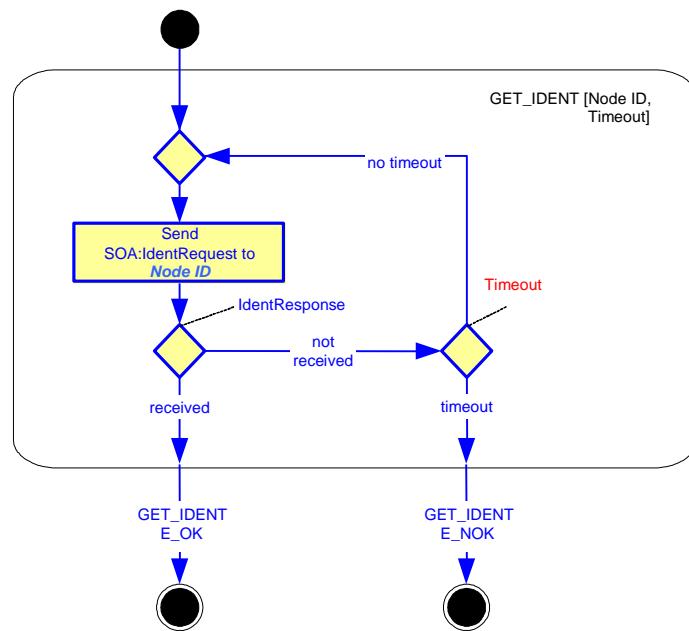


Fig. 96. Sub-state GET\_IDENT[Node ID]

### 7.4.2.2.2 BOOT\_STEP2

The purpose of BOOT\_STEP2 is to send the NMT command NMTEnableReadyToOperate to a CN and to check the CN's NMT state change to NMT\_CS\_READY\_TO\_OPERATE. Furthermore the initialisation of the error signaling shall be completed in BOOT\_STEP2.

BOOT\_STEP2 [Node ID] returns E\_OK, if the initialisation of the error signaling is completed and the CN NMT state is NMT\_CS\_READY\_TO\_OPERATE.

BOOT\_STEP2 [Node ID] returns E\_NOK, if the initialisation of the error signaling is not completed or the CN NMT state did not change from NMT\_CS\_PRE\_OPERATIONAL\_2 to NMT\_CS\_READY\_TO\_OPERATE state within a timeout interval (see MNTtimeoutPreOp2\_U32).

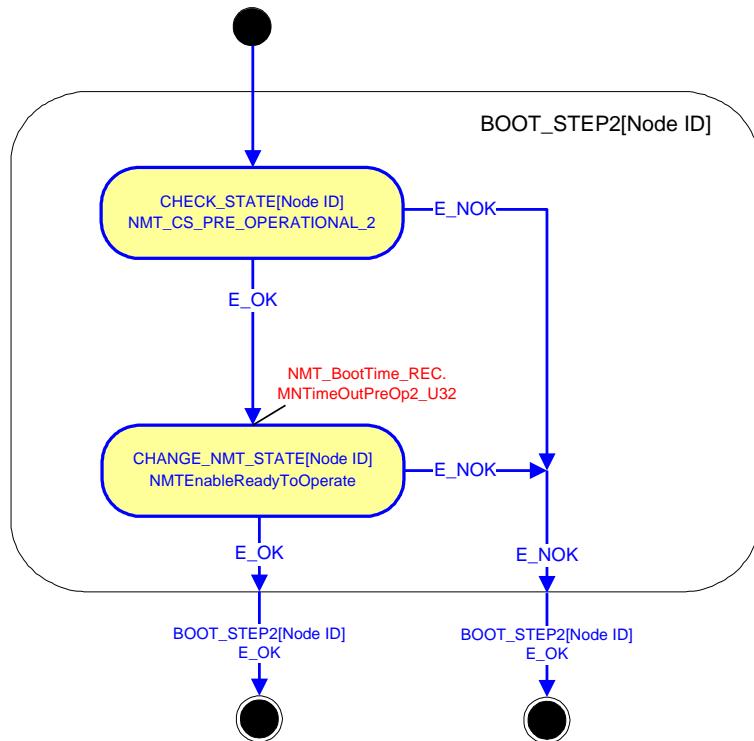


Fig. 97. Sub-state **BOOT\_STEP2[Node ID]**

**BOOT\_STEP2[Node ID]** proceeds as follows:

- Verify that the CN is in state **NMT\_CS\_PRE\_OPERATIONAL\_2** by **CHECK\_STATE**.
- Start the process **CHANGE\_NMT\_STATE** to send the NMT command **NMTEnableReadyToOperate** to the CN and to check the NMT state change of the CN.
- Check for the end of the initialisation of the error signaling.

### 7.4.2.2.3 CHECK\_COMMUNICATION

The purpose of CHECK\_COMMUNICATION[Node ID] is to check the communication with a CN after start of the isochronous POWERLINK cycle in the NMT state NMT\_MS\_READY\_TO\_OPERATE. CHECK\_COMMUNICATION[Node ID] will return E\_OK, if the PRes frame checking of isochronous CNs returns no error. Otherwise CHECK\_COMMUNICATION [Node ID] returns E\_NOK.

CHECK\_COMMUNICATION[Node ID] proceeds as follows:

- Check the received CN PRes:
  - No loss of frame occurs
  - The payload length is less or equal than the length configured in object
    - NMT\_PResPayloadLimitList\_AU16 [Node ID].
  - The frame receive time is less or equal than the time configured in object
    - NMT\_MNCNPResTimeout\_AU32[Node ID].
- CHECK\_COMMUNICATION[Node ID] returns E\_NOK, if the communication with the CN fails or the expected parameter values do not fit. Otherwise CHECK\_COMMUNICATION[Node ID] returns E\_OK.

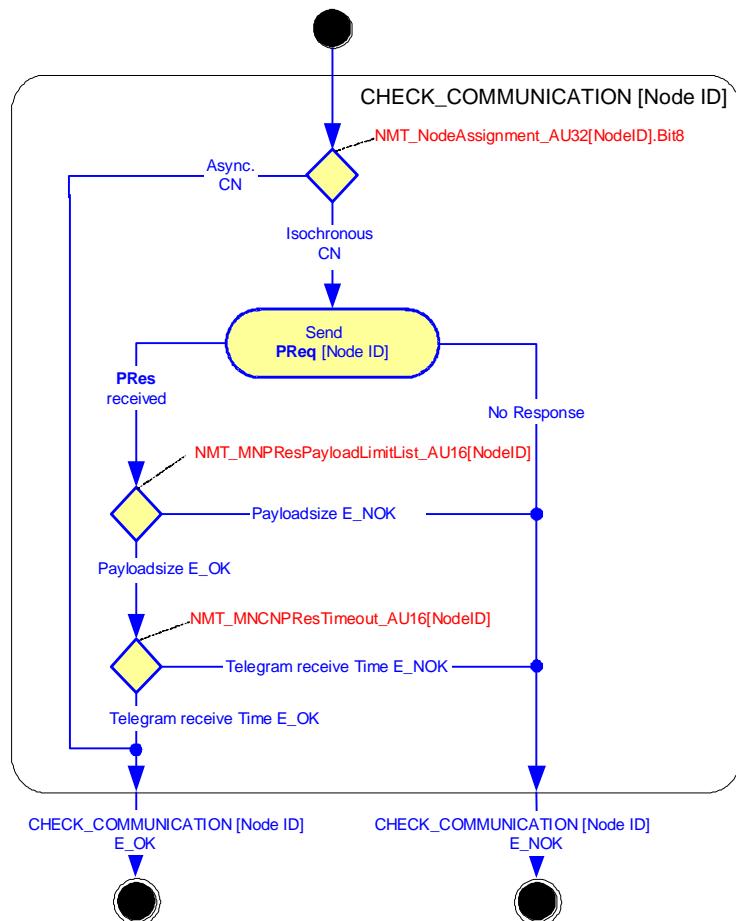


Fig. 98. Sub-state CHECK\_COMMUNICATION[Node ID]

### 7.4.2.2.4 START\_CN

In the sub-state START\_CN[Node ID] the MN sends NMTStartNode to an individual CN. START\_CN[Node ID] returns E\_OK, if the CN changes its state to NMT\_CS\_OPERATIONAL within a timeout interval, otherwise E\_NOK will be returned.

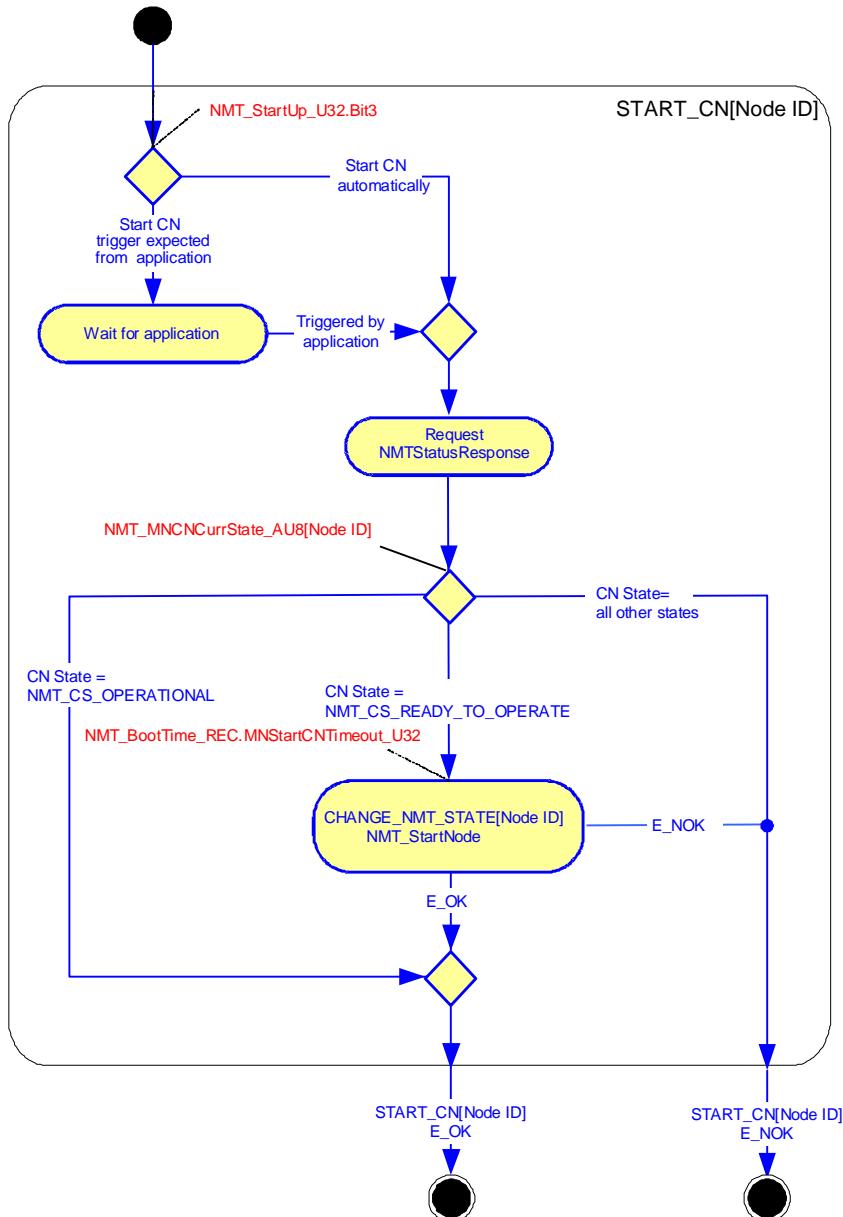


Fig. 99. Sub-state START\_CN[Node ID]

START\_CN[Node ID] proceeds as follows:

- START\_CN[Node ID] starts immediately or after the application trigger, depending on object
  - NMT\_StartUp\_U32.Bit3.
- The MN checks the current state of the CN. If the CN is in the state NMT\_CS\_OPERATIONAL, START\_CN[Node ID] returns E\_OK.
- If the CN is in the state NMT\_CS\_READY\_TO\_OPERATE the boot-up procedure enters the sub-state CHANGE\_NMT\_STATE[Node ID] to send NMTStartNode to the CN and control its state change.
- If CHANGE\_NMT\_STATE[Node ID] returns E\_OK, START\_CN[Node ID] returns E\_OK.

- If the CN is in a wrong state (neither NMT\_CS\_OPERATIONAL nor NMT\_CS\_READY\_TO\_OPERATE) or if CHANGE\_NMT\_STATE[Node ID] returns E\_NOK, START\_CN[Node ID] returns E\_NOK, too.

### 7.4.2.2.5 START\_ALL

In the sub-state START\_ALL, NMTStartNode is sent to all CNs as broadcast. START\_ALL returns E\_OK, if all CNs change their state to NMT\_CS\_OPERATIONAL within a timeout interval, otherwise START\_ALL returns E\_NOK.

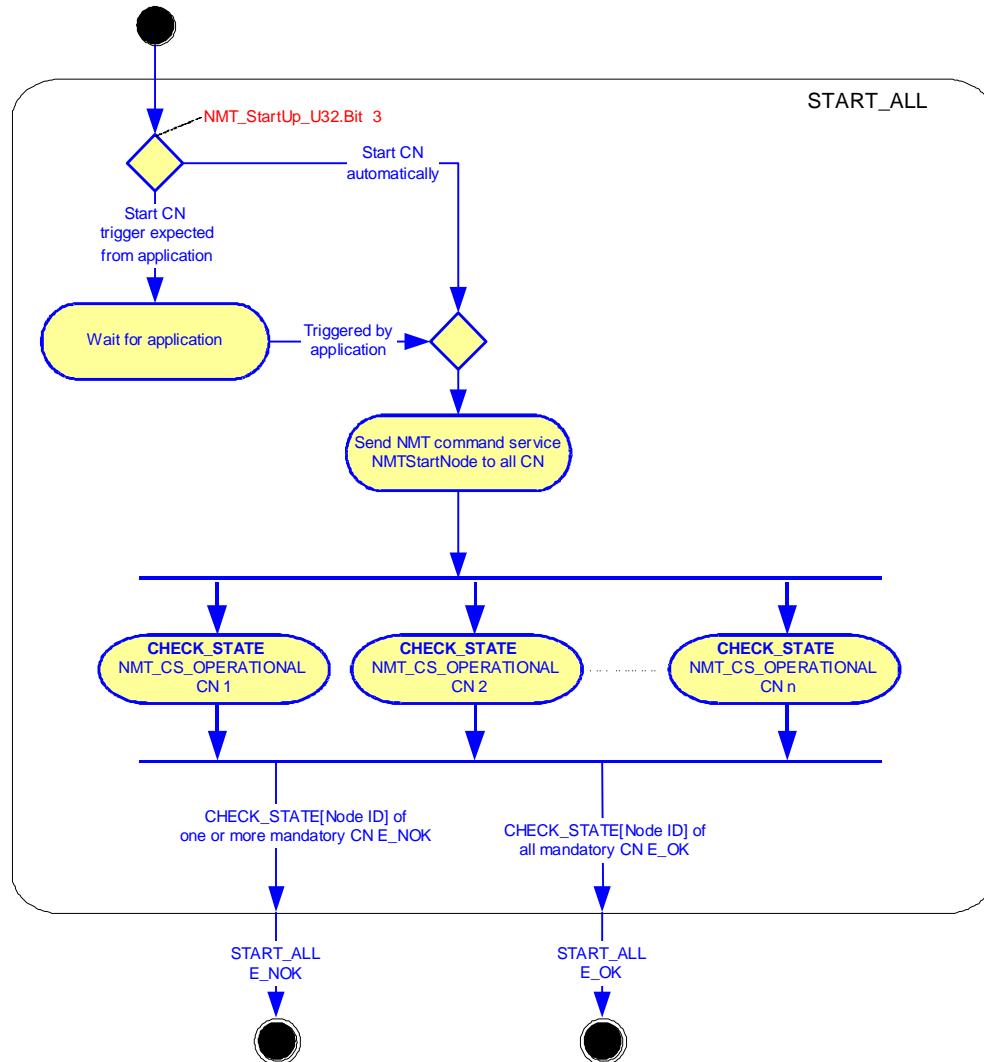


Fig. 100. Sub-state START\_ALL

START\_ALL proceeds as follows:

- START\_ALL proceeds immediately or after the application trigger, depending on the following object:
  - NMT\_StartUp\_U32.Bit3.
  - After transmission of the broadcast NMT command NMTStartNode, the MN starts the sub-state CHECK\_STATE for all identified CNs. If all CNs are in the state NMT\_CS\_OPERATIONAL, START\_ALL returns E\_OK. If one or more mandatory CNs caused an error in CHECK\_STATE, START\_ALL returns E\_NOK.

### 7.4.2.2.6 CHECK\_STATE

The purpose of CHECK\_STATE is to check the current NMT state of a CN.

CHECK\_STATE returns E\_OK, if the current CN state is already equal to the expected state. It returns E\_NOK, if the CN did not switch to the target state within a timeout interval.

If the CN is in an unexpected state, CHECK\_STATE returns E\_NOK.

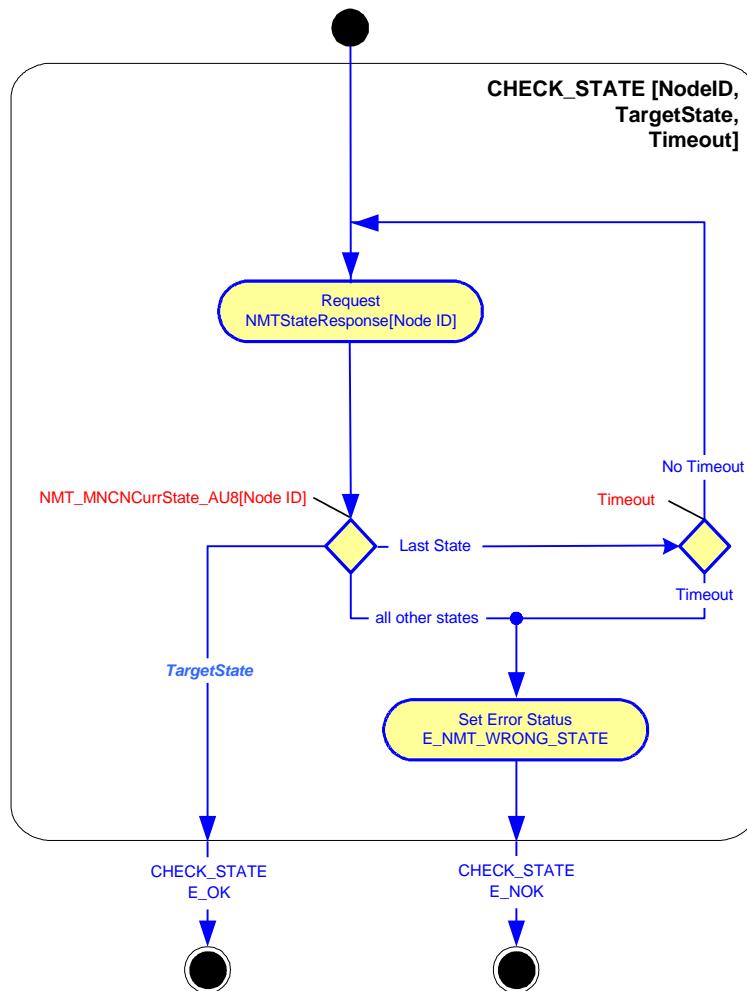


Fig. 101. Sub-state CHECK\_STATE

CHECK\_STATE proceeds as follows:

- If the CN state equals to the target state, CHECK\_STATE returns E\_OK.
- If the CN state corresponds to the last state, the CN state is further checked within a time interval until the state changes to the target state or a timeout occurs.  
For a state change to NMT\_CS\_READY\_TO\_OPERATE the timeout is configured via object NMT\_BootTime\_REC.MNTTimeoutPreOp2\_U32.  
In all other cases the timeout is defined by C\_NMT\_STATE\_TOLERANCE.  
In case of a timeout CHECK\_STATE returns E\_NOK.
- If the CN is in an unexpected state, CHECK\_STATE returns E\_NOK.

### 7.4.2.2.7 CHANGE\_NMT\_STATE

CHANGE\_NMT\_COMMAND sends a NMT command to a particular CN. The resulting NMT state change is checked. Before checking the status, the MN waits for C\_NMT\_STATE\_TOLERANCE cycles to allow the CN's NMT state change.

CHANGE\_NMT\_STATE returns E\_OK, if the CN state is equal to the target state. Otherwise E\_NOK will be returned.

If the CN state is not in an expected state, CHANGE\_NMT\_STATE returns with status E\_NOK.

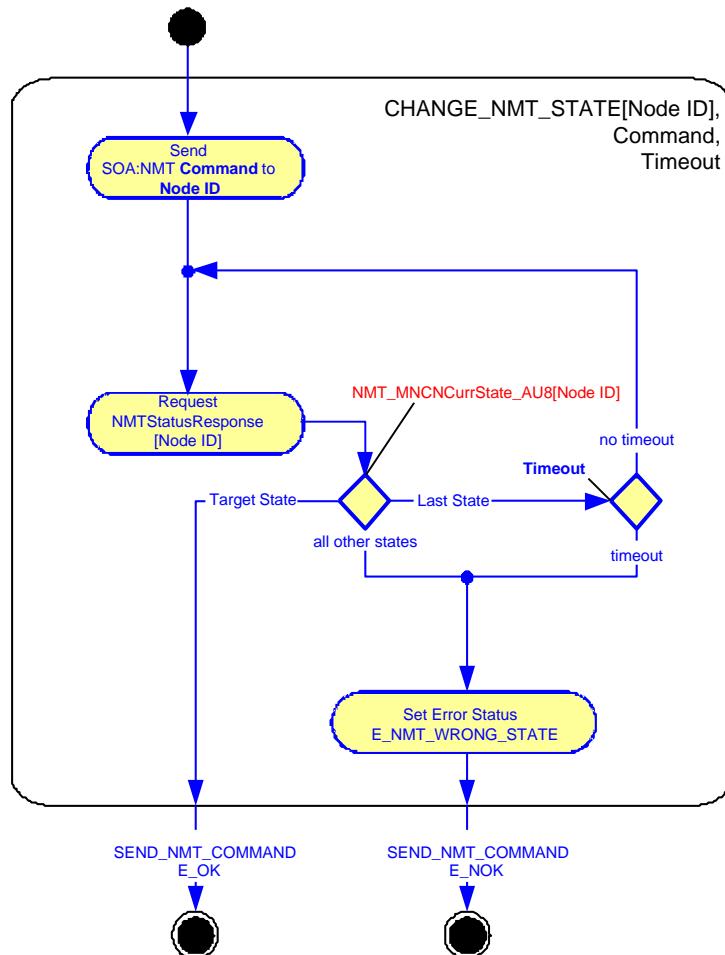


Fig. 102. Sub-state CHANGE\_NMT\_STATE

### 7.4.2.2.8 OPERATIONAL

In the OPERATIONAL state, the MN supervises all CNs that are in NMT\_CS\_OPERATIONAL state. If a mandatory CN generates an error such as response timeouts or wrong NMT states, the ERROR\_TREATMENT substate shall be entered.

Errors of optional nodes in state OPERATIONAL are reported to the application. The boot-up of these optional CNs shall be restarted after sending the NMT command NMTRestartNode.

### 7.4.2.2.9 ERROR\_TREATMENT

In the sub-state ERROR\_TREATMENT the MN will, depending on the NMT\_StartUp\_U32.Bit4 and NMT\_StartUp\_U32.Bit6, either send a NMTRestartNode or a NMTRestartNode to all CNs or it shall handle errors on CN in an application specific manner.

Errors of optional CNs do not have any influence on the MN or the other CNs.

The steps of the ERROR\_TREATMENT process are as follows:

- Depending on the following bits:
  - NMT\_StartUp\_U32.Bit6
  - NMT\_StartUp\_U32.Bit4

the errors are treated different. For optional nodes these bits are always ignored as only the individual CN error treatment is allowed.

- If NMT\_StartUp\_U32.Bit6 is true, the NMT command NMTStopNode shall be transmitted to all CNs
- If NMT\_StartUp\_U32.Bit4 is true, the NMT command NMTRestNode shall be transmitted to all CNs. If NMT\_StartUp\_U32.Bit4 is false, errors are treated individually by the application.

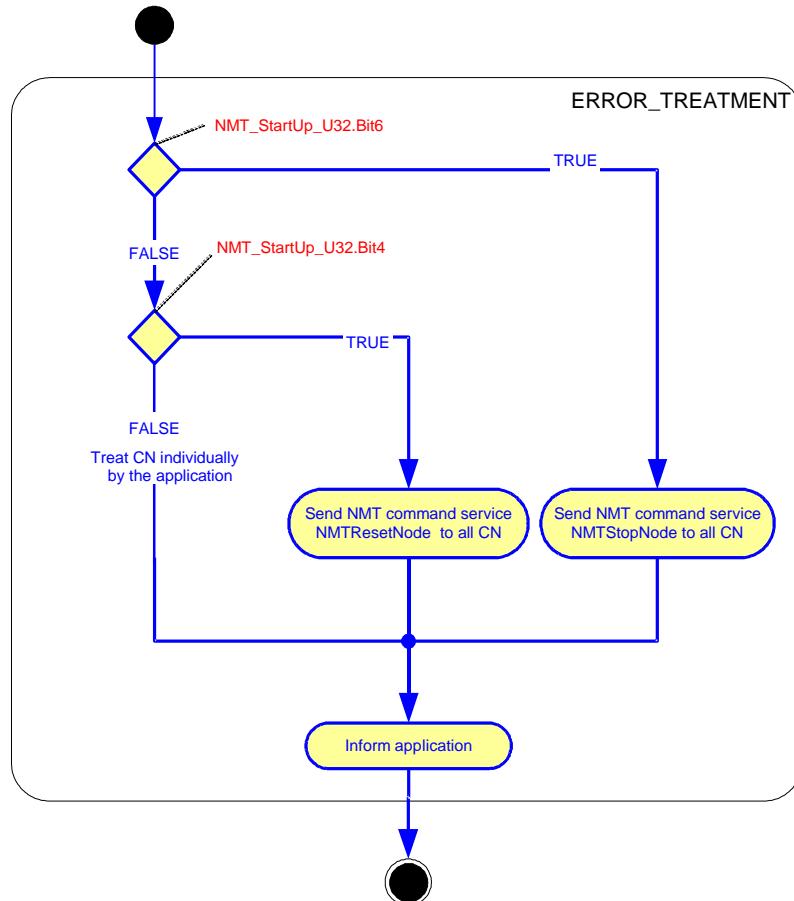


Fig. 103. Sub-state ERROR\_TREATMENT

### 7.4.3 Boot-up Errors

#### 7.4.3.1 Bus activity

- **Error Source**

The device is configured as an MN and detects another MN (SoC, IdentRequest, ...).

- **Handling**

The MN shall halt the boot procedure. The error is logged in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code | Timestamp | Additional Information |
|----------------|------------------|------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BA1  | XXXX      |                        |

### 7.4.3.2 BOOT\_STEP1 failed

- **Error Source**

Boot process in the NMT\_MS\_PRE\_OPERATIONAL\_1 state failed. One or more mandatory CNs failed.

- **Handling**

The MN shall stop the boot-up procedure. The error is logged in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code | Timestamp | Additional Information |
|----------------|------------------|------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1 | XXXX      |                        |

### 7.4.3.3 BOOT\_STEP2 failed

- **Error Source**

Boot process in the NMT\_MS\_PRE\_OPERATIONAL\_2 state failed. One or more of the mandatory configured CNs failed.

- **Handling**

The MN shall stop the boot-up procedure. The error is logged in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code | Timestamp | Additional Information |
|----------------|------------------|------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO2 | XXXX      |                        |

### 7.4.3.4 Boot-up in NMT\_MS\_READY\_TO\_OPERATE failed

- **Error Source**

The process CHECK\_COMMUNICATION in the NMT\_MS\_READY\_TO\_OPERATE state failed because of the reception of at least one incorrect or missing PRes.

- **Handling**

The MN shall stop the boot-up procedure and log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code | Timestamp | Additional Information |
|----------------|------------------|------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BRO  | XXXX      |                        |

### 7.4.3.5 Get Ident failed

- **Error Source**

No response on an Ident Request was received.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_GET_IDENT | XXXX      |                        |

### 7.4.3.6 Device Type Invalid

- **Error Source**

Actual device type of the CN (value of object NMT\_DeviceType\_U32 reported via IdentResponse) did not match the expected device type configured on the MN via object NMT\_MNDeviceTypeIdList\_AU32.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code             | Timestamp | Additional Information |
|----------------|------------------|------------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_DEVICE_TYPE | XXXX      |                        |

### 7.4.3.7 Vendor ID invalid

- **Error Source**

Vendor ID reported via IdentResponse of the CN did not match the expected vendor ID configured on the MN via object NMT\_MNVendorIdList\_AU32.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_VENDOR_ID | XXXX      |                        |

### 7.4.3.8 Configuration failed

- **Error Source**

Configuration update failed.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_CF_VERIFY | XXXX      |                        |

### 7.4.3.9 Product Code invalid

- **Error Source**

Product Code reported via IdentResponse of the CN did not match the expected product code configured on the MN via object NMT\_MNProductCodeList\_AU32.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code              | Timestamp | Additional Information |
|----------------|------------------|-------------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_PRODUCT_CODE | XXXX      |                        |

### 7.4.3.10 Revision number invalid

- **Error Source**

Revision number reported via IdentResponse of the CN did not match the expected revision number configured on the MN via object NMT\_MNRevisionNoList\_AU32.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code             | Timestamp | Additional Information |
|----------------|------------------|------------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_REVISION_NO | XXXX      |                        |

### 7.4.3.11 Serial number invalid

- **Error Source**

Serial number reported via IdentResponse of the CN did not match the expected serial number configured on the MN via object NMT\_MNSerialNoList\_AU32.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_SERIAL_NO | XXXX      |                        |

### 7.4.3.12 NMT state invalid

- **Error Source**

Reported CN NMT state does not match the expected NMT state.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code        | Timestamp | Additional Information |
|----------------|------------------|-------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_WRONG_STATE | XXXX      |                        |

### 7.4.3.13 Invalid Software

- **Error Source**

Software version information for a certain CN not available on the MN.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code            | Timestamp | Additional Information |
|----------------|------------------|-----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_SW_INVALID | XXXX      |                        |

### 7.4.3.14 Invalid NMT state for SW update

- **Error Source**

Software update failed due to wrong NMT state of the target CN.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code          | Timestamp | Additional Information |
|----------------|------------------|---------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_SW_STATE | XXXX      |                        |

### 7.4.3.15 SW update not allowed

- **Error Source**

Software update of a CN not allowed.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_SW_REJECT | XXXX      |                        |

### 7.4.3.16 SW update failed

- **Error Source**

An error occurred during the application SW update process.

- **Handling**

Log the error in the error history.

- **Registration**

History Entry Object ERR\_History\_ADOM:

| Mode           | Profile          | Error Code           | Timestamp | Additional Information |
|----------------|------------------|----------------------|-----------|------------------------|
| 3 <sub>h</sub> | 002 <sub>h</sub> | E_NMT_BPO1_SW_UPDATE | XXXX      |                        |

## 7.4.4 Minimal Boot-up MN

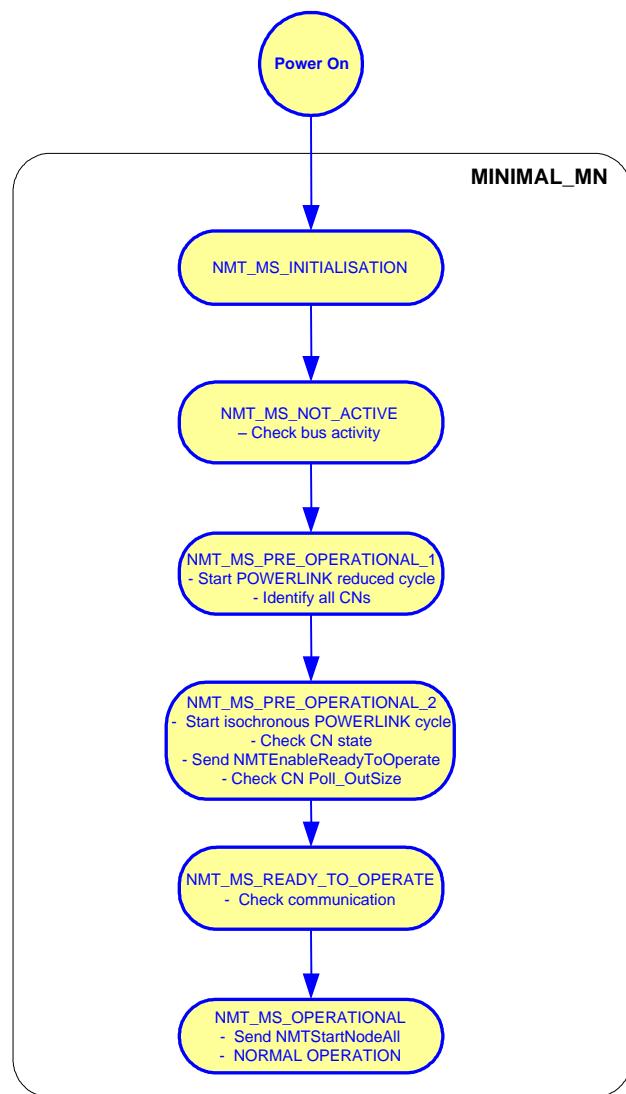


Fig. 104. Minimal NMT boot-up process

Fig. 104 depicts a minimal MN boot-up, which allows to implement a simple NMT state machine on low-cost MNs. Only small applications should use this type of boot-up as there is no error treatment or configuration checking. The minimal MN boot-up guarantees only that all devices reach the operational state and that no device violates the POWERLINK cycle.

## 7.4.5 Example Boot-up Sequence

In this example a typical boot-up with a single CN and without boot-up errors is depicted. The example also shows a configuration update of the CN in BOOT\_STEP1.

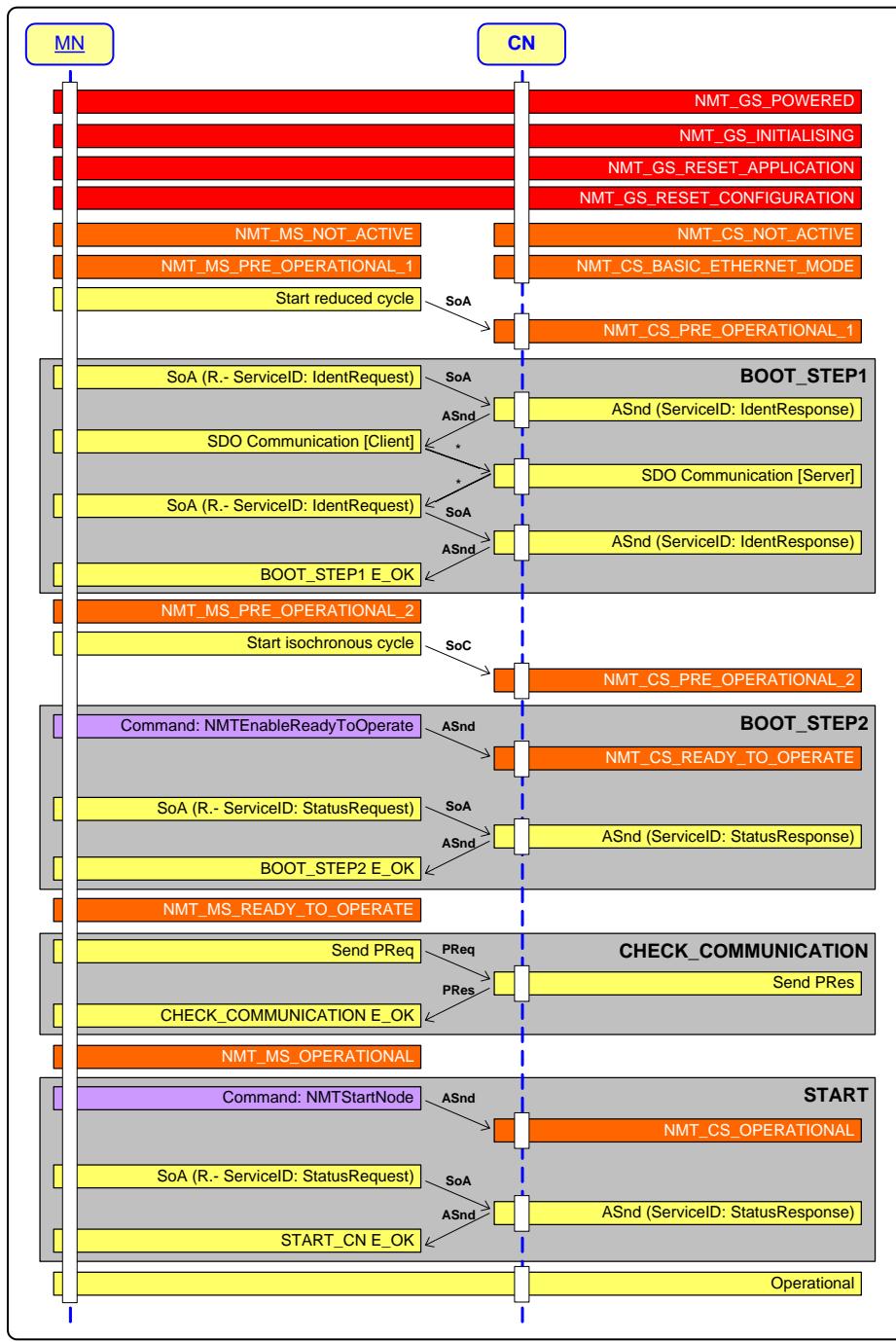


Fig. 105. Boot procedure example for a single CN

## 7.4.6 Application Notes

- It is permitted to boot-up one device after another (sequential boot-up) or all in parallel. If the sequential boot-up procedure is implemented, it must be ensured that a particular boot step is still performed for all devices even if an error of a mandatory node occurred within that boot step. The reaction of the boot-up process upon an error within a mandatory node shall be delayed until the boot step was finished for all the other devices, too.

- The defined processes shall give an overview to the boot-up, they do not define a specific API. The main purpose is to have common rules for the MN boot-up procedure and to give CNs the knowledge, what they have to expect on boot-up.
- The same procedure, except for the error handling, will be used if an optional CN has to be booted while the rest of the system is already running, e.g. after a reset or error. In that case the NMT command NMTStartNode may always be sent individually.

## 8 Diagnostics

### 8.1 Diagnostic Object Dictionary Entries

#### 8.1.1 Object 1101<sub>h</sub>: DIA\_NMTTelegrCount\_REC

The object's sub-indices count cycles, synchronous and asynchronous frames, SDO telegrams and StatusRequests since NMT\_GS\_RESET\_COMMUNICATION. The counters shall be reset in NMT\_GS\_RESET\_COMMUNICATION.

The application may provide additional means to reset the counters.

|           |                         |             |        |
|-----------|-------------------------|-------------|--------|
| Index     | 1101 <sub>h</sub>       | Object Code | RECORD |
| Name      | DIA_NMTTelegrCount_REC  |             |        |
| Data Type | DIA_NMTTelegrCount_TYPE | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 8               | Access      | const |
| Default Value | 8               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: IsochrCyc\_U32**

|             |                 |          |    |
|-------------|-----------------|----------|----|
| Sub-Index   | 01 <sub>h</sub> |          |    |
| Name        | IsochrCyc_U32   |          |    |
| Data Type   | UNSIGNED32      | Category | M  |
| Value Range | UNSIGNED32      | Access   | ro |

This sub-index holds the number of transmitted (MN) or received (CN) SoC frames.

- **Sub-Index 02<sub>h</sub>: IsochrRx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub> |             |    |
| Name          | IsochrRx_U32    |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of received PReq and PRes frames.

- **Sub-Index 03<sub>h</sub>: IsochrTx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 03 <sub>h</sub> |             |    |
| Name          | IsochrTx_U32    |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of transmitted PReq and PRes frames.

- **Sub-Index 04<sub>h</sub>: AsyncRx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 04 <sub>h</sub> |             |    |
| Name          | AsyncRx_U32     |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of received asynchronous frames (POWERLINK ASnd, IP frames etc., but not SoA).

- **Sub-Index 05<sub>h</sub>: AsyncTx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 05 <sub>h</sub> |             |    |
| Name          | AsyncTx_U32     |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of transmitted asynchronous frames (POWERLINK ASnd, IP frames etc., but not SoA).

- **Sub-Index 06<sub>h</sub>: SdoRx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 06 <sub>h</sub> |             |    |
| Name          | SdoRx_U32       |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of received SDO telegrams via UDP/IP or POWERLINK ASnd.

- **Sub-Index 07<sub>h</sub>: SdoTx\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 07 <sub>h</sub> |             |    |
| Name          | SdoTx_U32       |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of transmitted SDO telegrams via UDP/IP or POWERLINK ASnd.

- **Sub-Index 08<sub>h</sub>: Status\_U32**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 08 <sub>h</sub> |             |    |
| Name          | Status_U32      |             |    |
| Data Type     | UNSIGNED32      | Category    | O  |
| Value Range   | UNSIGNED32      | Access      | ro |
| Default Value | 0               | PDO Mapping | No |

This sub-index holds the number of received StatusRequest SoA telegrams.

## 8.1.2 Object 1102<sub>h</sub>: DIA\_ERRStatistics\_REC

The object's sub-indices count events which occurred in the Error Signaling module. When reaching their maximum value the counters shall continue with 0.

|           |                        |             |        |
|-----------|------------------------|-------------|--------|
| Index     | 1102 <sub>h</sub>      | Object Code | RECORD |
| Name      | DIA_ERRStatistics_REC  |             |        |
| Data Type | DIA_ERRStatistics_TYPE | Category    | O      |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 7               | Access      | const |
| Default Value | 7               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: HistoryEntryWrite\_U32**

|               |                       |             |    |
|---------------|-----------------------|-------------|----|
| Sub-Index     | 01 <sub>h</sub>       |             |    |
| Name          | HistoryEntryWrite_U32 |             |    |
| Data Type     | UNSIGNED32            | Category    | O  |
| Value Range   | UNSIGNED32            | Access      | ro |
| Default Value | 0                     | PDO Mapping | No |

This sub-index holds the number of entries written to the error history (see 6.5.1).

- **Sub-Index 02<sub>h</sub>: EmergencyQueueWrite\_U32**

|               |                         |             |    |
|---------------|-------------------------|-------------|----|
| Sub-Index     | 02 <sub>h</sub>         |             |    |
| Name          | EmergencyQueueWrite_U32 |             |    |
| Data Type     | UNSIGNED32              | Category    | O  |
| Value Range   | UNSIGNED32              | Access      | ro |
| Default Value | 0                       | PDO Mapping | No |

This sub-index holds the number of successful write actions to the emergency queue. Write actions which can not be performed because the queue is full shall not be counted here.

- **Sub-Index 03<sub>h</sub>: EmergencyQueueOverflow\_U32**

|               |                            |             |    |
|---------------|----------------------------|-------------|----|
| Sub-Index     | 03 <sub>h</sub>            |             |    |
| Name          | EmergencyQueueOverflow_U32 |             |    |
| Data Type     | UNSIGNED32                 | Category    | O  |
| Value Range   | UNSIGNED32                 | Access      | ro |
| Default Value | 0                          | PDO Mapping | No |

This sub-index holds the number of write actions to the emergency queue which could not be done because the queue was full.

- **Sub-Index 04<sub>h</sub>: StatusEntryChanged\_U32**

|               |                        |             |    |
|---------------|------------------------|-------------|----|
| Sub-Index     | 04 <sub>h</sub>        |             |    |
| Name          | StatusEntryChanged_U32 |             |    |
| Data Type     | UNSIGNED32             | Category    | O  |
| Value Range   | UNSIGNED32             | Access      | ro |
| Default Value | 0                      | PDO Mapping | No |

This sub-index holds the number of changes in the StatusEntries.

- **Sub-Index 05<sub>h</sub>: StaticErrorBitFieldChanged\_U32**

|               |                                |             |    |
|---------------|--------------------------------|-------------|----|
| Sub-Index     | 05 <sub>h</sub>                |             |    |
| Name          | StaticErrorBitFieldChanged_U32 |             |    |
| Data Type     | UNSIGNED32                     | Category    | O  |
| Value Range   | UNSIGNED32                     | Access      | ro |
| Default Value | 0                              | PDO Mapping | No |

This sub-index holds the number of changes in the Static Error Bit Field.

- **Sub-Index 06<sub>h</sub>: ExceptionResetEdgePos\_U32**

|               |                           |             |    |
|---------------|---------------------------|-------------|----|
| Sub-Index     | 06 <sub>h</sub>           |             |    |
| Name          | ExceptionResetEdgePos_U32 |             |    |
| Data Type     | UNSIGNED32                | Category    | O  |
| Value Range   | UNSIGNED32                | Access      | ro |
| Default Value | 0                         | PDO Mapping | No |

This sub-index holds the number of detected 0 to 1 transitions of the bit ER.

- **Sub-Index 07<sub>h</sub>: ExceptionNewEdge\_U32**

|               |                      |             |    |
|---------------|----------------------|-------------|----|
| Sub-Index     | 07 <sub>h</sub>      |             |    |
| Name          | ExceptionNewEdge_U32 |             |    |
| Data Type     | UNSIGNED32           | Category    | O  |
| Value Range   | UNSIGNED32           | Access      | ro |
| Default Value | 0                    | PDO Mapping | No |

This sub-index holds the number of all generated transitions on the bit EN.

## 8.1.3 Diagnostics Object Types

### 8.1.3.1 Object 0437<sub>h</sub>: DIA\_NMTTelegrCount\_TYPE

| Index                               | 0437 <sub>h</sub> | Object Code       | DEFSTRUCT  |
|-------------------------------------|-------------------|-------------------|------------|
| <b>Name</b> DIA_NMTTelegrCount_TYPE |                   |                   |            |
| Sub-Index                           | Component Name    | Value             | Data Type  |
| 00 <sub>h</sub>                     | NumberOfEntries   | 06 <sub>h</sub>   |            |
| 01 <sub>h</sub>                     | IsochrCyc_U32     | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub>                     | IsochrRx_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub>                     | IsochrTx_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub>                     | AsyncRx_U32       | 0007 <sub>h</sub> | UNSIGNED32 |
| 05 <sub>h</sub>                     | AsyncTx_U32       | 0007 <sub>h</sub> | UNSIGNED32 |
| 06 <sub>h</sub>                     | SdoRx_U32         | 0007 <sub>h</sub> | UNSIGNED32 |
| 07 <sub>h</sub>                     | SdoTx_U32         | 0007 <sub>h</sub> | UNSIGNED32 |
| 08 <sub>h</sub>                     | Status_U32        | 0007 <sub>h</sub> | UNSIGNED32 |

### 8.1.3.2 Object 0438<sub>h</sub>: DIA\_ERRStatistics\_TYPE

| Index           | 0438 <sub>h</sub>              | Object Code       | DEFSTRUCT  |
|-----------------|--------------------------------|-------------------|------------|
| Name            | DIA_ERRStatistics_TYPE         |                   |            |
| Sub-Index       | Component Name                 | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries                | 07 <sub>h</sub>   |            |
| 01 <sub>h</sub> | HistoryEntryWrite_U32          | 0007 <sub>h</sub> | UNSIGNED32 |
| 02 <sub>h</sub> | EmergencyQueueWrite_U32        | 0007 <sub>h</sub> | UNSIGNED32 |
| 03 <sub>h</sub> | EmergencyQueueOverflow_U32     | 0007 <sub>h</sub> | UNSIGNED32 |
| 04 <sub>h</sub> | StatusEntryChanged_U32         | 0007 <sub>h</sub> | UNSIGNED32 |
| 05 <sub>h</sub> | StaticErrorBitFieldChanged_U32 | 0007 <sub>h</sub> | UNSIGNED32 |
| 06 <sub>h</sub> | ExceptionResetEdgePos_U32      | 0007 <sub>h</sub> | UNSIGNED32 |
| 07 <sub>h</sub> | ExceptionNewEdge_U32           | 0007 <sub>h</sub> | UNSIGNED32 |

## 9 Routing

A POWERLINK Router is a coupling element in a network that allows IP communication between a POWERLINK segment and any other datalink layer protocol carrying IP e.g. legacy Ethernet, POWERLINK etc.

### 9.1 Routing Type 1

A POWERLINK Router Type 1 is a coupling element in a network that allows IP communication between a POWERLINK network and any other datalink layer protocol carrying IP e.g. Legacy Ethernet, POWERLINK etc. It enables the communication between two networks using IP and ICMP. Other network layer protocols besides IP cannot be coupled via the Routing Type 1. MN and CNs may have router functionality implemented. A POWERLINK Router runs an application that forwards IP and ICMP datagrams from the POWERLINK network to an external network and vice versa. Fig. 106 illustrates the black box model of the POWERLINK Router.

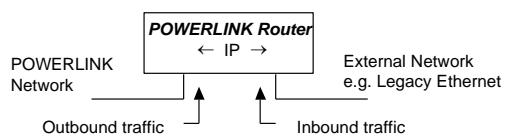


Fig. 106. POWERLINK router, black box model

Traffic from the external network to the POWERLINK network is called inbound traffic. The traffic from the POWERLINK network to the external network e.g. Legacy Ethernet is called outbound traffic.

Routing Type 1 is optional. Support shall be indicated by D\_RT1\_RT1Support\_BOOL.

#### 9.1.1 Core Tasks of a POWERLINK Router

This section describes some relevant application scenarios in more detail. Possible application scenarios for the use of a POWERLINK Router are listed below. Only scenarios for communication paths that involve a POWERLINK Router are listed.

- Diagnostics, Remote Maintenance, Monitoring
- Alarm Messages
- Software Download
- Configuration / Engineering
- Secure Access
- SDO communication

Each data transmission that originates from the POWERLINK network and terminates out-side of it (including inside the router), falls within the responsibility of the POWERLINK Router. The same applies for traffic that originates from the external network and terminates inside the POWERLINK network.

There are several use cases that require the above mentioned traffic pattern. While they can all be subsumed under the view that they require the functionality of the POWERLINK Router, they will be outlined below because each use case has its own peculiarities. In any case, the coupling by means of a POWERLINK Router is only possible on layer 3, because the POWERLINK Router only handles data link frames that contain IP and ICMP packets. Fig. 107 illustrates these use cases.

1. Access from the Factory Floor network (e. g., for diagnostics)
2. Access from the Company Network (e. g., for inventory purposes)  
In this scenario, additional security effort may be necessary, which may require further processing in the data path POWERLINK Router <-> company network.
3. Remote Access (e. g., for remote maintenance)  
Strict security aspects should be taken into account. These are outside the scope of this document.
4. POWERLINK inter-segment communication (e. g., for exchange of manufacturing service data)

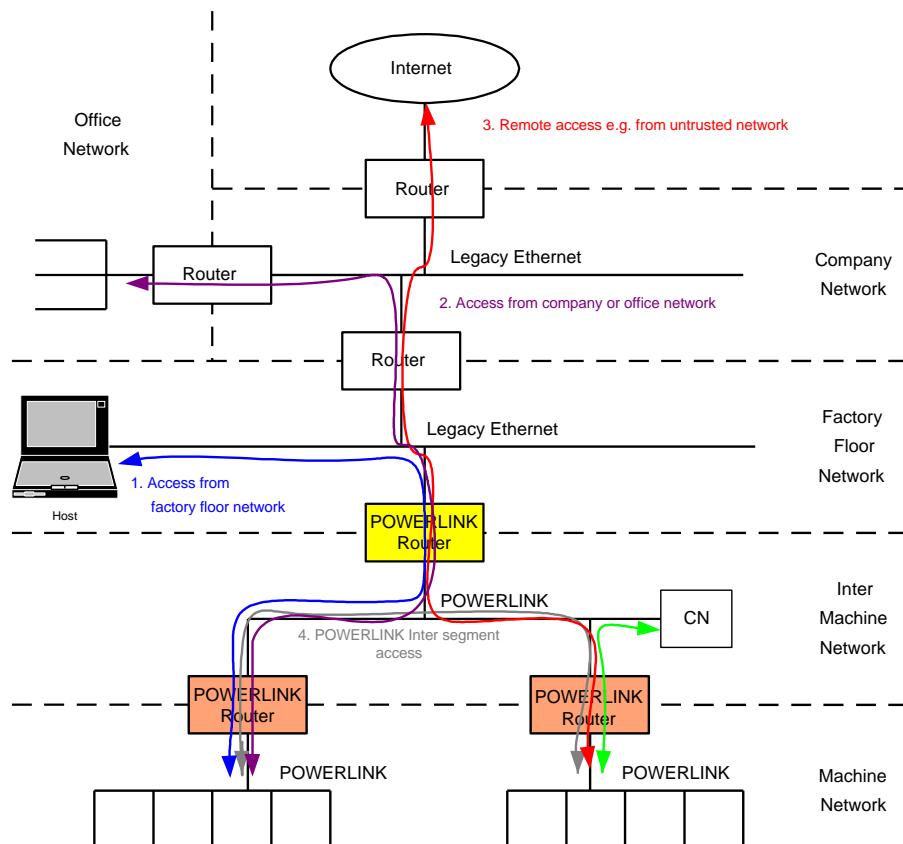


Fig. 107. Possible communication relations via a POWERLINK router

A POWERLINK Router shall provide at least one interface of type POWERLINK. In Fig. 107 that is the POWERLINK Router (yellow) between Factory Floor and Inter Machine Network and the POWERLINK Routers (orange) between Inter Machine and Machine Network. The other routers do not contain a POWERLINK interface and are therefore outside the scope of this specification.

## 9.1.2 Reference Model

The reference model of the POWERLINK Router is shown below.

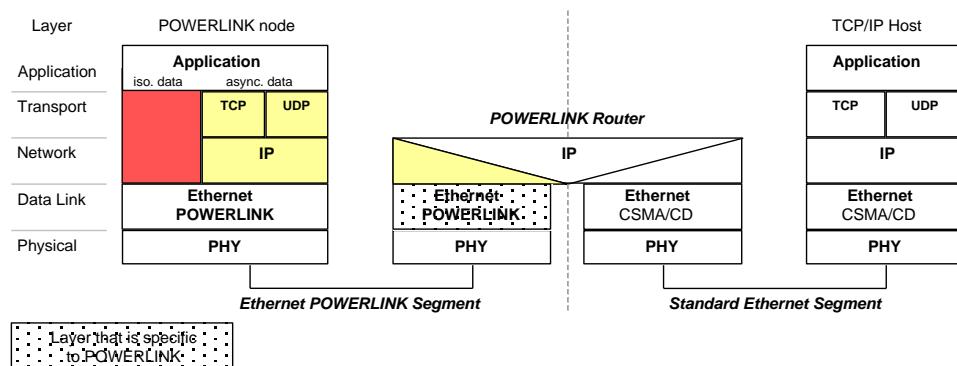


Fig. 108. POWERLINK router reference model

## 9.1.3 Data Link Layer

A POWERLINK Router shall at least provide two interfaces. One interface to the POWERLINK and a second interface to the external network. IP operates on top of the data link layer – e.g. POWERLINK, Legacy Ethernet.

### 9.1.3.1 DLL POWERLINK Interface

The Interface of the POWERLINK Router to the POWERLINK network shall behave exactly like a POWERLINK CN (see 4.2.2.2).

### 9.1.3.2 DLL interface to the external network

Depending on the application requirements, the interface to the external network can be chosen. Any datalink layer protocol that can embed IP may be used. Legacy Ethernet, POWERLINK, X.25, etc. may be used for the interface connecting to the external network.

## 9.1.4 Network Layer

The POWERLINK Router connects a POWERLINK network to an external network only via the Internet Protocol (IP). Therefore, the router's tasks on this layer are basically the same as that of any other standard IP router, as described in RFC 1812. These include the routing or forwarding and Network Address Translation (NAT) described in 9.1.4.2. If other protocols are used, they shall be encapsulated in IP to communicate via the POWERLINK Router.

### 9.1.4.1 Communication between POWERLINK and the external network

The POWERLINK Router shall forward the following types of packets:

- The POWERLINK Router shall forward only the IP and ICMP packets from the external network that are addressed and permitted (see 9.1.5) to the POWERLINK network. How the IP datagrams are sent on the POWERLINK network, depends on the POWERLINK network state – see 9.1.4.2.
- The POWERLINK Router shall forward only the IP and ICMP packets from the POWERLINK network that are addressed to the external network.

The ICMP datagrams supported by the POWERLINK Router are given in 9.1.4.2.1.

### 9.1.4.2 IP Coupling

The main difference between an IP router using only Legacy Ethernet and a POWERLINK Router is that the POWERLINK Router forwards IP and ICMP packets depending on the current POWERLINK network state. Note, that the POWERLINK Router accesses the POWERLINK network in the same way a CN does, because a POWERLINK Router runs only the IP coupling application.

- In the NMT\_CS\_EPL\_MODE state, the POWERLINK Router shall forward the respective IP and ICMP packets to and from the asynchronous phase. The POWERLINK Router shall respect the network access rules of the POWERLINK network, i.e. it's only allowed to send, when invited by the MN.
- In the NMT\_CS\_BASIC\_ETHERNET state, the POWERLINK Router shall access the POWERLINK network like an IEEE802.3 compliant node using CSMA/CD. There is no cycle time interval in basic ethernet mode.

Since a POWERLINK network uses fixed IP addresses, supporting only IP routing would limit the flexibility of the system. This restriction is removed using IP routing and Network Address Translation (NAT).

#### 9.1.4.2.1 IP Routing

Forwarding an IP datagram generally requires the router to choose the relevant local interface and the address of the next-hop router resp. (for the final hop) of the destination host. This choice depends upon a route database within the router. The route database is called routing table RT1\_IpRoutingTable\_XXh\_REC.

Ipv4 routing is specified in RFC 1812. Nothing should prevent a POWERLINK Router from implementing standard IP routing procedures, but it is recommended that the following points be considered:

A router's functionality, as given in RFC 1812 (Requirements for IP Version 4 Routers) is rather extensive and complex, even if only mandatory functions are implemented. A considerable fraction of these functions are neither required nor of meaningful use for a typical POWERLINK Router

application scenario. It seems therefore sensible to define the POWERLINK Router's functionality with reference to RFC 1812 and explicitly list the functions that vary from RFC1812.

- The POWERLINK Router shall use static routing. The POWERLINK Router may not support any dynamic routing algorithms like RIP or OSPF. Especially for the interface to the POWERLINK network, the POWERLINK Router should be aware of the limited bandwidth. Therefore, the POWERLINK Router shall not use dynamic routing algorithms on the POWERLINK network.
- The POWERLINK Router shall not support IP multicasting.
- The POWERLINK Router shall support IP fragmentation. IP datagrams larger than the MTU of the respective network shall be fragmented.
- The POWERLINK Router may not support MTU discovery. The MTU of the POWERLINK network is given in NMT\_CycleTiming\_REC.AsyncMTU\_U16. The POWERLINK Router shall not send frames longer than the respective Layer 2 MTU limit. However, it may receive and process frames addressed to it exceeding this limit.
- The POWERLINK Router shall use the standard ARP (RFC 826) protocol to find out the IP to MAC address relation – see 5.1.3
- Traffic precedence features (Layer 2 priority (IEEE 802.1Q-1999) as well as Layer 3 IP TOS) may be supported.
- The typical location for a POWERLINK Router is comparable to what RFC 1812 calls a “fringe router”, i. e., it connects a local network to a network of another hierarchy.
- The POWERLINK Router shall support the following ICMP messages: Echo Request/Reply, Destination Unreachable, Redirect, Time Exceeded, Parameter Problem, Address Mask Request
- The POWERLINK Router need not support the following options: Time Stamp, Source Route, Record Route.

### 9.1.4.2.1.1 Configuration

Routers shall be manageable by POWERLINK SDO and optionally by the Simple Network Management Protocol version 3 (SNMPv3).

#### 9.1.4.2.1.1.1 SNMP

The POWERLINK Router may support SNMPv3 RFC 3410-3418 on non-POWERLINK interfaces. If so the following standard MIBs for management shall be supported.

- The System, Interface, IP, ICMP, and UDP groups of MIB-II “Management Information Base of TCP/IP-Based Internets: MIB-II”, RFC 1213 shall be implemented.
- If the router implements TCP (e.g., for Telnet) then the TCP group of MIB-II “Management Information Base of TCP/IP-Based Internets: MIB-II”, STD 16, RFC1213 shall be implemented
- The IP Forwarding Table MIB “IP Forwarding Table MIB”, RFC 1354 shall be implemented.

#### 9.1.4.2.1.1.2 SDO

The functionality, which can be configured and retrieved via SDO, is a subset of that provided by SNMP. For the POWERLINK Router configuration and diagnostics the relevant objects from MIB-II RFC 1213 and RFC 1354 are mapped to SDO – see 9.1.7. Objects, which are not accessible via SDO, can be accessed via SNMP.

### 9.1.4.2.2 Network Address Translation (NAT)

NAT allows POWERLINK nodes within a POWERLINK network to transparently communicate with hosts in the external network. Basic NAT and NAPT are two varieties of NAT. Since POWERLINK nodes have fixed IP addresses, each POWERLINK Router shall implement Basic NAT which is specified in:

- RFC 2663 – IP Network Address Translator (NAT) Terminology and Considerations
- RFC 3022 – Traditional IP Network Address Translator (Traditional NAT) – extends RFC 1631

With Basic NAT, a block of external IP addresses are set aside for translating IP addresses of POWERLINK nodes in the POWERLINK network. POWERLINK nodes that must be addressed from the external network, shall be configured in the NAT table RT1\_NatTable\_XXh\_REC. The NAT table contains the POWERLINK IP EpIppAddr\_IPAD to external IP ExtIpAddr\_IPAD address translation. For

datagrams outbound from the POWERLINK network, the source IP address EpIppAddr\_IPAD shall be translated to the associated ExtIpAddr\_IPAD called Source-NAT. For inbound packets, the destination IP address ExtIpAddr\_IPAD shall be translated to the associated EpIppAddr\_IPAD, called Destination-NAT. Independent from inbound or outbound IP telegrams, the related fields such as IP, TCP, UDP and ICMP header checksums shall be corrected. This kind of NAT is also called bidirectional NAT or 1to1 NAT (see RFC 2663). Fig. 109 illustrates an example for bidirectional NAT.

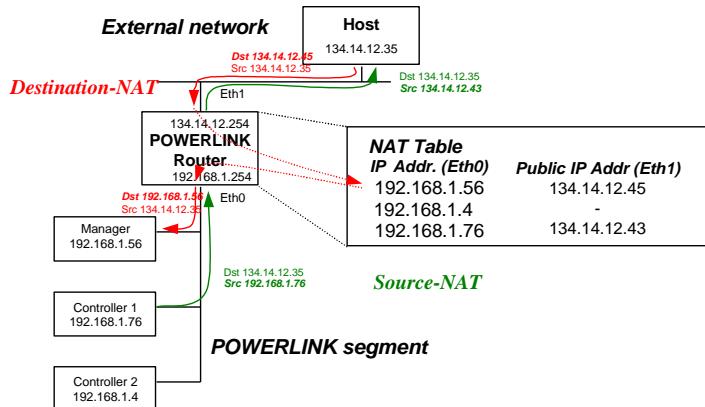


Fig. 109. Symmetrical n-to-n NAT

The visibility of POWERLINK nodes to the external network can be controlled by bidirectional NAT. Therefore, nodes that should not be accessed from the external network, can be concealed. POWERLINK nodes that are not configured in the NAT table may not communicate with nodes in the external network i.e. the packets are dropped.

In more detail POWERLINK differentiates between Source-NAT where the source address is changed and Destination-NAT where the destination address is changed. Fig. 110 illustrates the general NAT architecture.

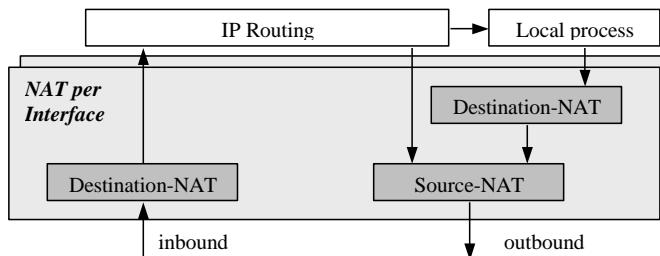


Fig. 110. NAT architecture

For outbound IP datagrams, that is from the POWERLINK network to the external network we shall use Source-NAT (S-NAT). S-NAT changes the source address of the IP / ICMP packet. This is done in the output chain, just before it is finally sent out. This is an important detail, since it means that anything else on the Router itself (routing, packet filtering) will see the packet unchanged.

For inbound IP datagrams, that is from the external network to the POWERLINK network we shall use Destination-NAT (D-NAT). D-NAT changes the destination address of the IP / ICMP packet. D-NAT is done in the input chain, just as the packet comes in. This means that anything else on the Router itself (routing, packet filtering) will see the packet going to its 'real' destination.

The following figure illustrates the interaction between a POWERLINK network and an external network. It is presented for informative purpose only.

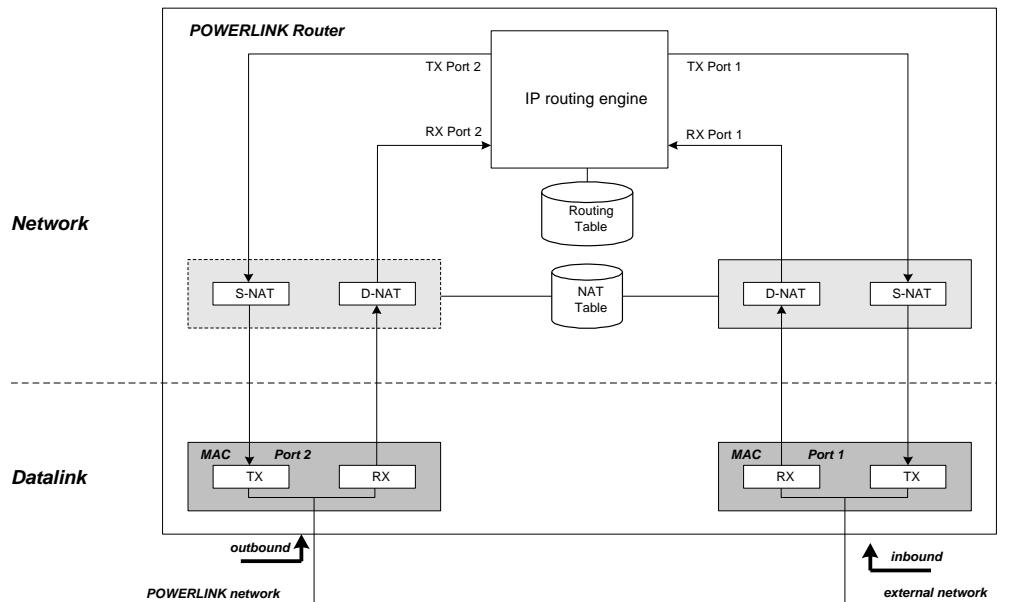


Fig. 111. Integration of NAT in the POWERLINK router

If bidirectional NAT is configured for a POWERLINK-host a host on the external network who wants to communicate with that POWERLINK-host it looks as if the POWERLINK-host is on the same network like itself.

External addresses of the NAT table are in the subnet of the external network:

On the interface to the external network the POWERLINK Router shall behave like a host, accepting all external IP addresses *ExtIpAddr\_IPAD* listed in the NAT table RT1\_NatTable\_XXh\_REC. Note, that the POWERLINK Router must respond to an ARP request, requesting one of the external IP addresses *StaticExtIpAddr\_IPAD* listed in the NAT table. Therefore this interface does not obtain the IP packet like a router does i.e. the packet is addressed to the router if it is not in the subnet.

- External addresses of the NAT table are not in the subnet of the direct connected external network:  
The interface to the external Network is addressed like a router.
- In every case:  
The interface of the POWERLINK Router to the POWERLINK network shall act like a router.

Additional information about NAT is given in <http://www.netfilter.org>, RFC 2993 – Architectural Implications of NAT and RFC 3027 – Protocol Complications with the IP Network Address Translator.

### 9.1.4.2.2.1 Configuration

Network address translation shall be manageable by POWERLINK SDO and optionally by SNMPv3.

#### 9.1.4.2.2.1.1 SNMP

The POWERLINK Router MIB specifies the managed objects to configure NAT. Therefore the NAT Group of the POWERLINK Router MIB shall be implemented.

#### 9.1.4.2.2.1.2 SDO

The RT1\_NatTable\_XXh\_REC object specifies the NAT table.

## 9.1.5 Security

Connecting a POWERLINK network via the POWERLINK Router to an external network (e.g. a LAN) enables the communication with other resources and services. This presents an enormous benefit to the entire system. On the other hand a POWERLINK Router represents a security risk, giving hackers, crackers and intruders the opportunity to access nodes in the POWERLINK network. The POWERLINK Router is the best place to add security mechanisms to protect a POWERLINK network, since the POWERLINK Router connects an un-trusted network with the trusted POWERLINK network.

Security mechanisms consist of rules and restriction but also must ensure availability and ease of use. Therefore, security must always be used at the right level. When applying security, a risk assessment must typically be performed. The risk assessment shows the level of security that must be supported. A risk assessment examines the following questions.

- Which network is connected to the POWERLINK network (trusted or un-trusted) ?
- Must we cope with accidental “attacks/errors” (handling errors) ?
- Must we cope with “evil-minded” malicious attacks (hackers, crackers, intruders, sabotage) ?
- ...

Depending on the result of the risk assessment, the appropriate mechanisms, listed below, must be used.

- Secrecy
- Integrity
- Authentication and Authorisation
- Availability of the information

This specification for the POWERLINK Router assumes that:

- The POWERLINK Router is connected to a trusted network (e. g. the factory floor network). Note that additional security considerations must be taken if the factory floor network is connected to the office network or –even harder – to the internet.
- The POWERLINK Router does not protect the POWERLINK network against evil minded attacks such as ICMP attacks, spoofing, etc..

The security assumption stated above, requires that a POWERLINK Router shall provide a basic security level. The basic security level is achieved using a Packet Filter (stateless Firewall). For higher security demands stateful Firewalls, VPN servers and Intrusion Detection systems must be considered. However, this is outside the scope of this specification.

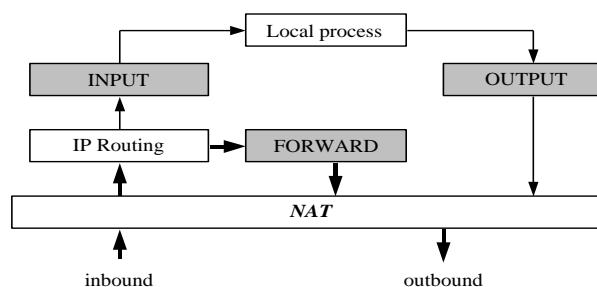
Support of Routing Type 1 security features is optional. Support shall be indicated by D\_RT1\_RT1SecuritySupport\_BOOL.

### 9.1.5.1 Packet Filter – Firewall

A Packet Filter is a firewall element that analyses and controls inbound and outbound traffic of the datalink-, network and transport layers. A firewall in the sense of a Packet Filter physically decouples an un-trusted network from a trusted network. This enables a global point of security control. The Packet Filter functionality shall be implemented on the POWERLINK Router since the POWERLINK Router separates both networks.

In an effort to protect the POWERLINK network from various risks, both accidental and malicious, a Packet Filter should be deployed at a network’s ingress points – the POWERLINK Router. A Packet Filter maintains the access between the interfaces through Access Control Lists (ACLs).

This specification defines the filter entries and the tables that shall be implemented. Fig. 112 illustrates the involved tables that represent also the position where the IP datagrams shall be evaluated.



*The bold arrows present the route between POWERLINK and external network.*

Fig. 112. Filter tables of the packet filter

The INPUT table RT1\_AclInTable\_Xh\_REC shall contain the filter entries for packets that are addressed to the POWERLINK Router itself. The FORWARD table RT1\_AclFwdTable\_XXh\_REC shall contain the filter entries for packets routed through the POWERLINK Router. The OUTPUT table RT1\_AclOutTable\_Xh\_REC shall contain the filter entries for packets locally generated. Each table

shall have its default policy (RT1\_SecurityGroup\_REC.InTablePolicy\_U8, RT1\_SecurityGroup\_REC.FwdTablePolicy\_U8, RT1\_SecurityGroup\_REC.OutTablePolicy\_U8).

### **9.1.5.1.1 ACL – Filter Entries**

An Access Control List is a sequential list of permit and deny conditions known as a rule. The list defines the connections permitted to pass through the POWERLINK Router as well as connections that are denied. ACL's act as a basic method of limiting access to the POWERLINK network.

A POWERLINK Router can support the following optional filter entries:

Datalink Layer Ethernet MAC frames (DIX2):

- Source MAC address (*SrcMac\_MAC*) of the Ethernet MAC header.

A POWERLINK Router shall support the following filter entries:

Network Layer

- Source IP address (*SrcIp\_IPAD*) field of the IP header / Source IP network mask (*SrcMask\_IPAD*)
- Destination IP address (*DstIp\_IPAD*) field of the IP header / Destination IP network mask (*DstMask\_IPAD*)
- Protocol (*Protocol\_U8*) field of the IP header.

Transport Layer if the Protocol is either UDP or TCP

- Source L4 Port (*SrcPort\_U16*) of the TCP or UDP header.
- Destination L4 Port (*DstPort\_U16*) of the TCP or UDP header.

### **9.1.5.1.2 Filter strategy**

A firewall rule specifies criteria for a packet, and a target. A target specifies what do with this packet if the rule matches. A rule matches if all specified entries from the assessed packet match the corresponding entry of the current rule. If the packet does not match, the next rule in the respective table is examined; if it does match, then the target (*Target\_U8*) of the rule is executed, which is either ACCEPT or DROP. ACCEPT means to let the packet through. DROP means to drop the packet on the floor.

If no rule matches, it shall be up to the policy of the respective table (RT1\_SecurityGroup\_REC.InTablePolicy\_U8, RT1\_SecurityGroup\_REC.FwdTablePolicy\_U8, RT1\_SecurityGroup\_REC.OutTablePolicy\_U8) to process the packet.

### **9.1.5.1.3 Configuration**

The security settings and the ACLs shall be manageable by POWERLINK SDO and optionally by SNMPv3.

#### **9.1.5.1.3.1 SNMP**

The POWERLINK Router MIB specifies the managed objects to configure the Packet Filter. Therefore the Security Group of the POWERLINK Router MIB shall be implemented.

#### **9.1.5.1.3.2 SDO**

The following objects shall be implemented to configure the Packet Filter:

- *RT1\_SecurityGroup\_REC*
- *RT1\_AclFwdTable\_XXh\_REC*
- *RT1\_AclInTable\_Xh\_REC*
- *RT1\_AclOutTable\_Xh\_REC*

## **9.1.6 Additional Services of a POWERLINK Router**

Besides the data transport service between the POWERLINK and the normal Ethernet network, the POWERLINK Router may offer extended services. These are:

- Precision Time Protocol (IEEE1588) boundary clock functionality,
- BOOTP/DHCP Relay,
- Address Allocation DHCP (Option 82),

- Enhanced security mechanisms such as IEEE 802.1X-2001 Port-Based Network Access Control Virtual Private Network (VPN) Server, Intrusion Detection.
- DNS Server / Cache

## 9.1.7 Object description

### 9.1.7.1 Object 1E80<sub>h</sub>: RT1\_EplRouter\_REC

RT1\_EplRouter\_REC specifies attributes for POWERLINK Router configuration. This object shall only be implemented for routing type 1.

|           |                    |             |             |
|-----------|--------------------|-------------|-------------|
| Index     | 1E80 <sub>h</sub>  | Object Type | RECORD      |
| Name      | RT1_EplRouter_REC  |             |             |
| Data Type | RT1_EplRouter_TYPE | Category    | Conditional |

- Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 2               | Access      | ro |
| Default Value | 2               | PDO Mapping | No |

- Sub-Index 01<sub>h</sub>: EnableNat\_BOOL**

|             |                 |          |     |
|-------------|-----------------|----------|-----|
| Sub-Index   | 01 <sub>h</sub> |          |     |
| Name        | EnableNat_BOOL  |          |     |
| Data Type   | BOOL            | Category | M   |
| Value Range | BOOL            | Access   | rws |

Enables or disables the Network Address Translation on the POWERLINK Router.

- Sub-Index 02<sub>h</sub>: EnablePacketFiltering\_BOOL**

|             |                            |          |     |
|-------------|----------------------------|----------|-----|
| Sub-Index   | 02 <sub>h</sub>            |          |     |
| Name        | EnablePacketFiltering_BOOL |          |     |
| Data Type   | BOOL                       | Category | M   |
| Value Range | BOOL                       | Access   | rws |

Depending on the value of EnablePacketFiltering\_BOOL, the Packet Filer on the POWERLINK Router is enabled or disabled.

### 9.1.7.2 Object 1E90<sub>h</sub> .. 1ECF<sub>h</sub>: RT1\_IpRoutingTable\_XXh\_REC

The RT1\_IpRoutingTable\_XXh\_REC object is a subset of RFC1354, which defines the routers forwarding table. The routing table shall have 64 entries that may be configured via SDO.

To allow access by name “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1E90<sub>h</sub>. It shall be incremented up to “\_3Fh” corresponding to object index 1ECF<sub>h</sub>. This object shall only be implemented for routing type 1.

|           |  |             |             |
|-----------|--|-------------|-------------|
| Index     | 1E90 <sub>h</sub> .. 1ECF <sub>h</sub> | Object Type | RECORD      |
| Name      | RT1_IpRoutingTable_XXh_REC             |             |             |
| Data Type | RT1_IpRoutingTable_TYPE                | Category    | Conditional |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 7               | Access      | ro |
| Default Value | 7               | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub>: IpForwardDest\_IPAD**

|               |                    |             |     |
|---------------|--------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub>    |             |     |
| Name          | IpForwardDest_IPAD |             |     |
| Data Type     | IP_ADDRESS         | Category    | M   |
| Value Range   | IP_ADDRESS         | Access      | rws |
| Default Value | -                  | PDO Mapping | No  |

The destination IP address of this route. An entry with a value of 0.0.0.0 is considered a default route. This object may not take a Multicast (Class D) address value. Any assignment (implicit or otherwise) of an instance of this object to a value x must be rejected if the bitwise logical-AND of x with the value of the corresponding instance of the IpForwardMask\_IPAD object is not equal to x.

- **Sub-Index 02<sub>h</sub>: IpForwardMask\_IPAD**

|               |                    |             |     |
|---------------|--------------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub>    |             |     |
| Name          | IpForwardMask_IPAD |             |     |
| Data Type     | IP_ADDRESS         | Category    | M   |
| Value Range   | IP_ADDRESS         | Access      | rws |
| Default Value | 0.0.0.0            | PDO Mapping | No  |

Indicate the mask to be logical-ANDed with the destination address before being compared to the value in the IpForwardDest\_IPAD field. For those systems that do not support arbitrary subnet masks, an agent constructs the value of the IpForwardMask\_IPAD by reference to the IP Address Class. Any assignment (implicit or otherwise) of an instance of this object to a value x must be rejected if the bitwise logical-AND of x with the value of the corresponding instance of the IpForwardDest\_IPAD object is not equal to IpForwardDest\_IPAD.

- **Sub-Index 03<sub>h</sub>: IpForwardNextHop\_IPAD**

|               |                       |             |     |
|---------------|-----------------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub>       |             |     |
| Name          | IpForwardNextHop_IPAD |             |     |
| Data Type     | IP_ADDRESS            | Category    | M   |
| Value Range   | IP_ADDRESS            | Access      | rws |
| Default Value | -                     | PDO Mapping | No  |

On remote routes, the address of the next system en route; Otherwise, 0.0.0.0.

- **Sub-Index 04<sub>h</sub>: IpForwardType\_U8**

|               |  |             |     |
|---------------|--|-------------|-----|
| Sub-Index     | 04 <sub>h</sub>  |             |     |
| Name          | IpForwardType_U8   |             |     |
| Data Type     | UNSIGNED8  | Category    | M   |
| Value Range   | Other (1), -- not specified<br>invalid (2), -- logically deleted<br>local (3), -- local interface<br>remote (4), -- remote destination | Access      | rws |
| Default Value | invalid (2)  | PDO Mapping | No  |

The type of route. Note that local(3) refers to a route for which the next hop is the final destination; remote(4) refers to a route for which the next hop is not the final destination. Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the IpForwardTable\_REC object. That is, it effectively disassociates the destination identified with said entry from the route identified with said entry. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management nodes must be prepared to receive tabular information from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant IpForwardType\_U8 object.

- **Sub-Index 05<sub>h</sub>: IpForwardAge\_U32**

|               |                  |             |    |
|---------------|------------------|-------------|----|
| Sub-Index     | 05 <sub>h</sub>  |             |    |
| Name          | IpForwardAge_U32 |             |    |
| Data Type     | UNSIGNED32       | Category    | M  |
| Value Range   | UNSIGNED32       | Access      | ro |
| Default Value | -                | PDO Mapping | No |

The number of seconds since this route was last updated or otherwise determined to be correct. Note that no semantics of 'too old' can be implied except through knowledge of the routing protocol by which the route was learned.

- **Sub-Index 06<sub>h</sub>: IpForwardItfIndex\_U16**

|               |                       |             |     |
|---------------|-----------------------|-------------|-----|
| Sub-Index     | 06 <sub>h</sub>       |             |     |
| Name          | IpForwardItfIndex_U16 |             |     |
| Data Type     | UNSIGNED16            | Category    | M   |
| Value Range   | UNSIGNED16            | Access      | rws |
| Default Value | -                     | PDO Mapping | No  |

The IpForwardItfIndex\_U16 identifies the local interface (NMT\_LocItfGroupN\_REC.-ItfIndex\_U16) through which the next hop of this route should be reached.

- **Sub-Index 07<sub>h</sub>: IpForwardMetric1\_S32**

|               |                      |             |     |
|---------------|----------------------|-------------|-----|
| Sub-Index     | 07 <sub>h</sub>      |             |     |
| Name          | IpForwardMetric1_S32 |             |     |
| Data Type     | INTEGER32            | Category    | M   |
| Value Range   | INTEGER32            | Access      | rws |
| Default Value | -1                   | PDO Mapping | No  |

An alternate routing metric for this route. If this metric is not used, its value should be set to -1. A metric indicates the cost of using a route, which is typically the number of hops to the IP destination. Anything on the local subnet is one hop, and each router crossed after that is an additional hop. If there are multiple routes to the same destination with different metrics, the route with the lowest metric is selected.

### 9.1.7.3 Object 1D00<sub>h</sub> .. 1DFF<sub>h</sub>: RT1\_NatTable\_XXh\_REC

This object specifies the NAT table located on the POWERLINK Router for bidirectional (1to1) NAT. The NAT table shall have 256 entries that may be configured via SDO.

To allow access by name “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1D00<sub>h</sub>. It shall be incremented up to “\_FFh” corresponding to object index 1DFF<sub>h</sub>. This object shall only be implemented for routing type 1.

|           |  |             |             |
|-----------|--|-------------|-------------|
| Index     | 1D00 <sub>h</sub> .. 1DFF <sub>h</sub> | Object Type | RECORD      |
| Name      | RT1_NatTable_XXh_REC                   |             |             |
| Data Type | RT1_NatTable_TYPE                      | Category    | Conditional |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |    |
|---------------|-----------------|-------------|----|
| Sub-Index     | 00 <sub>h</sub> |             |    |
| Name          | NumberOfEntries |             |    |
| Value Range   | 4               | Access      | ro |
| Default Value | 4               | PDO Mapping | No |

- **Sub-Index 01<sub>h</sub>: EpIpAddr\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | EpIpAddr_IPAD   |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

StaticEpIpAddr\_IPAD contains the IP address to the POWERLINK network.

- **Sub-Index 02<sub>h</sub>: ExtIpAddr\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | ExtIpAddr_IPAD  |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

StaticExtIpAddr\_IPAD contains the IP address to the external network.

- **Sub-Index 03<sub>h</sub>: Mask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | Mask_IPAD       |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

This is the Network-mask for EpIpAddr\_IPAD and ExtIpAddr\_IPAD. Bidirectional NAT only works for single hosts or IP-address ranges which are equal in size for source and destination. Thus only one mask is needed.

- **Sub-Index 04<sub>h</sub>: Type\_U8**

|               |   |             |     |
|---------------|---|-------------|-----|
| Sub-Index     | 04 <sub>h</sub>   |             |     |
| Name          | Type_U8   |             |     |
| Data Type     | UNSIGNED8   | Category    | M   |
| Value Range   | other (1), -- not specified<br>invalid (2), -- logically deleted<br>bidirectional-nat (3) – bidirectional NAT | Access      | rws |
| Default Value | invalid (2)   | PDO Mapping | No  |

Setting this object to the value invalid(2) has the effect of invalidating the corresponding entry in the Type\_U8 object. That is, it effectively disassociates the respective entry from Table\_REC. It is an implementation-specific matter as to whether the agent removes an invalidated entry from the table. Accordingly, management nodes must be prepared to receive tabular information

from agents that corresponds to entries not currently in use. Proper interpretation of such entries requires examination of the relevant Type\_U8.

### 9.1.7.4 Object 1E81<sub>h</sub>: RT1\_SecurityGroup\_REC

The RT1\_SecurityGroup\_REC contains information about the security settings of the POWERLINK Router. This object shall only be implemented for routing type 1.

|           |                        |             |             |
|-----------|------------------------|-------------|-------------|
| Index     | 1E81 <sub>h</sub>      | Object Type | RECORD      |
| Name      | RT1_SecurityGroup_REC  |             |             |
| Data Type | RT1_SecurityGroup_TYPE | Category    | Conditional |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 3               | Access      | const |
| Default Value | 3               | PDO Mapping | No    |

- Sub-Index 01<sub>h</sub>: FwdTablePolicy\_U8

|               |                                      |             |     |
|---------------|--------------------------------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub>                      |             |     |
| Name          | FwdTablePolicy_U8                    |             |     |
| Data Type     | UNSIGNED8                            | Category    | M   |
| Value Range   | accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | accept                               | PDO Mapping | No  |

FwdTablePolicy\_U8 specifies the default policy of the FORWARD table (RT1\_AclFwdTable\_XXh\_REC).

- Sub-Index 02<sub>h</sub>: InTablePolicy\_U8

|               |                                      |             |     |
|---------------|--------------------------------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub>                      |             |     |
| Name          | InTablePolicy_U8                     |             |     |
| Data Type     | UNSIGNED8                            | Category    | M   |
| Value Range   | accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | accept                               | PDO Mapping | No  |

InTablePolicy\_U8 specifies the default policy of the INPUT table (RT1\_AclInTable\_Xh\_REC).

- Sub-Index 03<sub>h</sub>: OutTablePolicy\_U8

|               |                                      |             |     |
|---------------|--------------------------------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub>                      |             |     |
| Name          | OutTablePolicy_U8                    |             |     |
| Data Type     | UNSIGNED8                            | Category    | M   |
| Value Range   | accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | accept                               | PDO Mapping | No  |

OutTablePolicy\_U8 specifies the default policy of the OUTPUT table (RT1\_AclOutTable\_Xh\_REC).

### 9.1.7.5 Object 1B00<sub>h</sub>..1BFF<sub>h</sub>: RT1\_AclFwdTable\_XXh\_REC

This object specifies the Access Control List (ACL) for the FORWARD table located on the POWERLINK Router – see 9.1.5.1. The FORWARD table shall have 256 entries that may be configured via SDO.

To allow access by name “\_XXh” shall be replaced by a name index. Name index shall be “\_00h” if object index is 1B00<sub>h</sub>. It shall be incremented up to “\_3Fh” corresponding to object index 1BFF<sub>h</sub>. This object shall only be implemented for routing type 1.

|           |   |             |             |
|-----------|---|-------------|-------------|
| Index     | 1B00 <sub>h</sub> ... 1BFF <sub>h</sub> | Object Type | RECORD      |
| Name      | RT1_AclFwdTable_XXh_REC                 |             |             |
| Data Type | RT1_AclTable_TYPE                       | Category    | Conditional |

- **Sub-Index 00<sub>h</sub>: NumberOfEntries**

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 9               | Access      | const |
| Default Value | 9               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: SrcIp\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | SrcIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcIp\_IPAD specifies a plain source IP address for the respective entry. A value of 0.0.0.0 and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 02<sub>h</sub>: SrcMask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | SrcMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcMask\_IPAD is the network mask to the according source IP address SrcIp\_IPAD for the respective entry. A value of 0.0.0.0 for SrcIp\_IPAD and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 03<sub>h</sub>: DstIp\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | DstIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstIp\_IPAD specifies a plain destination IP address for the respective entry. A value of 0.0.0.0 and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 04<sub>h</sub>: DstMask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 04 <sub>h</sub> |             |     |
| Name          | DstMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstMask\_IPAD is the network mask to the according destination IP address DstIp\_IPAD for the respective entry. A value of 0.0.0.0 for DstIp\_IPAD and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 05<sub>h</sub>: Protocol\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 05 <sub>h</sub> |             |     |
| Name          | Protocol_U8     |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

The protocol of the rule or the packet to check. In the Internet Protocol version 4 (Ipv4) [RFC791] there is a field, called “Protocol”, to identify the next level protocol. This is an 8 bit field. The specified protocol is a numeric number listed in <http://www.iana.org/assignments/protocol-numbers>. The number zero is equivalent to all protocols.

- **Sub-Index 06<sub>h</sub>: SrcPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 06 <sub>h</sub> |             |     |
| Name          | SrcPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

SrcPort\_U16 specifies the source port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 07<sub>h</sub>: DstPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 07 <sub>h</sub> |             |     |
| Name          | DstPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

DstPort\_U16 contains the destination port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 08<sub>h</sub>: SrcMac\_MAC**

|               |                   |             |     |
|---------------|-------------------|-------------|-----|
| Sub-Index     | 08 <sub>h</sub>   |             |     |
| Name          | SrcMac_MAC        |             |     |
| Data Type     | MAC_ADDRESS       | Category    | M   |
| Value Range   | MAC_ADDRESS       | Access      | rws |
| Default Value | 00:00:00:00:00:00 | PDO Mapping | No  |

Match source MAC address. A value of 00:00:00:00:00:00 specifies any source MAC.

- **Sub-Index 09<sub>h</sub>: Target\_U8**

|               |   |             |     |
|---------------|---|-------------|-----|
| Sub-Index     | 09 <sub>h</sub>                                     |             |     |
| Name          | Target_U8   |             |     |
| Data Type     | UNSIGNED8   | Category    | M   |
| Value Range   | Invalid (0)<br>accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | Invalid (0)   | PDO Mapping | No  |

Specifies the target of the rule. If the value is zero, the entry shall be invalid. If the rule matches the target, the value is either accept, drop or reject.

### 9.1.7.6 Object 1ED0<sub>h</sub>.. 1EDF<sub>h</sub>: RT1\_AclInTable\_Xh\_REC

This object specifies the Access Control List (ACL) for the INPUT table located on the POWERLINK Router – see 9.1.5.1. The INPUT table shall have 16 entries that may be configured via SDO.

To allow access by name “\_Xh” shall be replaced by a name index. Name index shall be “\_0h” if object index is 1ED0<sub>h</sub>. It shall be incremented up to “\_Fh” corresponding to object index 1EDF<sub>h</sub>. This object shall only be implemented for routing type 1.

|           |  |             |             |
|-----------|--|-------------|-------------|
| Index     | 1ED0 <sub>h</sub> .. 1EDF <sub>h</sub> | Object Type | RECORD      |
| Name      | RT1_AclInTable_Xh_REC                  |             |             |
| Data Type | RT1_AclTable_TYPE                      | Category    | Conditional |

- Sub-Index 00<sub>h</sub>: NumberOfEntries

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 9               | Access      | const |
| Default Value | 9               | PDO Mapping | No    |

- Sub-Index 01<sub>h</sub>: SrcIp\_IPAD

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | SrcIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcIp\_IPAD contains a plain source IP address for the respective entry. A value of 0.0.0.0 and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- Sub-Index 02<sub>h</sub>: SrcMask\_IPAD

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | SrcMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcMask\_IPAD is the network mask to the according source IP address SrcIp\_IPAD for the respective entry. A value of 0.0.0.0 for SrcIp\_IPAD and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- Sub-Index 03<sub>h</sub>: DstIp\_IPAD

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | DstIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstIp\_IPAD specifies a plain destination IP address for the respective entry. A value of 0.0.0.0 and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 04<sub>h</sub>: DstMask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 04 <sub>h</sub> |             |     |
| Name          | DstMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstMask\_IPAD is the network mask to the according destination IP address DstIp\_IPAD for the respective entry. A value of 0.0.0.0 for DstIp\_IPAD and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 05<sub>h</sub>: Protocol\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 05 <sub>h</sub> |             |     |
| Name          | Protocol_U8     |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

The protocol of the rule or the packet to check. In the Internet Protocol version 4 (Ipv4) [RFC791] there is a field, called "Protocol", to identify the next level protocol. This is an 8 bit field. The specified protocol is a numeric number listed in <http://www.iana.org/assignments/protocol-numbers>. The number zero is equivalent to all protocols.

- **Sub-Index 06<sub>h</sub>: SrcPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 06 <sub>h</sub> |             |     |
| Name          | SrcPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

SrcPort\_U16 specifies the source port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 07<sub>h</sub>: DstPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 07 <sub>h</sub> |             |     |
| Name          | DstPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

DstPort\_U16 contains the destination port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 08<sub>h</sub>: SrcMac\_MAC**

|               |                   |             |     |
|---------------|-------------------|-------------|-----|
| Sub-Index     | 08 <sub>h</sub>   |             |     |
| Name          | SrcMac_MAC        |             |     |
| Data Type     | MAC_ADDRESS       | Category    | M   |
| Value Range   | MAC_ADDRESS       | Access      | rws |
| Default Value | 00:00:00:00:00:00 | PDO Mapping | No  |

Match source MAC address. A value of 00:00:00:00:00:00 specifies any source MAC.

- **Sub-Index 09<sub>h</sub>: Target\_U8**

|               |   |             |     |
|---------------|---|-------------|-----|
| Sub-Index     | 09 <sub>h</sub>                                     |             |     |
| Name          | Target_U8   |             |     |
| Data Type     | UNSIGNED8   | Category    | M   |
| Value Range   | Invalid (0)<br>accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | Invalid (0)   | PDO Mapping | No  |

Specifies the target of the rule. If the value is zero, the entry shall be invalid. If the rule matches the target, the value is either accept, drop or reject.

### **9.1.7.7 Object 1EE0<sub>h</sub>..1EEF<sub>h</sub>: RT1\_AclOutTable\_Xh\_REC**

This object specifies the Access Control List (ACL) for the OUTPUT table located on the POWERLINK Router – see 9.1.5.1. The routing table shall have 16 entries that may be configured via SDO.

To allow access by name “\_Xh” shall be replaced by a name index. Name index shall be “\_0h” if object index is 1EE0h. It shall be incremented up to “\_Fh” corresponding to object index 1EEFh. This object shall only be implemented for routing type 1.

|           |  |             |             |
|-----------|--|-------------|-------------|
| Index     | 1EE0 <sub>h</sub> .. 1EEF <sub>h</sub> | Object Type | RECORD      |
| Name      | RT1_AclOutTable_Xh_REC                 |             |             |
| Data Type | RT1_AclTable_TYPE                      | Category    | Conditional |

- Sub-Index  $00_h$ : NumberOfEntries

|               |                 |             |       |
|---------------|-----------------|-------------|-------|
| Sub-Index     | 00 <sub>h</sub> |             |       |
| Name          | NumberOfEntries |             |       |
| Value Range   | 9               | Access      | const |
| Default Value | 9               | PDO Mapping | No    |

- **Sub-Index 01<sub>h</sub>: SrcIp\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 01 <sub>h</sub> |             |     |
| Name          | SrcIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcIp\_IPAD specifies a plain source IP address for the respective entry. A value of 0.0.0.0 and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 02<sub>h</sub>: SrcMask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 02 <sub>h</sub> |             |     |
| Name          | SrcMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

SrcMask\_IPAD is the network mask to the according source IP address SrcIp\_IPAD for the respective entry. A value of 0.0.0.0 for SrcIp\_IPAD and a SrcMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 03<sub>h</sub>: DstIp\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 03 <sub>h</sub> |             |     |
| Name          | DstIp_IPAD      |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstIp\_IPAD specifies a plain destination IP address for the respective entry. A value of 0.0.0.0 and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 04<sub>h</sub>: DstMask\_IPAD**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 04 <sub>h</sub> |             |     |
| Name          | DstMask_IPAD    |             |     |
| Data Type     | IP_ADDRESS      | Category    | M   |
| Value Range   | IP_ADDRESS      | Access      | rws |
| Default Value | -               | PDO Mapping | No  |

DstMask\_IPAD is the network mask to the according destination IP address DstIp\_IPAD for the respective entry. A value of 0.0.0.0 for DstIp\_IPAD and a DstMask\_IPAD of 0.0.0.0 shall indicate any IP address.

- **Sub-Index 05<sub>h</sub>: Protocol\_U8**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 05 <sub>h</sub> |             |     |
| Name          | Protocol_U8     |             |     |
| Data Type     | UNSIGNED8       | Category    | M   |
| Value Range   | UNSIGNED8       | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

The protocol of the rule or the packet to check. In the Internet Protocol version 4 (Ipv4) [RFC791] there is a field, called "Protocol", to identify the next level protocol. This is an 8 bit field. The specified protocol is a numeric number listed in <http://www.iana.org/assignments/protocol-numbers>. The number zero is equivalent to all protocols.

- **Sub-Index 06<sub>h</sub>: SrcPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 06 <sub>h</sub> |             |     |
| Name          | SrcPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

SrcPort\_U16 specifies the source port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 07<sub>h</sub>: DstPort\_U16**

|               |                 |             |     |
|---------------|-----------------|-------------|-----|
| Sub-Index     | 07 <sub>h</sub> |             |     |
| Name          | DstPort_U16     |             |     |
| Data Type     | UNSIGNED16      | Category    | M   |
| Value Range   | UNSIGNED16      | Access      | rws |
| Default Value | 0               | PDO Mapping | No  |

DstPort\_U16 contains the destination port if the protocol (Protocol\_U8) TCP or UDP is specified. A value of zero indicates any protocol.

- **Sub-Index 08<sub>h</sub>: SrcMac\_MAC**

|               |                   |             |     |
|---------------|-------------------|-------------|-----|
| Sub-Index     | 08 <sub>h</sub>   |             |     |
| Name          | SrcMac_MAC        |             |     |
| Data Type     | MAC_ADDRESS       | Category    | M   |
| Value Range   | MAC_ADDRESS       | Access      | rws |
| Default Value | 00:00:00:00:00:00 | PDO Mapping | No  |

Match source MAC address. A value of 00:00:00:00:00:00 specifies any source MAC.

- **Sub-Index 09<sub>h</sub>: Target\_U8**

|               |   |             |     |
|---------------|---|-------------|-----|
| Sub-Index     | 09 <sub>h</sub>                                     |             |     |
| Name          | Target_U8   |             |     |
| Data Type     | UNSIGNED8   | Category    | M   |
| Value Range   | Invalid (0)<br>accept (1)<br>drop (2)<br>reject (3) | Access      | rws |
| Default Value | Invalid (0)   | PDO Mapping | No  |

Specifies the target of the rule. If the value is zero, the entry shall be invalid. If the rule matches the target, the value is either accept, drop or reject.

## 9.1.7.8 Router Type I Object Types

### 9.1.7.8.1 Object 0430<sub>h</sub>: RT1\_EplRouter\_TYPE

|                 |                            |                   |           |
|-----------------|----------------------------|-------------------|-----------|
| Index           | 0430 <sub>h</sub>          | Object Type       | DEFSTRUCT |
| Name            | RT1_EplRouter_TYPE         |                   |           |
| Sub-Index       | Component Name             | Value             | Data Type |
| 00 <sub>h</sub> | NumberOfEntries            | 02 <sub>h</sub>   |           |
| 01 <sub>h</sub> | EnableNat_BOOL             | 0001 <sub>h</sub> | BOOLEAN   |
| 02 <sub>h</sub> | EnablePacketFiltering_BOOL | 0001 <sub>h</sub> | BOOLEAN   |

### 9.1.7.8.2 Object 0431<sub>h</sub>: RT1\_IpRoutingTable\_TYPE

|                 |                         |                   |            |
|-----------------|-------------------------|-------------------|------------|
| Index           | 0431 <sub>h</sub>       | Object Type       | DEFSTRUCT  |
| Name            | RT1_IpRoutingTable_TYPE |                   |            |
| Sub-Index       | Component Name          | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries         | 07 <sub>h</sub>   |            |
| 01 <sub>h</sub> | IpForwardDest_IPAD      | 0402 <sub>h</sub> | IP_ADDRESS |
| 02 <sub>h</sub> | IpForwardMask_IPAD      | 0402 <sub>h</sub> | IP_ADDRESS |
| 03 <sub>h</sub> | IpForwardNextHop_IPAD   | 0402 <sub>h</sub> | IP_ADDRESS |
| 04 <sub>h</sub> | IpForwardType_U8        | 0005 <sub>h</sub> | UNSIGNED8  |
| 05 <sub>h</sub> | IpForwardAge_U32        | 0007 <sub>h</sub> | UNSIGNED32 |
| 06 <sub>h</sub> | IpForwardItfIndex_U16   | 0007 <sub>h</sub> | UNSIGNED16 |
| 07 <sub>h</sub> | IpForwardMetric1_S32    | 0004 <sub>h</sub> | INTEGER32  |

### 9.1.7.8.3 Object 0432<sub>h</sub>: RT1\_NatTable\_TYPE

|                 |                   |                   |            |
|-----------------|-------------------|-------------------|------------|
| Index           | 0432 <sub>h</sub> | Object Type       | DEFSTRUCT  |
| Name            | RT1_NatTable_TYPE |                   |            |
| Sub-Index       | Component Name    | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries   | 04 <sub>h</sub>   |            |
| 01 <sub>h</sub> | EplIpAddr_IPAD    | 0402 <sub>h</sub> | IP_ADDRESS |
| 02 <sub>h</sub> | ExtIpAddr_IPAD    | 0402 <sub>h</sub> | IP_ADDRESS |
| 03 <sub>h</sub> | Mask_IPAD         | 0402 <sub>h</sub> | IP_ADDRESS |
| 04 <sub>h</sub> | Type_U8           | 0005 <sub>h</sub> | UNSIGNED8  |

### 9.1.7.8.4 Object 0433<sub>h</sub>: RT1\_SecurityGroup\_TYPE

|                 |                        |                   |           |
|-----------------|------------------------|-------------------|-----------|
| Index           | 0433 <sub>h</sub>      | Object Type       | DEFSTRUCT |
| Name            | RT1_SecurityGroup_TYPE |                   |           |
| Sub-Index       | Component Name         | Value             | Data Type |
| 00 <sub>h</sub> | NumberOfEntries        | 03 <sub>h</sub>   |           |
| 01 <sub>h</sub> | FwdTablePolicy_U8      | 0005 <sub>h</sub> | UNSIGNED8 |
| 02 <sub>h</sub> | InTablePolicy_U8       | 0005 <sub>h</sub> | UNSIGNED8 |
| 03 <sub>h</sub> | OutTablePolicy_U8      | 0005 <sub>h</sub> | UNSIGNED8 |

### 9.1.7.8.5 Object 0434<sub>h</sub>: RT1\_AclTable\_TYPE

|                 |                   |                   |            |
|-----------------|-------------------|-------------------|------------|
| Index           | 0434 <sub>h</sub> | Object Type       | DEFSTRUCT  |
| Name            | RT1_AclTable_TYPE |                   |            |
| Sub-Index       | Component Name    | Value             | Data Type  |
| 00 <sub>h</sub> | NumberOfEntries   | 09 <sub>h</sub>   |            |
| 01 <sub>h</sub> | SrcIp_IPAD        | 0402 <sub>h</sub> | IP_ADDRESS |

|                 |              |                   |             |
|-----------------|--------------|-------------------|-------------|
| 02 <sub>h</sub> | SrcMask_IPAD | 0402 <sub>h</sub> | IP_ADDRESS  |
| 03 <sub>h</sub> | DstIp_IPAD   | 0402 <sub>h</sub> | IP_ADDRESS  |
| 04 <sub>h</sub> | DstMask_IPAD | 0402 <sub>h</sub> | IP_ADDRESS  |
| 05 <sub>h</sub> | Protocol_U8  | 0005 <sub>h</sub> | UNSIGNED8   |
| 06 <sub>h</sub> | SrcPort_U16  | 0007 <sub>h</sub> | UNSIGNED16  |
| 07 <sub>h</sub> | DstPort_U16  | 0007 <sub>h</sub> | UNSIGNED16  |
| 08 <sub>h</sub> | SrcMac_MAC   | 0401 <sub>h</sub> | MAC_ADDRESS |
| 09 <sub>h</sub> | Target_U8    | 0005 <sub>h</sub> | UNSIGNED8   |

## 9.1.8 POWERLINK Router MIB

The POWERLINK Router Management Information Base (MIB) specifies the managed objects which are accessible via SNMP.

## 9.2 Routing Type 2

A POWERLINK Router Type 2 is a coupling element that allows communication between nodes in a POWERLINK network and nodes in an CANopen network.

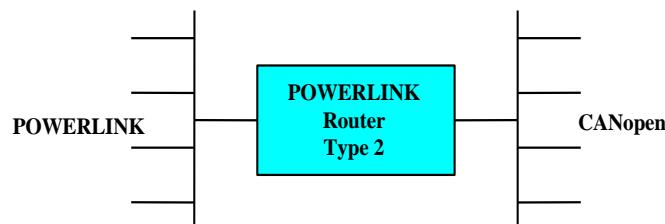


Fig. 113. POWERLINK router type 2

Routing Type 2 will provide CANopen compliant SDO communication between POWERLINK and CANopen nodes.

CANopen to CANopen cross traffic over a POWERLINK based Machine Network (see Fig. 107) will be provided.

Access from the Factory Floor Network and further IP based networks (see Fig. 107) to CANopen nodes via POWERLINK Router Type 1 and POWERLINK Router Type 2 will be possible.

POWERLINK Routing Type 2 will be specified by a separate standard.

Routing Type 2 is optional. Support shall be indicated by D\_RT2\_RT2Support\_BOOL.

## 10 Indicators

Each POWERLINK node shall support:

- either two LEDs
  - a red ERROR LED
  - a green STATUS LED
- or a combination of both using one bicolor (green/red) LED called S/E LED.  
The red subfunction of the S/E LED is equivalent to the ERROR LED, the green one to the STATUS LED.

The following POWERLINK specific LEDs may be additionally used:

- per node
  - Transmit data (TX LED): yellow
- per port:
  - Receive data (RX LED): yellow
  - Ethernet Link (LINK LED): green
  - Collision (COL LED): red (recommended)
  - For the LINK LED and the COL LED, a bicolor (green/red) LED (L/C LED) may be used.

Alternative following POWERLINK specific LED may be additionally used:

- per port:
  - Ethernet Link/Data Activity (LINK/DATA ACTIVITY LED) bicolor (green/yellow) LED or
  - Ethernet Link/Data Activity (LINK/DATA ACTIVITY LED) green LED

### 10.1 Indicator states and flash rates

The following indicator states are distinguished:

|                  |   |
|------------------|---|
| LED on           | constantly on   |
| LED off          | constantly off  |
| LED flickering   | equal on and off times with a frequency of approximately 10 Hz: on for approximately 50 ms and off for approximately 50 ms.   |
| LED blinking     | equal on and off times with a frequency of approximately 2,5 Hz: on for approximately 200 ms followed by off for approximately 200 ms.  |
| LED single flash | one short flash (approximately 200ms) followed by a long off phase (approximately 1000 ms).   |
| LED double flash | a sequence of two short flashes (approximately 200ms), separated by an off phase (approximately 200ms). The sequence is finished by a long off phase (approximately 1000 ms).   |
| LED triple flash | a sequence of three short flashes (approximately 200ms), separated by an off phase (approximately 200ms). The sequence is finished by a long off phase (approximately 1000 ms). |

Tab. 146 LED indicator states

## 10.2 Indicator Signaling

- **ERROR LED**

ERROR LED function is controlled by NMT state machine transitions.

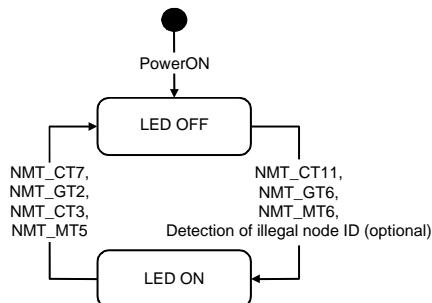


Fig. 114. ERROR LED state machine

In case of detection of an illegal Node ID switching on the error LED is optional.

*Example for an illegal Node ID: Node ID setting 240 on a CN only device.*

- **STATUS LED**

STATUS LED function is controlled by NMT state machine states.

| STATUS LED       | State   |
|------------------|---|
| LED off          | NMT_GS_OFF, NMT_GS_INITIALISATION,<br>NMT_CS_NOT_ACTIVE / NMT_MS_NOT_ACTIVE |
| LED flickering   | NMT_CS_BASIC_ETHERNET   |
| LED single flash | NMT_CS_PRE_OPERATIONAL_1 / NMT_MS_PRE_OPERATIONAL_1                         |
| LED double flash | NMT_CS_PRE_OPERATIONAL_2 / NMT_MS_PRE_OPERATIONAL_2                         |
| LED triple flash | NMT_CS_READY_TO_OPERATE / NMT_MS_READY_TO_OPERATE                           |
| LED on           | NMT_CS_OPERATIONAL / NMT_MS_OPERATIONAL                                     |
| LED blinking     | NMT_CS_STOPPED  |

Tab. 147 Status LED states

- **S/E LED**

combination of ERROR LED and STATUS LED,  
STATUS LED is dominant over the ERROR LED function, e.g. if a STATUS LED flash is required, the ERROR LED will be off during the flash

- **TX LED**

shall be LED on, when data are currently transmitted

- **RX LED**

shall be LED on, when data are currently received

- **LINK LED**

shall be LED on, when etherlink link is established

- **COLLISION LED**

shall be LED on, when an ethernet frame collision is recognized

- **LINK/DATA ACTIVITY LED**

- **Bicolor (yellow/green) LED**

shall be LED on green when ethernet link is established, shall be on yellow when data received or transmitted it is allowed to indicate only the TX- or RX- direction or both, the DATA ACTIVITY LED is dominant over the LINK LED

- **green LED**  
shall be LED on green when ethernet link is established, shall be flash when data received or transmitted it is allowed to indicate only the TX- or RX- direction or both, the DATA ACTIVITY LED is dominant over the LINK LED

## 10.3 Recommended labelling

- „BS“ or „Status“ for the STATUS LED
- „BE“ or „Error“ for the ERROR LED
- „S/E“ for the S/E LED
- „Tx“ for the TX LED
- „Rx“ for the RX LED
- „L“ for the LINK LED
- „C“ for the COLLISION LED
- „L/C“ for the combined LINK LED and COL LED
- „L/A“ or alternatively „LS/DA“ for the LINK/DATA ACTIVITY LED

Case of labelling shall not be significant, e.g. „RUN“, „run“ and „Run“ shall be equivalent.

## App. 1 Summary Object Library (normative)

### App. 1.1 Object Dictionary Entries, sorted by index

| Index  | Name  | Store | Category | Object Type |
|--|---|-------|----------|-------------|
| 0001 <sub>h</sub>                            | BOOLEAN   |       |          | DEFTYPE     |
| 0002 <sub>h</sub>                            | INTEGER8  |       |          | DEFTYPE     |
| 0003 <sub>h</sub>                            | INTEGER16                                       |       |          | DEFTYPE     |
| 0004 <sub>h</sub>                            | INTEGER32                                       |       |          | DEFTYPE     |
| 0005 <sub>h</sub>                            | UNSIGNED8                                       |       |          | DEFTYPE     |
| 0006 <sub>h</sub>                            | UNSIGNED16                                      |       |          | DEFTYPE     |
| 0007 <sub>h</sub>                            | UNSIGNED32                                      |       |          | DEFTYPE     |
| 0008 <sub>h</sub>                            | REAL32  |       |          | DEFTYPE     |
| 0009 <sub>h</sub>                            | VISIBLE_STRING                                  |       |          | DEFTYPE     |
| 000A <sub>h</sub>                            | OCTET_STRING                                    |       |          | DEFTYPE     |
| 000B <sub>h</sub>                            | UNICODE_STRING                                  |       |          | DEFTYPE     |
| 000C <sub>h</sub>                            | TIME_OF_DAY                                     |       |          | DEFTYPE     |
| 000D <sub>h</sub>                            | TIME_DIFFERENCE                                 |       |          | DEFTYPE     |
| 000F <sub>h</sub>                            | DOMAIN  |       |          | DEFTYPE     |
| 0010 <sub>h</sub>                            | INTEGER24                                       |       |          | DEFTYPE     |
| 0011 <sub>h</sub>                            | REAL64  |       |          | DEFTYPE     |
| 0012 <sub>h</sub>                            | INTEGER40                                       |       |          | DEFTYPE     |
| 0013 <sub>h</sub>                            | INTEGER48                                       |       |          | DEFTYPE     |
| 0014 <sub>h</sub>                            | INTEGER56                                       |       |          | DEFTYPE     |
| 0015 <sub>h</sub>                            | INTEGER64                                       |       |          | DEFTYPE     |
| 0016 <sub>h</sub>                            | UNSIGNED24                                      |       |          | DEFTYPE     |
| 0018 <sub>h</sub>                            | UNSIGNED40                                      |       |          | DEFTYPE     |
| 0019 <sub>h</sub>                            | UNSIGNED48                                      |       |          | DEFTYPE     |
| 001A <sub>h</sub>                            | UNSIGNED56                                      |       |          | DEFTYPE     |
| 001B <sub>h</sub>                            | UNSIGNED64                                      |       |          | DEFTYPE     |
| 0023 <sub>h</sub>                            | IDENTITY  |       |          | DEFSTRUCT   |
| 0040 <sub>h</sub><br>..<br>005F <sub>h</sub> | Manufacturer Specific Complex Data Types        |       |          | DEFSTRUCT   |
| 0060 <sub>h</sub><br>..<br>007F <sub>h</sub> | Device Profile (0) Specific Standard Data Types |       |          | DEFTYPE     |
| 0080 <sub>h</sub><br>..<br>009F <sub>h</sub> | Device Profile (0) Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 00A0 <sub>h</sub><br>..<br>00BF <sub>h</sub> | Device Profile 1 Specific Standard Data Types   |       |          | DEFTYPE     |
| 00C0 <sub>h</sub><br>..<br>00DF <sub>h</sub> | Device Profile 1 Specific Complex Data Types    |       |          | DEFSTRUCT   |
| 00E0 <sub>h</sub><br>..<br>00FF <sub>h</sub> | Device Profile 2 Specific Standard Data Types   |       |          | DEFTYPE     |
| 0100 <sub>h</sub><br>..<br>011F <sub>h</sub> | Device Profile 2 Specific Complex Data Types    |       |          | DEFSTRUCT   |

| Index  | Name  | Store | Category | Object Type |
|--|---|-------|----------|-------------|
| 0120 <sub>h</sub><br>..<br>013F <sub>h</sub> | Device Profile 3 Specific Standard Data Types |       |          | DEFTYPE     |
| 0140 <sub>h</sub><br>..<br>015F <sub>h</sub> | Device Profile 3 Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 0160 <sub>h</sub><br>..<br>017F <sub>h</sub> | Device Profile 4 Specific Standard Data Types |       |          | DEFTYPE     |
| 0180 <sub>h</sub><br>..<br>019F <sub>h</sub> | Device Profile 4 Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 01A0 <sub>h</sub><br>..<br>01BF <sub>h</sub> | Device Profile 5 Specific Standard Data Types |       |          | DEFTYPE     |
| 01C0 <sub>h</sub><br>..<br>01DF <sub>h</sub> | Device Profile 5 Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 01E0 <sub>h</sub><br>..<br>01FF <sub>h</sub> | Device Profile 6 Specific Standard Data Types |       |          | DEFTYPE     |
| 0200 <sub>h</sub><br>..<br>021F <sub>h</sub> | Device Profile 6 Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 0220 <sub>h</sub><br>..<br>023F <sub>h</sub> | Device Profile 7 Specific Standard Data Types |       |          | DEFTYPE     |
| 0240 <sub>h</sub><br>..<br>025F <sub>h</sub> | Device Profile 7 Specific Complex Data Types  |       |          | DEFSTRUCT   |
| 0401 <sub>h</sub>                            | MAC_ADDRESS                                   |       |          | DEFTYPE     |
| 0402 <sub>h</sub>                            | IP_ADDRESS                                    |       |          | DEFTYPE     |
| 0403 <sub>h</sub>                            | NETTIME                                       |       |          | DEFTYPE     |
| 0420 <sub>h</sub>                            | PDO_CommParamRecord_TYPE                      |       |          | DEFSTRUCT   |
| 0422 <sub>h</sub>                            | SDO_ParameterRecord_TYPE                      |       |          | DEFSTRUCT   |
| 0424 <sub>h</sub>                            | DLL_ErrorCntRec_TYPE                          |       |          | DEFSTRUCT   |
| 0425 <sub>h</sub>                            | NWL_IpGroup_TYPE                              |       |          | DEFSTRUCT   |
| 0426 <sub>h</sub>                            | NWL_IpAddrTable_TYPE                          |       |          | DEFSTRUCT   |
| 0427 <sub>h</sub>                            | PDL_LocVerApplSw_TYPE                         |       |          | DEFSTRUCT   |
| 0428 <sub>h</sub>                            | INP_ProcessImage_TYPE                         |       |          | DEFSTRUCT   |
| 0429 <sub>h</sub>                            | NMT_ParameterStorage_TYPE                     |       |          | DEFSTRUCT   |
| 042B <sub>h</sub>                            | NMT_InterfaceGroup_Xh_TYPE                    |       |          | DEFSTRUCT   |
| 042C <sub>h</sub>                            | NMT_CycleTiming_TYPE                          |       |          | DEFSTRUCT   |
| 042E <sub>h</sub>                            | NMT_BootTime_TYPE                             |       |          | DEFSTRUCT   |
| 042F <sub>h</sub>                            | NMT_MNCycleTiming_TYPE                        |       |          | DEFSTRUCT   |
| 0430 <sub>h</sub>                            | RT1_EplRouter_TYPE                            |       |          | DEFSTRUCT   |
| 0431 <sub>h</sub>                            | RT1_IpRoutingTable_TYPE                       |       |          | DEFSTRUCT   |
| 0432 <sub>h</sub>                            | RT1_NatTable_TYPE                             |       |          | DEFSTRUCT   |
| 0433 <sub>h</sub>                            | RT1_SecurityGroup_TYPE                        |       |          | DEFSTRUCT   |
| 0434 <sub>h</sub>                            | RT1_AclTable_TYPE                             |       |          | DEFSTRUCT   |
| 0435 <sub>h</sub>                            | CFM_VerifyConfiguration_TYPE                  |       |          | DEFSTRUCT   |
| 0437 <sub>h</sub>                            | DIA_NMTTelegrCount_TYPE                       |       |          | DEFSTRUCT   |

| <b>Index</b>                                 | <b>Name</b>   | <b>Store</b> | <b>Category</b>             | <b>Object Type</b> |
|--|---|--------------|-----------------------------|--------------------|
| 0438 <sub>h</sub>                            | DIA_ERRStatistics_TYPE                                |              |                             | DEFSTRUCT          |
| 0439 <sub>h</sub>                            | NMT_EPLNodeID_TYPE                                    |              |                             | DEFSTRUCT          |
| 043A <sub>h</sub>                            | NMT_RequestCmd_TYPE                                   |              |                             | DEFSTRUCT          |
| 043B <sub>h</sub>                            | DLL_MNRingRedundancy_TYPE used by EPSG DS302-A [1]    |              |                             | DEFSTRUCT          |
| 1000 <sub>h</sub>                            | NMT_DeviceType_U32                                    | -            | M                           | VAR                |
| 1001 <sub>h</sub>                            | ERR_ErrorRegister_U8                                  | -            | M                           | VAR                |
| 1003 <sub>h</sub>                            | ERR_History_ADOM                                      | -            | O                           | ARRAY              |
| 1006 <sub>h</sub>                            | NMT_CycleLen_U32                                      | x            | M                           | VAR                |
| 1008 <sub>h</sub>                            | NMT_ManufactDevName_VS                                | -            | O                           | VAR                |
| 1009 <sub>h</sub>                            | NMT_ManufactHwVers_VS                                 | -            | O                           | VAR                |
| 100A <sub>h</sub>                            | NMT_ManufactSwVers_VS                                 | -            | O                           | VAR                |
| 1010 <sub>h</sub>                            | NMT_StoreParam_REC                                    | -            | O                           | RECORD             |
| 1011 <sub>h</sub>                            | NMT_RestoreDefParam_REC                               | -            | O                           | RECORD             |
| 1016 <sub>h</sub>                            | NMT_ConsumerHeartbeatTime_AU32                        | 1-254        | O                           | ARRAY              |
| 1018 <sub>h</sub>                            | NMT_IdentityObject_REC                                | -            | M                           | RECORD             |
| 1020 <sub>h</sub>                            | CFM_VerifyConfiguration_REC                           | 1-3          | M                           | RECORD             |
| 1021 <sub>h</sub>                            | CFM_StoreDevDescrFile_DOM                             | x            | O                           | VAR                |
| 1022 <sub>h</sub>                            | CFM_StoreDevDescrFormat_U16                           | x            | Cond                        | VAR                |
| 1030 <sub>h</sub><br>..<br>1039 <sub>h</sub> | NMT_InterfaceGroup_Xh_REC                             | 8-9          | M(1030 <sub>h</sub> )/<br>O | RECORD             |
| 1050 <sub>h</sub>                            | DLL_RelativeLatencyDiff_AU32 used by EPSG DS302-C [3] |              | Cond                        | ARRAY              |
| 1101 <sub>h</sub>                            | DIA_NMTTelegrCount_REC                                | -            | O                           | RECORD             |
| 1102 <sub>h</sub>                            | DIA_ERRStatistics_REC                                 | -            | O                           | RECORD             |
| 1200 <sub>h</sub><br>..<br>127F <sub>h</sub> | SDO_ServerContainerParam_XXh_REC                      | 1-4          | O                           | RECORD             |
| 1280 <sub>h</sub><br>..<br>12FF <sub>h</sub> | SDO_ClientContainerParam_XXh_REC                      | 1-4          | O                           | RECORD             |
| 1300 <sub>h</sub>                            | SDO_SequLayerTimeout_U32                              | x            | M                           | VAR                |
| 1301 <sub>h</sub>                            | SDO_CmdLayerTimeout_U32                               | x            | O                           | VAR                |
| 1302 <sub>h</sub>                            | SDO_SequLayerNoAck_U32                                | x            | O                           | VAR                |
| 1400 <sub>h</sub><br>..<br>14FF <sub>h</sub> | PDO_RxCommParam_XXh_REC                               | 1-2          | Cond                        | RECORD             |
| 1600 <sub>h</sub><br>..<br>16FF <sub>h</sub> | PDO_RxMappParam_XXh_AU64                              | 0-254        | Cond                        | ARRAY              |
| 1800 <sub>h</sub><br>..<br>18FF <sub>h</sub> | PDO_TxCommParam_XXh_REC                               | 1-2          | Cond                        | RECORD             |
| 1A00 <sub>h</sub><br>..<br>1AFF <sub>h</sub> | PDO_TxMappParam_XXh_AU64                              | 0-254        | Cond                        | ARRAY              |
| 1B00 <sub>h</sub><br>..<br>1BFF <sub>h</sub> | RT1_AclFwdTable_XXh_REC                               | 1-9          | Cond                        | RECORD             |
| 1C00 <sub>h</sub>                            | DLL_MNCRCError_REC                                    | 3            | MN: M<br>CN: -              | RECORD             |
| 1C01 <sub>h</sub>                            | DLL_MNCollision_REC                                   | 3            | MN: O<br>CN: -              | RECORD             |

| Index  | Name  | Store       | Category                       | Object Type |
|--|---|-------------|--------------------------------|-------------|
| 1C02 <sub>h</sub>                            | DLL_MNCycTimeExceed_REC                           | 3           | MN: O<br>CN: -                 | RECORD      |
| 1C03 <sub>h</sub>                            | DLL_MNLossOfLinkCum_U32                           | -           | MN: Cond<br>CN: -              | VAR         |
| 1C04 <sub>h</sub>                            | DLL_MNCNLatePResCumCnt_AU32                       | -           | MN: Cond<br>CN: -              | ARRAY       |
| 1C05 <sub>h</sub>                            | DLL_MNCNLatePResThrCnt_AU32                       | -           | MN: Cond<br>CN: -              | ARRAY       |
| 1C06 <sub>h</sub>                            | DLL_MNCNLatePResThreshold_AU32                    | 1-254       | MN: Cond<br>CN: -              | ARRAY       |
| 1C07 <sub>h</sub>                            | DLL_MNCNLossPResCumCnt_AU32                       | -           | MN: O<br>CN: -                 | ARRAY       |
| 1C08 <sub>h</sub>                            | DLL_MNCNLossPResThrCnt_AU32                       | -           | MN: M<br>CN: -                 | ARRAY       |
| 1C09 <sub>h</sub>                            | DLL_MNCNLossPResThreshold_AU32                    | 1-254       | MN: M<br>CN: -                 | ARRAY       |
| 1C0A <sub>h</sub>                            | DLL_CNCollision_REC                               | 3           | MN: -<br>CN: O                 | RECORD      |
| 1C0B <sub>h</sub>                            | DLL_CNLossSoC_REC                                 | 3           | MN: -<br>CN: M                 | RECORD      |
| 1C0C <sub>h</sub>                            | DLL_CNLossSoA_REC                                 | 3           | MN: -<br>CN: Cond              | RECORD      |
| 1C0D <sub>h</sub>                            | DLL_CNLossPReq_REC                                | 3           | MN: -<br>CN: Cond              | RECORD      |
| 1C0E <sub>h</sub>                            | DLL_CNSoCJitter_REC                               | 3           | MN: -<br>CN: Cond              | RECORD      |
| 1C0F <sub>h</sub>                            | DLL_CNCRCError_REC                                | 3           | MN: -<br>CN: M                 | RECORD      |
| 1C10 <sub>h</sub>                            | DLL_CNLossOfLinkCum_U32                           | -           | MN: -<br>CN: Cond              | VAR         |
| 1C12 <sub>h</sub>                            | DLL_MNCycleSuspendNumber_U32                      | x           | MN: M<br>CN: -                 | VAR         |
| 1C13 <sub>h</sub>                            | DLL_CNSoCJitterRange_U32                          | x           | MN: -<br>CN: Cond              | VAR         |
| 1C14 <sub>h</sub>                            | DLL_CNLossOfSocTolerance_U32                      | x           | MN: -<br>CN: M                 | VAR         |
| 1C15 <sub>h</sub>                            | DLL_MNLossStatusResCumCnt_AU32                    | -           | MN: O<br>CN: -                 | ARRAY       |
| 1C16 <sub>h</sub>                            | DLL_MNLossStatusResThrCnt_AU32                    | -           | MN: M<br>CN: -                 | ARRAY       |
| 1C17 <sub>h</sub>                            | DLL_MNLossStatusResThreshold_AU32                 | 1-254       | MN: M<br>CN: -                 | ARRAY       |
| 1C40h  | DLL_MNRingRedundancy_REC used by EPSG DS302-A [1] |             | Cond                           | RECORD      |
| 1C80 <sub>h</sub>                            | PDO_ErrMapVers_OSTR                               | -           | O                              | VAR         |
| 1C81 <sub>h</sub>                            | PDO_ErrShort_RX_OSTR                              | -           | O                              | VAR         |
| 1D00 <sub>h</sub><br>..<br>1DFF <sub>h</sub> | RT1_NatTable_XXh_REC                              | 1-4         | Cond                           | RECORD      |
| 1E40 <sub>h</sub><br>..<br>1E49 <sub>h</sub> | NWL_IpAddrTable_Xh_REC                            | (2,3,)<br>5 | Cond<br>(1E40 <sub>h</sub> )/O | RECORD      |
| 1E4A <sub>h</sub>                            | NWL_IpGroup_REC                                   | (1,) 2      | Cond                           | RECORD      |
| 1E80 <sub>h</sub>                            | RT1_EplRouter_REC                                 | 1,2         | Cond                           | RECORD      |

| Index  | Name                            | Store       | Category          | Object Type |
|--|---------------------------------|-------------|-------------------|-------------|
| 1E81 <sub>h</sub>                            | RT1_SecurityGroup_REC           | 1-3         | Cond              | RECORD      |
| 1E90 <sub>h</sub><br>..<br>1ECF <sub>h</sub> | RT1_IpRoutingTable_XXh_REC      | 1-4,<br>6,7 | Cond              | RECORD      |
| 1ED0 <sub>h</sub><br>..<br>1EDF <sub>h</sub> | RT1_AclInTable_Xh_REC           | 1-9         | Cond              | RECORD      |
| 1EE0 <sub>h</sub><br>..<br>1EEF <sub>h</sub> | RT1_AclOutTable_Xh_REC          | 1-9         | Cond              | RECORD      |
| 1F20 <sub>h</sub>                            | CFM_StoreDcfList_ADOM           | 1-254       | Cond              | ARRAY       |
| 1F21 <sub>h</sub>                            | CFM_DcfStorageFormatList_AU8    | 1-254       | O                 | ARRAY       |
| 1F22 <sub>h</sub>                            | CFM_ConciseDcfList_ADOM         | 1-254       | O                 | ARRAY       |
| 1F23 <sub>h</sub>                            | CFM_StoreDevDescrFileList_ADOM  | 1-254       | Cond              | ARRAY       |
| 1F24 <sub>h</sub>                            | CFM_DevDescrFileFormatList_AU8  | 1-254       | O                 | ARRAY       |
| 1F25 <sub>h</sub>                            | CFM_ConfCNRequest_AU32          | -           | O                 | ARRAY       |
| 1F26 <sub>h</sub>                            | CFM_ExpConfDateList_AU32        | 1-254       | O                 | ARRAY       |
| 1F27 <sub>h</sub>                            | CFM_ExpConfTimeList_AU32        | 1-254       | O                 | ARRAY       |
| 1F28 <sub>h</sub>                            | CFM_ExpConfdList_AU32           | 1-254       | O                 | ARRAY       |
| 1F50 <sub>h</sub>                            | PDL_DownloadProgData_ADOM       | -           | O                 | ARRAY       |
| 1F51 <sub>h</sub>                            | PDL_ProgCtrl_AU8                | -           | Cond              | ARRAY       |
| 1F52 <sub>h</sub>                            | PDL_LocVerApplSw_REC            | -           | Cond              | RECORD      |
| 1F53 <sub>h</sub>                            | PDL_MnExpAppSwDateList_AU32     | 0-254       | Cond              | ARRAY       |
| 1F54 <sub>h</sub>                            | PDL_MnExpAppSwTimeList_AU32     | 0-254       | Cond              | ARRAY       |
| 1F70 <sub>h</sub>                            | INP_ProcessImage_REC            | -           | O                 | RECORD      |
| 1F80 <sub>h</sub>                            | NMT_StartUp_U32                 | x           | MN: M<br>CN: -    | VAR         |
| 1F81 <sub>h</sub>                            | NMT_NodeAssignment_AU32         | x           | MN: M<br>CN: Cond | ARRAY       |
| 1F82 <sub>h</sub>                            | NMT_FeatureFlags_U32            | -           | M                 | VAR         |
| 1F83 <sub>h</sub>                            | NMT_EPLVersion_U8               | -           | M                 | VAR         |
| 1F84 <sub>h</sub>                            | NMT_MNDeviceTypeIdList_AU32     | 0-254       | MN: M<br>CN: -    | ARRAY       |
| 1F85 <sub>h</sub>                            | NMT_MNVendorIdList_AU32         | 0-254       | MN: O<br>CN: -    | ARRAY       |
| 1F86 <sub>h</sub>                            | NMT_MNProductCodeList_AU32      | 0-254       | MN: O<br>CN: -    | ARRAY       |
| 1F87 <sub>h</sub>                            | NMT_MNRevisionNoList_AU32       | 0-254       | MN: O<br>CN: -    | ARRAY       |
| 1F88 <sub>h</sub>                            | NMT_MNSerialNoList_AU32         | 0-254       | MN: O<br>CN: -    | ARRAY       |
| 1F89 <sub>h</sub>                            | NMT_BootTime_REC                | 1-9         | MN: M<br>CN: -    | RECORD      |
| 1F8A <sub>h</sub>                            | NMT_MNCycleTiming_REC           | 1,2,4       | MN: M<br>CN: -    | VAR         |
| 1F8B <sub>h</sub>                            | NMT_MNPReqPayloadLimitList_AU16 | 0-254       | MN: M<br>CN: -    | ARRAY       |
| 1F8C <sub>h</sub>                            | NMT_CurrNMTState_U8             | -           | M                 | VAR         |
| 1F8D <sub>h</sub>                            | NMT_PResPayloadLimitList_AU16   | 0-254       | MN: M<br>CN: O    | ARRAY       |
| 1F8E <sub>h</sub>                            | NMT_MNNodeCurrState_AU8         | -           | MN: M<br>CN: -    | ARRAY       |

| <b>Index</b>      | <b>Name</b>   | <b>Store</b> | <b>Category</b> | <b>Object Type</b> |
|-------------------|---|--------------|-----------------|--------------------|
| 1F8F <sub>h</sub> | NMT_MNNodeExpState_AU8                                      | -            | MN: O<br>CN: -  | ARRAY              |
| 1F92 <sub>h</sub> | NMT_MNCNPResTimeout_AU32                                    | 0-254        | MN: M<br>CN: -  | ARRAY              |
| 1F93 <sub>h</sub> | NMT_EPLNodeID_REC   | 3            | M               | RECORD             |
| 1F94 <sub>h</sub> | NMT_PdoNodeAssign_AU8 used by EPSG DS302-D [4]              |              | Cond            | ARRAY              |
| 1F98 <sub>h</sub> | NMT_CycleTiming_REC   | 4,5,<br>7-9  | M               | RECORD             |
| 1F99 <sub>h</sub> | NMT_CNBASICEthernetTimeout_U32                              | x            | MN: -<br>CN: M  | VAR                |
| 1F9A <sub>h</sub> | NMT_HostName_VSTR   | x            | Cond            | VAR                |
| 1F9B <sub>h</sub> | NMT_MultiCycleAssign_AU8                                    | 0-254        | Cond            | ARRAY              |
| 1F9C <sub>h</sub> | NMT_IsochrSlotAssign_AU8                                    | 0-254        | O               | ARRAY              |
| 1F9E <sub>h</sub> | NMT_ResetCmd_U8   | -            | M               | VAR                |
| 1F9F <sub>h</sub> | NMT_RequestCmd_REC  | -            | MN: M<br>CN: -  | RECORD             |
| 1FA0 <sub>h</sub> | NMT_PredecessorNodeNumberList_AU32 used by EPSG DS302-E [5] |              | Cond            | ARRAY              |
| 1FA1 <sub>h</sub> | NMT_PredecessorHubPortList_AU32 used by EPSG DS302-E [5]    |              | Cond            | ARRAY              |

## App. 1.2 Object Dictionary Entries, sorted by name

| Name  | Index  | Object Type |
|---|--|-------------|
| BOOLEAN   | 0001 <sub>h</sub>                            | DEFTYPE     |
| CFM_ConciseDcfList_ADOM                         | 1F22 <sub>h</sub>                            | ARRAY       |
| CFM_ConfCNRequest_AU32                          | 1F25 <sub>h</sub>                            | ARRAY       |
| CFM_DcfStorageFormatList_AU8                    | 1F21 <sub>h</sub>                            | ARRAY       |
| CFM_DevDescrFileFormatList_AU8                  | 1F24 <sub>h</sub>                            | ARRAY       |
| CFM_ExpConfDateList_AU32                        | 1F26 <sub>h</sub>                            | ARRAY       |
| CFM_ExpConfdList_AU32                           | 1F28 <sub>h</sub>                            | ARRAY       |
| CFM_ExpConfTimeList_AU32                        | 1F27 <sub>h</sub>                            | ARRAY       |
| CFM_StoreDcfList_ADOM                           | 1F20 <sub>h</sub>                            | ARRAY       |
| CFM_StoreDevDescrFile_DOM                       | 1021 <sub>h</sub>                            | VAR         |
| CFM_StoreDevDescrFileList_ADOM                  | 1F23 <sub>h</sub>                            | ARRAY       |
| CFM_StoreDevDescrFormat_U16                     | 1022 <sub>h</sub>                            | VAR         |
| CFM_VerifyConfiguration_REC                     | 1020 <sub>h</sub>                            | RECORD      |
| CFM_VerifyConfiguration_TYPE                    | 0435 <sub>h</sub>                            | DEFSTRUCT   |
| Device Profile (0) Specific Complex Data Types  | 0080 <sub>h</sub><br>..<br>009F <sub>h</sub> | DEFSTRUCT   |
| Device Profile (0) Specific Standard Data Types | 0060 <sub>h</sub><br>..<br>007F <sub>h</sub> | DEFTYPE     |
| Device Profile 1 Specific Complex Data Types    | 00C0 <sub>h</sub><br>..<br>00DF <sub>h</sub> | DEFSTRUCT   |
| Device Profile 1 Specific Standard Data Types   | 00A0 <sub>h</sub><br>..<br>00BF <sub>h</sub> | DEFTYPE     |
| Device Profile 2 Specific Complex Data Types    | 0100 <sub>h</sub><br>..<br>011F <sub>h</sub> | DEFSTRUCT   |
| Device Profile 2 Specific Standard Data Types   | 00E0 <sub>h</sub><br>..<br>00FF <sub>h</sub> | DEFTYPE     |
| Device Profile 3 Specific Complex Data Types    | 0140 <sub>h</sub><br>..<br>015F <sub>h</sub> | DEFSTRUCT   |
| Device Profile 3 Specific Standard Data Types   | 0120 <sub>h</sub><br>..<br>013F <sub>h</sub> | DEFTYPE     |
| Device Profile 4 Specific Complex Data Types    | 0180 <sub>h</sub><br>..<br>019F <sub>h</sub> | DEFSTRUCT   |
| Device Profile 4 Specific Standard Data Types   | 0160 <sub>h</sub><br>..<br>017F <sub>h</sub> | DEFTYPE     |
| Device Profile 5 Specific Complex Data Types    | 01C0 <sub>h</sub><br>..<br>01DF <sub>h</sub> | DEFSTRUCT   |
| Device Profile 5 Specific Standard Data Types   | 01A0 <sub>h</sub><br>..<br>01BF <sub>h</sub> | DEFTYPE     |

| Name  | Index  | Object Type |
|---|--|-------------|
| Device Profile 6 Specific Complex Data Types          | 0200 <sub>h</sub><br>..<br>021F <sub>h</sub> | DEFSTRUCT   |
| Device Profile 6 Specific Standard Data Types         | 01E0 <sub>h</sub><br>..<br>01FF <sub>h</sub> | DEFTYPE     |
| Device Profile 7 Specific Complex Data Types          | 0240 <sub>h</sub><br>..<br>025F <sub>h</sub> | DEFSTRUCT   |
| Device Profile 7 Specific Standard Data Types         | 0220 <sub>h</sub><br>..<br>023F <sub>h</sub> | DEFTYPE     |
| DIA_ERRStatistics_REC                                 | 1102 <sub>h</sub>                            | RECORD      |
| DIA_ERRStatistics_TYPE                                | 0438 <sub>h</sub>                            | DEFSTRUCT   |
| DIA_NMTTelegrCount_REC                                | 1101 <sub>h</sub>                            | RECORD      |
| DIA_NMTTelegrCount_TYPE                               | 0437 <sub>h</sub>                            | DEFSTRUCT   |
| DLL_CNCollision_REC                                   | 1C0A <sub>h</sub>                            | RECORD      |
| DLL_CNCRCError_REC                                    | 1C0F <sub>h</sub>                            | RECORD      |
| DLL_CNLossOfLinkCum_U32                               | 1C10 <sub>h</sub>                            | VAR         |
| DLL_CNLossPReq_REC                                    | 1C0D <sub>h</sub>                            | RECORD      |
| DLL_CNLossSoA_REC                                     | 1C0C <sub>h</sub>                            | RECORD      |
| DLL_CNLossSoC_REC                                     | 1C0B <sub>h</sub>                            | RECORD      |
| DLL_CNSoCJitter_REC                                   | 1C0E <sub>h</sub>                            | RECORD      |
| DLL_CNSoCJitterRange_U32                              | 1C13 <sub>h</sub>                            | VAR         |
| DLL_ErrorCntRec_TYPE                                  | 0424 <sub>h</sub>                            | DEFSTRUCT   |
| DLL_CNLossOfSocTolerance_U32                          | 1C14 <sub>h</sub>                            | VAR         |
| DLL_MNCNLatePResCumCnt_AU32                           | 1C04 <sub>h</sub>                            | ARRAY       |
| DLL_MNCNLatePResThrCnt_AU32                           | 1C05 <sub>h</sub>                            | ARRAY       |
| DLL_MNCNLatePResThreshold_AU32                        | 1C06 <sub>h</sub>                            | ARRAY       |
| DLL_MNCNLossPResCumCnt_AU32                           | 1C07 <sub>h</sub>                            | ARRAY       |
| DLL_MNCNLossPResThrCnt_AU32                           | 1C08 <sub>h</sub>                            | ARRAY       |
| DLL_MNCNLossPResThreshold_AU32                        | 1C09 <sub>h</sub>                            | ARRAY       |
| DLL_MNCollision_REC                                   | 1C01 <sub>h</sub>                            | RECORD      |
| DLL_MNCRCError_REC                                    | 1C00 <sub>h</sub>                            | RECORD      |
| DLL_MNCycleSuspendNumber_U32                          | 1C12 <sub>h</sub>                            | VAR         |
| DLL_MNCycTimeExceed_REC                               | 1C02 <sub>h</sub>                            | RECORD      |
| DLL_MNLossOfLinkCum_U32                               | 1C03 <sub>h</sub>                            | VAR         |
| DLL_MNLossStatusResCumCnt_AU32                        | 1C15 <sub>h</sub>                            | ARRAY       |
| DLL_MNLossStatusResThrCnt_AU32                        | 1C16 <sub>h</sub>                            | ARRAY       |
| DLL_MNLossStatusResThreshold_AU32                     | 1C17 <sub>h</sub>                            | ARRAY       |
| DLL_MNRingRedundancy_REC used by EPSG DS302-A [1]     | 1C40 <sub>h</sub>                            | RECORD      |
| DLL_MNRingRedundancy_TYPE used by EPSG DS302-A [1]    | 043B <sub>h</sub>                            | DEFSTRUCT   |
| DLL_RelativeLatencyDiff_AU32 used by EPSG DS302-C [3] | 1050 <sub>h</sub>                            | ARRAY       |
| DOMAIN  | 000F <sub>h</sub>                            | DEFTYPE     |
| ERR_ErrorRegister_U8                                  | 1001 <sub>h</sub>                            | VAR         |
| ERR_History_ADOM                                      | 1003 <sub>h</sub>                            | ARRAY       |
| IDENTITY  | 0023 <sub>h</sub>                            | DEFSTRUCT   |
| INP_ProcessImage_REC                                  | 1F70 <sub>h</sub>                            | RECORD      |
| INP_ProcessImage_TYPE                                 | 0428 <sub>h</sub>                            | DEFSTRUCT   |
| INTEGER16   | 0003 <sub>h</sub>                            | DEFTYPE     |

| Name                                     | Index  | Object Type |
|--|--|-------------|
| INTEGER24                                | 0010 <sub>h</sub>                            | DEFTYPE     |
| INTEGER32                                | 0004 <sub>h</sub>                            | DEFTYPE     |
| INTEGER40                                | 0012 <sub>h</sub>                            | DEFTYPE     |
| INTEGER48                                | 0013 <sub>h</sub>                            | DEFTYPE     |
| INTEGER56                                | 0014 <sub>h</sub>                            | DEFTYPE     |
| INTEGER64                                | 0015 <sub>h</sub>                            | DEFTYPE     |
| INTEGER8                                 | 0002 <sub>h</sub>                            | DEFTYPE     |
| IP_ADDRESS                               | 0402 <sub>h</sub>                            | DEFTYPE     |
| MAC_ADDRESS                              | 0401 <sub>h</sub>                            | DEFTYPE     |
| Manufacturer Specific Complex Data Types | 0040 <sub>h</sub><br>..<br>005F <sub>h</sub> | DEFSTRUCT   |
| NETTIME                                  | 0403 <sub>h</sub>                            | DEFTYPE     |
| NMT_BootTime_REC                         | 1F89 <sub>h</sub>                            | RECORD      |
| NMT_BootTime_TYPE                        | 042E <sub>h</sub>                            | DEFSTRUCT   |
| NMT_CNBasicEthernetTimeout_U32           | 1F99 <sub>h</sub>                            | VAR         |
| NMT_ConsumerHeartbeatTime_AU32           | 1016 <sub>h</sub>                            | ARRAY       |
| NMT_CurrNMTState_U8                      | 1F8C <sub>h</sub>                            | VAR         |
| NMT_CycleLen_U32                         | 1006 <sub>h</sub>                            | VAR         |
| NMT_CycleTiming_REC                      | 1F98 <sub>h</sub>                            | RECORD      |
| NMT_CycleTiming_TYPE                     | 042C <sub>h</sub>                            | DEFSTRUCT   |
| NMT_DeviceType_U32                       | 1000 <sub>h</sub>                            | VAR         |
| NMT_EPLNodeID_REC                        | 1F93 <sub>h</sub>                            | RECORD      |
| NMT_EPLNodeID_TYPE                       | 0439 <sub>h</sub>                            | DEFSTRUCT   |
| NMT_EPLVersion_U8                        | 1F83 <sub>h</sub>                            | VAR         |
| NMT_FeatureFlags_U32                     | 1F82 <sub>h</sub>                            | VAR         |
| NMT_HostName_VSTR                        | 1F9A <sub>h</sub>                            | VAR         |
| NMT_IdentityObject_REC                   | 1018 <sub>h</sub>                            | RECORD      |
| NMT_InterfaceGroup_TYPE                  | 042B <sub>h</sub>                            | DEFSTRUCT   |
| NMT_InterfaceGroup_Xh_REC                | 1030 <sub>h</sub><br>..<br>1039 <sub>h</sub> | RECORD      |
| NMT_IsochrSlotAssign_AU8                 | 1F9C <sub>h</sub>                            | ARRAY       |
| NMT_ManufactDevName_VS                   | 1008 <sub>h</sub>                            | VAR         |
| NMT_ManufactHwVers_VS                    | 1009 <sub>h</sub>                            | VAR         |
| NMT_ManufactSwVers_VS                    | 100A <sub>h</sub>                            | VAR         |
| NMT_MNCNPResTimeout_AU32                 | 1F92 <sub>h</sub>                            | ARRAY       |
| NMT_MNCycleTiming_REC                    | 1F8A <sub>h</sub>                            | VAR         |
| NMT_MNCycleTiming_TYPE                   | 042F <sub>h</sub>                            | DEFSTRUCT   |
| NMT_MNDeviceTypeIdList_AU32              | 1F84 <sub>h</sub>                            | ARRAY       |
| NMT_MNNodeCurrState_AU8                  | 1F8E <sub>h</sub>                            | ARRAY       |
| NMT_MNNodeExpState_AU8                   | 1F8F <sub>h</sub>                            | ARRAY       |
| NMT_MNPReqPayloadLimitList_AU16          | 1F8B <sub>h</sub>                            | ARRAY       |
| NMT_MNProductCodeList_AU32               | 1F86 <sub>h</sub>                            | ARRAY       |
| NMT_MNRevisionNoList_AU32                | 1F87 <sub>h</sub>                            | ARRAY       |
| NMT_MNSerialNoList_AU32                  | 1F88 <sub>h</sub>                            | ARRAY       |
| NMT_MNVendorIdList_AU32                  | 1F85 <sub>h</sub>                            | ARRAY       |
| NMT_MultipICycleAssign_AU8               | 1F9B <sub>h</sub>                            | ARRAY       |

| Name   | Index  | Object Type |
|--|--|-------------|
| NMT_NodeAssignment_AU32                                | 1F81 <sub>h</sub>                            | ARRAY       |
| NMT_ParameterStorage_TYPE                              | 0429 <sub>h</sub>                            | DEFSTRUCT   |
| NMT_PdoNodeAssign_AU8 used by EPSG DS302-D [4]         | 1F94 <sub>h</sub>                            | ARRAY       |
| NMT_PredecessorHubPortList used by EPSG DS302-E [5]    | 1FA0 <sub>h</sub>                            | ARRAY       |
| NMT_PredecessorNodeNumberList used by EPSG DS302-E [5] | 1FA1 <sub>h</sub>                            | ARRAY       |
| NMT_PResPayloadLimitList_AU16                          | 1F8D <sub>h</sub>                            | ARRAY       |
| NMT_RequestCmd_REC                                     | 1F9F <sub>h</sub>                            | RECORD      |
| NMT_RequestCmd_TYPE                                    | 043A <sub>h</sub>                            | DEFSTRUCT   |
| NMT_ResetCmd_U8  | 1F9E <sub>h</sub>                            | VAR         |
| NMT_RestoreDefParam_REC                                | 1011 <sub>h</sub>                            | RECORD      |
| NMT_StartUp_U32  | 1F80 <sub>h</sub>                            | VAR         |
| NMT_StoreParam_REC                                     | 1010 <sub>h</sub>                            | RECORD      |
| NWL_IpAddrTable_TYPE                                   | 0426 <sub>h</sub>                            | DEFSTRUCT   |
| NWL_IpAddrTable_Xh_REC                                 | 1E40 <sub>h</sub><br>..<br>1E49 <sub>h</sub> | RECORD      |
| NWL_IpGroup_REC  | 1E4A <sub>h</sub>                            | RECORD      |
| NWL_IpGroup_TYPE                                       | 0425 <sub>h</sub>                            | DEFSTRUCT   |
| OCTET_STRING   | 000A <sub>h</sub>                            | DEFTYPE     |
| PDL_DownloadProgData_ADOM                              | 1F50 <sub>h</sub>                            | ARRAY       |
| PDL_LocVerApplSw_REC                                   | 1F52 <sub>h</sub>                            | RECORD      |
| PDL_LocVerApplSw_TYPE                                  | 0427 <sub>h</sub>                            | DEFSTRUCT   |
| PDL_MnExpAppSwDateList_AU32                            | 1F53 <sub>h</sub>                            | ARRAY       |
| PDL_MnExpAppSwTimeList_AU32                            | 1F54 <sub>h</sub>                            | ARRAY       |
| PDL_ProgCtrl_AU8                                       | 1F51 <sub>h</sub>                            | ARRAY       |
| PDO_CommParamRecord_TYPE                               | 0420 <sub>h</sub>                            | DEFSTRUCT   |
| PDO_ErrMapVers_OSTR                                    | 1C80 <sub>h</sub>                            | VAR         |
| PDO_ErrShort_RX_OSTR                                   | 1C81 <sub>h</sub>                            | VAR         |
| PDO_RxCommParam_XXh_REC                                | 1400 <sub>h</sub><br>..<br>14FF <sub>h</sub> | RECORD      |
| PDO_RxMappParam_XXh_AU64                               | 1600 <sub>h</sub><br>..<br>16FF <sub>h</sub> | ARRAY       |
| PDO_TxCommParam_XXh_REC                                | 1800 <sub>h</sub><br>..<br>18FF <sub>h</sub> | RECORD      |
| PDO_TxMappParam_XXh_AU64                               | 1A00 <sub>h</sub><br>..<br>1AFF <sub>h</sub> | ARRAY       |
| REAL32   | 0008 <sub>h</sub>                            | DEFTYPE     |
| REAL64   | 0011 <sub>h</sub>                            | DEFTYPE     |
| RT1_AclFwdTable_XXh_REC                                | 1B00 <sub>h</sub><br>..<br>1BFF <sub>h</sub> | RECORD      |
| RT1_AclInTable_Xh_REC                                  | 1ED0 <sub>h</sub><br>..<br>1EDF <sub>h</sub> | RECORD      |
| RT1_AclOutTable_Xh_REC                                 | 1EE0 <sub>h</sub><br>..<br>1EEF <sub>h</sub> | RECORD      |

| Name                             | Index  | Object Type |
|----------------------------------|--|-------------|
| RT1_AclTable_TYPE                | 0434 <sub>h</sub>                            | DEFSTRUCT   |
| RT1_EplRouter_REC                | 1E80 <sub>h</sub>                            | RECORD      |
| RT1_EplRouter_TYPE               | 0430 <sub>h</sub>                            | DEFSTRUCT   |
| RT1_IpRoutingTable_TYPE          | 0431 <sub>h</sub>                            | DEFSTRUCT   |
| RT1_IpRoutingTable_XXh_REC       | 1E90 <sub>h</sub><br>..<br>1ECF <sub>h</sub> | RECORD      |
| RT1_NatTable_TYPE                | 0432 <sub>h</sub>                            | DEFSTRUCT   |
| RT1_NatTable_XXh_REC             | 1D00 <sub>h</sub><br>..<br>1DFF <sub>h</sub> | RECORD      |
| RT1_SecurityGroup_REC            | 1E81 <sub>h</sub>                            | RECORD      |
| RT1_SecurityGroup_TYPE           | 0433 <sub>h</sub>                            | DEFSTRUCT   |
| SDO_ClientContainerParam_XXh_REC | 1280 <sub>h</sub><br>..<br>12FF <sub>h</sub> | RECORD      |
| SDO_CmdLayerTimeout_U32          | 1301 <sub>h</sub>                            | VAR         |
| SDO_ParameterRecord_TYPE         | 0422 <sub>h</sub>                            | DEFSTRUCT   |
| SDO_SequLayerNoAck_U32           | 1302 <sub>h</sub>                            | VAR         |
| SDO_SequLayerTimeout_U32         | 1300 <sub>h</sub>                            | VAR         |
| SDO_ServerContainerParam_XXh_REC | 1200 <sub>h</sub><br>..<br>127F <sub>h</sub> | RECORD      |
| TIME_DIFFERENCE                  | 000D <sub>h</sub>                            | DEFTYPE     |
| TIME_OF_DAY                      | 000C <sub>h</sub>                            | DEFTYPE     |
| UNICODE_STRING                   | 000B <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED16                       | 0006 <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED24                       | 0016 <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED32                       | 0007 <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED40                       | 0018 <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED48                       | 0019 <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED56                       | 001A <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED64                       | 001B <sub>h</sub>                            | DEFTYPE     |
| UNSIGNED8                        | 0005 <sub>h</sub>                            | DEFTYPE     |
| VISIBLE_STRING                   | 0009 <sub>h</sub>                            | DEFTYPE     |

## App. 2 Device Description Entries (normative)

| Name                           | Description  | Type       | Category<br>MN | Category<br>CN | Default<br>MN | Default<br>CN |
|--------------------------------|--|------------|----------------|----------------|---------------|---------------|
| D_CFM_ConfigManager_BOOL       | Ability of a node to perform Configuration Manager functions   | BOOLEAN    | O              | O              | N             | N             |
| D_DLL_CNFeatureMultiplex_BOOL  | node's ability to perform control of multiplexed isochronous communication   | BOOLEAN    | -              | O              | -             | N             |
| D_DLL_ErrBadPhysMode_BOOL      | Support of Data Link Layer Error recognition: Incorrect physical operation mode  | BOOLEAN    | O              | O              | N             | N             |
| D_DLL_ErrMacBuffer_BOOL        | Support of Data Link Layer Error recognition: TX / RX buffer underrun / overrun  | BOOLEAN    | O              | O              | N             | N             |
| D_DLL_ErrMNMultipleMN_BOOL     | Support of MN Data Link Layer Error recognition: Multiple MNs  | BOOLEAN    | O              | -              | N             | -             |
| D_DLL_FeatureCN_BOOL           | node's ability to perform CN functions   | BOOLEAN    | O              | O              | Y             | Y             |
| D_DLL_FeatureMN_BOOL           | node's ability to perform MN functions   | BOOLEAN    | M              | M              | -             | -             |
| D_DLL_MNFeatureMultiplex_BOOL  | MN's ability to perform control of multiplexed isochronous communication   | BOOLEAN    | O              | -              | Y             | -             |
| D_DLL_MNFeaturePResTx_BOOL     | MN's ability to transmit PRes  | BOOLEAN    | O              | -              | Y             | -             |
| D_NMT_BootTimeNotActive_U32    | max. boot time from cold start to NMT_MS_NOT_ACTIVE resp.<br>NMT_CS_NOT_ACTIVE [μs]  | UNSIGNED32 | M              | M              | -             | -             |
| D_NMT_CNPreOp2ToReady2Op_U32   | maximum transition time of a CN from reception of<br>NMTEnableReadyToOperate until the CN is in state<br>NMT_CS_READY_TO_OPERATE [μs]  | UNSIGNED32 | -              | O              | -             | -             |
| D_NMT_CNSoC2PReq_U32           | CN SoC handling maximum time [ns],<br>a subsequent PReq won't be handled before SoC handling was finished  | UNSIGNED32 | -              | M              | -             | -             |
| D_NMT_CycleTimeGranularity_U32 | POWERLINK cycle time granularity [μs]<br>Value shall be 1 μs if POWERLINK cycle time settings may be taken from a continuum. Otherwise granularity should be a multiple of the base granularity values 100 μs or 125 μs. | UNSIGNED32 | O              | O              | 1             | 1             |
| D_NMT_CycleTimeMax_U32         | maximum POWERLINK cycle time [μs]  | UNSIGNED32 | M              | M              | -             | -             |
| D_NMT_CycleTimeMin_U32         | minimum POWERLINK cycle time [μs]  | UNSIGNED32 | M              | M              | -             | -             |
| D_NMT_EmergencyQueueSize_U32   | maximum number of history entries in the Error Signaling emergency queue<br>(see 6.5)  | UNSIGNED32 | O              | O              | 0             | 0             |
| D_NMT_ErrorEntries_U32         | maximum number of error entries (Status and History Entries) in the<br>StatusResponse frame (see 7.3.3.3.1)<br>value range: 2 .. 14  | UNSIGNED32 | M              | M              | -             | -             |
| D_NMT_ExtNmtCmds_BOOL          | Support of Extended NMT State Command  | BOOLEAN    | O              | O              | N             | N             |
| D_NMT_FlushArpEntry_BOOL       | Support of NMT Managing Command Service NMTFlushArpEntry   | BOOLEAN    | O              | O              | N             | N             |
| D_NMT_Isochronous_BOOL         | Device may be accessed isochronously   | BOOLEAN    | O              | O              | Y             | Y             |

|                                |   |            |   |   |     |     |
|--------------------------------|---|------------|---|---|-----|-----|
| D_NMT_MaxCNNodeID_U8           | maximum Node ID available for regular CNs<br>the entry provides an upper limit to the Node ID available for cross traffic PDO reception from a regular CN | UNSIGNED8  | O | O | 239 | 239 |
| D_NMT_MaxCNNumber_U8           | maximum number of supported regular CNs in the Node ID range 1 .. 239   | UNSIGNED8  | O | O | 239 | 239 |
| D_NMT_MaxHeartbeats_U8         | number of guard channels  | UNSIGNED8  | O | O | 254 | 254 |
| D_NMT_MinRedCycleTime_U32      | Minimum reduced cycle time [μs], i.e. minimum time between SoA frames   | UNSIGNED32 | - | O | -   | -   |
| D_NMT_MNASnd2SoC_U32           | minimum delay between end of reception of ASnd and start of transmission of SoC [ns]  | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNBasicEthernet_BOOL     | support of NMT_MS_BASIC_ETHERNET  | BOOLEAN    | O | - | N   | -   |
| D_NMT_MNMultiplCycMax_U8       | maximum number of POWERLINK cycles per multiplexed cycle  | UNSIGNED8  | O | - | 0   | -   |
| D_NMT_MNPRes2PReq_U32          | delay between end of PRes reception and start of PReq transmission [ns]   | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNPRes2PRes_U32          | delay between end of reception of PRes from CNn and start of transmission of PRes by MN [ns]  | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNPResRx2SoA_U32         | delay between end of reception of PRes from CNn and start of transmission of SoA by MN [ns]   | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNPResTx2SoA_U32         | delay between end of PRes transmission by MN and start of transmission of SoA by MN [ns]  | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNSoA2ASndTx_U32         | delay between end of transmission of SoA and start of transmission of ASnd by MN [ns]   | UNSIGNED32 | M | - | -   | -   |
| D_NMT_MNSoC2PReq_U32           | MN minimum delay between end of SoC transmission and start of PReq transmission [ns]  | UNSIGNED32 | M | - | -   | -   |
| D_NMT_NetHostNameSet_BOOL      | Support of NMT Managing Command Service NMTNetHostNameSet   | BOOLEAN    | O | O | N   | N   |
| D_NMT_NetTime_BOOL             | Support of NetTime transmission via SoC   | BOOLEAN    | O | - | N   | -   |
| D_NMT_NetTimelsRealTime_BOOL   | Support of real time via NetTime in SoC   | BOOLEAN    | O | - | N   | -   |
| D_NMT_NodeIDByHW_BOOL          | Ability of a node to support Node ID setup by HW  | BOOLEAN    | O | O | Y   | Y   |
| D_NMT_NodeIDBySW_BOOL          | Ability of a node to support Node ID setup by SW  | BOOLEAN    | O | O | N   | N   |
| D_NMT_ProductCode_U32          | Identity Object Product Code  | UNSIGNED32 | O | O | 0   | 0   |
| D_NMT_PublishActiveNodes_BOOL  | Support of NMT Info service NMTPublishActiveNodes   | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishConfigNodes_BOOL  | Support of NMT Info service NMTPublishConfiguredNodes   | BOOLEAN    | O | O | N   | N   |
| D_NMT_PublishEmergencyNew_BOOL | Support of NMT Info service NMTPublishEmergencyNew  | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishNodeState_BOOL    | Support of NMT Info service NMTPublishNodeStates  | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishOperational_BOOL  | Support of NMT Info service NMTPublishOperational   | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishPreOp1_BOOL       | Support of NMT Info service NMTPublishPreOperational1   | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishPreOp2_BOOL       | Support of NMT Info service NMTPublishPreOperational2   | BOOLEAN    | O | - | N   | -   |
| D_NMT_PublishReadyToOp_BOOL    | Support of NMT Info service NMTPublishReadyToOperate  | BOOLEAN    | O | - | N   | -   |

| Name                          | Description   | Type       | Category |    | Default |         |
|-------------------------------|---|------------|----------|----|---------|---------|
|                               |   |            | MN       | CN | MN      | CN      |
| D_NMT_PublishStopped_BOOL     | Support of NMT Info service NMTPublishStopped   | BOOLEAN    | O        | -  | N       | -       |
| D_NMT_PublishTime_BOOL        | Support of NMT Info service NMTPublishTime  | BOOLEAN    | O        | O  | N       | N       |
| D_NMT_RelativeTime_BOOL       | Support of RelativeTime transmission via SoC  | BOOLEAN    | O        | -  | N       | -       |
| D_NMT_RevisionNo_U32          | Identity Object Revision Number   | UNSIGNED32 | O        | O  | 0       | 0       |
| D_NMT_ServiceUdplp_BOOL       | Support of NMT services via UDP/IP  | BOOLEAN    | O        | -  | N       | -       |
| D_NMT_SimpleBoot_BOOL         | Ability of an MN node to perform only Simple Boot Process, if not set Individual Boot Process shall be provided | BOOLEAN    | M        | -  | -       | -       |
| D_NWL_Forward_BOOL            | Ability of node to forward datagrams  | BOOLEAN    | O        | O  | N       | N       |
| D_NWL_ICMPsupport_BOOL        | Support of ICMP   | BOOLEAN    | O        | O  | N       | N       |
| D_NWL_IPSupport_BOOL          | Ability of the node communicate via IP  | BOOLEAN    | O        | O  | Y       | Y       |
| D PDO_DynamicMapping_BOOL     | Support of dynamic PDO mapping  | BOOLEAN    | O        | O  | Y       | Y       |
| D PDO_Granularity_U8          | minimum size of objects to be mapped [bit]  | UNSIGNED8  | O        | O  | 8       | 8       |
| D PDO_MaxDescrMem_U32         | maximum cumulative memory consumption of TPDO and RPDO mapping describing objects [byte]                        | UNSIGNED32 | O        | O  | MAX_U32 | MAX_U32 |
| D PDO_RPDOChannelObjects_U8   | Number of supported mapped objects per RPDO channel   | UNSIGNED8  | O        | O  | 254     | 254     |
| D PDO_RPDOChannels_U16        | number of supported RPDO channels   | UNSIGNED16 | O        | O  | 256     | 256     |
| D PDO_RPDOCycleDataLim_U32    | maximum sum of data size of RPDO data to be received per cycle [Byte]   | UNSIGNED32 | O        | O  | MAX_U32 | MAX_U32 |
| D PDO_RPDOOverallObjects_U16  | maximum number of mapped RPDO objects, sum of all channels  | UNSIGNED16 | O        | O  | MAX_U16 | MAX_U16 |
| D PDO_SelfReceipt_BOOL        | node's ability to receive PDO data transmitted by itself  | BOOLEAN    | O        | O  | N       | N       |
| D PDO_TPDOChannelObjects_U8   | maximum Number of mapped objects per TPDO channel   | UNSIGNED8  | O        | O  | 254     | 254     |
| D PDO_TPDOChannels_U16        | number of supported TPDO channels   | UNSIGNED16 | O        | -  | 256     | -       |
| D PDO_TPDOCycleDataLim_U32    | maximum sum of data size of TPDO data to be transmitted per cycle [Byte]  | UNSIGNED32 | O        | O  | MAX_U32 | MAX_U32 |
| D PDO_TPDOOverallObjects_U16  | maximum number of mapped RPDO objects, sum of all channels  | UNSIGNED16 | O        | O  | MAX_U16 | MAX_U16 |
| D PHY_ExtEPLPorts_U8          | number of externally accessible Ethernet POWERLINK ports  | UNSIGNED8  | O        | O  | 2       | 2       |
| D PHY_HubDelay_U32            | network propagation delay of the hub integrated in the device in [ns]   | UNSIGNED32 | O        | O  | 1000    | 1000    |
| D PHY_HubIntegrated_BOOL      | indicates a hub integrated by the device  | BOOLEAN    | O        | O  | Y       | Y       |
| D PHY_HubJitter_U32           | jitter of the propagation delay caused by the integrated hub in [ns]  | UNSIGNED32 | O        | O  | 50      | 50      |
| D RT1_RT1SecuritySupport_BOOL | Support of Routing Type 1 security functions  | BOOLEAN    | O        | O  | N       | N       |
| D RT1_RT1Support_BOOL         | Support of Routing Type 1 functions   | BOOLEAN    | O        | O  | N       | N       |
| D RT2_RT2Support_BOOL         | Support of Routing Type 2 functions   | BOOLEAN    | O        | O  | N       | N       |
| D SDO_Client_BOOL             | device implements a SDO client  | BOOLEAN    | O        | O  | Y       | Y       |
| D SDO_CmdFileRead_BOOL        | Support of SDO command FileRead   | BOOLEAN    | O        | O  | N       | N       |

| Name                             | Description   | Type       | Category |    | Default |    |
|----------------------------------|---|------------|----------|----|---------|----|
|                                  |   |            | MN       | CN | MN      | CN |
| D_SDO_CmdFileWrite_BOOL          | Support of SDO command FileWrite  | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdLinkName_BOOL           | Support of SDO command LinkName   | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdReadAllByIndex_BOOL     | Support of SDO command ReadAllByIndex                                     | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdReadByName_BOOL         | Support of SDO command ReadByName   | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdReadMultParam_BOOL      | Support of SDO command ReadMultipleParam                                  | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdWriteAllByIndex_BOOL    | Support of SDO command WriteAllByIndex                                    | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdWriteByName_BOOL        | Support of SDO command WriteByName  | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_CmdWriteMultParam_BOOL     | Support of SDO command WriteMultParam                                     | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_MaxConnections_U32         | max. number of SDO connections  | UNSIGNED32 | O        | O  | 1       | 1  |
| D_SDO_MaxParallelConnections_U32 | max. number of SDO connections between a SDO client/server pair           | UNSIGNED32 | O        | O  | 1       | 1  |
| D_SDO_SeqLayerTxHistorySize_U16  | max. number of frames in SDO sequence layer sender history<br>value <= 31 | UNSIGNED16 | O        | O  | 5       | 5  |
| D_SDO_Server_BOOL                | device implements a SDO server  | BOOLEAN    | O        | O  | Y       | Y  |
| D_SDO_SupportASnd_BOOL           | Support of SDO via ASnd frames  | BOOLEAN    | M        | O  | Y       | N  |
| D_SDO_SupportPDO_BOOL            | Support of SDO via PDO frames   | BOOLEAN    | O        | O  | N       | N  |
| D_SDO_SupportUdplp_BOOL          | Support of SDO via UDP/IP frames  | BOOLEAN    | M        | O  | Y       | N  |

Abbreviations in the table:

- M device description entry is mandatory
- O device description entry is optional
- - item is irrelevant for the selected type of node (MN or CN), item may be provided if the device supports the other type, too

### Comments to Default

Default values shall be used if an optional entry is not provided by the device description. If no default is indicated, zero value resp. empty string shall be applied.

Device description entry names are created according to the following rules:

Name shall be in accordance to IEC 61131-3. It consists of:

- Device description entry identification prefix "D\_"
- domain prefix, indicating the association of the object to a functional domain, 3 uppercase characters followed by underline
- name (verbally). Composed of words, each word are leaded by an uppercase character followed by lowercase characters or digits, no underlines or spaces
- data type postfix, indicating the data type of the object (underline followed by up to 5 uppercase characters or digits)

## App. 3 Constant Value Assignments (normative)

### App. 3.1 POWERLINK Message Type Ids

| Message Type  | Abbr. | ID Value        |
|---|-------|-----------------|
| Start of Cycle  | SoC   | 01 <sub>h</sub> |
| PollRequest   | PReq  | 03 <sub>h</sub> |
| PollResponse  | PRes  | 04 <sub>h</sub> |
| Start of Asynchronous                                     | SoA   | 05 <sub>h</sub> |
| Asynchronous Send   | ASnd  | 06 <sub>h</sub> |
| Active Managing Node Indication, used by EPSG DS302-A [1] | AMNI  | 07 <sub>h</sub> |
| Asynchronous Invite, used by EPSG DS302-B [2]             | Ainv  | 0D <sub>h</sub> |

*ID values not listed by the table are reserved.*

### App. 3.2 AsyncSend Request Priorities

| Priority Level | Priority Name                     | ID value   |
|----------------|-----------------------------------|--|
| highest        | PRIO_NMT_REQUEST                  | 111 <sub>b</sub>                                       |
| higher         | available for application purpose | 110 <sub>b</sub> , 101 <sub>b</sub> , 100 <sub>b</sub> |
| medium         | PRIO_GENERIC_REQUEST              | 011 <sub>b</sub>                                       |
| lower          | available for application purpose | 010 <sub>b</sub> , 001 <sub>b</sub> , 000 <sub>b</sub> |

### App. 3.3 ASnd ServiceIDs

| Service Name                       | Service ID      | ID Value                           |
|------------------------------------|-----------------|------------------------------------|
| reserved                           |                 | 00h                                |
| IdentResponse                      | IDENT_RESPONSE  | 01 <sub>h</sub>                    |
| StatusResponse                     | STATUS_RESPONSE | 02 <sub>h</sub>                    |
| NMTRequest                         | NMT_REQUEST     | 03 <sub>h</sub>                    |
| NMTCommand                         | NMT_COMMAND     | 04 <sub>h</sub>                    |
| SDO                                | SDO             | 05 <sub>h</sub>                    |
| reserved, used by EPSG DS302-C [3] |                 | 06 <sub>h</sub>                    |
| reserved                           |                 | 07 <sub>h</sub> .. 9F <sub>h</sub> |
| Manufacturer specific              | MANUF_SVC_IDS   | A0 <sub>h</sub> .. FE <sub>h</sub> |
| reserved                           |                 | FF <sub>h</sub>                    |

## App. 3.4 SoA RequestedServiceIDs

| Service Name                       | Service ID         | ID Value                           |
|------------------------------------|--------------------|------------------------------------|
| NoService                          | NO_SERVICE         | 00 <sub>h</sub>                    |
| IdentRequest                       | IDENT_REQUEST      | 01 <sub>h</sub>                    |
| StatusRequest                      | STATUS_REQUEST     | 02 <sub>h</sub>                    |
| NMTRequestInvite                   | NMT_REQUEST_INVITE | 03 <sub>h</sub>                    |
| reserved                           |                    | 04 <sub>h</sub> .. 05 <sub>h</sub> |
| reserved, used by EPSG DS302-C [3] |                    | 06 <sub>h</sub>                    |
| reserved                           |                    | 07 <sub>h</sub> .. 9F <sub>h</sub> |
| Manufacturer specific              | MANUF_SVC_IDS      | A0 <sub>h</sub> .. FE <sub>h</sub> |
| UnspecifiedInvite                  | UNSPECIFIED_INVITE | FF <sub>h</sub>                    |

## App. 3.5 Object Dictionary Object Types

|           |                   |
|-----------|-------------------|
| NULL      | 0000 <sub>h</sub> |
| DEFTYPE   | 0005 <sub>h</sub> |
| DEFSTRUCT | 0006 <sub>h</sub> |
| VAR       | 0007 <sub>h</sub> |
| ARRAY     | 0008 <sub>h</sub> |
| RECORD    | 0009 <sub>h</sub> |

Values not listed by the table are reserved.

## App. 3.6 NMT States

|                  | Name                       | Value                  |             |
|------------------|----------------------------|------------------------|-------------|
| MN and CN States | NMT_GS_OFF                 | 0000 0000 <sub>b</sub> |             |
|                  | NMT_GS_POWERED             | xxxx 1xxx <sub>b</sub> | Super state |
|                  | NMT_GS_INITIALISATION      | xxxx 1001 <sub>b</sub> | Super state |
|                  | NMT_GS_INITIALISING        | 0001 1001 <sub>b</sub> |             |
|                  | NMT_GS_RESET_APPLICATION   | 0010 1001 <sub>b</sub> |             |
|                  | NMT_GS_RESET_COMMUNICATION | 0011 1001 <sub>b</sub> |             |
|                  | NMT_GS_RESET_CONFIGURATION | 0111 1001 <sub>b</sub> |             |
|                  | NMT_GS_COMMUNICATING       | xxxx 11xx <sub>b</sub> | Super state |
| CN States        | NMT_CS_NOT_ACTIVE          | 0001 1100 <sub>b</sub> |             |
|                  | NMT_CS_EPL_MODE            | xxxx 1101 <sub>b</sub> | Super state |
|                  | NMT_CS_PRE_OPERATIONAL_1   | 0001 1101 <sub>b</sub> |             |
|                  | NMT_CS_PRE_OPERATIONAL_2   | 0101 1101 <sub>b</sub> |             |
|                  | NMT_CS_READY_TO_OPERATE    | 0110 1101 <sub>b</sub> |             |
|                  | NMT_CS_OPERATIONAL         | 1111 1101 <sub>b</sub> |             |
|                  | NMT_CS_STOPPED             | 0100 1101 <sub>b</sub> |             |
|                  | NMT_CS_BASIC_ETHERNET      | 0001 1110 <sub>b</sub> |             |
| MN States        | NMT_MS_NOT_ACTIVE          | 0001 1100 <sub>b</sub> |             |
|                  | NMT_MS_EPL_MODE            | xxxx 1101 <sub>b</sub> | Super state |
|                  | NMT_MS_PRE_OPERATIONAL_1   | 0001 1101 <sub>b</sub> |             |
|                  | NMT_MS_PRE_OPERATIONAL_2   | 0101 1101 <sub>b</sub> |             |
|                  | NMT_MS_READY_TO_OPERATE    | 0110 1101 <sub>b</sub> |             |
|                  | NMT_MS_OPERATIONAL         | 1111 1101 <sub>b</sub> |             |
|                  | NMT_MS_BASIC_ETHERNET      | 0001 1110 <sub>b</sub> |             |

NMT\_CS\_OPERATIONAL and NMT\_MS\_OPERATIONAL can be easily identified by the most significant bit being set.

## App. 3.7 NMT Commands

| Name                                    | ID Value                           |
|---|------------------------------------|
| requestable ASnd ServiceIDs             | 01 <sub>h</sub> .. 1F <sub>h</sub> |
| IdentResponse                           | 01 <sub>h</sub>                    |
| StatusResponse                          | 02 <sub>h</sub>                    |
| Plain NMT State Commands                | 20 <sub>h</sub> .. 3F <sub>h</sub> |
| NMTStartNode                            | 21 <sub>h</sub>                    |
| NMTStopNode                             | 22 <sub>h</sub>                    |
| NMTEnterPreOperational2                 | 23 <sub>h</sub>                    |
| NMTEnableReadyToOperate                 | 24 <sub>h</sub>                    |
| NMTResetNode                            | 28 <sub>h</sub>                    |
| NMTResetCommunication                   | 29 <sub>h</sub>                    |
| NMTResetConfiguration                   | 2A <sub>h</sub>                    |
| NMTSwReset                              | 2B <sub>h</sub>                    |
| NMTGoToStandby used by EPSG DS302-A [1] | 2C <sub>h</sub>                    |
| Extended NMT State Commands             | 40 <sub>h</sub> .. 5F <sub>h</sub> |
| NMTStartNodeEx                          | 41 <sub>h</sub>                    |
| NMTStopNodeEx                           | 42 <sub>h</sub>                    |
| NMTEnterPreOperational2Ex               | 43 <sub>h</sub>                    |
| NMTEnableReadyToOperateEx               | 44 <sub>h</sub>                    |
| NMTResetNodeEx                          | 48 <sub>h</sub>                    |
| NMTResetCommunicationEx                 | 49 <sub>h</sub>                    |
| NMTResetConfigurationEx                 | 4A <sub>h</sub>                    |
| NMTSwResetEx                            | 4B <sub>h</sub>                    |
| NMT Managing Commands                   | 60 <sub>h</sub> .. 7F <sub>h</sub> |
| NMTNetHostNameSet                       | 62 <sub>h</sub>                    |
| NMTFlushArpEntry                        | 63 <sub>h</sub>                    |
| NMT Info services                       | 80 <sub>h</sub> .. BF <sub>h</sub> |
| NMTPublishConfiguredNodes               | 80 <sub>h</sub>                    |
| NMTPublishActiveNodes                   | 90 <sub>h</sub>                    |
| NMTPublishPreOperational1               | 91 <sub>h</sub>                    |
| NMTPublishPreOperational2               | 92 <sub>h</sub>                    |
| NMTPublishReadyToOperate                | 93 <sub>h</sub>                    |
| NMTPublishOperational                   | 94 <sub>h</sub>                    |
| NMTPublishStopped                       | 95 <sub>h</sub>                    |
| NMTPublishNodeStates                    | 96 <sub>h</sub>                    |
| NMTPublishEmergencyNew                  | A0 <sub>h</sub>                    |
| NMTPublishTime                          | B0 <sub>h</sub>                    |
| NMTInvalidService                       | FF <sub>h</sub>                    |

*ID values not listed as distinctive commands are reserved*

## App. 3.8 General Purpose Constants

| Name                      | Value             | Unit | Description   |
|---------------------------|-------------------|------|---|
| C_ADR_BROADCAST           | FF <sub>h</sub>   | --   | POWERLINK broadcast address   |
| C_ADR_DIAG_DEF_NODE_ID    | FD <sub>h</sub>   | --   | POWERLINK default address of diagnostic device  |
| C_ADR_DUMMY_NODE_ID       | FC <sub>h</sub>   | --   | POWERLINK dummy node address  |
| C_ADR_SELF_ADR_NODE_ID    | FB <sub>h</sub>   | --   | POWERLINK pseudo node address to be used by a node to address itself                        |
| C_ADR_INVALID             | 00 <sub>h</sub>   | --   | invalid POWERLINK address   |
| C_ADR_MN_DEF_NODE_ID      | F0 <sub>h</sub>   | --   | POWERLINK default address of MN   |
| C_ADR_RT1_DEF_NODE_ID     | FE <sub>h</sub>   | --   | POWERLINK default address of router type 1  |
| C_DLL_ASND_PRIO_NMTRQST   | 7                 | --   | ASnd request priority to be used by NMT Requests  |
| C_DLL_ASND_PRIO_STD       | 0                 | --   | standard ASnd request priority  |
| C_DLL_ETHERTYPE_EPL       | 88AB <sub>h</sub> | --   |   |
| C_DLL_ISOCHR_MAX_PAYL     | 1490              | Byte | maximum size of PReq and PRes payload data, requires C_DLL_MAX_ASYNC_MTU                    |
| C_DLL_MAX_ASYNC_MTU       | 1500              | Byte | maximum asynchronous payload in bytes including all headers (exclusive the Ethernet header) |
| C_DLL_MAX_PAYL_OFFSET     | 1499              | Byte | maximum offset of Ethernet frame payload, requires C_DLL_MAX_ASYNC_MTU                      |
| C_DLL_MAX_RS              | 7                 | --   |   |
| C_DLL_MIN_ASYNC_MTU       | 300               | Byte | minimum asynchronous payload in bytes including all headers (exclusive the Ethernet header) |
| C_DLL_MIN_PAYL_OFFSET     | 45                | Byte | minimum offset of Ethernet frame payload  |
| C_DLL_MULTICAST_AMNI      | 01-11-1E-00-00-05 | --   | POWERLINK Active Managing Node Indication, canonical form. Used by EPSG DS302-A [1]         |
| C_DLL_MULTICAST_ASND      | 01-11-1E-00-00-04 | --   | POWERLINK ASnd multicast MAC address, canonical form  |
| C_DLL_MULTICAST_PRES      | 01-11-1E-00-00-02 | --   | POWERLINK PRes multicast MAC address, canonical form  |
| C_DLL_MULTICAST_SOA       | 01-11-1E-00-00-03 | --   | POWERLINK SoA multicast MAC address, canonical form   |
| C_DLL_MULTICAST_SOC       | 01-11-1E-00-00-01 | --   | POWERLINK Soc multicast MAC address, canonical form   |
| C_DLL_PREOP1_START_CYCLES | 10                | --   | number of unassigning SoA frames at start of NMT_MS_PRE_OPERATIONAL_1                       |
| C_DLL_T_BITTIME           | 10                | ns   | Transmission time per bit on 100 Mbit/s network   |
| C_DLL_T_EPL_PDO_HEADER    | 10                | Byte | size of PReq and PRes POWERLINK PDO message header (see 4.6.1.1.3 and 4.6.1.1.4)            |
| C_DLL_T_ETH2_WRAPPER      | 18                | Byte | size of Ethernet type II wrapper consisting of header and checksum (see 4.6.1.1.1)          |
| C_DLL_T_IFG               | 960               | ns   | Ethernet Interframe Gap   |
| C_DLL_T_MIN_FRAME         | 5120              | ns   | Size of minimum Ethernet frame (without preamble and start-of-frame-delimiter)              |
| C_DLL_T_PREAMBLE          | 640               | ns   | Size of Ethernet frame preamble plus start-of-frame-delimiter                               |
| C_ERR_MONITOR_DELAY       | 10                |      | Error monitoring start delay  |

| Name                  | Value                    | Unit   | Description   |
|-----------------------|--------------------------|--------|---|
| C_IP_ADR_INVALID      | 00 00 00 00 <sub>h</sub> | --     | invalid IP address (0.0.0.0) used to indicate no change |
| C_IP_INVALID_MTU      | 0                        | Byte   | invalid MTU size used to indicate no change             |
| C_NMT_STATE_TOLERANCE | 5                        | Cycles | maximum reaction time to NMT state commands             |
| C_NMT_STATREQ_CYCLE   | 5                        | sec    | StatusRequest cycle time to be applied to AsyncOnly CNs |
| C_SDO_EPL_PORT        | 3819                     | --     | port to be used POWERLINK specific UDP/IP frames        |
| C_SDO_CMDLAYERTIMEOUT | 30000                    | ms     | Command layer timeout                                   |
| C_SDO_SEQLAYERNOACK   | 2                        | --     | Number of acknowledge requests                          |
| C_SDO_SEQLAYERTIMEOUT | 15000                    | ms     | Sequence layer timeout                                  |

General Purpose Constant names are created according to the following rules:

Name shall be in accordance to IEC 61131-3. It consists of:

- General Purpose Constant identification prefix “C\_”
- domain prefix, indicating the association of the object to a functional domain,  
3 uppercase characters followed by underline
- name (verbally).  
composed of words, words are separated by underline, uppercase characters only, no spaces

## App. 3.9 Error Code Constants

| Name                   | Value             | Description       |
|------------------------|-------------------|-------------------|
|                        | 816X <sub>h</sub> | HW errors         |
| E_DLL_BAD_PHYS_MODE    | 8161 <sub>h</sub> |                   |
| E_DLL_COLLISION        | 8162 <sub>h</sub> |                   |
| E_DLL_COLLISION_TH     | 8163 <sub>h</sub> |                   |
| E_DLL_CRC_TH           | 8164 <sub>h</sub> |                   |
| E_DLL_LOSS_OF_LINK     | 8165 <sub>h</sub> |                   |
| E_DLL_MAC_BUFFER       | 8166 <sub>h</sub> |                   |
|                        | 82XX <sub>h</sub> | Protocol errors   |
| E_DLL_ADDRESS_CONFLICT | 8201 <sub>h</sub> |                   |
| E_DLL_MULTIPLE_MN      | 8202 <sub>h</sub> |                   |
|                        | 821X <sub>h</sub> | Frame size errors |
| E_PDO_SHORT_RX         | 8210 <sub>h</sub> |                   |
| E_PDO_MAP_VERS         | 8211 <sub>h</sub> |                   |
| E_NMT_ASND_MTU_DIF     | 8212 <sub>h</sub> |                   |
| E_NMT_ASND_MTU_LIM     | 8213 <sub>h</sub> |                   |
| E_NMT_ASND_TX_LIM      | 8214 <sub>h</sub> |                   |
|                        | 823X <sub>h</sub> | Timing errors     |
| E_NMT_CYCLE_LEN        | 8231 <sub>h</sub> |                   |
| E_DLL_CYCLE_EXCEED     | 8232 <sub>h</sub> |                   |
| E_DLL_CYCLE_EXCEED_TH  | 8233 <sub>h</sub> |                   |
| E_NMT_IDLE_LIM         | 8234 <sub>h</sub> |                   |
| E_DLL_JITTER_TH        | 8235 <sub>h</sub> |                   |
| E_DLL_LATE_PRES_TH     | 8236 <sub>h</sub> |                   |
| E_NMT_PREQ_CN          | 8237 <sub>h</sub> |                   |
| E_NMT_PREQ_LIM         | 8238 <sub>h</sub> |                   |
| E_NMT_PRES_CN          | 8239 <sub>h</sub> |                   |
| E_NMT_PRES_RX_LIM      | 823A <sub>h</sub> |                   |
| E_NMT_PRES_TX_LIM      | 823B <sub>h</sub> |                   |

|                         |                   |                |
|-------------------------|-------------------|----------------|
|                         | 824X <sub>h</sub> | Frame errors   |
| E_DLL_INVALID_FORMAT    | 8241 <sub>h</sub> |                |
| E_DLL LOSS_PREQ_TH      | 8242 <sub>h</sub> |                |
| E_DLL LOSS_PRES_TH      | 8243 <sub>h</sub> |                |
| E_DLL LOSS_SOA_TH       | 8244 <sub>h</sub> |                |
| E_DLL LOSS_SOC_TH       | 8245 <sub>h</sub> |                |
| E_DLL LOSS_STATUSRES_TH | 8246 <sub>h</sub> |                |
|                         | 84X <sub>h</sub>  | Boot-up Errors |
| E_NMT_BA1               | 8410 <sub>h</sub> |                |
| E_NMT_BA1_NO_MN_SUPPORT | 8411 <sub>h</sub> |                |
| E_NMT_BPO1              | 8420 <sub>h</sub> |                |
| E_NMT_BPO1_GET_IDENT    | 8421 <sub>h</sub> |                |
| E_NMT_BPO1_DEVICE_TYPE  | 8422 <sub>h</sub> |                |
| E_NMT_BPO1_VENDOR_ID    | 8423 <sub>h</sub> |                |
| E_NMT_BPO1_PRODUCT_CODE | 8424 <sub>h</sub> |                |
| E_NMT_BPO1_REVISION_NO  | 8425 <sub>h</sub> |                |
| E_NMT_BPO1_SERIAL_NO    | 8426 <sub>h</sub> |                |
| E_NMT_BPO1_CF_VERIFY    | 8428 <sub>h</sub> |                |
| E_NMT_BPO1_SW_INVALID   | 8429 <sub>h</sub> |                |
| E_NMT_BPO1_SW_STATE     | 842A <sub>h</sub> |                |
| E_NMT_BPO1_SW_UPDATE    | 842B <sub>h</sub> |                |
| E_NMT_BPO1_SW_REJECT    | 842C <sub>h</sub> |                |
| E_NMT_BPO2              | 8430 <sub>h</sub> |                |
| E_NMT_BRO               | 8440 <sub>h</sub> |                |
| E_NMT_WRONG_STATE       | 8480 <sub>h</sub> |                |

*Error Code Constant names are created according to the following rules:*

*Name shall be in accordance to IEC 61131-3. It consists of:*

- *Error Code Constant identification prefix “E\_”*
- *domain prefix, indicating the association of the object to a functional domain, 3 uppercase characters followed by underline*
- *name (verbally), composed of words, words are separated by underline, uppercase characters only, no spaces*

## App. 3.10 SDO Abort Codes

| Name                    | Abort code             | Description   |
|-------------------------|------------------------|---|
|                         | 0503 0000 <sub>h</sub> | reserved  |
|                         | 0504 0000 <sub>h</sub> | SDO protocol timed out.   |
|                         | 0504 0001 <sub>h</sub> | Client/server Command ID not valid or unknown.  |
|                         | 0504 0002 <sub>h</sub> | Invalid block size.   |
|                         | 0504 0003 <sub>h</sub> | Invalid sequence number.  |
|                         | 0504 0004 <sub>h</sub> | reserved  |
|                         | 0504 0005 <sub>h</sub> | Out of memory.  |
|                         | 0601 0000 <sub>h</sub> | Unsupported access to an object.  |
|                         | 0601 0001 <sub>h</sub> | Attempt to read a write-only object.  |
|                         | 0601 0002 <sub>h</sub> | Attempt to write a read-only object.  |
|                         | 0602 0000 <sub>h</sub> | Object does not exist in the object dictionary.   |
|                         | 0604 0041 <sub>h</sub> | Object cannot be mapped to the PDO.   |
| E_PDO_MAP_OVERRUN       | 0604 0042 <sub>h</sub> | The number and length of the objects to be mapped would exceed PDO length.  |
|                         | 0604 0043 <sub>h</sub> | General parameter incompatibility.  |
| E_NMT_INVALID_HEARTBEAT | 0604 0044 <sub>h</sub> | Invalid heartbeat declaration   |
|                         | 0604 0047 <sub>h</sub> | General internal incompatibility in the device.   |
|                         | 0606 0000 <sub>h</sub> | Access failed due to an hardware error.   |
|                         | 0607 0010 <sub>h</sub> | Data type does not match, length of service parameter does not match  |
|                         | 0607 0012 <sub>h</sub> | Data type does not match, length of service parameter too high  |
|                         | 0607 0013 <sub>h</sub> | Data type does not match, length of service parameter too low   |
|                         | 0609 0011 <sub>h</sub> | Sub-index does not exist.   |
|                         | 0609 0030 <sub>h</sub> | Value range of parameter exceeded (only for write access).  |
|                         | 0609 0031 <sub>h</sub> | Value of parameter written too high.  |
|                         | 0609 0032 <sub>h</sub> | Value of parameter written too low.   |
|                         | 0609 0036 <sub>h</sub> | Maximum value is less than minimum value.   |
|                         | 0800 0000 <sub>h</sub> | General error   |
|                         | 0800 0020 <sub>h</sub> | Data cannot be transferred or stored to the application.  |
|                         | 0800 0021 <sub>h</sub> | Data cannot be transferred or stored to the application because of local control.   |
|                         | 0800 0022 <sub>h</sub> | Data cannot be transferred or stored to the application because of the present device state.  |
|                         | 0800 0023 <sub>h</sub> | Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of a file error). |
| E_CFM_DATA_SET_EMPTY    | 0800 0024 <sub>h</sub> | EDS, DCF or Concise DCF Data set empty  |

The abort codes not listed here are reserved.

## App. 4 Data Sheet Requirements (normative)

The hardware used to implement POWERLINK devices may be spread over a broad range of amount of resources as well as calculation power. These differences result in highly different reaction times and buffer sizes affecting the POWERLINK communication timing.

In order to give a compact overview to the project engineer, each manufacturer of a POWERLINK device shall provide a short data sheet displaying critical device properties.

- For CN devices, the data sheet shall provide the following parameters:
  - POWERLINK cycle time
    - minimum value, if device supports adjustment over a continuous range
    - list of values, if device supports discrete values only
  - size of isochronous transmit buffer (maximal size of isochronous frames)
  - size of isochronous receive buffer (maximal size of isochronous frames)
  - overall buffer size available for isochronous data
  - PReq to PRes latency (CN isochronous reaction time)
  - SoA to ASnd latency (CN asynchronous reaction time)
  - maximum asynchronous MTU
  - ability to support multiplexed isochronous access
  - asynchronous SDO transfer method (UDP/IP and/or POWERLINK ASnd)
- For MN devices, the data sheet shall provide the following parameters:
  - POWERLINK cycle time
    - minimum value, if device supports adjustment over a continuous range
    - list of values, if device supports discrete values only
  - size of isochronous transmit buffer (maximal size of isochronous frames)
  - size of isochronous receive buffer (maximal size of isochronous frames)
  - overall buffer size available for isochronous data
  - minimum transmit-to-transmit gap (controls sequence of MN frame transmission)
  - minimum receive-to-transmit gap (controls sequence of MN frame transmission)
  - maximum asynchronous MTU
  - ability to support multiplexed isochronous access

For device that support CN and MN, both parameter lists shall be independently provided.

**End of File**