PUBLICLY
AVAILABLE
SPECIFICATION

IEC
PAS 62408

First edition
2005-06

Real-time Ethernet Powerlink (EPL)

Reference number
IEC/PAS 62408:2005(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site (www.iec.ch)**

- **Catalogue of IEC publications**

  The on-line catalogue on the IEC web site (www.iec.ch/searchpub) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

  This summary of recently issued publications (www.iec.ch/online_news/ justpub) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

  If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

  Email: custserv@iec.ch
  Tel: +41 22 919 02 11
  Fax: +41 22 919 03 00

PUBLICLY AVAILABLE SPECIFICATION

IEC PAS 62408

First edition
2005-06

Real-time Ethernet Powerlink (EPL)

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE XH

For price, see current catalogue

CONTENT

Tables

Figures

INTERNATIONAL ELECTROTECHNICAL COMMISSION

REAL-TIME ETHERNET POWERLINK (EPL)

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

A PAS is a technical specification not fulfilling the requirements for a standard but made available to the public.

IEC-PAS 62408 has been processed by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

The text of this PAS is based on the following document:	This PAS was approved for publication by the P-members of the committee concerned as indicated in the following document
Draft PAS	**Report on voting**
65C/356A/NP	65C/372A/RVN

Following publication of this PAS, the technical committee or subcommittee concerned will transform it into an International Standard.

It is intended that the content of this PAS will be incorporated in the future new edition of the various parts of IEC 61158 series according to the structure of this series.

This PAS shall remain valid for an initial maximum period of three years starting from 2005-06. The validity may be extended for a single three-year period, following which it shall be revised to become another type of normative document or shall be withdrawn.

1 General

1.1 Scope and general description

ETHERNET Powerlink (EPL) is a communication profile for Real-Time Ethernet (RTE). It extends Ethernet according to the IEEE 802.3/ISO/IEC 8802-3 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.

1.1.1 Slot Communication Network Management (SCNM)

EPL 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. Synchronize networked nodes with high accuracy.
3. Transmit less time-critical data asynchronously on request. Data exchange is based on a point-to-point relationship. Asynchronous data communication can be used to transfer IP-based protocols like TCP or UDP and higher layer protocols such as HTTP, FTP,…

EPL 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).

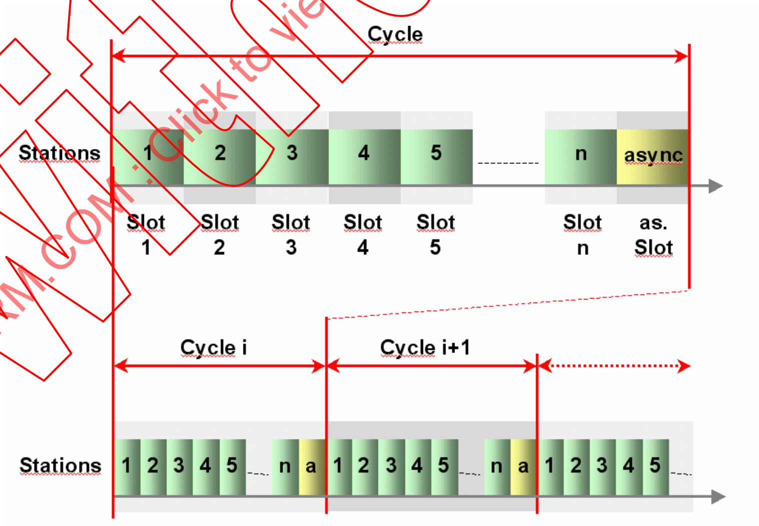


Figure 1 – Slot communication network management (SCNM)

1.1.2 EPL key features

EPL 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
    - IAONA Real-time class 4 (highest performance)
    - Down to 200µs cycle times
    - Ultra-low jitter (down to <1µs) for precise sycnhronisation 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

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

The ETHERNET Powerlink Standardization Group (EPSG) is working closely with the CiA (CAN in Automation) organisation to integrate CANopen with EPL. 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 EPL with CANopen combines profiles, high performance data exchange and open transparent communication with TCP/UDP/IP protocols.

The EPL 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 an EPL environment without changes.

A main focus of EPL 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

1.1.3 Integration

The advantages listed before result from protecting the EPL 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 EPL. 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 EPL 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.

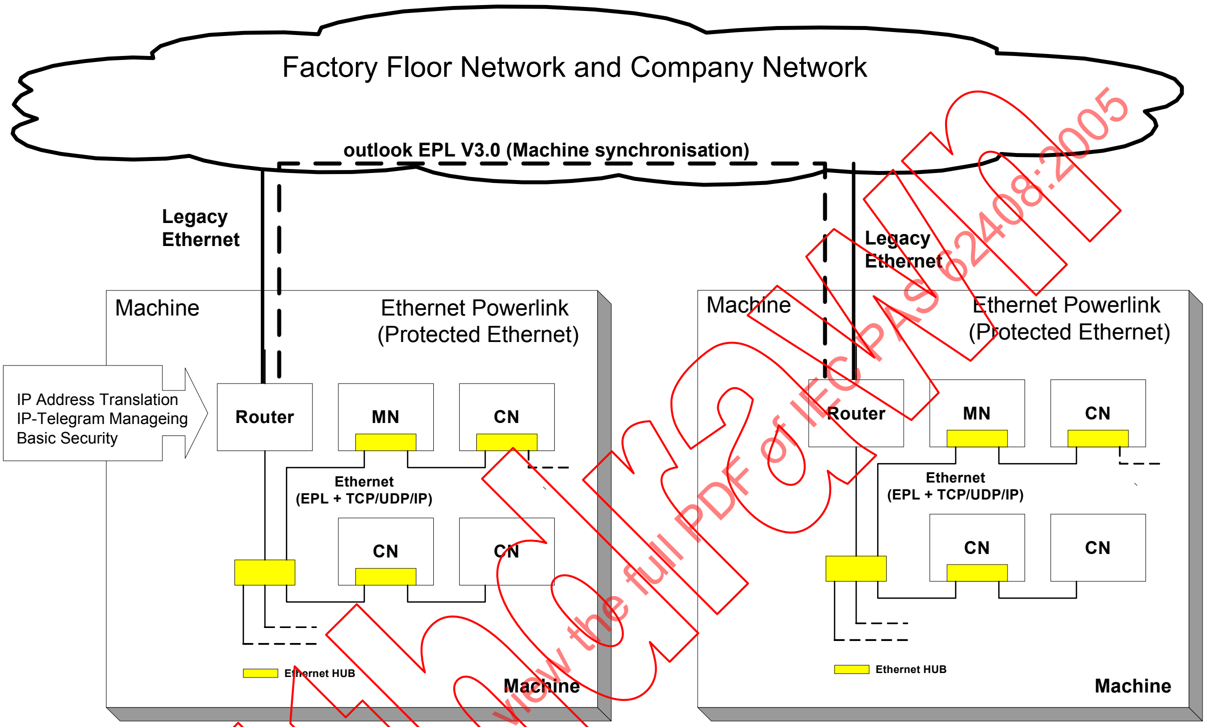


Figure 2 – Integration EPL based machines into the IT infrastructure of end customer

RTE based on EPL 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.

1.1.4 Modular Machines

A machine concept with autonomous machine modules is illustrated below. Every machine module can be designed separetly whith 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.

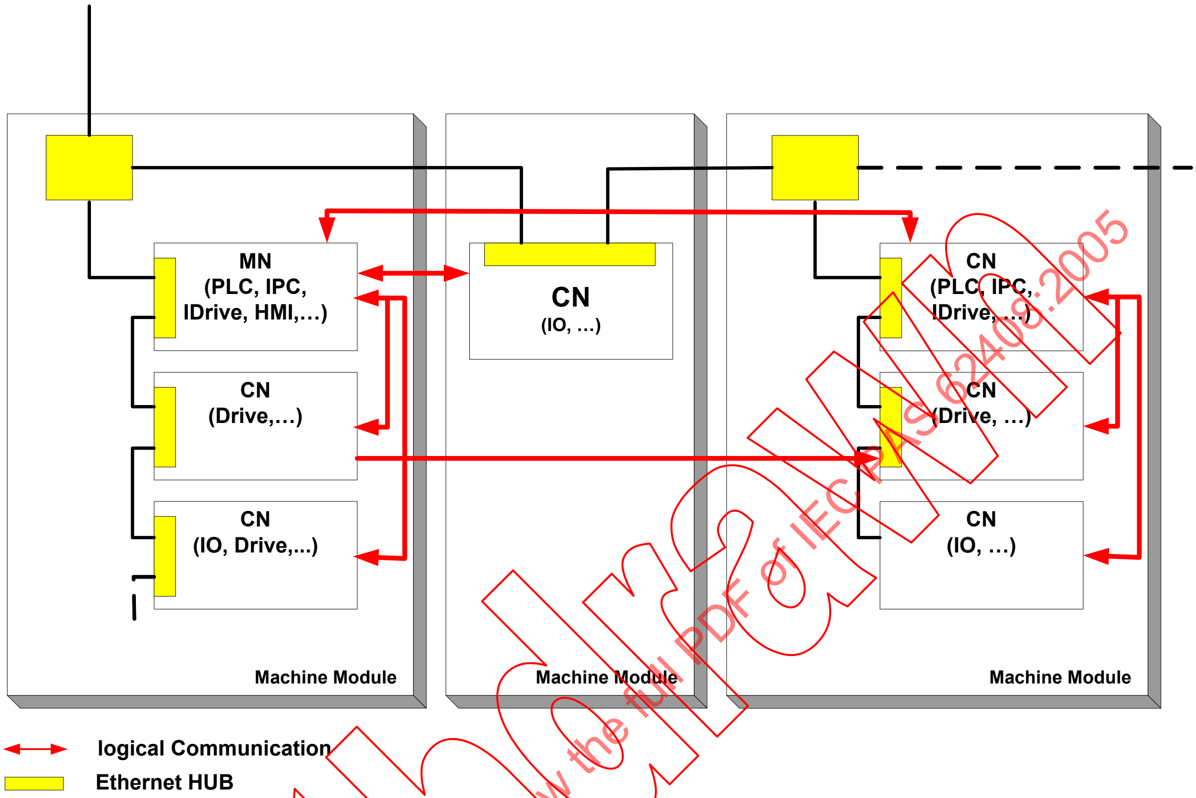


Figure 3 – Typical centralized and decentralized controller structures

1.2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

IEC 61076-2-101:2004, *Connector for electronic equipment – Part 2-101: Circular connectors – Detail specification for circular connectors M8 with scre- or snap-loking, M12 with scre-locking for low-voltage aplications*

IEC 61131-3, *Programmable controllers – Part 3: Programming languages*

IEEE 802.3:1998
ISO/IEC 8802-3:2000 *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

IEEE 754:1985, *IEEE standard for binary floating-point arithmetic*

IEEE 802.1Q:199, *Virtual Bridged Local Area Networks*

IEEE 802.1X:2001, *Port-Based Network Access Control Virtual Private Network (VPN) Server, Intrusion Detection*

IEEE 1588
IEC 61588 *Precision clock synchronization protocol for networked measurement and control systems*

EN 50325-4/CIA DS301 *CANopen Application Layer and Communication Profile*

EIA/TIA T568B: USOC, *RJ45 Pinout Wiring Diagram*

IETF RFC 768 *UDP*

IETF RFC 791 *IP*

IETF RFC 793 *TCP*

IETF RFC 826 *Address Resolution Protocol (ARP)*

IETF RFC1122 *Requirements for Internet Hosts -- Communication Layers*

IETF RFC1213 *Management Information Base for Network Management of TCP/IP-based internets:MIB-II*

IETF RFC1354 *IP Forwarding Table MIB*

IETF RFC 1631 *The IP Network Address Translator (NAT)*

IETF RFC 1812 *Requirements for IP Version 4 Routers*

IETF RFC1918 *Address Allocation for Private Internets*

IETF RFC 3410-3418 SNMPv3

IETF RFC 1354 *IP Forwarding* Table MIB

IETF RFC 2663 *IP Network Address Translator (NAT) Terminology and Considerations*

IETF RFC 2993 *Architectural Implications of NAT*

IETF RFC 3022 *Traditional IP Network Address Translator (Traditional NAT)*

IETF RFC 3027 *Protocol Complications with the IP Network Address Translator*

1.3 Definitions and Abbreviations

1.3.1 Definitions

Ageing	*Ageing* 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 it the task on the Application Layer
Async-only CN	An *Async-only CN* 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 an EPL network which is not time critical. Within the EPL cycle there is a specific period reserved for *Asynchronous Data* 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 *Asynchronous Period* is the second part of the EPL cycle, starting with a *Start of Asynchronous* (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	*Continuous* is an EPL communication class where isochronous communication takes place every cycle (the opposite to *multiplexed*).
Controlled Node (CN)	Node in an EPL network without the abilty to manage the SCNM mechanism.
Cycle State Machine	The *Cycle State Machine* controls the EPL 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 *Start of Cyclic* (SoC) frames – i.e. repeating – process. The *Cycle Time* 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	In the Device Configuration file (DCF) is stored the configuration parameters of a specific device.
Device Description File	All device dependent information's are stored in the Device Description File DDF) of each device.
D-NAT Destination NAT	D-NAT (Destination- Network Address Translation) changes the destination address of the IP / ICMP packet.
Domain	In the context of CANopen: A *Domain* is a data object of arbitrary type and length which can be transferred over an EPL network. In the context of internet protocols: A *Domain* is a part of the internet name space which is supported by the Domain Name System (DNS).
EPL Command Layer	The *EPL Command Layer* defines commands to access parameters of the object dictionary. This layer is on top of the *Sequence Layer* and distinguishes between an expedited and a block transfer.

EPL Cycle	Data exchange within an EPL 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.
EPL Mode	The *EPL Mode* includes all NMT states in which EPL cycles are run.
EPL Node ID	Each EPL node (MN, CN and Router) is addressed by an 8 bit *EPL Node ID* on the EPL layer. This ID has only local significance (i.e. it is unique within an EPL segment).
ETHERNET Powerlink (EPL)	An extension to *Legacy Ethernet* 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	*IdentRequests* are EPL frames sent by the MN in order to identify active CNs waiting to be included into the network.
IdentResponse	The *IdentResponse* is a special form of an ASnd frame in response to an *IdentRequest*.
Idle Period	The *Idle Period* is time interval remaining between the completed asynchronous period and the beginning of the next cycle.
IEEE 1588/IEC 61588	This standard defines a protocol enabling synchronization 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 continously and at a steady rate in close timing with the ability of connected devices.
Isochronous Data	Data in an EPL network which is to be transmitted every cycle (or every nth cycle in case of multiplexed isochronous data).
Isochronous Period	The *Isochronous Period* of an EPL cycle offers deterministic operation, i.e. it is reserved for the exchange of (*continuous* or *multiplexed*) 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 an EPL network.
Media Access Control (MAC)	One of the sub-layers of the Data Link Layer in the EPL reference model that controls who gets access to the medium to send a message.
Multiplexed	*Multiplexed* is an EPL communication class where cyclic communication takes place in such a way that m nodes are served in s cycles (the opposite to *continuous*).
Multiplexed CN	A node which is allowed to send isochronous data every n^{th} 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)	*Network Management* functions and services in the EPL model. It performs initialisation, configuration and error handling in an EPL network.
NMT State Machine	The state machine controlling the overall operating mode and status of an EPL node.
Object Directory	The repository of all data objects accessible over EPL communications.
Open Mode	The *Open Mode* provides isochronous communication and asynchronous communication. The nodes are synchronized by the Precision Time Protocol (PTP), specified by IEEE 1588.
PollRequest	A PollRequest is 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 frame, which is used in the isochronous part of the cyclic communication. The CN responses with this frame to a PollRequest frame from a MN.
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 EPL nodes.
EPL Mode	The *EPL Mode* provides isochronous communication and asynchronous communication. The nodes are synchronized with a dedicated EPL frame, which has an extremely low jitter.
Router Type 1	A Type 1 EPL Router is a coupling element in a network that allows IP communication between an EPL segment and any other datalink layer protocol carrying IP e.g. legacy Ethernet, EPL etc. It is usually a separate network element acting as Controlled Node within the EPL segment.
Router Type 2	A Type 2 EPL Router is a router between an EPL 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 EPL *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 an EPL 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 Isochronous Data, as well as for Asynchronous Data 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 *Legacy Ethernet*.
S-NAT Source NAT	S-NAT (Source - Network Address Translation) changes the source address of the IP / ICMP packet.
StatusRequest / StatusResponse	StatusRequest frames are used to poll *Async-only CNs*.A *StatusResponse* frame is transmitted by an CN upon assignment of the asynchronous slot via the *StatusRequest* in the SoA frame.
StatusResponse	A *StatusResponse* frame is transmitted by an CN upon assignment of the asynchronous slot via the *StatusRequest* in the SoA frame.

1.3.2 Abbreviations

ACL	Access Control List
ARP	Address Resolution Protocol
ASnd	Asynchronous Send (EPL frame type)
CAN	Controller Area Network
CiA	CAN in Automation
CN	EPL Controlled Node
DCF	Device Configuration File
DDF	Device Description File
EDS	Electronic Data Sheet
EIA	Electronic Industries Association
EMC	Electro Magnetic Compatibility
EPL	ETHERNET Powerlink
EPSG	ETHERNET Powerlink Standardization Group
IAONA	Industrial Automation Open Networking Alliance
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Comission
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
MAC	Media Access Control
MIB	Management Information Base
MN	EPL Managing Node
MS	Multiplexed Slot (flag in EPL frame)
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 EPL frame)
PReq	PollRequest (EPL frame type)
PRes	PollResponse (EPL frame type)
PS	Prescaled Slot (flag in EPL frame)
PTP	Precision Time Protocol
RD	Ready (flag in EPL frame)
RFC	Requests for Comments
RPDO	Receive Process Data Object
RS	Request to Send (flag in EPL frame)
S/UTP	Screened Unshielded Twisted Pair
EA	Exception Achnowledge (flag in EPL frame)
SCNM	Slot Communication Network Management
SDO	Service Data Object
EN	Exception New (flag in EPL frame)
SNMP	Simple Network Management Protocol
SoA	Start of Asynchronous (EPL frame type)
SoC	Start of Cyclic (EPL frame type)
TCP	Transmission Control Protocol
TIA	Telecommunications Industry Association
TPDO	Transmit Process Data Object
UDP	User Datagram Protocol
UTP	Unshielded Twisted Pair
VPN	Virtual Private Network

2 Modelling

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

2.1 Reference Model

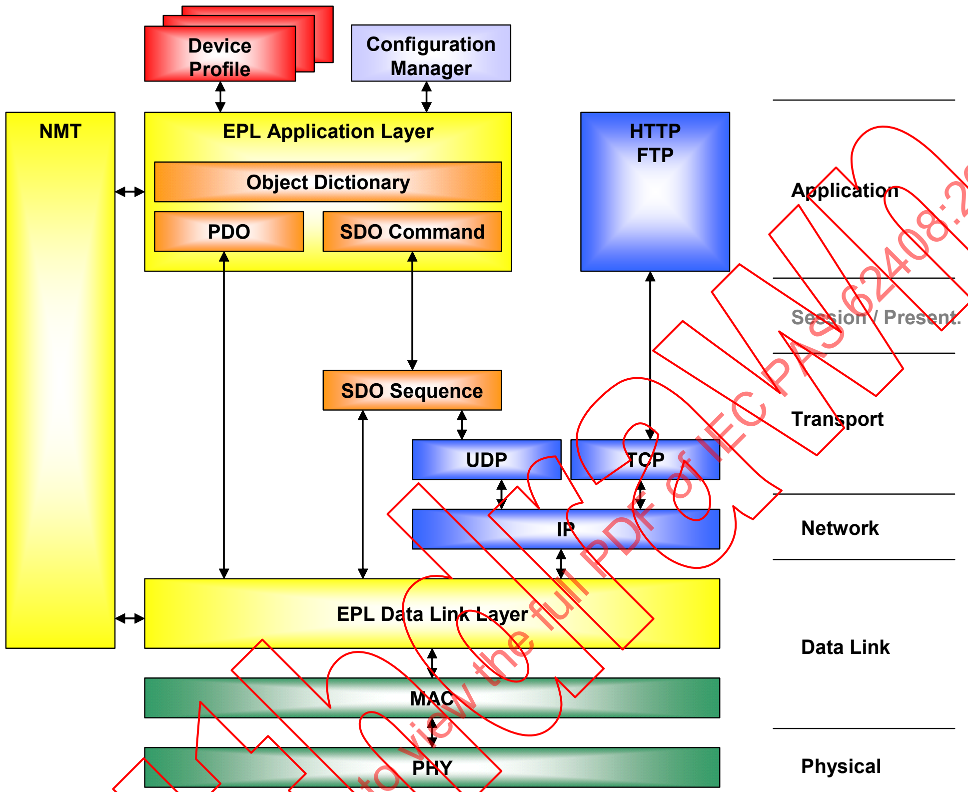


Figure 4 – Reference Model

The communication concept can be described with reference to the ISO-OSI Reference Model (right side of Figure 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 synchronization 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 the 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

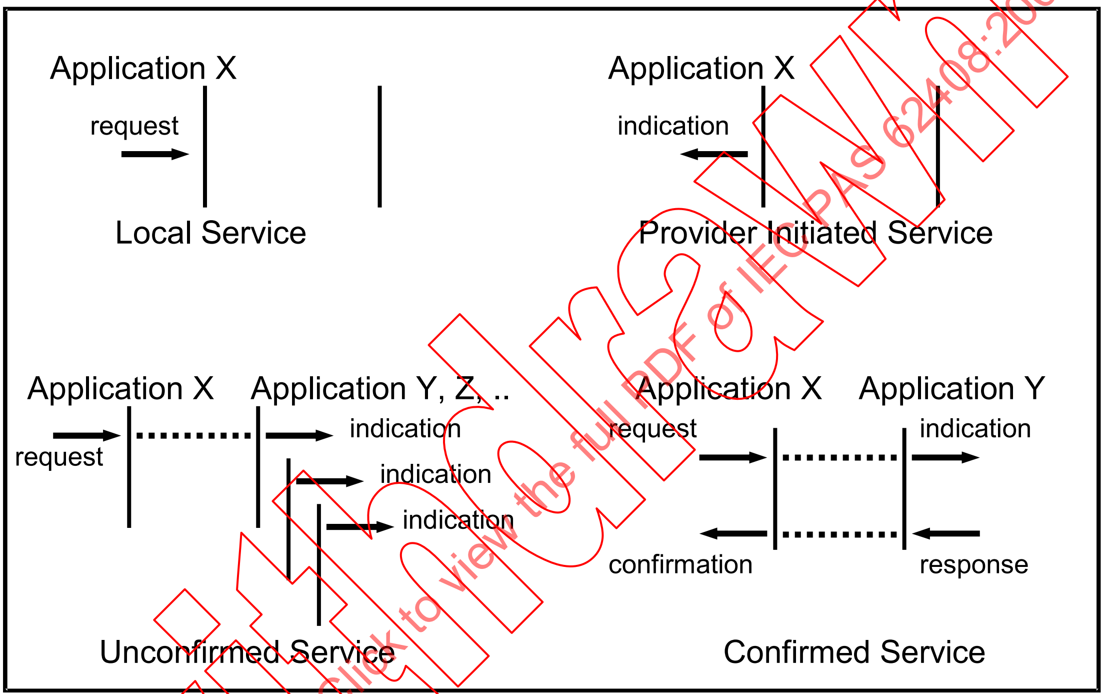


Figure 5 – Service Types

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 (seeFigure 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.*

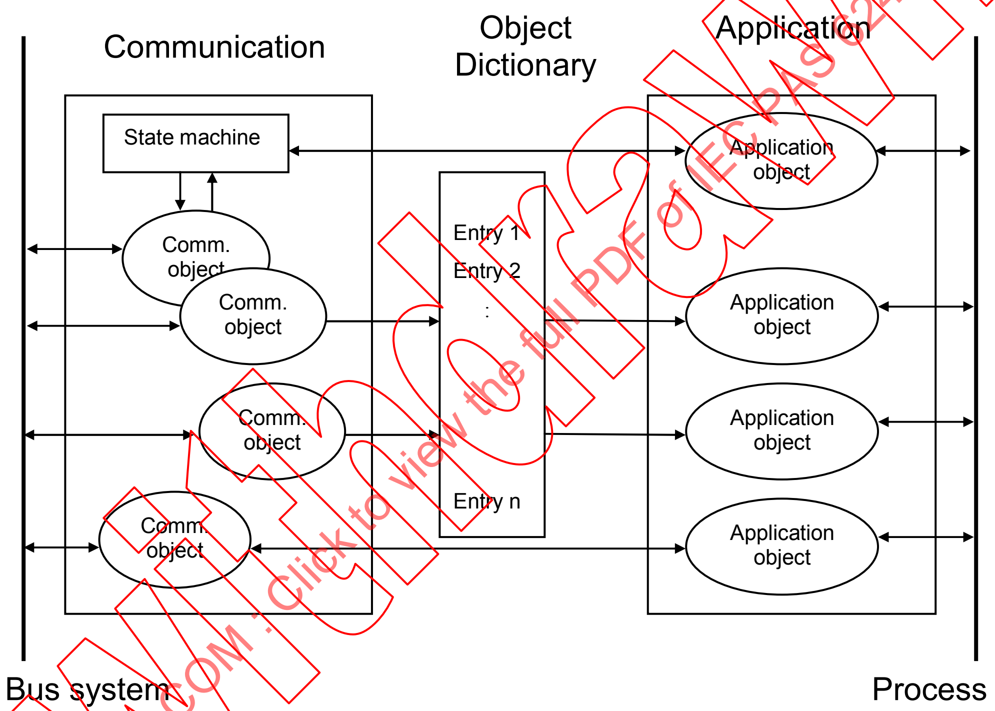


Figure 6 – Device Model

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 below. This layout closely conforms to other industrial serial bus system concepts:

Table 1 – Object Dictionary Structure

Index	Object
0000_h	not used
0001_h - $001F_h$	Static Data Types
0020_h - $003F_h$	Complex Data Types
0040_h - $005F_h$	Manufacturer Specific Complex Data Types
0060_h - $007F_h$	Device Profile Specific Static Data Types
0080_h - $009F_h$	Device Profile Specific Complex Data Types
$00A0_h$ - $03FF_h$	Reserved for further use
0400_h – $041F_h$	EPL Specific Static Data Types
0420_h – $04FF_h$	EPL Specific Complex Data Types
0500_h - $0FFF_h$	Reserved for further use
1000_h - $1FFF_h$	Communication Profile Area
2000_h - $5FFF_h$	Manufacturer Specific Profile Area
6000_h - $9FFF_h$	Standardised Device Profile Area
$A000_h$ - $BFFF_h$	Standardised Interface Profile Area
$C000_h$ - $FFFF_h$	Reserved for further use

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_h through $001F_h$ 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.

Complex Data Types at indices 0020_h through $003F_h$ 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_h through $005F_h$ 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_h - $007F_h$, the complex data types at indices 0080_h - $009F_h$.

A device may optionally provide the structure of the supported complex data types (indices 0020_h - $005F_h$ and 0080_h - $009F_h$) 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.

EPL Specific Static Data Types shall be described at indices 0400_h – $041F_h$. These entries are included for reference only; they cannot be read or written. EPL Specific Complex Data Types shall be described at indices 0420_h – $04FF_h$

The Communication Profile Area at indices 1000_h through $1FFF_h$ contains the communication specific parameters for the EPL network. These entries are common to all devices.

The standardised device profile area at indices 6000_h through $9FFF_h$ 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_h to $9FFF_h$ 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_h.

For Multiple Device Modules the object range 6000_h to $67FF_h$ is sub-divided as follows:

- 6000_h to $67FF_h$ device 0
- 6800_h to $6FFF_h$ device 1
- 7000_h to $77FF_h$ device 2
- 7800_h to $7FFF_h$ device 3
- 8000_h to $87FF_h$ device 4
- 8800_h to $8FFF_h$ device 5
- 9000_h to $97FF_h$ device 6
- 9800_h to $9FFF_h$ device 7

The PDO distribution shall be used for every segment of a Multiple Device Module with an offset of 64_d, e.g. the first PDO of the second segment gets the number 65_d. In this way a system with a maximum of 8 segments is supported.

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_h through $5FFF_h$ 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 EPL-specific communication objects of the EPL Mode and Basic Ethernet Mode. (4.2 resp. 4.3). The mechanism for Standard Ethernet communicationin 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 EPL Protected Mode only, asynchronous frames in EPL Protected 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 EPL Protected Mode cycle structure. The system is synchronised with SoC frames.. Asynchronous frames may be transmitted in the asynchronous slot of EPL Protected Mode cycle upon transmission grant by the EPL MN, or at any time in Basic Ethernet Mode.

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

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

2.3.1 Master/Slave relationship

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.

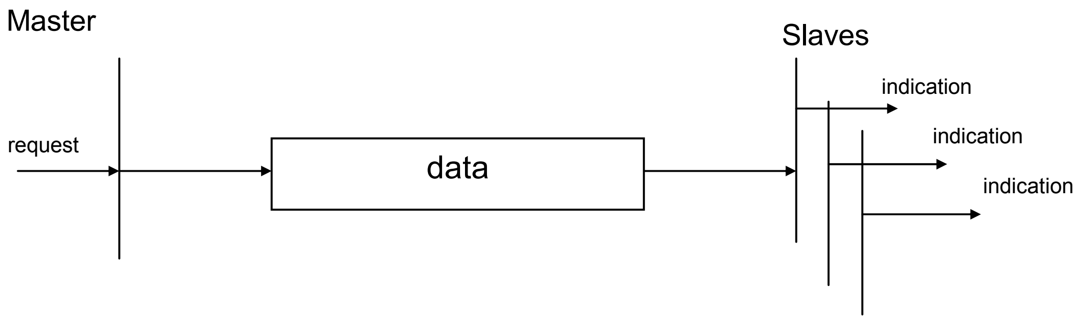


Figure 7 – Unconfirmed Master Slave Communication

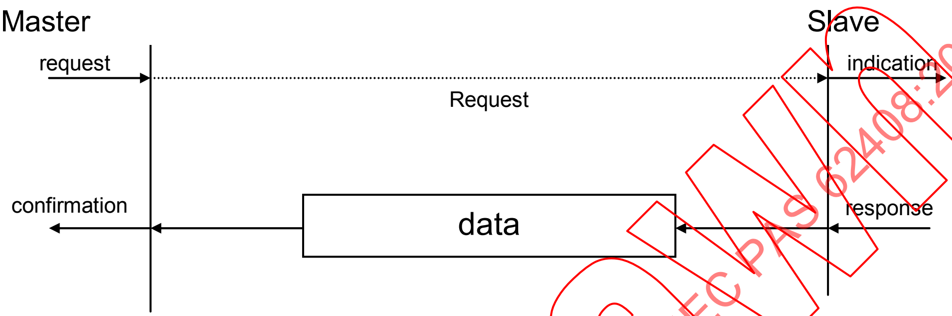


Figure 8 – Confirmed Master Slave Communication

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.

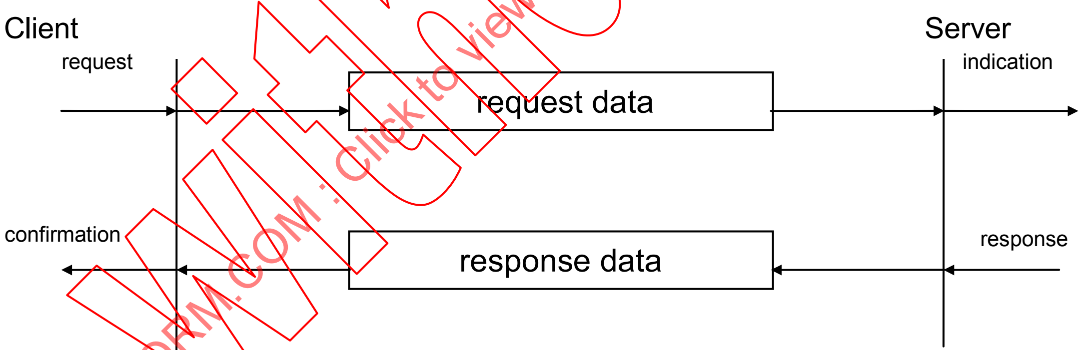


Figure 9 – Client/Server Communication

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.

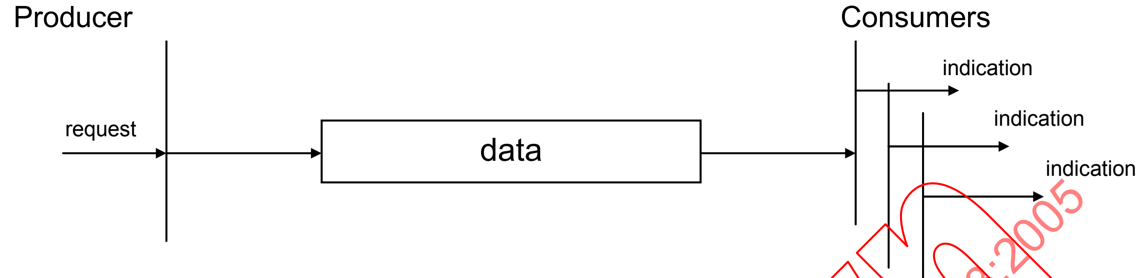


Figure 10 – Push model

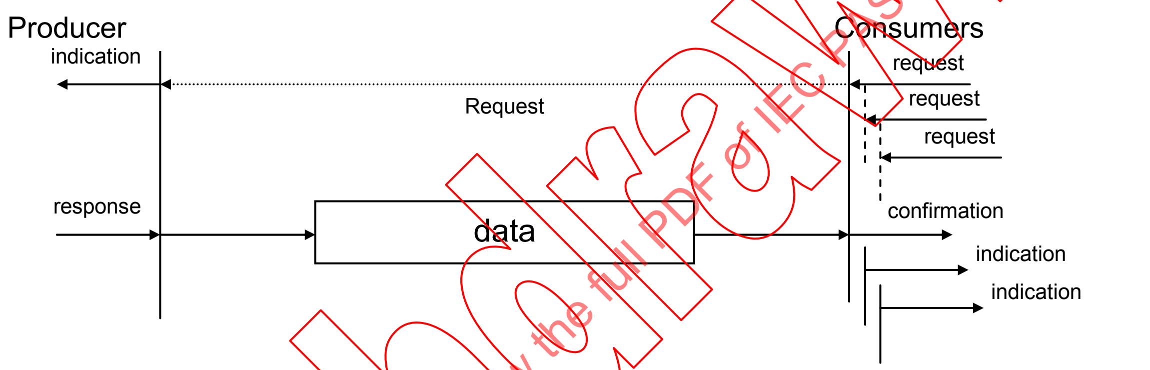


Figure 11 – Pull model

2.3.4 Superimposing of Communication Relationships

EPL collects more than one function into one frame (refer 4.6). That's why one of the mentioned communication relationships can't be usually applied to the complete frame but only to particulars services inside the frame.

PollResponse 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 trigger 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 Produccer/Consumer relationship.

3 Physical Layer

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

3.1 Topology

3.1.1 Hubs

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

Hubs have the advantage of reduced path delay value (about 500 ns) and have small frame jitter (about 40 ns).

Hubs may be integrated in the EPL interface cards.

3.1.2 Switches

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

3.2 Network Guidelines

EPL 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.

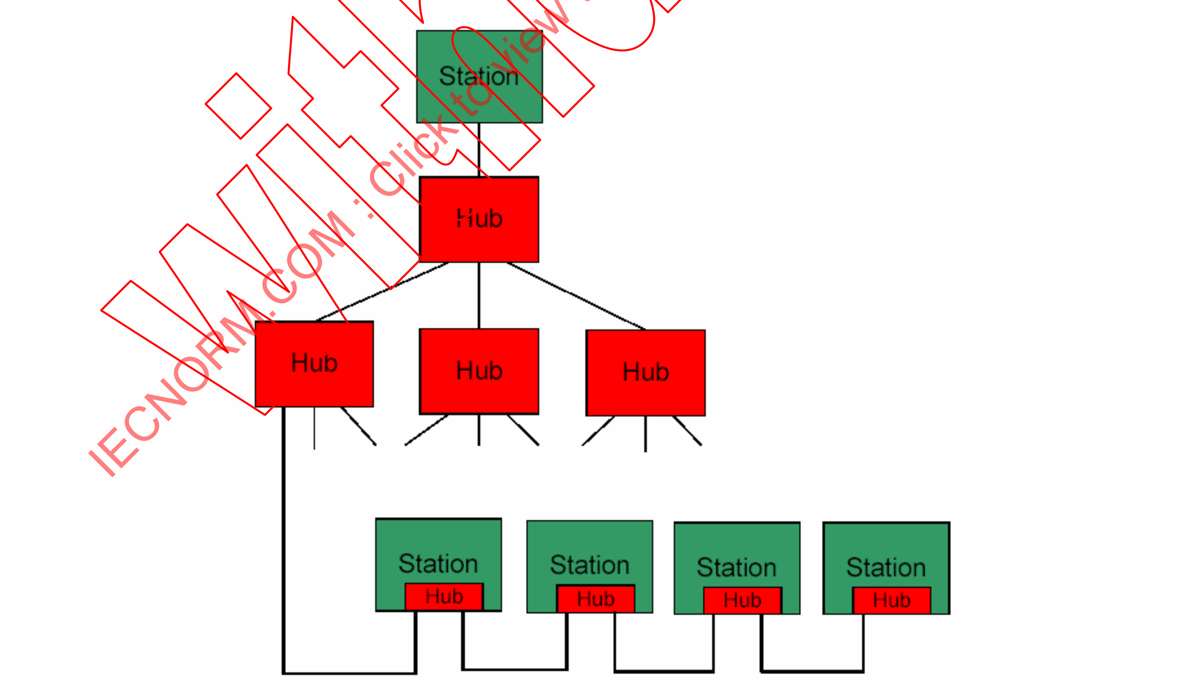


Figure 12 – Star topology and line topology

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 station failures. The default value of this timeout is 25µs. The round trip latency between the MN and a CN shall not exceed the timeout value. However the timeout value may be overridden globally or set for every single station.

3.3 Connectors

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

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

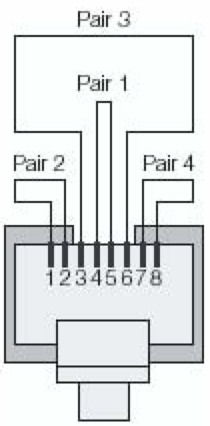
Both types may be mixed on the same cable.

For further information please refer to “IAONA Planing and Installation Guide, Release 3.0”.

3.3.1 RJ-45

Pin assignment as defined by EIA/TIA T568B.

The following is provided for convenience; please refer to the corresponding standard document.



Pin	Wire color code	Assignment 10BASE-T, 100BASE-TX	Assignment 1000BASE-TX
1	WHT/ORG	Tx+	BI_DA+
2	ORG	Tx–	BI_DA–
3	WHT/GRN	Rx+	BI_DB+
4	BLU		BI_DC+
5	WHT/BLU		BI_DC–
6	GRN	Rx–	BI_DB–
7	WHT/BRN		BI_DD+
8	BRN		BI_DD–

Figure 13 – RJ45 pin assignment

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.

The following is provided for convinience; please refer to the corresponding standard document.

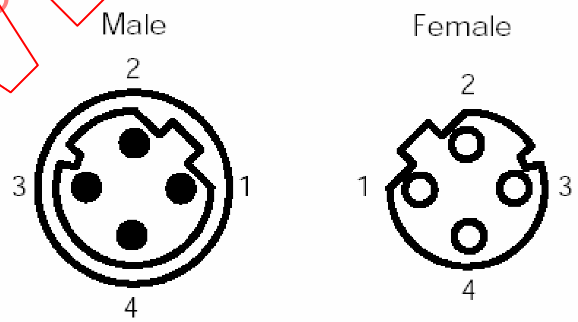


Figure 14 – IP67 connector pin assignment

Table 2 – Pin assignment IP67 connector

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

3.3.3 Cross Over Pin Assignment

The pin assignment shall be that of a cross over cable.

Therefore all devices can be interconnected by one type of cable.

The pin assignment of a cross over cable is defined as:

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

3.3.3.1 RJ45 to RJ45

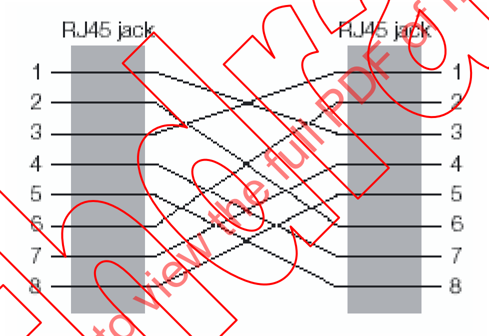


Figure 15 – recommended RJ45 to RJ45 pin assignment

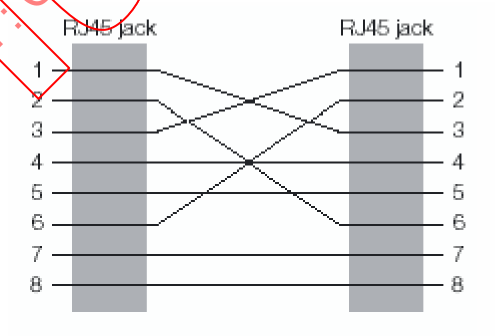


Figure 16 – not recommended RJ45 to RJ45 pin assignment

3.3.3.2 M12 to M12

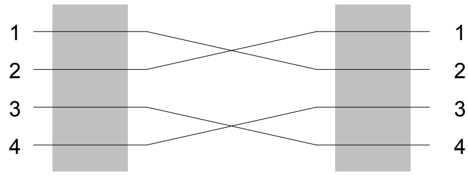


Figure 17 – M12 to M12 pin assignment

3.3.3.3 M12 to RJ45

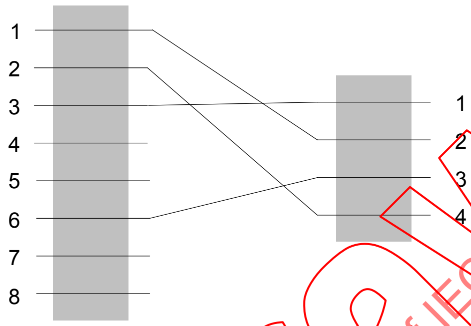


Figure 18 – M12 to RJ45 pin assignment

3.4 Cables (recommendation)

Standard patch cable (twisted pair, S/UTP, AWG26).

To increase noise immunity only cables with foil and copper netting shield should be used (S/UTP). This type of cable is usually called Cat5e. The maximum cable length (100 meters) predefined by Ethernet 100Base-TX shall apply.

Regarding wiring and EMC measures, the IAONA guidelines in the document "Industrial Ethernet Planning and Installation" shall be followed. The suclauses "Cable" and "System Installation" are relevant for EPL.

The pin assignment shall be that of a cross over cable.

4 Data Link Layer

4.1 Modes of Operation

Three operating modes are defined for EPL networks:

- EPL mode

  In EPL 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 EPL 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 EPL Mode.

  In EPL Mode most communication is transacted via EPL-specific messages. An asynchronous slot is available for non-EPL 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 (IEEE 802.3). 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.

- Open mode

  Open Mode provides isochronous communication and asynchronous communication. The nodes are synchronised by the Precision Time Protocol (PTP), see [IEEE 1588].

  The real-time nodes and regular Ethernet nodes don't need to be separated in protected network domains. In open mode, deterministic communication is still guaranteed, however timing constraints like cycle time (typically milliseconds) and jitter (typically 10th of microseconds) are more relaxed than in EPL mode and depending on the IEEE 1588 implementation (hardware or software clocks).

  Open Mode is not specified by this version of the ETHERNET Powerlink specification.

4.2 EPL Mode

4.2.1 Introduction

EPL 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 period.

Each node gets permission for sending its own frames by the EPL MN. Therefore, no collisions can occur and the collision-resolving CSMA/CD mechanisms responsible for the non-deterministic behaviour of Legacy Ethernet have no effect.

4.2.2 EPL Nodes

The node managing the cycle is called the EPL Managing Node (MN).

All other nodes communicate during their assigned time slots only and are called Controlled Nodes (CN).

4.2.2.1 EPL Managing Node

Only the MN can send messages independently – i.e. not in response to a received message. Controlled Nodes shall be only allowed to send when prompted 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: PollRequest), which then shall publish its data via multicast to all other nodes (frame: PollResponse).

The last frame in the isochronous period may be a multicast PollResponse frame of the MN (see Figure 19). With this frame the MN may publish its own data to the community of all other nodes.

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

Only one active MN is permitted in an EPL network.

4.2.2.2 EPL Controlled Node

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

4.2.2.2.1 Isochronous CN

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

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

PollRequests 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 PRes message transmitted by the MN.

Isochronous CNs may request the right to transmit asynchronous data from the MN.

4.2.2.2.2 Async-only CN

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

The MN shall cyclically poll each async-only CN during the asynchronous period by a StatusRequest – a special form of the Start of Asynchronous frame. The CN shall response by a StatusResponse, special form of Asynchronous Send frame. The poll interval shall be about 5 sec. It is afflicted by the asynchronous scheduling and thus non deterministic.

Async-only CNs may request the right to transmit asynchronous data from the MN.

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

4.2.3 Services

EPL provides three services:

- Isochronous Data Transfer

  One pair of messages per node shall be delivered every cycle, or every n-th cycle in the case of multiplexed CNs.

  Additionally, there may be one multicast message from the MN per cycle (frame: PollResponse).

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

  There may be one 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 exchanging data that is not time-critical (non real-time data).
- Synchronization of all nodes

4.2.4 EPL Cycle

The EPL cycle shall be controlled by the MN.

4.2.4.1 Isochronous EPL Cycle

Data exchange between nodes shall occur cyclically. It shall be repeated in a fixed interval (EPL cycle). The protocol is isochronous.

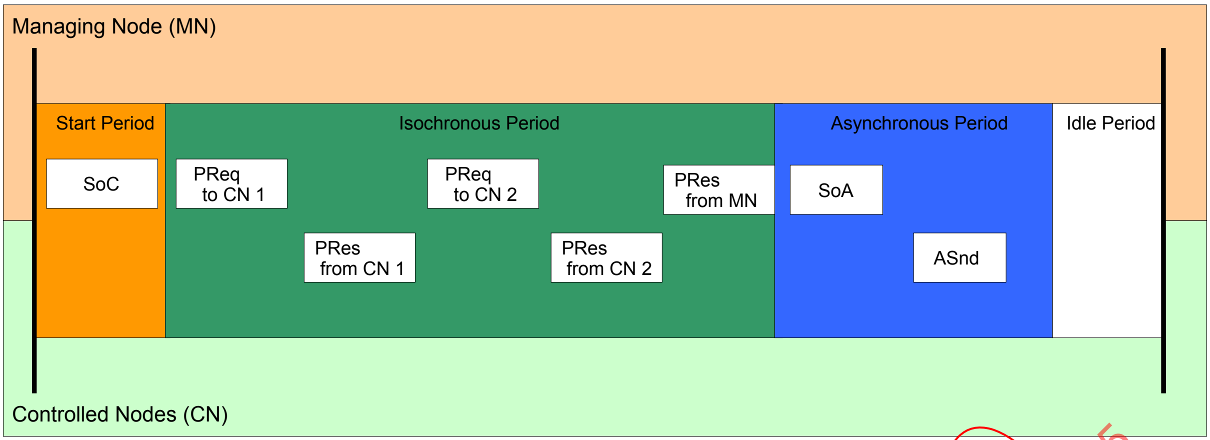


Figure 19 – EPL Cycle

The EPL cycle time shall be configured by the MN. The following time periods exist within one cycle:

- Start period
- Isochronous period
- Asynchronous period
- Idle period

It is important to keep the start time of an EPL cycle as exact (jitter-free) as possible. The length of individual periods can vary within the preset period of an EPL cycle.

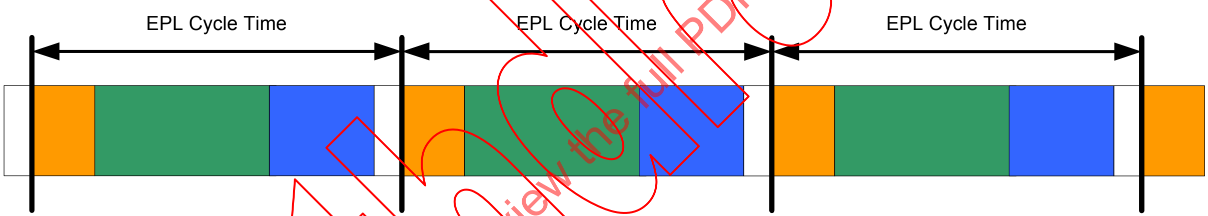


Figure 20 – EPL - an Isochronous Process

The network shall be configured so 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 behaviour, protecting the isochronous data (PReq and PRes) is not necessary or desired. Asynchronous data may be protected in higher protocol layers.

4.2.4.1.1 Start period

At the beginning of an EPL cycle, the MN shall send a Start of Cyclic (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).

4.2.4.1.2 Isochronous period

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.

The PReq shall be a directed unicast frame. It is received by the target node only. PRes shall be sent in a multicast way.

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

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

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

If all configured and active CNs have been processed, the MN may send its multicast PRes frame to all nodes. This frame is dedicated to data that are relevant for all or larger groups of CNs.

4.2.4.1.2.1 Multiplexed Timeslots

EPL supports communication classes that determine the cycles in which nodes are to be addressed.

- Continuous

  Continuous data shall be exchanged in every EPL cycle.
- Multiplexed

  Multiplexed data shall not be exchanged in every EPL cycle.

  For the whole set M of multiplexed data to resp. from all nodes, only a limited number of isochronous frames S shall be reserved. Thus, each cycle only S data frames of M are transferred. The next S data frames shall be transferred in the following cycle etc.

  S and M shall be configurable.

Continuous and multiplexed access scheme may be operated in parallel during one EPL cycle. The apportionment of the isochronous period to continuous and multiplexed subperiods shall be fixed by configuration.

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 positions from a few master axes, which are configured to continuous. The master axes are configured to communicate every cycle, accesses to the slave axes are 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.

4.2.4.1.3 Asynchronous period

In the asynchronous period of the cycle, access to the EPL network segment 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 EPL AsyncSend frame shall use the EPL addressing scheme and shall be sent via unicast or multicast to any other node.
- A Legacy Ethernet message may be sent. UDP/IP is the preferred type of message.

The MN shall start the asynchronous period with the Start of Asynchronous (SoA). The SoA shall used to identify inactive CNs, 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 period and is a signal for all CNs that all isochronous data has been exchanged during the isochronous period.

4.2.4.1.3.1 Asynchronous Scheduling

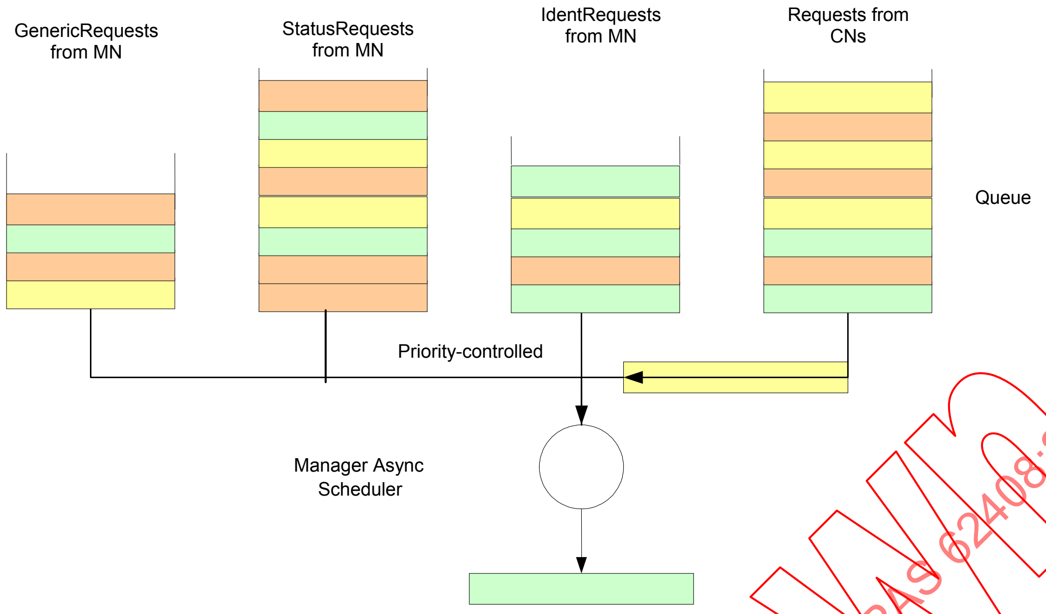


Figure 21 – Asynchronous Scheduling

Scheduling of all asynchronous data transfer is handled by the MN.

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

An asynchronous scheduler in the MN shall determine in which cycle the asynchronous frame will be sent. This guarantees 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 an SoA frame with a Requested Service Target identifying the node, which is allowed send an asynchronous frame.

The MN shall manage up to four queues for the dispatching of the asynchronous period.

- Generic transmit requests from the MN.
- IdentRequest frames from the MN to identify non active CNs
- StatusRequest frames to poll async-only CNs
- Generic transmit requests from CNs

4.2.4.1.3.2 Asynchronous Transmit Priorities

Asynchronous transmit requests may be prioritised by 3 PR bits in the PRes, the IdentResponse and StatusResponse frame (see 4.6.1.1.4, 7.4.3.2.1, 7.4.3.3.1).

EPL V. 2.0 knows two priority levels:

- NMTRequest (111b)

  NMT request priority may be applied, if an CN requests an NMT command to be issued by the MN
- GenericRequest (000_b)

  Generic request priority shall be applied for all non NMT requests.

4.2.4.1.3.3 Distribution of the Asynchronous period

With the PRes, IdentResponse resp. StatusResponse RS flag (3 bits, see 4.6.1.1.4, 7.4.3.2.1, 7.4.3.3.1) the CN shall indicate the number of send-ready packages in its queue.

An RS value of 0 (000_b) shall indicate that the queue is empty and an RS value of 7 (111b) shall indicate that 7 or more packages lines up.

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

Manufacturer specific solutions of multi-request assignment are allowed.

4.2.4.1.4 Idle Period

The Idle Period is time interval remaining between the completed asynchronous period and the beginning of the next cycle.

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

4.2.4.2 Reduced EPL Cycle

During system startup (state NMT_MN_PRE_OPERATIONAL_1, refer), a reduced EPL Cycle may be applied to diminish network load, while the system is being configured via SDO communication.

The Reduced EPL Cycle shall consist of queued asynchronous periods only. The duration of the asynchronous period and thus the duration of the Reduced EPL Cycle may vary from one cycle to next one.

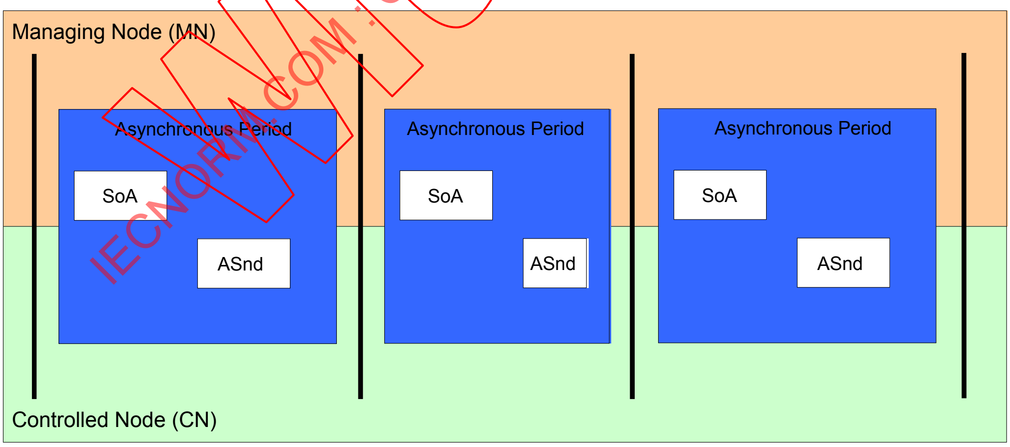


Figure 22 – Reduced EPL Cycle

The mechanism valid for the asynchronous period of the isochronous EPL cycle (4.2.4.1.3) shall be applied to the Reduced EPL Cycle.

4.2.4.3 EPL CN Cycle State Machine

4.2.4.3.1 Overview

The Cycle State Machine of the CN (DLL_CS) has to manage the communication within an EPL Cycle. The DLL_CS proves the flow order of the frames received within a cycle and reacts accordingly. The flow order is NMT_CS state dependent (see 4.2.4.3.4)

If an error in the communication is detected by the DLL_CS, an error event to the NMT CN State Machine (NMT_CS) will be generated. The scope of the DLL_CS is limited, so it should try to keep the communication online, even if some errors occur (e.g.: frame loss) till the NMT_CS state is changed.

4.2.4.3.2 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 till the beginning of the next cycle (triggered by a SoC frame from the MN). ASnd and IP frames may be received between the SoA and the SoC frames (asynchronous period). They must be handled in this state.

- **DLL_CS_WAIT_PREQ**

  After the beginning of the cycle, the state machine waits in this state for a PReq frame. After receiving it, the CN responses with a PRes frame. PRes frames from other CNs can be received during this state.

- **DLL_CS_WAIT_SOA**

  To signal the end of the isochronous period and the beginning of the asynchronous period, a SoA frame is sent. This state is used to verify that the isochronous part of the cycle flow was complete. PRes frames from other CNs can be received during this state.

4.2.4.3.3 Events

- **DLL_CE_SOC**

  This Event means that an EPL SoC frame was received from the MN. It marks the beginning of a new cycle and simultaneously the beginning of the isochronous period of the cycle.

- **DLL_CE_PREQ**

  This Event means that an EPL PReq frame was received from the MN.

- **DLL_CE_PRES**

  For cross traffic purposes, the CN listens to the PRes frames of other CNs. Every time a PRes frame is received, a DLL_CE_PRES event is produced.

- **DLL_CE_SOA**

  This event means that a SoC frame was received from the MN. It marks the end of the isochronous period of the cycle and the beginning of the asynchronous period.

- **DLL_CE_ASND**

  This event means that an ASnd frame or a non EPL frame was received. The frame types during the asynchronous period are not limited to EPL types, so this event describes also all other legal frame types (primarily IP, but also ARP, ICMP and so on).

- **DLL_CE_SOC_TIMEOUT**

  This event means that the cyclic connection was lost. It occurs, when the SoC frame was missed for some cycles and the CN SoC timer is up.

4.2.4.3.4 Usage of the the NMT_CS state by the DLL_CS

The state of the NMT_CS represents the network state and is used as a condition in some 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 wich are possible in a distinct NMT state.

A notation comment:

The transitions of DLL_CS could be displayed within a single diagram where the states of the NMT_CS are conditions for the transitions. 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 differencies are described in the transition table.

4.2.4.3.4.1 State NMT_CS_OPERATIONAL, NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE

In the NMT_CS_OPERATIONAL and NMT_CS_READY_TO_OPERATE states, there are three mandatory frames, which shall occur each cycle in the specified order: SoC, PReq and SoA.

In the NMT_CS_PRE_OPERATIONAL_2 state, there are two mandatory frames, which shall occur each cycle in the specified order: SoC and SoA. The PReq frame may occur.

The time segments between these frames were directly mapped to states of the Cycle State Machine. In these states, only distinct frame types are allowed and there is no further dependency of frame order within these states.

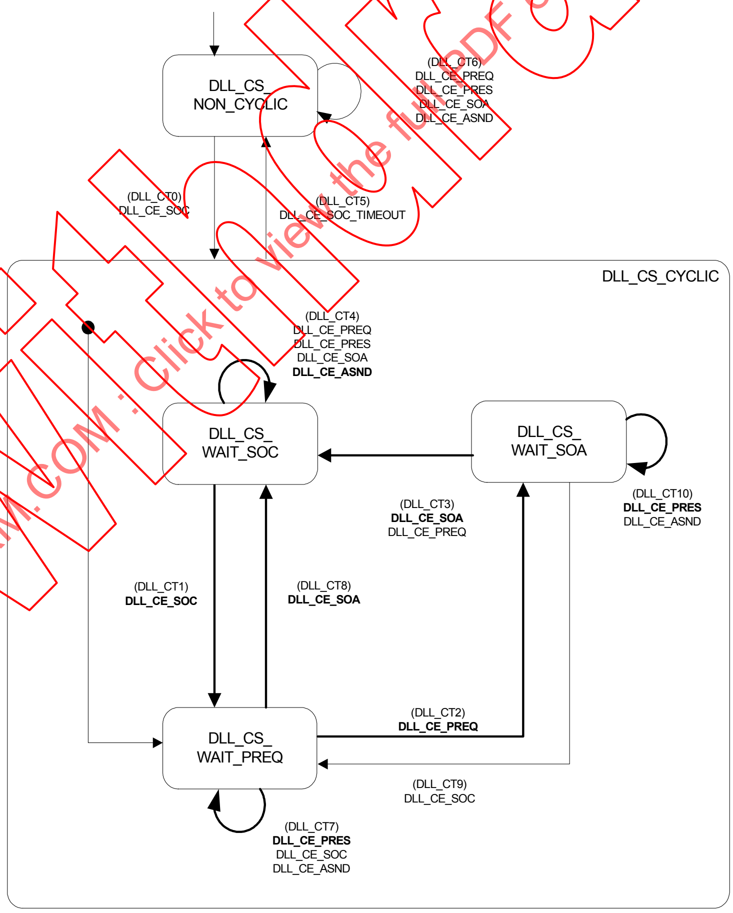


Figure 23 – CN Cycle State Machine, States NMT_CS_OPERATIONAL, NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE

The bold arrows and labels mark the errorless cycle flow.

4.2.4.3.4.1.1 Transitions

Table 3 – Transitions for CN Cycle State Machine, States NMT_CS_OPERATIONAL, NMT_CS_PRE_OPERATIONAL_2, NMT_CS_READY_TO_OPERATE

DLL_CT0	DLL_CE_SOC [ ] / synchronise to the next cycle begin
	If an SoC event occurred after a loss of connection or after the NMT state changes to NMT_CS_OPERATIONAL, NMT_CS_PRE_OPERATIONAL_2 or NMT_CS_PRE_OPERATIONAL_2, the communication of the CN will be synchronised to this start of the cycle.
DLL_CT1	DLL_CE_SOC [ ] / synchronise to the cycle begin, communicate with the application
	The occurence of the SoC event triggers the beginning of a new EPL Cycle. The asynchronous period of the previous cycle ends and the isochronous period of the next cycle begins.
DLL_CT2	DLL_CE_PREQ [ ] / accept the PReq frame and send a PRes frame
	The occurence of the PReq event corresponds to the normal cycle flow. The isochronous period of the communication is not finished yet. In the NMT state NMT_CS_PRE_OPERATIONAL_2 the CN sends dummy PRes frames. In the NMT state NMT_CS_READY_TO_OPERATE the CN sends PRes frames with invalid PDO. In the NMT state NMT_CS_OPERATIONAL the CN sends PRes frames with valid PDO.
DLL_CT3	DLL_CE_SOA [ ] / process SoA, if allowed send an ASnd frame or a non EPL frame DLL_CE_PREQ [ ] / ignore
	The isochronous period of the cycle is finished, when the SoA event occurred. The asynchronous period of the cycle begins. If there is an invite to the CN within the SoA frame, the CN is allowed to send his own ASnd frame or a non EPL frame. If DLL_CE_PREQ event occurs it means that at least two frames were lost (SoA and SoC) wich indicates serious problems. The state machine shall ignore the communication till it can synchronize to a new cycle start. The NMT_CS shall be notified.
DLL_CT4	DLL_CE_ASND [ ] / accept frame DLL_CE_SOA || DLL_CE_PRES [ ] / ignore DLL_CE_PREQ [] / response with PRes frame, ignore incoming data
	If a DLL_CE_ASND event has occurred during the asynchronous period of the cycle the corresponding frame shall be accepted. 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 period of the cycle. SoA, PRes frames shall be ignored. If a PReq frame was received, the incoming data may be ignored and a PRes frame shall be sent. The send data shall be marked as old/invalid by clearing the RD bit.
DLL_CT5	DLL_CE_SOC_TIMEOUT [ ] / go to DLL_CS_NON_CYCLIC
	When the SoC frame was missed for a predefined number of cycles and the SoC timeout is supported, the state machine changes to the state DLL_CS_NON_CYCLIC, where isochronous and asynchronous communication will be ignored by the Cycle State Machine (except SoC). The upper layer (NMT_CS) will be notified with an error event.
DLL_CT6	DLL_CE_ASND || DLL_CE_SOA || DLL_CE_PREQ || DLL_CE_PRES [ ] / ignore
	The isochronous and asynchronous communication, except the SoC frame, will be ignored by the Cycle State Machine (no state change). However, the frame types of asynchronous communication shall be forwarded to NMT_CS.
DLL_CT7	DLL_CE_PRES [ ] / process the cross traffic DLL_CE_SOC [ ] / synchronize to the cycle begin DLL_CE_ASND [ ] / ignore
	If a PRes frame of another CN was received (cross traffic), then the CN saves the data and waits for a PReq frame. The reaction on a SoC frame is state independent, the state machine assumes, that the originally expected frame was lost and synchronises to the start of new cycle. AsyncSend frames and non EPL frames will be ignored during the isochronous period.

DLL_CT8	DLL_CE_SOA [CN = multiplexed] / process SoA, send the own async. frame if allowed DLL_CE_SOA [CN != multiplexed] / process SoA, ..., generate error log
	If the CN is in the NMT_CS_OPERATIONAL or NMT_CS_READY_TO_OPERATE states there are two different behaviours. If the CN is not multiplexed, it assumes a loss of PReq frame. If the CN is multiplexed, the number of cycles since the last PReq is counted and interpreted accordingly. If the CN is in the NMT_CS_PRE_OPERATIONAL_2 state, the PReq frame is not mandatory.
DLL_CT9	DLL_CE_SOC [ ] / synchronise to the SoC
	The reaction on SoC is independent to the NMT state, the state machine assumes, that the originally expected frame was lost and synchronises on the start of new cycle.
DLL_CT10	DLL_CE_PRES [ ] / process the cross traffic DLL_CE_ASND [ ] / ignore
	If a PRes of another CN is received (cross traffic), then the controller saves the data. AsyncSend frames and non EPL frames will be ignored during the isochronous period.

Common issues:

- The loss of frames will be detected, when the next frame is received or the DLL_CE_SOC_TIMEOUT occurs. The received data will be accepted at the next SoC. The send data shall be marked as old/invalid by clearing the RD bit.
- The Cycle State Machine (DLL_CS) can handle several errors definitely and simply set the corresponding error status field. If the error can't be handled by the Cycle State Machine (see 4.7.2) an error notification to the NMT_CS will be sent. Than the NMT_CS can start a diagnosis to find out, what caused the error and will decide what has to be done.

4.2.4.3.4.2 Other States

In the following NMT states the Cycle State Machine stays in or changes to the DLL_CS_NON_CYCLIC state:

- NMT_GS_INITIALISATION
- NMT_CS_NOT_ACTIVE
- NMT_CS_PRE_OPERATIONAL_1
- NMT_CS_BASIC_ETHERNET

For description of these NMT states see 7.1.4.

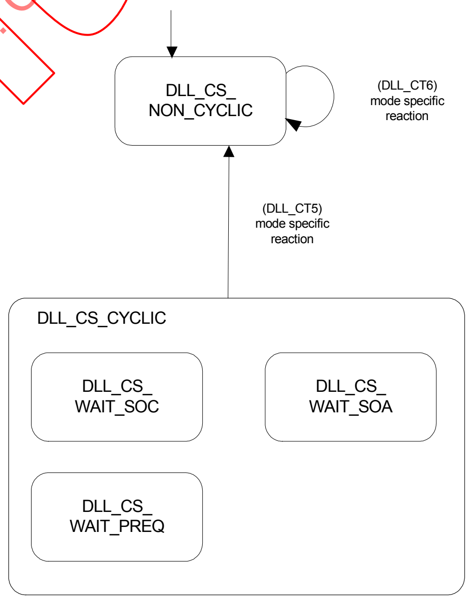


Figure 24 – CN Cycle State Machine, States NMT_CS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PRE_OPERATIONAL_1, NMT_CS_BASIC_ETHERNET

4.2.4.3.4.2.1 Transitions in other NMT states

Table 4 – Transitions for CN Cycle State Machine, States NMT_CS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PREOPERATIONAL_1, NMT_CS_BASIC_ETHERNET

DLL_CT6, DLL_CT5	DLL_CE_* [ ] / NMT state specific reaction
	The Cycle State Machine is not active in the NMT states NMT_CS_INITIALISATION, NMT_CS_NOT_ACTIVE, NMT_CS_PREOPERATIONAL_1 and NMT_CS_BASIC_ETHERNET. This means, after the initial transition to DLL_CS_NON_CYCLIC its state doesn't influence the reaction of the CN. The reactions are defined by the state of the NMT_CS only. (see 7.1.4)

4.2.4.4 EPL MN Cycle State Machine

4.2.4.4.1 Overview

The Cycle State Machine of the MN (DLL_MS) has to manage the communication within an EPL cycle.

The DLL_MS generates the flow of the frames during an EPL cycle and monitors the reaction of the CNs. The flow order is NMT_MS state dependent (see 4.2.4.4.4).

If an error in the communication is detected by the DLL_MS, an error event to NMT_MS will be generated.

4.2.4.4.2 States

- **DLL_MS_NON_CYCLIC**

  This state means that the cyclic communication isn't started yet or was stopt by the NMT_MS state machine (all NMT states except NMT_MS_OPERATIONAL). The state machine waits here till the NMT state changes to NMT_MS_OPERATIONAL. It depends on the current NMT state, which events will be processed and which will be ignored.

- **DLL_MS_WAIT_SOC_TIME**

  If the communication of the last cycle is finished, the state machine stays in this state until the next cycle begins with a DLL_ME_SOC_TIME.

- **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 send, the state machine waits in this state till the asynchronous period ends with the event DLL_ME_SOC_TIME. The asynchronous responses are accepted only during the asynchronous period.

4.2.4.4.3 Events

The DLL_MS is triggered by events which are generated by an event handler. There are two interfaces belonging to it:

- To the hardware
- To 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 interface to the hardware is out of the scope of this specification.

- **DLL_ME_PRES**

  This event means 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 an non EPL frame was received.

- **DLL_ME_SOC_TIME**

  This event means that a new cycle starts.

4.2.4.4.4 Usage of the NMT_MS state by the DLL_MS

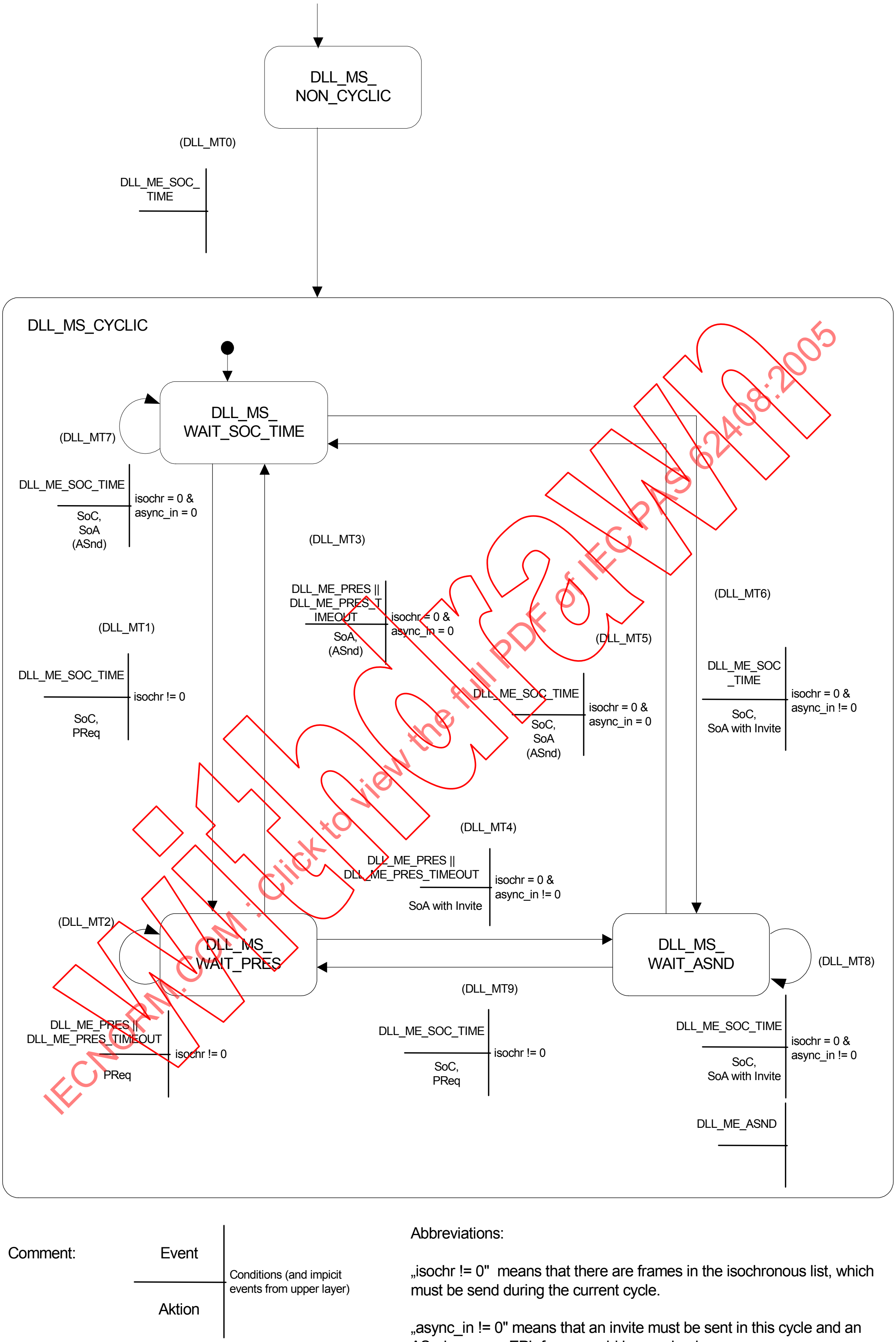
The state of the NMT_MS represents the network state and is used as a condition in some transitions of the DLL_MS. Because the NMT state influences the behaviour of the DLL_MS we could filter out the relevant DLL_MS transitions for a single NMT state, so we see only DLL_MS transitions wich are possible in a distinct NMT state.

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.

4.2.4.4.4.1 State NMT_MS_OPERATIONAL

In the NMT_MS_OPERATIONAL state, the Cycle State Machine of the MN generates the EPL Cycle and observes the behaviour of the CNs. Error handling is described in 4.7.



Abbreviations:

„isochr != 0" means that there are frames in the isochronous list, which must be send during the current cycle.

„async_in != 0" means that an invite must be sent in this cycle and an ASnd or a non EPL frame could be received.

Figure 25 – MN Cycle State Machine, States NMT_MS_OPERATIONAL

4.2.4.4.4.1.1 Transitions

Table 5 – Transitions for MN Cycle State Machine, State NMT_MS_OPERATIONAL

DLL_MT0	DLL_ME_SOC_TIME [ ] / go to state DLL_MS_WAIT_SOC_TIME
	If the NMT State Machine of the MN (NMT_MS) changes the mode to NMT_MS_OPERATIONAL it also starts the cycle timer, which generates the DLL_ME_SOC_TIME. The DLL_MS shall now prepare the system for the start of a first cycle.
DLL_MT1	DLL_ME_SOC_TIME [isochr != 0] / send SoC, PReq
	Immediately after DLL_ME_SOC_TIME event occurred an 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.
DLL_MT2	DLL_ME_PRES || DLL_ME_PRES_TIMEOUT [isochr != 0] / send next PReq
	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 doesn't change.
DLL_MT3	DLL_ME_PRES || DLL_ME_PRES_TIMEOUT [isochr = 0 & async_in = 0 & async_out = 0] / send SoA DLL_ME_PRES || DLL_ME_PRES_TIMEOUT [isochr = 0 & async_in = 0 & async_out = 1] / send SoA, ASnd
	The isochronous period ends with either a DLL_ME_PRES or a DLL_ME_PRES_TIMEOUT. 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_TIME. 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_TIME.
DLL_MT4	DLL_ME_PRES || DLL_ME_PRES_TIMEOUT [isochr = 0 & async_in != 0] / send SoA with Invite
	The isochronous period ends with either a DLL_ME_PRES or a DLL_ME_PRES_TIMEOUT. 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 send.
DLL_MT5	DLL_ME_SOC_TIME [isochr = 0 & async_in = 0 & async_out = 0] / send SoC, SoA DLL_ME_SOC_TIME [isochr = 0 & async_in = 0 & async_out = 1] / send SoC, SoA, ASnd
	Immediately after the DLL_ME_SOC_TIME event 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 additionaly sent. The state doesn't change. 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_TIME.
DLL_MT6	DLL_ME_SOC_TIME [isochr = 0 & async_in != 0] / send SoC, SoA with Invite
	Immediately after the DLL_ME_SOC_TIME 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 additionaly.
DLL_MT7	DLL_ME_SOC_TIME [isochr = 0 & async_in = 0 & async_out = 0] / send SoC, SoA DLL_ME_SOC_TIME [isochr = 0 & async_in = 0 & async_out = 1] / send SoC, SoA, ASnd
	Immediately after the DLL_ME_SOC_TIME event 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 additionaly sent. The state doesn't change. If there is outgoing asynchronous communication to be done in the current cycle, the MN sends this frame after the SoA.

DLL_MT8	DLL_ME_ASND [ ] / process the frame DLL_ME_SOC_TIME [isochr = 0 & async_in != 0] / send SoC, SoA with Invite
	Immediately after the DLL_ME_SOC_TIME 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 additionaly.
DLL_MT9	DLL_ME_SOC_TIME [isochr != 0] / send SoC, PReq
	Immediately after DLL_ME_SOC_TIME event occurred an 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.

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.
- „async_in != 0" means that an invite must be sent in this cycle and an ASnd or a non EPL frame could be received.
- “async_out != 0” means that an ASnd must be send in this cycle after an SoA was sent.

The reaction on unexpected events is not described in Figure 25 and Table 5 because of clarity purposes. A general statement for these events can be given:

- The unexpected frame types and unexpected sender shall be ignored. The state does not change. The PRes frames shall be passed to the NMT State Machine, wich 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 send by another MN only (SoC, SoA, PReq), it shall notify the NMT State Machine.
- If an unexpected internal event (e.g. timeout) occurs, an internal error will be assumed and the NMT_MS will be notified.
- Because the current cycle time have not to be measured, it could happen that the DLL_ME_SOC_TIME occurs in states where it was not expected. In this case the NMT_MS will be notified. The DLL_MS shall not send any frames till the end of the violated cycle. For details see 4.7.

4.2.4.4.4.2 Other Modes

If the NMT state does not equal NMT_MS_OPERATIONAL, the Cycle State Machine is not active. This means, after the initial transition to DLL_MS_NON_CYCLIC its state doesn’t influence the reaction of the MN. The reactions are defined by the state of the NMT_MS.

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 an CN receives an IdentRequest addressed to itself, it shall return an IdentResponse, a special form of an ASnd frame, in the same asynchronous period.

The CN shall be marked as active if the CN 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 acts according to the rules of the Legacy Ethernet (IEEE 802.3). The network is accessed according to CSMA/CD.

The network communication is collision-afflicted and not 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/IPThe 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 EPL networks. Higher layer protocols shall be applied to handle communication errors caused by collisions unresolved by CSMA/CD.

EPL 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 Adressing

An EPL 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 EPL shall be globally unique, or at least unique within the EPL 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:

Table 6 – Assigned Multicast Addresses

	MAC-Multicast address
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

4.4.3 MAC Broadcast

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

4.5 EPL Addressing

Each EPL node (MN, CN and Router) has a unique Node ID within an EPL 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. EPL Node IDs 1-239 may used for the CNs. Table 7 shows the EPL Node ID assignment.

Table 7 – EPL Node ID Assignment

EPL Node ID	**Description**
0	Invalid
1-239	EPL CNs
240	EPL MN
241-252	Reserved
253	Diagnostic device
254	EPL to legacy Ethernet Router
255	EPL broadcast

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

4.6 Frame Structures

4.6.1 Integration with Ethernet

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

EPL messages shall be encapsulated in Ethernet type II frames. The Ethernet Type (Ethertype) field shall be set to $88AB_h$.

The frame's length shall be restricted to the configured size; otherwise, the cycle period could not be guaranteed. Ethernet frames shall not be shorter than the specified minimum of 64 octets.

4.6.1.1 EPL Frame

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

4.6.1.1.1 EPL Basic Frame

The EPL Basic Frame format shall comprise 5 fields:

- Reserved (1 bit)
- MessageType (7 bits)
- Destination node address (1 octet), EPL addressing scheme (cf. 4.5)
- Source node address (1 octet), EPL addressing scheme (cf. 4.5)
- Data depending on service (n octets)

The EPL Basic Frame format shall be encapsulated in the Ethernet type II frame consisting of 14 octets Ethernet header (Destination and Source MAC addressesEtherType) and 4 octets CRC32 checksum.

Table 8 – EPL Basic Frame structure

	Bit Offset								**entry defined by**
Octet Offset [1]	**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 Powerlink
15	Destination								
16	Source								
17 - n	Data								
n+1 - n+4	CRC32								Ethernet type II

n ≥ 59

The ETHERNET Powerlink defined part of the Ethernet frame shall be regarded to be the EPL frame.

[1] Octet Offset refers to the start of the Ethernet frame.

Table 9 EPL – Basic Frame data fields

Field	Abbr.	Description	Value
Destination MAC Address	dmac	MAC address of the addressed node resp. nodes	see 4.4
Source MAC Address	smac	MAC address of the transmitting node	see 4.4
EtherType	etyp	Ethernet message type identification	$88AB_h$
MessageType	mtyp	EPL message type identification	see Table 10
Destination	dest	EPL Node ID of the addressed node resp. nodes	see 4.5
Source	src	EPL Node ID of the transmitting node	see 4.5
Data	data	Data depending on MessageType	refer below
CRC32	crc	CRC32 checksum	

The following message types shall be applied:

Table 10 EPL – Message types

Message Type		MAC Transfer type
Start of Cyclic (SoC)	01_h	Multicast
PollRequest (PReq)	03_h	Unicast
PollResponse (PRes)	04_h	Multicast
Start of Asynchronous (SoA)	05_h	Multicast
AsynchronousSend (ASnd)	06_h	Unicast / Multicast / Broadcast

Refer to 4.4.2 for Multicast adresses to be used be the respective message type.

Reserved values shall be set to 0.

4.6.1.1.2 Start Of Cyclic (SoC)

Table 11 – SoC Frame structure

	Bit Offset							
Octet Offset [2]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	res	MessageType = 01_h						
1	Destination = FF_h							
2	Source = MN Node ID							
3	reserved							
4	MS	PS	res	res	res	res	res	res
5	res	res	res	res	reserved			
6 - 13	NetTime							
14 - 45	reserved							

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

[2] Octet Offset refers to the start of the EPL frame. Offset to the start of the Ethernet frame is 14 Octets.

Table 12 – SoC Frame data fields

Field	Abbr.	Description	Value
MessageType	mtyp	EPL message type identification	01_h
Destination	dest	EPL Node ID of the addressed node resp. nodes	FF_h
Source	src	EPL Node ID of the transmitting node	$F0_h$ (MN NodeID)
Multiplexed Slot	MS	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). This prescaled signal is useful for “slow“ nodes, which can not react every cycle (the cycle interrupt to the application will be generated every n-th cycle).	
NetTime	time	MN shall distribute the starting time of the EPL cycle. NetTime shall be of type (QUERVERWEIS Datatypes)	

4.6.1.1.3 PollRequest (PReq)

Table 13 – PReq Frame structure

	Bit Offset							
Octet Offset [3]	**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	res	res	res	RS			
6	PDOVersion							
7	res							
8 – 9	Size							
10 – n	Payload							

n<1500

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

[3] Octet Offset refers to the start of the EPL frame. Offset to the start of the Ethernet frame is 14 Octets.

Table 14 – PReq Frame data fields

Field	Abbr.	Description	Value
MessageType	mtyp	EPL message type identification	03_h
Destination	dest	EPL Node ID of the addressed node resp. nodes	CN NodeID
Source	src	EPL Node ID of the transmitting node	$F0_h$ (MN NodeID)
Multiplexed Slot	MS	Flag: Shall be set in PReq frames to CNs that are served by a multiplexed timeslot	
Exception Acknowledge	EA	Flag: Error signalling, refer 6.6.1.3	
Ready	RD	Flag: Shall be set if the transferred payload data are valid. It shall be set by the application process of the MN. An CN shall be allowed to accept data only when this bit is set	
RequestToSend	RS	Flags: Shall indicate the number of pending RequestToSend from the MN's point of view.	0 - 7
PDOVersion	pdov	Shall indicate the version of the PDO encoding used by the payload data, refer 6.4.1	
Size	size	Shall indicate the number of payload data octets	0 - 1490
Payload	pl	Isochronous payload data sent from the MN to the addressed CN. The lower layer shall be responsible for padding. Payload to be used by PDO, refer 6.4	

4.6.1.1.4 PollResponse (PRes)

Table 15 – PRes Frame structure

	Bit Offset							
Octet Offset [4]	**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	PR			RS			
6	PDOVersion							
7	reserved							
8 – 9	Size							
10 – n	Payload							

n<1500

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

[4] Octet Offset refers to the start of the EPL telegram slot. Offset to the start of the Ethernet telegram is 14 Octets.

Table 16 – PRes Frame data fields

Field	Abbr.	Description	Value
MessageType	mtyp	EPL message type identification	04_h
Destination	dest	EPL Node ID of the addressed node resp. nodes	FF_h
Source	src	EPL Node ID of the transmitting node	CN NodeID
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 signalling, refer 6.6.1.3	
Ready	RD	Flag: Shall be set if the transferred payload data are valid. It shall be handled by the application process in the CN. All other CNs and the MN shall be allowed to take over data only if RD is set	
Priority	PR	Flags: Shall indicate the priority of the requested asynchronous frame. (cf. 4.2.4.1.3.2)	0, 7
RequestToSend	RS	Flags: Shall indicate the number of pending requests to send at the CN. The value 7 shall indicate 7 or more requests, 0 shall indicate no pending requests	0 - 7
PDOVersion	pdov	Shall indicate the version of the PDO encoding used by the payload data, refer 6.4.1	
Size	size	Shall indicate the number of payload data octets	0 - 1490
Payload	pl	Isochronous payload data sent from the MN to the addressed CN. The lower layer shall be responsible for padding. . Payload to be used by PDO, refer 6.4	

4.6.1.1.5 Start Of Asynchronous (SoA)

Table 17 – SoA Frame structure

	Bit Offset							
Octet Offset [5]	**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	res	res
5	res	res	res	res	RS/res			
6	RequestedServiceID							
7	RequestedServiceTarget							
8	EPLVersion							
9 – 45	reserved							

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

[5] Octet Offset refers to the start of the EPL telegram slot. Offset to the start of the Ethernet telegram is 14 Octets.

Table 18 – SoA Frame data fields

Field	Abbr.	Description	Value
MessageType	mtyp	EPL message type identification	05_h
Destination	dest	EPL Node ID of the addressed node resp. nodes	FF_h
Source	src	EPL Node ID of the transmitting node	$F0_h$ (MN NodeID)
NMTStatus	stat	Shall report the current status of the MN's NMT state machine	
Exception Acknowledge	EA	Flag: Error signalling, refer 6.6.1.3 EA bit shall be operated only, if SoA is transporting a StatusRequest or an IdentRequest (refer RequestedServiceID). The bit shall address the node that the StatusRequest resp. IdentRequest is addressed to (refer RequestedServiceTarget)	
RequestToSend	RS	Flags: Shall indicate the number of pending Requests to Send from the MN's point of view. The RS bits shall be operated only, if SoA is transporting a StatusRequest or an IdentRequest (refer RequestedServiceID) The bit shall address the node that the StatusRequest resp. IdentRequest is addressed to (refer RequestedServiceTarget).	0 - 7
RequestedServiceID	svid	Shall qualify the asynchronous service ID dedicated to the SoA and to the following asynchronous slot (refer below). 00h (NoService) shall indicate that the following asynchronous slot will be applied by the MN itself, or that there will be no usage of the asynchronous slot.	
RequestedServiceTarget	svtg	Shall indicate the EPL address of the CN, to that the right to send is granted. 00h (NoTarget) shall indicate that the the following asynchronous slot will be applied by the MN itself, or that there will be no usage of the asynchronous slot	
EPLVersion	eplv	Shall indicate the current EPL version of the MN	

RequestedServiceID and RequestedServiceTarget together form the **AsyncInvite Command**.

4.6.1.1.5.1 RequestedServiceID s

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

Table 19 – Definition of the RequestedServiceID in the SoA frame

Requested ServiceID	Description	Comment
0	**NoService**	NoService service ID shall be used if the asynchronous slot isn't assigned to any CN: • There is no pending request for the asynchronous slot from any CN • The asynchronous slot will be used by MN
1	**IdentRequest**	IdentRequest shall be used to test the existence of an inactive CN and to query the identification data of a CN. The addressed CN shall answer immediately after the reception of the SoA, e.g. during the current EPL cycle, with its IdentResponse frame, a special from the ASnd frame.
2	**StatusRequest**	StatusRequest shall be used to request the current status of a CN including detailed error information. Async-only CNs shall be cyclically queried by StatusRequest to supervise their status and to query their requests for the asynchronous slot. The addressed CN shall answer immediately after the reception of the SoA, e.g. during the current EPL cycle, with its StatusResponse frame, a special from the ASnd frame.
3	**NMTRequestInvite**	NMTRequestInvite shall be used to assign the asynchronous transmit access to a CN that has indicated a pending NMTRequest via a Request to Send (RS bit of PRes, StatusResponse or IdentResponse) with the priority level of NMTRequest. The addressed CN shall answer immediately after the reception of the SoA, e.g. during the current EPL cycle, with its NMTRequest frame, a special from the ASnd frame..
4 – 191	Free to use	
192 – 255	Reserved	
255	**UnspecifiedInvite**	UnspecifiedInvite shall be used to assign the asynchronous transmit access to a CN that has indicated a pending transmit request via a Request to Send (RS bit of PRes, StatusResponse or IdentResponse). The addressed CN shall answer immediately after the reception of the SoA, e.g. during the current EPL cycle, with any kind of EPL ASnd frame as well as with any Legacy Ethernet frame.

4.6.1.1.6 AsynchronousSend (ASnd) – EPL format

Table 20 – ASnd Frame frame structure

	Bit Offset							
Octet Offset [6]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	res	MessageType						
1	Destination							
2	Source							
3	ServiceID							
4 – n	Payload							

n<1500

6 Octet Offset refers to the start of the EPL telegram slot. Offset to the start of the Ethernet telegram is 14 Octets.

The transmission of an ASnd frame by a CN shall occur immediately after the reception of a SoA frame assigning the right to transmit to the CN.

ASnd frames shall be transmitted using a unicast, multicast or broadcast MAC address (cf. 4.4).

Table 21 – SoA Frame data fields

Field	Abbr.	Description	Value
MessageType	mtyp	EPL message type identification	05_h
Destination	dest	EPL Node ID of the addressed node resp. nodes	NodeID
Source	src	EPL Node ID of the transmitting node	NodeID
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous slot (refer below)	
Data	data	Shall contain data, that are specific for the current ServiceID	

ServiceID and Data shall be regarded to form the **ASnd service slot**.

4.6.1.1.6.1 ASnd ServiceID values

The following values shall be used for the ServiceID entry:

Table 22 – ServiceID values in the ASnd frame

Service ID	Description	Comment
0	Reserved	
1	**IdentResponse**	IdentResponse shall be issued by a CN that received an IdentRequest via SoA.
2	**StatusResponse**	StatusResponse shall be issued by a CN that received a StatusRequest via SoA.
3	**NMTRequest**	NMTRequest shall be issued by a CN that received a NMTRequestInvite via SoA.
4	**NMTCommand**	NMTCommand shall be issued by the MN upon an internal request or upon an external request via NMTRequest.
5 – 191	Free to use	
192 – 255	Reserved	

4.6.1.2 Non-EPL Frames

Non-EPL frames may be transmitted in accordance with the specifications of any Legacy Ethernet protocol. Preferred protocols are:

- UDP/IP
  defined by RFC 791 (IP) and RFC 768 (UDP)
- TCP/IP
  defined by RFC 791 (IP) and RFC 793 (TCP)

Non-EPL frame transmission is carried out by the MN as required, and by a CN after the assignment of a send authorization via a SoA UnspecifiedInvite frame.

Refer 5.2 for special requirements to Non-EPL 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 / 100MBit / full duplex)
  - Transmission Errors detected by CRC errors
  - Rx buffer overflow
  - Tx buffer underrun
- EPL Data Link Layer error symptoms
  - Loss of frame
  - SoC-Frame/ SoA-Frame
  - PollRequest / PollResponse Frame
  - ASnd Frame / IP Frame
  - Collisions
  - Cycle Time exceeded
  - EPL Address Conflict
  - Multiple Managing Nodes
  - Format Errors within Protocol objects
  - Timing Violation (late Response)

4.7.2 Error Handling Table for CN

Table 23 – CN Error Handling Table

Error Symptoms detected by the CNs	Cumulative Ctr	Threshold Ctr	Direct Reaction	DLL Local Handling	Error Codes	NMT Local Handling
Loss of link	o		o		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	These are considered to be error sources	E_DLL_MAC_BUFFER	NMT_GT6 Internal Communication Error (handling of internal SW errors) Logging in Error History
CRC Error	m	o			E_DLL_CRC_TH	NMT_CT11, Error Condition Logging in Error History
Collision	o	o			E_DLL_COLLISION_TH	NMT_GT6 Internal Communication Error (handling of internal SW errors) Logging in Error History
Invalid Format			m		E_DLL_INVALID_FORMAT	NMT_GT6 Internal Communication Error (handling of internal SW errors) Logging in Error History
SoC jitter out of range	o	o	o		E_DLL_JITTER_TH	NMT_CT11, Error Condition Logging in Error History
Loss of PollRequest	o	o			E_DLL_LOSS_PREQ_TH	NMT_CT11, Error Condition Logging in Error History
Loss of SoA	o	o			E_DLL_LOSS_SOA_TH	NMT_CT11, Error Condition Logging in Error History
Loss of SoC	m	m		CN sends (PResp) the last or actual values. Invalid data shall not be sent in any case.	E_DLL_LOSS_SOC_TH	NMT_CT11, Error Condition Logging in Error History

Remarks:

- Change of NMT state is signalled to all nodes (reason can be read at Object ERR_History_ADOM)
- In Object ERR_History_ADOM, all logging events have to be registered (even if they are not signalled by emergency).
- Node of the described Error symptoms on the CN will be signalled by emergency to the MN.
- m → mandatory (Counters: shall be implemented / Detection: shall be detected)
- → 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

4.7.3 Error Handling Table for MN

Table 24 – MN Error Handling Table

Error Symptoms	Cumulative Ctr	Threshold Ctr	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) Logging in Error History
CRC error	m	o			E_DLL_CRC_TH	NMT_MT6 Logging in Error History
Collision	o	o			E_DLL_COLLISION_TH	NMT_GT6 Internal Communication Error (handling of internal SW errors) Logging in Error History
Collision			m	Communication suspends for a configurable number of cycles. Changes its State to: 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. Logging in Error History
Multiple MNs			o		E_DLL_MULTIPLE_MN	State != NMT_MS_NOT_ACTIVE -> NMT_GT6 Internal Communication Error State == NMT_MS_NOT_ACTIVE -> reside in NMT_MS_NOT_ACTIVE Logging in Error History
EPL Address conflict			m s		E_DLL_ADDRESS_CONFLICT	Remove all involved CNs from configuration
Loss of PollResponse	o s	m s			E_DLL_LOSS_PRES_TH	Remove respective CN from configuration, Send NMT State Command "NMTResetNode" to respective CN. Logging in Error History
Late PollResponse	o s	m s			E_DLL_LATE_PRES_TH	Remove respective CN from configuration, Send NMT State Command "NMTResetNode" to respective CN. Logging in Error History

Cycle time exceeded	o	o			E_DLL_CYCLE_ EXCEED_TH	NMT_MT6 Logging in Error History
Cycle time exceeded			m	Skip next cycle	E_DLL_CYCLE_EXCEED	Logging in Error History

Remarks:

- Change of NMT state is signalled to all nodes (See Object ERR_History_ADOM)
- In Object ERR_History_ADOM, all logging events have to be registered, even if they are not signalled by emergency.
- None of the described Error symptoms on the CN will be signalled by emergency to the MN.
- m → mandatory (Counters: shall be implemented / Detection: shall be detected)
- → 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

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 the may get registered. On each node an Error History exists, where the occurred error symptoms are stored.

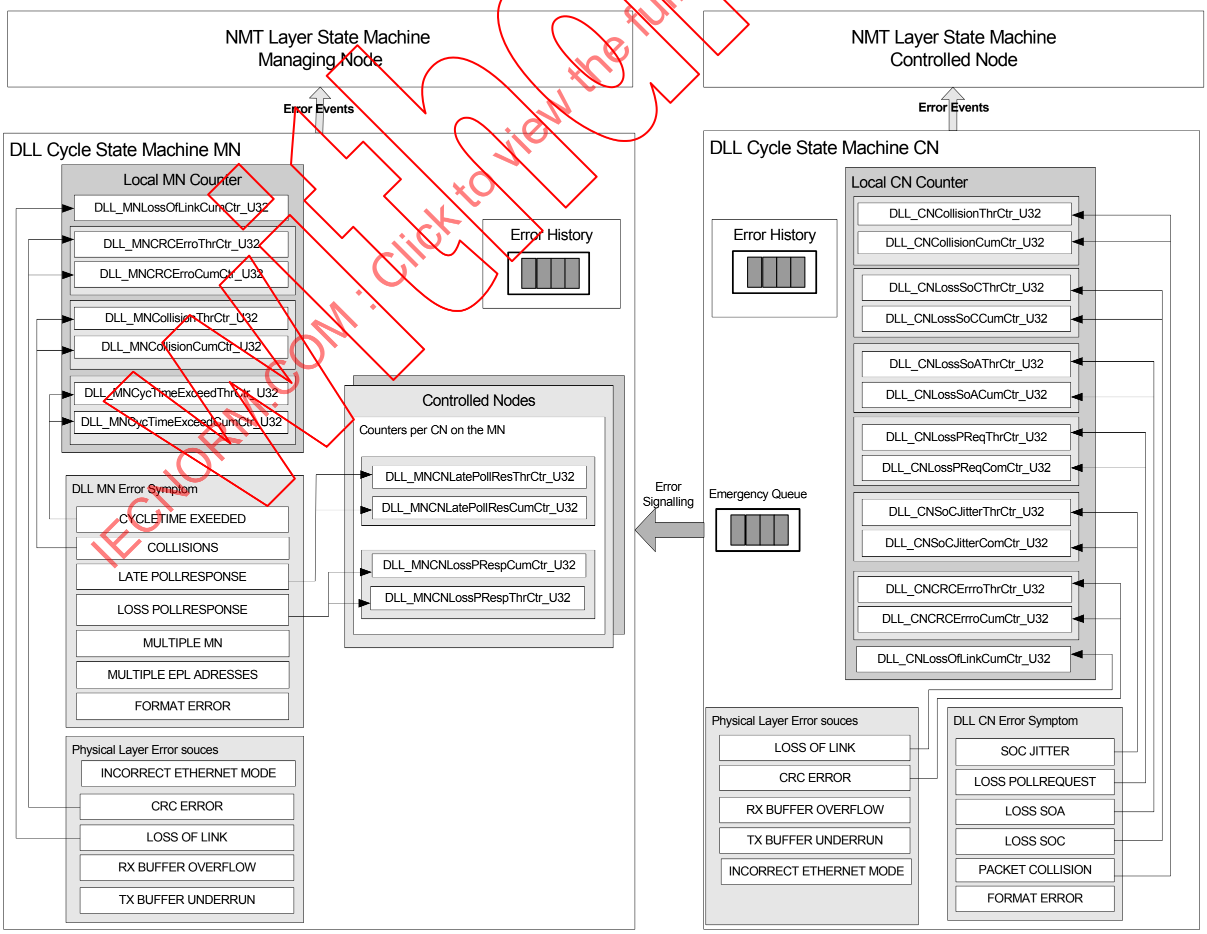


Figure 26 – Error Registration

4.7.4.1 Threshold counters

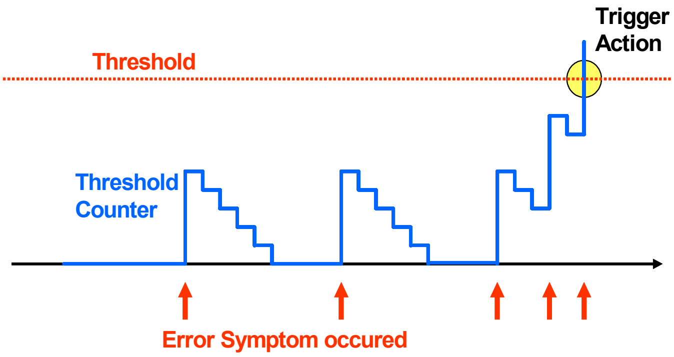


Figure 27 – Threshold counter

Every time an Error symptom occurs the Threshold counter is incremented by 8. After each successful cycle without error the counter is decremented by one (Threshold counter 8:1). When the Threshold value is reached, it will trigger an action and the counter will be reset. The Threshold value is configurable.

4.7.4.2 Cumulative Counter

The Cumulative counter is incremented every time an error symptom occurs, thus an overflow is possible.

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 wire breaks, somebody pulls out the network cable or a hub in the EPL network is defect.

- **Error recognition**

  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. Every time one of the following error symptoms below occurs, the MN resp. CN controls if the cause of the error symptom was a Loss of Link:

  - Frame Loss
  - Slot time exceeded
  - Cycle time exceeded

- **Handling**

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

- **Registration**

  Optionally a cumulative counter (DLL_MNLossOfLinkCumCtr_U32, resp. DLL_CNLossOfLinkCumCtr_U32) is incremented.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LOSS_OF_LINK	XXXX	

4.7.5.2 Incorrect physical Ethernet operating modes

- **Error source**

  During the initialisation the new node may check whether it is 100MBit half duplex Ethernet operational mode. Otherwise it can't guarantee the timing requirements. This situation can occur if auto negotiation is used (not allowed see 3) and the communication partner is using 10 MBits (e.g. a Hub) or 100 MBit full duplex (e.g. a Switch).

- **Error recognition**

  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:

  - Frame Loss
  - Slot time exceeded
  - Cycle time exceeded

- **Handling**

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

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_BAD_PHYS_MODE	XXXX	

4.7.6 Rx MAC buffer overflow / Tx MAC buffer underrun

- **Error source**

  If the receive MAC buffer of a CN or MN overflows, it couldn't proceed received frames for a while. The MAC receive FIFO was full when it needed to store a received byte. This is a serious internal failure. The Transmit MAC Buffer underrun error on the physical layer occurs; whenever a MAC transmits FIFO becomes empty during transmission. This is a serious internal failure.

- **Error recognition**

  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. Every time, when one of the following error symptoms occurs, the MN resp. CN checks if the cause of the error symptom was an Rx MAC buffer overflow or a TX MAC buffer underrun:

  - Frame Loss
  - Slot time exceeded
  - Cycle time exceeded

- **Handling**

  If a Buffer error is detected, it is logged in the error history (Object) 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.

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_MAC_BUFFER	XXXX	

4.7.6.1 Transmission / CRC Errors

- **Error source**

  Transmission errors are detected by hardware (CRC-Check) in the Ethernet-Controller. A CRC failure can occur through Electromagnetic compatibility (EMC) influence. Received frames containing CRC errors are simply discarded.

- **Error recognition**

  Whenever a loss of a frame occurs, the MN checks the Physical Layer for a CRC error on the Ethernet MAC controller. Every time a frame is lost, the MN or CN checks if a CRC Error has occurred.

- **Handling**

  If a CRC error is detected, it is logged in the error history (Object) and the NMT layer is notified. The CN NMT state machine handles this error source as an "error condition" (NMT_CT11) and changes its state to NMT_CS_PRE_OPERATIONAL_1. The MN NMT state machine handles this error source as an "error condition" (NMT_MT6) and changes its state to NMT_MS_PRE_OPERATIONAL_1.

- **Registration**

  Every time a CRC Error occurs a cumulative counter (DLL_MNCRCErrorCumCtr_U32, resp. DLL_CNCRCErrorCumCtr_U32) shall be incremented. Optionally a Threshold Counter (DLL_MNCRCErrorThrCtr_U32, resp. DLL_CNCRCErrorThrCtr_U32) may be incremented.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_CRC_TH	XXXX	

4.7.7 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.

4.7.7.1 Timing Violation

4.7.7.1.1 Slot Time Exceeded

Certain timing constellations must be distinguished when a frame is received (distinction in relation to reaction and error frames). The timing behaviour of nodes shall be monitored, otherwise the entire cycle time can exceed.

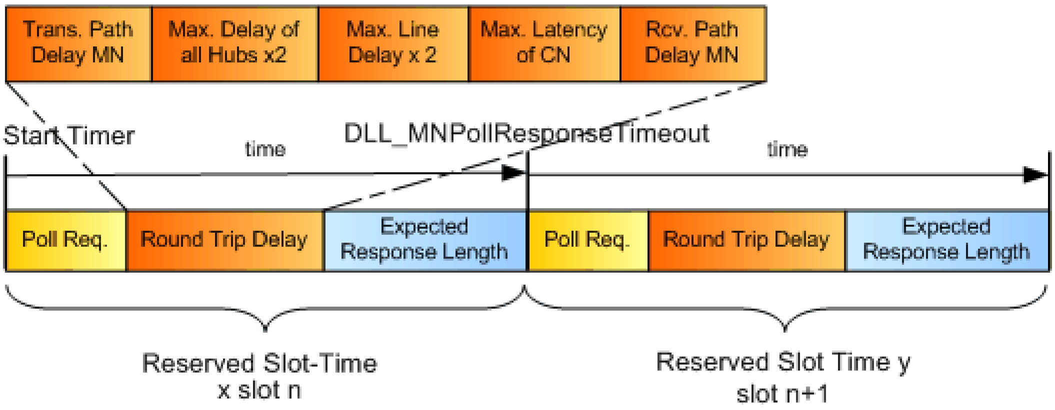


Figure 28 – Timeouts

The PollResponse frame of the CN must have been received completely before the slot time expires. Therefore the monitoring of the reserved slot time is necessary. The DLL_MNPollResponseTime is a configurable parameter for each CN. The violation of the slot time produces the situations described below.

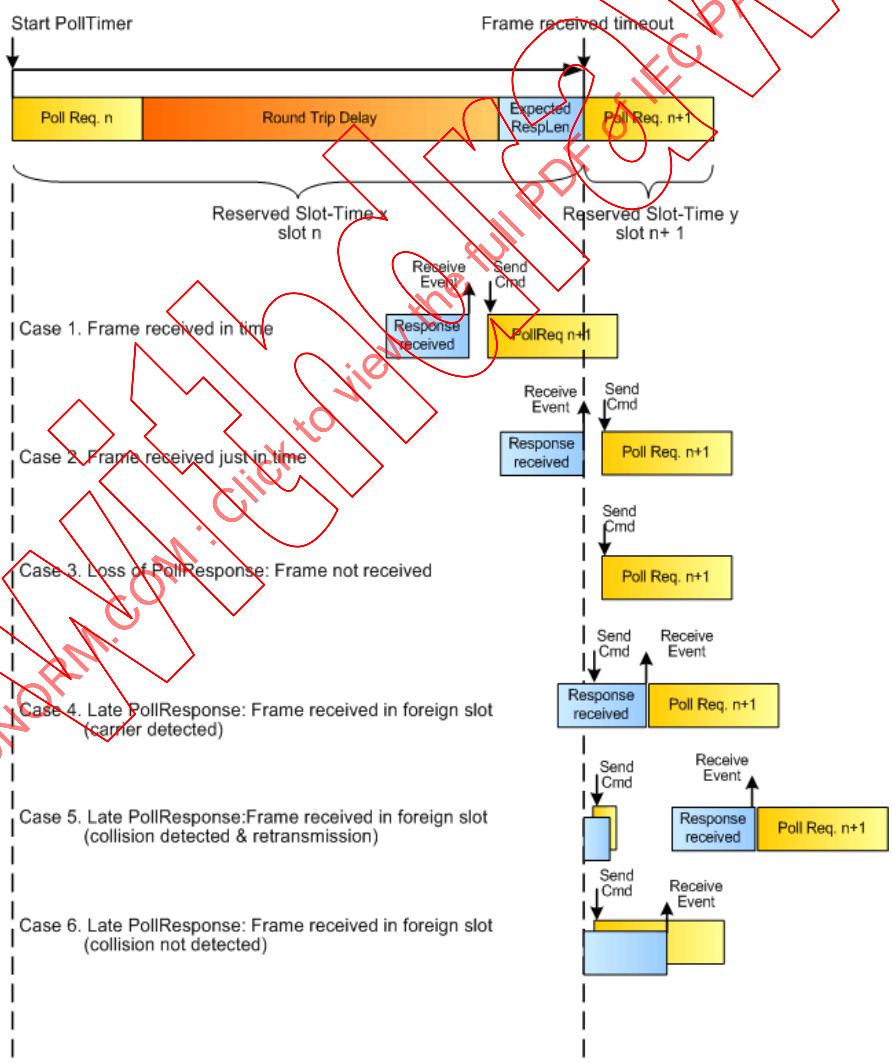


Figure 29 – Timing violation

4.7.7.1.1.1 Case 1-2 Frame received in time

The behaviour in the first two cases is identical. The second case shows the latest possible time for the frame receipt. The current slot timer is switched off immediately after the frame is received.

4.7.7.1.1.2 Case 3 Loss of PollResponse: Frame not received

The Loss of a PollResponse frame is detected by the PollTimeout. If no frame was received, till the timeout occurs, the PollResponse frame of a CN was lost.

4.7.7.1.1.3 Case 4-6 Late PollResponse: Frame received in foreign slot (also collisions)

The cases 4-6 have one thing in common: A frame is received, which doesn't belong to the current slot. The worst case could lead to a violation of the cycle. This kind of errors can disturb the entire power link communication. Only in case 4 and 6 a direct detection of a Late PollResponse error can be detected. In case 5 a collision occurs as a result of a Late PollResponse error.

4.7.7.2 Loss of PollResponse

- **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 EPL 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)

- **Handling**

  After detecting a CN by the DLL_MNPollResponseTimeout, the MN proceeds with the PollRequest for the next CN (or SoA if the end is reached). If a CN fails more than a configurable number (DLL_MNCNLossPResThrLim_AU32[CN NodeID]) of consecutive cycles, the MN NMT State machine considers the CN inactive and removes it from the isochronous processing. Additionally it sends to the respective CN the command "NMTResetNode". On this command the CN will change its state (See 7.1.4). Whenever this error symptom is detected, the MN checks for a physical layer error source. The loss of frames in the asynchronous communication aren't detected.

- **Registration**

  The MN shall implement for each CN in the EPL network a threshold counter and optional a cumulative counter. Every loss of a PollResponse frame increments the threshold counter (DLL_MNCNLossPResThrCnt_AU32[CN NodeID]) and if implemented the cumulative counter (DLL_MNCNLossPResCumCnt_AU32[CN NodeID]) of the respective CN. The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LOSS_PRES_TH	XXXX	

4.7.7.3 Late PollResponse

- **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 EPL 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 PollResponse error symptom may be detected on the MN, when while trying to send the PollRequest frame the carrier is busy (See Slot Time exceeded: Case 4) or when it receives a PollResponse frame from an unexpected CN (See Slot Time exceeded: Case 6).

- **Handling**

  After detecting a Late PollResponse error, the MN proceeds with the PollRequest for the next CN (or SoA if the end is reached). If a CN fails more than a configurable number (DLL_MNCNLatePResThrLim_AU32[CN NodeID]) of consecutive cycles, the MN NMT State machine considers the CN inactive and removes it from the isochronous processing. Additionally it sends to the respective CN the command "NMTResetNode". On this command the CN changes its state (See 7.1.4). If a PollResponse frame, which does not belong to the current slot, is received, the frame must be rejected in any case. Previously the MN has to determine the sender. For the collection of multiple assigned node addresses a response counter has to be incremented (see multiple used node numbers).

  The Frame length violation and answer latency violation could be computed from the frame receipt time and the frame length. Whenever this error symptom is detected, the MN checks for a physical layer error source. The loss of frames in the asynchronous communication aren't be detected.

- **Registration**

  The MN shall implement for each CN in the EPL network a threshold counter and optional a cumulative counter. Every occurrence of a Late PollResponse error symptom increments the threshold counter (DLL_MNCNLatePResThrLim_AU32[CN NodeID]) and if implemented the cumulative counter (DLL_MNCNLatePResThrLim_AU32[CN NodeID]) of the respective CN. The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LATE_PRES_TH	XXXX	

4.7.7.4 Cycle Time Exceeded

- **Error source**

  Following possible error sources could cause this error symptom:

  EPL 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 EPL 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.

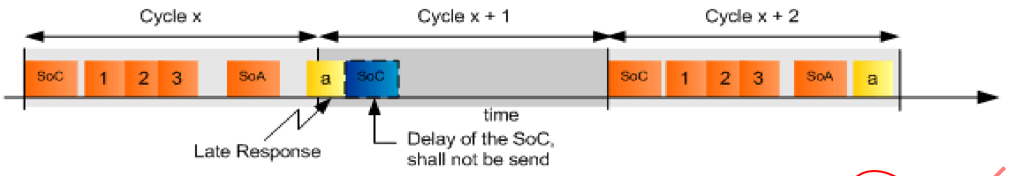


Figure 30 – Cycle Time exceeded

To prevent this timing violation it is necessary that the MN always controls the cycle time before sending a frame. If there is not enough time left to send and receive a frame it drops the invitation. This is an optionally behaviour. Normally the cycle should be dimensioned for the worst case (with max response times / timeouts).

According to the polltimer (DLL_MNPollResponseTimeout) a configurable max asynchronous slot time (DLL_MNAsyncSlotTime_U32) is necessary to control the asynchronous communication. Prior to sending the SoA telegram, the MN checks whether the current time plus DLL_MNAsyncSlotTime_U32 exceeds the cycle time. The MN detects a cycle time exceeded timing violation, when the carrier is busy, while trying to send a SoC frame.

- **Handling**

A cycle time violation is considered a configuration error, hence the default behaviour is to suspend one cycle and log the error symptom (Error Code: E_DLL_CYCLE_EXCEED) in the Error History (Object ERR_History_ADOM). Optionally a threshold counter and a cumulative counter may be implemented. Every time the threshold value is reached or, if no Threshold counter implemented, every time the error symptom occurs, the MN NMT state machine is notified. It handles this error source as an "error condition" (NMT_MT6) and changes its state to NMT_MS_PRE_OOPERATIONAL_1.

- **Registration**

The MN may implement a threshold counter and a cumulative counter. Every occurrence of a Cycle Time exceeded error symptom will increment the threshold (DLL_MNCycTimeExceedThrCtr_U32) and the cumulative counter (DLL_MNCycTimeExceedCumCtr). The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached (Error Code: E_DLL_CYCLE_EXCEED_TH).

History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_CYCLE_EXCEED	XXXX	
0x01	XXXX	E_DLL_CYCLE_EXCEED_TH	XXXX	

4.7.7.5 Collisions

- **Error Source**

ETHERNET powerlink doesn't depend on the discovery of collisions. Because standard Ethernet controllers are used, collisions can be detected only in some cases. The number of hubs in the EPL network isn't limited on two as in Fast Ethernet. Because of that the collision domain is violated.

In the operational mode no collisions should occur due to the EPL cycle design. If a node doesn't follow 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 CNs.

- **Error recognition**

  If the Ethernet controller discovers a collision in the EPL network, it starts the standard Ethernet procedure for collisions.

- **Handling**

  The MN logs the error symptom (Error Code: E_DLL_COLLISION) in the Error History (Object ERR_History_ADOM) and suspends for a configurable number (DLL_MNCycleSuspendNumber_U32) of cycles (min. rest of the cycle), before continuing again with the isochronous and asynchronous communication. The MN data link layer state machine changes its state to DLL_MS_ WAIT_SOC_TIME. Optionally a threshold counter and a cumulative counter may be implemented. Every time the threshold value is reached or, if no Threshold counter implemented, every time the error symptom occurs, the MN NMT state machine is notified. It handles this error source as an "internal Communication Error (NMT_GT6)" and changes its state to NMT_GS_RESET_COMMUNICATION.

- **Registration**

  The MN may implement a threshold counter and a cumulative counter. Every occurrence of a Collision error symptom increments the threshold (DLL_MNCollisionThrCtr_U32) and the cumulative counter (DLL_MNCollisionCumCtr). The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached (Error Code: E_DLL_COLLISION_TH).

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_COLLISION	XXXX	
0x01	XXXX	E_DLL_COLLISION_TH	XXXX	

4.7.7.6 Invalid Formats

- **Error Source**

  Invalid Formats can result from software faults or hardware errors in the nodes. Data falsifications during the transmission do not belong to this point. This kind of failure is recognised and the frame filtered by the Ethernet controller if a false CRC sum is detected. Format Errors should only be detected by Poll Response frames. Format Errors in the asynchronous communication shall be ignored. Invalid Format Errors can also occur if there are various firmware versions within the EPL network. In that case nodes may not support frame formats of other nodes.

- **Error recognition**

  An invalid Format error symptom occurs if an EPL frame header contains an unsupported value. This could be a false ServiceID (not defined), Node number, etc. An invalid format error is also caused, if the received frame size is larger than the predicted buffer input size.

- **Handling**

  If a CN causes an invalid format error, the MN NMT State machine considers the CN inactive and removes it from the isochronous processing. Additionally it sends to the respective CN the command "NMTResetNode". On this command the CN will change its state (See 7.1.4). The error symptom is logged, every time it occurs.

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_INVALID_FORMAT	XXXX	

4.7.7.7 EPL 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 Broadcast (SoA frame) to a Unicast powerlink address, several CNs with the same powerlink address are able to respond.

- **Error recognition**

  The MN detects multiple used EPL addresses in a network in the way it counts the responses on an AsyncIdent frame.

- **Handling**

  If the MN detects that multiple CNs cause EPL address conflicts, the MN NMT State machine considers the involved CNs inactive and removes them from the configuration. The error symptom is logged, every time it occurs.

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_ADDRESS_CONFLICT	XXXX	Status of the MN

4.7.7.8 Multiple MNs on a single EPL Network

- **Error source**

  Before a MN starts the communication, it waits in the state NMT_MS_NON_ACTIVE . In this time it observers the EPL network whether another MN is already running. If multiple MNs start simultaneous the communication, a combination of error symptoms indicates the circumstance. It causes with high probability multiple error symptoms on the MN. For example:

  - Collisions errors
  - Cycle time exceeded
  - Etc.

- **Error recognition**

  Contrary to a standard EPL cycle a further MN receives a SoC or SoA Frame within a cycle.

- **Handling**

  If a new MN is in the state NMT_MS_NON_ACTIVE and detects that already another MN is running, it shall reside in its state. If multiple MNs start the communication simultaneously and a 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 is logged every time it occurs.

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_MULTIPLE_MN	XXXX	

4.7.8 Communication Error Symptoms detected by the CN

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

4.7.8.1 Collisions

- **Error Source**

  ETHERNET powerlink doesn't depend on the discovery of collisions. Because standard Ethernet controllers are used, collisions can be detected only in some cases. The number of hubs is in the EPL network not limited on two as in Fast Ethernet, thus the collision domain is violated (i.e. so called " late collisions" may occur).

  In the operational mode no collisions should occur because of the EPL cycle design. If a node doesn't fullfill 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 CNs.

- **Error recognition**

  If the Ethernet controller discovers a collision in the power link network, it starts the standard Ethernet procedure for collisions.

- **Handling**

  If the CN is able to detect a Collision a threshold counter and optionally cumulative counter may be implemented. Every time the threshold value is reached, the MN NMT state machine is notified. It handles this error source as an "internal Communication Error (NMT_GT6)" and changes its state to NMT_GS_RESET_COMMUNICATION.

- **Registration**

  The MN may implement a threshold counter and a cumulative counter. Every occurrence of a Collision error symptom increments the threshold (DLL_CNCollisionThrCtr_U32) and the cumulative counter (DLL_CNCollisionCumCtr). The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached (Error Code: E_DLL_COLLISION_TH).

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_COLLISION_TH	XXXX	

4.7.8.2 Invalid Formats

- **Error Source**

  Invalid Formats can result from software faults or hardware errors in the nodes. Data falsifications during the transmission do not belong to this point. This kind of failure is recognised and the frame filtered by the Ethernet controller if a false CRC sum is detected. Format Errors should only be detected by Poll Response frames. Format Errors in the asynchronous communication shall be ignored. Invalid Format Errors can also occur if there are various firmware versions within the EPL network. In that case nodes may not support frame formats of other nodes.

- **Error recognition**

  An invalid Format error symptom occurs if an EPL frame header contains an unsupported value. This could be a false ServiceID (not defined), Node number, etc. An invalid format error is also caused, if the received frame size is larger than the predicted buffer input size.

- **Handling**

  If a CN detects an invalid format error, it notifies 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 every time it occurs.

- **Registration**

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_INVALID_FORMAT	XXXX	

4.7.8.3 Loss of Frames

A CN considers a cycle as valid if it receives SoC frame, PollRequest frame and SoA frame. The CN can only detect the loss of frames, which were sent from the MN.

- **Error source**

  Following 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 EPL network
      - Frame Collision
    - A component of the network structure is defect.
    - Power failure on a CN or a MN
    - Etc.

4.7.8.3.1 Loss of SoC

- **Error recognition**

  If the CN receives a frame, which isn't expected in the current state of the CN DLL (data link layer) state machine or it receives multiple PReq within a cycle, a SoC frame has been lost. Another possibility to detect a loss of SoC is by a timeout. The timer is reset on every receipt of a SoC frame.

- **Handling**

  If the CN misses the SoC-frame, it still replies on any invitation with the data of the previous cycle. Invalid data shall not be sent in any case. The CN accept the new isochronous or asynchronous data.

  If the CN gets for a configurable number (DLL_MNCNLossSoCThr_U32) of consecutive cycles no SoC frame, it notifies its NMT state machine and logs the error symptom. The CN NMT state machine handles this error source as an "error condition (NMT_CT11)" and changes its state to NMT_CS_PRE_OPERATIONAL_1. Whenever this error symptom is detected, the CN checks for a physical layer error source.

- **Registration**

  The CN shall implement a threshold counter and a cumulative counter. Every loss of a SoC frame will increment the threshold counter (DLL_CNLossSoCThrCtr_U32) and the cumulative counter (DLL_CNLossSoCCumCtr). The error symptom will be logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LOSS_SOC_TH	XXXX	

4.7.8.3.2 Loss of SoA

- **Error recognition**

  If the CN doesn't receive a SoA frame within a cycle, the frame has been lost.

- **Handling**

  If the CN detects a loss of SoA frame, it increments a threshold counter and a cumulative counter. The error symptom will be logged and its NMT state machine notified after a configurable number (DLL_CNLossSoAThr_U32) of error events. The CN NMT state machine handles this error source as an "error condition (NMT_CT11)" and changes its state to NMT_CS_PRE_OPERATIONAL_1. Whenever this error symptom is detected, the CN checks for a physical layer error source.

- **Registration**

  The CN may implement a threshold counter or a cumulative counter. Every loss of a SoA frame will increment the threshold counter (DLL_CNLossSoAThrCtr_U32) and the cumulative counter (DLL_CNLossSoACumCtr). The error symptom will be logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LOSS_SOA_TH	XXXX	

4.7.8.3.3 Loss of PollRequest

- **Error recognition**

  If a CN, which is included in the isochronous communication doesn't receive a PReq frame within a cycle, the frame has been lost.

- **Handling**

  If the CN detects a loss of PollRequest frame, it increments a threshold counter and a cumulative counter. The error symptom will be logged and its NMT state machine notified after a configurable number (DLL_CNLossPReqThr_U32) of error events. The CN NMT state machine handles this error source as an "error condition (NMT_CT11)" and changes its state to NMT_CS_PRE_OPERATIONAL_1. Until the CN NMT state machine changes its state it still keeps on with the communication and listen to the cross traffic. Whenever this error symptom is detected, it checks for a physical layer error source.

- **Registration**

  The CN may implement a threshold counter or a cumulative counter. Every loss of a PollRequest frame increments the threshold counter (DLL_CNLossPReqThrCtr_U32) and the cumulative counter (DLL_CNLossPReqCumCtr). The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_LOSS_PREQ_TH	XXXX	

4.7.8.4 SoC Jitter out of Range

- **Error source**

  This error could have various error sources:
  - Jitter on not EPL conform network components
  - Collisions in the EPL network
  - Failure during the transmission
  - MN failures
  - Late response of CN causes a delay of the SoC
  - Etc.

- **Error recognition**

  Each CN may control the SoC cycle time jitter. Every time it receives the SoC frame, it measures the cycle time. A SoC jitter error occurs, when the measured time is out of a configurable range (DLL_CNSoCJitterRange_U32).

- **Handling**

  If the CN detects that SoC Jitter is out of a specified range (DLL_CNSoCJitterRange_U32), it increments a threshold counter and a cumulative counter. The error symptom is logged and its NMT state machine will be notified after a configurable number (DLL_CNSoCJitterThr_U32) of error events. The CN NMT state machine handles this error source as an “error condition (NMT_CT11)” and changes its state to NMT_CS_PRE_OPERATIONAL_1. The received data until the NMT state machine changes its state is still valid.

- **Registration**

  The CN may implement a threshold counter or a cumulative counter. Every loss of a PollRequest frame increments the threshold counter (DLL_CNSoCJitterThrCtr_U32) and the cumulative counter (DLL_CNSoCJitterCumCtr). The error symptom is logged in the Error History (ERR_History_ADOM) every time the threshold value is reached.

  History Entry Object ERR_History_ADOM:

Mode	Vendor ID	Error Code	Timestamp	Additional Information
0x01	XXXX	E_DLL_JITTER_TH	XXXX	

4.7.9 Error Handling Parameters

In this subclause the for the error handling used parameters are descripted.

4.7.9.1 Object 1C00$_h$: 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	1C00h
Name	DLL_MNCRCError_REC
Object Code	RECORD
Category	M

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_MNCRCErrorCum_U32**

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

Sub-Index	1_h
Description	DLL_MNCRCErrorCum_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCRCErrorThr_U32**

The threshold counter shall be incremented every time a CRC error occurs on the MN and decremented after every successful cycle. Its value monitors the quality of network in relation to the CRC error occurrence.

Sub-Index	2_h
Description	DLL_MNCRCErrorThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCRCErrorThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.6.1)

Sub-Index	3_h
Description	DLL_MNCRCErrorThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.2 Object $1C01_h$: 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	$1C01_h$
Name	DLL_MNCollision_REC
Object Code	RECORD
Category	O

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_MNCollisionCum_U32**

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

Sub-Index	1_h
Description	DLL_MNCollisionCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCollisionThr_U32**

The threshold counter shall be incremented every time a Collision error symptom occurs on the MN and decremented after every successful cycle. Its value monitors the quality of network in relation to the Collision error occurrence.

Sub-Index	2_h
Description	DLL_MNCollisionThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCollisionThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.7.5)

Sub-Index	3_h
Description	DLL_MNCollisionThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.3 Object 1C02$_h$: 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	1C02h
Name	DLL_MNCycTimeExceed_REC
Object Code	RECORD
Category	O

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_MNCycTimeExceedCum_U32**

The cumulative counter shall be incremented 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	1_h
Description	DLL_MNCycTimeExceedCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCycTimeExceedThr_U32**

The threshold counter shall be incremented every time a “Cycle Time exceeded” error symptom occurs on the MN and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Cycle Time exceeded error occurrence.

Sub-Index	2_h
Description	DLL_MNCycTimeExceedThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_MNCycTimeExceedThreshold**

  Every time the threshold is reached, a defined action shall proceed. (See 4.7.7.4)

Sub-Index	3_h
Description	DLL_MNCycTimeExceedThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.4 Object $1C10_h$: DLL_CNLossOfLink_REC

The following objects are used to monitor the "Loss of Link" error source. The record consists of a cumulative counter data object.

Index	$1C10_h$
Name	DLL_CNLossOfLink_REC
Object Code	RECORD
Category	O

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0h -4h
Default Value	3h

- **Sub-Index 1h: DLL_MNLossOfLinkCum_U32**

  The cumulative counter shall be incremented every time a "Loss of Link"error symptom occurs. Its value monitors all "Loss of Link" error sources that were detected by the MN.

Sub-Index	1_h
Description	DLL_MNLossOfLinkCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.5 Object 1C04$_h$: DLL_MNCNLatePResCumCnt_AU32

This Array on the MN contains for every CN in the EPL network a cumulative counter of Late PRes events.

The cumulative counter of the respective CN shall be incremented every time a “Late PollResponse“error symptom occurs. Its value monitors all “Late PollResponse” error symptoms that were detected by the MN.

Index	1C04$_h$
Name	DLL_MNCNLatePResCumCnt_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	O

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: CumCnt**

Sub-Index	01$_h$ – FE$_h$
Description	CumCnt
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.6 Object 1C05$_h$: DLL_MNCNLatePResThrCnt_AU32

This Array on the MN contains for every CN in the EPL network a threshold counter of late Pres.

The threshold counter of the respective CN shall be incremented every time a “Late PollResponse” error symptom occurs on the MN and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Late PollResponse” error occurrence.

Index	1C05$_h$
Name	DLL_MNCNLatePResThrCnt_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: ThrCnt**

Sub-Index	01$_h$ – FE$_h$
Description	ThrCnt
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.7 Object 1C06$_h$: DLL_MNCNLatePResThrLim_AU32

This Array on the MN contains for every CN in the EPL network a threshold limit of late Pres.

Every time the respective DLL_MNCNLatePResThrCnt_AU32 value reaches the correponding limit, a defined action shall proceed. (See 4.7.7.3)

Index	1C06$_h$
Name	DLL_MNCNLatePResThrLim_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: ThrLim**

Sub-Index	01$_h$ – FE$_h$
Description	ThrLim
Type	UNSIGNED32
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.8 Object 1C07$_h$: DLL_MNCNLossPResCumCnt_AU32

This Array on the MN contains for every CN in the EPL network a cumulative counter of Loss of PRes events.

The cumulative counter of the respective CN shall be incremented 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$_h$
Name	DLL_MNCNLossPResCumCnt_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	O

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: CumCnt**

Sub-Index	01$_h$ – FE$_h$
Description	CumCnt
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.9 Object 1C08$_h$: DLL_MNCNLossPResThrCnt_AU32

This Array on the MN contains for every CN in the EPL network a threshold counter of Loss of PRes.

The threshold counter of the respective CN shall be incremented every time a “Loss of PollResponse” error symptom occurs on the MN and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Loss of PollResponse” error occurrence.

Index	1C08$_h$
Name	DLL_MNCNLossPResThrCnt_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: ThrCnt**

Sub-Index	01$_h$ – FE$_h$
Description	ThrCnt
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.10 Object 1C09$_h$: DLL_MNCNLossPResThrLim_AU32

This Array on the MN contains for every CN in the EPL network a threshold limit of late Pres.

Every time the respective DLL_MNCNLatePResThrCnt_AU32 value reaches the correponding limit, a defined action shall proceed. (See 4.7.7.2)

Index	1C09$_h$
Name	DLL_MNCNLossPResThrLim_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	01$_h$ – FE$_h$
Default Value	FE$_h$

- **Sub-Index 01$_h$ – FE$_h$: ThrLim**

Sub-Index	01$_h$ – FE$_h$
Description	ThrLim
Type	UNSIGNED32
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.11 Object 1C0A$_h$: 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.

Index	1C0A$_h$
Name	DLL_CNCollision_REC
Object Code	RECORD
Category	O

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNCollisionCum_U32**

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

Sub-Index	1$_h$
Description	DLL_CNCollisionCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNCollisionThr_U32**

The threshold counter shall be incremented every time a “Collision”error symptom occurs and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Collision” error occurrence.

Sub-Index	2$_h$
Description	DLL_CNCollisionThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNCollisionThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.8.1)

Sub-Index	3$_h$
Description	DLL_CNCollisionThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.12 Object 1C0B$_h$: 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$_h$
Name	DLL_CNLossSoC_REC
Object Code	RECORD
Category	M

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNLossSoCCum_U32**

The cumulative counter shall be incremented 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	1$_h$
Description	DLL_CNLossSoCCum_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNLossSoCThr_U32**

The threshold counter shall be incremented every time a “Loss of SoC”error symptom occurs and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Loss of SoC” error occurrence.

Sub-Index	2$_h$
Description	DLL_CLossSoCThr_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3h: DLL_CNLossSoCThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.8.3.1)

Sub-Index	3$_h$
Description	DLL_CNLossSoCThreshold
Type	UNSIGNED32
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.13 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.

Index	$1C0C_h$
Name	DLL_CNLossSoA_REC
Object Code	RECORD
Category	Optionally

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNLossSoACum_U32**

The cumulative counter shall be incremented 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	1_h
Description	DLL_CNLossSoACum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNLossSoAThr_U32**

The threshold counter shall be incremented every time a “Loss of SoA”error symptom occurs and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Loss of SoA” error occurrence.

Sub-Index	2_h
Description	DLL_CLossSoAThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3h: DLL_CNLossSoAThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.8.3.2)

Sub-Index	3_h
Description	DLL_CNLossSoAThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.14 Object 1C0D$_h$: DLL_CNLossPReq_REC

The following objects are used to monitor “Loss of PReq” 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	1C0D$_h$
Name	DLL_CNLossPReq_REC
Object Code	RECORD
Category	Optionally

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNLossPReqCum_U32**

The cumulative counter shall be incremented 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	1$_h$
Description	DLL_CNLossPReqCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNLossPReqThr_U32**

The threshold counter shall be incremented every time a “Loss of PReq”error symptom occurs and decremented after every successful cycle. Its value monitors the quality of network in relation to the “Loss of PReq” error occurrence.

Sub-Index	2$_h$
Description	DLL_CLossPReqThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3h: DLL_CNLossPReqThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.8.3.3)

Sub-Index	3$_h$
Description	DLL_CNLossPReqThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.15 Object 1C0E$_h$: 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.

Index	1C0E$_h$
Name	DLL_CNSoCJitter_REC
Object Code	RECORD
Category	Optionally

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNSoCJitterCum_U32**

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

Sub-Index	1$_h$
Description	DLL_CNSoCJitterCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNSoCJitterThr_U32**

The threshold counter shall be incremented every time a “SoC Jitter”error symptom occurs and decremented after every successful cycle. Its value monitors the quality of network in relation to the “SoC Jitter” error occurrence.

Sub-Index	2$_h$
Description	DLL_CSoCJitterThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNSoCJitterThreshold**

Every time the Threshold is reached, a defined action shall proceed. (See 0)

Sub-Index	3$_h$
Description	DLL_CNSoCJitterThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.16 Object 1C0F$_h$: DLL_CNCRCError_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$_h$
Name	DLL_CNCRCError_REC
Object Code	RECORD
Category	Mandatory

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

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNCRCErrorCum_U32**

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

Sub-Index	1$_h$
Description	DLL_CNCRCErrorCum_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNCRCErrorThr_U32**

The threshold counter shall be incremented every time a CRC error occurs on the CN and decremented after every successful cycle. Its value monitors the quality of network in relation to the CRC error occurrence.

Sub-Index	2$_h$
Description	DLL_MNCRCErrorThr_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2h: DLL_CNCRCErrorThreshold**

Every time the threshold is reached, a defined action shall proceed. (See 4.7.6.1)

Sub-Index	3$_h$
Description	DLL_MNCRCErrorThreshold
Type	UNSIGNED32
Entry Category	Optionally
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.17 Object $1C10_h$: DLL_CNLossOfLink_REC

The following objects are used to monitor the “Loss of Link” error source. The record consists of a cumulative counter data object.

Index	$1C10_h$
Name	DLL_CNLossOfLink_REC
Object Code	RECORD
Category	Optionally

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

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	0h -4h
Default Value	3h

- **Sub-Index 1h: DLL_CNLossOfLinkCum_U32**

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

Sub-Index	1_h
Description	DLL_CNLossOfLinkCum_U32
Type	UNSIGNED32
Entry Category	Optionally
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

4.7.9.18 Object $1C11_h$: DLL_MNAsyncSlotTimeout_U32

This Timeout data object is used to monitor the asynchronous slot, similar to the polltimeout (See 4.7.7.1).

Index	$1C11_h$
Name	DLL_MNAsyncSlotTimeout_U32
Object Code	USIGNED32
Category	Optionally
Access	rw
PDO Mapping	No
Default Value	

After receiving the SoA (or sending in case of MN) the station which is invited sends its asynchronous frame (ASnd or IP-Frame). Some frame fields has to be compared (e.g. node numbers) to decide whether the local node has the right to send. The Time a node needs to process the received frame, including HW latencies (e.g. DMA transfer). This time should be as short as possible. The received data should not be interpreted in any way but e.g. copied to some internal buffer for further processing at lower priority. The maximum of all latencies has to be considered for calculating the DLL_MNAsyncSlotTimeout_U32:

DLL_MNAsyncSlotTimeout_U32 := MNSendLatency + 2* NetTransmitTime(17µs :10Hubs+600m Cable) + ASndLatenceTime + AsyncSendDataSlotSize / 100MBit

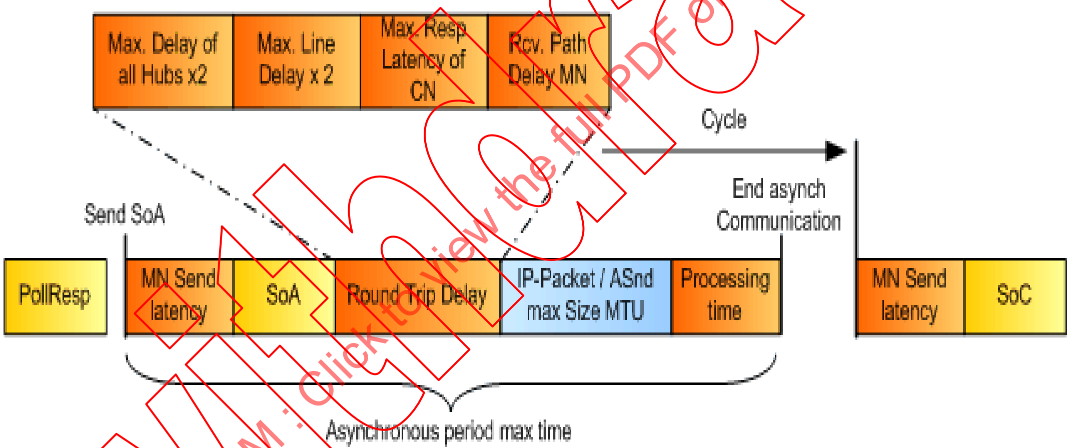


Figure 31 – AsyncSlot timeout

4.7.9.19 Object 1C12$_h$: DLL_MNCycleSuspendNumber_U32

The DLL_MNCycleSuspendNumber_U32 parameter is used to define the number of cycle that will be suspended, when a collision is occurred

Index	1C12$_h$
Name	DLL_MNCycleSuspendNumber_U32
Object Code	UNSIGNED32
Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	1 (0 means that it will finish the current cycle and continue with the followed cycle; 1 means, that it suspends the followed cycle; and so on)

4.7.9.20 Object 1C13$_h$: DLL_CNSoCJitterRange_U32

The DLL_CNSoCJitterRange_U32 parameter is used to define the range within the SoCJitter can vary.

Index	1C13$_h$
Name	DLL_CNSoCJitterRange_U32
Object Code	USIGNED32
Category	Optionally
Access	rw
PDO Mapping	No
Value Range	USIGNED32 in ns
Default Value	

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 period.

- 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 an EPL node implementation of the Internet Protocol. To use IP transparently in the asynchronous period, the EPL nodes shall be conformed to RFC1122 “Requirements for Internet Hosts -- Communication Layers”. However, this would prohibit several low-end EPL nodes, communicating with IP in the asynchronous period. Therefore the following conformance classes are introduced.

5.1.1.1 Nodes without IP Communication

Nodes that neither need SDO nor IP communication do not need an IP stack.

5.1.1.2 Minimum Requirements for SDO Communication

This conformance class shall be fulfilled, to ensure that an EPL node is able to communicate in the asynchronous period 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 period, the EPL node shall at least cope with 256 Bytes SDO payload. Therefore, the IP stack shall process at least C_IP_MINIMUM_STACK_SIZE 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 period shall be equal or bigger than 256 Bytes SDO payload.

5.1.1.2.2 UDP Requirements

An EPL 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 EPL 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.

5.1.2 IP Addressing

Each IP-capable EPL 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 an EPL network – see RFC1918. A class C network provides 254 (1-254) IP addresses, which matches the number of valid EPL Node ID's. The Host ID of the private class C Net ID 192.168.100.0 shall be identical to the EPL Node ID. Hence the last byte of the IP address (Host ID) has the same value as the EPL Node ID. The following figure illustrates the construction of the IP address.

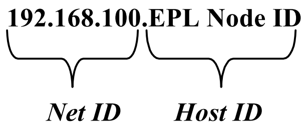


Figure 32 – Construction of the IPv4 address

  Remarks:

  Knowing the Node ID of an EPL node, its IP address and vice versa can be determined easily without any communication overhead.

- **Subnet mask**

  The subnet mask of an EPL node shall be 255.255.255.0. This is the subnet mask of a class C net.

- **Default Gateway**

  The default Gateway shall use the IP address 192.168.100.254.

Generally the IP parameters of an EPL node shall be fixed. The downside of the fixed IP parameters are compensated by the EPL Router using Network Address Translation (see 8.1.4.2.2).

The following table summarises the default IP parameters.

Table 25 – IP Parameters of an EPL Node

IP Parameter	IP address
IP address	192.168.100.<EPL Node ID>
Subnet mask	255.255.255.0
Default Gateway	192.168.100.254

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 an EPL node. Depending on the EPL node state:

- NMT_CS_EPL_MODE and NMT_MS_EPL_MODE state: ARP shall be performed in the asynchronous period. To reduce the traffic in the asynchrinus perid, 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* 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 EPL network. The EPL node may process *NMTFlushArpEntry*.

Alternativley an *unsolicited ARP request* frame (containing its IP address) may be broadcasted initiated by the respective EPL node at startup. As a result, the neighbours ARP caches shall be updated.

5.1.4 .Hostname

Each IP capable EPL node shall have a hostname. The hostname is of type VISIBLE_STRING15. The hostname can be used to access EPL nodes with its name instead of its IP address.

The admissible values of type VISIBLE_STRING for the hostname shall be restricted to:

- 30_h - 39_h (0 - 9)
- 41_h - $5A_h$ (A - Z)
- 61_h - $6A_h$ (a - z)
- $2D_h$ (-)

The data are interpreted as ISO 646-1973(E) 7-bit coded characters.

The *default* hostname shall be constructed from the EPL Node ID and the Vendor ID parted by the character "-" (<EPL Node ID>-<Vendor ID>). If no hostname is explicitly assigned, the EPL node shall use the *default* hostname instead.

The hostname located on the EPL node shall be set with the NMT Managing command *NMTNetSetHostname* (refer 7.4.2.1.2). Modification of the hostname value shall not take effect until the EPL node enters the NMT_GS_INITALISATION 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 EPL nodes within an EPL network. This service configures for example the DNS table of the DNS server located on the EPL to legacy Ethernet Router or a local hostname table on a diagnosis device.

5.1.5 Object description

5.1.5.1 Object $1E4B_h$: NWL_IpGroup_REC

The NWL_IpGroup_REC object is a subset of the IP Group RFC1213. The object specifies information about the IP stack.

Index	$1E4B_h$
Name	NWL_IpGroup_REC
Object Code	RECORD
Data Type	NWL_IpGroup_TYPE
Category	Conditional for IP capable nodes

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	**00_h**
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	3
Default value	3

- **Sub-Index 01_h: Forwarding_BOOL**

  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).

Sub-Index	**01_h**
Description	Forwarding_BOOL
Data Type	BOOL
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	Not-forwarding(0) forwarding(1)
Default value	Not-forwarding(0)

- **Sub-Index 02_h: DefaultTTL_U16**

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	**02_h**
Description	DefaultTTL_U16
Data Type	UNSIGNED16
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	UNSIGNED16
Default value	64

- **Sub-Index 03_h: ForwardDatagrams_U32**

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.

Sub-Index	**03_h**
Description	ForwardDatagrams_U32
Data Type	UNSIGNED32
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	-

5.1.5.2 Object $1E40_h$ – $1E4F_h$: 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 10 entries that may be configured via SDO.

Index	$1E40_h$ – $1E4F_h$
Name	NWL_IpAddrTable_*Xh*_REC
Object Code	RECORD
Data Type	NWL_IpAddrTable_*Xh*_TYPE
Category	M

- **Sub-Index 00$_h$: NumberOfEntries_U8**

Sub-Index	**00$_h$**
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	4
Default value	4

- **Sub-Index 01$_h$: IfIndex_U16**

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_ItfGroup_*Xh*_REC.IfIndex_U16.

Sub-Index	**01$_h$**
Description	IfIndex_U16
Data Type	UNSIGNED16
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	UNSIGNED16
Default value	-

- **Sub-Index 02$_h$: Addr_IPAD**

The IP address to which this entry's addressing information pertains.

Sub-Index	**02$_h$**
Description	Addr_IPAD
Data Type	IP_ADDRESS
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	IP_ADDRESS
Default value	-

- **Sub-Index 03$_h$: NetMask_IPAD**

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.

Sub-Index	**03$_h$**
Description	NetMask_IPAD
Data Type	IP_ADDRESS
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	IP_ADDRESS
Default value	-

- **Sub-Index 04$_h$: ReasmMaxSize_U16**

The size of the largest IP datagram which this entity can re-assemble from incoming IP fragmented datagrams received on this interface.

Sub-Index	**04_h**
Description	ReasmMaxSize_U16
Data Type	UNSIGNED16
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	UNSIGNED16
Default value	-

5.2 EPL conformant UDP/IP format

In order to enable the transmission of EPL frames encapsulated in UDP/IP frames, the payload portion of the UDP/IP frame shall be leaded by a slightly modified EPL frame header.

Table 26 – EPL conformant UDP/IP frame structure

	Bit Offset								**entry defined by**
Octet Offset [7]	**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	reserved(ServiceID ?)								
46 - n	Payload Data								Application
n+1 - n+4	CRC32								Ethernet type II

$n \geq 59$

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 EPL header shall be reserved but shall not be supported, when transmission occurs via UDP/IP.

5.3 EPL Sequence Layer

Refer to 6.3.3.

[7] 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 = $88AB_h$). The rules shall be valid for the EPL-Content, service specific header and data payload embedded into the Ethernet frame. The encoding of the Ethernet frame shall follow the rules of IEEE 802.3.

Figure 33 – EPL frame structure

The rules shall be furthermore valid for EPL specific payload, that are embedded into Non-EPL frame types (EtherType ≠ $88AB_h$), e.g. SDO-Transfer via UDP/IP etc. The Ethertype specific encoding of these frames, described by RFC 791 (IP) and RFC 768 (UDP), shall not be concerned by these rules.

Figure 34 – EPL conformant UDP/IP frame structure

The encoding of Non-EPL frames (EtherType ≠ $88AB_h$) without EPL specific payload shall not be concerned by these rules.

Figure 35 – Legacy Ethernet frame structure

6.1.1 General Description of Data Types and Encoding Rules

To be able to exchange meaningful data across the EPL network, the format of this data and its meaning have to be known by the producer and consumer(s). This specification models this by the concept of data types.

The encoding rules define the representation of values of data types and the EPL 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 Table 27.

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).

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 IEC 61131-3).

data_definition	::=	type_name data_name
type_definition	::=	constructor type_name
constructor	::=	compound_constructor | 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’ | ‘VOID’ bit_size | ‘INTEGER’ bit_size | ‘UNSIGNED’ bit_size | ‘REAL32’ | ‘REAL64’ | ‘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 called basic (res. compound) when the constructor is basic_constructor (res. 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 ≥ 0.

Let $b_0, ..., b_{n-1}$ be bits, n a positive integer. Then

$b = b_0\ b_1 \ldots b_{n-1}$

is called a bit sequence of length |b| = n.

The empty bit sequence of length 0 is denoted ε.

Examples: 10110100, 1, 101, etc. are bit sequences.

The inversion operator (¬) on bit sequences assigns to a bit sequence

$b = b_0\ b_1 \ldots b_{n-1}$

the bit sequence

$\neg b = \neg b_0\ \neg b_1 \ldots \neg\ b_{n-1}$

Here ¬0 = 1 and ¬1 = 0 on bits.

The basic operation on bit sequences is concatenation.

Let $a = a_0 \ldots a_{m-1}$ and $b = b_0 \ldots b_{n-1}$ be bit sequences. Then the concatenation of a and b, denoted ab, is

$a_b = a_0 \ldots a_{m-1}\ b_0 \ldots b_{n-1}$

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 an EPL network a bit sequence is reordered into a sequence of octets. Here and in the following hexadecimal notation is used for octets. Let $b = b_0 \ldots b_{n-1}$ 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 Table 27. The bits b_i, $i \geq n$ of the highest numbered octet are do not care bits.

Table 27 – Transfer Syntax for Bit Sequences

octet number	1.	2.	k.
	$b_7 .. b_0$	$b_{15} .. b_8$	$b_{8k-1} .. b_{8k-8}$

Octet 1 is transmitted first and octet **k** is transmitted last. The bit sequence is transferred as follows across the EPL network:

$b_7, b_6, \ldots, b_0, b_{15}, \ldots, b_8, \ldots$

Example:

Bit 9	*...*	*Bit 0*
10_b	0001_b	1100_b
2_h	1_h	C_h
		$= 21C_h$

The bit sequence $b = b_0 .. b_9 = 0011\ 1000\ 01_b$ 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 ε.

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 (res. FALSE) is represented by the bit sequence 1 (res. 0).

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 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

$b = b_0 \ldots b_{n-1}$

is assigned the value

$UNSIGNEDn(b) = b_{n-1}\ 2^{n-1} + ... + b_1\ 2^1 + b_0\ 2^0$

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:

Table 28 – Transfer syntax for data type UNSIGNEDn

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	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

The data types UNSIGNED24, UNSIGNED40, UNSIGNED48 and UNSIGNED56 should not be applied by new applications.

6.1.4.5 Signed Integer

Data of basic data type INTEGERn has values in the integers. The value range is $-2^{n-1}, \ldots , 2^{n-1}-1$.

The data is represented as bit sequences of length n.

The bit sequence

$b = b_0 .. b_{n-1}$

is assigned the value

$INTEGERn(b) = b_{n-2}\ 2^{n-2} + \ldots + b_1\ 2^1 + b_0\ 2^0$ if $b_{n-1} = 0$

and, performing two's complement arithmetic,

$INTEGERn(b) = -\ INTEGERn(\hat{}b) - 1$ 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 transfered in two octets across the bus, first $F6_h$ and then FE_h.

The following INTEGERn data types are transfered as shown below:

Table 29 – Transfer syntax for data type INTEGERn

octet number	0	1	2	3	4	5	6	7
INTEGER8	$b_7..b_0$							
INTEGER16	$b_7..b_0$	$b_{15}..b_8$						
INTEGER24	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$					
INTEGER32	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$				
INTEGER40	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$			
INTEGER48	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$		
INTEGER56	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	
INTEGER64	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$	$b_{39}..b_{32}$	$b_{47}..b_{40}$	$b_{55}..b_{48}$	$b_{63}..b_{56}$

The data types INTEGER24, INTEGER40, INTEGER48 and INTEGER56 should not be applied by new applications.

6.1.4.6 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)

$REAL32(b) = (-1)^S \, 2^{E-127} \, (1+ F)$

Here

$S = b_{31}$ is the sign.

$E = b_{30}\, 2^7 + \dots + b_{23}\, 2^0$, $0 < E < 255$, is the un-biased exponent.

$F = 2^{-23} \; (b_{22}\, 2^{22} + \dots + b_1\, 2^1 + b_0\, 2^0)$ 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) with
E =129 =27 +20 and
F = 2-1 +2-4 = 2 -23(222+219) hence the number is represented as:

S	*E*	*F*
b_{31}	b_{30} .. b_{23}	b_{22} .. b_0
0	*100 0000 1*	*100 1000 0000 0000 0000 0000*

6.25 = b_0 .. b_{31} *= 0000 0000 0000 0000 0001 0011 0000 0010*

It is transferred in the following order:

Table 30 – Transfer syntax of data type REAL32

octet number	0	1	2	3
REAL32	00_h	00_h	$C8_h$	40_h
	$b_7..b_0$	$b_{15}..b_8$	$b_{23}..b_{16}$	$b_{31}..b_{24}$

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

$b_0(1)$ .. $b_{n-1}(1)$ .. $b_{n-1}(N)$.

Example: Consider the data type

STRUCT OF

INTEGER10 x,
UNSIGNED5 u

NewData

Assume x = - 423_d = 259_h and u = 30_d = $1E_h$. Let b(x) and b(u) denote the bit sequences representing the values of x and u, respectively. Then:

b(x)	*= $b_0(x)$ .. $b_9(x)$*	*= 1001101001*
b(u)	*= $b_0(u)$ .. $b_4(u)$*	*= 01111*
b(xu) = b(x) b(u)	*= $b_0(xu)$ .. $b_{14}(xu)$*	*= 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 length of the octet string.

ARRAY [length] OF UNSIGNED8 OCTET_STRING*length*

6.1.6.2 Visible String

The data type VISIBLE_STRING**length** is defined below. The admissible values of data of type **VISIBLE_CHAR** are 0_h and the range from 20_h to $7E_h$. The data are interpreted as ISO 646-1973(E) 7-bit coded characters. **length** is the length of the visible string.

UNSIGNED8 VISIBLE_CHAR

ARRAY [length] OF VISIBLE_CHAR VISIBLE_STRING*length*

There is no 0_h necessary to terminate the string.

6.1.6.3 Unicode String

The data type UNICODE_STRING**length** is defined below; **length** is the length of the unicode string.

ARRAY [length] OF UNSIGNED16 UNICODE_STRING*length*

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 arbitrary 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.

6.2 Object Dictionary

6.3 Service Data (SDO)

To access the entries of the object dictionary of a device via ETHERNET Powerlink a set of command services are specified.

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.

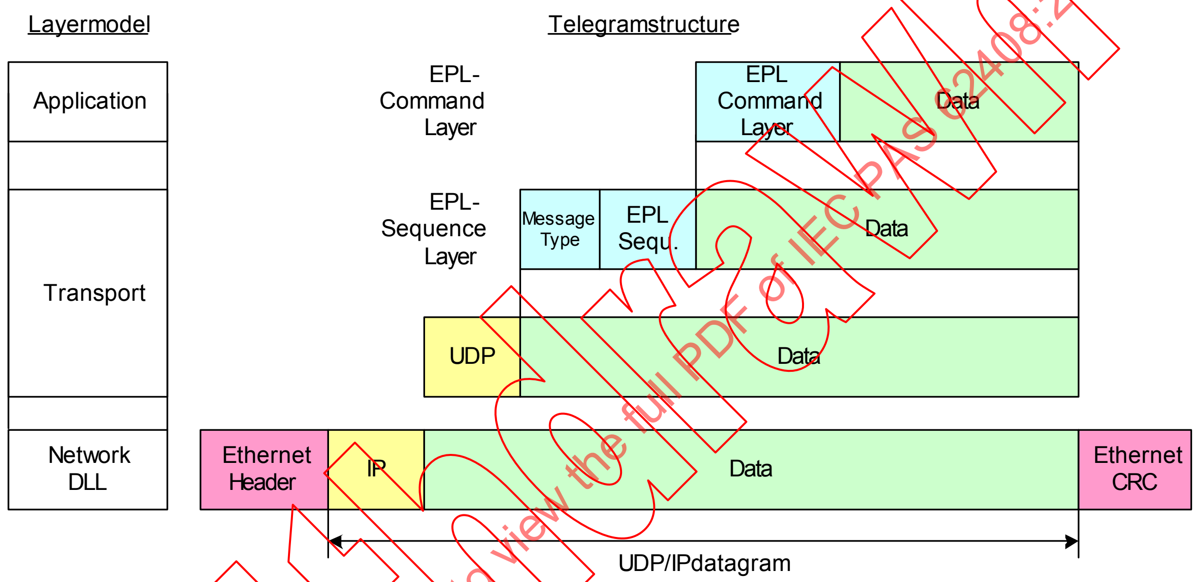


Figure 36 – EPL Command Embedded in UDP/IP Frame

Note:

It is possible to access the parameter of a device without using UDP/IP, using instead the AsyncSend frame can be used. This method is not in the scope of this EPL specification.

Two sublayers are distinguished in the EPL Protocol:

- **EPL Sequence Layer (6.3.3)**

  The Sequence Layer is responsible for sorting the segments of a segmented transfer command so that a correct byte stream is offered to the EPL Command Layer.

- **EPL Command Layer (6.3.4)**

  The Command Layer defines commands to access the parameters of the object dictionary. This layer distinguishes between an expedited and a segmented transfer.

For applications that do not require short cycle times and low jitter, EPL telegrams may be inserted in a UDP/IP datagram, in effect running EPL over UDP/IP. For this reason the message type (defined in the Data Link Layer) is inserted in front of the Sequence Layer (see 6.3.2).

6.3.1 UDP Layer

The UDP Header contains the following informations:

Table 31 – UDP Header

Field	Size	Description	Used in EPL
Source Port	2 Byte	Port Number of the sending process	Logical channel[8]
Destination Port	2 Byte	Port Number of the receiving process	
Length	2 Byte	Datalength of the whole UDP frame incl. header	not used
Checksum	2 Byte	optional Checksum	not used

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_EPL_Port**.

The client starts the transfer using the standard destination port C_EPL_port and a client dependent source port (>1024). The server responds to the request with the source port C_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.

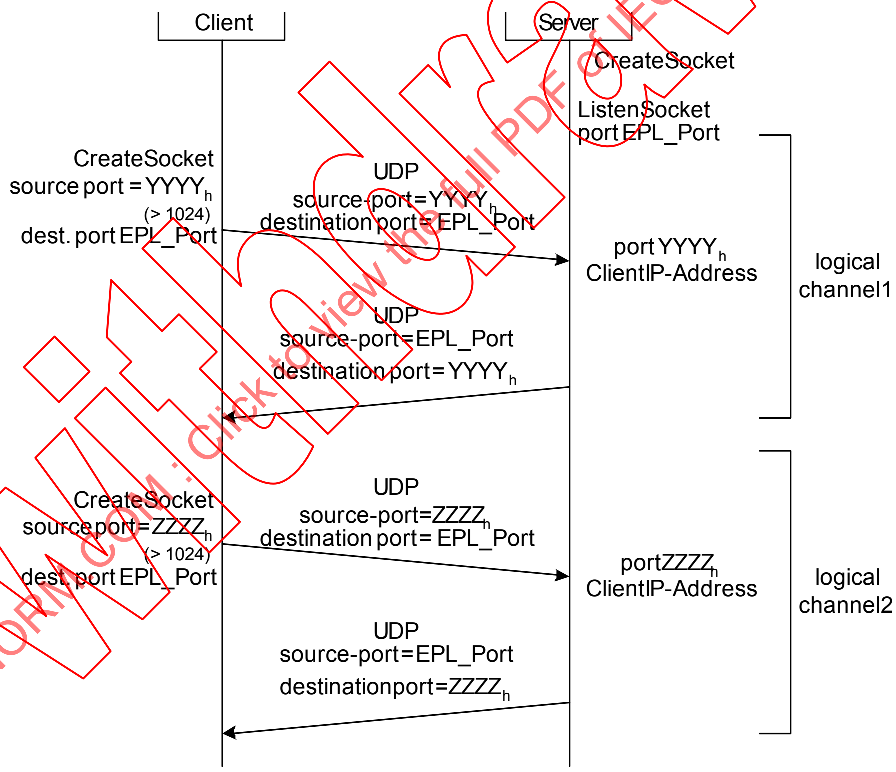


Figure 37 – UDP Socket

6.3.2 SDO EPL Message Type

In future it should be possible to transfer EPL frames (SOA, PollResponse etc.) via UDP/IP. This might be useful in networks with very low demands on timing (e.g. home automation). Therefore the EPL header used in the EPL frames is also inserted in the UDP/IP frames.

At the same time this mechanism allows the transfer of SDOs via standard EPL ASend frames.

[8] together with the client and server IP address.

Table 32 – SDO EPL Message Type Field

	Bit Offset							
Byte Offset	7	6	5	4	3	2	1	0
0	MessageType = ASend							
1	reserved							
2	reserved							
3	ServiceID = SDO							
4 – 7	Sequence Layer Protocol							
8 – k-1	Command Layer Protocol							
k – 1472	Optional Payload Data							

Table 33 – SDO EPL Message Type Field Interpretation

Field	Abbr.	Description	Value
MessageType	mt	Message type as described in the data link layer (Table 10)	ASend
reserved	res	These fields are reserved They are used for EPL Destination and Source Address when sending SDO without UDP/IP These bytes are set to '0' for EPL Message Types because the EPL Destination and Source Address are not needed.	Zero, when embedded in UDP/IP Frame else mt specific
ServiceID	sid	Qualifies the ASend Frame (Table 22)	SDO
Sequence Layer Protocol		EPL Sequence Layer see subclause 6.3.3	
Command Layer Protocol		EPL Command Layer see subclause 6.3.4.1	

6.3.3 SDO Sequence Layer

The EPL 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 in the SDO Command Layer (6.3.4).

6.3.3.1 Asynchronous Sequence Layer

The EPL 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 station, as well a connection state and a connection acknowledge.

The Sequence Layer Header shall end on a 4-byte boundary, so there may be the need for some padding bytes in front of the Sequence Layer Header.

Table 34 – EPL Sequence Layer in asynchronous data frame

	Bit Offset							
Octet Offset	7	6	5	4	3	2	1	0
0	ReceiveSequenceNumber						ReceiveCon	
1	SendSequenceNumber						SendCon	

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.

Table 35 – Fields of EPL Sequence Layer in asynchronous data frame

Field	Abbr.	Description	Value
ReceiveSequenceNumber	rsnr	Sequence Number of the last correctly received frame.	0 ... 63
ReceiveCon	rcon	Acknowledge of connection code to the receiver	0: No connection 1: Initialization 2: Connection valid 3: Error Response (retransmission request)
SendSequenceNumber	ssnr	Sequence Number of the frame, shall be increased by 1 with every new frame.	0 ... 63
SendCon	scon	Connection code of the sender.	0: No connection, 1: Initialization 2: Connection valid 3: Connection valid with acknowledge request

6.3.3.1.1 Connection

6.3.3.1.1.1 Initialization of Connection

Sender shall request initialization by setting scon to 1. This shall be responded by receiver with the same connection code and the sequence number.

If the receiver had a connection to the sender this connection shall be shut down.

Additional the sequence number shall be initialized at the receiver.

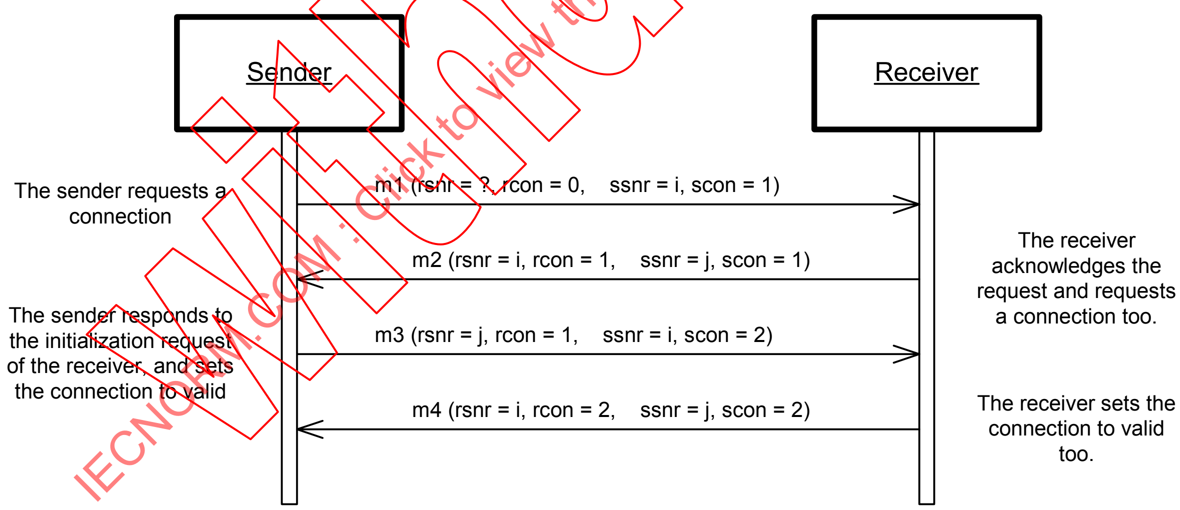


Figure 38 – Initialization of a asynchronous connection

After this the bi-directional connection is established.

The sequence numbers shall not be incremented until the bi-directional connection initialisation has been completed. No command shall be sent while initializing the bi-directional connection.

6.3.3.1.1.2 Closing a connection

A connection should be shut down, when it is not needed any longer.

A node may shut down a connection if it needs the resources for other reasons.

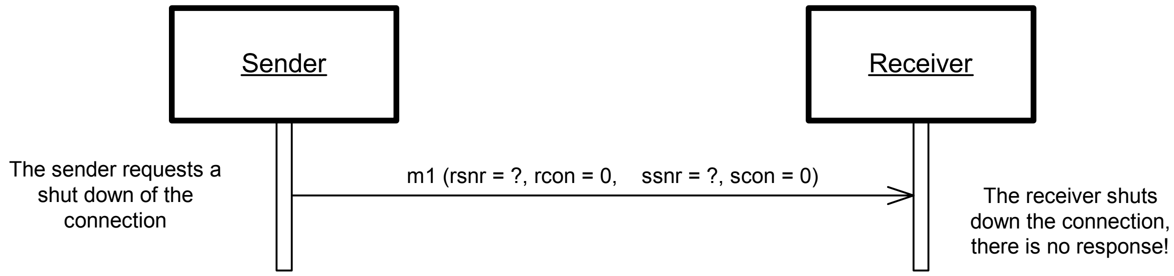


Figure 39 – Closing of asynchronous connection

Closing a connection shall be indicated by rcon = 0 and scon = 0.

There shall be no acknowledging for closing a connection.

6.3.3.1.1.3 Normal Connection

When the connection is established each station is allowed to send frames. Each station shall keep the sent data in some sort of a history buffer until it is acknowledged.

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.

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.

The sender shall not forward more than 31 frames (sliding window) before receiving an acknowledgement, to make it 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.

6.3.3.1.2.6).

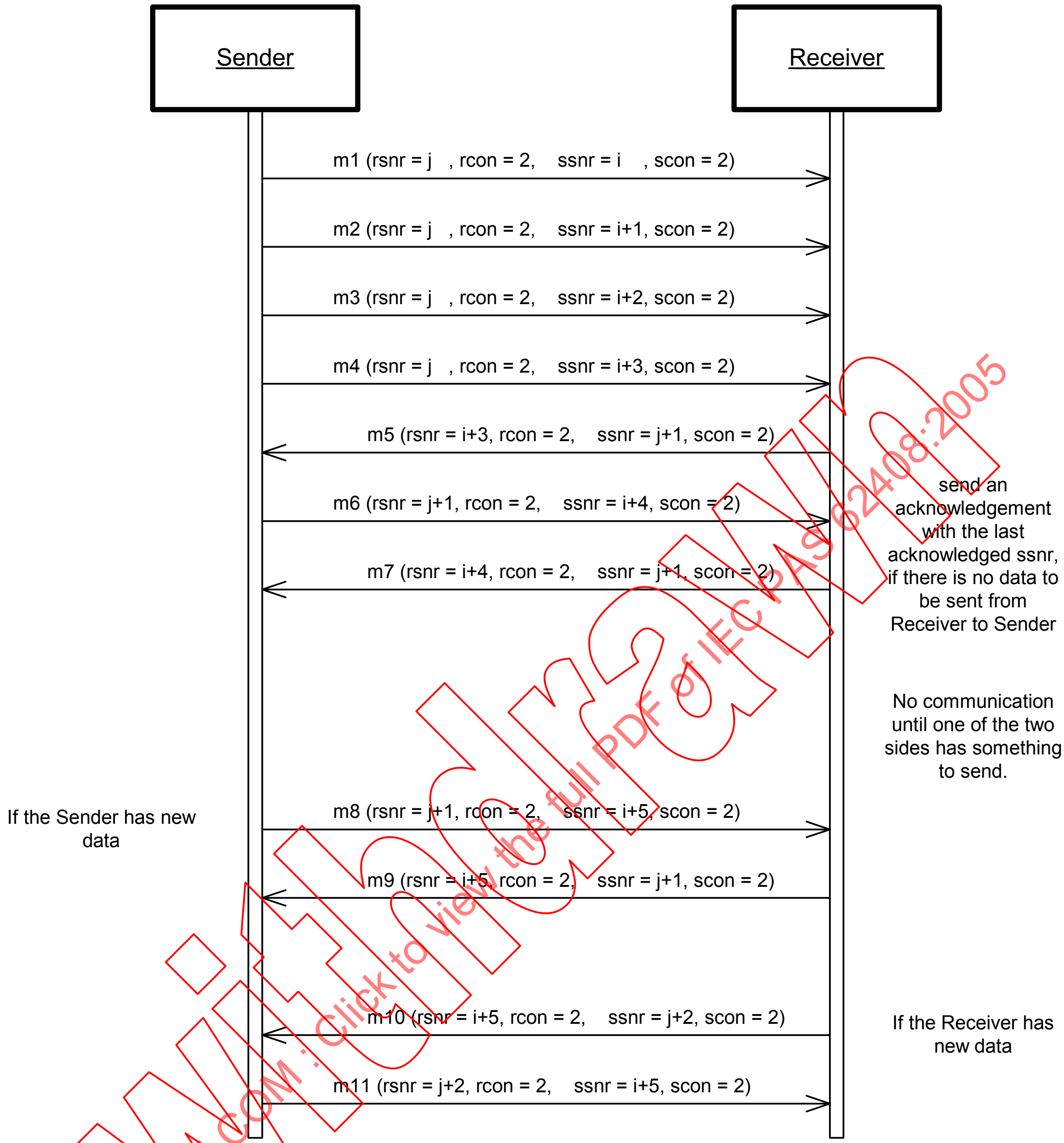


Figure 40 – Normal asynchronous communication

6.3.3.1.1.4 Connection 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 31 frames before receiving an acknowledgement.

This example shows a configuration where the frames are delayed. A typical situation where frames are delayed is when EPL Networks are connected by means of routers over a legacy ethernet.

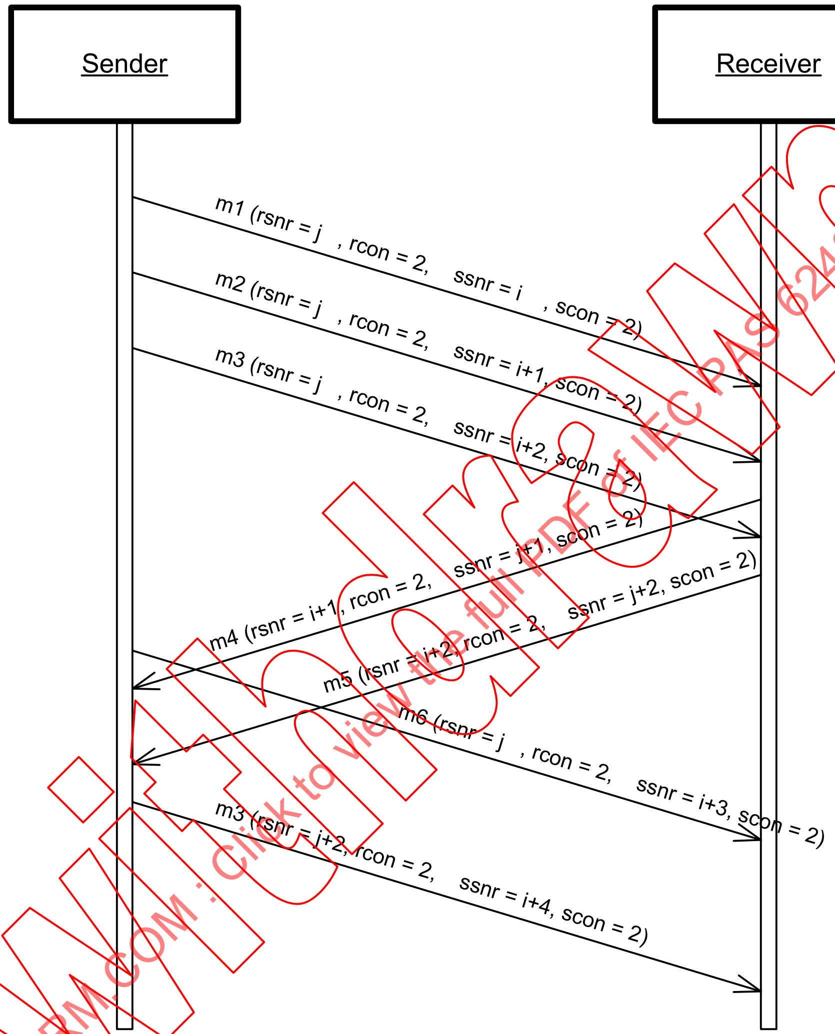


Figure 41 – Delayed asynchronous communication

6.3.3.1.1.5 Sender History Full

When the buffer for keeping frames is full (sliding window size exhausted), the sending station may explicitly request an acknowledgement by sending a frame with acknowledge request.

The receiver shall acknowledge this frame with an empty acknowledge frame if it has no own frames to be sent.

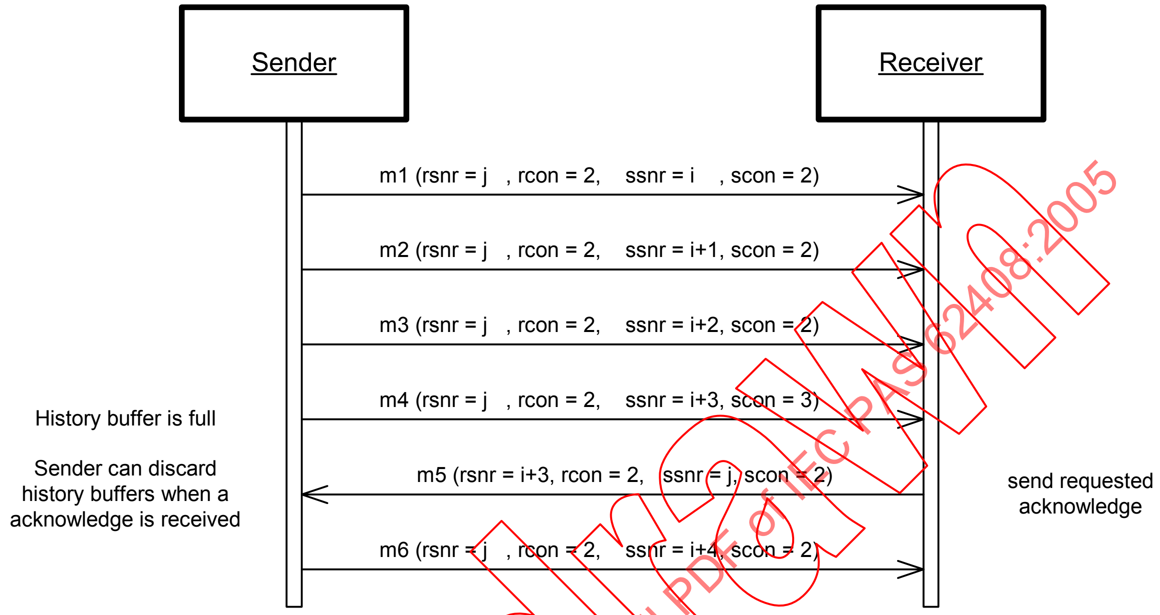


Figure 42 – Asynchronous communication when history buffer gets full

6.3.3.1.2 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

6.3.3.1.2.1 Error: Loss of Frame with Data

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 be included in a normal command frame.

The sender shall repeat all the frames that followed the responded sequence number.

The acknowledge fields in the repeated frames shall be updated.

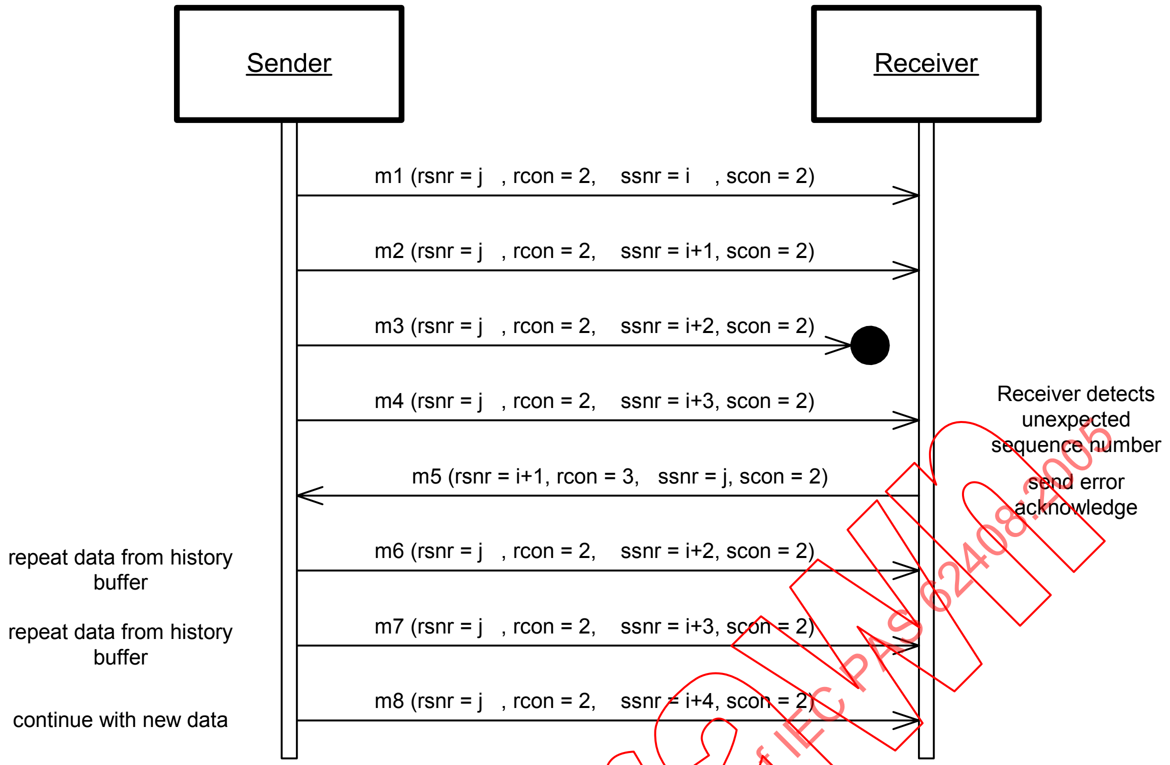


Figure 43 – Error loss of asynchronous frame

6.3.3.1.2.2 Error: Loss of Acknowledge Frame

If the sender is waiting for an acknowledgement and has no new frames to be sent, it shall repeat the latest message with acknowledge request after a timeout.

A suitable timeout for acknowledge receive before repeating the message would be number of stations in the network times the cycle time, but at least 10 cycles. To avoid congestion the sender shall double the timeout after each repeat.

<TBD NMT shall tell the CN the number of Stations in the Network or the start timeout for the sequence layer>

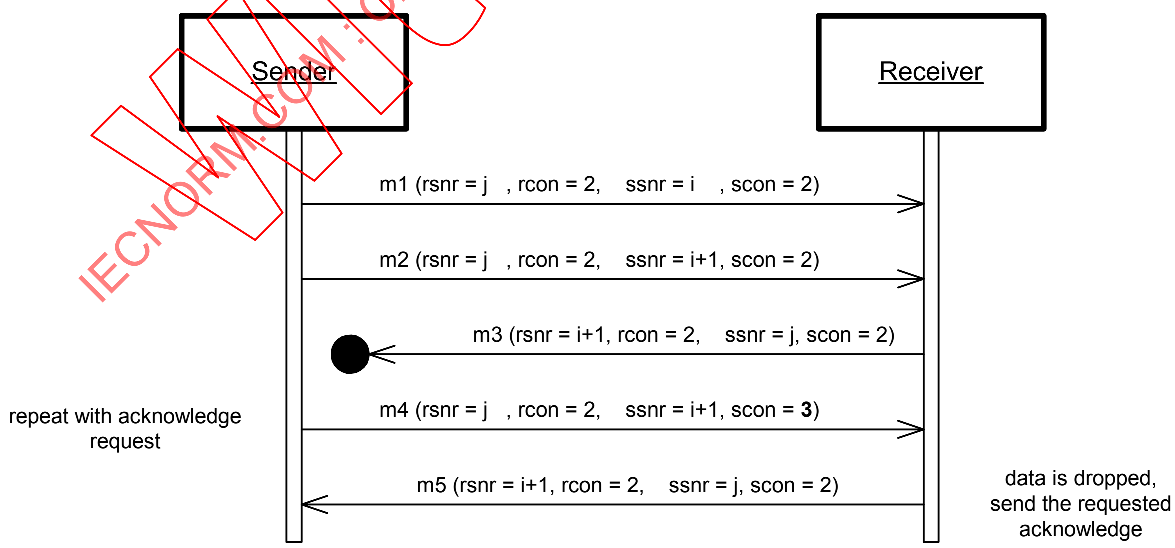


Figure 44 – Error loss of asynchronous acknowledge

6.3.3.1.2.3 Error: Duplication of 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.

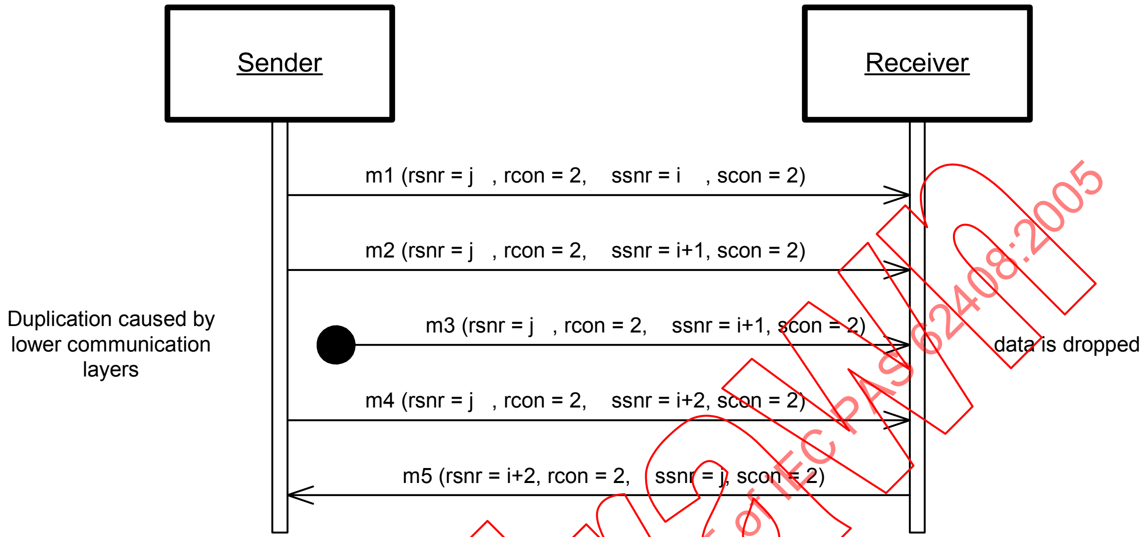


Figure 45 – Error duplication of asynchronous frame

6.3.3.1.2.4 Error: Overtaking of Frames

When a frame overtakes, this looks to the receiver at the first as if a frame was lost, so the receiver shall send an error response to the sender so that the sender repeats the frame.

Then the overtaken frame arrives, and the receiver takes it.

After this the newly requested frame arrives and is dropped at the receiver, because it looks like a normal duplicated frame.

Remark:

Overtaking of frames will never occur inside the EPL network domain, but in connection over Internet using Type I routers overtaking can occur on the Legacy Ethernet side.

6.3.3.1.2.5 Broken Connection

It shall be detected, that the connection is broken, if the opposite station is shut down or disconnected from the network.

The connection shall be considered broken, when there is no acknowledgement after sending multiple acknowledges requests until a timeout. The default shall be about 30 seconds. *<TBD index and sub-index for timeout.>*

With the method stated in 6.3.3.1.2.2 the acknowledge request shall be sent more than once within the default timeout of 30 seconds.

A timeout of 30 seconds will be high *enough* even for diagnosing stations over the internet.

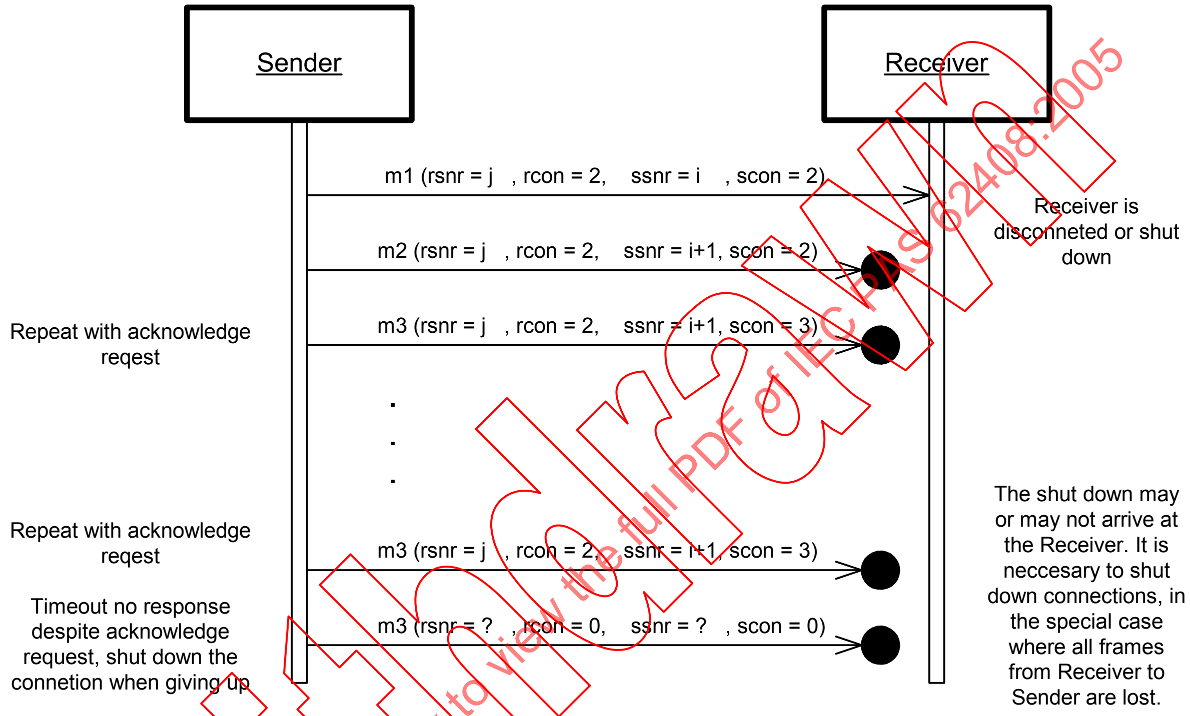


Figure 46 – Error asynchronous communication broken

6.3.3.1.2.6 Error: Flooding with commands

If the sender sends commands at a too high rate, the command layer at the receiver may not be able to fetch 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.

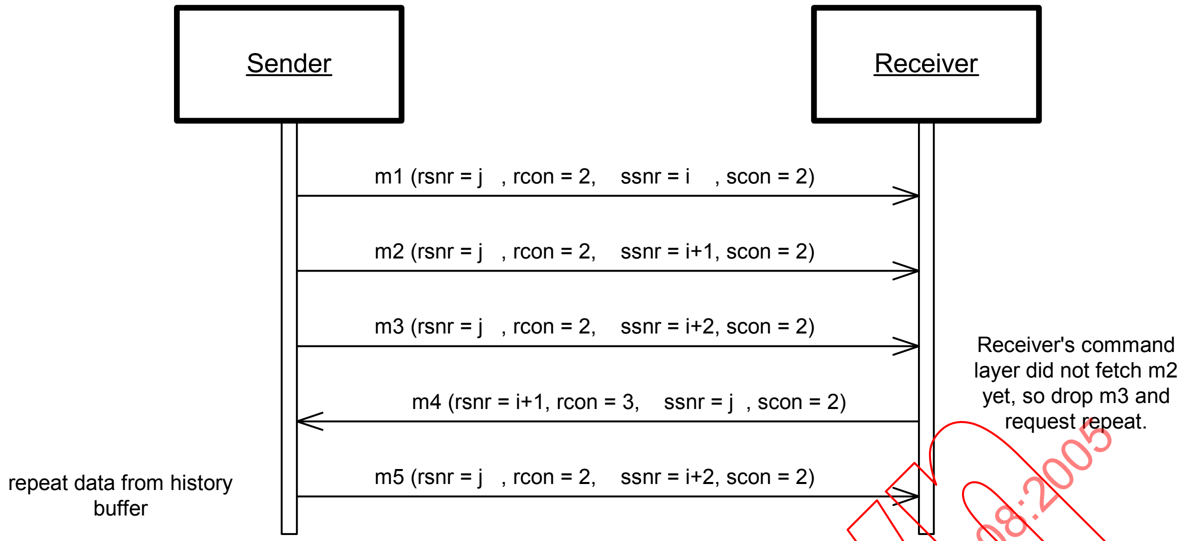


Figure 47 – Error Flooding with asynchronous commands

6.3.3.2 Embedded Sequence Layer for SDO in Cyclic Data

For embedding of SDO in cyclic data (PollRequest and PollResponse) the first byte within EPL Command Layer is reserved for EPL Sequence Layer.

Table 36 – EPL Sequence Layer for embedding of SDO in cyclic data

	Bit Offset							
Byte Offset	7	6	5	4	3	2	1	0
0	SequenceNumber						Connection	
1 …	Command Layer Protocol							

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 the Connection bits.

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.

Table 37 – Fields of EPL Sequence Layer for embedding of SDO in cyclic data

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 1: Initialization 2: Connection valid 3: Error Response (Retransmission Request)

6.3.3.2.1 Connection

6.3.3.2.1.1 Initialization of Connection

Connection is not initialized (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.

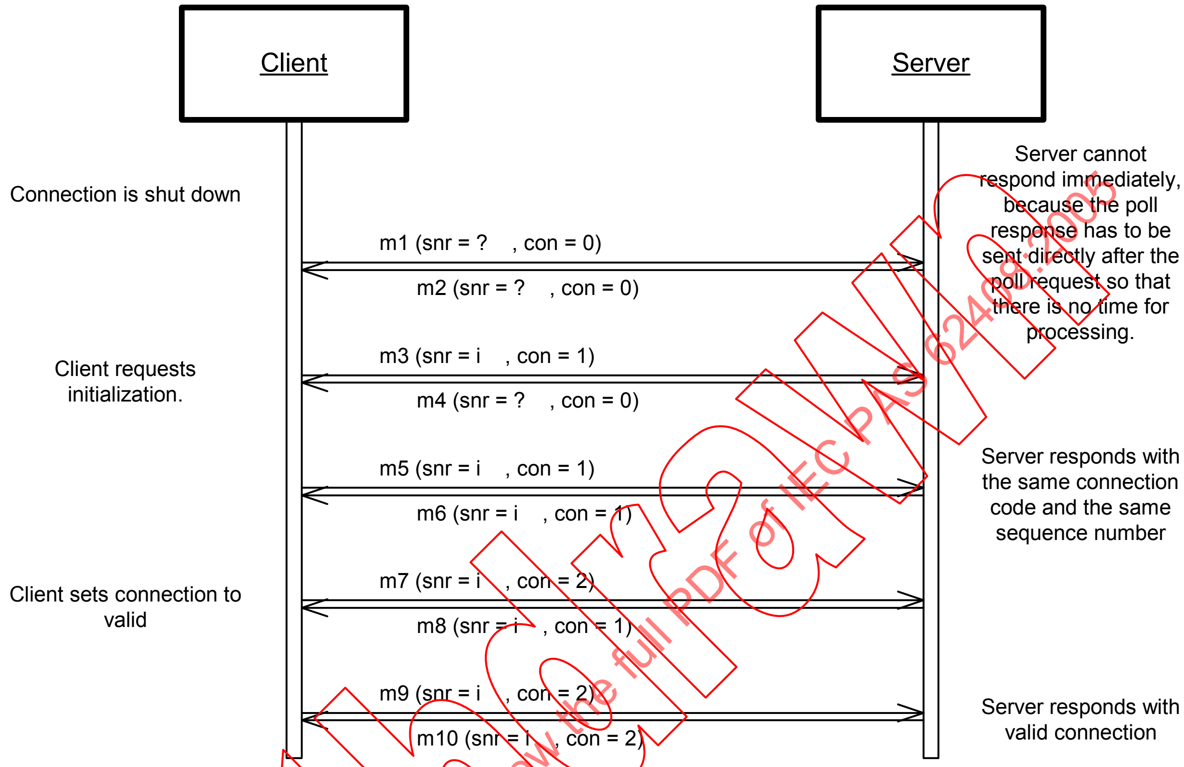


Figure 48 – Initialization of embedded connection

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 initialization.

6.3.3.2.1.2 Normal Connection

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 subclause) 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:

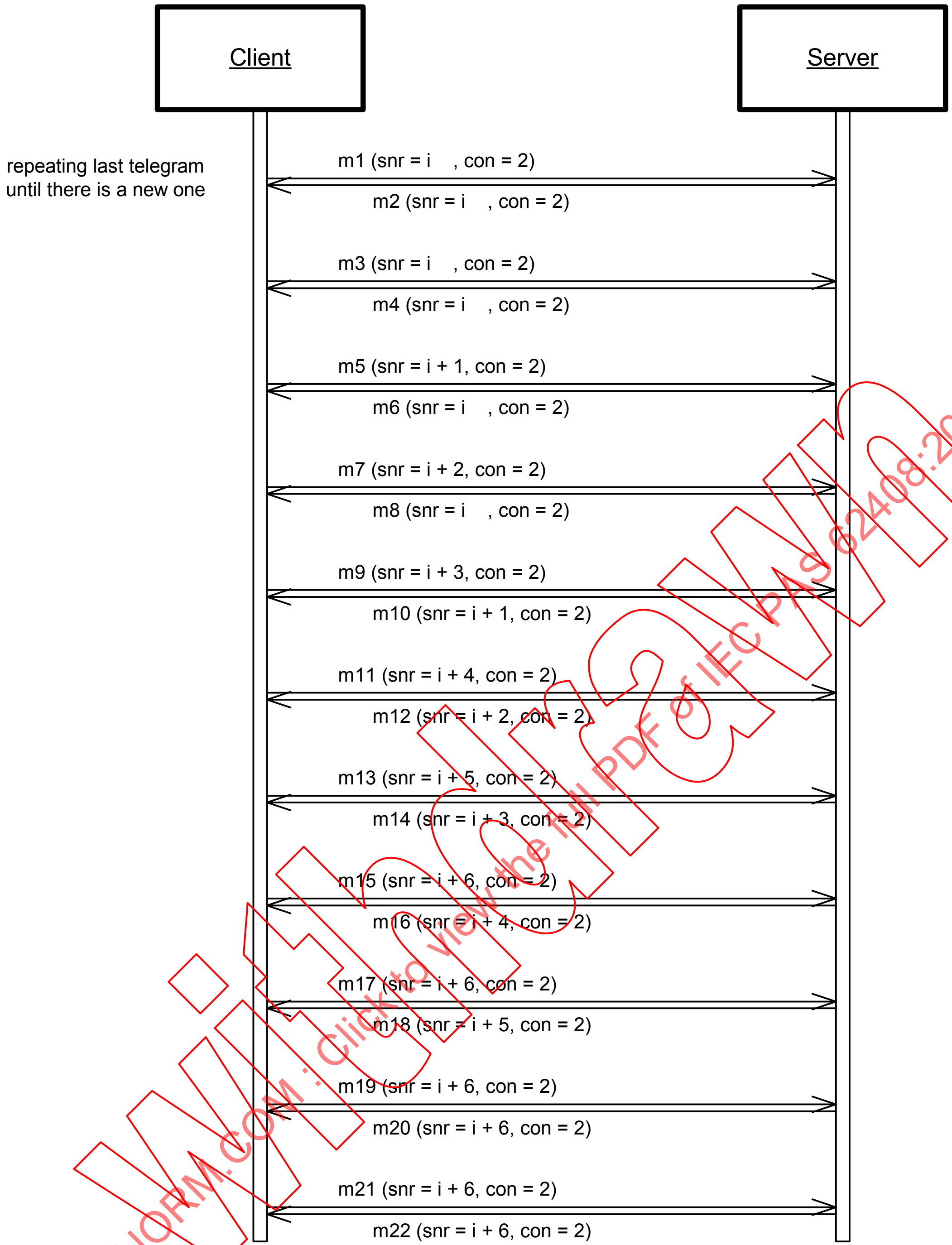


Figure 49 – Normal embedded communication

6.3.3.3 Errors

6.3.3.3.1 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.

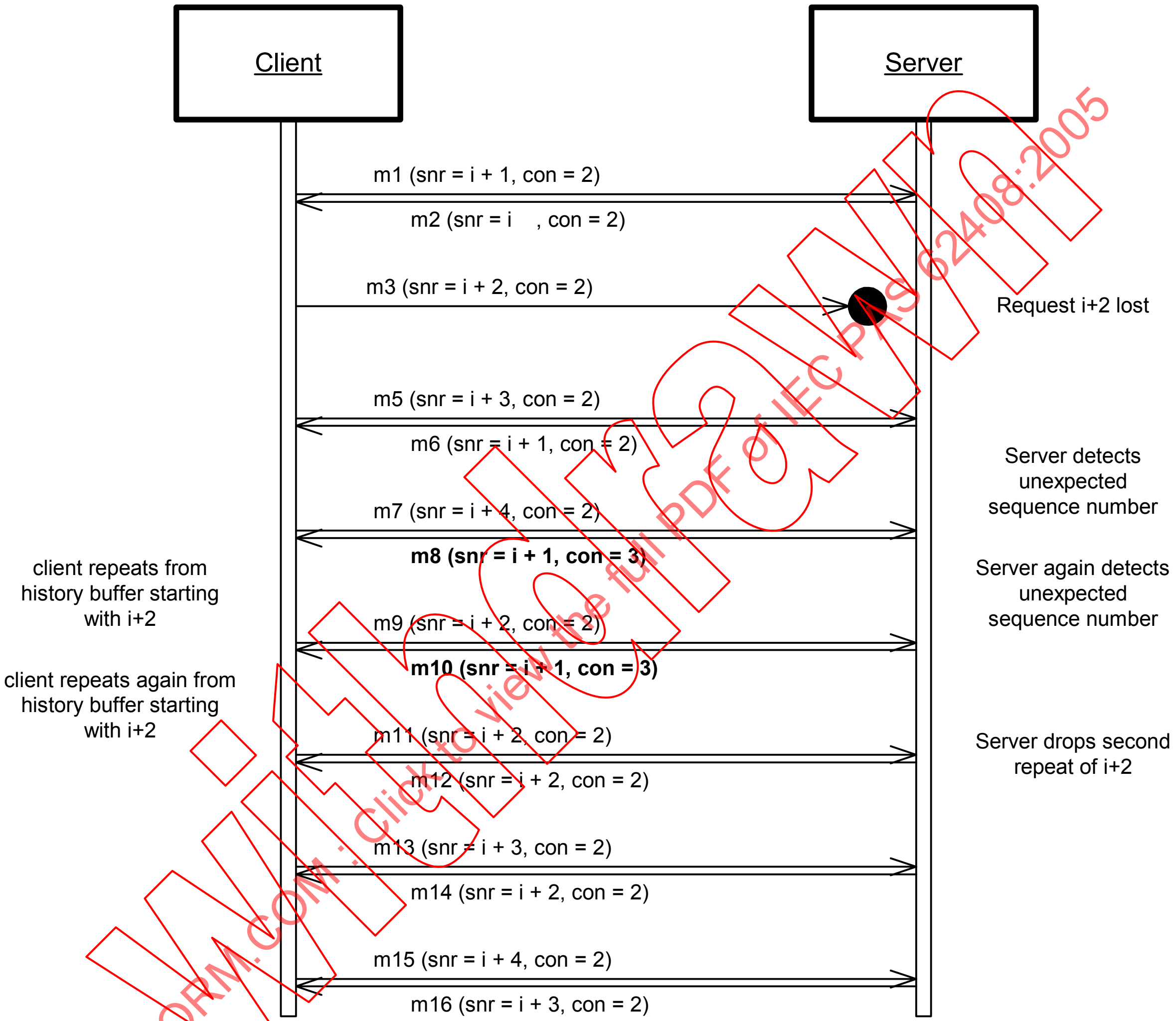


Figure 50 – Error embedded request lost

6.3.3.3.2 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.

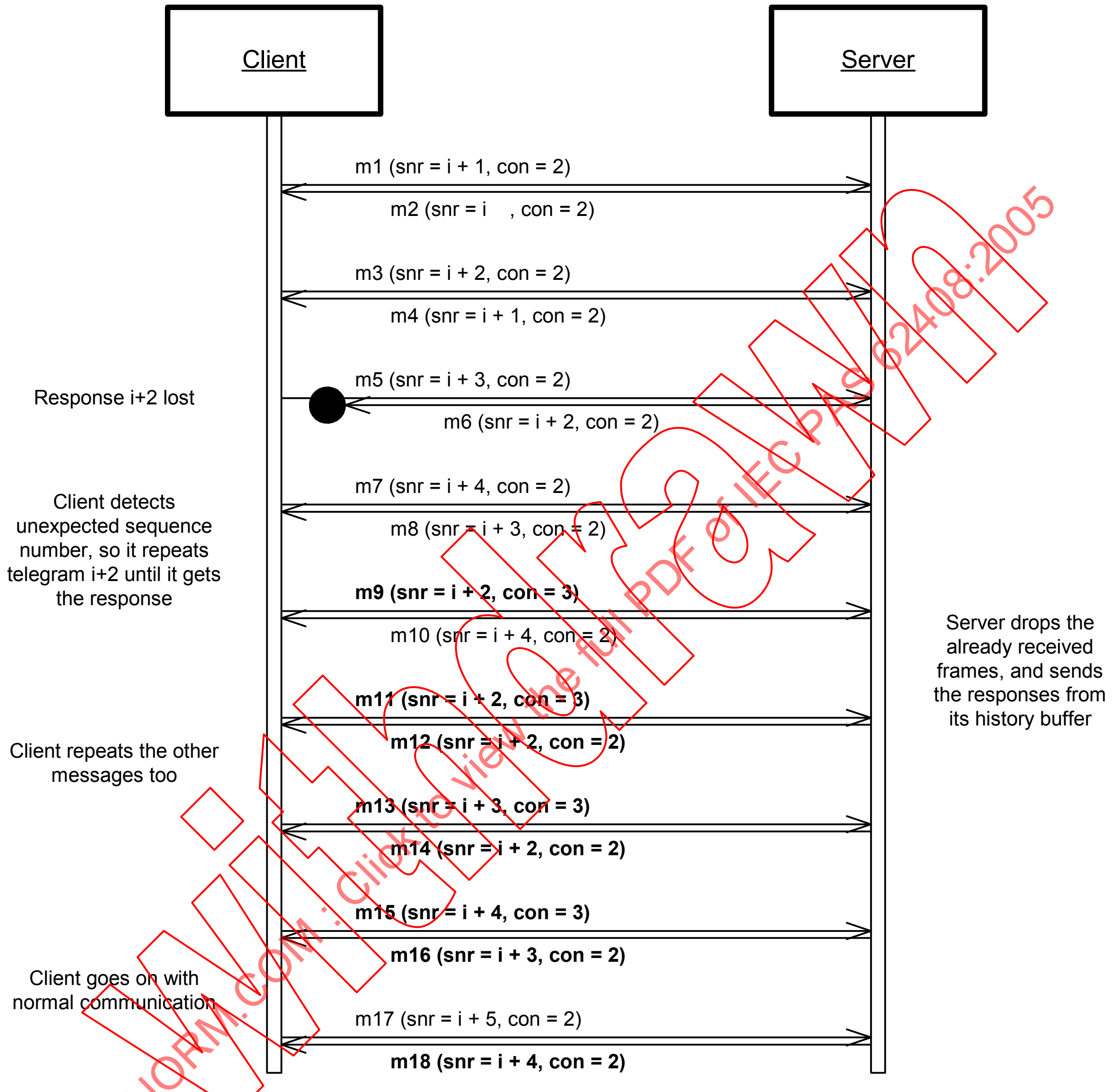


Figure 51 – Error embedded response lost

6.3.3.4 Handling of Segmented Transfers

6.3.3.4.1 Segmented Download from Client to Server

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.

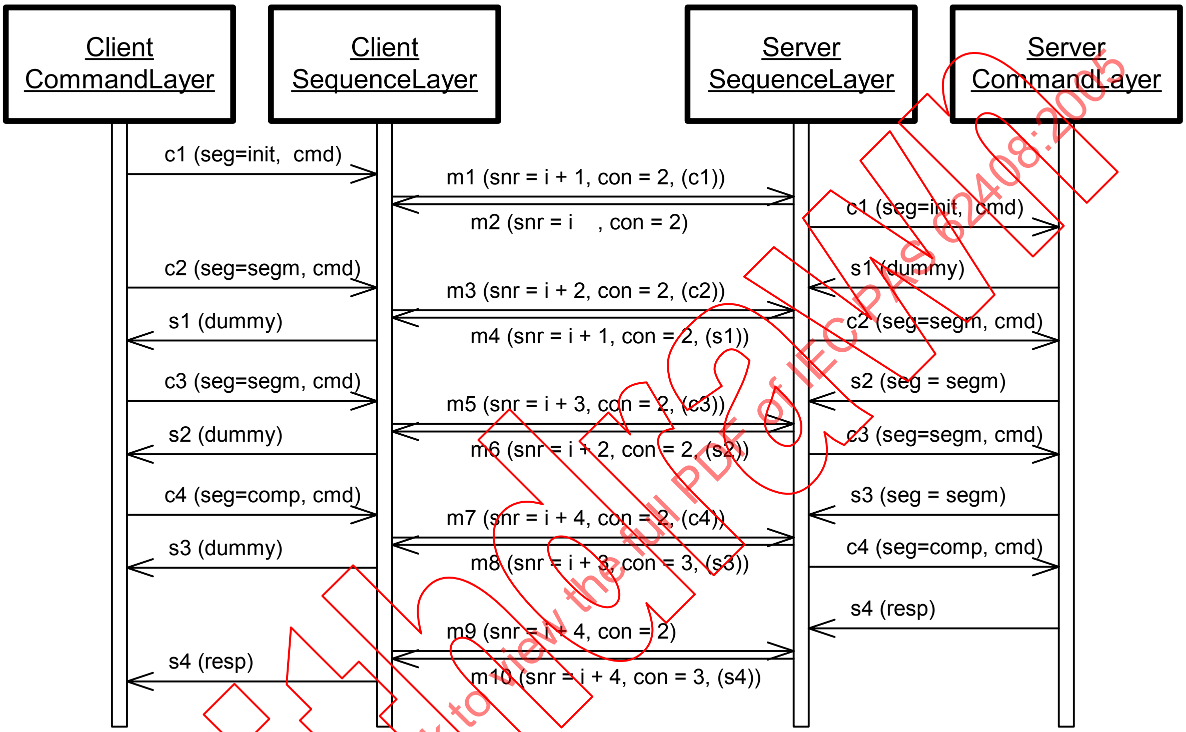


Figure 52 – Embedded segmented download

6.3.3.4.2 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.

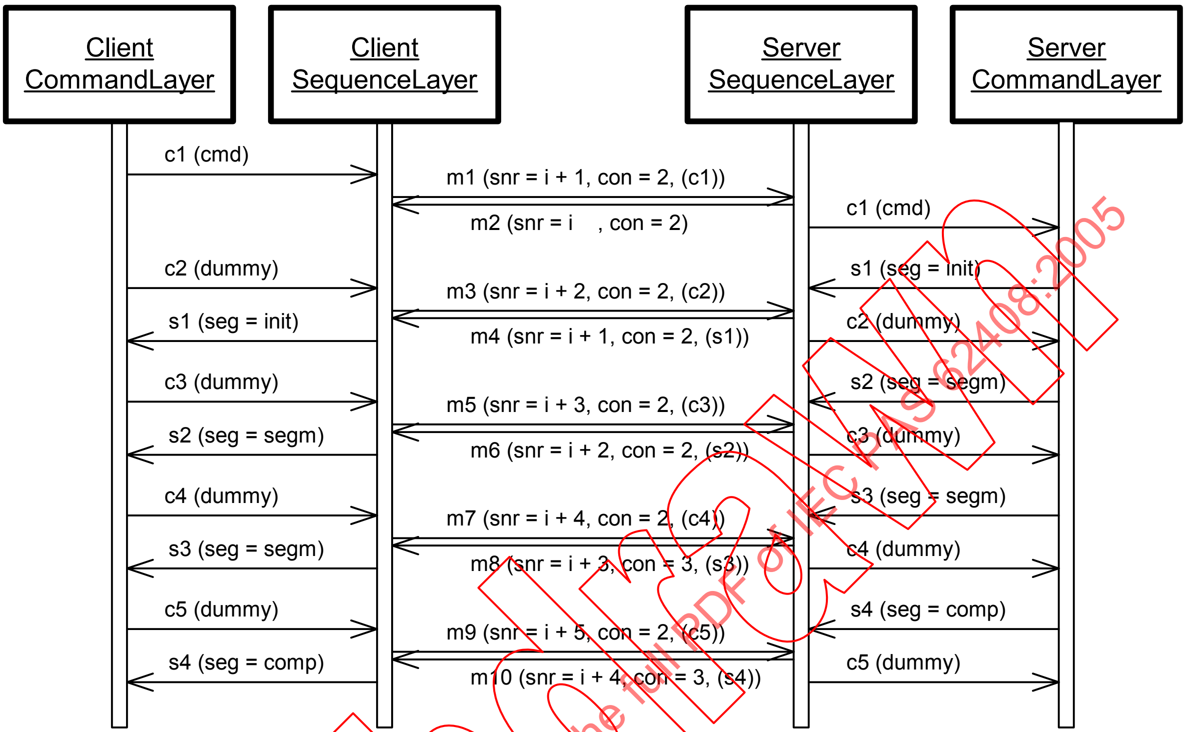


Figure 53 – Embedded segmented upload

6.3.4 SDO Command Layer

Tasks of the EPL Command Layer

- Addressing of the parameters, e.g. via index/sub-index or via name
- Provide additional services
- Distinguish between expedited and segmented transfer

The EPL Command Layer is embedded in the EPL Sequence Layer. If a large block is to be transferred the EPL 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.

Figure 54 shows the structure of the information in the EPL Command Layer Header.

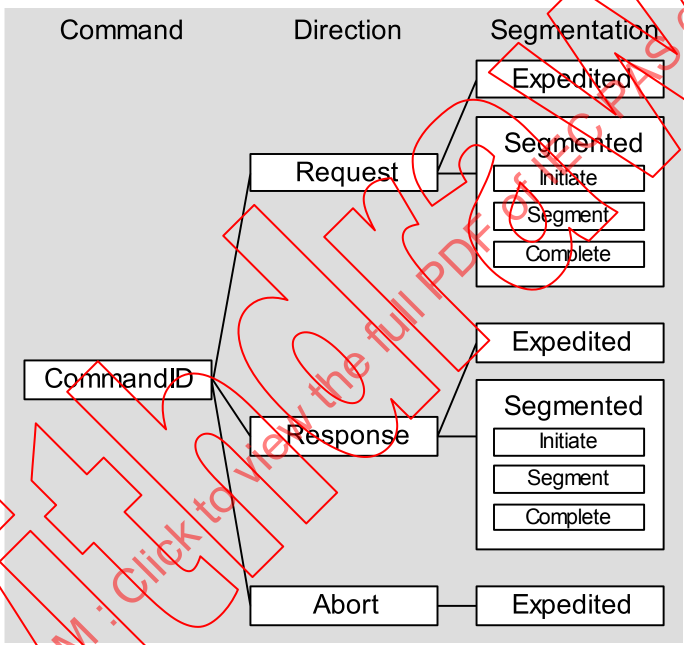


Figure 54 – Information Structure EPL Command Layer

6.3.4.1 EPL Command Layer Protocol

This subclause defines the fix part of the EPL Command Layer protocol.

The EPL Command Layer is structured in the following way:

Table 38 – EPL Command Layer

Byte Offset	Bit Offset								
	7	6	5	4	3	2	1	0	
0	reserved								Fixed part
1	Transaction ID								
2	Res-ponse	Abort	Segment ation	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 – 1465	Optional Payload Data								

k = Length of Command ID specific header; k < 1465
d: if seg = Initiate d = 1
else d = 0

Table 39 – EPL Command Layer Field Interpretation

Field	**Abbr.**	**Description**	**Value**
reserved	res	Reserved This byte is used when embedding the SDO in cyclic data (subclause 0).	0
Transaction ID	tid	Unambiguous transaction ID for a command. Changed by the client with every new command.	0 – 255
Response	rsp	Request / Response	0: Request 1: Response
Abort	a	The requested Transfer could not be completed by the client/server	0: transfer ok 1: abort transfer
Segmentation	seg	Differentiates between expedited and segmented transfer	0: Expedited Transfer 1: Initiate Transfer 2: Segment 3: Transfer Complete
Command ID	cid	Specifies the command	see Table 43
Segment Size	ss	Length of segment data. Counting from the end of the command header (beginning with Byte Offset 8)	0 – 1458
Data Size	ds	Contains the number of bytes of the transferred block. Counting from the end of the command header (beginning with Byte Offset 8) Only used for the Initiate Transfer Frame (seg = Initiate) If ds = 0000h, the size is not indicated	0 – 2^{32}-1
Command ID specific		Specifies the command referenced by the cid.	see subclause6.3.4.2

The Transaction ID can be used to support several logic channels parallel via the same UDP socket. It is therefore not necessary to open several tasks as it is usual for UDP sockets.

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 1458 bytes (i.e. 1500 byte payload data for the Ethernet frame) may be supported. The client can use the command "Maximum Segment Size" (see 6.3.4.2.4.1) to get the maximum usable size for a communication to a server.

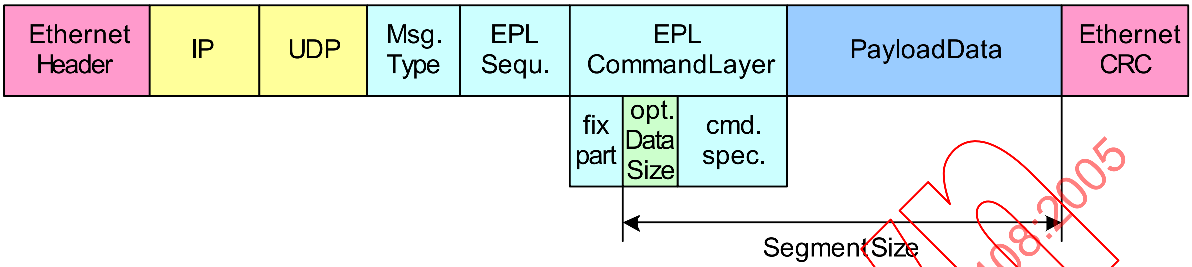


Figure 55 – Definition of Segment Size

For Initiate Domain (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 4 (Expedited Transfer) or 8 (Segmented Transfer).

6.3.4.1.1 Download Protocol

The download service is used by the client of an SDO to download data to the server (owner of the object dictionary).

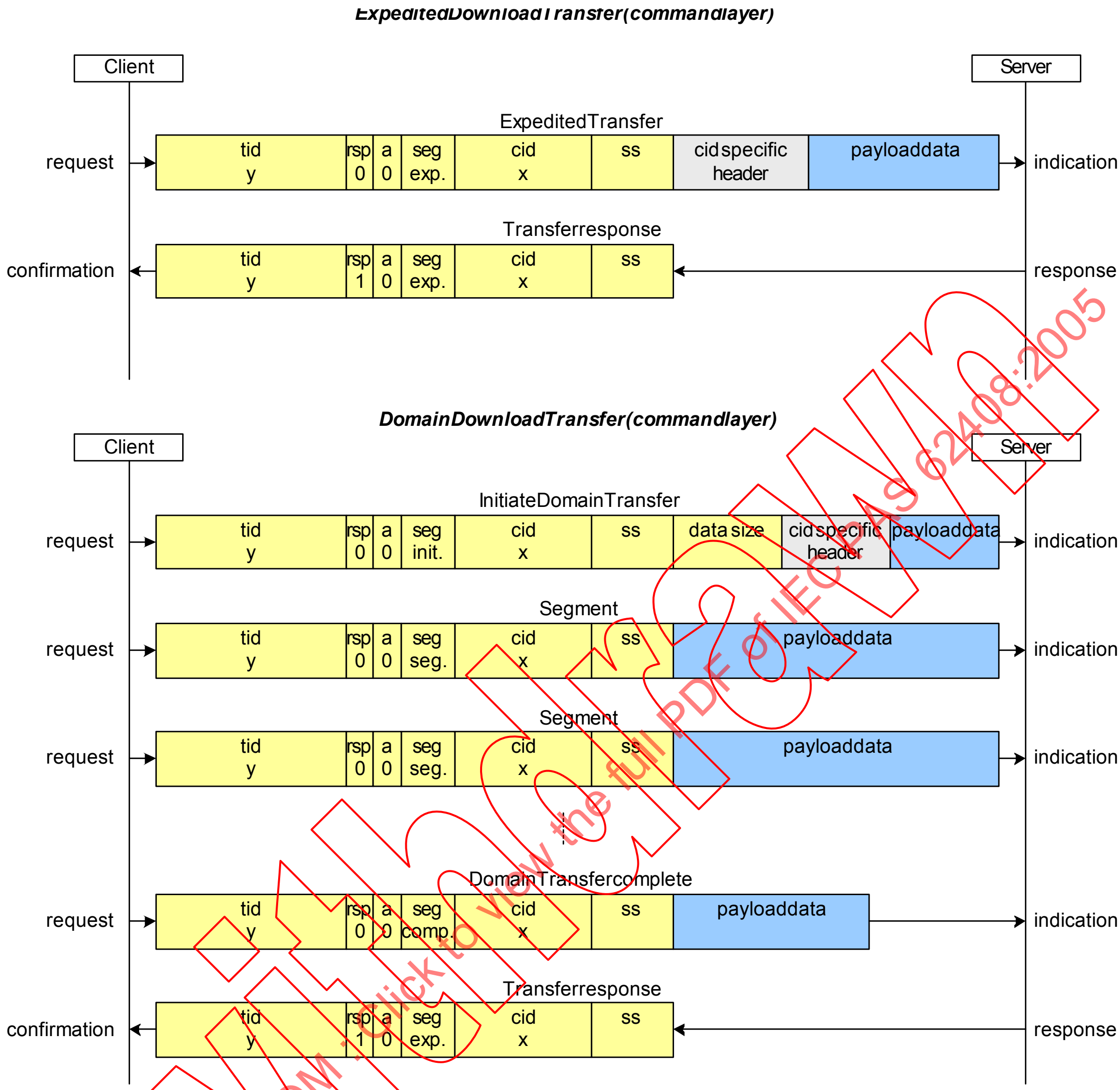


Figure 56 – EPL Command Layer: Typical Download Transfer

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 Figure 56. There may be more frames involved in the initialization of the SDO Sequence Layer, see subclause 6.3.3 for details.

6.3.4.1.2 Upload Protocol

The Upload service is used by the client of an SDO to upload data from the server.

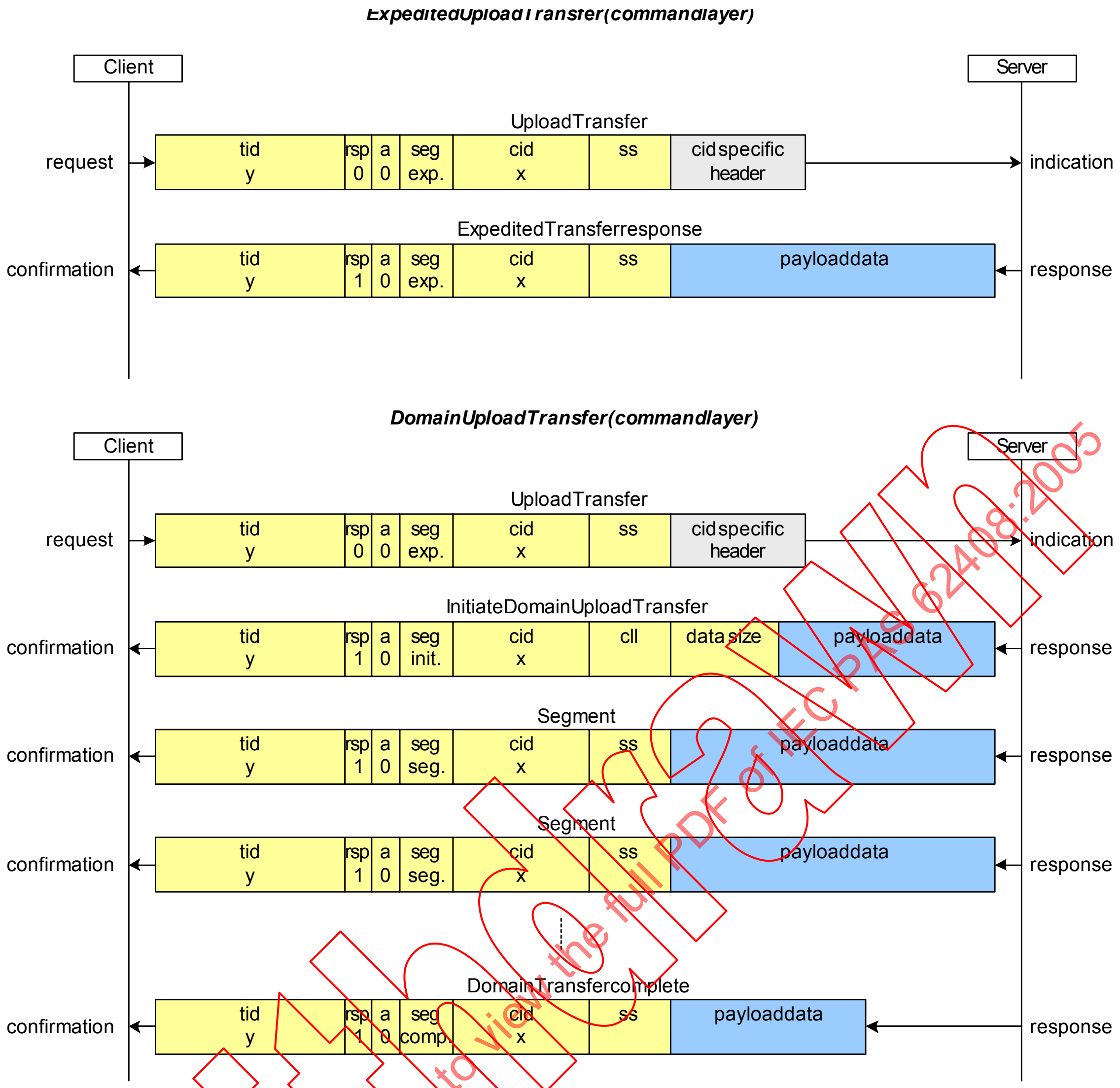


Figure 57 – EPL Command layer: Typical Upload Transfer

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.

6.3.4.1.3 Abort Transfer

The Abort Transfer service aborts the up- or download of the SDO referenced by the Transaction ID. The reason is indicated.

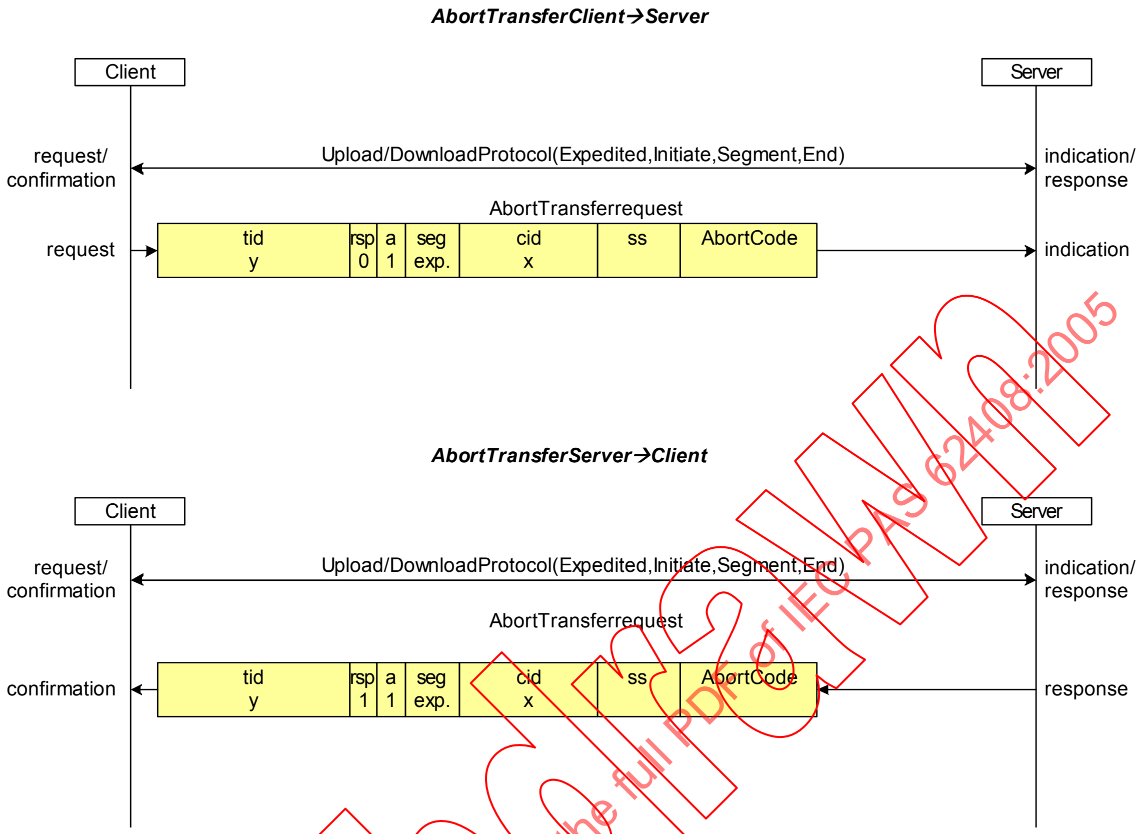


Figure 58 – Abort Protocol

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.

Table 40 – Abort Transfer Frame

Byte Offset	Bit Offset							
	7	6	5	4	3	2	1	0
0	reserved							
1	tid							
2	rsp x	Abort 1	seg x		reserved			
3	cid							
4 - 5	Segment Size							
6 - 7	reserved							
8 - 11	Abort Code							

Table 41 – Abort Transfer Frame Field Interpretation

Field	Abbr.	Description	Value
Abort Code	ac	Reason of the abort	see Table 42

The abort code is encoded as UNSIGNED32 value.

Table 42 – SDO Abort Codes

Abort code	Description
0503 0000_h	reserved
0504 0000_h	SDO protocol timed out.
0504 0001_h	Client/server Command ID not valid or unknown.
0504 0002_h	Invalid block size.
0504 0003_h	Invalid sequence number.
0504 0004_h	reserved
0504 0005_h	Out of memory.
0601 0000_h	Unsupported access to an object.
0601 0001_h	Attempt to read a write-only object.
0601 0002_h	Attempt to write a read-only object.
0602 0000_h	Object does not exist in the object dictionary.
0604 0041_h	Object cannot be mapped to the PDO.
0604 0042_h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043_h	General parameter incompatibility.
0604 0047_h	General internal incompatibility in the device.
0606 0000_h	Access failed due to an hardware error.
0607 0010_h	Data type does not match, length of service parameter does not match
0607 0012_h	Data type does not match, length of service parameter too high
0607 0013_h	Data type does not match, length of service parameter too low
0609 0011_h	Sub-index does not exist.
0609 0030_h	Value range of parameter exceeded (only for write access).
0609 0031_h	Value of parameter written too high.
0609 0032_h	Value of parameter written too low.
0609 0036_h	Maximum value is less than minimum value.
0800 0000_h	General error
0800 0020_h	Data cannot be transferred or stored to the application.
0800 0021_h	Data cannot be transferred or stored to the application because of local control.
0800 0022_h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023_h	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).

The abort codes not listed here are reserved.

6.3.4.2 Commands

This subclause describes the Command ID specific part of the EPL Command Layer.

The following commands can be used to access the parameters of a server:

Table 43 – Command Services and Command ID

Command	**Description**	**Command ID**	**M/O**
NIL	**Not in List**	**0h**	**M**
SDO Protocol	**Service data object**		
Write by Index	Write a parameter, addressing via index/sub-index	1h	M
Read by Index	Read a parameter, addressing via index/sub-index	2h	M
Write All by Index	Write a parameter, addressing via index, all sub-indices	3h	O
Read All by Index	Read a parameter, addressing via index, all sub-indices	4h	O
Write by Name	Write a parameter, addressing via name	5h	O
Read by Name	Read a parameter, addressing via name	6h	O
File Transfer			
File Write	Simple file transfer	20h	O
File Read	Simple file transfer	21h	O
Variable groups			
Write Multiple Parameter by Index	Write multiple parameters within one command, addressing via index/sub-index	31h	O
Read Multiple Parameter by Index	Read multiple parameters within one command, addressing via index/sub-index	32h	O
Parameter Services			
Maximum Segment Size	Exchange the maximum segment size	70h	Cond.[9]
Link Name to Index	Link Objects only accessible via name to an index/sub-index	71h	O
Manufacturer specific		**80h - FFh**	**O**

[9] Conditional: Only necessary if a segment size > 256 Byte should be transferred

6.3.4.2.1 SDO Protocol

6.3.4.2.1.1 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.

Table 44 – Command: Write 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+4*d	Index							
9+4*d								
10+4*d	Sub-Index							
11+4*d	reserved							
12+4*d – 1465	Payload Data							

d: if seg = Initiate d = 1
else d = 0

Table 45 – Write by Index Request Field Interpretation

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

6.3.4.2.1.2 Command: Read by Index

Usingthe 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.

Table 46 – Command: Read by Index Request

	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							

Table 47 – Read by Index Request Field Interpretation

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

6.3.4.2.1.3 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.

Table 48 – Command: Write All 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+4*d	Index							
9+4*d								
10+4*d	reserved							
11+4*d	reserved							
12+4*d – 1452	Payload Data							

d: if seg = Initiate d = 1
else d = 0

Table 49 – Write All by Index Request Field Interpretation

Field	Abbr.	Description	Value
Index	i	Specifies an entry of the device object dictionary	0 – 65.535

6.3.4.2.1.4 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.

Table 50 – Command: Read All by Index Request

	Bit Offset							
Byte Offset	7	6	5	4	3	2	1	0
0 – 7	Command Layer (fixed part)							
8 – 9	Index							
10 – 11	reserved							

Table 51 – Read All by Index Request Field Interpretation

Field	Abbr.	Description	Value
Index	i	Specifies an entry of the device object dictionary	0 – 65.535

6.3.4.2.1.5 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.

Table 52 – Command: Write by Name 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+4*d 9+4*d	Offset Payload Data (k)							
10+4*d … k-1 4-aligned	Name							
k – 1465	Payload Data							

k = Offset of the name length; k < 1464, 4-aligned
d: if seg = Initiate d = 1
else d = 0

Table 53 – Write by Name Request Field Interpretation

Field	**Abbr.**	**Description**	**Value**
Offest 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 – 1464
Name	n	Specifies an entry of the device object dictionary	see below

- The name may not be terminated by a \0.
- The definitions made in IEC 61131-3 for identifiers are adapted to the name conventions[10]. 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.
  - At least six unique characters of shall be supported in all systems
- The payload data must be 4-byte-aligned. Therefore the name may have to be 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.

6.3.4.2.1.6 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.

[10] The IEC 61131-3 defines several keywords utilized as individual syntax elements. These keywords shall not be used as a parameter name. EPL does not define further keywords.

Table 54 – Command: Read by Name 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+4*d … k	Name							

k < 1465

d: if seg = Initiate d = 1
else d = 0

Table 55 – Read by Name Request Field Interpretation

Field	Abbr.	Description	Value
Name	N	Specifies an entry of the device object dictionary	see subclause 0

The payload data must be 4-byte-aligned. Therefore the name may have to be padded.

6.3.4.2.2 File Transfer

A simple File transfer protocol is defined.

For file access, in addition to to the name conventions (subclause 0) 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.

6.3.4.2.2.1 Command: File Write

The File Write protocol combines several typical operation system commands for file access:

- File open
- File seek
- File write

Table 56 – Command: File Write 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+4*d	Address							
9+4*d								
10+4*d								
11+4*d								
12+4*d	Offset Payload Data (k)							
13+4*d								
14+4*d	reserved							
15+4*d								
16+4*d	File Name							
….								
k-1								
k – 1464	Payload Data							

k = Offset of the payload data; k < 1464, 4-aligned
d: if seg = Initiate d = 1
else d = 0

Table 57 – File Write Request Field Interpretation

Field	**Abbr.**	**Description**	**Value**
Address	addr	Address of the data from the beginning of the file.	$0 – 2^{32}-1$
Offest 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 – 1464
File Name	fn	File name (complete path)	

The Address field indicates the relative address of the data in the file.

6.3.4.2.2.2 Command: File Read

Table 58 – Command: File Read 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+4*d	Address							
9+4*d								
10+4*d								
11+4*d								
12+4*d	File Name							
….								
k								

k < 1465
d: if seg = Initiate d = 1
else d = 0

Table 59 – File Read Request Field Interpretation

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)	

6.3.4.2.3 Variable groups

6.3.4.2.3.1 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.

6.3.4.2.3.1.1 Write Multiple Parameter by Index Request

Table 60 – Command: 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+4*d	Byte Offset of next Data Set (k)							
9+4*d								
10+4*d								
11+4*d								
12+4*d	Index							
13+4*d								
14+4*d	Sub-Index							
15+4*d	reserved						Padding Length	
16+4*d	Payload Data							
..								
k-1								
k (4-aligned)	Byte Offset of next Data Set (m)							
k+1								
k+2								
k+3								
k+4	Index							
k+5								
k+6	Sub-Index							
k+7	reserved						Padding Length	
k+8	Payload Data							
..								
m-1								
m (4-aligned)	…							
…	…							

d: if seg = Initiate d = 1
else d = 0

Table 61 – Write Multiple Parameter by Index Request Field Interpretation

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 end of the fix command header (beginning with byte offset 8) 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
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

6.3.4.2.3.1.2 Write Multiple Parameter by Index Response

Table 62 – Command: Write Multiple Parameter by Index Response

<table>
<tr><td></td><td colspan="8">Bit Offset</td></tr>
<tr><td>Byte Offset</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>0 – 7</td><td colspan="8">Command Layer (fixed part)</td></tr>
<tr><td>8 – 11</td><td colspan="8">Data Size (only if Segmentation = Initiate)</td></tr>
<tr><td>8+4*d</td><td colspan="8" rowspan="2">Index</td></tr>
<tr><td>9+4*d</td></tr>
<tr><td>10+4*d</td><td colspan="8">Sub-Index</td></tr>
<tr><td>11+4*d</td><td>Sub-Abort</td><td colspan="7">reserved</td></tr>
<tr><td>12+4*d</td><td colspan="8" rowspan="4">Sub-Abort Code</td></tr>
<tr><td>13+4*d</td></tr>
<tr><td>14+4*d</td></tr>
<tr><td>15+4*d</td></tr>
<tr><td>16+4*d</td><td colspan="8" rowspan="2">Index</td></tr>
<tr><td>17+4*d</td></tr>
<tr><td>18+4*d</td><td colspan="8">Sub-Index</td></tr>
<tr><td>19+4*d</td><td>Sub-Abort</td><td colspan="7">reserved</td></tr>
<tr><td>20+4*d</td><td colspan="8" rowspan="4">Sub-Abort Code</td></tr>
<tr><td>21+4*d</td></tr>
<tr><td>22+4*d</td></tr>
<tr><td>23+4*d</td></tr>
<tr><td>24+4*d</td><td colspan="8">…</td></tr>
<tr><td>…</td><td colspan="8">…</td></tr>
<tr><td></td><td colspan="8"></td></tr>
</table>

d: if seg = Initiate d = 1
else d = 0

Table 63 – Write Multiple Parameter by Index Response Field Interpretation

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 served by the server	0: transfer ok 1: abort transfer
reserved	res	Reserved for future use	0
Sub-Abort Code	sac	Reason of the sub-abort	see Table 42

In the response a list of all invalid object accesses is transferred.

The Abort flag (a) in the Command Header is set to signal a general error condition.

The list entries consist of the index and sub-index of the object, a Sub-Abort flag (sa), corresponding to the response of the Read Multiple Parameter command (next subclause), and the sub-abort code (sac).

If all accesses are valid and processed by the server the Abort flag (a) in the response is not set and the command specific header is empty, i.e. the list of faulty accesses is empty.

6.3.4.2.3.2 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.

6.3.4.2.3.2.1 Read Multiple Parameter by Index Request

Table 64 – Command: Read 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+4*d	Index							
9+4*d								
10+4*d	Sub-Index							
11+4*d	reserved							
12+4*d	Index							
13+4*d								
14+4*d	Sub-Index							
15+4*d	reserved							
16+4*d	…							
…	…							

d: if seg = Initiate d = 1
else d = 0

Table 65 – Read Multiple Parameter by Index Request Field Interpretation

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
reserved	res	Reserved for alignment	0

6.3.4.2.3.2.2 Read Multiple Parameter by Index Response

Table 66 – Command: 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+4*d	Byte Offset of next Data Set (k)							
9+4*d								
10+4*d								
11+4*d								
12+4*d	Index							
13+4*d								
14+4*d	Sub-Index							
15+4*d	Sub-Abort	reserved					Padding Length	
16+4*d	Payload Data / Sub-Abort Code							
.								
k-1								
k	Byte Offset of next Data Set (m)							
k+1								
k+2								
k+3								
k+4	Index							
k+5								
k+6	Sub-Index							
k+7	Sub-Abort	reserved					Padding Length	
k+8	Payload Data / Sub-Abort Code							
..								
m-1								
m	…							
…	…							

d: if seg = Initiate d = 1
else d = 0

Table 67 – Read Multiple Parameter by Index Response Field Interpretation

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 end of the fixed command header (beginning with Byte Offset 8) 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 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	ac	Reason of the abort	see Table 42

6.3.4.2.4 Parameter Services

6.3.4.2.4.1 Command: Maximum Segment Size

The Maximum Segment Size 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 transfer the maximum segment size.

Table 68 – Command: Maximum Segment Size

	Bit Offset							
Byte Offset	7	6	5	4	3	2	1	0
0 – 7	Command Layer (fix part)							
8 – 9	MSS Client							
10 – 11	MSS Server							

Table 69 – Maximum Segment Size Field Interpretation

Field	Abbr.	Description	Value
MSS Client	mssc	MSS of the client	256 – 1458
MSS Server	msss	MSS of the server If 0000_h the length is not indicated (request from client to server)	256 – 1458

The MSS is limited to 1458, because an Ethernet frame can carry a maximum of 1500 bytes. The remaining bytes are needed for the protocol overhead (IP, UDP, EPL Sequence Layer and the fixed part of the EPL Command Layer), see 0.

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.

Used MSS = min {mssc; msss}

6.3.4.2.4.2 Command: Link Name to Index

Under consideration.

6.3.5 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 protocol is used to access the data.

The Header of the container starts with the fixed part of the EPL Command Layer protocol but

- the reserved field in the Command Layer for the asynchronous period (Byte Offset 0) is now used for a simple Sequence Layer.
- as the container has a fixed length, the valid data length has to be indicated. Therefore the field "valid payload length" is inserted in byte-offset 7 (original a reserved byte).

Up to 255 bytes of payload data can be transferred in a container.

Table 70 – Command: Write by Index Request via PDO

	Bit Offset							
Byte Offset	7	6	5	4	3	2	1	0
0	Sequence Layer embedded in PDO							
1	Transaction ID							
2	Res-ponse	Abort	Segmentation		reserved			
3	Command ID							
4 – 5	Index							
6	Sub-Index							
7	Valid Payload Length							
8 – (k-8)	Payload Data							

k = Length of container in byte

Table 71 – Write by Index Request via PDO Field Interpretation

Field	**Abbr.**	**Description**	**Value**
Sequence Layer			
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 1: Abort transfer
Response	rsp	Request / Response	0: Request 1: Response
Segmentation	seg	Differentiates between expedited and segmented transfer	0: Expedited Transfer 1: Initiate Transfer 2: Segment 3: Transfer Complete
Command ID	cid	Specifies the command	See Table 43
Command ID specific			
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
Valid Payload Length	vpl	Length of valid payload data in the container in bytes.	0 – 255

The container needs a minimum 8 bytes of header information.

Remark:
Though not indicated in the Table 70 it is possible (but not recommended) to perform a segmented transfer in the container. In this case the data size field must be inserted for the initiate frame as defined in the EPL Command Layer Protocol so the header length will be 12 bytes.

The embedded SDO transfer establishes a peer-to-peer communication channel between two devices. If a device needs to transfer data using this method to several other devices it must establish an SDO communication channels for each.

The client SDO container (CSDO) and the server SDO container (SSDO) parameter are described in the SDO communication parameter object. For each SDO pair the communication parameters are mandatory.

6.3.6 Object Description

6.3.6.1 Object 0422$_h$: SDO_ParameterRecord_TYPE

The SDO communication parameter contains the following informations:

Table 72 – SDO Parameter Record (data type)

Index	Sub-Index	Field in SDO Parameter Record	Data Type
0422h	0_h	Number of supported entries	UNSIGNED8
	1_h	EPL Node ID of SDO client	UNSIGNED8
	2_h	EPL Node ID of SDO server	UNSIGNED8
	3_h	Max. data length of the container (incl. header) in byte	UNSIGNED8
	4_h	Server Response history size or Client Request history size (for sequence layer)	UNSIGNED6

6.3.6.2 Object 1200$_h$ – 127F$_h$: 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 must be mapped.

To allow access by name "_*XXh*" shall be replaced by a name index. Name index shall be "_*00h*" if object index is 1400$_h$. It shall be incremented up to "_*FFh*" corresponding to object index 14FF$_h$.

6.3.6.3 Object 1280$_h$ – 12FF$_h$: 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 must be mapped.

To allow access by name "_*XXh*" shall be replaced by a name index. Name index shall be "_*00h*" if object index is 1400$_h$. It shall be incremented up to "_*FFh*" corresponding to object index 14FF$_h$.

6.4 Process Data Object (PDO)

The real-time data transfer is performed by means of Process Data Objects (PDO).

PDO communication in EPL is always performed isochronously via PReq and PRes frames. The PRes frames sent as broadcasts or multicasts following the publisher/subscriber scheme, while the PReq frames with unicast addresses follow the master/slave relationship.

Data type and mapping of application objects into PDOs is determined by the corresponding PDO mapping structures within the Device Object Dictionary. 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.

The length of PDOs of a device is application-specific and has to be specified via the corresponding mapping object.

There are two uses for PDOs. The first is data transmission and the second data reception. Transmit PDOs (TPDOs) and Receive PDOs (RPDOs) can be distinguished. Devices supporting TPDOs are PDO producer or PDO masters and devices which are able to receive PDOs are called PDO consumer or PDO slaves.

PDOs are described by the PDO communication parameter and the PDO mapping parameter. For each PDO the pair of communication and mapping parameter is mandatory. The structures of these data types are explained.

- PDO communication parameter describes the communication capabilities of the PDO.
- PDO mapping parameter describes a mapping for each object contained in PDO payload to object dictionary entries and vice versa

The indices of the corresponding Object Dictionary entries are computed according to the following rules

6.4.1 PDO Mapping Version

The PDO Mapping version is defined in PDO_CommParamRecord_TYPE.MappingVersion.

It shall be used by PDO_TxMappParam_*XXh*_AU64.MappingVersion.This version shall be transmitted by the producer with every PDO and shall be checked by the PDO subscriber.

If the value received by the consumer differs from the corresponding entry of PDO_RxCommParam_*XXh*_REC.MappingVersion, the PDO shall be ignored.

The usage of the PDO Mapping version is application specific.

The higher nibble shall be used as a main version and the lower nibble as a sub version.

Table 73 – Structure of the Mapping versions:

High nibble	Low nibble
Main version	Sub version

If the PDO Mapping is changed in a compatible manner e.g. expanding the PDO contents, the sub version should be incremented.

PDOs with the same Mapping main version but different sub version should be accepted.

6.4.2 Container

A container may be used to exchange objects with complex data types (refer 6.3.3.2).

For every Container a referencing object shall be implemented. The index of the referencing object is written in the corresponding PDO_TxMappParam_*XXh*_AU64.

6.4.3 Multiplexed timeslots

EPL supports communication classes which determine the cycles in which nodes are to be operated.

- Cyclic
  Cyclic data is exchanged in every single EPL cycle.
- Multiplexed
  Multiplexed data is not exchanged in every single EPL cycle. For the whole set M of multiplexed data from all nodes, only a limited amount of isochronous time slots S is reserved. Thus, each cycle only S data frames of M are transferred. The next S data frames are transferred the following cycle etc. S and M are configurable.

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

This procedure can be used (e.g. in "Motion Control") so that a few master axes transmit their actual positions to a large number of slave axes. The EPL unit for the master axes is configured to cyclic. The EPL unit for the slave axes is configured to „multiplexed“. The master axes can transmit their data to the (monitoring) slave axes in each cycle. The slave axes also take part in the communication in a slower cycle.

This kind of nodes makes it possible to operate a very large number of nodes at low cycle times.

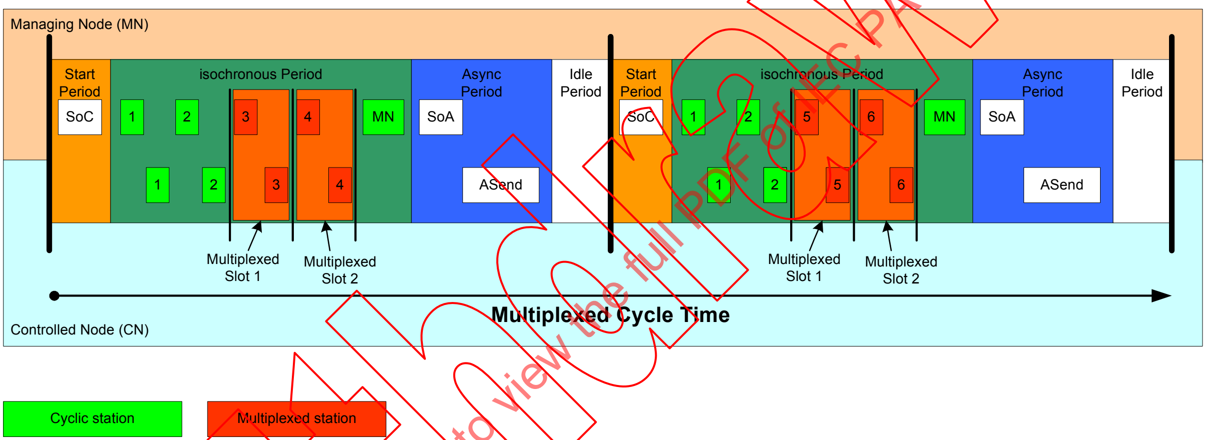


Figure 59 – Multiplexed EPL Cycle

The information about the Multiplexed timeslots and stations is part of the PDO_CommParamRecord_TYPE:

- MultiplexedStation_BOOL holding the information whether it is a Multiplexed station.
- MultiplexedCycleTime_U32 contains the Multiplex Cycle time in µs, this information is necessary for the MN to generate the Multiplex cycle and for the CN for monitoring purposes.
- MultiplexedSlotCount_U8 is only for the MN and contains the amount of Multiplexed slots per cycle.

6.4.4 Transmit PDOs

The TPDO communication parameters are described by indices PDO_TxCommParam_*XXh*_REC. A CN has only one TPDO, therefore only the first index is implemented.

The TPDO mapping parameters are described by indices PDO_TxMappParam_*XXh*_AU64. A CN has only one TPDO, therefore only the first index is implemented.

If sub-index 0 of the mapping object is 0 the RD Flag of the TPDO is reset (TPDO is invalid).

Sending PDO data is implicitly isochronous for a node in the state NMT_xS_OPERATIONAL. In NMT_xS_READY_TO_OPERATE (isochronously communicating substate), PDO data are sent in the same way, but they are not valid. (refer to 7.1.3 and 7.1.4).

6.4.5 Receive PDOs

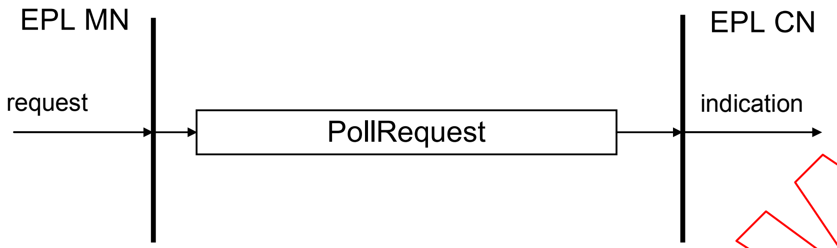
The PDO data released may be valid in the nodes in the state NMT_xS_OPERATIONAL and transfered to the communication objects assigned to them by the RPDO mapping parameters. The application accesses the received PDO data by reading out these communication objects.

If the length of the data actually received is less than the length of the mapped objects the received data has to be ignored and a fault situation occurs.

6.4.5.1 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.



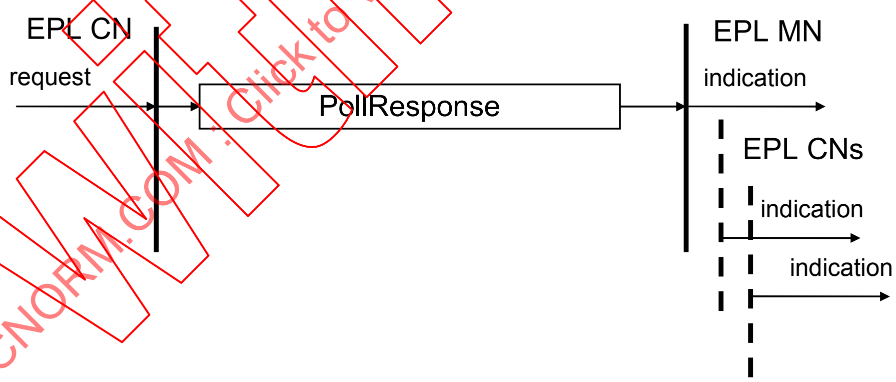
The following data elements in the PReq frame (For the frame structure see 4.6.1.1.3) are relevant for PDO transport:

- The RD flag indicates if the PDO data are valid. If the bit is 0_b, the PDO data are not valid and shall not be interpreted by the EPL CN.
- Size indicates the user data length of the PDO payload data.
- Payload indicates the PDO data.

6.4.5.2 PDO via PRes

PDO via PRes transmission follows the publisher/subscriber relationship as described in 2.3.2.

PDO via PRes 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 are valid. If the bit is 0_b, the PDO data are not valid and shall not be interpreted by the EPL CN.
- Size indicates the user data length of the PDO payload data.
- Payload indicates the PDO data.

6.4.6 PDO Error Handling

6.4.6.1 Dynamic Errors

If an incompatible PDO Mapping version is received, the PDO has to be ignored.

This error situation has to be logged and signaled to the application. Normally this error occurs many times, so the error has to be logged and signaled once for every received wrong PDO mapping version.

Error code	Description
E_PDO_wrong_Mapping_version	PDO with wrong Mapping version received, PDO ignored

If a PDO is received which is shorter then the amount of mapped objects, the PDO has to be ignored.

This error situation has to be logged and signaled to the application. Normally this error occurs many times, so the error has to be logged and signaled once for each PDO.

Error code	Description
E_PDO_length_too_short	PDO length too short, PDO ignored

6.4.6.2 Configuration Errors

If an attempt to change the PDO mapping results in an amount of mapped objects that exceeds the configured Poll size, this attempt has to be rejected.

The same check has to be done if the Poll In Size or Poll Out Size should be changed.

Error code	Description
E_PDO_Mapping_exceeds_Poll_Size	Mapping exceeds Poll Size

6.4.7 Object Description

6.4.7.1 Object 0420_h:PDO_CommParamRecord_TYPE

Index	0420h
Name	PDO_CommParamRecord_TYPE
Category	O, M if PDO supported

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00_h
Description	NumberOfEntries_U8
Data Type	UNSIGNED8

- **Sub-Index 01_h: NodeID_U8**

Sub-Index	01_h
Description	NodeID
Data Type	UNSIGNED8

- **Sub-Index 02_h: MultiplexedStation_BOOL**

Sub-Index	02_h
Description	Multiplexed station
Data Type	BOOLEAN

- **Sub-index 03_h: MultiplexedCycleTime_U32**

Sub-index	03_h
Description	Multiplexed Cycle time [us]
Data Type	UNSIGNED32

- **Sub-index 04_h: MultiplexedSlotCount_U8**

Sub-index	04_h
Description	Multiplexed Slot count
Data Type	UNSIGNED8

- **Sub-Index 05_h: MappingVersion_U8**

Sub-Index	05_h
Description	PDO Mapping Version
Data Type	UNSIGNED8

6.4.7.2 Object 0421$_h$:PDO_MappParamArray_TYPE

For every PDO up to 254 objects may be mapped.

Every object has a theoretical maximum length of 65535 bit.

The offset inside the PDO shall be defined for every mapped object. Only the offset shall be used for the adressing of mapped objects.

Sub index 0 shall determine the number of objects that have been mapped.

To change the PDO mapping, first the PDO has to be deleted and then the sub-index 0 must be set to 0 (mapping is deactivated). The objects may then be remapped.

Index	0421h
Name	PDO_MappParamArray_TYPE
Category	O, M if PDO supported

- **Sub-Index 00$_h$: NumberOfEntries_U8**

Sub-Index	00$_h$
Description	NumberOfEntries_U8
Data Type	UNSIGNED8

- **Sub-Index 01$_h$ … FE$_h$: 1st … 254th ObjectMapping_U64**

Sub-Index	01$_h$ … FE$_h$
Description	Objects to be mapped
Data Type	UNSIGNED64

- **Sub-Index 01$_h$ … FE$_h$ Value Interpretation**

  Every ObjectMapping_U64 entry is structured as described below:

Table 74 – Structure of PDO Mapping Entry

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 inside the PDO (Bit count)
6 – 7	Length	Length for the mapped object (Bit count)

6.4.7.3 Object 1400_h – $14FF_h$: PDO_RxCommParam_XXh_REC

The validity of the respective object depends on the the NumberOfEntries_U8 entry of the respective RPDO mapping index .

To allow access by name "*_XXh*" shall be replaced by a name index. Name index shall be "*_00h*" if object index is 1400_h. It shall be incremented up to "*_FFh*" corresponding to object index $14FF_h$.

Index	1400h – 14FFh
Name	PDO_RxCommParam_XXh_REC
Object Code	RECORD
Data Type	PDO_CommParamRecord_TYPE
Category	O, M if PDO supported

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00_h
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	5
Default value	5

- **Sub-Index 01_h: NodeID_U8**

Sub-Index	01_h
Description	NodeID of the node transmitting the corresponding PRes 0 is reserved for PReq
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 1 … 239, 240, 241 – 252, 253, 254
Default value	0

- **Sub-Index 02_h: MultiplexedStation_BOOL**

Sub-Index	02_h
Description	Multiplexed Station
Data Type	BOOLEAN
Entry Category	O
Access	rw
PDO Mapping	No
Value range	FALSE, TRUE
Default value	FALSE

- **Sub-Index 03$_h$: MultiplexedCycleTime_U32**

This parameter is for the CN only for monitoring purposes.

Sub-Index	03$_h$
Description	Multiplexed Cycle Time [µs]
Data Type	UNSIGNED32
Entry Category	O
Access	rw
PDO Mapping	No
Value range	0 … 2^{32}-1
Default value	0

- **Sub-Index 04$_h$: MultiplexedSlotCount_U8**

not supported at RPDO

- **Sub-Index 05$_h$: MappingVersion_U8**

Sub-Index	05$_h$
Description	Version of the RPDO-mapping. 0 = no mapping version available
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0 … 255
Default value	0

6.4.7.4 Object 1600_h – $16FF_h$: PDO_RxMappParam_XXh_AU64

To allow access by name "*_XXh*" shall be replaced by a name index. Name index shall be "*_00h*" if object index is 1600_h. It shall be incremented up to "*_FFh*" corresponding to object index $16FF_h$.

Index	1600h – 16FFh
Name	PDO_RxMappParam_XXh_AU64
Object Code	ARRAY
Data Type	PDO_MappParamArray_TYPE
Category	O, M if PDO supported

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00_h
Description	Number mapped objects, 0 indicates that the mapping index and the corresponding RPDO communication index (6.4.7.3) is unused
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 1 … 254
Default value	0

- **Sub-Index 01_h … FE_h: ObjectMapping_U64**

Sub-Index	01_h … FE_h
Description	1^{st} … 254^{th} Mapping entry
Entry Category	O
Access	rw
PDO Mapping	No
Value range	UNSIGNED64
Default value	0

6.4.7.5 Object 1800_h – $18FF_h$: PDO_TxCommParam_XXh_REC

The validity of the respective object depends on the the NumberOfEntries_U8 entry of the respective TPDO mapping index (0).

To allow access by name "*_XXh*" shall be replaced by a name index. Name index shall be "*_00h*" if object index is 1800_h. It shall be incremented up to "*_FFh*" corresponding to object index $18FF_h$.

Index	1800h – 18FFh
Name	PDO_TxCommParam_XXh_REC
Object Code	RECORD
Data Type	PDO_CommParamRecord_TYPE
Category	O, M if PDO supported

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00_h
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	5
Default value	5

- **Sub-Index 01_h: NodeID_U8**

Sub-Index	01_h
Description	CN: not used (0) MN: NodeID of the corresponding PReq target (1 – 239, 253, 254)
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0 ... 240
Default value	0

- **Sub-Index 02_h: MultiplexedStation_BOOL**

Sub-Index	02_h
Description	Multiplexed Station
Data Type	BOOLEAN
Entry Category	O
Access	rw
PDO Mapping	No
Value range	FALSE, TRUE
Default value	FALSE

- **Sub-Index 03_h: MultiplexedCycleTime_32**

This parameter is for the CN only for monitoring purposes.

Sub-Index	03_h
Description	Multiplexed Cycle Time [µs]
Data Type	UNSIGNED32
Entry Category	O
Access	rw
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

- **Sub-index 04_h: MultiplexedSlotCount_U8**

Sub-index	04_h
Description	Multiplexed Slot count
Data Type	UNSIGNED8
Entry Category	O
Access	rw
PDO Mapping	No
Value range	UNSIGNED8
Default value	0

- **Sub-Index 05_h: MappingVersion_U8**

Sub-Index	05_h
Description	Version of the TPDO-mapping. 0 = no mapping version available
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	UNSIGNED8
Default value	0

6.4.7.6 Object $1A00_h$ – $1AFF_h$: PDO_TxMappParam_*XXh*_AU64

If Sub index 0 is 0 the RD Flag of the TPDO shall be reset (TPDO is invalid).

To allow access by name "*_XXh*" shall be replaced by a name index. Name index shall be "*_00h*" if object index is $1A00_h$. It shall be incremented up to "*_FFh*" corresponding to object index $1AFF_h$.

Index	1A00h – 1AFFh
Name	PDO_TxMappParam_XXh_AU64
Object Code	ARRAY
Data Type	PDO_MappParamArray_TYPE
Category	O, M if PDO supported

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00_h
Description	Number mapped objects
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	1 … 254
Default value	1

- **Sub-Index 01_h … FE_h: ObjectMapping**

Sub-Index	01_h … FE_h
Description	1^{st} … 254^{th} Mapping entry
Entry Category	M
Access	rw
PDO Mapping	No
Value range	UNSIGNED64
Default value	0

6.5 Synchronisation (SYNC)

Synchronisation is an integral feature of the Isochronous EPL Cycle (cf. 4.2.4.1).

Synchronisation shall be implemented device specific. It shall be based on the reception of the SoC frame.

6.6 Error Handling and Diagnostics

6.6.1 Error Signalling

This subclause describes how to record errors and events which are generated by an EPL 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.

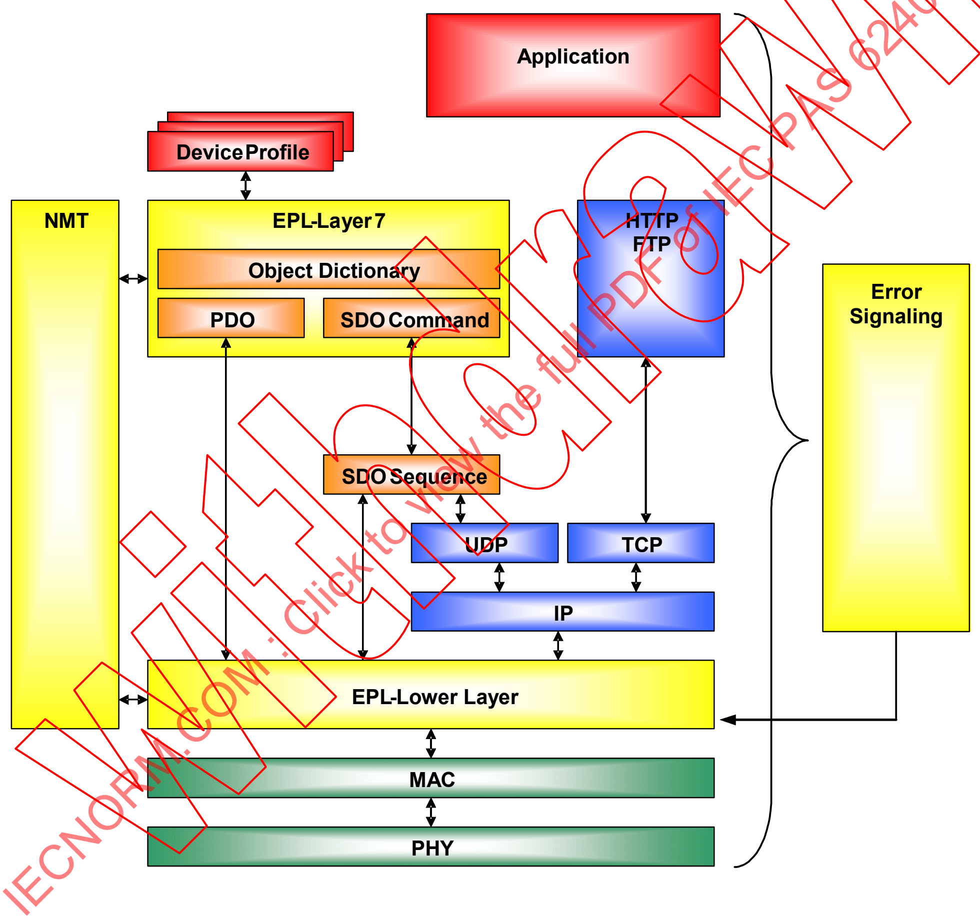


Figure 60 – Error Signaling - Reference Model

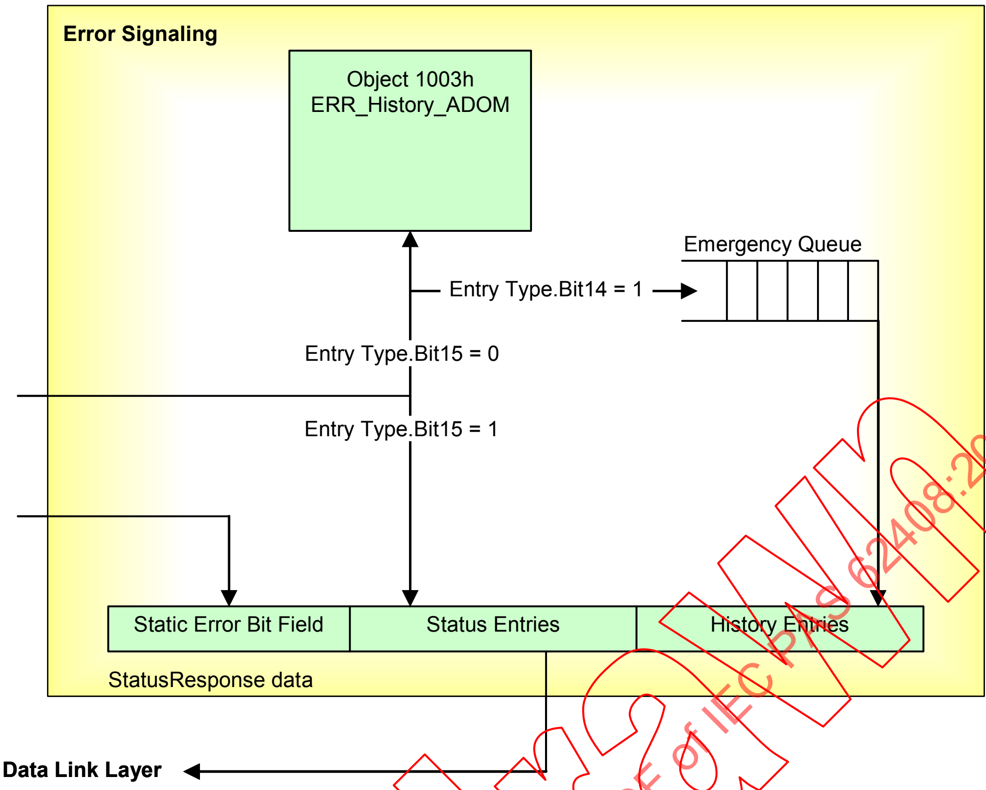


Figure 61 – Error Signaling - Overview

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 Status Response data and pass it to the DLL at the next query.

6.6.1.1 Error Register

The object ERR_ErrorRegister_U8 is compatible to the object 'error register' of the standard communication profile CiA DS 301.

6.6.1.2 Error History

This subclause describes the function and the data format of the error history in object ERR_History_ADOM.

The object holds the errors and events that have occurred on the device. In doing so it provides a history.

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.

The entries HistoryEntry_DOM of the object ERR_History_ADOM have the following format:

Table 75 – Format of one entry

Octet Offset	Description
0 - 1	Entry Type
2 - 3	Error Code
4 - 11	Time Stamp
12 - 19	Additional Information

Table 76 – Description of one entry

Field	Abbr.	Description	Value
Entry Type	type	see Table 77	see Table 77
Error Code	code	Depending on the Entry Type the codes have to be described in device profile or the device description	UNSIGNED16
Time Stamp	time	Nettime at the time when the error/event is entered in the history	UNSIGNED64
Additional Information	add	The format of this field is to be specified in the in the device profile (profile specific) or device description (vendor specific) in case the used device profile allows vendor specific areas in this field.	UNSIGNED64

Table 77 – Format of the field Entry Type

Octet	Bit	Value	Description
0 - 1	15	0_b	History Entry
		1_b	Status Entry in StatusResponse frame (Bit 14 shall be set to 0_b)
	14	0_b	History only
		1_b	Additional to the history the entry shall also be entered in to the Emergency Queue of the Error Signaling.
	13 - 12 (Mode)	0_h	Not allowed in the history object ERR_History_ADOM. 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_h	An error has occurred and is active (e.g. short circuit of output detected)
		2_h	An active error was cleared (e.g. no short circuit anymore)
		3_h	An error / event occurred
	11 - 0 (Profile)	000_h	Reserved
		001_h	The field Error Code in Table 75 contains a vendor specific error code
		002_h	The field Error Code in Table 75 contains EPL specific errors (Network errors, communication errors, data link errors …)
		003_h - FFF_h	The field Error Code in Table 75 contains device profile specific errors (Cross reference to profile number list)

6.6.1.3 Error Signaling Bits

To avoid that the MN has to poll the history (Object 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 CNs StatusResponse frame have changed.

The following bits shall be used for a reliable transmission from the CN to the MN:

Table 78 – Error Signaling Bits

Field	Abbr.	Description
Exception Reset	ER	Initialization of the Error Signaling When a CN receives the value 1_b it shall reset its EN bit to 0_b and clear the Emergency Queue. The MN shall send the ER bit with the following frames: • SoA(StatusRequest) • SoA(IdentRequest)
Exception Clear	EC	In this bit the CN shall mirror the last received ER from the MN. This is required to indicate the MN that the initialization of the Error Signaling was done. A CN shall send the EC bit with the following frames: • ASnd(StatusResponse) • ASnd(IdentResponse)
Exception New	EN	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. A CN shall send the EN bit with the following frames: • PRes • ASnd(StatusResponse) **For isynchronous CNs the MN shall evaluate the EN bit of the ASnd(StatusResponse) only in NMT_CS_PRE_OPERATIONAL_1.**
Exception Acknowledge	EA	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 StatusResponse frame was received by the MN successfully the MN shall set EA=EN. 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. The MN shall send the EA bit with the following frames: • PReq • SoA(UnspecifiedInvite) **Isynchronous CNs shall evaluate the EA bit of the SoA(UnspecifiedInvite) only in NMT_CS_PRE_OPERATIONAL_1.**

6.6.1.4 Initialisation

With this initialisation the MN shall prepare the CN for the Error Signaling.

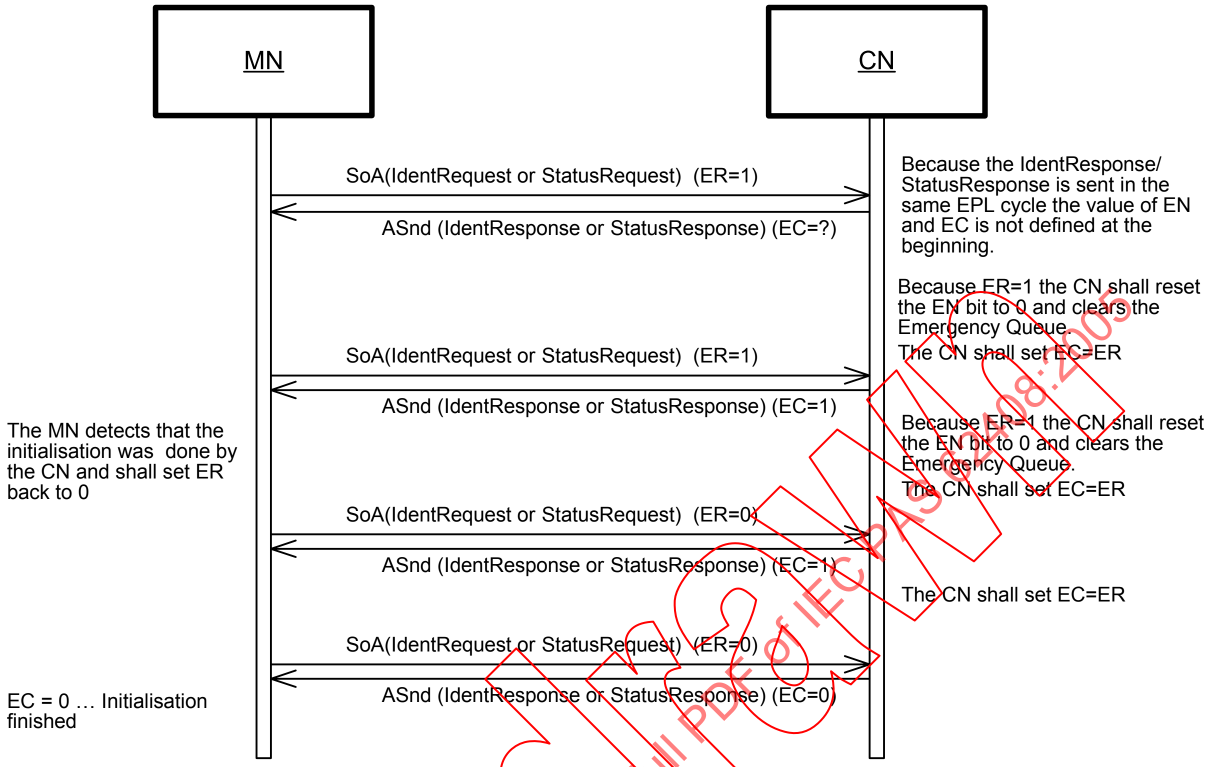


Figure 62 – Error Signaling Initialisation

The IdentRequest and IdentResponse frames of the NMT Boot Up of a CN shall be already used for this initialisation.

6.6.1.5 Error Signaling with RReq and PRes frames

For isynchronous CNs only the PReq and PRes frames shall be used for the Error Signaling.

If an isynchronous CN is in the state NMT_CS_PRE_OPERATIONAL_1 the Error Signaling shall behave like an Async-only CN (6.6.1.6)

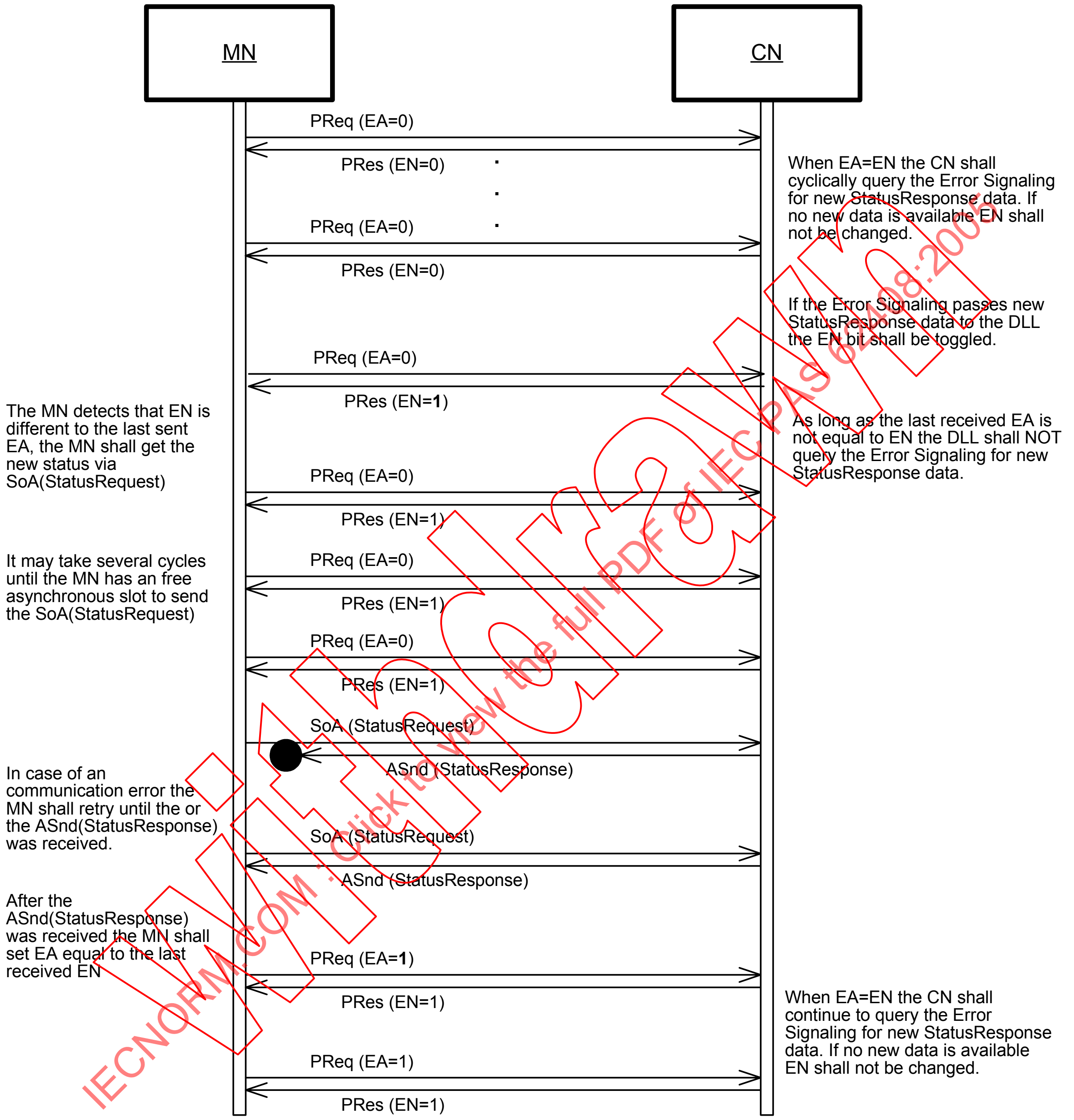


Figure 63 – Error Signaling with PReq and PRes

6.6.1.6 Error Signaling with Async-only CNs

The MN shall periodically send StatusRequest frames to Async-only CNs.
This shall be also done for isynchronous CNs in NMT_CS_PRE_OPERATIONAL_1.

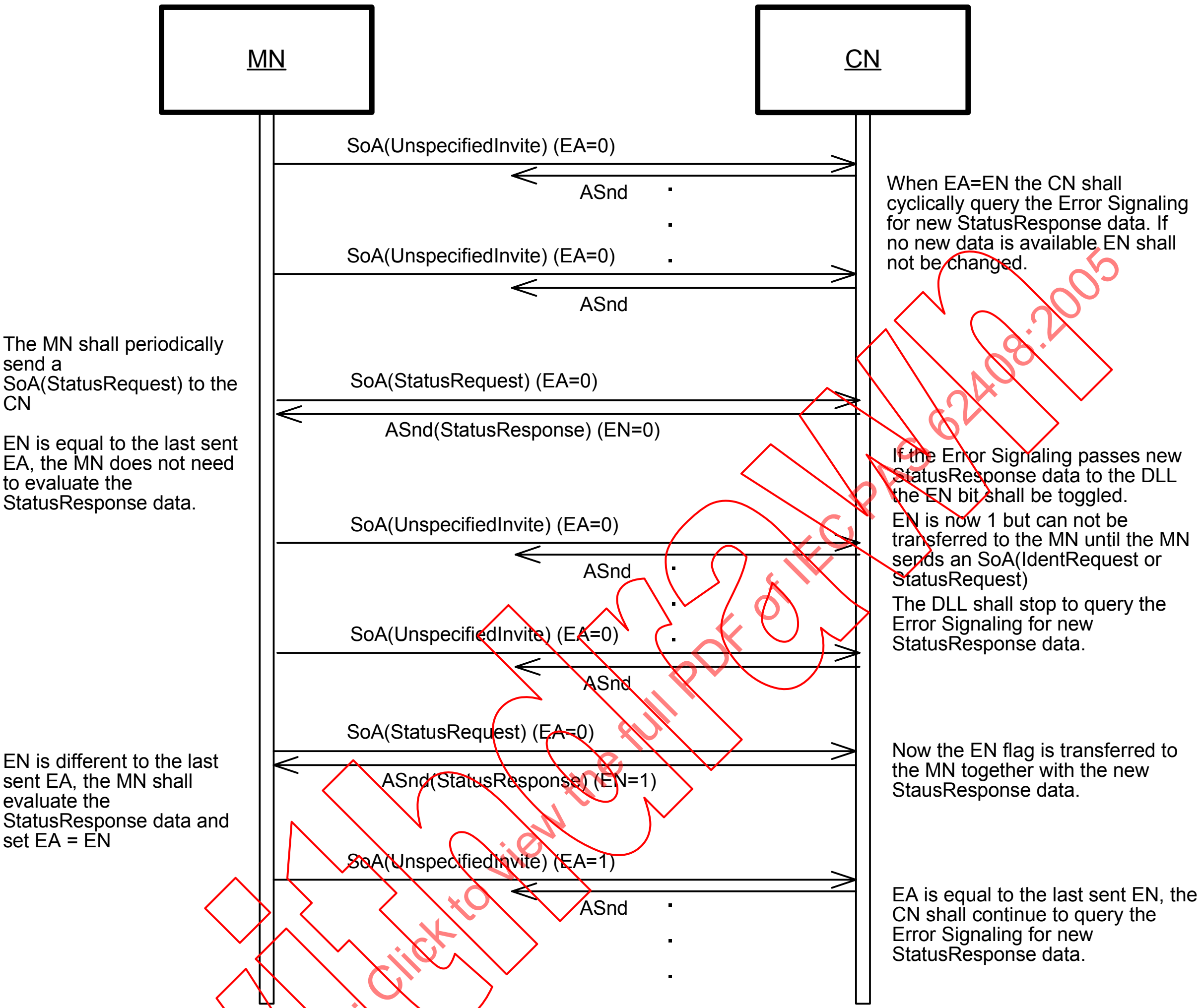


Figure 64 – Error Signaling for Async-only CNs and CNs in NMT_CS_PRE_OPERATIONAL_1

6.6.1.7 Format of StatusResponse Data

Refer to Querverweis Network Management Objects for StatusResponse frame structure.

6.6.1.7.1 Static Error Bit Field

Table 79 – Static Error Bit Field

Octet Offset[11]	Description
7	Content of Object 1001h ERR_ErrorRegister_U8
8	Reserved
9 - 14	Device profile specific errors which can be active(1) or inactive(0) Depending on the device profile a part of this area can be assigned to vendor specific errors. Bits which are defined in the device profile shall be described in a device profile description.

6.6.1.7.2 Status and History Entries

If not all available entries are used an entry with Mode 0 (see Table 77) shall be used to terminate the list and declare 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.

[11] Byte Offset relates to the beginning of the of ASnd service slot

6.6.1.8 Object descriptions

6.6.1.8.1 Object 1001h : ERR_ErrorRegister_U8

Index	1001h
Name	ERR_ErrorRegister_U8
Object Code	VAR
Data Type	INTEGER8
Category	M

- **Value description**

Access	ro
PDO Mapping	Optional
Value range	UNSIGNED8
Default value	No

- **Value Interpretation**

Bit	**M/O**	**Description**
0	M	generic error This bit shall be set to 1_b if the Static Error Bit Field or the Status Entries in the StatusResponse frame show one or more errors. If this bit is 0_b 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

6.6.1.8.2 Object 1003h : ERR_History_ADOM

Index	1003h
Name	ERR_History_ADOM
Object Code	ARRAY
Data Type	DOMAIN
Category	O

- **Sub-index 00$_h$: NumberOfEntries_U8**

sub-index	**00$_h$**
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	0-254
Default value	0

- **Sub-index 01$_h$ - FE $_h$: HistoryEntry_DOM**

sub-index	**01$_h$ - FF$_h$**
Description	HistoryEntry_DOM
Data Type	DOMAIN
Entry Category	O
Access	ro
PDO Mapping	No
Value range	see Table 75 – Format of one entry
Default value	No

6.7 Program Download

In this subclause, a common way for program downloading to a device via its object dictionary is specified. Here only the mechanism for performing the program download is specified but not the structure of program data and not 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 EPL bootstrap-loader) or only parts of a program (e.g. specific tasks of real-time systems). The data structure of the transferred program data has to be specified by the manufacturer (e.g. INTEL-HEX format or binary format).

Further specifications for the program download have to be made in specific device profiles (e.g. in CiA DS-405 for the download of PLC programs).

For the download of the program data a new object is introduced:

Index	Object	Name	Type	Attr.	M/O
$1F50_h$	ARRAY	Download program data PDL_DownloadProgData_ADOM	Domain	rw	O

The sub-objects for the download program data object are:

Index	Sub-index	Field in Download Program Data	Data Type
$1F50_h$	00_h	Number of different programs supported on the node	Unsigned8
	01_h	program number 1	Domain
	02_h	program number 2	Domain
	:		
	FEH	program number 254	Domain

If the download fails, the Device responds with an Abort SDO Transfer (error code 0606 0000_h).

A second object is specified for controlling the execution of stored programs:

Index	Object	Name	Type	Attr.	M/O
1F51H	ARRAY	Program Control PDL_ProgCtrl_AU8	Unsigned8	RW	O

The sub-objects for the Program Control Object are:

Index	Sub-Index	Field in Program Control	Data Type
1F51H	0H	Number of different programs on the node	Unsigned8
	1H	program number 1	Unsigned8
	2H	program number 2	Unsigned8
	:		
	FEH	program number 254	

The range for the sub-objects of the Program Control Object is 0 to 2.

The values have the following meaning:

0 - stop program (W) / program stopped (R)

1 - start program (W) / program running (R)

2 - reset program (W) / program stopped (R)

If the action is not possible, the device responds with an Abort SDO Transfer (error code 0800 0024_h).

A third object is defined to support verification of the version of the stored program number 1 (application software)[12]:

[12] Note that the object NMT_ManufactSwVers_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	Object	Name	Type	Attr.	M/O
1F52H	RECORD	Verify Application Software PDL_LocVerApplSw_REC	Unsigned32	RW	O

The sub-object of the Verify Application Software -object are:

Index	Sub-Index	Field in Program Control	Data Type
1F52H	0H	Number of supported entries NumberOfEntries_U8	Unsigned8
	1H	Application software date ApplSwDate_U32	Unsigned32
	2H	Application software time ApplSwTime_U32	Unsigned32

Application software date contains the number of days since January 1, 1984. Application software time contains the number of milliseconds after midnight (00:00).

Note that only the date and time of a single application program is supported. Dates and times of programs 2 to 254 are not supported. Hence, if the application software is a single entity, it should be the program number 1. 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 into the object 1F54.

6.7.1 EPL manager owned objects

Two objects are specified for verification of the version of the application software at the CNs:

The sub-objects for the Program Control Object are:

Index	Object	Name	Type	Attr.	M/O
$1F53_h$	ARRAY	Expected application SW date PDL_MnExpAppSwDateList_AU 32	UNSIGNED32	rw	O
$1F54_h$	ARRAY	Expected application SW time PDL_MnExpAppSwDateTime_A U32	UNSIGNED32	rw	O

ExpectedApplicationSWDate contains the number of days since January 1, 1984. ExpectedApplicationSWTime contains the number of milliseconds after midnight (00:00).

Sub-Index 0 NrOfSupportedObjects has the RO value 254. Sub-Index i (with i != Node-ID of reserved nodes): Stores the expected application software date/time of the CN that has Node-ID i.

6.8 MN Configuration Manager

The Configuration Manager is an optional application task and has to configure network devices at network boot-up. For this it has to know all the (application dependent) parameter values. This information is set-up in the Device Configuration File (DCF) of each device.

The Configuration Manager has to reside on the MN. For an EPL node there may be a DCF File, this File shall be stored on the MN.

6.8.1 DCF storage

Index	Object	Name	Type	Attr.	M/O
$1F20_h$	ARRAY	CFG_StoreDcfList_ADOM	DOMAIN	rw	O
$1F21_h$	ARRAY	CFG_DcfStorageFormatList_AU8	UNSIGNED8	rw	O

Index $1F20_h$, sub-index 00_h describes the number of entries. This is equal to the maximum possible Node-ID (254_d). Each sub-index points to the Node-ID of the device, for which the DCF belongs.

Downloading the DCF of a device from a Configuration Tool to the Configuration Manager is done by writing the DCF file as a domain to the object $1F20_h$ with the sub-index equal to the devices Node-ID.

Uploading the DCF of a device from the Configuration Manager to a Tool is done by reading the object $1F20_h$ with the sub-index equal to the devices Node-ID.

The filename does not need to be stored separately since every DCF contains its own filename.

Object 1F21$_h$ describes the format of the storage. This allows the usage of compressed formats.

Value	Format
00h	ASCII, not compressed
01$_h$ to FF$_h$	reserved

The device may always store the file compressed internally. The object describes the external behaviour.

If no data had been stored, an SDO Read Request to 1F20$_h$ or 1F21$_h$ is aborted with the error code 0800 0024$_h$ "Data set empty"

6.8.2 Concise configuration storage

The concise device configuration does not contain every information of the DCF. It is recommended to use this if the complete DCF storage is not possible.

The information to be stored consists of the parameter values of the object dictionary entries.

Index	Object	Name	Type	Attr.	M/O
1F22$_h$	ARRAY	CFG_ConciseDcfList_ADOM	DOMAIN	rw	O

Sub-index 00$_h$ describes the number of entries. This is equal to the maximum possible Node-ID (254$_d$). Each sub-index points to the Node-ID of the device, to which the configuration belongs.

The content is a stream with the following structure:

Number of supported entries	Unsigned32
Index 1	UNSIGNED16
Sub-index 1	UNSIGNED8
Data size of parameter 1	UNSIGNED32
Data of parameter 1	DOMAIN
Index 2	UNSIGNED16
Sub-index 2	UNSIGNED8
Data size of parameter 2	UNSIGNED32
Data of parameter 2	DOMAIN
:::::::	
Index n	UNSIGNED16
Sub-index n	UNSIGNED8
Data size of parameter n	UNSIGNED32
Data of parameter n	DOMAIN

The Data Size is counting bytes (i.e. Unsigned16 has size 2; size of Boolean is given as 1).

Downloading the configuration of a device from a Configuration Tool to the Configuration Manager is done by writing the stream to the object 1F22$_h$ with the sub-index equal to the devices Node-ID.

Uploading the configuration of a device from the Configuration Manager to a Tool is done by reading the object 1F22$_h$ with the sub-index equal to the devices Node-ID.

Application hint:

- *An empty data set can be written by the following concise stream:*

Number of supported entries	*0 (UNSIGNED32)*

- *If no data has been stored, an SDO Read Request will return a valid concise stream with the following content:*

Number of supported entries	*0 (UNSIGNED32)*

6.8.3 Check configuration process

EPL defines the object 1020_h Verify Configuration. If a device supports the saving of parameters in non volatile memory, a network configuration tool or a EPL manager can use this object to verify the configuration after a devices reset and to check if a reconfiguration is necessary. The configuration tool has to store the date and time in that object and has to store the same values in the DCF. Now the configuration tool lets the device save its configuration by writing to index 1010_h Sub-Index 1 the signature "save". After a reset the device restores the last configuration and the signature automatically or by request. If any other command changes boot-up configuration values, the device has to reset the object Verify Configuration to 0.

The Configuration Manager compares signature and configuration with the value from the DCF and decides if a reconfiguration is necessary or not. The comparison values are stored on the Configuration Manager in the objects.

Index	Object	Name	Type	Attr.	M/O
$1F26_h$	ARRAY	CFG_ExpConfDateList_AU32	UNSIGNED32	rw	O
$1F27_h$	ARRAY	CFG_ExpConfTimeList_AU32	UNSIGNED32	rw	O

Sub-Index 00_h NrOfSupportedObjects has the ro value 254_d.

Sub-Index i (with i != Reserved EPL Node Addresses): Stores the Configuration date/time of the CN that has Node-ID i.

The usage of Check Configuration is described in 7.6.2.3.1.2.

Application hint: The usage of this object allows a significant speed-up of the boot-up process. If it is used, the system integrator has to consider the that a user may change a configuration value and afterwards activate the command store configuration 1010_h without changing the value of 1020_h. So the system integrator has to ensure a 100% consequent usage of this feature. It should be a feature of configuration tools to force or at least encourage a correct usage of object 1020_h.

6.8.4 Request configuration

In applications there might be situations, where it is necessary to configure the CNs at run-time. An example is, that a CN fails and re-boots. The NMT master will recognize this and will inform the application (see 7.6.2.3.1.2). With the object Configure CN the application is able to tell the Configuration Manager, that it shall configure that CN.

Another example is the connection of a new machine part with several devices. The application needs a possibility to start the Configuration Manager at least for the new nodes.

Index	Object	Name	Type	Attr.	M/O
$1F25_h$	ARRAY	CFG_ConfSlaveList_AU32	UNSIGNED32	Sub 00_h: ro Sub 01_h to FE_h: wo	O

Sub-Index 00_h NrOfSupportedObjects has the ro value 255_d.

Sub-Index i (with i != Reserved EPL Node Addresses): Request re-configuration for the CN with Node ID i.

Sub-Index FF_h: Request re-configuration for all Nodes.

To avoid accidental access, the signature ‘conf’ (this equals the UNSIGNED32 number 666E $6F63_h$) has to be written to initiate the process.

If no data had been stored, an SDO Write Request to 01_h to FE_h is aborted with the error code 0800 0024_h "Data set empty". An SDO Write Request to sub-index FF_h returns without error even if there is no data stored.

Application hint: The latter allows to implement simple applications to request the CMT without knowing the actual project configuration. If the application wants to configure the network with more control it can use a loop over all known CNs.

6.8.5 „DEVICE DESCRIPTION FILE“ storage

The DEVICE DESCRIPTION FILE is the base for the DCF and for some devices it may be possible to store the „DEVICE DESCRIPTION FILE“ also. This has some advantages:

- The manufacturer does not have the problem of distributing the „DEVICE DESCRIPTION FILE“ via disks
- Management of different „DEVICE DESCRIPTION FILE“ versions for different software versions is less error prone, if they are stored together
- The complete network settings may be stored in the network. This makes the task of analysing or reconfiguring a network easier for Tools and more transparent for the users.

For those devices which are not able to store their „DEVICE DESCRIPTION FILE“, the Configuration Manager may take over this task. For this the following objects are defined in the Configuration Manager:

Index	Object	Name	Type	Attr.	M/O
$1F23_h$	ARRAY	CFG_StoreDevDescrFileList_ADOM	DOMAIN	rw	O
$1F24_h$	ARRAY	CFG_DevDescrFileFormatList_AU8	UNSIGNED8	rw	O

Index $1F23_h$, sub-index 00_h describes the number of entries. This is equal to the maximum possible Node-ID (254_d). Each sub-index points to the Node-ID of the device, for which the „DEVICE DESCRIPTION FILE“ belongs.

Downloading the „DEVICE DESCRIPTION FILE“ of a device from a Configuration Tool to the Configuration Manager is done by writing the „DEVICE DESCRIPTION FILE“ file as a domain to the object $1F23_h$ with the sub-index equal to the devices Node-ID.

Uploading the „DEVICE DESCRIPTION FILE“ of a device from the Configuration Manager to a Tool is done by reading the object $1F23_h$ with the sub-index equal to the devices Node-ID. If no data had been stored, this is aborted with the error code $0800\ 0024_h$ "Data set empty"

The filename does not need to be stored since every „DEVICE DESCRIPTION FILE“ contains its own filename.

Object $1F24_h$ have the same description and behaviour as object Storage Format in the DCF storage (object $1F21_h$).

6.9 Input from a Programmable Device

6.9.1 Basics

In a network programmable nodes can be characterised as a process having input variables and output variables. The set of variables will be arguments of the program and hence will be only known in a final state when the program has been written. The arguments must be handled as variables located in the object dictionary.

The marking of such parameters depends on the programming system (e.g. IEC 61131-3) and can not be standardised here. But it can be assumed that there is a set of network variables with the logic attribute EXTERN.

Compiling/Linking (or interpreting) a program including EXTERN variables requires relocation information. Within EPL devices this information is the index (and sub-index) of the variable. Most of the programming systems know the mechanism of a resource definition. This can be used to assign the EPL attributes (index, sub-index, rw, Assignment of EPL data type to local data type etc.) to the corresponding symbolic names (variable name in the program). The resource definition may be created with a simple editor by the user or with much more comfort by a configuration tool. On systems with a disk-based file system a direct exchange via the DCF format is possible.

The names of variables have to meet the rules of the underlying programming system. EPL makes no restrictions for this. So this is the responsibility of the programmer/manufacturer.

Defining EXTERN variables requires a rule for distributing the indices. It is called "dynamic index assignment".

6.9.2 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 segment. These segments must not overlap. Variables of same type are gathered in one array. If all elements of an array are defined (sub-index 01_h to FE_h), the next free object of the area is allocated.

In order to allow programmable devices the use of a process picture, they may implement a conversion formula which calculates the offset of a variable in the process picture in direct dependence from the index and sub-index.

Definition of the abstract object segment:

A segment is a range of indexes in the object dictionary with the following attributes:

- Data type

  This is the data type of the objects which can be defined in this segment.
- Direction

  This flag distinguishes between inputs and outputs. The values are 'wo' for outputs and 'ro' for inputs. The distinction is important to know whether the variable can be mapped into a receive PDO (wo) or transmit PDO (ro). This does not concern the access possibilities via SDO.
- Index range

  Range of indices with start index and end index.
- PPOffset

  Offset in the process picture, where the first object of this segment is allocated.

  For byte and multi-byte variables this 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.

Many devices distinguish strictly between different segments in the process picture for different data types. For those devices the PPOffset of the first segment will be 0, the PPOffset 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 forth. If this does not exactly meet the physical configuration, the device software is free to implement this on a logical point of view by using internal segment descriptors/offsets.

Other devices mix different data types in the same segment. For those devices all PPOffset attributes will have the value 0. Configuration Tools which allocate space in that process picture by assigning indexes have to take into account, that in this case indexes have to be left out to avoid overlapping. (For special applications it may be a feature to explicitly overlap variables. This helps interpreting memory cells as different types in debuggers.)

Any mixed form of those two device types is possible.

6.9.3 Object dictionary entries

Accessing the network variables is done via the entries described by the segments. In some applications it is desirable to read or write the complete process picture as one block:

6.9.3.1 Object 1F70$_h$: Process picture

Index	1F70$_h$
Name	Process picture
Object code	RECORD
Category	Optional

- **Entry Description**

Sub-index	00$_h$
Description	Number of entries
Entry category	Mandatory
Access	ro
PDO mapping	No
Value range	02$_h$
Default Value	02$_h$

- **Entry Description**

Sub-index	01$_h$
Description	Selected range
Entry category	Mandatory
Access	rw
PDO mapping	No
Value range	UNSIGNED32
Default value	0000 0000$_h$

- **Entry Description**

Sub-index	02$_h$
Description	Process picture domain
Entry category	Mandatory
Access	rw
PDO mapping	No
Value range	DOMAIN
Default value	No

After writing the selected range in sub-index 01$_h$ the corresponding data can be read from or written to the addressed area with sub-index 02$_h$ as an unstructured stream of bytes.

The structure of *selected range* is as follows:

31	16	15		0	
Data length		Object segment			
MSB					LSB

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 those indexes.

The Data Length gives the maximum amount in bytes for the transfer. If the value is 0, the complete segment is to be accessed.

7 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 EPL 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 PowerUp, PowerDown and reset levels common to MN and CN.

The MN specific branch is described by 7.1.3, 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 Figure 68 the initialisation of the NMT state machine, common to MN and CN is shown. Figure 65 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.

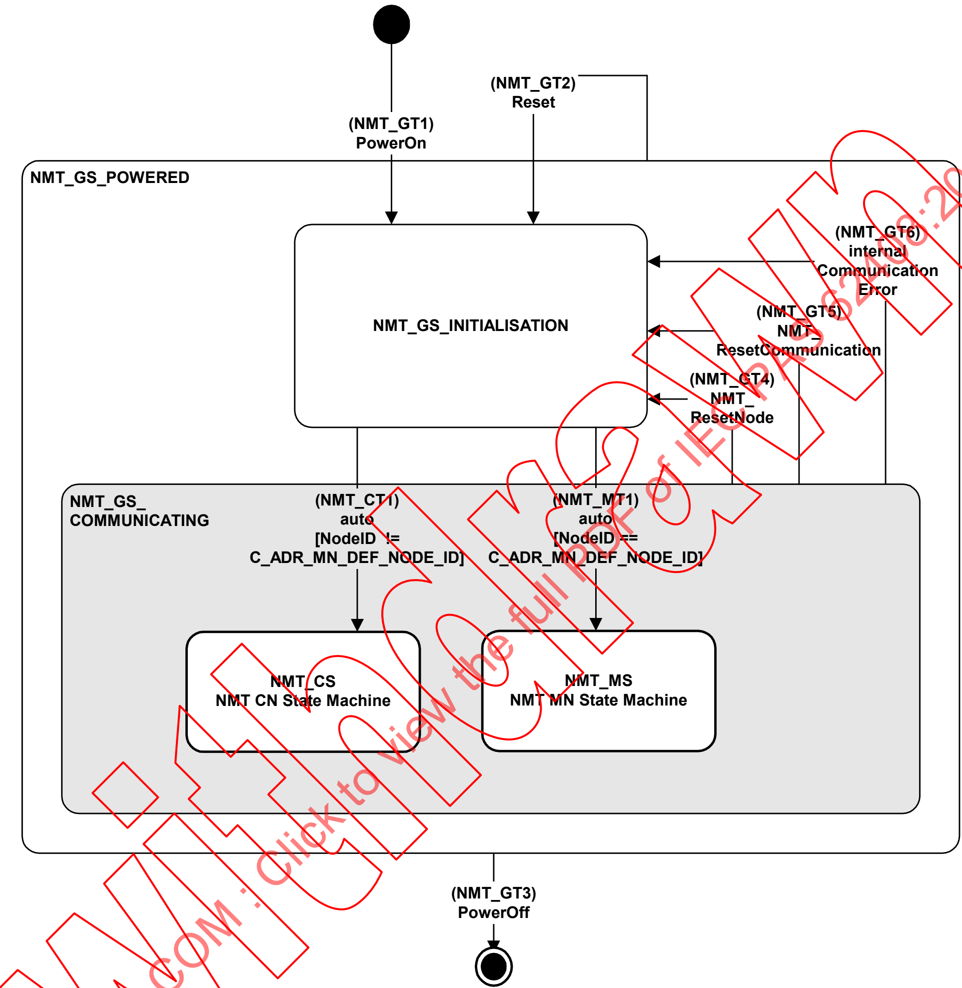


Figure 65 – Common Initialisation NMT State Machine

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) or after a hardware or software Reset (NMT_GT2). 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_INITIALISATION

After system start, the node attains the state NMT_GS_INITIALISATION. The node automatically shall enter this state, an NMT command shall not be necessary. In the state NMT_GS_INITIALISATION, the network functionality shall be initialised.

NMT_GS_INITIALISATION and its sub-states are node internal states only. They won't be signalled over the network by NMTStatus.

7.1.2.1.1.1.1 Sub-states

The state NMT_GS_INITIALISATION is divided into three sub-states in order to enable a complete or partial reset of a node.

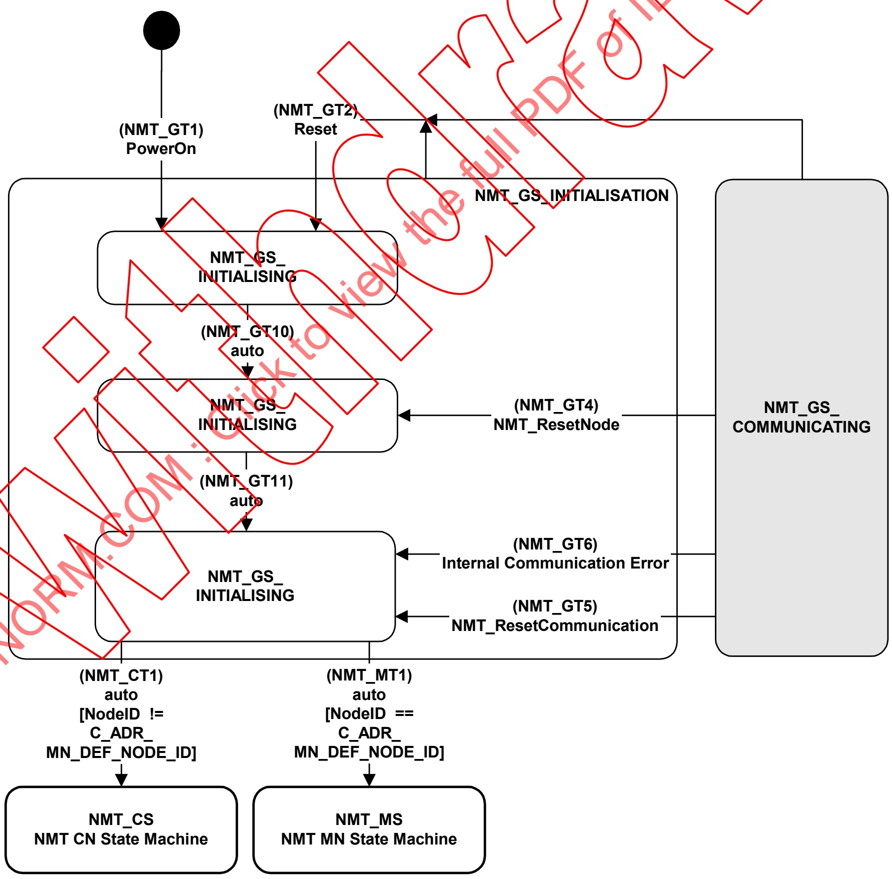


Figure 66 – Structure of the NMT_GS_INITIALISATION state

- NMT_GS_INITIALISING

  This is the first sub-state the EPL node shall enter after Power On (NMT_GT1) or hardware resp. software Reset (NMT_GT2). After finishing the basic node initialisation, the EPL 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 of the PowerOn values, the sub-state NMT_GS_RESET_COMMUNICATION shall be autonomously entered (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 MN resp. CN NMT State machine.

- NMT_GS_RESET_COMMUNICATION

  In this sub-state the parameters of the communication profile area shall be set to their PowerOn values.

  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 NMT MN state machine (NMT_MT1), otherwise the NMT CN state machine shall be entered (NMT_CT1).

  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 storing is not supported or has not been executed or if the Reset was preceded by a restore_default command (object NMT_RestoreDefParam_REC), the PowerOn values shall be the default values according to the communication and device profile specifications.

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 NMT MN state machine (refer 0) as well as the NMT CN state machine (refer 0).

There shall be a transition form NMT_GS_COMMUNICATING to NMT_GS_INITIALISATION if an NMTResetNode (NMT_GT4) or NMTResetCommunication (NMT_GT5) 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

Table 80 – Common Initialisation NMT State Transitions

(NMT_GT1)	PowerON [ ] / start basic node initialisation
	On PowerON, NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING shall be entered autonomously.
(NMT_GT2)	Reset [ ] / start basic node initialisation
	After Hardware or an CN local software Reset, NMT_GS_INITIALISATION sub-state NMT_GS_INITIALISING shall be entered autonomous.
(NMT_GT3)	PowerOFF [ ] /
	EPL node was powered off in NMT_GS_POWERED
(NMT_GT4)	NMTResetNote [ ] / start application initialisation
	If an NMTResetNode command is received in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_APPLICATION shall be entered.
(NMT_GT5)	NMTResetCommunication [ ] / start communication initialisation
	If an NMTResetCommunication 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
	If an Internal Communication Error is recognized in NMT_GS_COMMUNICATING, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION shall be entered.

(NMT_GT10)	Auto [basic node initialisation completed] / start application initialisation
	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
	NMT_GS_INITIALISATION sub-state NMT_GS_RESET_APPLICATION completed, NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION shall be entered autonomously.
(NMT_MT1)	Auto [communication initialisation completed, Node ID == C_ADR_MN_DEF_NODE_ID] / start observing network traffic
	If NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION 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 NMT MN state machine shall be entered.
(NMT_CT1)	Auto [communication initialisation completed, Node ID ≠ C_ADR_MN_DEF_NODE_ID] / start observing network traffic
	If NMT_GS_INITIALISATION sub-state NMT_GS_RESET_COMMUNICATION 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 NMT CN state machine shall be entered.

7.1.3 MN NMT State Machine

7.1.3.1 Overview

In Figure 68 the NMT state diagram of an MN is shown.

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.

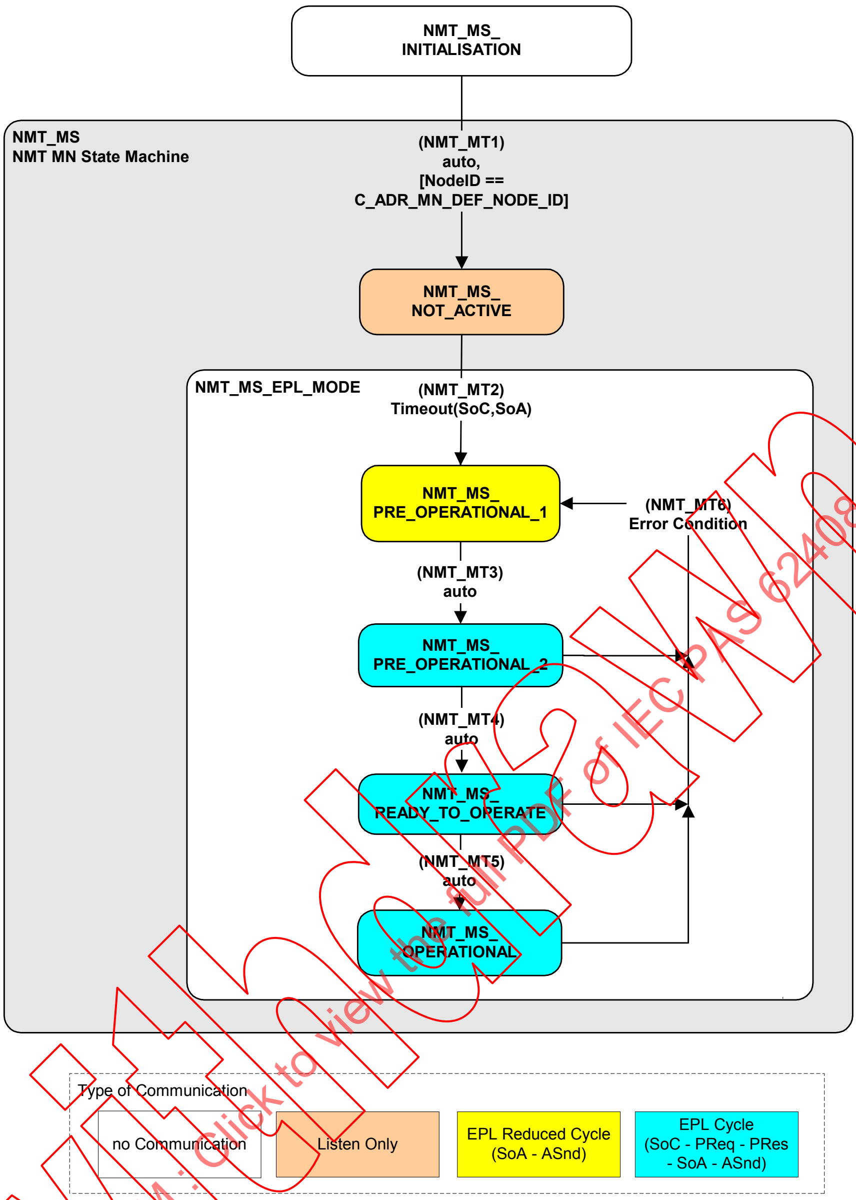


Figure 67 – NMT State Diagram of an MN

7.1.3.2 States

The current state of the MN shall define the current state of the EPL network .

7.1.3.2.1 NMT_MS_NOT_ACTIVE

In NMT_MS_NOT_ACTIVE, the MN shall observe network traffic 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 working on the network. 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.

In normal operation (no error), the transition from NMT_MS_NOT_ACTIVE to NMT_MS_PRE_OPERATIONAL_1 (NMT_MT2) 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 doesn’t support MN mode shall stay in state NMT_MS_NOT_ACTIVE. An error message shall be issued to the application.

7.1.3.2.2 NMT_MS_EPL_MODE

NMT_MS_EPL_MODE is a super-state that won't be 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 EPL cycle.

It shall identify the configured CNs (index NMT_CNAssignment_AU32[Node ID].Bit1) by IdentRequest / IdentResponse frame exchange and may check their identification. Identification check completion may be delayed if application SW or configuration data have to be downloaded by the CN.

Identified nodes CNs shall be cyclically accessed via IdentRequest.

In case of timeout conditions or wrong identification, an error shall be signalled to the MN application. The current MN state shall be maintained.

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 it's configuration in NMT_MS_PRE_OPERATIONAL_1.

Refer 7.6.2.3 for further information about NMT_MS_PRE_OPERATIONAL_1.

7.1.3.2.2.1.1 NMT_MS_PRE_OPERATIONAL_2

In the state NMT_MS_PRE_OPERATIONAL_2, the MN shall start executing the isochronous EPL cycle.

It shall access the identified CNs, that are in state NMT_CS_READY_TO_OPERATE and that are not marked as AsyncOnly CNs, by PReq frames in order to start PDO transfer, synchronisation and heartbeat. The transmitted PReq frames shall in accordance 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_CNAssignment_AU32[Node ID].Bit8) shall be cyclically accessed via IdentRequest.

Configured but unidentified CNs shall be searched for via SoA IdentRequest frames.

CNs that haven't completed configuration process in NMT_MS_PRE_OPERATIONAL_1, shall be allowed to continue configuration and shall be re-identified to test configuration compatibility, if they have set the StatusResponse CF flag.

In case of timeout conditions or wrong CN states, an error shall be signalled to the MN application. The current MN state shall be maintained.

The MN shall enable the state transition to NMT_MS_READY_TO_OPERATE of all mandatory CNs via the NMT command service NMTEnableReadyToOperate.

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 7.6.2.4 for further information about NMT_MS_PRE_OPERATIONAL_2.

7.1.3.2.2.1.2 NMT_MS_READY_TO_OPERATE

In NMT_MS_READY_TO_OPERATE, the MN shall execute the isochronous EPL 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_MNPReqPayloadList_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 for via SoA IdentRequest frames.

CNs that haven't completed configuration process in NMT_MS_PRE_OPERATIONAL_2, shall be allowed to continue configuration and shall be re-identified to test configuration compatibility, if they have set the StatusResponse CF flag.

In case of timeout conditions or wrong PRes frames, an error shall be signalled to the MN application. The current MN state shall be maintained.

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 lengthand timing.

Refer 7.6.2.5 for further information about NMT_MS_READY_TO_OPERATE.

7.1.3.2.2.1.3 NMT_MS_OPERATIONAL

NMT_MS_OPERATIONAL is the normal operating state of the EPL MN. The MN shall execute the isochronous EPL 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 details 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.

CNs that haven't completed configuration process in NMT_MS_PRE_OPERATIONAL_2, shall be allowed to continue configuration and shall be re-identified to test configuration comp ability, if they have set the StatusResponse CF flag.

All mandatory CNs shall be in NMT_CS_OPERATIONAL and free of error. If a mandatory CN is lost, if it has a state ≠ 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 7.6.2.6 for further information about NMT_MS_OPERATIONAL.

7.1.3.3 Transitions

Table 81 – MN Specific State Transitions

(NMT_MT1)	Refer Table 80
(NMT_MT2)	Timeout (SoC, SoA) [ ] / enable EPL reduced cycle communication
	If the node doesn't receive any SoA or SoC frame during a definable timeout period 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
(NMT_MT3)	Auto [All mandatory CNs identified] / enable isochronous EPL cycle communication, configured PReq, invalid
	If the node has identified all mandatory CNs, the node shall change to NMT_MS_PRE_OPERATIONAL_2. The state transition may be delayed by the application.
(NMT_MT4)	Auto [MN configuration completed, all mandatory CNs in NMT_CS_READY_TO_OPERATE] / configured PReq, not valid
	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. The state transition may be delayed by the application.
(NMT_MT5)	Auto [The isochronous communication is error free] / enable configured PRes, valid, start operation
	If all mandatory CNs transmit their PRes frames with correct frame length and timingndthe MN shall change to the state NMT_MS_OPERATIONAL. The state transition may be delayed by the application
(NMT_MT6)	Auto [Mandatory CN lost, mandatory CN not in NMT_CS_OPERATIONAL, error] / enable EPL reduced cycle communication
	If the MN detects errors of a mandatory CN, it shall change over to NMT_MS_PRE_OPERATIONAL_1

Refer Figure 65 and Table 80 for state transitions defined by the common Initialisation NMT state, which have to be applied to the MN NMT state machine.

7.1.4 CN NMT State Machine

In Figure 68 the NMT state diagram of a CN is shown.

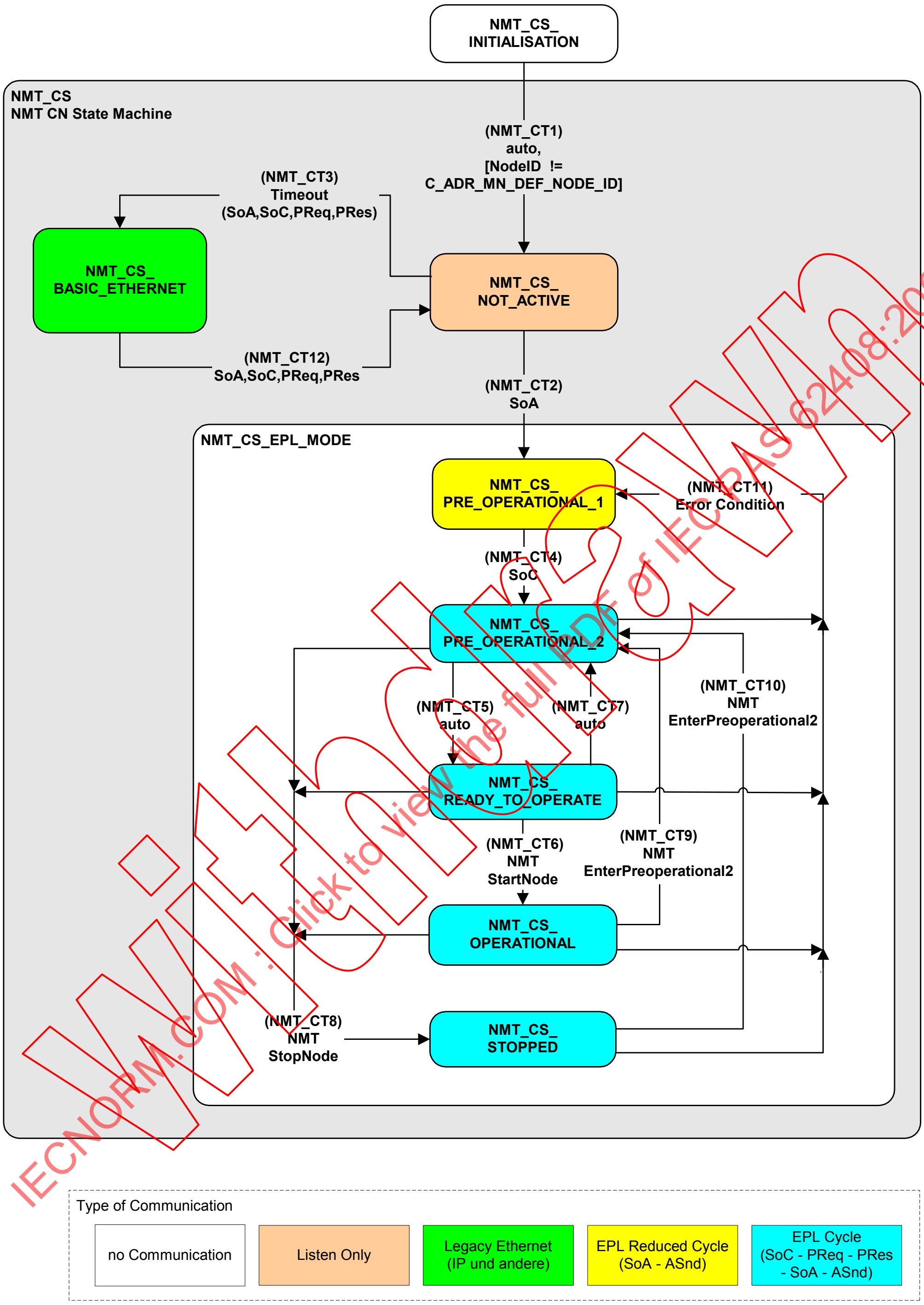


Figure 68 – State Diagram of an CN

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.

The node shall attach particular importance to the fact that a CN immediately shall recognize upcoming MN frames, e.g. the network state (refer 7.1.3.2) changing from the Basic Ethernet to the EPL Mode, and that the CN relates to it, so that disturbances in deterministic communication in the EPL Mode are prevented.

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 the NMT_CS_NOT_ACTIVE state. The node shall be able to recognize the NMT commands NMTResetNode and NMTResetCommunication sent via ASnd.

The transition from NMT_CS_NOT_ACTIVE to the NMT_CS_PRE_OPERATIONAL_1 state shall be triggered by a SoA frame being received.

The transition from NMT_CS_NOT_ACTIVE to the NMT_CS_BASIC_ETHERNET state shall be triggered by timeout for SoC, PReq, PRes and SoA frames.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.

7.1.4.1.1.1 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 AsyncInvite command.

In NMT_CS_PRE_OPERATIONAL_1 the node shall be identified by the MN via IdentRequest (see 7.4.3.2) and the CN shall download its configuration data from a configuration server, if required. Both processes may be completely or partionally 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.

The transition from NMT_CS_PRE_OPERATIONAL_1 to the following state shall be triggered by a SoC frame being received (see Figure 68).

There is no PDO communication in NMT_CS_PRE_OPERATIONAL_1.

7.1.4.1.1.2 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 should not be queried by the MN via PReq, but in case of an erroneous query it shall respond by a dummy PRes (with zero bytes payload data, the lower layer shall be responsible for padding. The PRes payload data shall be declared invalid by resetting 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 AsyncInvite 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.

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 transition from NMT_CS_PRE_OPERATIONAL_2 to NMT_CS_READY_TO_OPERATE shall be triggered by the completion of configuration and synchronisation and the reception of NMT state command NMTEnableReadyToOperate (see 7.4.1.2.1)

The transition from NMT_CS_PRE_OPERATIONAL_2 to NMT_CS_PRE_OPERATIONAL_1 shall be triggered by an error recognition (see 4.7).

The transition from NMT_CS_PRE_OPERATIONAL_2 to NMT_CS_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.4.1.2.1).

7.1.4.1.1.3 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. NMT_CS_READY_TO_OPERATE is the state, where the CN shall be initially queried via PReq by the MN and thus starts the isochronous frame exchange.

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 AsyncInvite 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 PDO data received from the MN via PReq and from other CNs and the MN via PRes may be interpreted if selected by the CN application.

The PDO data sent via PRes shall be declared invalid by resetting the RD flag by the CN application. The length of the PRes frame shall be equal to configured size (Object NMT_CycleTiming_REC.PresActPayload_U16). The transmitted data shall correspond to the requirements defined by the PDO mapping (see 6.4.7.4).

The transition from NMT_CS_READY_TO_OPERATE to NMT_CS_OPERATIONAL shall be triggered by the reception of NMT state command NMTStartNode (see 7.4.1.2.1)

If the CN recognizes its configuration state being changed within NMT_CS_READY_TO_OPERATE, it automatically shall change back to NMT_CS_PRE_OPERATIONAL_2.

The transition from NMT_CS_READY_TO_OPERATE to NMT_CS_PRE_OPERATIONAL_1 shall be triggered by an error recognition(see 4.7).

The transition from NMT_CS_READY_TO_OPERATE to NMT_CS_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.4.1.2.1).

7.1.4.1.1.4 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.

A acyclic-only CN isn't queried by the MN via PReq and thus doesn't respond via PRes.

Both types of CN shall respond to AsyncInvite commands via SoA. If not invited by the MN, there is no standard Ethernet frame transmission allowed at the NMT_CS_OPERATIONAL state.

The CN may perform surveillance of other nodes using the NMT guarding mechanism (7.4.5).

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 PDO data sent via PRes may be declared valid by setting the RD flag by the CN application. Temporary clearing the RD flag may be allowed if PDO data are not valid. The length of the PRes frame shall be equal to configured size (Object NMT_CycleTiming_REC.PresActPayload_U16). The transmitted data shall correspond to the requirements defined by the PDO mapping (see 6.4.7.4).

The transition from NMT_CS_OPERATIONAL to NMT_CS_PRE_OPERATIONAL_2 shall be triggered by the reception of NMT state command NMTEnterPreoperational2 (see 7.4.1.2.1).

The transition from NMT_CS_OPERATIONAL to NMT_CS_PRE_OPERATIONAL_1 shall be triggered by an error recognition(see 4.7).

The transition from NMT_CS_OPERATIONAL to NMT_CS_STOPPED shall be triggered by reception of NMT state command NMTStopNode (see 7.4.1.2.1).

7.1.4.1.1.5 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 arriving.

It shall not be queried by the MN via PRes.

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 standard 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 7.4.1.2.1)

The transition from NMT_CS_STOPPED to NMT_CS_PRE_OPERATIONAL_1 shall be triggered by an error recognition (see 4.7).

7.1.4.1.2 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 EPL specific network traffic control. The CSMA/CD collision handling shall control the network access. The node is allowed to transmit autonomiously.

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 EPL network traffic, upcoming when the node is in the NMT_CS_BASIC_ETHERNET state, 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_NOT_ACTIVE.

7.1.4.2 Transitions

Table 82 – CN Specific State Transitions

(NMT_CT1)	Refer Table 80
(NMT_CT2)	SoA [ ] / enable EPL reduced cycle communication
	If SoA 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
	If the node doesn't receive any SoC, PReq, PRes or SoA frame during a definable timeout period after entering the NMT_CS_NOT_ACTIVE state, the node shall change over to NMT_CS_BASIC_ETHERNET. The timeout period shall be defined by Object NMT_CnStateMachineTimeouts_AU32 sub-index BasicEthernetTimeout_U32.
(NMT_CT4)	SoC [ ] / enable EPL cycle communication, dummy PRes only
	If the node receives an SoC frame in NMT_CS_PRE_OPERATIONAL_1, the node shall change over to NMT_CS_PRE_OPERATIONAL_2.
(NMT_CT5)	Auto [NMTEnableReadyToOperate, completion of configuration and synchronisation] / enable configured PRes, not valid
	The CN shall automatically change over to NMT_CS_READY_TO_OPERATE
(NMT_CT6)	NMTStartNode [configuration valid] / enable configured PRes, valid, start operation
	If the CN receives the NMTStartNode command in NMT_CS_READY_TO_OPERATE, it shall change over to NMT_CS_OPERATIONAL
(NMT_CT7)	Auto [configuration lost] / disable valid PRes, dummy PRes only, reset CF flag
	If the CN recognizes its configuration state being changed in NMT_CS_READY_TO_OPERATE, it automatically shall change back to NMT_CS_PRE_OPERATIONAL_2.
(NMT_CT8)	NMTStopNode [ ] / freeze cyclic communication
	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)	NMTEnterPreoperational2 [ ] / disable valid PRes, dummy PRes only
	If the node receives the NMTEnterPreoperational2 command in NMT_CS_OPERATIONAL, it shall change over to NMT_CS_PRE_OPERATIONAL_2
(NMT_CT10)	NMTEnterPreoperational2 [ ] / re-enable EPL cycle communication, dummy PRes only
	If the node receives the NMTEnterPreoperational2 command in NMT_CS_STOPPED, it shall change over to NMT_CS_PRE_OPERATIONAL_2.
(NMT_CT11)	Error condition [ ] / enable EPL reduced cycle communication
	If the node recognizes an error condition (refer 4.7) 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, start observing network traffic
	If an SoC, PReq, PRes or SoA frame is received in NMT_CS_BASIC_ETHERNET, the node shall change over to NMT_CS_NOT_ACTIVE.

Refer Figure 65 and Table 80 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

Table 83 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.

Table 83 – States and Communication Objects

	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
EPL controlled network traffic								
SoC	-	-	R/S	R	R	R	-	R/S
PReq	-	-	-	R	R	R	-	R/S
PDO reception	-	-	-	-	$(x)^1$	x	-	-
PRes receive	-	-	-	-	R	R	-	R/S
PRes transmit	-	-	-	(T)	T	T	-	-
PDO transmission	-	-	-	-	$(x)^2$	x	-	-
SoA	-	R/S	R	R	R	R	R	R/S
IdentRequest	-	x	x	x	x	x	x	-
StatusRequest	-	-	x	x	x	x	x	-
NMTRequestInvite	-	-	x	x	x	x	-	-
UnspecifiedInvite	-	-	x	x	x	x	-	-
ASnd reception	-	R	R	R	R	R	R	R
UDP/IP reception	-	-	x	x	x	x	-	-
SDO reception	-	-	x	x	x	x	-	-
NMT Command	-	$(x)^3$	x^4	x^4	x^4	x^4	x^4	$(x)^3$
ASnd transmission, assigned by SoA	-	-	T	T	T	T	T	-
UDP/IP transmission	-	-	x	x	x	x	-	-
SDO transmission	-	-	x	x	x	x	-	-
NMTRequest transmission	-	-	x	x	x	x	-	-
IdentResponse	-	x	x	x	x	x	x	-
StatusResponse	-	-	x	x	x	x	x	-
Network traffic not controlled by EPL								
Legacy Ethernet reception	-	-	-	-	-	-	-	R
UDP/IP reception	-	-	-	-	-	-	-	x
SDO reception	-	-	-	-	-	-	-	x
Legacy Ethernet transmission	-	-	-	-	-	-	-	T
UDP/IP, autonomiously sent	-	-	-	-	-	-	-	x
SDO transmission	-	-	-	-	-	-	-	x

R *frame accepted*
R/S *frame accepted, triggers state transition*
T *frame transmitted*
(T) *dummy PRes only*
x *frame data interpreted resp. transmitted*
$(x)^1$ *frame data may be interpreted*
$(x)^2$ *data invalidated by resetting the RD flag*
$(x)^3$ *only selected NMT commands accepted, shall cause state transition, refer 7.4.1.2.1, reception requires previous loss of SoA*
x^4 *may cause state transition, refer 7.4.1.2.1*
- *no frame handling*

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 to 4.2.4.3 and 4.2.4.4).

7.2 NMT CN Objects

7.2.1 Object 1000h: NMT_DeviceType_U32

Contains information about the device type. The object at index 1000_h describes the type of device and its functionality.

INDEX	1000_h
Name	NMT_DeviceType_U32
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

- **Entry Description**

Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **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_h indicates a device that does not follow a standardised device profile. For multiple device modules the Additional Information parameter contains $FFFF_h$ and the device profile number referenced by object 1000_h 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 $67FF_h + x * 800_h$ with x = internal number of the device (0 – 7). These entries describe the device type of the preceding device.

Byte:	MSB	LSB
	Additional Information	Device Profile Number

7.2.2 Object 1006_h: NMTCycleTime_U32

This object defines the communication cycle period in µs. This period defines the SYNC interval.

INDEX	1006_h
Name	NMTCycleTime_U32
Object Code	VAR
Data Type	UNSIGNED32
Category	Mandatory

- **Entry Description**

Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

7.2.3 Object 1008$_h$: NMT_ManufactDevName_VS

Contains the manufacturer device name.

INDEX	1008$_h$
Name	NMT_ManufactDevName_VS
Object Code	VAR
Data Type	Visible String
Category	Optional

- **Entry Description**

Access	const
PDO Mapping	No
Value Range	No
Default Value	No

7.2.4 Object 1009$_h$: NMT_ManufactHwVers_VS

Contains the manufacturer hardware version description.

INDEX	1009$_h$
Name	manufacturer hardware version
Object Code	VAR
Data Type	Visible String
Category	Optional

- **Entry Description**

Access	const
PDO Mapping	No
Value Range	No
Default Value	No

7.2.5 Object 100Ah: NMT_ManufactSwVers_VS

Contains the manufacturer software version description.

INDEX	100A$_h$
Name	NMT_ManufactSwVers_VS
Object Code	VAR
Data Type	Visible String
Category	Optional

- **Entry Description**

Access	const
PDO Mapping	No
Value Range	No
Default Value	No

7.2.6 Object 1010h: 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_h
Name	NMT_StoreParam_REC
Object Code	RECORD
Category	Optional

- **Sub-Index 0_h: NumberOfEntries**

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1_h – $7F_h$
Default Value	No

- **Sub-Index 1_h: SaveAllParam_U32**

Refers to all parameters that can be stored on the device.

Sub-Index	1_h
Description	SaveAllParam_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2_h: SaveCommParam_U32**

Refers to communication related parameters (Index 1000_h - $1FFF_h$ manufacturer specific communication parameters).

Sub-Index	2_h
Description	SaveCommParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3_h: SaveApplParam_U32**

Refers to application related parameters (Index 6000_h - $9FFF_h$ manufacturer specific application parameters).

Sub-Index	3_h
Description	SaveApplParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 4_h – $7F_h$: SaveManufParam_U32**

  May store their choice of parameters individually.

Sub-Index	4_h - $7F_h$
Description	SaveManufParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Value Interpretation of Sub-Index 1_h – $7F_h$**

  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“.

Table 84 – NMT_StoreParam_REC Storage write access signature

Signature	MSB			LSB
ISO 8859 (“ASCII”)	e	v	a	s
hex	65_h	76_h	61_h	73_h

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 failed, 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:

Table 85 – NMT_StoreParam_REC Storage read access structure

	UNSIGNED32		
	MSB	LSB	
bits	31-2	1	0
	reserved (=0)	0/1	0/1

Table 86 – NMT_StoreParam_REC Structure of read access

bit	value	meaning
31-2	0	reserved (=0)
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

Autonomous saving means that a device stores the storable parameters in a non-volatile manner without user request.

7.2.7 Object 1011_h: 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_h
Name	NMT_RestoreDefParam_REC
Object Code	RECORD
Category	Optional

- **Sub-Index 0_h: NumberOfEntries**

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1h – 7Fh
Default Value	No

- **Sub-Index 1_h: RestoreAllParam_U32**

restore all default parameters

Sub-Index	1_h
Description	RestoreAllParam_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3_h: RestoreCommParam_U32**

Restore communication default parameters, refers to communication related parameters (Index 1000_h - $1FFF_h$ manufacturer specific communication parameters).

Sub-Index	2_h
Description	RestoreCommParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3_h: RestoreApplParam_U32**

Restore application default parameters, refers to application related parameters (Index 6000_h - $9FFF_h$ manufacturer specific application parameters).

Sub-Index	3_h
Description	RestoreApplParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 4_h - $7F_h$: RestoreManufParam_U32**

Restore manufacturer defined default parameters. manufacturers may restore their individual choice of parameters.

Sub-Index	4_h - $7F_h$
Description	RestoreManufParam_U32
Type	UNSIGNED32
Entry Category	Optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 1_h – $7F_h$ 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“.

Table 87 – NMT_RestoreDefParam_REC Restoring write access signature

Signature	MSB			LSB
ASCII	d	a	o	l
hex	64h	61h	6Fh	6Ch

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.

The default values are set valid after the device is reset (reset node for sub-index 1_h – $7F_h$, reset communication for sub-index 2_h) or power cycled.

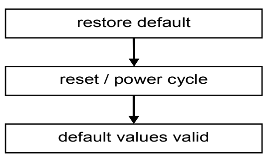


Figure 69 – NMT_RestoreDefParam_REC restore procedure

On read access to the appropriate sub-index the device provides information about its default parameter restoring capability with the following format:

Table 88 – NMT_RestoreDefParam_REC Restoring default values read access structure

	UNSIGNED32	
	MSB	LSB
bits	31-1	0
	reserved (=0)	0/1

Table 89 – NMT_RestoreDefParam_REC Structure of restore read access

bit number	value	meaning
31-1	0	reserved (=0)
0	0	Device does not restore default parameters
	1	Device restores parameters

7.2.8 Object 1016h: NMT_ConsumerHeartbeatTime_AU32

The consumer heartbeat time defines the expected heartbeat cycle time and thus has to be higher than the corresponding producer heartbeat time configured on the device producing this heartbeat. Monitoring starts after the reception of the first heartbeat. If the consumer heartbeat time is 0 the corresponding entry is not used. The time has to be a multiple of 1ms.

INDEX	1016_h
Name	NMT_ConsumerHeartbeatTime_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	Optional

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	0_h
Description	NumberOfEntries_U8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	254
Default Value	254

- **Sub-Index 01_h – FE_h: ConsumerHeartbeatTime**

Sub-Index	01_h – FE_h
Description	ConsumerHeartbeatTime
Entry Category	optional
Access	rw
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	0

- **Sub-Index 01_h – FE_h Value Description**

	UNSIGNED32		
	MSB		LSB
Bits	31-24	23-16	15-0
Value	reserved (value: 00h)	Node-ID	heartbeat time
Encoded as	-	UNSIGNED8	UNSIGNED16

At an attempt to configure several consumer heartbeat times unequal 0 for the same Node-ID the device aborts the SDO download with abort code 0604 0043_h

7.2.9 Object 1018h: NMT_IdentityObject_REC

The object at index 1018h contains general information about the device.

INDEX	1018_h
Name	NMT_IdentityObject_REC
Object Code	RECORD
Data Type	Identity
Category	Mandatory

- **Sub-Index 0_h: NumberOfEntries**

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	1 - 4
Default Value	No

- **Sub-Index 1_h: VendorId_U32**

The Vendor ID contains a unique value allocated to each manufacturer.

Sub-Index	1_h
Description	VendorId_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 2_h: ProductCode_U32**

The manufacturer-specific Product code identifies a specific device version.

Sub-Index	2_h
Description	ProductCode_U32
Type	UNSIGNED32
Entry Category	Optional
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index 3_h: RevisionNo_U32**

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.

Table 90 – Structure of Revision number

31	16	15	0
major revision number		minor revision number	
MSB			LSB

Sub-Index	3_h
Description	RevisionNo_U32
Type	UNSIGNED32
Entry Category	Optional
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

- **Sub-Index$_h$: SerialNo_U32**

The manufacturer-specific Serial number identifies a specific device.

Sub-Index	4_h
Description	SerialNo_U32
Type	UNSIGNED32
Entry Category	Optional
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	No

7.2.10 Object 1030_h - 103Fh: NMT_ItfGroup_*Xh*_REC

The following objects are used to configure and retrieve parameters of the interfaces (physical or virtual) via SDO. Each interface shall have one entry. The ItfGroup_REC object is a subset of the Interface Group RFC1213.

To allow access by name "_*Xh*" shall be replaced by a name index. Name index shall be "_*0h*" if object index is 1030_h. It shall be incremented up to "_*Fh*" corresponding to object index $103F_h$.

Index	1030h - 103Fh
Name	NMT_ItfGroup_Xh_REC
Object Code	RECORD
Data Type	ItfGroup_REC_Type
Category	M

- **Sub-Index 00_h: NumberOfEntries_U8**

Sub-Index	00h
Description	NumberOfEntries_U8
Data Type	UNSIGNED8
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	7, 14, 21, …, 252
Default value	7

- **Sub-Index 01_h: ItfIndex*N*_U16**

Interface index of the router interface No. *N*. Index *N* shall be set to 1 on sub-index 01_h. *N* shall be increased by 1 on index 08_h, $0F_h$ etc.

Sub-Index	01h
Description	ItfIndexN_U16
Data Type	UNSIGNED16
Entry Category	M, Sub-Index 01h O, Sub-Index 08h, 0Fh …
Access	Rw
PDO Mapping	No
Value range	UNSIGNED16
Default value	-

- **Sub-Index 02_h: ItfDescription_VSTR**

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.

Sub-Index	02h
Description	ItfDescription_VSTR
Data Type	VISIBLE_STRINGXX
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	VISIBLE_STRINGXX
Default value	-

- **Sub-Index 03$_h$: ItfType_U8**

The type of interface, distinguished according to the physical/link protocol(s) immediately `below' the network layer in the protocol stack.

Sub-Index	03h
Description	ItfType_U8
Data Type	USINGED8
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	Other(1), -- none of the following ethernet-csmacd(6), iso88023-csmacd(7), Ethernet-Powerlink(xx) CAN(x)
Default value	Ethernet-csmacd(6)

- **Sub-Index 04$_h$: ItfMtu_U32**

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.

Sub-Index	04h
Description	ItfMtu_U32
Data Type	USINGED32
Entry Category	M
Access	Ro ???
PDO Mapping	No
Value range	USINGED32
Default value	-

- **Sub-Index 05$_h$: ItfPhysAddress_OSTR**

The interface's address at the protocol layer immediately `below' the network layer in the protocol stack. For interfaces which do not have such an address (e.g., a serial line), this object should contain an octet string of zero length.

Sub-Index	**05$_h$**
Description	ItfPhysAddress_OSTR
Data Type	OCTET_STRING
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	OCTET_STRING
Default value	-

- **Sub-Index 06$_h$: ItfName_VSTR**

A user reference name for the interface.

Sub-Index	**06$_h$**
Description	ItfName_VSTR
Data Type	VISIBLE_STRINGXX
Entry Category	M
Access	Rw
PDO Mapping	No
Value range	VISIBLE_STRINGXX
Default value	-

- **Sub-Index 07$_h$: ItfOperStatus_BOOL**

The current operational state of the interface.

Sub-Index	**07$_h$**
Description	ItfOperStatus_BOOL
Data Type	BOOL
Entry Category	M
Access	Ro
PDO Mapping	No
Value range	Down(0), up(1)
Default value	-

7.2.11 Object 1F98$_h$: NMT_CycleTiming_REC

NMT_CycleTiming_REC provides node specific timing parameters, that influence the EPL cycle timing. All entries shall be supported by a CN. On the MN, some of the entries are irrelevant.

INDEX	1F98$_h$
Name	NMT_CycleTiming_REC
Object Code	RECORD
Data Type	
Category	Mandatory

- **Sub-Index 0$_h$: NumberOfEntries**

Sub-Index	0$_h$
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	6
Default Value	6

- **Sub-Index 1ₕ: PReqMaxPayload_U16**

  Provides the device specific limit for PollRequest payload data size in octets received by the CN. The value is the upper limit to PReqActPayload_U16 (refer below)

  On the MN, the entry shall describe an upper limit to the PollRequest payload data size in octets to be transmitted to the CNs.

Sub-Index	1_h
Description	PReqMaxPayload_U16
Type	UNSIGNED16
Entry Category	CN. MN: Mandatory
Access	ro
PDO Mapping	No
Value Range	0 – C_DLL_MAX_PAYLOAD_PREQ
Default Value	0

- **Sub-Index 2_h: PResMaxPayload_U16**

  Provides the device specific limit for PollResponse payload data size in octets transmitted by the node. The value is the upper limit to PResActPayload_U16 (refer below)

Sub-Index	2_h
Description	PResMaxPayload_U16
Type	UNSIGNED16
Entry Category	CN, MN: Mandatory
Access	ro
PDO Mapping	No
Value Range	0 - C_DLL_MAX_PAYLOAD_PRES
Default Value	0

- **Sub-Index 3_h: PResMaxLatency_U32**

  Provides the time in [ns], that is required by the CN to respond to PReq.

Sub-Index	3_h
Description	PResMaxLatency_U32
Type	UNSIGNED32
Entry Category	CN: Mandatory MN: not supported
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	5 000

- **Sub-Index 4_h: PReqActPayload_U16**

  Provides the actual PollRequest payload data size in octets expected by the CN.

Sub-Index	4_h
Description	PReqActPayload_U16
Type	UNSIGNED16
Entry Category	CN: Mandatory MN: not supported
Access	rw
PDO Mapping	No
Value Range	0 - C_DLL_MAX_PAYLOAD_PREQ
Default Value	0

- **Sub-Index 5_h: PResActPayload_U16**

  Provides the actual PollResponse payload data size in octets transmitted by the node.

Sub-Index	5_h
Description	PResActPayload_U16
Type	UNSIGNED16
Entry Category	CN, MN: Mandatory
Access	rw
PDO Mapping	No
Value Range	0 - C_DLL_MAX_PAYLOAD_PRES
Default Value	0

7.2.12 Object $1F99_h$: NMT_CnStateMachineTimeouts_REC

NMT_CnStateMachineTimeouts_REC provides CN specific timeout values in [ns] required by the CN NMT state machine.

INDEX	$1F99_h$
Name	NMT_CnStateMachineTimeouts_REC
Object Code	RECORD
Data Type	
Category	Mandatory

- **Sub-Index 0_h: NumberOfEntries**

Sub-Index	0_h
Description	NumberOfEntries
Type	UNSIGNED8
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	2
Default Value	2

- **Sub-Index 1_h: BasicEthernetTimeout_U32**

  Provide the time in [µs] to be applied before changing from NMT_CS_NOT_ACTIVE to NMT_CS_BASIC_ETHERNET.

Sub-Index	1_h
Description	BasicEthernetTimeout_U32
Type	UNSIGNED32
Entry Category	Mandatory
Access	ro
PDO Mapping	No
Value Range	UNSIGNED32
Default Value	5 000 000

7.3 NMT MN Objects

The NMT Master provides services for controlling the network behaviour of nodes. Only one NMT Master can exist in an Ethernet Powerlink (EPL) Network. In EPL the NMT Master is located in the MN.MNCN

MNThe NMT Master Control Settings activate the MN functions and define the boot behaviour and the error reactions.

7.3.1 NMT Master Start Up Behaviour

7.3.1.1 Object 1F80$_h$: NMT_StartUp_U32

This object configures the boot behaviour of a device that is able to perform the NMT.

Object NMT_StartUp_U32 is a configuration object. Internal state transitions must not change this object.

Index	1F80h
Name	NMT_StartUp_U32
Object Code	VAR
Data Type	UNSIGNED32
Category	M

- **Value description**

Access	ro
PDO Mapping	No
Value range	Bit field, refer below
Default value	Bit field, refer below

- **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	Enter myself automatically to state NMT_MS_OPERATIONAL
		1_b	Do not enter myself automatically to 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 treat the CN individually.
		1_b	On error event from guarding a mandatory CN perform NMTResetNode with broadcast addressing. Refer to Bit 6 and 1F81$_h$, Bit 3
	5	---	Reserved (0_b)

	6	0_b	On error event from guarding a mandatory CN treat the CN according to Bit 4
		1_b	On error event from guarding a mandatory CN send NMTStopNode with broadcast addressing. Ignore Bit 4.
	7	0_b	Enter myself automatically to state NMT_MS_PRE_OPERATIONAL_2
		1_b	Do not enter myself automatically to state NMT_MS_PRE_OPERATIONAL_2. Application will decide, when to enter the state.
1	8	0_b	Enter myself automatically to state NMT_MS_ READY_TO_OPERATE
		1_b	Do not enter myself automatically to state 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_MNDeviceTypeIdList_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 - 15	---	Reserved (000_b)
2 - 3	16 - 31	---	Reserved ($00\ 00_h$)

7.3.1.2 Object 1F89$_h$: NMT_BootTime_REC

This object describes time interval values, to be used by the MN when it starts the network.

The maximum times are in µs, the master will wait for all mandatory CNs in before signalling an error. If the time is zero (0), it will wait endlessly.

Index	1F89h
Name	NMT_BootTime_REC
Object Code	RECORD
Data Type	NMT_BootTime_TYPE
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	4
Default value	4

- **Sub-Index 01$_h$: MNWaitNotAct_U32**

This sub-index shall describe the time interval in [µs], the MN shall remain in state NMT_MS_NOT_ACTIVE and listen for EPL frames on the network, before it changes over to NMT_MS_PRE_OPERATIONAL_1.

Sub-Index	**01h**
Name	MNWaitNotAct_U32
Data Type	UNSIGNED32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	>=250
Default value	1 000 000

- **Sub-Index 02h: MNTimeoutPreOp1_U32**

This sub-index shall describe the time interval in [µs], 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 will be no timeout for CN identification.

Sub-Index	**02h**
Name	MNTimeoutPreOp1_U32
Data Type	UNSIGNED32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 50 000 – 5 000 000
Default value	500 000

- **Sub-Index 03$_h$: MNWaitPreOp1_U32**

  This sub-index shall describe the time interval in [µs], the MN shall rest 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	**02h**
Name	MNWaitPreOp1_U32
Data Type	UNSIGNED32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 50 000 – 5 000 000
Default value	500 000

- **Sub-Index 04$_h$: MNTimeoutPreOp2_U32**

  This sub-index shall describe the time interval in [µs], the MN shall wait in state NMT_MS_PRE_OPERATIONAL_2 for all mandatory CNs to be in state NMT_CS_READY_TO_OPERATE, before it signals an error to the application.

  If the timeout value is zero (0), there will be no timeout, e.g. the MN will wait endlessly.

Sub-Index	**04h**
Name	MNTimeoutPreOp2_U32
Data Type	UNSIGNED32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 50 000 – 5 000 000
Default value	500 000

- **Sub-Index 05$_h$: MNTimeoutReadyToOp_U32**

  This sub-index shall describe the time interval in [µs], 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 will be no timeout, e.g. the MN will wait endlessly.

Sub-Index	**05h**
Name	MNTimeoutReadyToOp_U32
Data Type	UNSIGNED32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0, 50 000 – 5 000 000
Default value	500 000

7.3.2 NMT Master Network Node Lists

The Network List consists of some objects, that give information which CNs have to be managed, how they have to be booted and about requested actions on Error events.

7.3.2.1 Object 1F81$_h$: NMT_CNAssignment_AU32

This object assigns CNs to the NMT Master.

Each sub-index in the array corresponds to the CN with the Node ID equal to the sub-index. The sub-index equal the MN's Node ID is ignored.

Index	1F81h
Name	NMT_CNAssignment_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	00$_h$
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ - FE$_h$: CNAssignment_U32**

Sub-Index	01$_h$ - FE$_h$
Name	CNAssignment_U32
Entry Category	M
Access	rw
PDO Mapping	No
Value range	Bitfield, refer below
Default value	Bitfield, refer below

- **CNAssignment_U32 Value Interpretation**

Octet	Bit	Value	Description
0	0	0_b	Node with this ID is not existing, Bits 1ff invalid
		1_b	Node with this ID is existing
	1	0_b	Node with this ID is not an CN, Bits 2ff invalid
		1_b	Node with this ID is an CN. After configuration (with Configuration Manager) the Node will be set to state NMT_CS_OPERATIONAL
	2	0_b	On detection of a booting CN inform the application but do NOT automatically configure and start the node.
		1_b	On detection of a booting CN inform the application and do continue the process “Start Boot CN”.
	3	0_b	Optional CN
		1_b	Mandatory CN
	4	0_b	The CN node may be reset with the NMTResetCommunication command independent of its state. Hence, no checking of its state has to be executed prior to NMT Reset Communication.
		1_b	MN must not send NMTResetCommunication for this node if it notices the CN to be in NMT_CS_OPERATIONAL state..
	5	0_b	Application software version verification for this node is not required
		1_b	Application software version verification for this node is required
	6	0_b	Automatic application software update (download) is not allowed
		1_b	Automatic application software update (download) is allowed
	7	0_b	Continuous CN
		1_b	Multiplexed CN
1	8	0_b	Isochronously accessed CN
		1_b	AsyncOnly CN
	9 - 15	---	Reserved ($0000\ 000_b$)
2 - 3	16 - 31	---	Reserved ($00\ 00_h$)

Bits that control a feature that is not supported by the implementation are ro.

7.3.2.2 Object 1F84$_h$: NMT_MNDeviceTypeIdList_AU32

This object holds a list of the expected NMT_DeviceTypeId_U32 value for each configured CN.

Index	1F84h
Name	NMT_MNDeviceTypeIdList_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	00$_h$
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ - FE$_h$: CNDeviceTypeId_U32**

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 CN is assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_DeviceTypeId_U32 value is reported to the MN via IdentResponse. The MN shall compare the received value to the respective CNDeviceTypeId_U32 sub-index value. The Boot-Up for that device is only continued on exact equality.

If the value in CNDeviceTypeId_U32 is 0, this read access only gives information about the principle existence of a device with this Node ID. There shall be no comparison to the reported NMT_DeviceTypeId_U32 value.

For multi-device-modules the application may perform additional checks.

Sub-Index	**01h - FEh**
Name	CNDeviceTypeId_U32
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

7.3.2.3 Object 1F85$_h$: NMT_MNVendorIdList_AU32

This object holds a list of the expected NMT_IdentityObject_REC.VendorId_U32 value for each configured CN.

Index	1F85h
Name	NMT_MNVendorIdList_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	00$_h$
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ - FE$_h$: CNVendorId_U32**

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 CN is assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_IdentityObject_REC.VendorId_U32 value is reported to the MN via IdentResponse. The MN shall compare the received value to the respective CNVendorId_U32 sub-index value. The Boot-Up for that device is only continued on exact equality.

If the value in CNVendorId_U32 is 0, this read access only gives information about the principle existence of a device with this Node ID. There shall be no comparison to the reported NMT_IdentityObject_REC.VendorId_U32 value.

For multi-device-modules the application may perform additional checks.

Sub-Index	**01h - FEh**
Name	CNVendorId_U32
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

7.3.2.4 Object $1F86_h$: NMT_MNProductCodeList_AU32

This object holds a list of the expected NMT_IdentityObject_REC.ProductCode_U32 value for each configured CN.

Index	1F86h
Name	NMT_MNProductCodeList_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00_h: NumberOfEntries**

Sub-Index	00_h
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01_h - FE_h: CNProductCode_U32**

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 CN is assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_IdentityObject_REC.ProductCode_U32 value is reported to the MN via IdentResponse. The MN shall compare the received value to the respective CNProductCode_U32 sub-index value. The Boot-Up for that device is only continued on exact equality.

If the value in CNProductCode_U32 is 0, this read access only gives information about the principle existence of a device with this Node ID. There shall be no comparison to the reported NMT_IdentityObject_REC.ProductCode_U32 value.

For multi-device-modules the application may perform additional checks.

Sub-Index	**01h - FEh**
Name	CNProductCode_U32
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

7.3.2.5 Object 1F87$_h$: NMT_MNRevisionNoList_AU32

This object holds a list of the expected NMT_IdentityObject_REC.RevisionNo_U32 value for each configured CN.

Index	1F87h
Name	NMT_MNRevisionNoList_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	00$_h$
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ - FE$_h$: CNRevisionNo_U32**

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 CN is assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_IdentityObject_REC.RevisionNo_U32 is reported to the MN via IdentResponse. The MN shall compare the received value to the respective CNRevisionNo_U32 sub-index value. The Boot-Up for that device is only continued on exact equality.

If the value in CNRevisionNo_U32 is 0, this read access only gives information about the principle existence of a device with this Node ID. There shall be no comparison to the reported NMT_IdentityObject_REC.RevisionNo_U32 value.

For multi-device-modules the application may perform additional checks.

Sub-Index	**01h - FEh**
Name	CNRevisionNo_U32
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

7.3.2.6 Object 1F88$_h$: NMT_MNSerialNoList_AU32

This object holds a list of the expected NMT_IdentityObject_REC.RevisionNo_U32 value for each configured CN.

Index	1F88h
Name	NMT_MNSerialNoList_AU32
Object Code	ARRAY
Data Type	UNSIGNED32
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	00$_h$
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ - FE$_h$: CNSerialNo_U32**

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 CN is assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_IdentityObject_REC.SerialNo_U32 is reported to the MN via IdentResponse. The MN shall compare the received value to the respective CNSerialNo_U32 sub-index value. The Boot-Up for that device is only continued on exact equality.

If the value in CNSerialNo_U32 is 0, this read access only gives information about the principle existence of a device with this Node ID. There shall be no comparison to the reported NMT_IdentityObject_REC.SerialNo_U32 value.

For multi-device-modules the application may perform additional checks.

Sub-Index	**01h - FEh**
Name	CNSerialNo_U32
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED32
Default value	0

7.3.3 Network Timing

The indices described by this paragraph control the timing behaviour of the EPL network traffic. Values given by the MN implementation are provided by index NMT_MNCycleTiming_REC. Values that are specific to each individual CN are stored at NMT_MNPReqPayload_AU16, NMT_MNCNResTime_AU16 and NMT_MNPResPayload_AU16.

7.3.3.1 Object 1F8A$_h$: NMT_MNCycleTiming_REC

This object shall describe time interval values, to be used by the MN when it controls the network traffic.

Index	1F8Ah
Name	NMT_MNCycleTiming_REC
Object Code	RECORD
Data Type	NMT_ MNCycleTiming _TYPE
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	2
Default value	2

- **Sub-Index 01$_h$: MNMultiplSlotCnt_U8**

This sub-index shall describe the number of multiplexed slots in the isochronous EPL cycle.

Sub-Index	**01h**
Name	MNMultiplSlotCnt_U8
Data Type	UNSIGNED8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0 - 10
Default value	0

- **Sub-Index 02$_h$: MNAsyncMTUSize_U16**

This sub-index shall describe the maximum asynchoronous frame size in [octets]. The value shall be applyable by ASnd frames as well as by UDP/IP and other legacy Ethernet type frames. That's why, the value shall describe the length of the complete Ethernet frame minus 14 octets Ethernet header and 4 octets checksum.

This entry shall be valid in NMT states NMT_MS_PRE_OPERATIONAL_2, NMT_MS_READY_TO_OPERATE and NMT_MS_OPERATIONAL. In NMT_MS_PRE_OPERATIONAL_1 a fixed AsyncMTUSize of 1500 shall be applied.

Sub-Index	**02h**
Name	MNAsyncMTUSize_U16
Data Type	UNSIGNED16
Entry Category	M
Access	rw
PDO Mapping	No
Value range	288 – 1 500
Default value	288

7.3.3.2 Object $1F8B_h$: NMT_MNPReqPayloadList_AU16

This object holds a list of the PReq Payload data size in [octets] for each configured CN that has to be accessed isochronously, e.g. via PReq / PRes frame exchange.

Index	1F8Bh
Name	NMT_MNPReqPayloadList_AU16
Object Code	ARRAY
Data Type	UNSIGNED16
Category	M

- **Sub-Index 00_h: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01_h – FE_h: CNPReqPayload_U16**

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_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1, 2 and 8.

On Boot-Up, the CN's NMT_CycleTiming_REC.PReqPayload_U16 value is reported to the MN via IdentResponse. The MN shall store the value to the sub-index equal to the CN's Node ID.

Each value shall be updated, when the respective CN signals, that it is in state NMT_CS_READY_TO_OPERATE.

Sub-Index	01_h – FE_h
Name	CNPReqPayload_U16
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0 – C_DLL_MAX_PAYLOAD_PREQ
Default value	0

7.3.3.3 Object 1F8Ch: NMT_MNCNRespTimeList_AU16

This object holds a list of maximum frame response times for each configured CN in [ns].

Index	1F8Ch
Name	NMT_MNCNRespTimeList_AU16
Object Code	ARRAY
Data Type	UNSIGNED16
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ – FE$_h$: CNResponseTime_U16**

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_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

On Boot-Up, the CN's NMT_CycleTiming_REC.ResponseTime_U16 value is reported to the MN via IdentResponse. The MN shall store the value to the sub-index equal to the CN's Node ID.

The ResponseTime provided by the CN is stored in this parameter for informational purpose only. No internal calculations are done with this value.

Sub-Index	01$_h$ – FE$_h$
Name	CNResponseTime_U16
Entry Category	M
Access	ro
PDO Mapping	No
Value range	UNSIGNED16
Default value	1 000

7.3.3.4 Object 1F92$_h$: NMT_MNCNResTimeout_AU16

This object holds a list of all configured CNs in [ns] with PollRequest to PollResponse timeouts.

Index	1F92h
Name	NMT_MNCNResTimeout_AU16
Object Code	ARRAY
Data Type	UNSIGNED16
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ – FE$_h$: CNResTimeout_U16**

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_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

This parameter describes the EPL node specific timeout values in ns. Whenever a PollRequest frame is sent to a CN this timer will be started. See 4.7.

Sub-Index	01$_h$ – FE$_h$
Name	CNResTimeout_U16
Entry Category	M
Access	rw
PDO Mapping	No
Value range	UNSIGNED16
Default value	25 000$_d$

7.3.3.5 Object 1F8D$_h$: NMT_MNPResPayloadList_AU16

This object holds a list of the PRes Payload data size in [octets] for each configured CN that has to be accessed isochronously, e.g. via PReq / PRes frame exchange.

Index	1F8Dh
Name	NMT_MNPResPayloadList_AU16
Object Code	ARRAY
Data Type	UNSIGNED16
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ – FE$_h$: CNPResPayload_U16**

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_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1, 2 and 8.

Sub-index F0$_h$ shall indicate the payload size of the PRes frame issued by the MN. If the F0$_h$ sub-index value is 0, there shall be no PRes frame transmitted by the MN.

On Boot-Up, the CN's NMT_CycleTiming_REC.PResPayload_U16 value is reported to the MN via IdentResponse. The MN shall store the value to the sub-index equal to the CN's Node ID.

Each value shall be updated, when the respective CN signals, that it is in state NMT_CS_READY_TO_OPERATE.

Sub-Index	01$_h$ – FE$_h$
Name	CNPResPayload_U16
Entry Category	M
Access	rw
PDO Mapping	No
Value range	0 – C_DLL_MAX_PAYLOAD_PRES
Default value	0

7.3.4 CN NMT State Surveillance

The objects described by this paragraph shall be used by the MN surveillance of the CN NMT states as described at 7.1.4.

7.3.4.1 Object 1F8E$_h$: NMT_MNCNCurrState_AU8

This object holds a list of the current NMT states of the configured CNs reported via PRes or StatusResponse.

Index	1F8Eh
Name	NMT_MNCNCurrState_AU8
Object Code	ARRAY
Data Type	UNSIGNED8
Category	M

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ – FE$_h$: CNCurrState_U8**

  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 CN assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1 and 2.

  Sub-index F0$_h$ shall indicate the current state of the MN state machine. It shall hold values described by 7.1.3.

On Boot-Up of the MN, the valid sub-indices shall be set to NMT_CS_NOT_ACTIVE.

Sub-Index	01$_h$ – FE$_h$
Name	CNCurrState_U8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	refer 7.1.4 resp. 7.1.3
Default value	NMT_CS_NOT_ACTIVE

7.3.4.2 Object 1F8F$_h$: NMT_MNCNExpState_AU8

This object holds a list of the NMT states of the configured CNs expected by the EPL NM in accordance to the CNs boot up behaviour and the NMT state commands transmitted by the MN. Refer 7.1.4.

Index	1F8F$_h$
Name	NMT_MNCNExpState_AU8
Object Code	ARRAY
Data Type	UNSIGNED8
Category	O

- **Sub-Index 00$_h$: NumberOfEntries**

Sub-Index	**00h**
Name	NumberOfEntries
Data Type	UNSIGNED8
Entry Category	M
Access	ro
PDO Mapping	No
Value range	1-239
Default value	239

- **Sub-Index 01$_h$ – FE$_h$: CNExpState_U8**

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 CN assigned to the Node ID by index NMT_CNAssignment_AU32.CNAssignment_U32[sub-index] Bits 1and 2.

On Boot-Up of the MN, the valid sub-indices shall be set to NMT_CS_NOT_ACTIVE.

Sub-Index	01$_h$ – FE$_h$
Name	CNExpState_U8
Entry Category	M
Access	rw
PDO Mapping	No
Value range	refer 7.1.4
Default value	NMT_CS_NOT_ACTIVE

7.4 Network Management Services

The Network Management (NMT) is node-oriented and follows a master/slave relationship.

The function of the **NMT master** is taken on by the MN.

CNs are administered as **NMT slaves** by the master. An NMT slave is uniquely identified in the network by its EPL node ID.

According to the definition, the Network Management is directed from the NMT master (MN) to the NMT slaves (CNs).

EPL defines five categories of NMT services:

- **NMT State Command** Services

  By the mean of NMT State Command Services, the MN shall control the state machine of the CNs.
- **NMT Managing Command** Services

  By the mean of NMT Managing Command Services, the MN may access NMT data items of the CNs in a fast coordinated way.
- **NMT Response** Services

  NMT Response Services shall indicate the current NMT state of an CN to the MN.
- **NMT Info** Services

  NMT Info Services may be used to transmit information close to NMT from the MN to an CN.
- **NMT Guard** services

  Through **NMT Guard** Services the MN and CNs detect failures in an EPL network.

An CN may **request NMT** command and info services to be issued by the MN (NMTRequest, see 7.4.6).

7.4.1 NMT State Command Services

The MN shall control the state of the CN via **NMT State Command Services**. The transition between the states shall follow the rules of the **CN state machine** (see 7.1.4).

EPL distinguishes between implicit and explicit NMT State Commands.

7.4.1.1 Implicit NMT State Command Services

At system startup, the reception resp. the timeout of SoA, PRes, PReq resp. SoC frames triggers CN state machine transitions from the state NMT_CS_NOT_ACTIVE to following states. In NMT_CS_PRE_OPERATIONAL_1 the reception of SoC triggers the transition to NMT_CS_PRE_OPERATIONAL_2. SoA, PRes, PReq and SoC are used to synchronise an CN with the current network mode after system start or reset. (see 7.1.4).

At Basic Ethernet Mode, the reception of EPL SoA, PRes, PReq resp. SoC will trigger a change over to NMT_CS_EPL_MODE. (see 7.1.4).

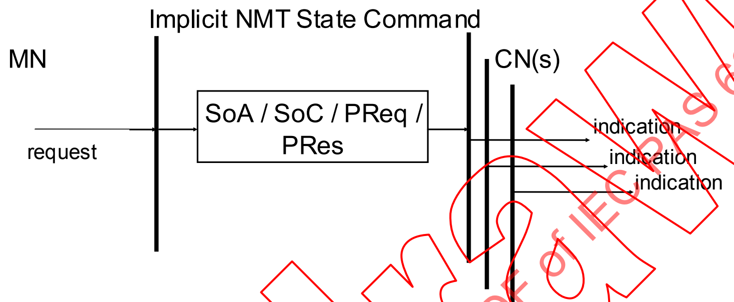


Figure 70 – Implicit NMT State Command Service Protocol

SoA, PRes, PReq resp. SoC acting in the shown way shall be termed **implicit NMT state commands**. They shall be valid regardless their data content and without further extensions.

Table 91 displays implicit NMT state commands in relationship to the initial CN state, where the command is received. SoA resp. SoC reception or timeouts not mentioned by the table don't trigger state transitions.

Table 91 – Implicit NMT State Commands

Initial State	Implicit NMT State Command	Destination State
NMT_CS_NOT_ACTIVE	SoA	NMT_CS_PRE_OPERATIONAL_1
	Timeout (SoC, PReq, PRes, SoA)	NMT_CS_BASIC_ETHERNET
NMT_CS_PRE_OPERATIONAL_1	SoC	NMT_CS_PRE_OPERATIONAL_2
NMT_CS_BASIC_ETHERNET	SoC, PReq, PRes, SoA	NMT_CS_NOT_ACTIVE

7.4.1.1.1 Implicit NMT State Command Transmission

Implicit NMT State Command services (7.4.1.1) don't require explicit NMT frame transmission by the MN. Regular SoA, PRes, PReq resp. SoC frames (refer 4.6.1.1) are valid Implicit NMT State Commands by their own.

SoA, PRes and SoC frame can't be sent directly to a single node, but the MN always has to send it as a multicast. The CN shall decide upon its own current state whether the implicit NMT State command is active.

7.4.1.2 Explicit NMT State Command Services

An **Explicit NMT State Command** shall be transmitted via **ASnd** by the MN.

The target CNs shall be addressed via the ASnd Destination field as unicast to one CN or broadcast to all CNs. A command sent via broadcast shall be inactive at the MN.

Special NMT commands issued as broadcast may determine the validity of the command to the CNs by masking out groups of CNs (see 7.4.1.2.1.2).

The services shall be identified by ASnd **ServiceID = NMT_COMMAND**.

This protocol is used to implement the explicit NMT State Command Services.

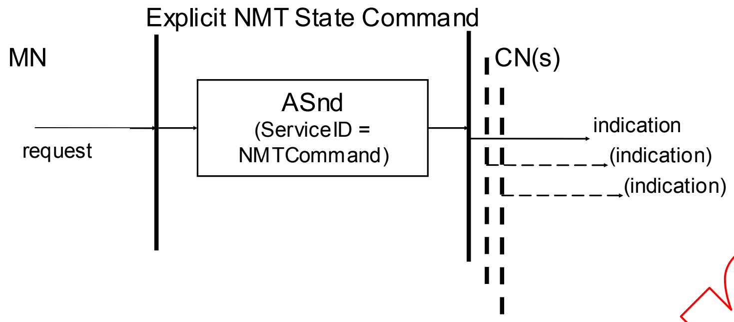


Figure 71 – Explicit NMT State Command Service Protocol

The specific NMT State Command including its data are located at the ASnd "**Service Specific Data**" field. The data block is used as **NMT Service Slot**.

Table 92 – NMT State Command Service, NMT Managing Command Service and NMT Info Service ASnd Service Field Structure

	Bit Offset							
Octet offset [13]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	NMTCommandID							
2	reserved							
3 – n	NMTCommandData							

Table 93 – ASnd Service Data Fields of Explicit NMT State Command Services

Field	**Abbr.**	**Description**	**Value**
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT state command	Table 94, Table 95
NMTCommandData	cdat	shall transport data specific to the NMT state command	

[13] Byte Offset refers to begin of ASnd Service field. Offset relative to Ethernet frame is 17 octets

7.4.1.2.1 Plain NMT State Command

Plain NMT State Commands shall be addressed unicast to a specific CN or broadcast to all CNs. For Plain NMT State Commands the "NMTCommandData" shall be ignored.

The following Plain NMT State Commands are defined:

Table 94 – Plain NMT State Commands

NMTCommandID		M/O	Initial state	Destination state
NMTStartNode	21_h	M	NMT_CS_ READY_TO_OPERATE	NMT_CS_ OPERATIONAL
NMTStopNode	22_h	M	NMT_CS_ PRE_OPERATIONAL_2	NMT_CS_ STOPPED
			NMT_CS_ READY_TO_OPERATE	
			NMT_CS_OPERATIONAL	
NMTEnter-PreOperational2	23_h	M	NMT_CS_OPERATIONAL	NMT_CS_ PRE_OPERATIONAL_2
			NMT_CS_STOPPED	
NMTEnable-ReadyTo Operate	24_h	M	NMT_CS_ PRE_OPERATIONAL_2	NMT_CS_ READY_TO_OPERATE
NMTResetNode	28_h	M	NMT_CS/MS_NOT_ACTIVE	NMT_GS_ INITIALISATION sub-state NMT_GS_ RESET_APPLICATION
			NMT_CS/MS_ PRE_OPERATIONAL_1	
			NMT_CS/MS_ PRE_OPERATIONAL_2	
			NMT_CS/MS_ READY_TO_OPERATE	
			NMT_CS/MS_ OPERATIONAL	
			NMT_CS_STOPPED	
			NMT_CS_ BASIC_ETHERNET	
NMTReset-Communication	29_h	M	NMT_CS/MS_NOT_ACTIVE	NMT_GS_ INITIALISATION sub-state NMT_GS_ RESET_COMMUNICATION
			NMT_CS/MS_ PRE_OPERATIONAL_1	
			NMT_CS/MS_ PRE_OPERATIONAL_2	
			NMT_CS/MS_ READY_TO_OPERATE	
			NMT_CS/MS_ OPERATIONAL	
			NMT_CS_STOPPED	
			NMT_CS_ BASIC_ETHERNET	

The NMTCommandIDs between 20_h and $3F_h$ are reserved for Plain NMT State Commands.

The commands shall be only active in the respective initial states. The command shall be ignored in the other states and an entry in the error log of the CN shall be made.

7.4.1.2.1.1 NMT State Commands to the MN

NMTResetNode and NMTResetCommunication may be requested by a diagnostic node (refer 7.4.6) to be addressed as unicast to the MN. The MN shall perform the requested reset.

7.4.1.2.1.2 Extended NMT State Command

Extended NMT State Commands may be used to **access groups** of CNs.

The ASnd transporting the command shall be addressed as broadcast to all CNs.

The **"NMTCommand Specific Data"** field shall contain an **EPL Node List** according to the EPL Node List Format (see 7.4.1.2.1.3). The EPL Node List shall **indicate the validity** of the command for the individual nodes. Node IDs to that the command is addressed shall be indicated by 1_b. Nodes that have to ignore the command shall be indicated by 0_b.

Table 95 lists the Extended NMT State Commands. Initial State and Destination State and the validity of the commands at the initial states shall be identical to the respective plain commands.

Table 95 – Extended NMT State Commands

NMTCommandID		M/O
NMTStartNodeEx	41_h	O
NMTStopNodeEx	42_h	O
NMTEnterPreOperational2Ex	43_h	O
NMTEnableReadyToOperateEx	44_h	O
NMTResetNodeEx	48_h	O
NMTResetCommunicationEx	49_h	O

The NMTCommandIDs between 40_h and $5F_h$ are reserved for Extended NMT State Commands.

Support of Extended NMT State Commands shall be indicated by Index FeatureFlags Bit xyz.

7.4.1.2.1.3 EPL Node List Format

The **EPL Node List** format provides one bit for each Node ID **identifying** an EPL Node.

The EPL Node IDs shall be assigned to the bit of the data field as follows:

Table 96 – EPL Node List: Node ID to Bit Assignment

	Bit offset							
Octet offset [14]	**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

[14] Byte Offset refers to begin of NMTCommand Specific Data. Offset relative to Ethernet frame is 20 octets

7.4.2 NMT Managing Command Services

The MN may use **NMT Managing Command** Services to **administer NMT-relevant entries** in the database of the CN. These commands do not directly influence the state machine of the CN.

An NMT Managing Commands shall be transmitted via ASnd by the MN. The target CNs shall be addressed via the ASnd Destination field as unicast to one CN or broadcast to all CNs. The validity of the respective addressing scheme shall be considered regarding each individual command.

The services shall be identified by ASnd ServiceID = NMTCommand.

This protocol is used to implement the NMT Managing Command Services.

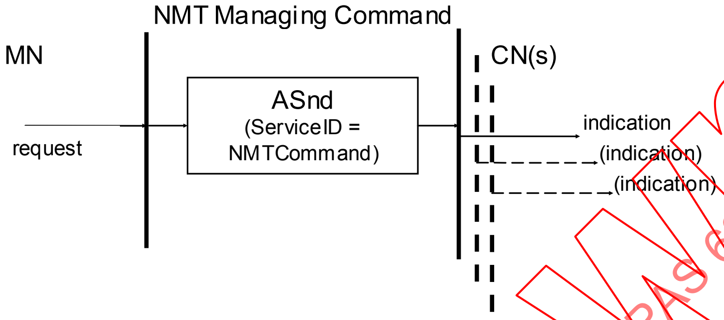


Figure 72 – NMT Managing Command Service Protocol

The specific NMT Managing Commands incl. its data are located at the ASnd "**Payload**" field (see 0). The data block is used as NMT service slot .

Table 97 – ASnd Service Data Fields of NMT Managing Command Services

Field	Abbr.	Description	Value
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT Managing Command	Table 98
NMTCommandData	cdat	Shall transport data specific to the NMT Managing Command	

The following NMT Managing Command Services are defined:

Table 98 – NMT Managing Command Services

NMTCommandID		M/O	Short description
NMTNetParameterSet	61_h	M	Sets IP parameter of an individual CN
NMTNetHostNameSet	62_h	M	Sets host name of an individual CN
NMTFlushArpEntry	63_h	M	Clears local MAC and IP address list at all CNs

The NMTCommandIDs between 60_h and $7F_h$ are reserved for NMT Managing Commands.

7.4.2.1 Service Descriptions

7.4.2.1.1 NMTNetParameterSet

NMTNetParameterSet shall **define the IP address parameter** of the addressed CN. Zero values shall be used to indicate parameter entries, that have to be ignored by the CN, e.g. don't change current IP settings.

Table 99 – NMTNetParameterSet ASnd Service Slot Structure

	Bit Offset							
Octet Offset [15]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	NMTCommandID							
2	reserved							
3 - 6	IPAddress							
7 – 10	SubnetMask							
11 – 14	DefaultGateway							
15 – 18	MTU							
19 – 42	reserved							

Table 100 – ASnd Service Slot Data Fields of NMTNetParameterSet

Field	**Abbr.**	**Description**	**Value**
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT Managing Command	61_h
IPAddress	ipa	May be used to modify CN's local IP address setting, 0000_h shall be used to indicate no change of current local settings.	
SubnetMask	snm	May be used to modify CN's local IP Subnet Mask setting, 0000_h shall be used to indicate no change of current local settings	
DefaultGateway	gtw	May be used to modify CN's local IP Default Gateway setting, 0000_h shall be used to indicate no change of current local settings	
MTU	mtu	Shall be the size of the largest IP frame that can be transmitted over the network, including the size of the transport header.	64_d – 1518_d a value of 0_d shall indicate no change of current local settings

IPAddress, SubnetMask and DefaultGateway are not supported by EPL V2.0. These fields shall be considered reserved.

The command shall be addressed unicast to an individual CN.

[15] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

After execution of NMTNetParameterSet command, the **modified parameters shall be published** by the CN. Publishing shall be executed in the following manner:

1. CN shall indicate an NMTRequest by the RS bits of PRes resp. StatusResponse using the priority level PrioNMTRequest.
2. MN shall assign the asynchronous period to the CN via SoA to be used by an NMTRequest ASnd frame.
3. CN shall request an IdentRequest to itself using the NMTRequest ASnd frame.
4. MN shall transmit an SoA including an IdentRequest to the requesting CN.
5. CN shall publish it's modified IP parameter via an IdentResponse ASnd frame.

7.4.2.1.2 NMTNetSetHostName

NMTNetSetHostName shall set the **host name** of the CN.

Table 101 – NMTNetSetHostName ASnd Service Slot Structure

	Bit Offset							
Octet offset [16]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	NMTCommandID							
2	reserved							
3 – 34	HostName							
35 – 42	reserved							

Table 102 – ASnd Service Slot Data Fields of NMTNetSetHostName

Field	**Abbr.**	**Description**	**Value**
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT Managing Command	62h
HostName	hn	May be used to modify CN's local DNS host name	

The command shall be addressed unicast to an individual CN.

After execution of NMTNetSetHostName command, the **modified host name shall be published** by the CN. Publishing shall be executed by the following manner:

1. CN shall indicate an NMTRequest by the RS bits of PRes resp. StatusResponse using the priority level PrioNMTRequest.
2. MN shall assign the asynchronous period to the CN via SoA to be used by an NMTRequest ASnd frame.
3. CN shall request an IdentRequest to itself using the NMTRequest ASnd frame.
4. MN shall transmit an SoA including an IdentRequest to the requesting CN.
5. CN shall publish it's modified IP parameter via an IdentResponse ASnd frame.

[16] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

7.4.2.1.3 NMTFlushArpEntry

NMTFlushArpEntry shall **remove the entries for an CN** from the address tables of all other CNs. The entry to be eliminated shall defined by the Node ID of the CN to be removed.

Table 103 – NMTFlushArpEntry ASnd Service Slot Structure

	Bit Offset							
Octet offset [17]	7	6	5	4	3	2	1	0
0	ServiceID							
1	NMTCommandID							
2	reserved							
3	Node ID							
4 – 42	reserved							

Table 104 – ASnd Service Slot Data Fields of NMTFlushArpEntry

Field	**Abbr.**	**Description**	**Value**
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT Managing Command	63h
NodeID	nid	shall identify the node whose ARP entry shall be deleted	

The command shall be addressed broadcast to all CNs.

7.4.3 NMT Response Services

NMT Response Services shall be used by the MN to **query NMT information** from the CN, e.g. current state, error and setup data.

7.4.3.1 NMTStateResponse

The CNs shall signal their state to the MN via **NMTStateResponse services**.

CNs that communicate synchronously via PReq / PRes shall use the **PRes** frame to indicate their current state.

This protocol is used to implement the NMT State Response Service from isochronously communicating CNs.

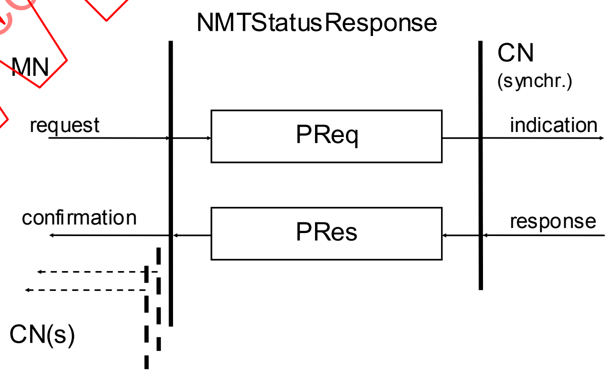


Figure 73 – NMT Response Service Protocol (Isochronous CN)

MN shall receive NMTStateResponse, CNs may do so.

[17] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

CNs which do not communicate isochronously shall signal their state via **StatusResponse and IdentResponse ASnd** frames, which may be Async-only CNs as well as CNs in state NMT_CS_PRE_OPERATIONAL_1.

This protocol is used to implement the NMT Response Service from not isochronously communicating CNs.

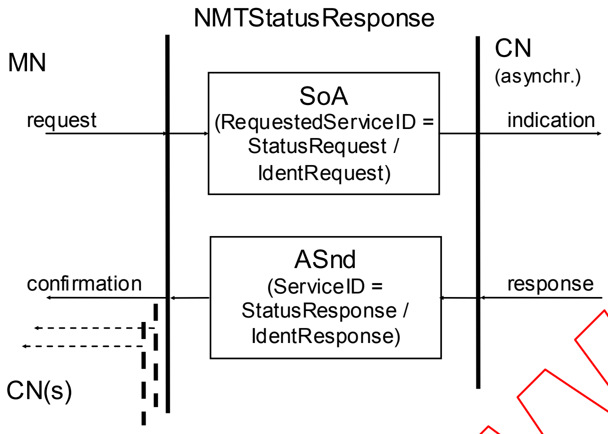


Figure 74 – NMT Response Service Protocol (Acyclic-only CN)

MN shall receive NMTStateResponse. CNs may do so, if the NMTStatusResponse is transmitted via IdentResponse.

NMTStatusResponse shall use the following data fields of the PRes resp. StatusResponse or IdentResponse frame:

- NMTStatus
  reports the current NMT status of the CN
- EmergencyNew (EN)
  reports unread entries at the CN emergency queue (see 6.6.1.3)

Refer 7.4.3.2.1 and 7.4.3.3.1 for detailed information about frame format.

7.4.3.2 IdentResponse Service

IdentResponse service shall be used by the MN to **identify** configured but unrecognized CNs at system startup or after loss of communication. The service may be used after startup to **query the CN's setup information**.

This protocol is used to implement the IdentResponse service.

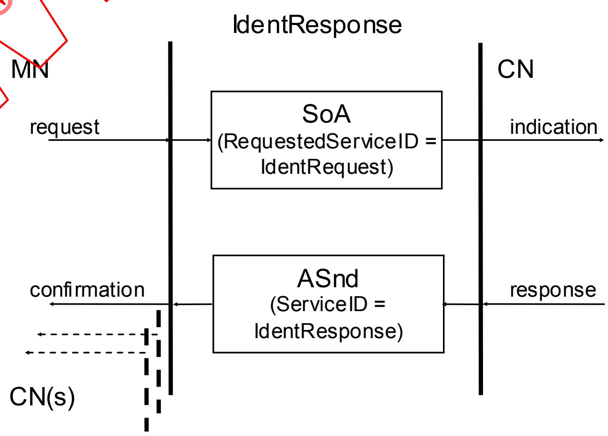


Figure 75 – IdentResponse Service Protocol

MN shall receive IdentResponse, CNs may do so.

IdentResponse service can be initiated by an CN via the NMT Request mechanism (7.4.6). The NMTRequestedCommandID field of the NMT requesting ASnd frame shall be set to IDENT_REQUEST.

ASnd transporting IdentResponse shall be addressed as broadcast.

7.4.3.2.1 IdentResponse Frame

Table 105 – IdentResponse ASnd Service Slot Structure

	Bit Offset							
Octet Offset [18]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	res	res	res	EN	res	res	CF	res
2		PR			RS			
3	NMTStatus							
4	reserved							
5 - 6	EPLProfileVersion							
7 - 10	FeatureFlags							
11 - 12	MTU							
13 – 14	PollInSize							
15 - 16	PollOutSize							
17 - 20	ResponseTime							
21 - 22	reserved							
23 – 26	DeviceType							
26 – 30	VendorID							
31 - 34	ProductCode							
35 - 38	RevisionNumber							
39 - 42	SerialNumber							
43 - 50	VendorSpecificExtension1							
51 – 54	VerifyConfigurationDate							
55 – 58	VerifyConfigurationTime							
59 - 62	ApplicationSwDate							
63 - 66	ApplicationSwTime							
67 - 70	IPAddress							
71 – 74	SubnetMask							
75 - 78	DefaultGateway							
79 - 110	HostName							
111 – 158	VendorSpecificExtension2							
159 - 234	reserved							

[18] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

Table 106 – ASnd Service Slot Data Fields of IdentResponse

Field	Abbr	Description	Value
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	IDENT_ RESPONSE
ExceptionNew	EN	Flag: Error signalling (see 6.6.1.3)	
Configured	CN	Flag: Shall be set to 1_b is the CN has completed it's configuration	
Priority	PR	Flags: Shall indicate the priority of the requested asynchronous frame. (see 4.2.4.1.3.2)	0, 7
RequestToSend	RS	Flags: Shall indicate the number of pending requests to send at the CN. The value 7 shall indicate 7 or more requests, 0 shall indicate no pending requests	0 - 7
NMTStatus	stat	Shall report the current status of the CN's NMT state machine	
EPL Profile Version	ever	Shall indicate the EPL specification version to that the CN conforms	
FeatureFlags	feat	tbd	
MTU	mtu	Shall be the size of the largest IP frame that can be transmitted over the network, including the size of the transport header.	64 – 1518
PollInSize	pis	Shall report the CN setting for PollRequest data block size, index NMT_CycleTiming_REC.PReqActPayload_U16	
PollOutSize	pos	Shall reports the CN setting for PollResponse data block size, index NMT_CycleTiming_REC.PResActPayload_U16	
ResponseTime	rst	Shall report the time, that is required by the CN to respond to PReq or the assignment of the asynchronous period via SoA, index NMT_CycleTiming_REC.PResMaxLatency_U32	
DeviceType	dt	Shall report the CN's Device type, index NMT_DeviceType_U32	
VendorID	vid	Shall report the CN's Vendor ID, index NMT_IdentityObject_REC.VendorId_U32	
ProductCode		Shall report the CN's Product Code, index NMT_IdentityObject_REC.ProductCode_U32	
RevisionNumber	rno	Shall report the CN's Revision Number, index NMT_IdentityObject_REC.RevisionNo_U32	
SerialNumber	sno	Shall report the CN's Serial Number, index NMT_IdentityObject_REC.SerialNo_U32	
VendorSpecific-Extension1	vex1	May be used for vendor specific purpose, shall be filled with 0s if not in use	
Verify-ConfigurationDate	vcd	Shall report the CN's Configuration date, index CFG_LocVerifyConfig_REC.VerifyConfigDate_U32	
Verify-ConfigurationTime	vct	Shall report the CN's Configuration time,index CFG_LocVerifyConfig_REC.VerifyConfigTime_U32	
ApplicationSW-Date	ad	Shall report the CN's Application SW date, index PDL_LocApplSw_REC.ApplSwDate_U32	
ApplicationSW-Time	ad	Shall report the CN's Application SW time, index PDL_LocApplSw_REC.ApplSwTime_U32	
IPAddress	ipa	Shall report the current IP address value of the CN, index NWL_IpAddrTable_*Xh*_REC.Addr_IPAD	
SubnetMask	snm	Shall report the current IP subnet mask value of the CN, index NWL_IpAddrTable_*Xh*_REC.NetMask_IPAD	
DefaultGateway	gtw	Shall report the current IP default gateway value of the CN	
HostName	hn	Shall report the current DNS host name of the CN	
VendorSpecific-Extension2	vex2	May be used for vendor specific purpose, shall be filled with 0s if not in use	

7.4.3.3 StatusResponse service

StatusResponse service shall be used by the MN to **query the current status** of not isochronously communicating CNs.

This protocol is used to implement the StatusResponse Service.

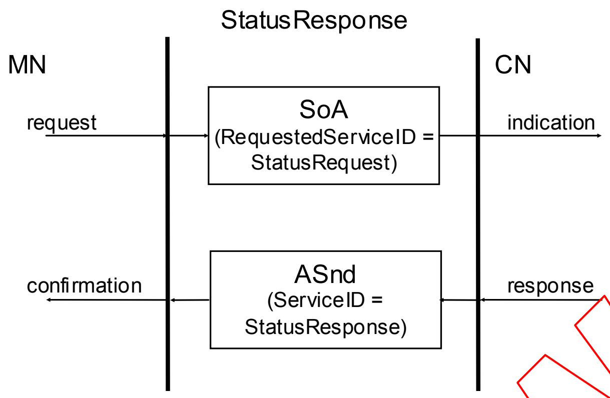


Figure 76 – StatusResponse Service Protocol

StatusResponse ASnd frame is transmitted to the MN only.

StatusResponse service can be initiated by an CN via the NMT Request mechanism (7.4.6). The NMTRequestedCommandID entry of the NMT requesting ASnd frame shall be set to StatusRequest.

StatusResponse shall be addressed as unicast to the MN.

7.4.3.3.1 StatusResponse Frame

Table 107 – StatusResponse ASnd Service Slot Structure

	Bit Offset							
Octet offset [19]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID = StatusResponse							
1	res	res	res	EN	res	res	CF	res
2	reserved		PR			RS		
3	NMT Status							
4 - 6	reserved							
7 - 14	StaticErrorBitField							
15 – 15+n*20	List of Errors / Events (see 6.6.1.7) 20 Byte / Entry , minimum n=2							

n (2-14) : Number of error/event entries

[19] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

Table 108 – ASnd Service Slot Data Fields of StatusResponse

Field	Abbr	Description	Value
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	STATUS_ RESPONSE
ExceptionNew	EN	Flag: Error signalling (see 6.6.1.3)	
Priority	PR	Flags: Shall indicate the priority of the requested asynchronous frame. (see 4.2.4.1.3.2)	0, 7
RequestToSend	RS	Flags: Shall indicate the number of pending requests to send at the CN. The value 7 shall indicate 7 or more requests, 0 shall indicate no pending requests	0 - 7
NMTStatus	stat	Shall report the current status of the CN's NMT state machine	
StaticErrorBitfield	seb	This entry shall indicate pending errors at the CN by specific bits being set (see 6.6.1.7.1) for encoding of errors in the bitfield)	
ErrorCodeList	el	This entry shall contain a list of error, that have occurred at the CN. Each Error code has a size of 8 octets (see 6.6.1.7.2)	

7.4.4 NMT Info Services

NMT Info Services may be 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.

An NMT Info services shall be transmitted via **ASnd** by the MN.

The target CNs shall be addressed via the ASnd Destination field as unicast to one CN or broadcast to all CNs.

The services shall be identified by ASnd **ServiceID** = NMTCommand.

This protocol is used to implement the NMT Info services.

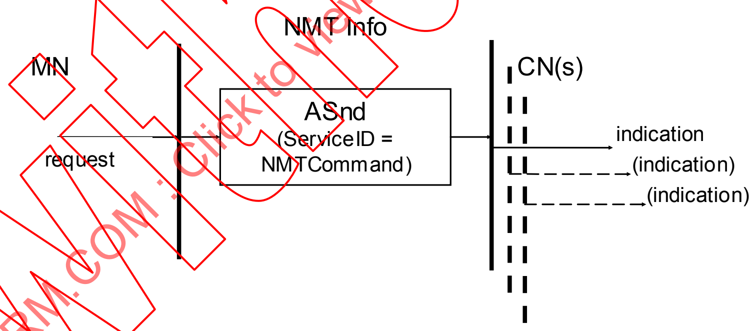


Figure 77 – NMT Info Service Protocol

The specific NMT Info services including its data are located in the ASnd "**Service specific data**" field. The data block is used as NMT service slot (refer. **Error! Reference source not found.**).

The NMT Info Services use the following parameters of the NMT Service slot:

Table 109 – ASnd Service Slot Data Fields of NMT Managing Info Services

Field	Abbr.	Description	Value
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT Info service	Table 110
NMTCommandData	cdat	Shall transport data specific to the NMT Info service	

The following NMT Info services are defined:

Table 110 – NMT Info Services

NMTCommandID		M/O MN	M/O CN	Short description
NMTPublishConfiguredCN	80_h	O	O	Provides CNs designated in the configuration of the MN
NMTPublishActiveCN	90_h	O	O	Provides CNs that have been identified by the MN
NMTPublishPreOperational1	91_h	O	O	Provides active CNs in the state NMT_CS_PRE_OPERATIONAL_1
NMTPublishPreOperational2	92_h	O	O	Provides active CNs in the state NMT_CS_PRE_OPERATIONAL_2
NMTPublish-ReadyToOperate	93_h	O	O	Provides CNs in the state NMT_CS_READY_TO_OPERATE
NMTPublishOperational	94_h	O	O	Provides CNs in the state NMT_CS_OPERATIONAL
NMTPublishStopped	95_h	O	O	Provides CNs in the state NMT_CS_STOPPED
NMTPublishStatusNew	$A0_h$	O	O	Provides active CNs with the emergency new flag (EN) set
NMTPublishTime	$B0_h$	O	O	Provides the system time

The NMTCommandIDs between 80_h and BF_h are reserved for NMT Info services.

Support of Extended NMT Info services shall be indicated by index FeatureFlags tbd.

7.4.4.1 Service Descriptions

7.4.4.1.1 NMTPublishConfiguredCN

By means of this service, the MN may publish a **list of CNs being configured** in its configuration.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to configured CNs shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNAssignment_AU32.Bit1.

7.4.4.1.2 NMTPublishActiveCN

By means of this service, the MN may publish a **list of the active CNs**. An CN is active after the MN has recognized an **IdentResponse** ASnd frame as a response to the corresponding requests.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to active CNs shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8. CN 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 shall be regarded to be active.

7.4.4.1.3 NMTPublishPreOperational1

By means of this service, the MN may publish a **list of CNs in the state NMT_CS_PRE_OPERATIONAL_1**.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to active CNs in the state NMT_CS_PRE_OPERATIONAL_1 shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8.

7.4.4.1.4 NMTPublishPreOperational2

By means of this service, the MN may publish a **list of CNs in the state NMT_CS_PRE_OPERATIONAL_2**.

The service uses the **EPL Node List format** (7.4.1.2.1.3). Node IDs that correspond to active CNs in the state NMT_CS_PRE_OPERATIONAL_2 shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8.

7.4.4.1.5 NMTPublishReadyToOperate

By means of this service, the MN may publish a **list of CNs in the state NMT_CS_READY_TO_OPERATE**.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to active CNs in the state NMT_CS_READY_TO_OPERATE shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8.

7.4.4.1.6 NMTPublishOperational

By means of this service, the MN may publish a **list of CNs in the state NMT_CS_OPERATIONAL.**

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to active CNs in the state NMT_CS_OPERATIONAL shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8.

7.4.4.1.7 NMTPublishStopped

By means of this service, the MN may publish a **list of CNs in the state NMT_CS_STOPPED**.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to active CNs in the state NMT_CS_STOPPED shall be indicated by 1_b.

Information to be published shall be gathered from index NMT_CNCurrState_AU8.

7.4.4.1.8 NMTPublishEmergencyNew

By means of this service, the MN may publish a **list of CNs with the EmergencyNew flag (EN)** being set in the NMTStatusResponse feedback.

The service uses the **EPL Node List** format (7.4.1.2.1.3). Node IDs that correspond to CNs with the set EN flag shall be indicated by 1_b.

7.4.4.1.9 NMTPublishTime

By means of this service, the MN shall publish **date and time**.

Table 111 – NMTPublishTime ASnd Service Slot Structure

	Bit Offset							
Octet Offset [20]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	NMTCommandID							
2	reserved							
3 – 8	DateTime							
9 – 42	reserved							

Table 112 – ASnd Service Slot Data Fields of NMT Managing Command Services

Field	**Abbr.**	**Description**	**Value**
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMT_COMMAND
NMTCommandID	cid	Shall qualify the NMT info service	B0h
DateTime	dt	current RTC time value of the CN coded as TIME_OF_DAY	

[20] Byte Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets.

7.4.5 NMT Guard Services

Through **NMT Guard Services** MN and CNs detect failures in an Ethernet Powerlink-based network.

Local errors in a node may e.g. lead to a reset or change of state. The definition of these local errors does not fall into the scope of this specification.

7.4.5.1 Guarding EPL Controlled Nodes

ETHERNET Powerlink uses a guard mechanism to monitor the CNs. The MN queries the CNs and receives their reply.

In order to query **synchronously-addressed** CNs in the EPL Protected Mode, the guard mechanism shall use the synchronous **PReq / PRes message exchange**.

Using this way, also CNs are able to monitor other CNs. The monitoring time is configurable via the object dictionary entry **NMT_ConsumerHeartbeatTime_AU32**.

An asynchronous channel shall be allocated to nodes which are **not addressed synchronously**. They shall be queried via an **StatusRequest** in the SoA telegram and respond with an ASnd **StatusResponse** message.

7.4.5.2 Guarding EPL Managing Node

The CNs shall control the function of the MN by **timeout-monitoring the SoC** frames (see 7.4.1.1). If they do not receive any frames, they shall change to the state NMT_CS_PRE_OPERATIONAL_1. This transition shall be signalled to the application of the CN.

7.4.6 Requesting NMT Services by an CN

An 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** shall initiate an NMTRequest by the **RS bits of PRes resp. StatusResponse** using the priority level C_DLL_PRIO_NMT_REQUEST.
2. The **MN** shall **assign the asynchronous period** to the CN via **SoA** to be used by an NMTRequest ASnd frame (SoA RequestedServiceID = NMTRequestInvite).
3. **CN** shall request the desired service using the **NMTRequest ASnd** frame (ASnd ServiceID = NMTRequest).

7.4.6.1 NMTRequest ASnd Frame

NMTRequests shall be transmitted by an CN upon assignment of the asynchronous period via an NMTRequestInvite in the SoA frame.

Table 113 – NMTRequest ASnd Service Slot Structure

	Bit Offset							
Octet offset [21]	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
0	ServiceID							
1	NMTRequestedCommandID							
2	NMTRequestedCommandTarget							
2 – n	NMTRequestedCommandData							

[21] Octet Offset refers to begin of ASnd service slot. Offset relative to Ethernet frame is 17 octets

Table 114 – ASnd Service Slot Data Fields of NMT Managing Command Services

Field	Abbr.	Description	Value
ServiceID	svid	Shall qualify the service ID dedicated to the asynchronous period	NMTRequest
NMTRequested-CommandID	rcid	Shall qualify the NMT service to be issued by the MN by its NMTCommandID value. StatusResponse and IdentResponse service shall be qualified by their ASnd ServiceID values	
NMTRequested-CommandTarget	rct	Shall indicate the target node of the requested NMT command	
NMTRequested-CommandData	rcd	May contain 40 – 1493 octets of NMT command specific data to be issued by the MN. The lower layer shall be responsible for padding.	

The NMTRequest transporting ASnd frame shall be addressed as unicast to the MN.

7.4.6.1.1 Invalid NMTRequests

If the CN requests an NMT services not supported by the MN, the MN shall respond by an unicast ASnd frame with **ServiceID** = NMTCommand to the requesting CN.

The following NMTCommandID shall be used:

- FF_h **NMTInvalidService**

This special service is mandatory. It shall not influence the state machine of the CNs and shall not transport any data, e.g. **NMTCommand specific data** shall be ignored.

7.5 Boot-Up CN

The Boot-Up procedure of an CN shall be implemented device specific.

The requirements defined by the CN NMT State Machine (see 7.1.4) and the additional cooperation requirements defined by the MN Boot-Up (see 7.6) shall be fulfilled.

7.6 Boot-Up MN

7.6.1 EPL Managing Node, Terms and Definitions

Besides the application process several different additional functionalities can may exist in an EPL system. These functionalities are referred to by different terms. This subclause is intended to clarify these terms.

Within a distributed system the application process is divided into several parts running on different nodes. From the applications point of view usually one node is responsible for the control of the system. This node is called *application master* (e.g. a PLC).

From the network's point of view there are several additional functionalities which not directly deal with the application but provide application supporting functions. These additional functionalities are based on a master / slave, client / server or producer / consumer relationship.

- NMT Master

  The Network Management (NMT) provides services for controlling the network behavior of nodes as defined in 7.1.4. All nodes of a network referred to as NMT Slaves are controlled by services provided by an NMT master NMT Master and which have to be executed by an NMT master NMT Master application. Usually the NMT master NMT Master application is also part of the application master. It shall reside on the MN.

- Configuration Manager

  The Configuration Manager is an optional functionality which provides mechanisms for configuration of nodes in an EPL network during boot-up as defined in 6.8. The mechanisms are called Configuration Management CMT.

- SYNC Producer

  The SYNC Producer is a functionality, which is responsible for transmitting the Synchronization frame SoC. It shall reside on the MN.

- TIME Producer

  The TIME Producer is an optional functionality, which is responsible for transmitting the NetTime in the SoC. It shall reside on the MN.

Because it is usual to combine several of the additional functionalities on one node an additional term is introduced: the EPL Managing Node (MN).

Basically all object dictionary entries referenced inby this document paragraph are optional. If denoted as mandatory, this is valid if the concerned functionality is provided by the device. Some objects consist of a set of bits, specifying several kinds of behavior (as e.g. 1F80h). Only those bits have to be implemented that correspond to a supported behavior.

7.6.2 Boot-Up Procedure

7.6.2.1 Overview

When a EPL Node starts after PowerOn (NMT_GT_1) or Reset (NMT_GT_2), it will perform the state machine according to the NMT state diagram of an EPL node (see 7.1.2) and will attain the NMT_GS_INITIALISATION state automatically. After completion of the initialization, the node enters the super state NMT_GS_COMMUNICATING. Within this super state the MN NMT state machine or the CN NMT state machine is taken depending on the configured Node ID. There shall be transitions back to the NMT_GS_INITIALISATION state from every state within this super state after the reception of the NMT command service NMTResetNode (NMT_GT4) or NMTResetCommunication (NMT_GT5) or after internal errors (NMT_GT6).

7.6.2.2 NMT_MS_NOT_ACTIVE

The purpose of the NMT_MS_NOT_ACTIVE state is the recognition of bus activity. Normally only the MN is allowed to start communication with the CNs. If the MN detects SoC or SoA frames during this state, a second MN is active on the network.

This state is entered from the NMT_GS_INITIALISATION (NMT_MT1) state.

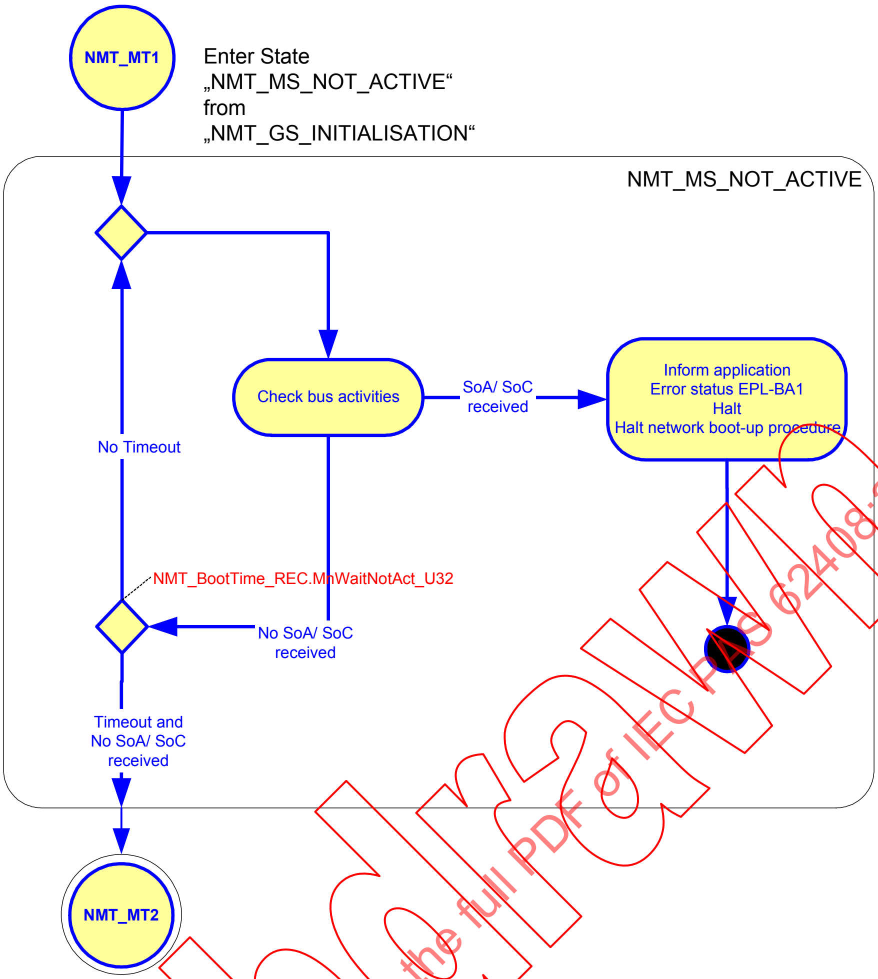


Figure 78 – Detail state NMT_MS_NOT_ACTIVE

The steps of the NMT_MS_NOT_ACTIVE state are as follows:

- Check the bus activity. Reception of SoC or SoA frames indicates that there is another MN working on the network. In case of the reception of SoA or SoC frames, the following error shall be signaled to the MN application:

  EPL-BA1: Device is configured as an MN and detecting a second MN on the bus (SOC, IdentRequest,….)

  The current MN state shall be maintained. The MN shall halt network boot up procedure. A transition to the NMT_MS_INITIALISATION state is only possible after the reception of NMT command service NMTResetCommunication or NMTResetNode.
- In normal operation (no error), the transition from NMT_MS_NOT_ACTIVE to NMT_MS_PRE_OPERATIONAL_1 (NMT_MT2) shall be triggered, if there are no SoA or SoC frames received within a time interval defined by index NMT_BootTime_REC.MNWaitNoAct_U32.

7.6.2.3 NMT_MS_PRE_OPERATIONAL_1

The purpose of the NMT_MS_PRE_OPERATIONAL_1 state is the identification of all configured CNs and the examination of their configuration versions. The MN communicates with the CN using the Reduced EPL Cycle with SoA and ASnd frames (see 4.2.4.2). In case of communication errors the MN communicates the error to the application and 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.

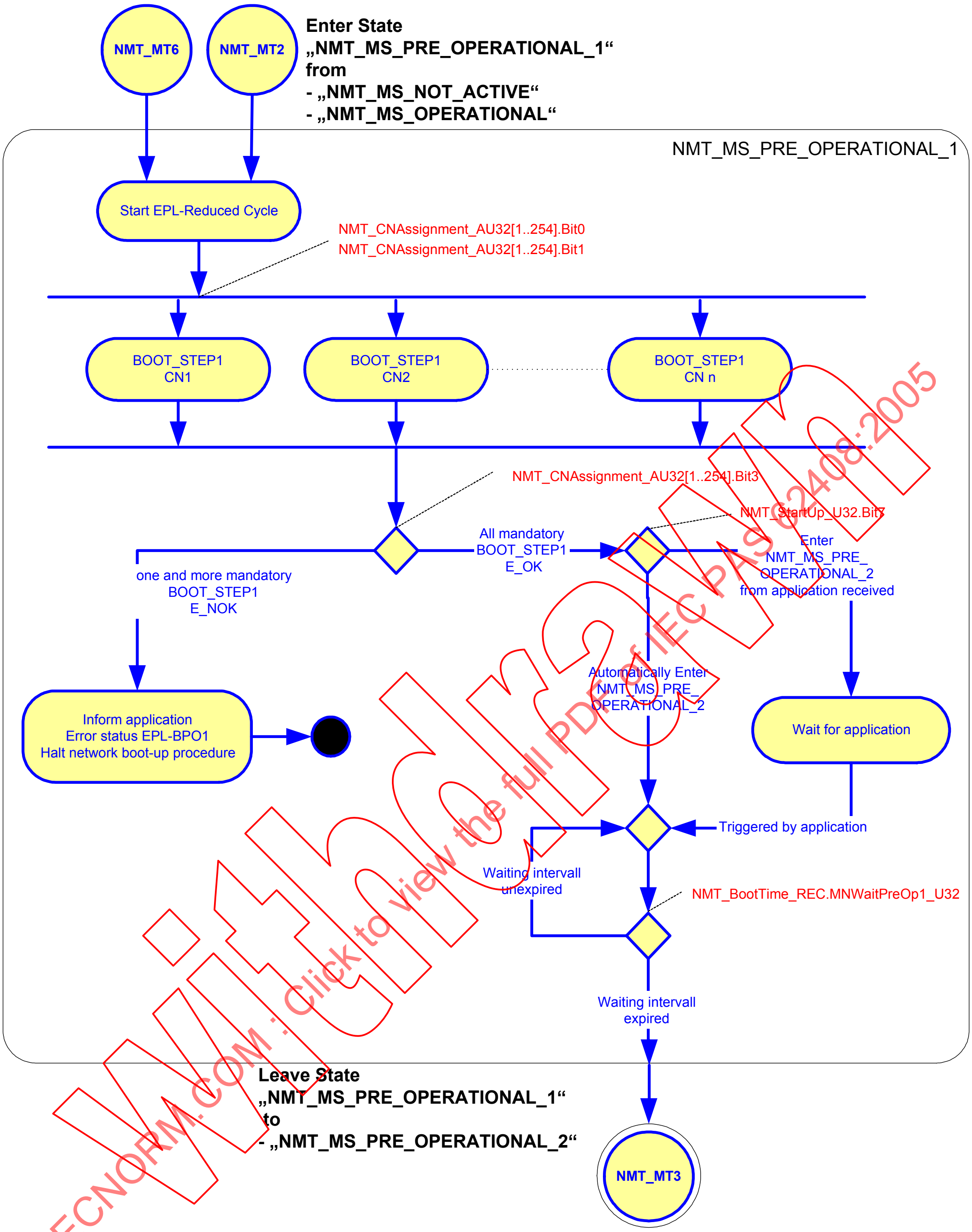


Figure 79 – Detail state NMT_MS_PRE_OPERATIONAL_1

The steps of the NMT_MS_PRE_OPERATIONAL_1 process are as follows:

- Start communication with the CNs in the Reduced EPL cycle.
- Start parallel process BOOT_STEP1 for all configured CN. For optional CNs the process runs endlessly until the CN is found. Configured CNs are identified by:
  - NMT_CNAssignment_AU32[1..254].Bit0
  - NMT_CNAssignment_AU32[1..254].Bit1
- After the BOOT_STEP1 process are terminated successfully for all mandatory CN, the MN will switch to the state NMT_MS_PRE_OPERATIONAL_2. Mandatory CNs are identified by :
  - NMT_CNAssignment_AU32[1..254].Bit3,
- The MN will switch to the state NMT_MS_PRE_OPERATIONAL_2 direct or after the request of the application. The transition policy is defined by:
  - NMT_Startup_U32.Bit7.

  If the waiting time interval, the MN has to stay in the NMT_MS_PRE_OPEARTIONAL_1 state
  - NMT_BootTime_REC.MnWaitPreOp1_U32

  has not expired, the MN waits until the waiting time expired
- In case of errors in the BOOT_STEP1 state of one or more mandatory CN, the following error shall be signaled to the MN application:

  EPL-BPO1 Boot configuration in the NMT_MS_PRE_OPERATIONAL_1 state failed. All or some of the mandatory configured slaves failed

  The current MN state shall be maintained. The MN shall halt network boot up procedure. A transition to the NMT_MS_INITIALISATION state is only possible after the reception of NMT command service NMTResetCommunication (NMT_GT5) or NMTResetNode (NMT_GT4).

7.6.2.3.1 BOOT_STEP1

The BOOT_STEP1 process is started parallel for all configured CN. The purpose of the BOOT_STEP1 state is the identification of a CN and the examination of its configuration.

The BOOT_STEP1[Node ID] returns with status E_OK after all examinations are terminated successfully. If one of the examinations fails, BOOT_STEP1[Node ID] returns with status E_NOK

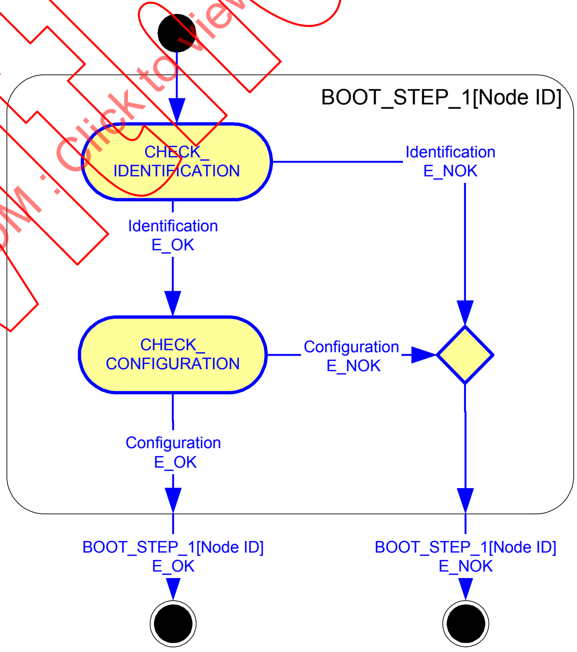


Figure 80 – Sub-state BOOT_STEP1

The steps of the BOOT_STEP1 process are as follows:

- Check Identification (see 7.6.2.3.1.1)
- Check Configuration (see 7.6.2.3.1.2).

7.6.2.3.1.1 CHECK_IDENTIFICATION

The purpose of the CHECK_IDENTIFICATION state is the examination of the identification of a CN. The CHECK_IDENTIFICATION [Node ID] returns with status E_OK after all identifications are terminated successfully. If one of the identifications fails, CHECK_IDENTIFICATION [Node ID] returns with status E_NOK.

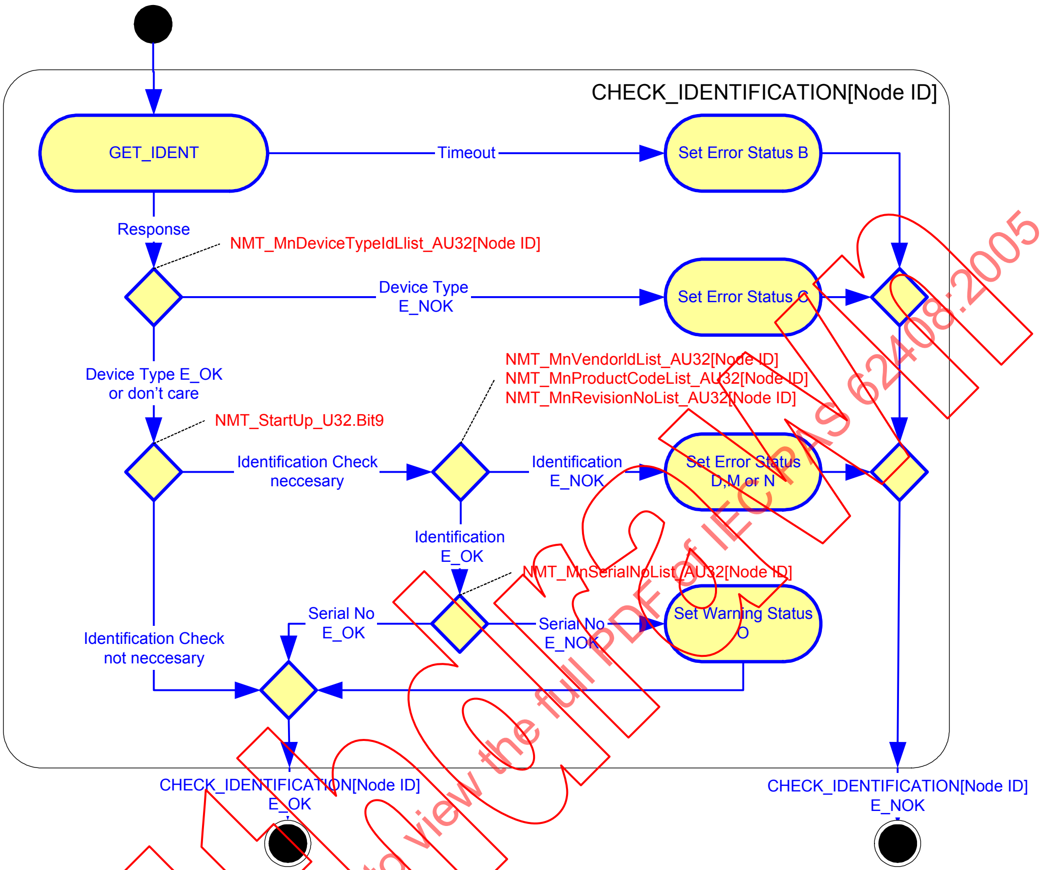


Figure 81 – Sub-state CHECK_IDENTIFICATION[Node ID]

The steps of the CHECK_IDENTIFICATION[Node ID] process are as follows:

- Request IdentResponse from the CN (see 7.4.3.2). In case of timeout, Error Status B is set and the CHECK_IDENTIFICATION[Node ID] state is finished with status E_NOK, otherwise continue with the next step.
- Examine the DeviceType of the CN depending on the following object:
  - NMT_MnDeviceTypeIdList_AU32[Node ID].

  If the examination of the DeviceType fails, the Error Status C is set and the CHECK_IDENTIFICATION[Node ID] state is finished with status E_NOK, otherwise the next step is implemented.
- If identification is not necessary (depending on the following object:
  - NMT_Start_Up_U32.Bit9),

  CHECK_IDENTIFICATION[Node ID] is finished with status E_OK, otherwise the CN identification is checked on the basis of the following objects:
  - NMT_MnVendorIdList_AU32[Node ID]
  - NMT_MnProductCodeList_AU32[Node ID]
  - NMT_MnRevisionNoList_AU32[Node ID]

  If the identification fails, the CHECK_IDENTIFICATION[Node ID] state is finished with status E_NOK and the Error Status D, M or N is set, otherwise the next step is implemented.
- The CN serial number is checked on the basis of the following object:
  - NMT_MnSerialNoList_AU32[Node ID].

If the identification fails, the Error Warning O is set and the CHECK_IDENTIFICATION[Node ID] state is finished with status E_OK, otherwise the CHECK_IDENTIFICATION[Node ID] state is finished with status E_OK.

7.6.2.3.1.2 CHECK_CONFIGURATION

The purpose of the CHECK_CONFIGURATION state is the examination of the configuration of a CN and update the configuration if necessary. The CHECK_CONFIGURATION [Node ID] returns with status E_OK, if the configuration fits, otherwise CHECK_ CONFIGURATION [Node ID] returns with status E_NOK.

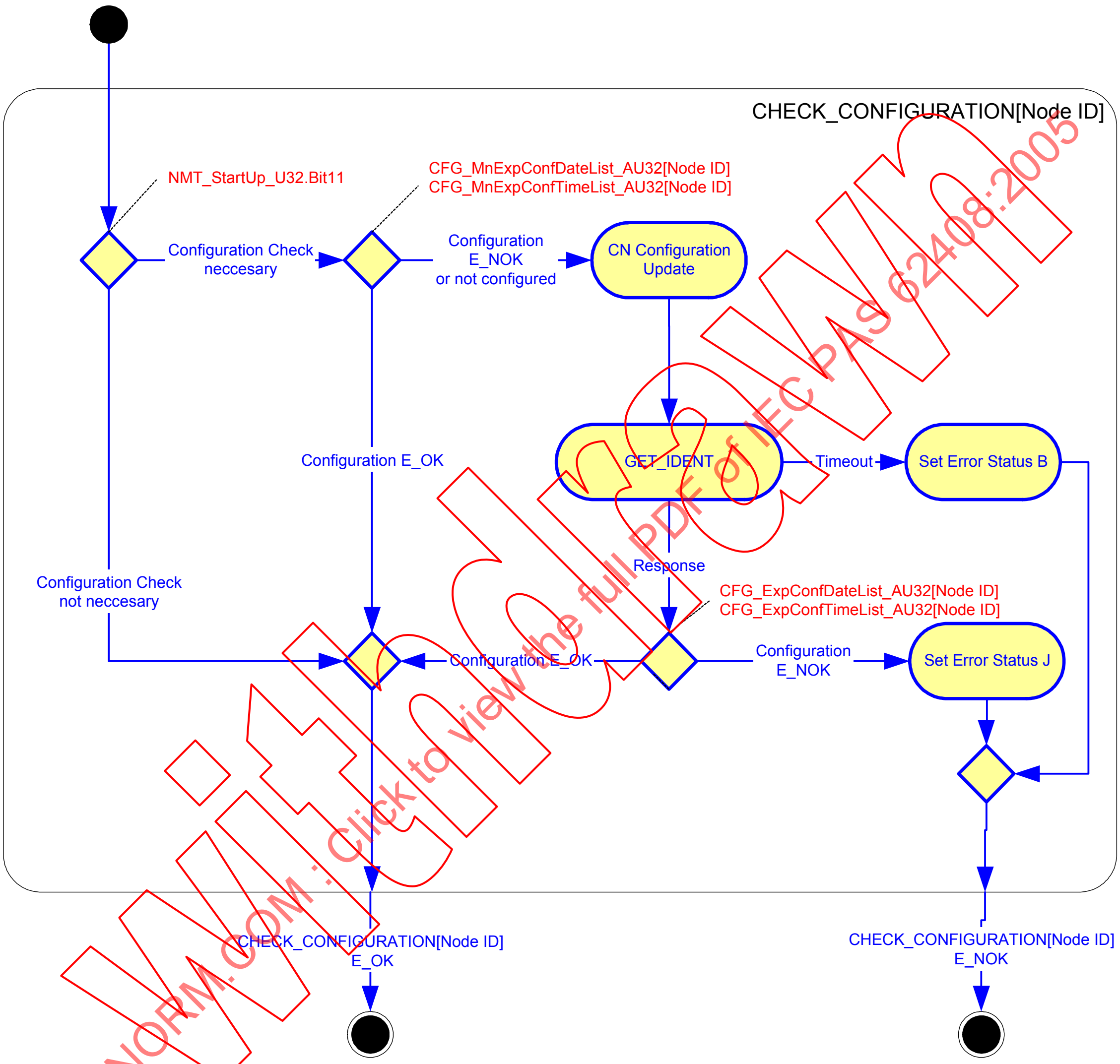


Figure 82 – Sub-state CHECK_CONFIGURATION[Node ID]

The steps of the CHECK_CONFIGURATION[Node ID] state are as follows:

- If the verification of the configuration is not necessary (depending on the following object:
  - NMT_StartUp_U32.Bit11),

  CHECK_CONFIGURATION is finished with status E_OK, otherwise configuration is checked on the basis of the following objects:
  - CFG_MnExpConfDateList_AU32[Node ID]
  - CFG_MnExpConfTimeList_AU32[Node ID]

  If the configuration date and time is E_OK, the CHECK_CONFIGURATION[Node ID] state is finished with status E_OK, otherwise continue with the next step.
- After configuration update, request IdentResponse from the CN (see 7.4.3.2). In case of timeout, Error status B is set and the CHECK_CONFIGURATION[Node ID] state is finished with status E_NOK, otherwise the next step is implemented.
- Configuration is checked on the basis of the following objects:
  - CFG_MnExpConfDateList_AU32[Node ID]
  - CFG_MnExpConfTimeList_AU32[Node ID]

  If the verification of the configuration fails, the Error status J is set and the CHECK_CONFIGURATION[Node ID] state is finished with status E_NOK, otherwise the CHECK_CONFIGURATION[Node ID] state is finished with status E_OK.

7.6.2.3.1.2.1 GET_IDENT

The purpose of the GET_IDENT state is the request of the IdentResponse from a CN. The GET_IDENT [Node ID] returns with status E_OK, if the CN answers within a timeout interval, otherwise GET_IDENT [Node ID] returns with status E_NOK. The timeout interval depends on the current NMT state of the MN.

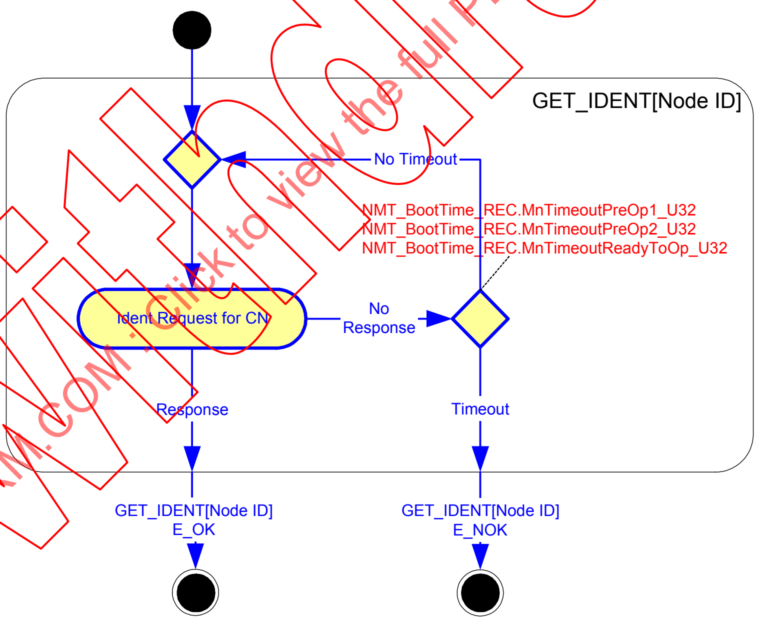


Figure 83 – Sub-state GET_IDENT[Node ID]

The steps of the GET_IDENT[Node ID] state are as follows:

- Request IdentResponse from the CN. In case of timeout, the GET_IDENT[Node ID] state is finished with status E_NOK. The timeout interval depends on the current state of the MN and the following objects:
  - NMT_BootTime_REC.MNTimeoutPreOp1_U32
  - NMT_BootTime_REC.MNTimeoutPreOp2_U32
  - NMT_BootTime_REC.MNTimeoutReadyToOp_U32
- If the CN answers with its IdentResponse, leave GET_IDENT[Node ID] with status E_OK.

7.6.2.4 NMT_MS_PRE_OPERATIONAL_2

The purpose of the NMT_MS_PRE_OPERATIONAL_2 state is to synchronize all configured CN to the isochronous EPL Cycle and examine the state of all CNs. The MN communicates with the CN using the isochronous EPL Cycle with SoC/SoA and PRes frames (see 4.2.4.1). After all mandatory CN reached the state NMT_CS_READY_TO_OPERATE, the MN also changes to the state NMT_MS_READY_TO_OPERATE.

In case of communication errors, the MN communicates the error to the application and shall remain in this state.

This state is entered from the NMT_MS_OPERATIONAL_1 (NMT_MT3) state.

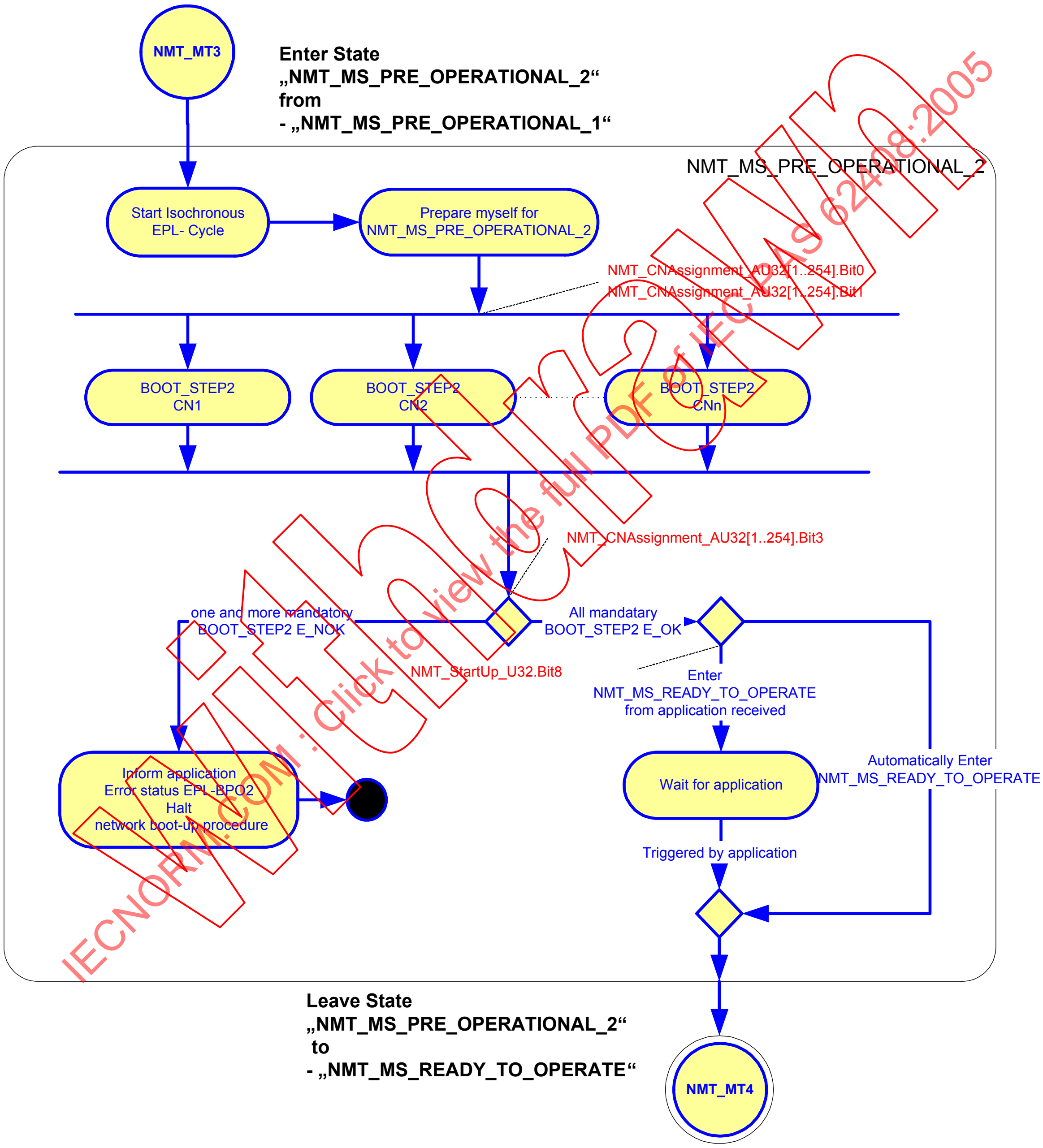


Figure 84 – Detail state NMT_MS_PRE_OPERATIONAL_2