INTERNATIONAL STANDARD

ISO 17356-5

First edition
2006-02-01

Road vehicles — Open interface for embedded automotive applications —

Part 5: OSEK/VDX Network Management (NM)

Véhicules routiers — Interface ouverte pour applications automobiles embarquées —

Partie 5: Gestion du réseau OSEK/VDX (NM)

Reference number
ISO 17356-5:2006(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 17356-5 was prepared by Technical Committee ISO/TC 22, *Road vehicles*, Subcommittee SC 3, *Electrical and electronic equipment*.

ISO 17356 consists of the following parts, under the general title *Road vehicles — Open interface for embedded automotive applications*:

— *Part 1: General structure and terms, definitions and abbreviated terms*

— *Part 2: OSEK/VDX specifications for binding OS,COM and NM*

— *Part 3: OSEK/VDX Operating System (OS)*

— *Part 4: OSEK/VDX Communication (COM)*

— *Part 5: OSEK/VDX Network Management (NM)*

— *Part 6: OSEK/VDX Implementation Language (OIL)*

0 Introduction

0.1 General

There is an increasing tendency for electronic control units (ECUs) made by different manufacturers to be networked within vehicles by serial data communication links.

Therefore, standardization of basic and non-competitive infrastructure in ECUs aims at avoiding the design of unnecessary variants and saving development time.

In the scope of OSEK/VDX cooperation, the Network Management system (NM) provides standardized features which ensure the functionality of inter-networking by standardized interfaces.

The essential task of NM is to ensure the safety and the reliability of a communication network for ECUs.

In a vehicle, a networked ECU is expected to provide certain features:

— each node accessible for authorized entities;

— maximum tolerance with regard to temporary failures; and

— support of network-related diagnostic features.

At a basic configuration stage, NM implementations complying with OSEK specifications are implemented in all networked nodes. This implies a solution for NM which can be implemented throughout the broad range of available hardware offered in today's ECUs.

Therefore, the status of the network is recorded and evaluated uniformly at all ECUs at intervals. Thus, each node features a determined behaviour as regards the network and the application concerned.

NM offers two alternative mechanisms for network monitoring:

— indirect monitoring by monitored application messages; and

— direct monitoring by dedicated NM communication using token principle.

However, the use of these mechanisms is up to the system responsible. Processing of information collected by these mechanisms is in accordance with requirements as regards to the entire networked system.

0.2 System status

In view of the application, NM comprises two standardized interfaces:

— Software: Application program <-> NM

— Network behaviour: Station <-> Communication medium

The resulting entire system is open. Thus, it can adapt to new requirements within the restrictions defined by the system design.

0.3 Remarks by the authors

This part of ISO 17356 describes the concept and the API of a Network Management, which can be used for ECUs in vehicles. It is not a product description which relates to a specific implementation.

General conventions, explanations of terms and abbreviations have been compiled in ISO 17356-1.

0.4 Summary

In order to achieve the essential task of a network monitoring, i.e. ensure safety and reliability of a communication network for ECUs, NM describes node-related (local) and network-related (global) management methods. The global NM component is optional. However, it requires a minimum local component to be operational.

Therefore, the following services are provided:

— initialization of ECU resources, e.g. network interface;

— start-up of network;

— providing network configuration;

— management of different mechanisms for node monitoring;

— detecting, processing and signalling of operating states for network and node;

— reading and setting of network- and node-specific parameters;

— coordination of global operation modes (e.g. network wide sleep mode); and

— support of diagnosis.

There are two main parts within the document: *Direct Network Management* described in Clause 2 and *Indirect Network Management* described in Clause 3. Both clauses describe the concepts, algorithms and behaviour.

Subclause 2.1, *Concept,* describes the fundamental aspects of the configuration management, the operating states and operating state management.

Subclause 3.3, *Algorithms and behaviour,* describes the protocol used for communication between nodes.

Clause 4 describes the *API (Application Programming Interface)* comprising the pure specification of the services offered for both direct and indirect NM. Input and output data, the functional description, particularities, etc. are described for each service. Furthermore, *System generation* services are described within this clause.

Clause 5 describes the impacts on the infrastructure of ISO 17356 and gives a brief description of all requirements for COM, OS and the data link layer for both direct and indirect NM.

Road vehicles — Open interface for embedded automotive applications —

Part 5: OSEK/VDX Network Management (NM)

1 Scope

1.1 Embedding of the Network Management (NM)

NM defines a set of services for node monitoring. Figure 1 shows how the NM is embedded into a system and that the NM shall be adapted to specific requirements of the bus system used or to the resources of the nodes.

NM consists of the following:

— interface to interact with the Application Programming Interface(API);

— algorithm for node monitoring;

— internal interfaces (NM <-> COM, etc.);

— algorithm for transition into sleep mode; and

— NM protocol data unit (NMPDU).

1.2 Adaptation to bus protocol specific requirements

Adaptation to bus protocol specific requirements consists of the following:

— CAN, VAN, J1850, K-BUS, D2B, etc.;

— error handling, e.g. bus-off handling in a CAN, transmission line error handling; and

— interpretation of the status information, e.g. overrun or error active/passive in a CAN.

1.3 Adaptation to node resources

Adaptation to node resources consists of the following:

— scaling of the NM as a requirement of the node; and

— application-specific usage of the NM services.

1.4 Adaptation to hardware-specific requirements

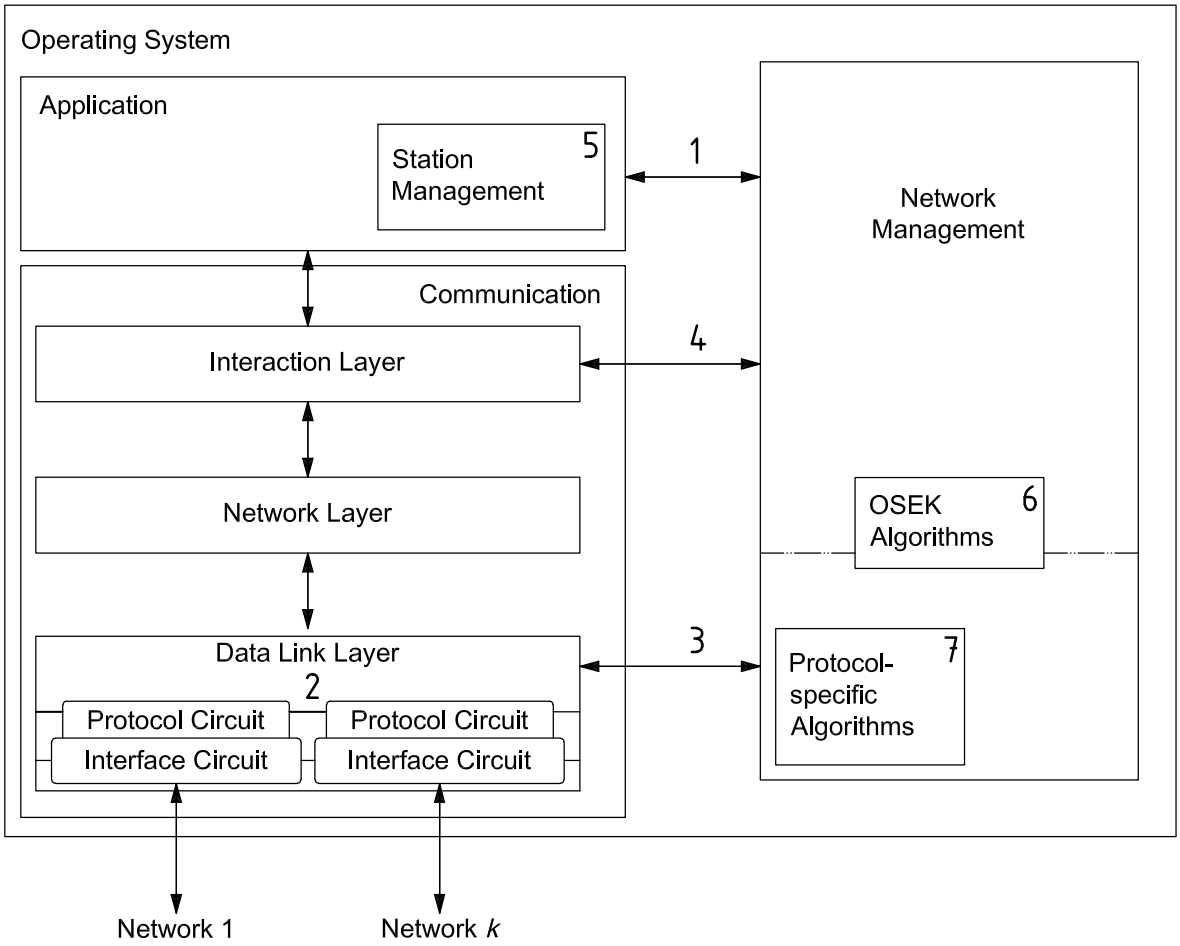
This consists of adaptation to a protocol circuit and a physical layer circuit, e.g. switching the bus hardware to one of the possible physically power save modes.

1.5 Station management (system-specific algorithms)

There are a variety of additional tasks involved in coordinating a network. These are not described in ISO 17356, since they are system-dependent. Hence, these tasks are done by the application, e.g. by a module called station management.

1.6 Philosophy of node monitoring

Node monitoring is used to inform the application about the nodes on the network. Thus, the application can check with the appropriate service if all stations required for operation are present on the network.



Key

1 API
2 several buses connected to one μController
3 interface to DLL, COM-specific, protocol-specific
4 interface to COM Interaction Layer
5 station management (outside the scope of ISO 17356)
6 algorithms
7 protocol-specific management algorithms

Figure 1 — Responsibility of interface and algorithms

2 Direct Network Management

2.1 Concept

2.1.1 Node monitoring

2.1.1.1 General

NM supports the direct node monitoring by dedicated NM communication. A node is a logical whole to which a communication access is possible. A microprocessor with two communication modules connected to two different communication media (e.g. low speed CAN and a high-speed CAN) represents two nodes from the NM point of view.

The rate of the NM communication is controlled across the network (minimization of bus load and consumption of resources) and the messages are synchronized (avoiding negative effects on application data by message bursts).

Every node is actively monitored by every other node in the network. For this purpose, the monitored node sends an NM message according to a dedicated and uniform algorithm.

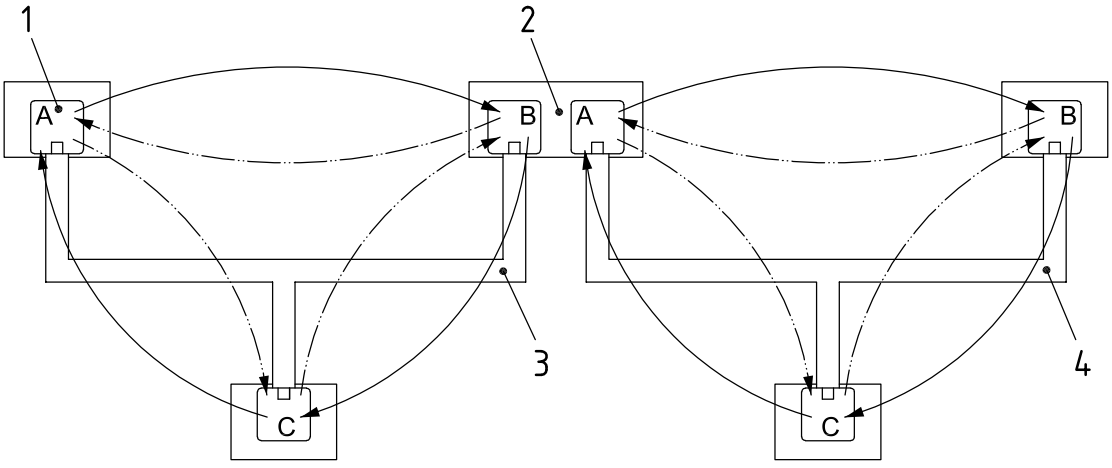
Direct node monitoring requires a network-wide synchronization of NM messages. For this purpose, a logical ring is used.

2.1.1.2 Logical ring

2.1.1.2.1 General

In a logical ring, the communication sequence is defined independently from the network structure. Therefore, each node is assigned a logical successor. The logically first node is the successor of the logically last node in the ring.

Thus, the decentralized control of the overall amount of NM messages is ensured and the bus load due to these messages is determined. The communication sequence of the logical ring synchronizes NM communication. Any node shall be able to send NM messages to all other nodes and receive messages from them.



Key

1 node
2 Electronic Communication Unit (ECU)
3 communication media 1
4 communication media 2

Figure 2 — Infrastructure of the NM (logical ring), example with two buses

2.1.1.2.2 Principle

The direct NM transmits and receives two types of messages to build the logical ring. An alive message introduces a new transmitter to the logical ring. A ring message is responsible for the synchronized running of the logical ring. It will be passed from one node to another (successor) node.

— Receive alive message: Interpretation as transmitter-related registration to the logical ring.

— Receive ring message: Interpretation as transmitter-specific alive signal and synchronization to initiate transmission of own NM message according to the logical ring algorithm.

— Time-out on ring message: Interpretation as transmitter-specific breakdown.

2.1.1.2.3 State of a node

A monitoring node is able to distinguish two states of a monitored node:

— node present → specific NM message received (alive or ring);

— node absent → specific NM message not received during time-out.

A monitoring node is able to distinguish two states of itself:

— present or not mute → specific NM message transmitted (alive or ring);

— absent or mute → specific NM message not transmitted during time-out.

2.1.2 Addressing

2.1.2.1 Status

The status of nodes and of the network shall be acquired and evaluated uniformly at intervals. For this purpose, all nodes shall communicate via their NM.

The NM communication is independent of the underlying bus protocol. Each node can communicate unidirectionally and address-related with any other node of the network. Therefore, individual and group addressing of nodes is required.

2.1.2.2 Node addressing

2.1.2.2.1 General

Address-related communication takes into account receiver and emitter. Each node has a unique identification which is known in the network.

Each address-related communication message contains certain data, the emitter identification and the receiver identification. NM does not specify the encoding of these components into selected bus protocols.

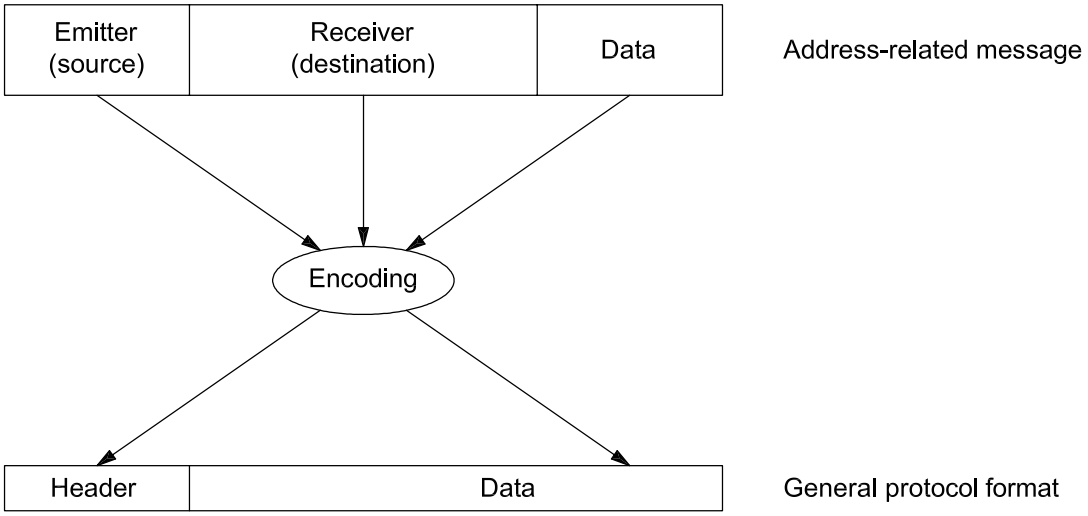


Figure 3 — Exemplary representation of encoding of an NM communication message onto a general protocol format

Individual addressing is implemented by node addressing using 1:1 connections. Group addressing is implemented by node addressing using 1:k connections ($k <$ number of nodes in the network). For this purpose, groups of receivers join group addresses.

2.1.2.2.2 Features of node addressing

These include the following:

— Each node is assigned a unique identification known within the whole network.

— Emitter and receiver identifications are explicitly included in the message.

— 1:k connections are implemented using group addresses.

— All messages are broadcast.

— Integrating a new node in an existing network does not require notification of the existing nodes.

2.1.3 NM infrastructure for data exchange

The NM supports the transfer of application data via its infrastructure (the logical ring). During the time delay between the reception and the transmission of the ring message, the application is able to modify the data.

It is possible for the application to specify and implement management algorithms which are not provided by NM.

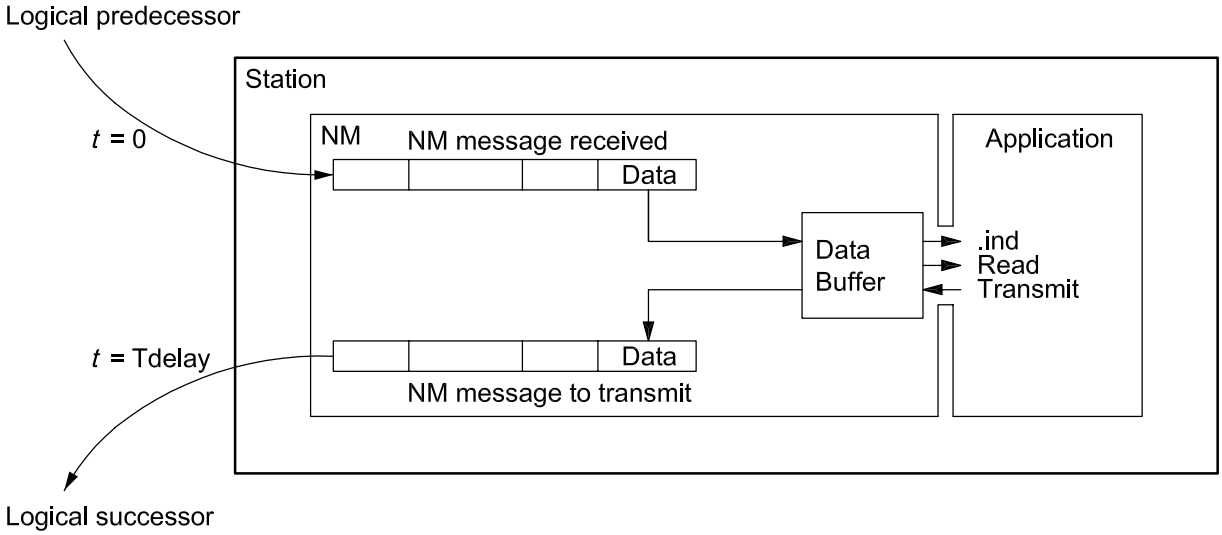


Figure 4 — Mechanism to transfer application data via the logical ring

2.1.4 Standard functionality

Initializations are performed with any system start (“cold start”), e.g. timer services required from the operating system or communication hardware via the data link layer interface.

Before the system is switched off, or switches off automatically, NM can be “shut down”, so that it can restore, e.g. to the previously known network history when the system is started up again.

The NM handles individual parameters, e.g. time-outs and node identifications and, if necessary, version numbers to identify hardware and software versions.

2.1.5 Configuration management

2.1.5.1 Network configurations

In the absence of any faults, the networked nodes are activated at different times, e.g.

— stations on terminal 30 (permanent power): Wake-up via external event;

— stations on terminal 15 (ignition): Switch ON via ignition key;

— stations with switch in supply line: switching ON and OFF at random, by driver.

However, the actual configuration is also altered by faulty nodes and by defects in the communication network. Consequently, different actual configurations can result for the individual nodes in the course of time, which are additionally subject to external influences, e.g. actions by the driver.

As a rule, each node wants to start its application as quickly as possible. In view of NM, this means that an actual configuration is made available to the applications as soon as possible. Finally, it is up to the application to decide whether to start communication immediately after it has become operable, or whether to wait until a minimum configuration is detected by NM.

NM distinguishes between:

— actual configuration: set of nodes to which access is possible; and

— limp home configuration: set of nodes which due to failure cannot participate in the logical ring.

Therefore, NM provides the following services:

— supply of the actual configuration;

— comparison of a configuration with a target configuration;

— indication of changed configuration.

2.1.5.2 Detection of a node in fault condition

2.1.5.2.1 Operability of a node

A node is considered operable, in terms of NM, if the node participates in the logical ring.

2.1.5.2.2 Detection of failures

Only a node which is expected to be operable on the network can be recognized as having failed. The application recognizes node failures by comparison to the previous knowledge regarding the target configuration. There are several possible ways by which the application can acquire this knowledge:

— the last stable state of the actual configuration;

— one or several programmed target configuration(s);

— the target/actual configuration determined by NM since system startup.

The NM recognizes its own node as having failed if it cannot send via the bus or if it cannot receive any messages from the bus, i.e. it is no longer operable.

Another node is considered as having failed if its NM message is not received or a NM message is received signalling an error state.

2.1.5.2.3 Reaction to a node failure

The NM of a node detecting a failure cannot distinguish whether the failed node is no longer able to communicate due to a line fault or due to a complete failure, without additional support. Any possible reactions, e.g. changeover to redundant physical paths, shall be specified together with entire system requirements.

2.1.5.3 Internal Network Management States

NM is specified in a hierarchical way. NM can enter the internal states listed below:

— NMOff: NM is shut off;

— NMOn: NM is switched on;

— NMShutDown: Selective shutoff of NM entity.

NMOn:

— NMInit: NM initialization;

— NMAwake: Active state of the NM;

— NMBusSleep: NM in sleep mode;

— NMActive: NM communication enabled;

— NMPassive: NM communication disabled.

NMAwake:

— NMReset: The operability of the own node determined;

— NMNormal: Processing of direct node monitoring;

— NMLimpHome: Handling of failure in own node.

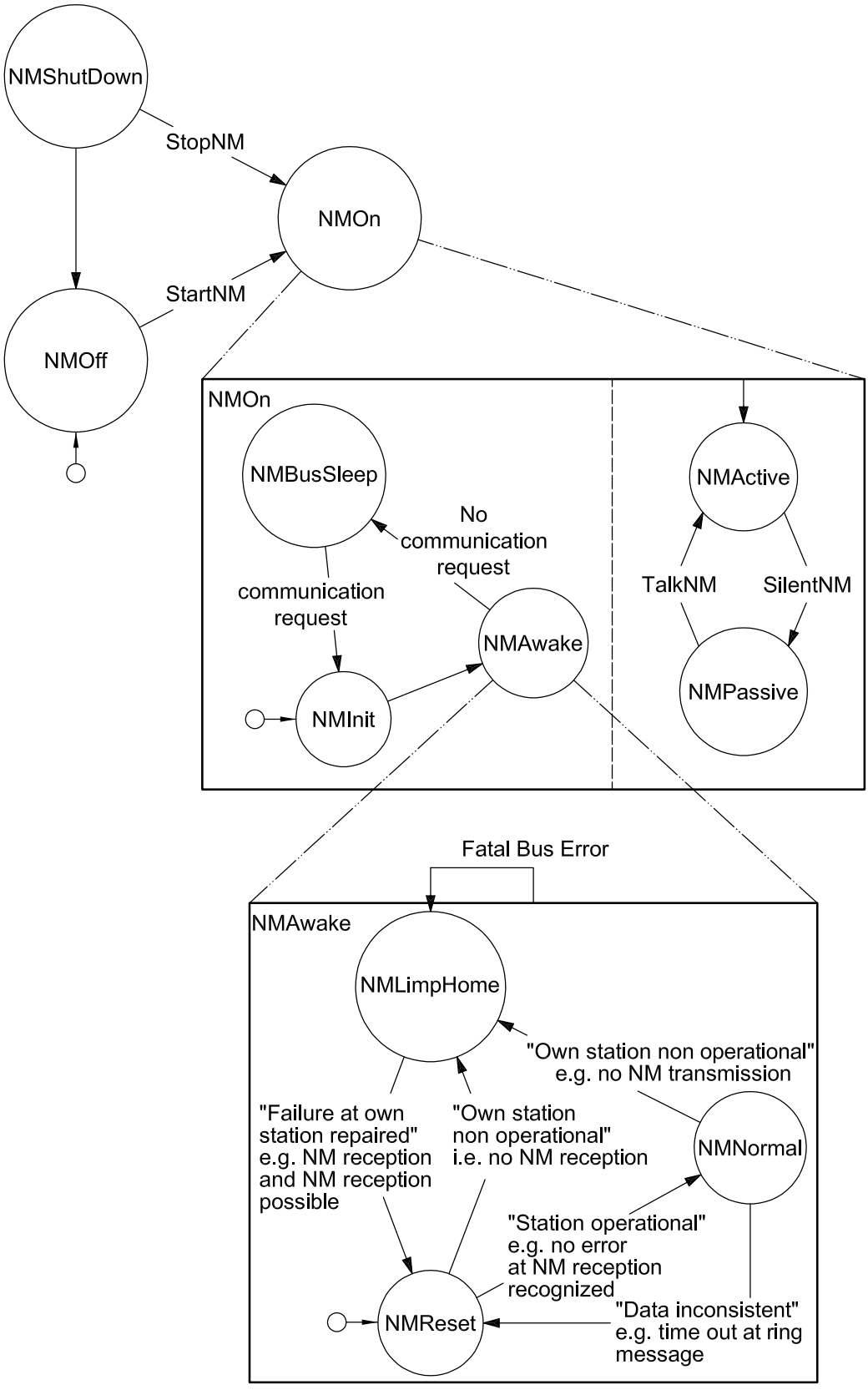


Figure 5 — Simplified state transition diagram of the direct NM

2.1.6 Operating modes

The NM does not manage application modes, but exclusively manages NM operating modes. NM distinguishes two main operating modes. The modes of the NM are directly mapped to internal NM states:

NMAwake (NMActive): In NMAwake, the node participates in NM communication (logical ring) and monitors all nodes with a NM in NMAwake.

NMBusSleep: If a node is in NMBusSleep, it does not participate in NM communication. Depending on the hardware integrated in the networks, nodes can switch into NMBusSleep simultaneously.

The NM provides services for:

— adjustment of NM operation modes; and

— indication of NM operating modes.

2.1.7 Network error detection and treatment

Only a limited part of the network activities is “visible” for the NM to detect errors.

The problem with error detection is that many errors appear identical from the node’s point of view:

— The fact that a node on the network is not transmitting messages may be due to various reasons:

  — it may be due to a control unit which has failed completely, or which has not been installed;

  — the communication module or the bus driver may be defective; or

  — bus lines may have been disconnected or the connector may be defective.

— Great interest is attributed to any information which helps detect the cause of an error clearly, so as to enable replacement or repair of the faulty component or to initiate an NMLimpHome.

— Most errors occur during the course of assembly of the network during production and after repairs. If connectors are interchanged or contacts are pushed back, this will have fatal consequences for the network. Lines which are laid incorrectly, e.g. directly along components with sharp edges, can also cause operating malfunctions within the network.

2.1.8 Support of diagnostic application

The NM supports the diagnostic application in the ECU by providing on request:

— status information of NM;

— configuration information acquired.

The NM is not responsible for recording the error history.

2.2 Algorithms and behaviour

2.2.1 Communication of the Network Management system

2.2.1.1 NM protocol data unit (NMPDU)

2.2.1.1.1 General

Any NM message contains the NM protocol data unit (NMPDU). The NMPDU defined hereafter represents the NM data to be communicated in order to control NM performance.

In order to fulfil all requirements with regards to communication and NM the NMPDU contains the following elements:

— NM address field:

  — source Id,

  — destination Id;

— NM control field:

  — OpCode,

— NM data field (optional):

  — application specific data.

NM does not define network addresses. This parameter is dedicated to specific system design and therefore is the responsibility of the respective system developer.

Table 1 — NMPDU – the representation of the data is not fixed

Address field		Control field		Data field
Source Id	Dest. Id	OpCode		Data
Mandatory				Optional
		Res	Ring Message (4 types)	
			Alive Message (2 types)	
			Limp Home Message (2 types)	

To guarantee the interoperability, the data representation and the NMPDU encoding and decoding algorithms shall be fixed. It is necessary to initialize the reserved area of the OpCode for future expansions. Whenever a Network Management message is received and transmitted after T_{Typ}, the reserved part of the OpCode is copied to the transmitted message.

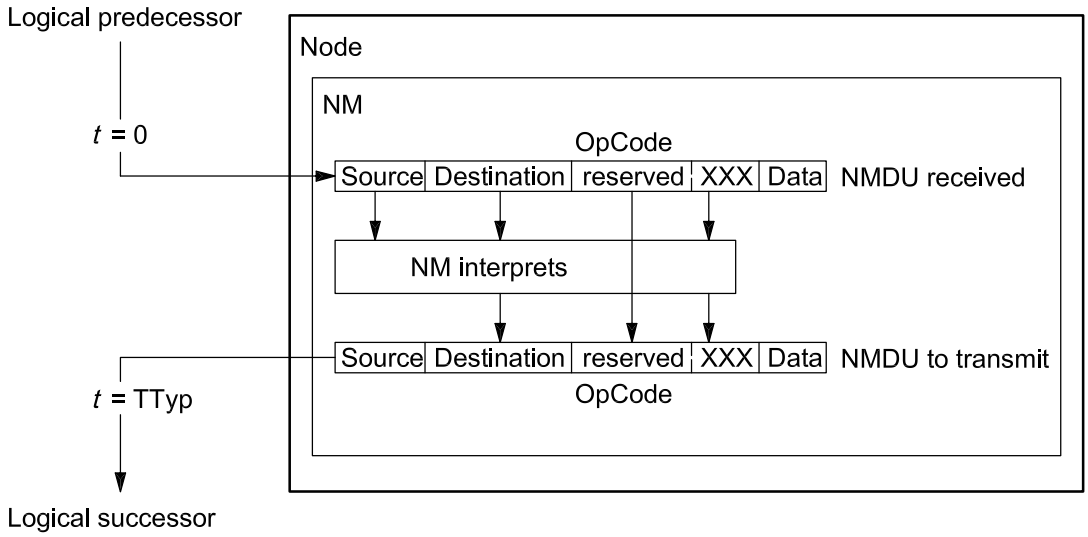


Figure 6 — NM actions with the reserved area of the OpCode (XXX encoding of NM message types)

2.2.1.1.2 Data consistency

The NM guarantees the data consistency of the NMPDU, e.g. during the reception of a burst of NMPDUs. The overrun of complete NMPDUs is possible.

2.2.1.1.3 NMPDU length

NM neither fixes the length of the NMPDU nor determines whether the data length is static or dynamic. Dynamic means that the length of the user data may change from NM message to NM message without affecting the specified algorithms.

2.2.1.2 Addressing mechanisms used by the NM

Each node in the network is assigned a global identification known by all nodes within the entire network.

NM communication is performed by directional communication of NM messages using 1:1 connections. The communication sequence complies with the definition of the logical ring in the respective network.

Therefore, node addressing mechanisms are used for NM communication. NM protocol data units shall include global identifications of source and destination among other data.

These identifications are transferred into address-related NM messages. Each node transmits NM messages with its global node identification and addresses the receiver by specifying its global node identification, e.g. in the message header or in the data field.

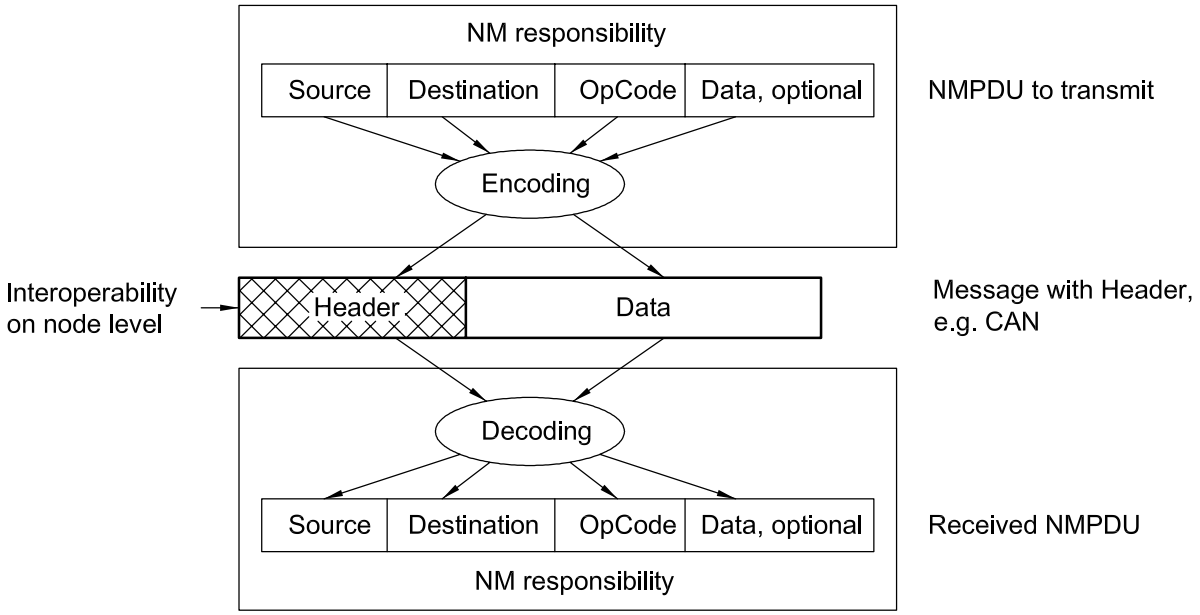


Figure 7 — Encoding/decoding of the NMPDU to/from a message on the bus

Examples for mapping node identifiers into address-related NM messages are given in Annex A.

In order to simplify the handling of that amount of similar communication objects for NM communication, the data link layer shall provide the interface of a window communication mechanism. The window mechanism shall be defined by a WindowMask and an IdBase. However, the window mechanism shall be implemented by the respective NM system responsible.

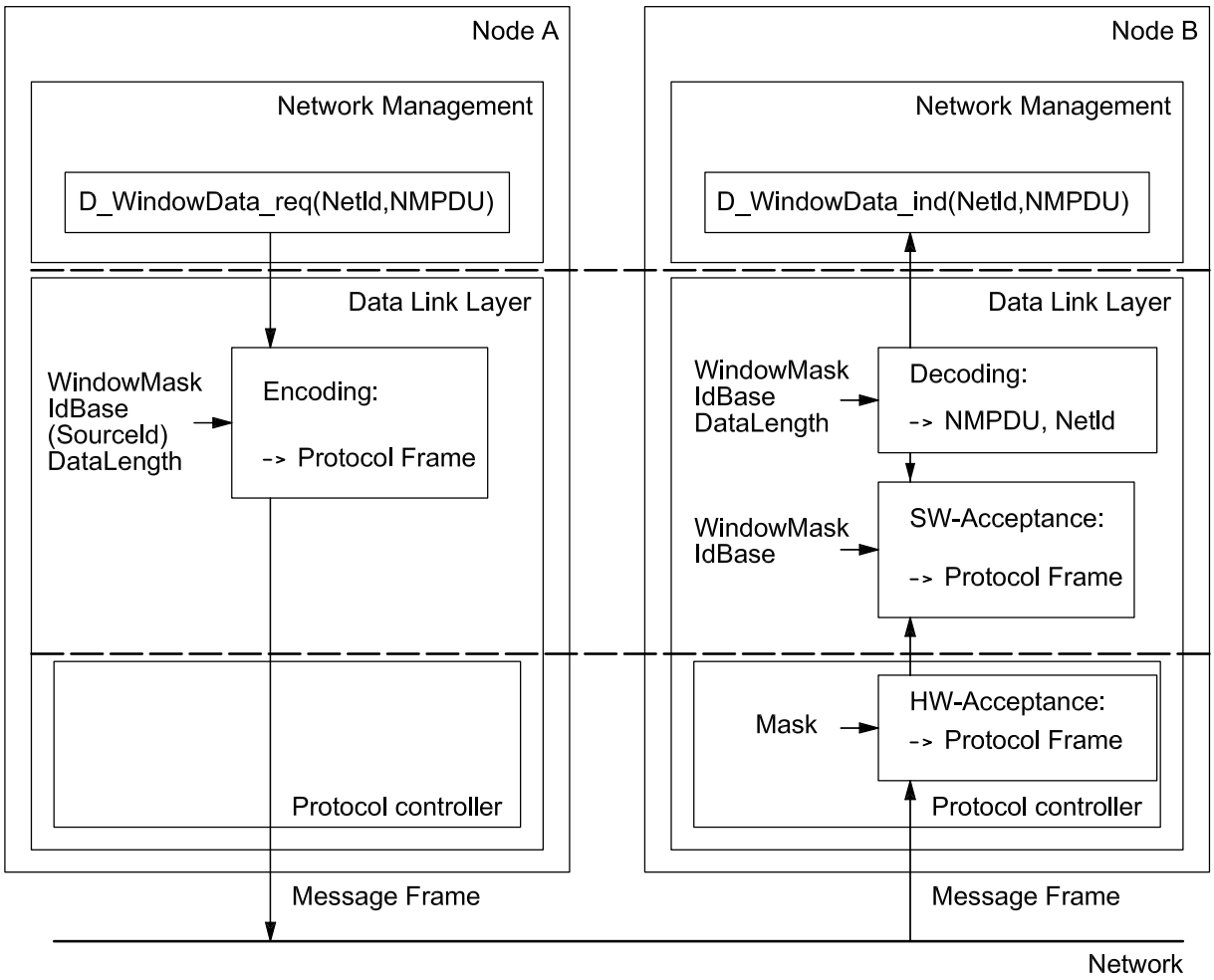


Figure 8 — Transmission and reception of NM protocol data units (NMPDU)

NOTE It depends on the system generation functionality whether the parameter DataLength is static and located inside the DLL or whether it is dynamic and located inside NM.

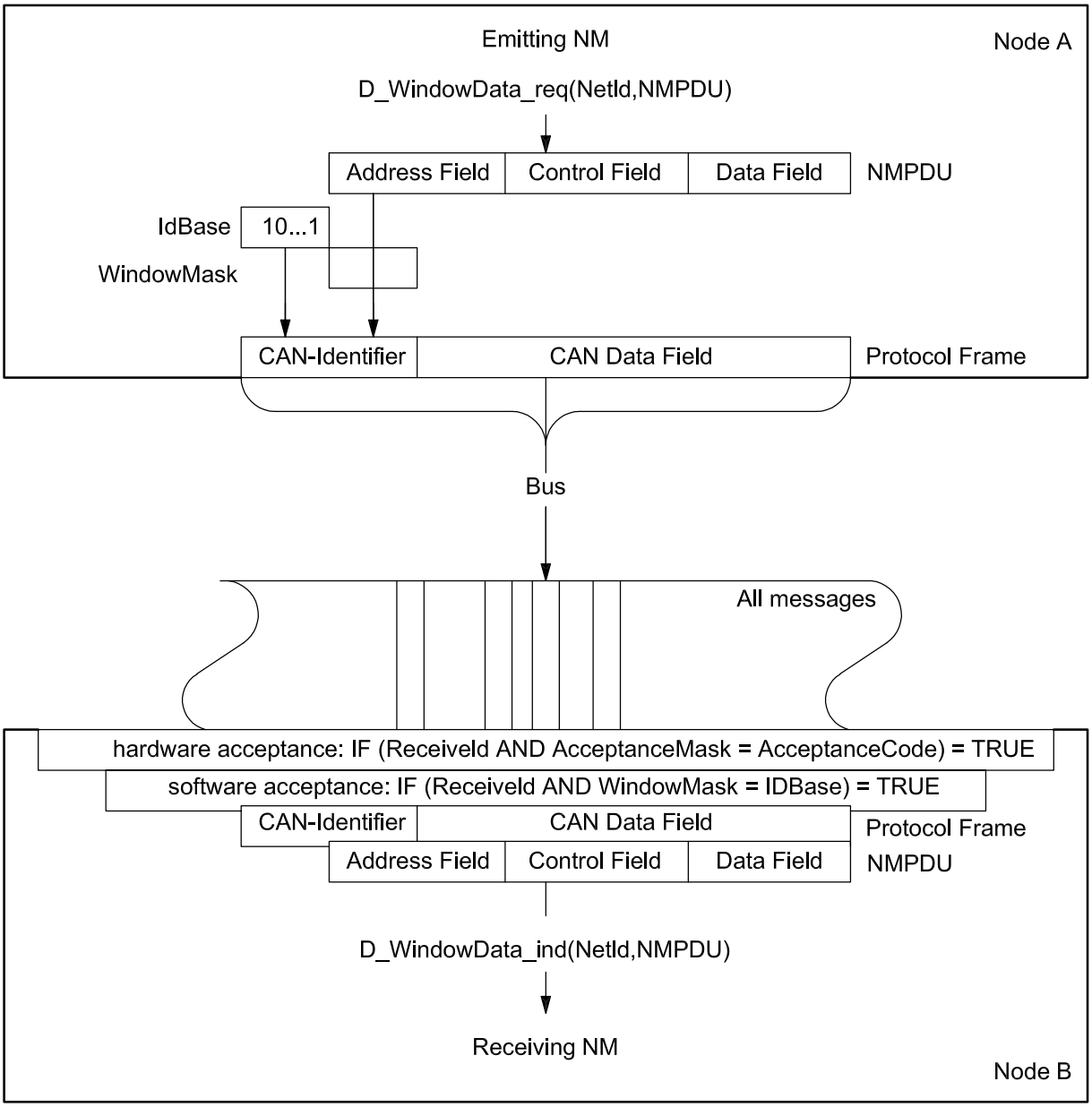


Figure 9 — CAN, example for the transmission and reception mechanisms of a NMPDU

The CAN identifier consists of two parts:

— a fixed IdBase; and

— some bits of the address field, chosen by a mask.

2.2.2 NM infrastructure for data exchange

2.2.2.1 General

The NM does not monitor the contents of the NMPDU data field. Every received ring message will be indicated to the application. The data field will be copied immediately into the buffer. The buffer will be copied into the data field, when the ring message shall be passed to the logical successor.

2.2.2.2 Data consistency

The NM uses the following mechanisms to guarantee the data consistency:

— The application can modify the ring data only between the reception of a ring message from the logical predecessor and the emission of the ring message to the logical successor.

— The NM allows the access to the ring data only if the logical ring runs in a stable state. The logical ring runs stable if the configuration does not change and there is no NM message during the allowed access time of the application to the ring data.

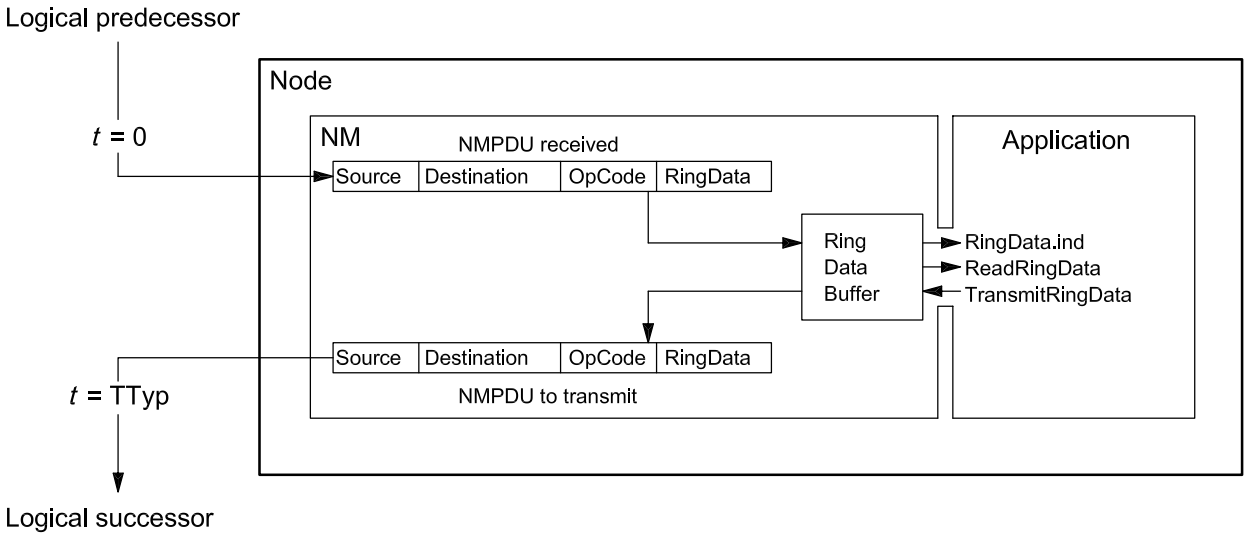


Figure 10 — Handling of data exchange between NM and application

2.2.3 Standard tasks

2.2.3.1 NM parameters

All NM parameters introduced in the concept description are known at compile time for a specific implementation and stored in the ROM of all ECUs.

Table 2 — NM parameters

NM parameter	Definition	Valid area
NodeId	Relative identification of the node-specific NM messages	Local for each node specific
T_{Typ}	Typical time interval between two ring messages	Global for all nodes
T_{Max}	Maximum time interval between two ring messages	Global for all nodes
T_{Error}	Time interval between two ring messages with NMLimpHome identification	Global all nodes
$T_{waitBusSleep}$	Time the NM waits before transmission in NMBusSleep	Global all nodes
T_{Tx}	Delay to repeat the transmission request of a NM message if the request was rejected by the DLL	Local for each node specific

To ensure the implementation of open and adaptive systems, all parameters of each node should be stored in a non-volatile, but erasable and writeable memory.

These can thus be adapted whenever required, e.g. by a diagnostic node. As regards transfer of parameters, reference is made to a specific download mode which is not dealt with in implementation specific system definitions.

2.2.3.2 Network status

The NM informs the application on request about the network status it has acquired. The interpretation of these data is system-specific and therefore with the application.

NM implementation should comply with minimum requirements to memory size which enables representation and storage of the network state, can appear as shown in Table 3.

Table 3 — Encoding of the network status

Network status	Interpretation
Present network configuration stable[a]	0 No 1 Yes
Operating mode of network interface	0 No error[b] 1 Error, bus blocked[c]
Operation modes	0 NMPassive 1 NMActive
	0 NMOn 1 NMOff
	0 No NMLimpHome 1 NMLimpHome
	0 No NMBusSleep 1 NMBusSleep
	0 No NMTwbsNormal and no NMTwbsLimpHome 1 NMTwbsNormal or NMTwbsLimpHome
	0 Using of Ring Data allowed 1 Using of Ring Data not allowed
	0 Service GotoMode(Awake) called 1 Service GotoMode(BusSleep) called

[a] Configuration did not change during the last loop of the NM message in the logical ring.

[b] Reception and transmission of NM messages were successful.

[c] CAN-busoff is an example.

2.2.3.3 Extended network status

The extended network status is specific to the user.

Table 4 — Example for the encoding of the extended network status

Extended network status	Interpretation
Operating mode of network interface	00 No error[a] 01 Error, communication possible[b] 10 Error, Communication not possible[c] 11 Reserved
Number of nodes which participate in the monitoring algorithm “logical ring”	Up to the user
Number of nodes which signal its limp home	Up to the user
Time since the logical ring is in a stable state	Up to the user
Time since the logical ring is in a dynamic state	Up to the user

[a] Reception and transmission of NM messages were successful.

[b] Communication was via one wire.

[c] CAN-busoff for a “long” time is an example.

2.2.4 Configuration management

2.2.4.1 General

Direct node monitoring is based on decentralized configuration management. The respective procedures are described by one state transition diagram. The NM algorithm for decentralized configuration management can be used for:

— regular NM communication, i.e. transmission of ring messages according to the communication sequence; and

— exceptional NM communication, i.e. startup and limp home/failure modes.

2.2.4.2 Timing reference

Implementation of decentralized communication management requires several timing criteria to be respected. To the resulting time intervals a relatively high jitter may be applied in the individual nodes.

In order to minimize the negative effect on user software, NM shall not require any sharp timing criteria in general. The following timing criteria apply with NM implementations:

— T_{Typ} typical interval between two ring messages on the bus;

— T_{Max} maximum admissible interval between two ring messages on the bus;

— T_{Error} cycle time in which a node signals that an error has occurred;

— T_{Tx} delay to repeat the transmission request of a NM message if the preceding request was rejected.

2.2.4.3 Monitoring counter

To determine if a node is operational, the writing path and the reading path of the node should be checked explicitly by the NM.

This is accomplished most easily by indirect mechanisms, using monitoring counters which are incremented or decremented by different events. Their states — contents greater or less than the predefined limits — are considered as information pertaining to the node's readiness for operation.

2.2.4.4 State transition diagram

From the point of view of the application, the basic states of NM are:

— **NMReset:** In NMReset, the node notifies its presence once in the network. For that purpose, the alive message is transmitted. The NM then changes immediately over to NMNormal.

— **NMNormal:** In NMNormal, the NM tries to pass one ring message cyclically with T_{Typ} from one node to another. If a node is unable to receive or transmit any NM messages, it switches over into NMLimpHome.

— **NMLimpHome:** In NMLimpHome the NM signals its limp home status by a limp home message cyclically with T_{Error} and repetitively until it is able to transmit its own ring message to the bus and until it is able to receive NM messages of other nodes correctly.

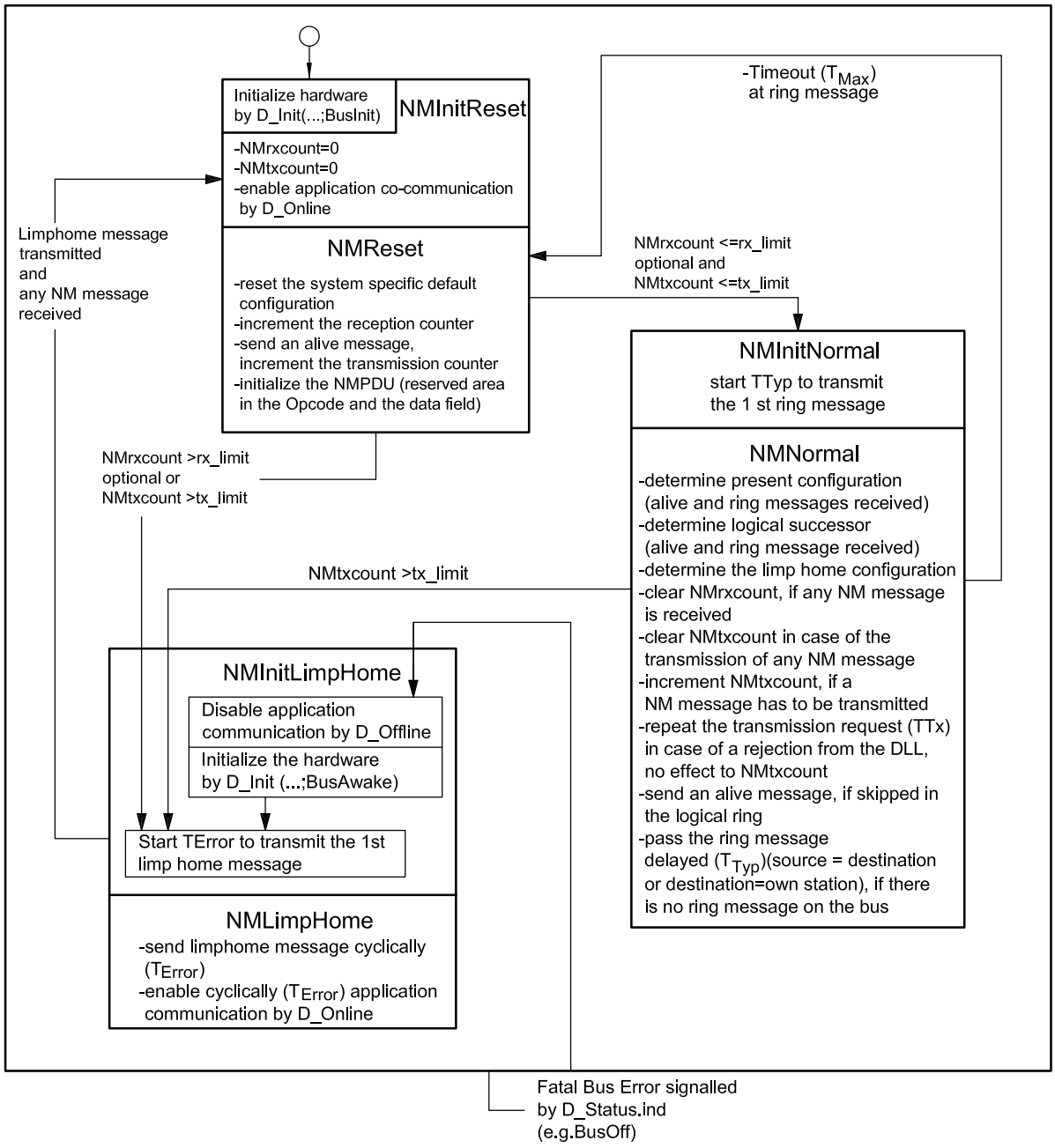


Figure 11 — State transition diagram of the NM algorithms for initialization, start-up and monitoring of a network (logical ring and limp home)

Some hints include:

— Time-out TMax: In case of ring messages: another node in the logical ring has disappeared.

— NMrxcount: This counter is used to detect a failure at the receive functionality of the NM.

— NMtxcount: This counter is used to detect a failure at the transmit functionality of the NM.

— Enter NMLimpHome: This state is entered, if NMtxcount or NMrxcount is greater than system specific limits (rx_limit, tx_limit). Typical value for rx_limit is 4 and a typical value for tx_limit is 8.

— Leave NMLimpHome: This state is left, if the receive functionality and the transmit functionality is always available for the NM.

— Node skipped: If a node is skipped, it transmits an alive message asynchronously.

— System-specific default configuration: “I am present at the network and I am my own logical successor”

— Start-up of the logical ring: By entering the state NMNormal every node starts the alarm T_{Typ}.

— Registration of a node: Alive messages and ring messages are used to introduce a node in the network.

— Delay TTx: A transmit request can be rejected by the lower communication layer and shall be repeated with a delay.

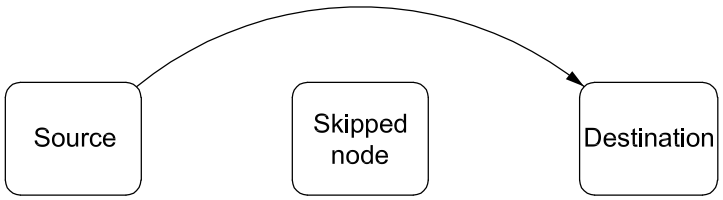


Figure 12 — Skipped in the logical ring

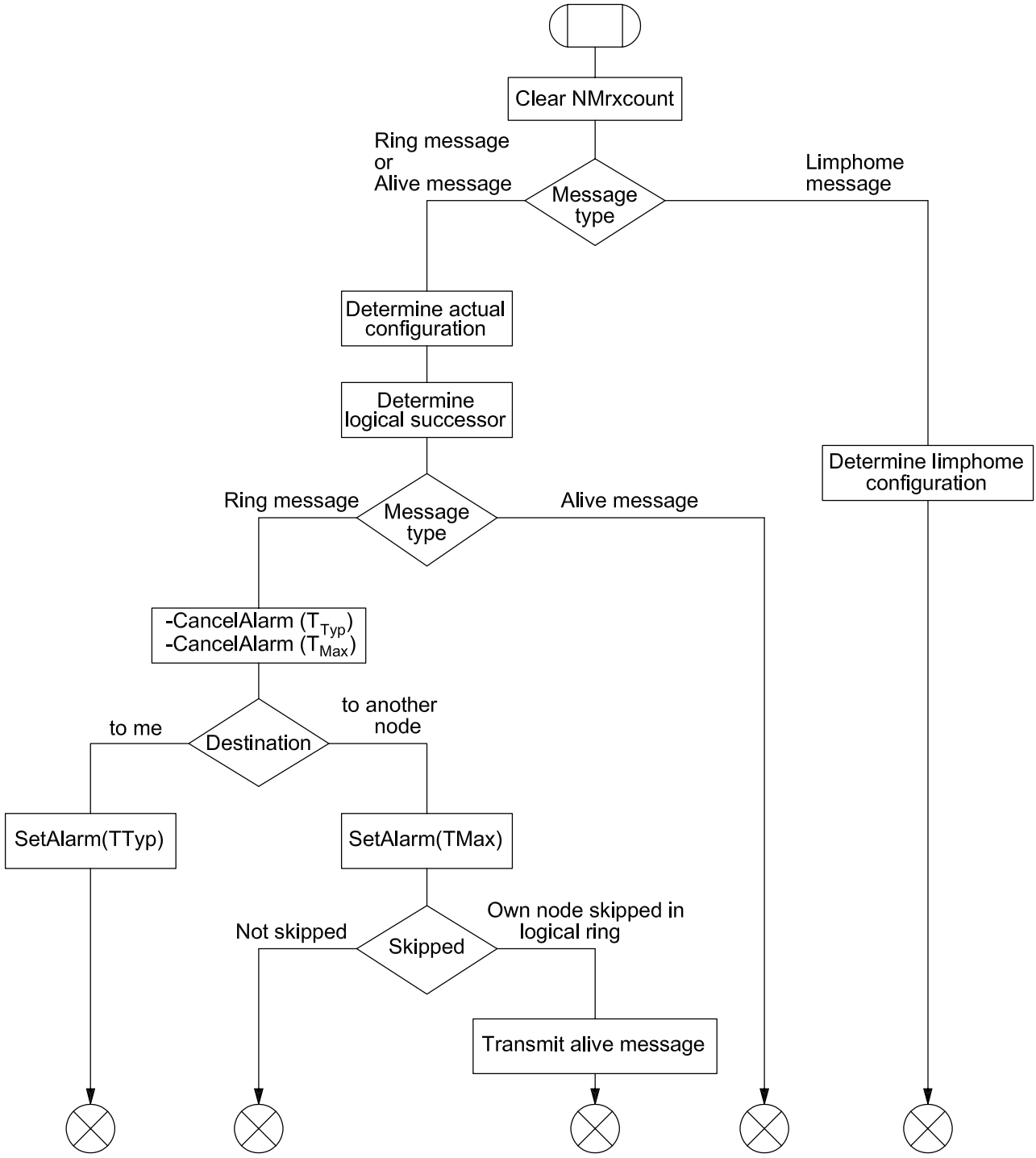


Figure 13 — Actions during NMNormal in case an NM message is received "at a time"

During the establishment of the logical ring, NM transmits and receives alive messages and ring messages from the network interface.

Starting with a stable NM communication in the logical ring the management of two configuration failures — dynamic introduction of a "new" node in the NM communication (node no. 3); and failure condition of a node leading to its disappearance from the logical ring (node no. 1) — are shown in Figure 14.

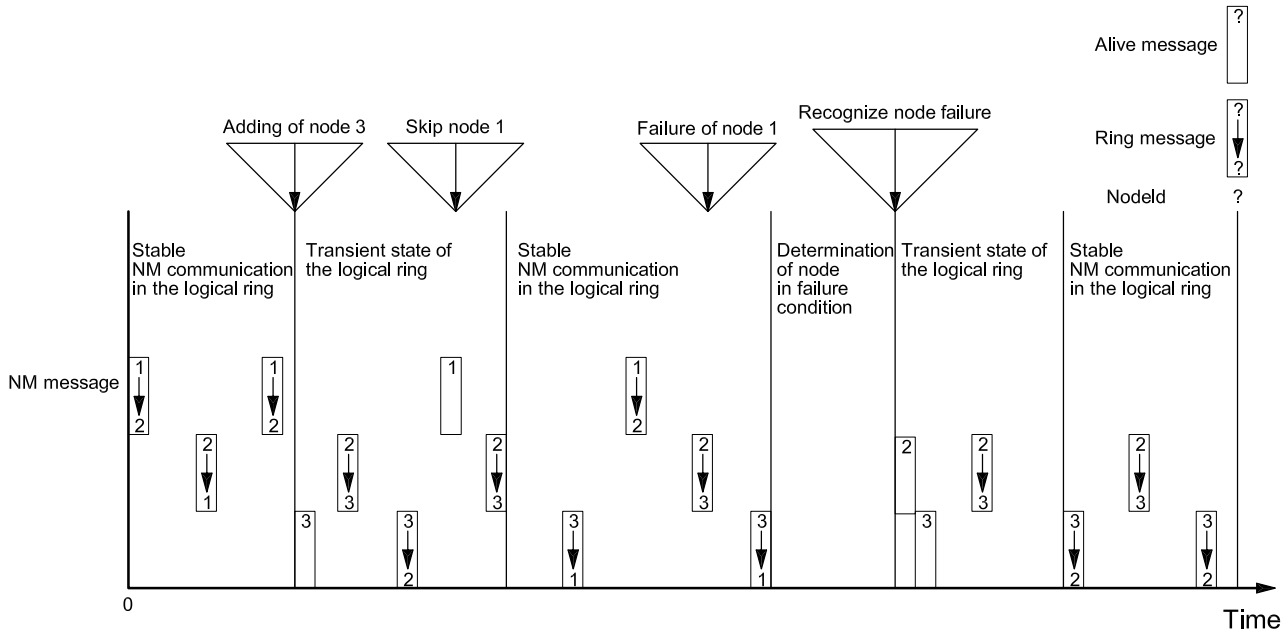


Figure 14 — Regeneration principle of decentralized configuration management as a basis for NM communication in the logical ring

2.2.4.5 Particularities regarding implementation

2.2.4.5.1 The emitting of a message is not interruptible

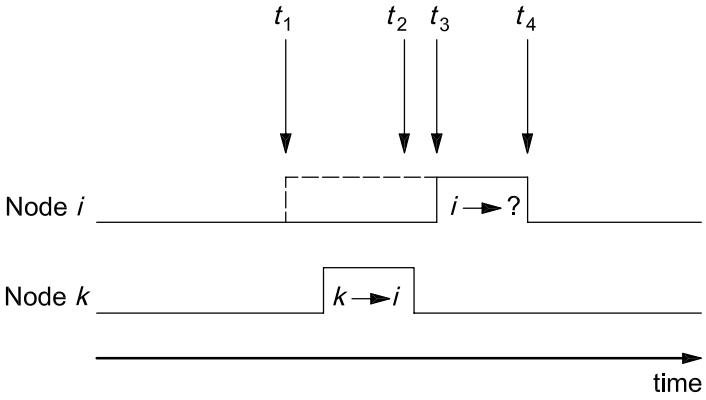
During normal operation, a ring message shall be transmitted or passed with a delay unless another ring message has been received during the delay.

Due to particularities of some asynchronous protocol implementations, this task cannot be executed directly in line with the verbal statement.

In view of node i, there is no way of preventing an external ring message being received which really prohibits the transmission of the node's own ring message between the decision to send the ring message of its own and the actual transmission.

This effect is only critical if the external ring message received is destined to node i. In this case, two ring messages can be maintained permanently, as exactly the same constellation may occur at the logical successor.

Figure 15 shows a constellation of ring messages which enables the simultaneous occurrence of two ring messages without specific measures.



Key

t_1 The timer T_{Typ} in node i has elapsed and the ring message of node i is released for transmission. As the bus is busy, this ring message cannot be transferred.

t_2 Node i receives the respective ring message from node k.

t_3 The ring message of node i is transmitted to the bus.

t_4 The ring message of node i was transmitted to the bus successfully.

Figure 15 — Ring messages from the nodes i and k on an asynchronous bus

Node i would really pass the ring message received at t_2 with a delay of T_{Typ}. However, in this case, it would have to terminate the ring message requested at t_1 which has not yet been emitted. This is not possible in most cases.

To avoid two simultaneous ring messages occurring at the same time, each node shall ignore a ring message addressed to it between the moments t_1 and t_4.

2.2.4.5.2 Timer structure in the state "NMNormal"

The timers T_{Typ} and T_{Max} are set, reset and cancelled for supervision of the NM communication.

The applicability of alarm services SetAlarm and CancelAlarm is assumed.

Table 5 — Timer actions in NMNormal, during various bus actions

	T_{Typ}		T_{Max}	
	SetAlarm	**CancelAlarm**	**SetAlarm**	**CancelAlarm**
Ring message received	-	✓	✓	✓
Addressed by ring message or source equal destination	✓		-	
Ring message transmitted	-	✓1)	✓	✓
Transition from NMReset to NMNormal	✓	-	-	-
NOTE A duplicated ring is avoided.				

This application fulfils the bus-specific requirement to avoid several ring messages. The table shows the activities of the timers in NMNormal. Individual timer requests are terminated abnormally and/or set as required by the bus activities detected. In this context, the fact that a duplicate ring is avoided is of particular interest. Between the moment when the decision to pass the node's own ring message is made and the

moment when it is actually transmitted, any additional request to pass the ring message shall be ignored. So, if the request T_{Typ} is cancelled as a precautionary measure whenever its own ring message is transmitted, this task is accomplished with minimum effort.

Processing a timer request only necessitates triggering two actions in NMNormal. Timer T_{Typ} is responsible for passing the ring message, whereas timer T_{Max} monitors the cyclic occurrence of the ring messages; it serves to detect a general configuration error.

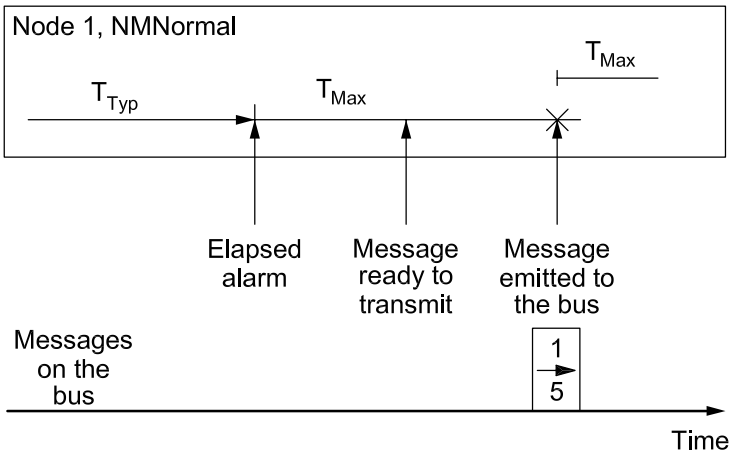
Table 6 — Main actions which are triggered by an expired timer in NMNormal.

	T_{Typ} elapses	T_{Max} elapses
Send ring message	✓	-
Go to NMReset	-	✓

2.2.4.5.3 How two ring messages are prevented

The NM was specified on the base of a broadcast channel and a serial bus protocol. Therefore, every node receives every NM message at nearly the same time. NM adjustments are overwritten by a received NM message; NM messages are handled with time priority.

One of the basic principals of the NM is the synonym between an elapsed T_{Typ} alarm and the emission of a regular ring message to a logical successor. The specified algorithms guarantee that a running T_{Typ} alarm exists always in only one node inside the whole network. A received regular ring message retriggers in the addressed node (destination of the message) the T_{Typ} alarm and cancels in all other nodes the T_{Typ} alarms.

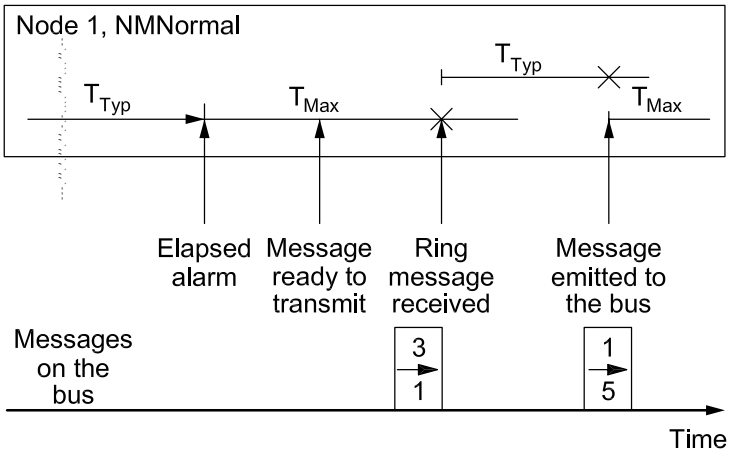


Passing of a ring message during the fixed state of the logical ring

Node 1 received a regular ring message and set a T_{Typ} alarm to pass the ring message.

When T_{Typ} elapses, the alarm T_{Max} is set and the regular ring message is prepared for transmission.

After emitting the prepared regular ring message on the bus, the running alarm T_{Max} is retriggered.

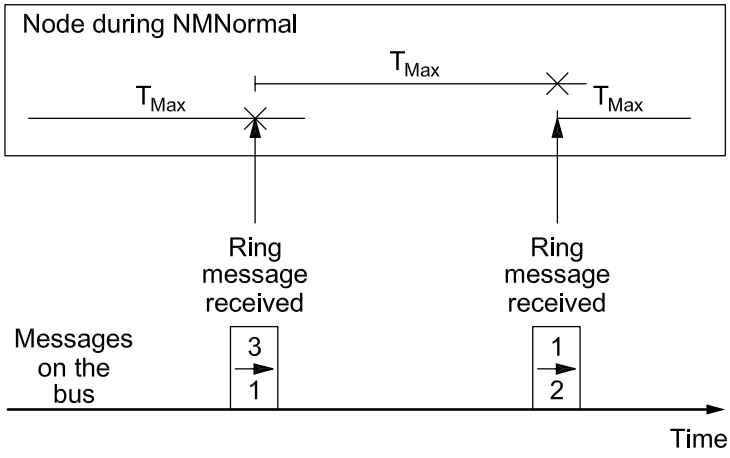


Passing of a ring message during the dynamic state of the logical ring — mechanism to avoid two ring messages

Node 1 received a regular ring message and set a T_{Typ} alarm to pass the ring message.

When T_{Typ} elapses, the alarm T_{Max} is set and the regular ring message is prepared for transmission.

A regular ring message was received and a T_{Typ} alarm was set while preparing the emission of the regular ring message. The emitted regular ring message now retriggers the T_{Max} alarm.



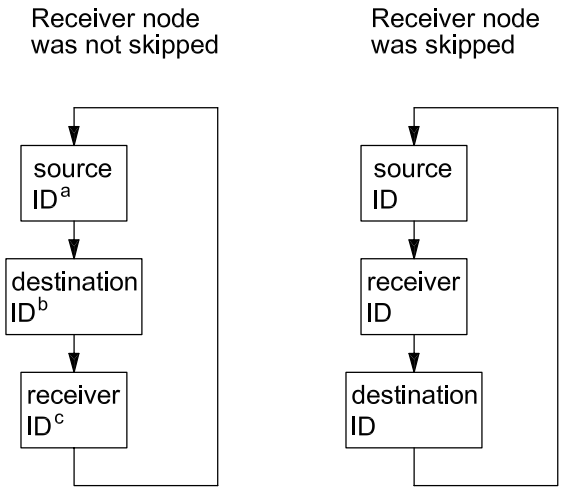
Monitoring of ring messages during the fixed state of the logical ring

Every received regular ring message retriggers a set T_{Max} alarm.

Figure 16 — Examples for mechanisms to synchronize the NM alarms and their effects on the behaviour of the NM

2.2.5 Example — Skipped in the logical ring

Every node is able to define a temporary logical ring in case of the reception of a ring message to any node in the network. The ring is given by the identifications of the receiver node, the source node of the message and the addressed destination node.



a Transmitter of the ring message.

b Addressed node.

c Receiver of the ring message.

Figure 17 — Temporary logical ring for the test, whether the receiver node has been skipped or not

By arranging the node identifications in a numerical order, one will get:

SDR	(Source →)	<	(→Destination)	<	(Receiver)	✓
RSD	(Receiver)	<	(Source →)	<	(→Destination)	✓
DRS	(→Destination)	<	(Receiver)	<	(Source →)	✓
DSR	(→Destination)	<	(Source →)	<	(Receiver)	skipped
RDS	(Receiver)	<	(→Destination)	<	(Source →)	skipped
SRD	(Source →)	<	(Receiver)	<	(→Destination)	skipped

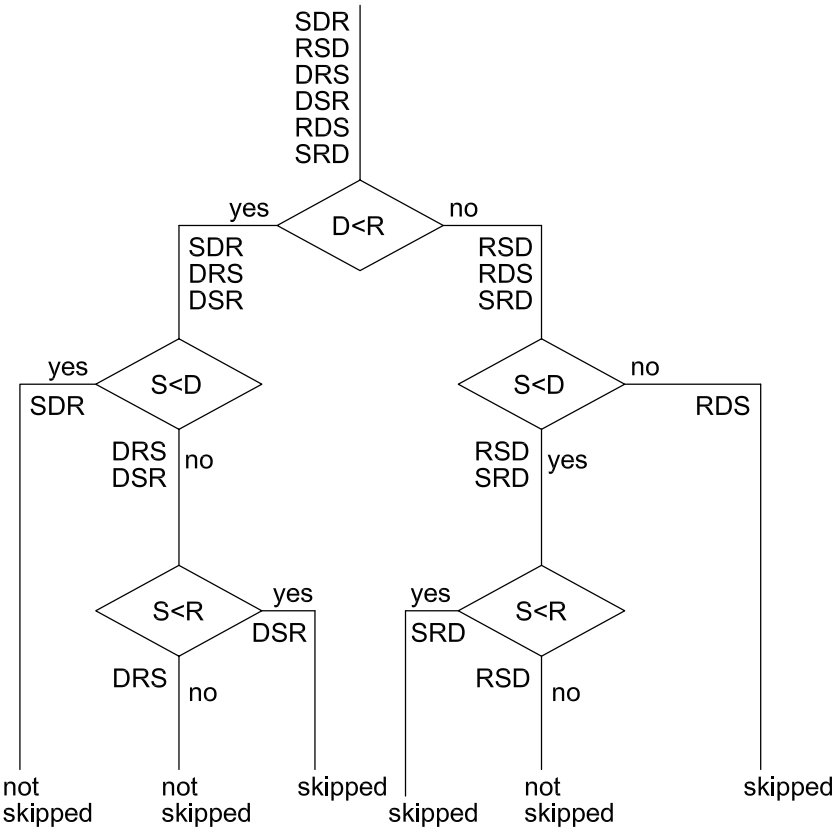
The receiver node has been skipped in the lower three combinations. An alive message shall be emitted asynchronously by the receiver node.

NOTE It is not always necessary to look for skipping at the reception of a ring message:

S=D The source node does not know anything about other nodes.

D=R The receiver node of the ring message itself was addressed.

S=R The receiver node was the sender of the message.



Components
S node identification of the source
R node identification of the receiver
D node identification of the destination

Two or three IF conditions shall be present.

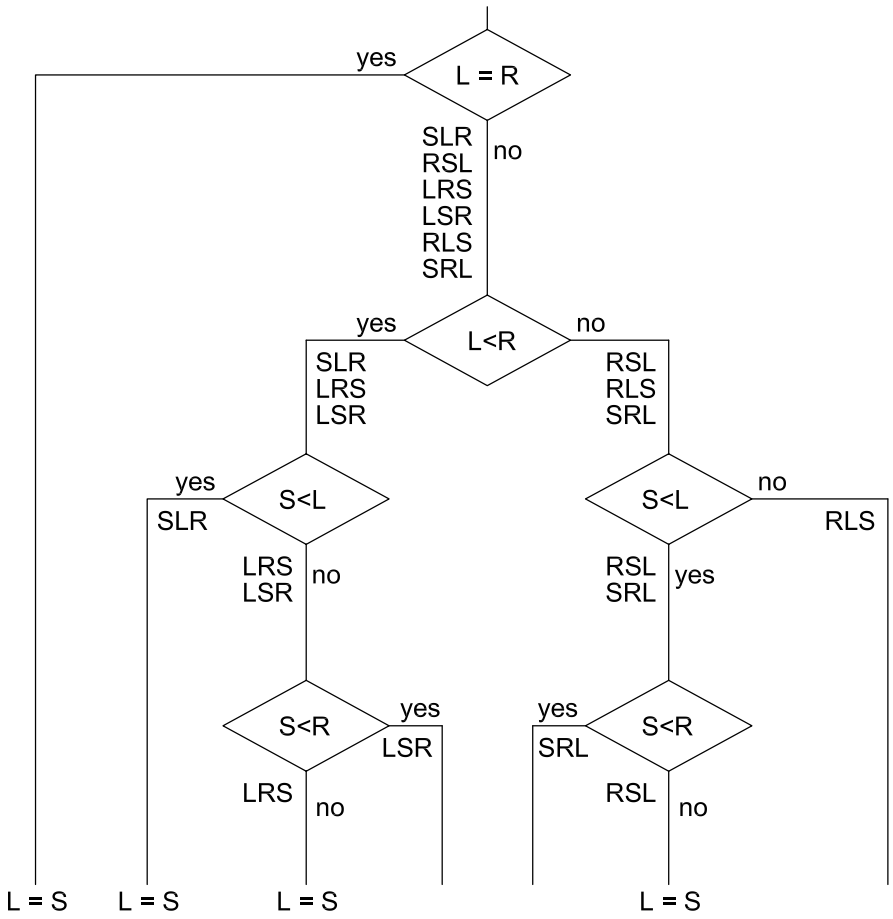
Figure 18 — IF conditions for the test "Was a receiver node skipped by a ring message on the logical ring?"

2.2.6 Example: Logical successor

The source node of any received NM message could turn to the logical successor of the received node. To reach a decision on whether the source node is the new logical successor of the receiver node, the receiver node shall look to the receiver identification, the source identification and to the identification of the logical successor.

SLR	(Source)	<	(Log. successor)	<	(Receiver)	new logical successor
RSL	(Receiver)	<	(Source)	<	(Log. successor)	new logical successor
LRS	(Log. successor)	<	(Receiver)	<	(Source)	new logical successor
LSR	(Log. successor)	<	(Source)	<	(Receiver)	✓
RLS	(Receiver)	<	(Log. successor)	<	(Source)	✓
SRL	(Source)	<	(Receiver)	<	(Log. successor)	✓

The state NMReset initializes the system-related basic configuration. Therefore, the values L (Log. successor identification) and R (Receiver identification) are equal. The algorithm shall be initialized. The source of the first received NM message shall be the logical successor.



Components
S node identification of the source
R node identification of the receiver
D node identification of the destination

Three or four IF conditions shall be present.

Figure 19 — IF conditions to determine a logical successor

NOTE It is possible to determine the logical successor from the stored present configuration when a ring message has to be emitted.

2.2.7 Operating mode

2.2.7.1 NMActive-NMPassive

In heterogeneous networks, individual nodes can suspend their network communication due to their specific requirements.

Each node owns a *silent mark* which can be set and reset by the application:

— silent mark set: NMPassive desired = "1";

— silent mark cleared: NMActive desired = "0".

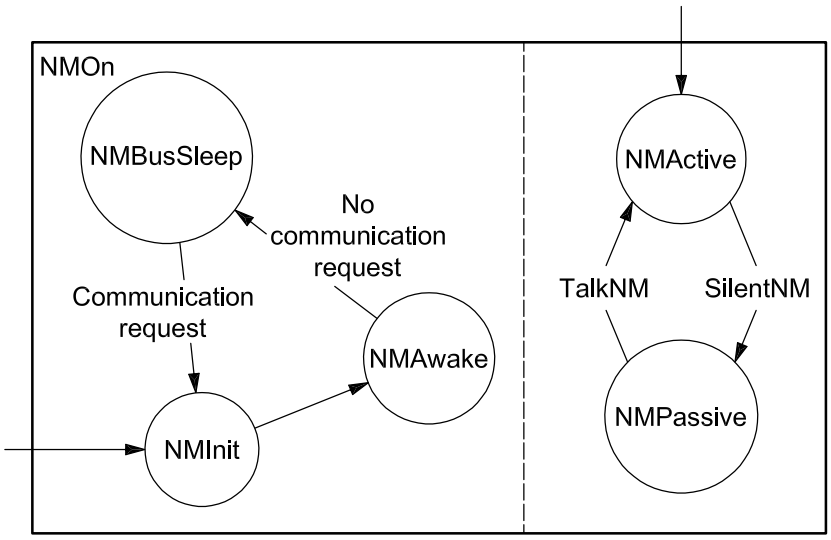


Figure 20 — Toggling between NMActive and NMPassive

2.2.7.2 NMBusSleep-NMAwake

2.2.7.2.1 General

The NM controls the access to the communication media on demand of the application. If the application in all nodes does not require the communication media, then the NM changes to the state NMBusSleep.

2.2.7.2.2 Principle for transition into BusSleep mode

Each node owns a sleep mark, which can be set and cleared by the application.

Table 7 — Services to change between the states NMBusSleep and NMAwake.

Service	Description	Sleep mark
GotoMode(BusSleep)	NMBusSleep desired	set
GotoMode(Awake)	NMBusSleep not desired	cleared

The NM maps this sleep mark (e.g. represented by a sleep bit) into each ring message (→ bit sleep.ind). If a set bit sleep.ind is transmitted, the NM internally changes to the state *NM~PrepBusSleep* (~: Normal or LimpHome).

When the sleep mark is set, NM prepares for notified and network-wide confirmed sleep mode.

The request for global NMBusSleep is set in a ring message. At each node participating in the logical ring, this request for global sleep shall be confirmed. The sleep mode initiating node shall wait for network-wide confirmation of his request.

If the NM message has completely looped in the logical ring, then the request is confirmed by a ring message with a set bit sleep.ack. The signaling specified by InitIndDeltaStatus is carried out and the transition is performed after a global delay $T_{WaitBusSleep}$. After the successful transmission of the ring message with a set bit sleep.ack, there can still be user messages in the transmit queues. Nodes in the state LimpHome are transmitting limp-home messages delayed by T_{Error}. Several limp-home messages can be received in this time period, thus a transition in the state NMBusSleep is possible without problems.

If the NM message has completely looped in the logical ring and the request is not confirmed, a NM message with different content is received or a NM message did not loop completely, the signalling specified by InitIndDeltaStatus cannot be carried out.

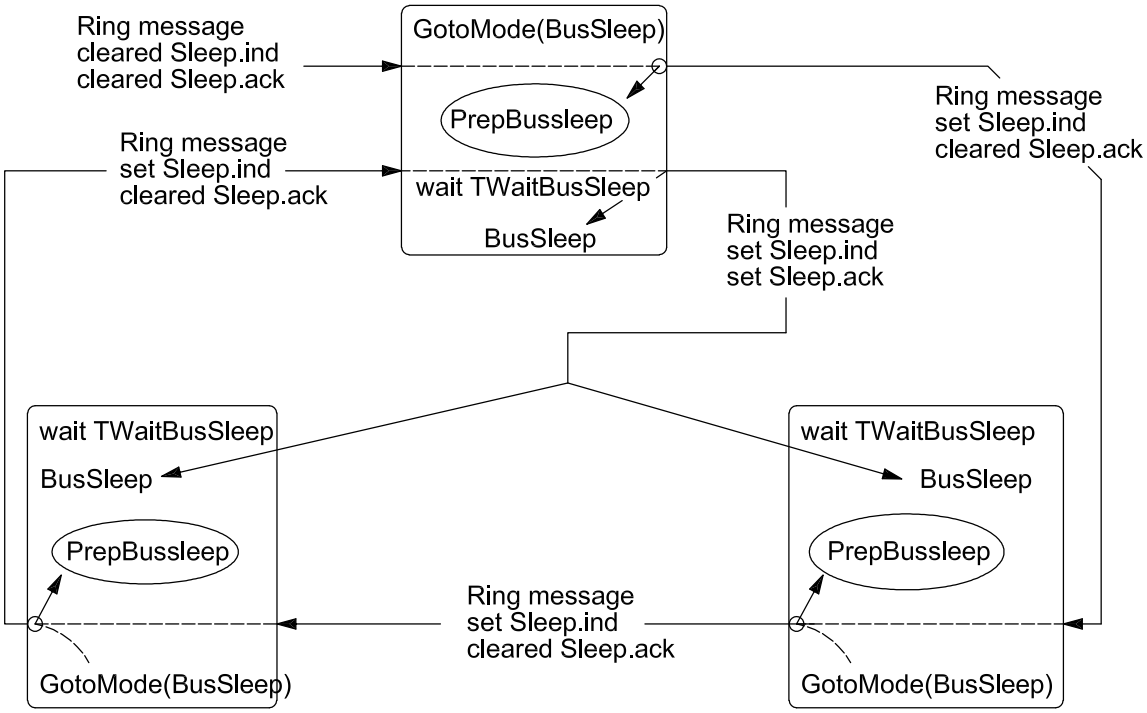


Figure 21 — Algorithm of the transition: NMNormal → NMBusSleep

All nodes are ready to change over into NMBusSleep only if the signalling specified by InitIndDeltaStatus is carried out. Up to that moment, application and NM shall operate in their normal mode (i.e. NMNormal). The application still continues with its communication in the network, thus preventing error messages by the asynchronous transition of the nodes into NMBusSleep.

For transition into *network-wide sleep mode*, the following cases are dealt with differently:

— transition from NMNormal into NMBusSleep;

— transition form NMLimpHome into NMBusSleep.

2.2.7.2.3 Transition from NMNormal into Network-wide BusSleep mode

The NM is informed about the mode requested by the local function GotoMode(BusSleep). Figure 22 shows the respective definitions.

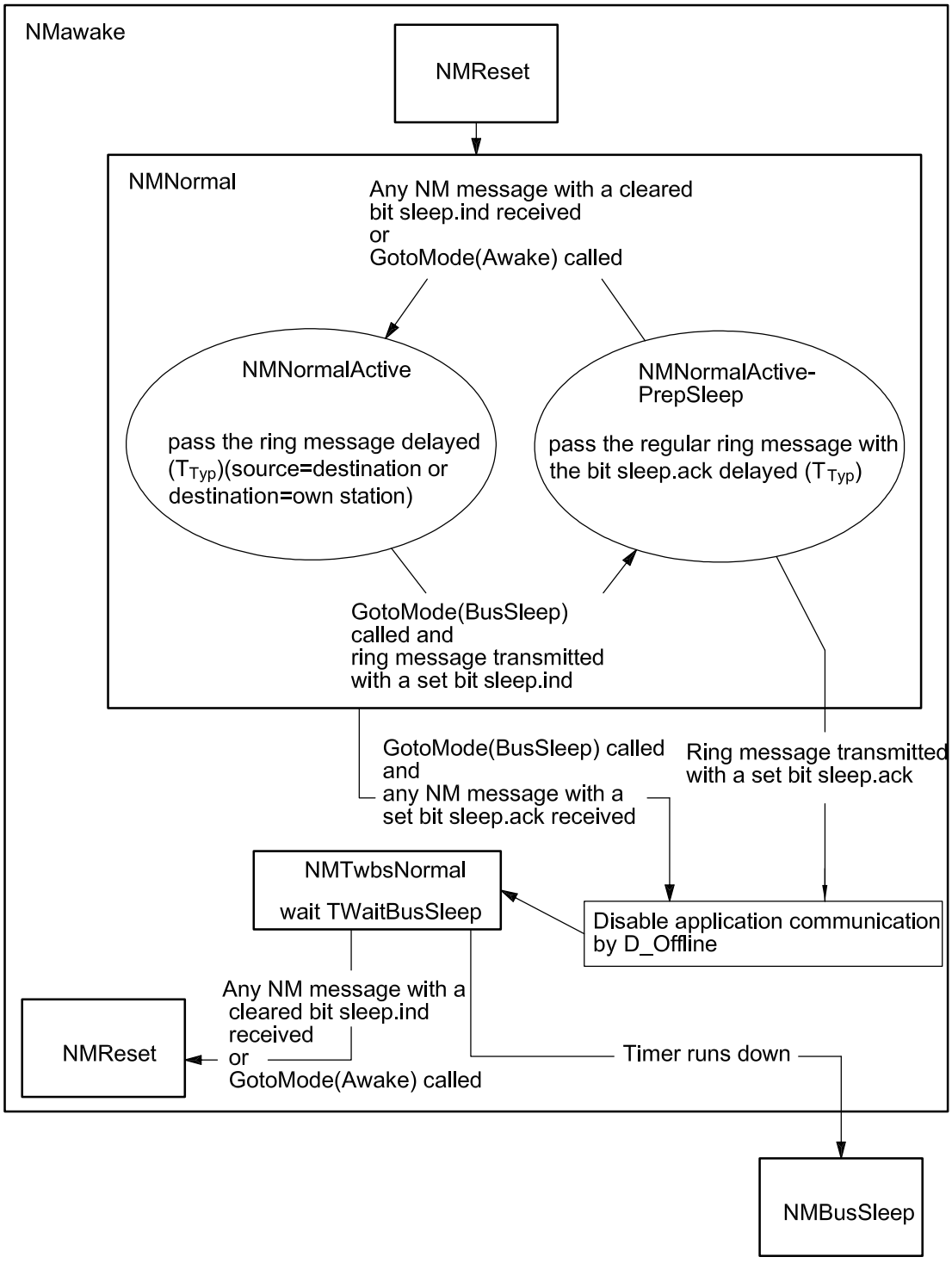


Figure 22 — Algorithm for transition NMNormal → NMBusSleep

2.2.7.2.4 Transition from NMLimpHome into network wide BusSleep mode

The function GotoMode(BusSleep) can also be called while NM operates in the mode NMLimpHome. Figure 23 shows the respective definitions.

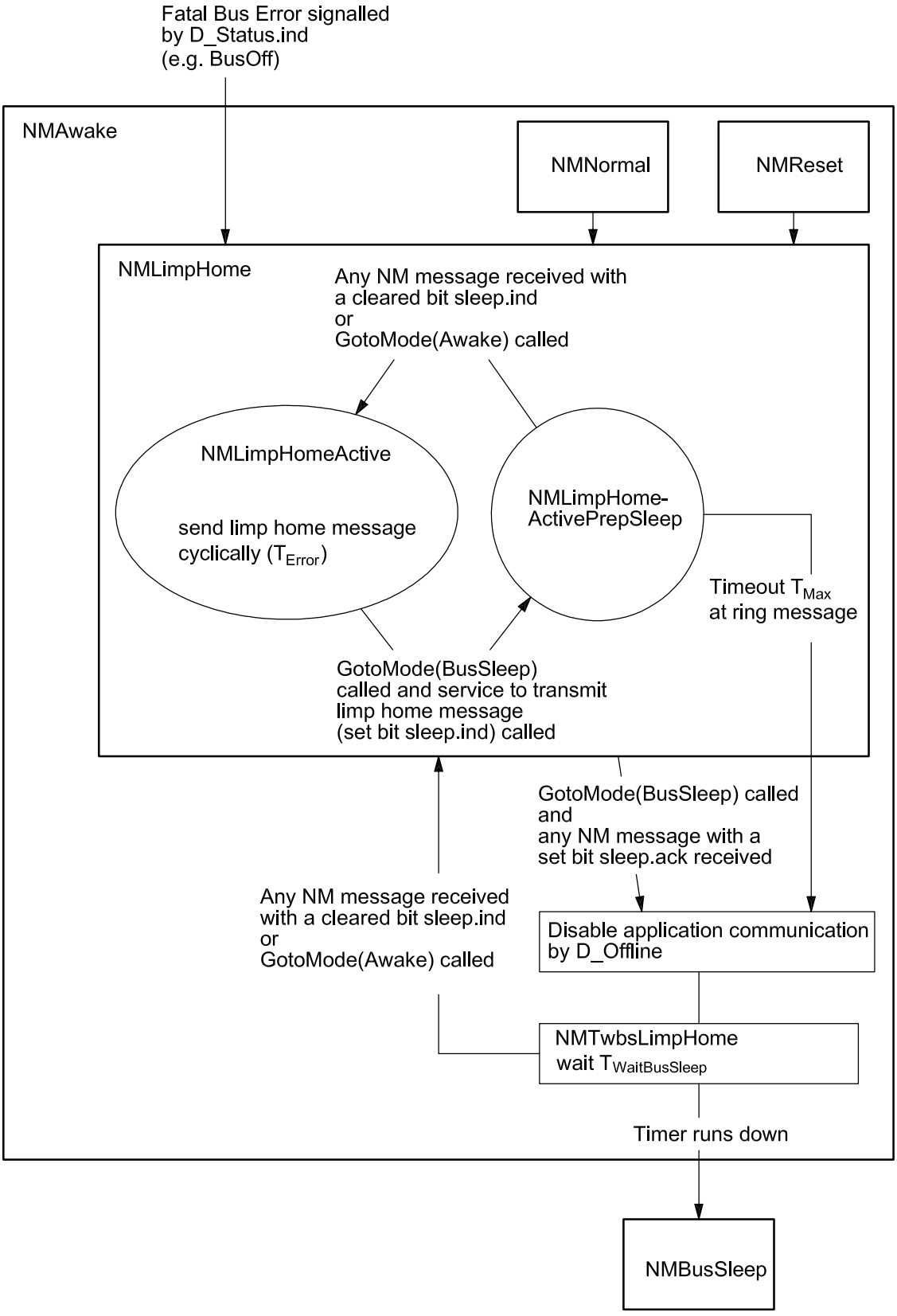


Figure 23 — Algorithm for transition NMLimpHome → NMBusSleep

2.2.7.2.5 Transition from Network-wide BusSleep mode to NMAwake

The state NMBusSleep is left if the service GotoMode(Awake) is called or if any NM message is received, i.e. a communication request exists.

2.2.8 Fusion of configuration management and operating modes

2.2.8.1 State Diagrams

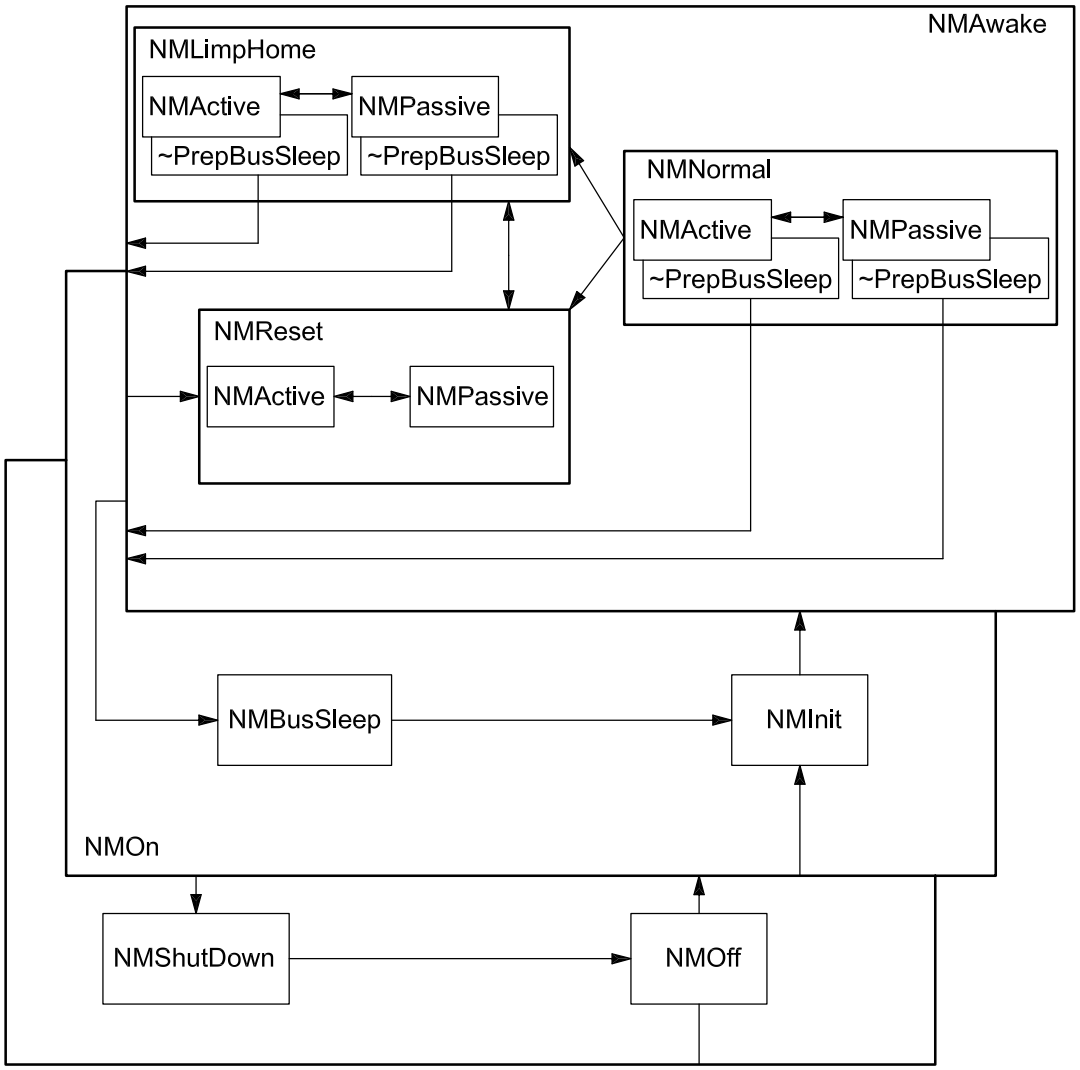


Figure 24 — Summary of simplified state transition diagram of the direct NM configuration management and operation modes

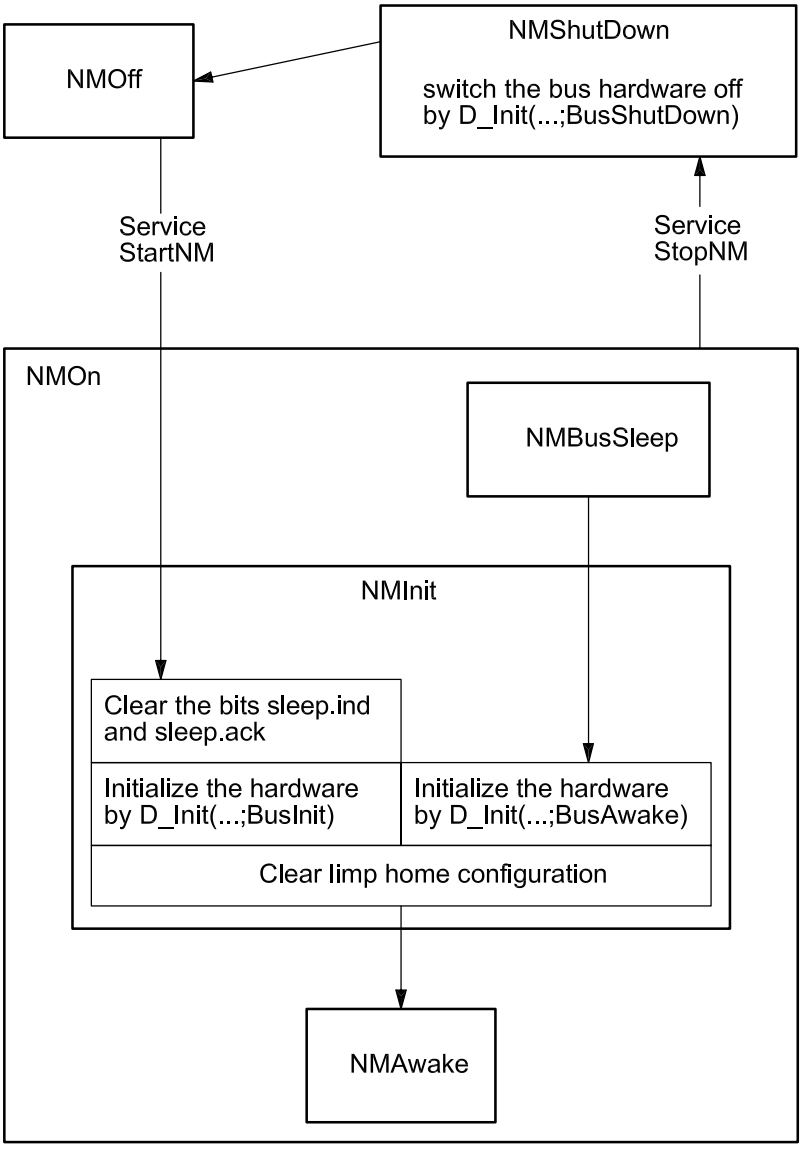


Figure 25 — State transition diagram of NMInit

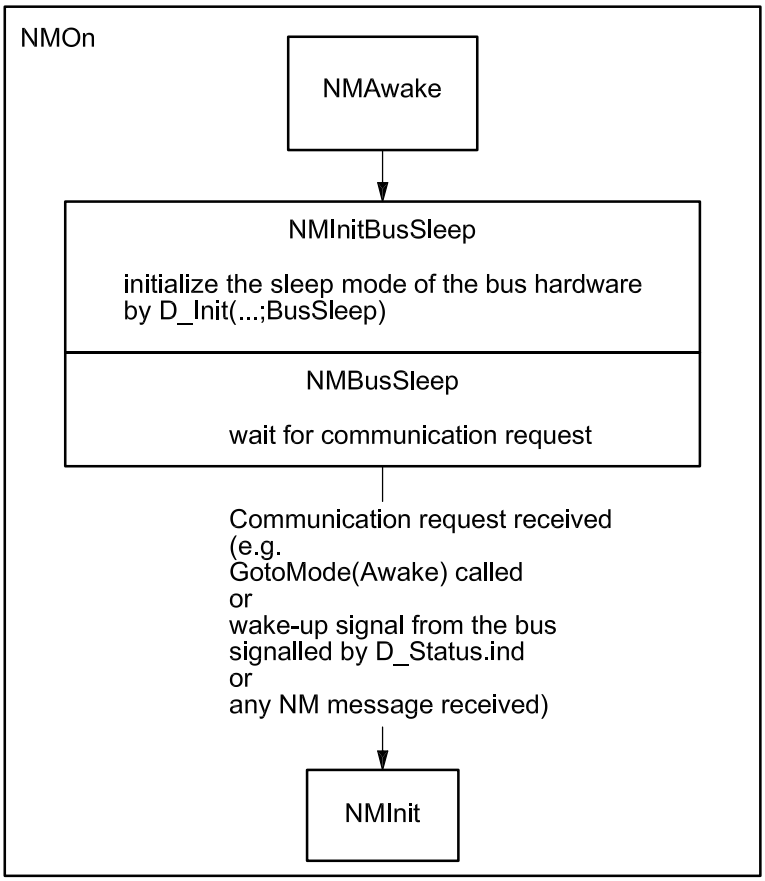


Figure 26 — State transition diagram of NMBusSleep

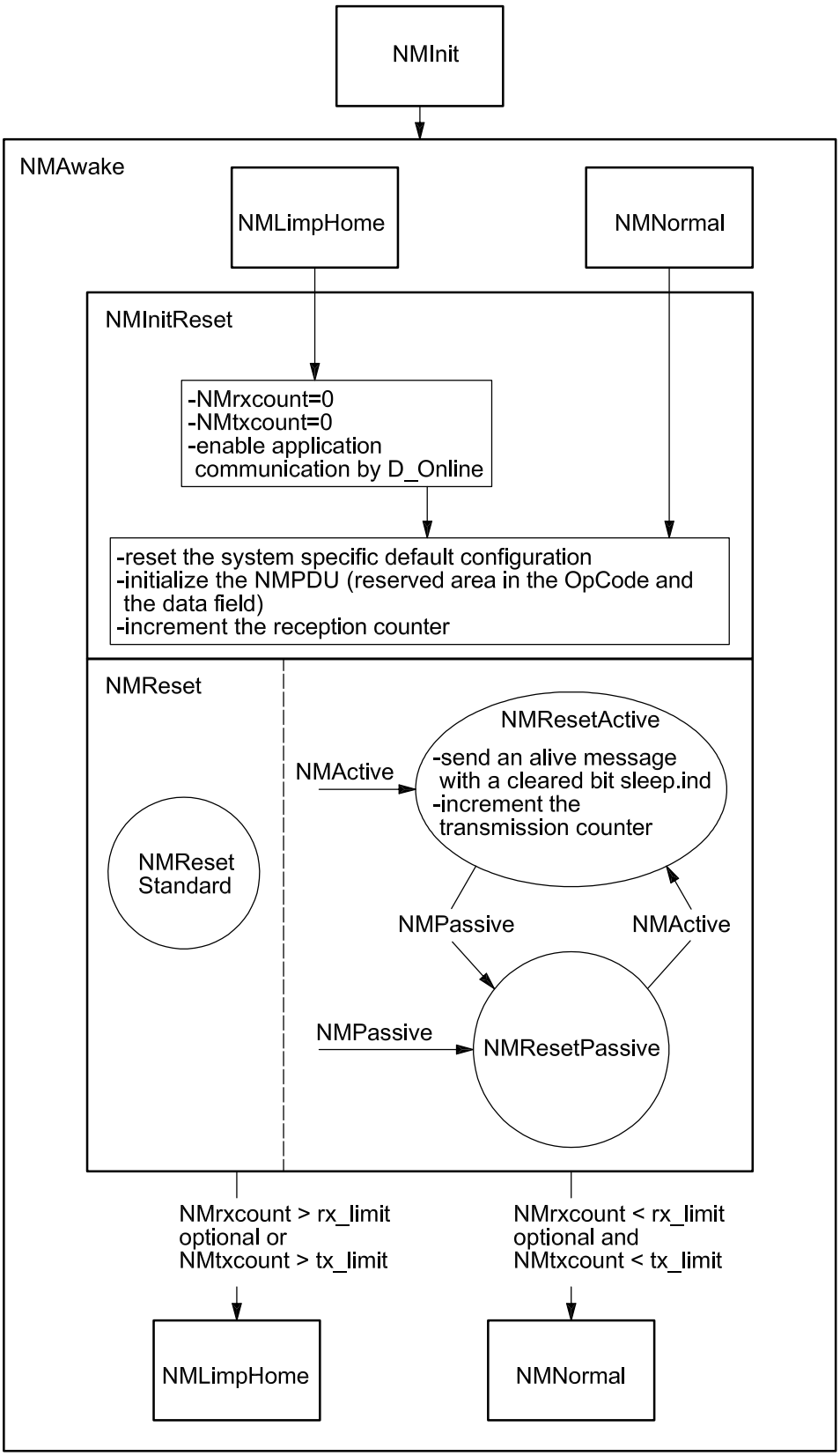


Figure 27 — State transition diagram of NMReset

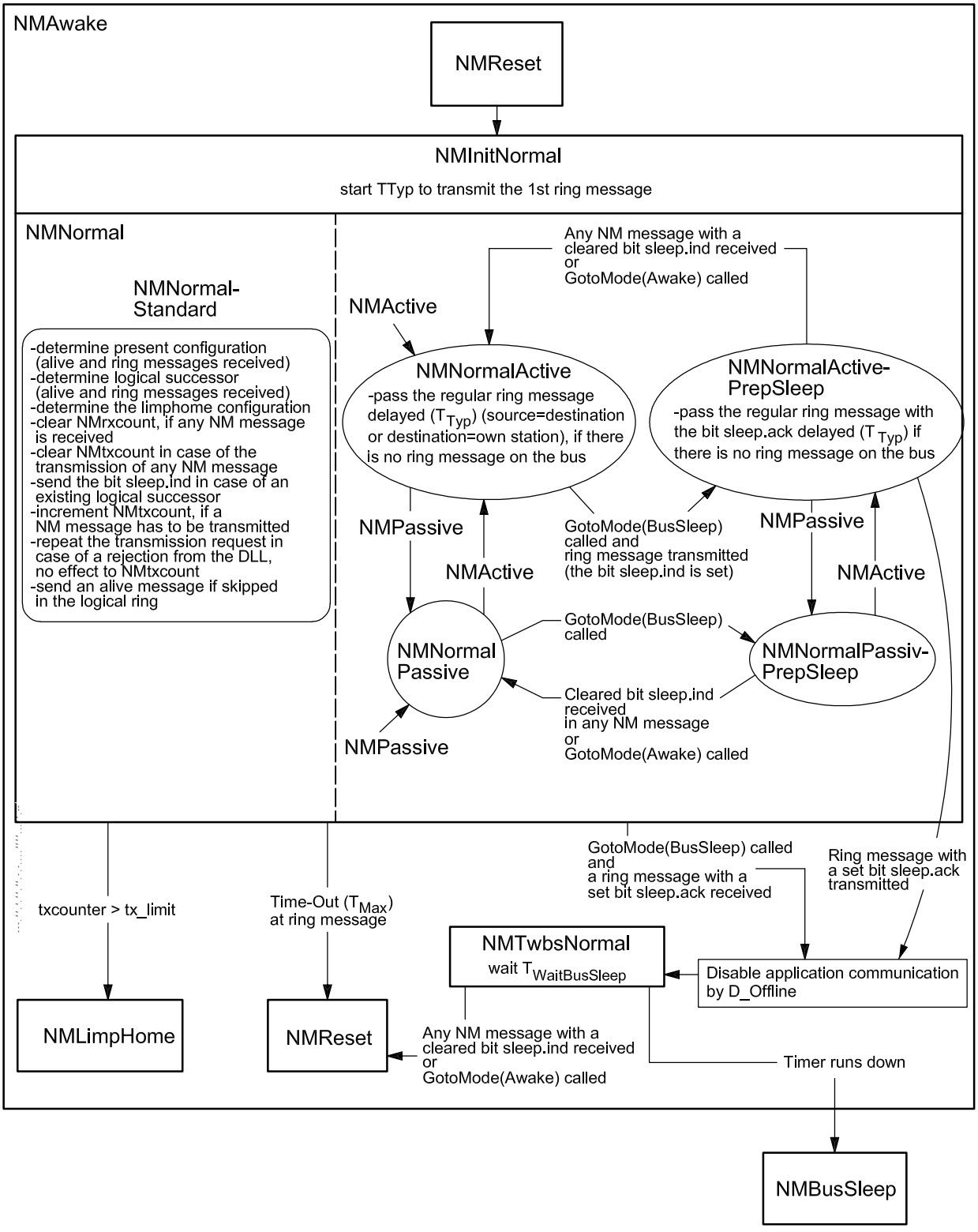


Figure 28 — State transition diagram of NMNormal

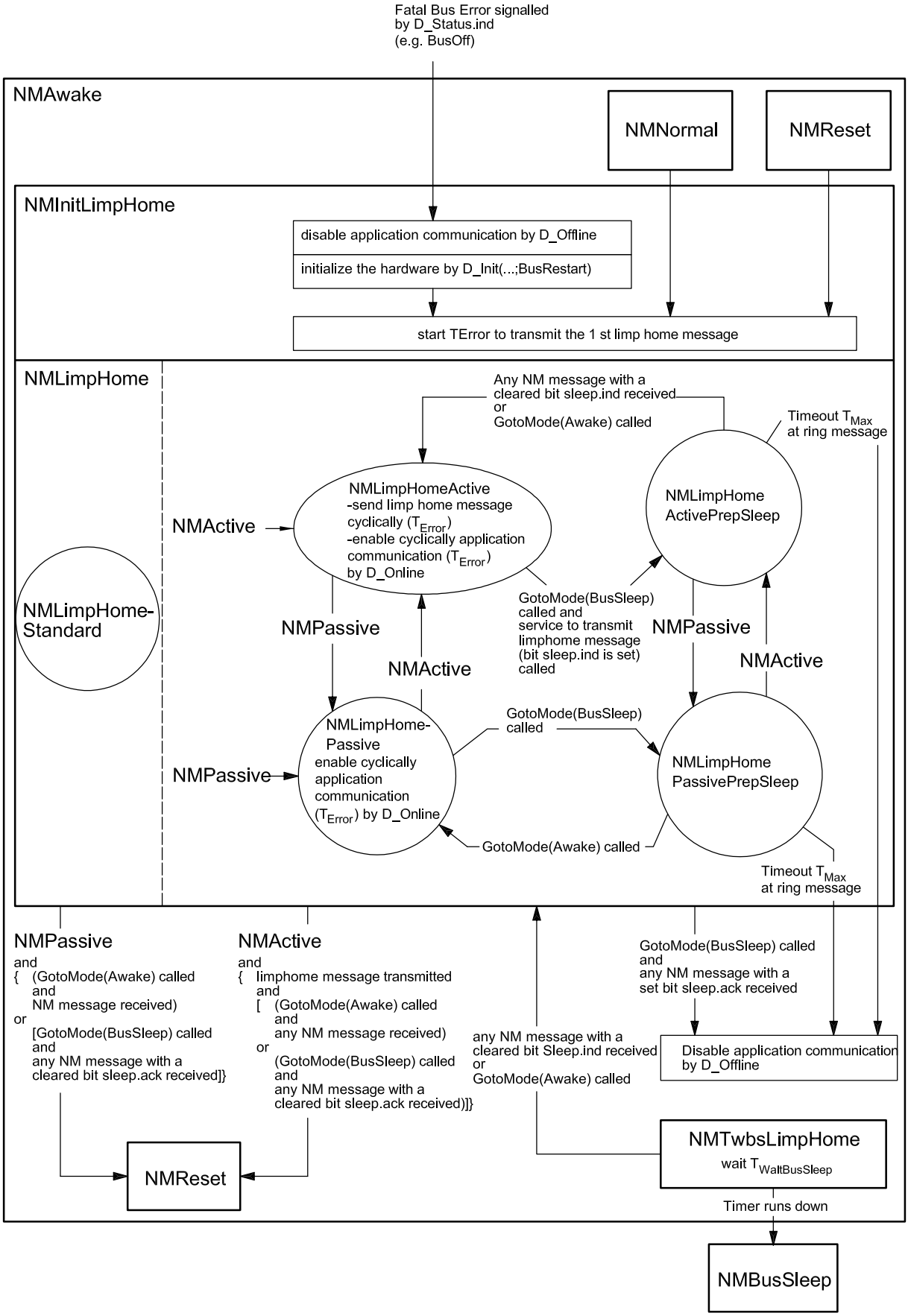


Figure 29 — State transition diagram of NMLimpHome

2.2.8.2 SDL diagrams

2.2.8.2.1 General

The specified behaviour is represented by the state transition diagrams. This subclause describes a proposed SDL realization.

The following abbreviations are used:

— sleep.ind: the bit “sleep.ind” from the actual received or transmitted NM message;

— sleep.ack: the bit “sleep.ack” from the actual received or transmitted NM message;

— networkstatus.bussleep: the bit of the network status “service GotoMode(Awake) called” or “service GoToMode(BusSleep) called”.

2.2.8.2.2 Start-up of the network

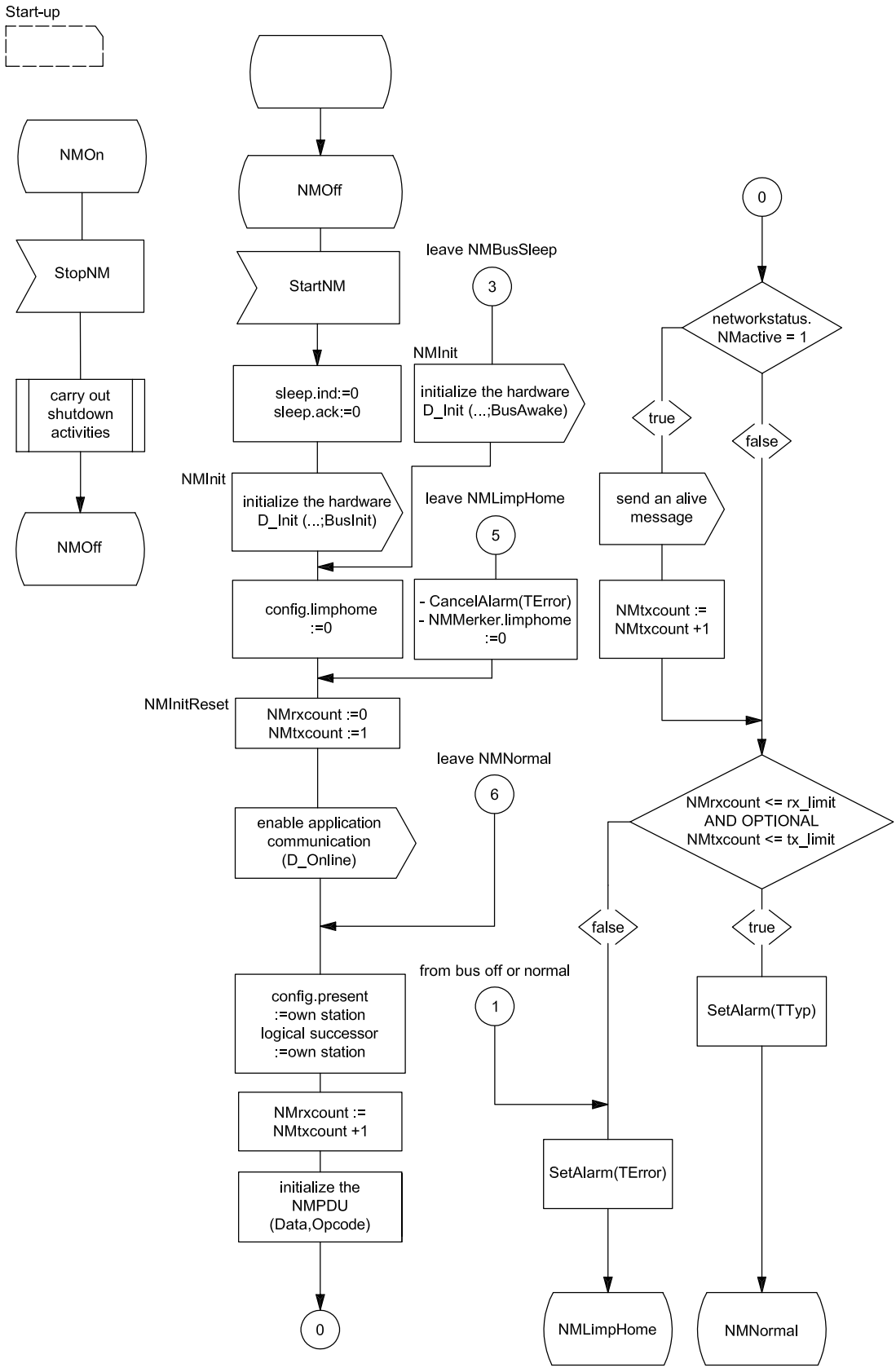


Figure 30 — Start-up of the network

2.2.8.2.3 State NMOn

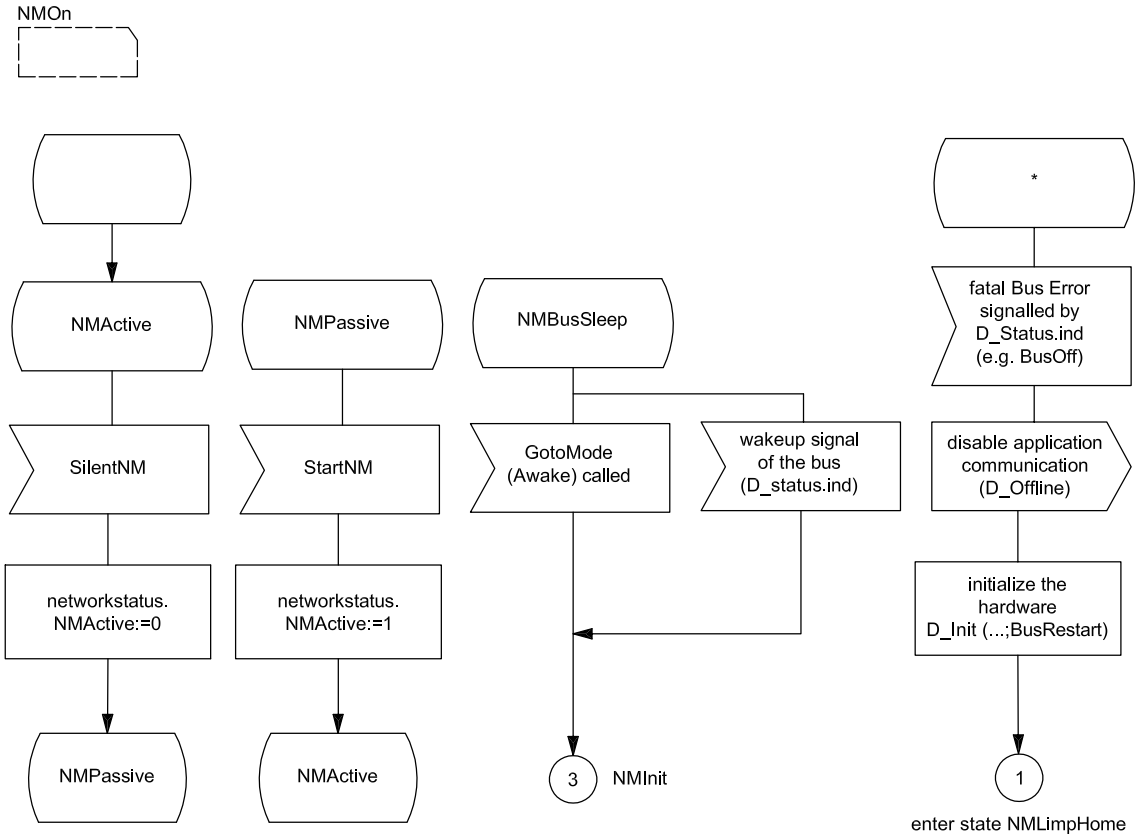


Figure 31 — Transitions between NMActive and NMPassive, wakeup from NMBusSleep, and bus off event

2.2.8.2.4 State NMNormal

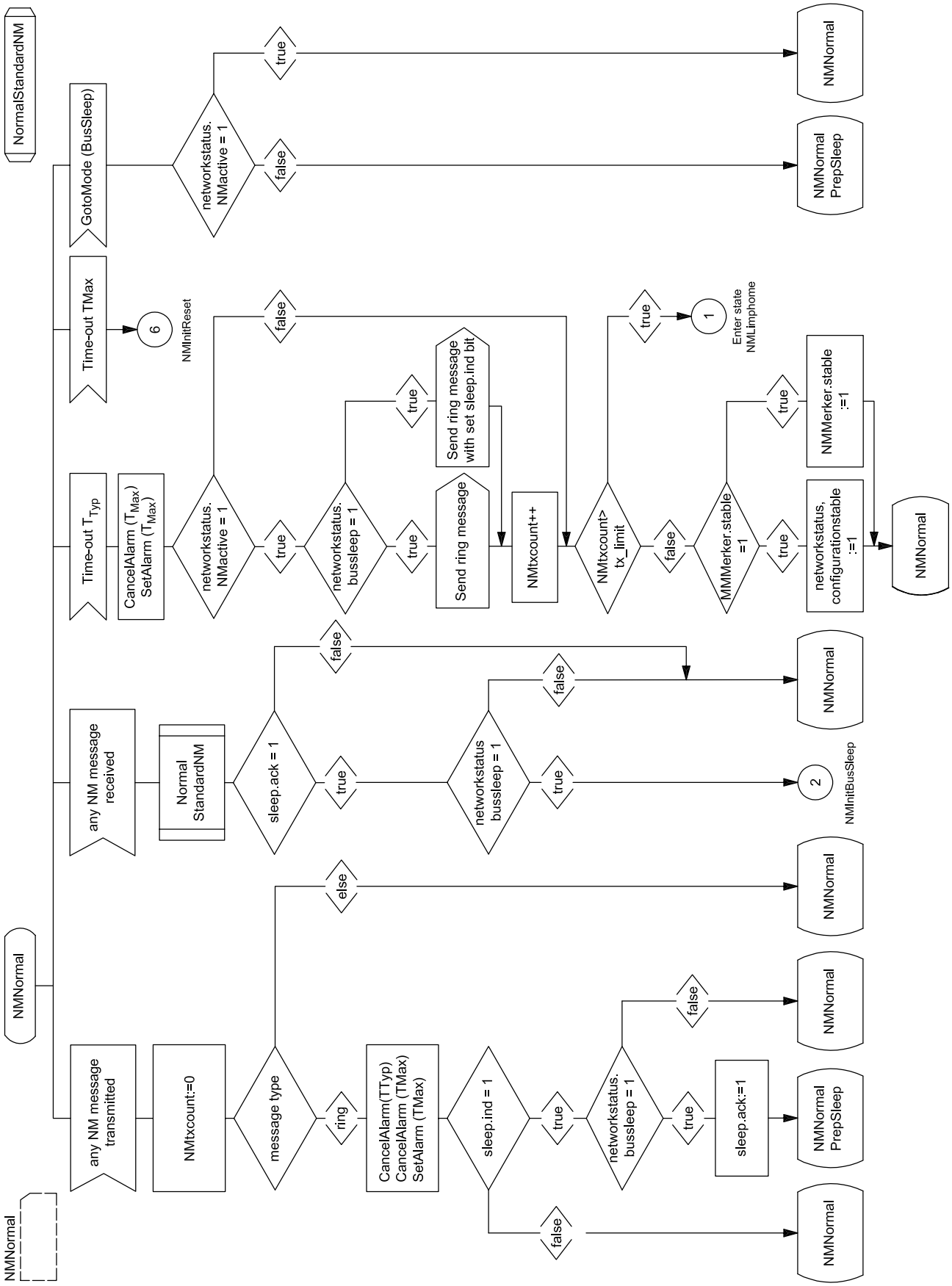


Figure 32 — Actions during the state NMNormal and transitions to leave the state NMNormal

2.2.8.2.5 State NMNormalPrepSleep

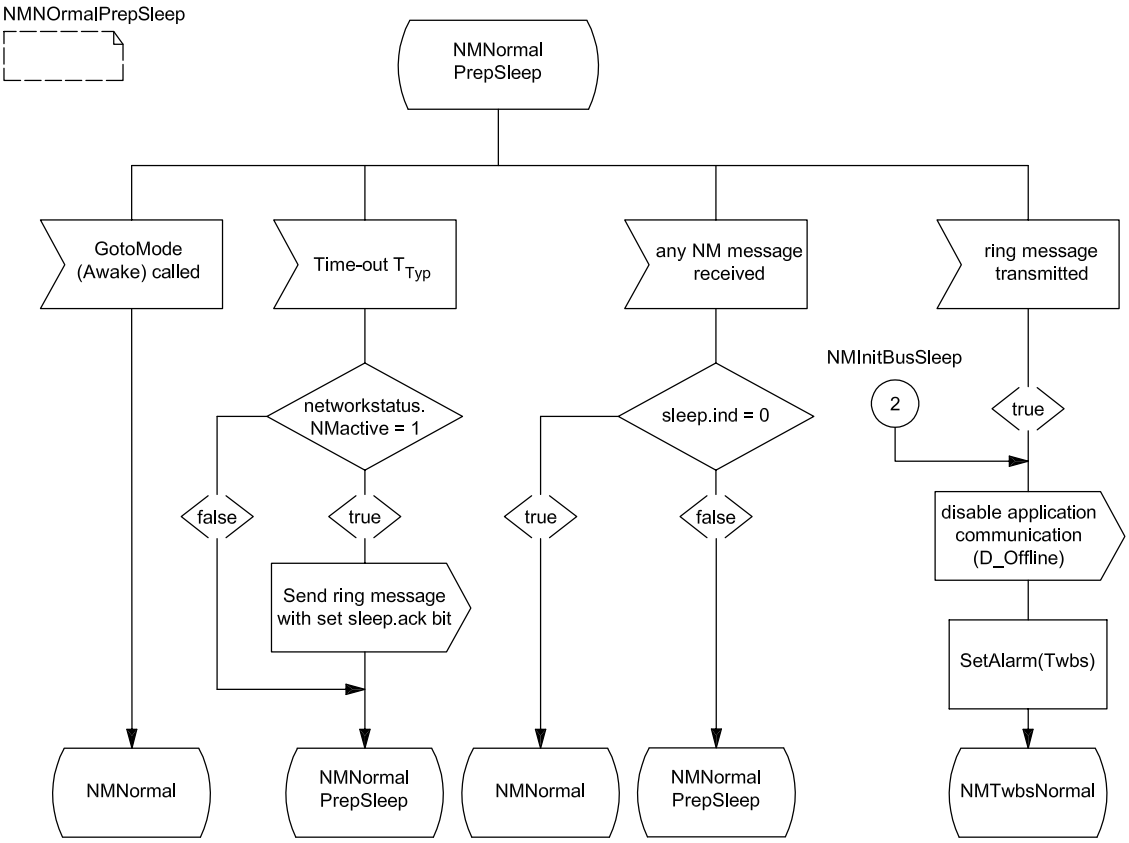


Figure 33 — Actions during the state NMNormalPrepSleep and transitions to leave the state NMNormalPrepSleep

2.2.8.2.6 State NMTwbsNormal

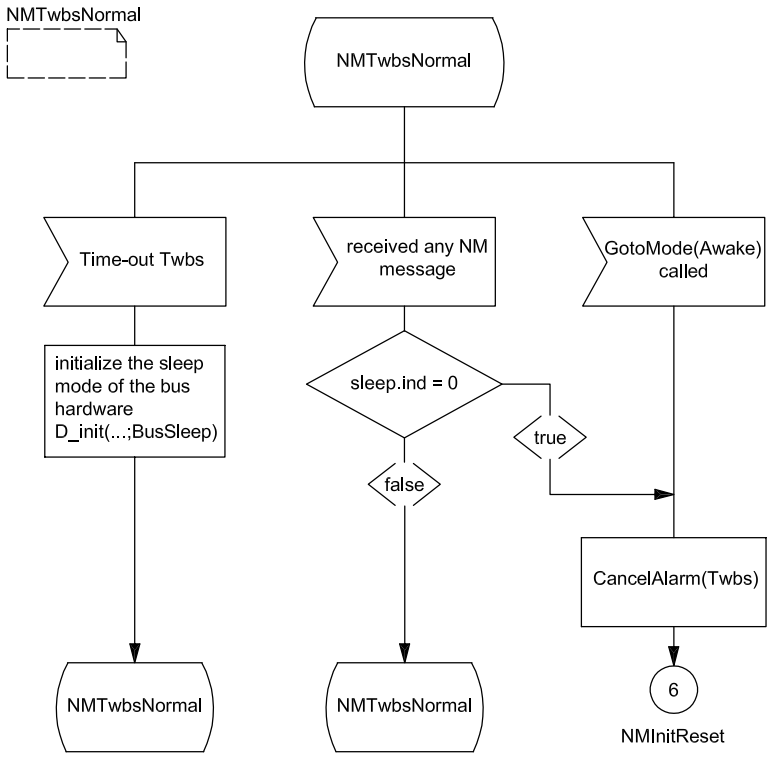


Figure 34 — Transitions to leave state NMTwbsNormal

2.2.8.2.7 State NMLimpHome

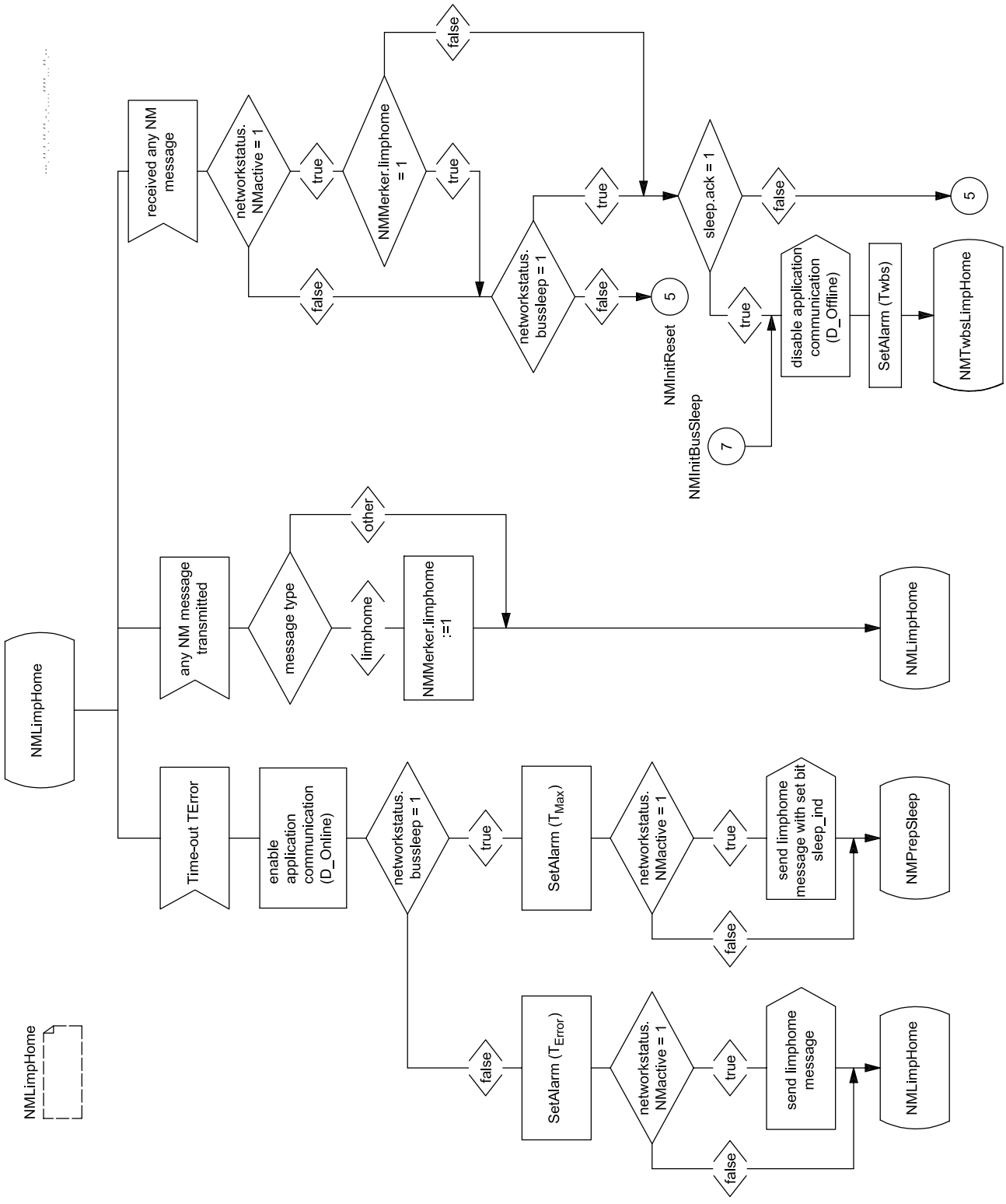


Figure 35 — Actions during the state NMLimpHome and transitions to leave the state NMLimpHome

2.2.8.2.8 State NMLimpHomePrepSleep

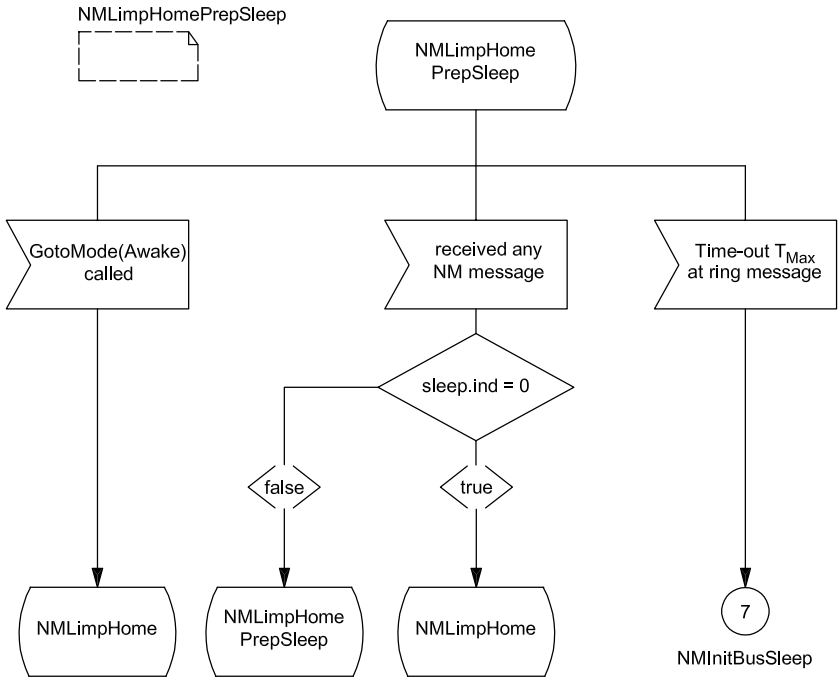


Figure 36 — NMLimpHomePrepSleep

2.2.8.2.9 State NMTwbsLimpHome

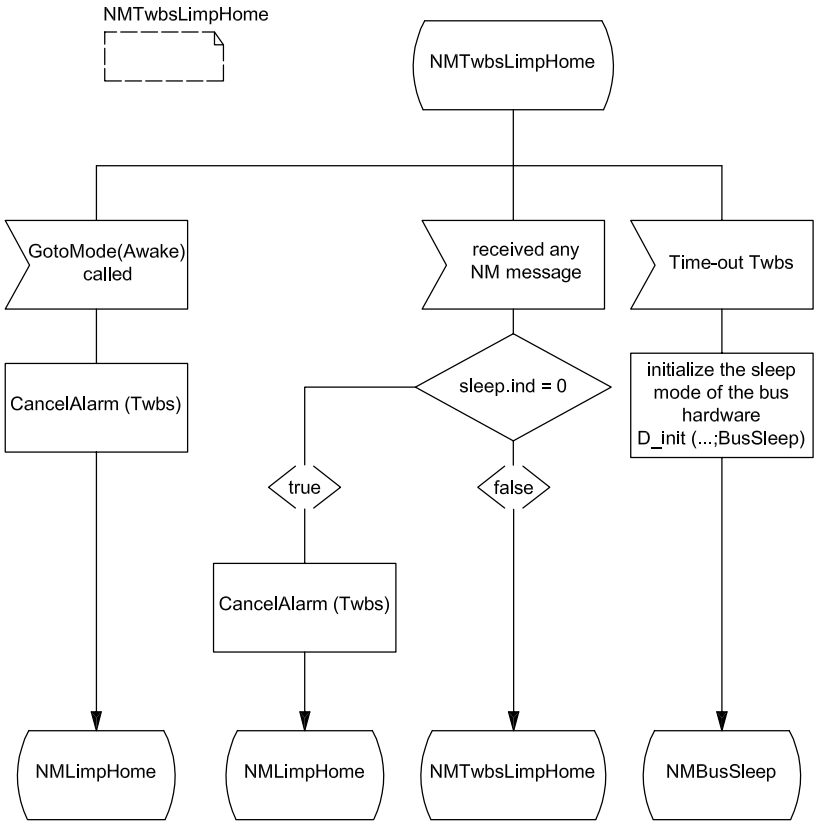


Figure 37 — Transmissions to leave the state NMTwbsLimpHome

2.2.8.2.10 Procedure NormalStandardNM

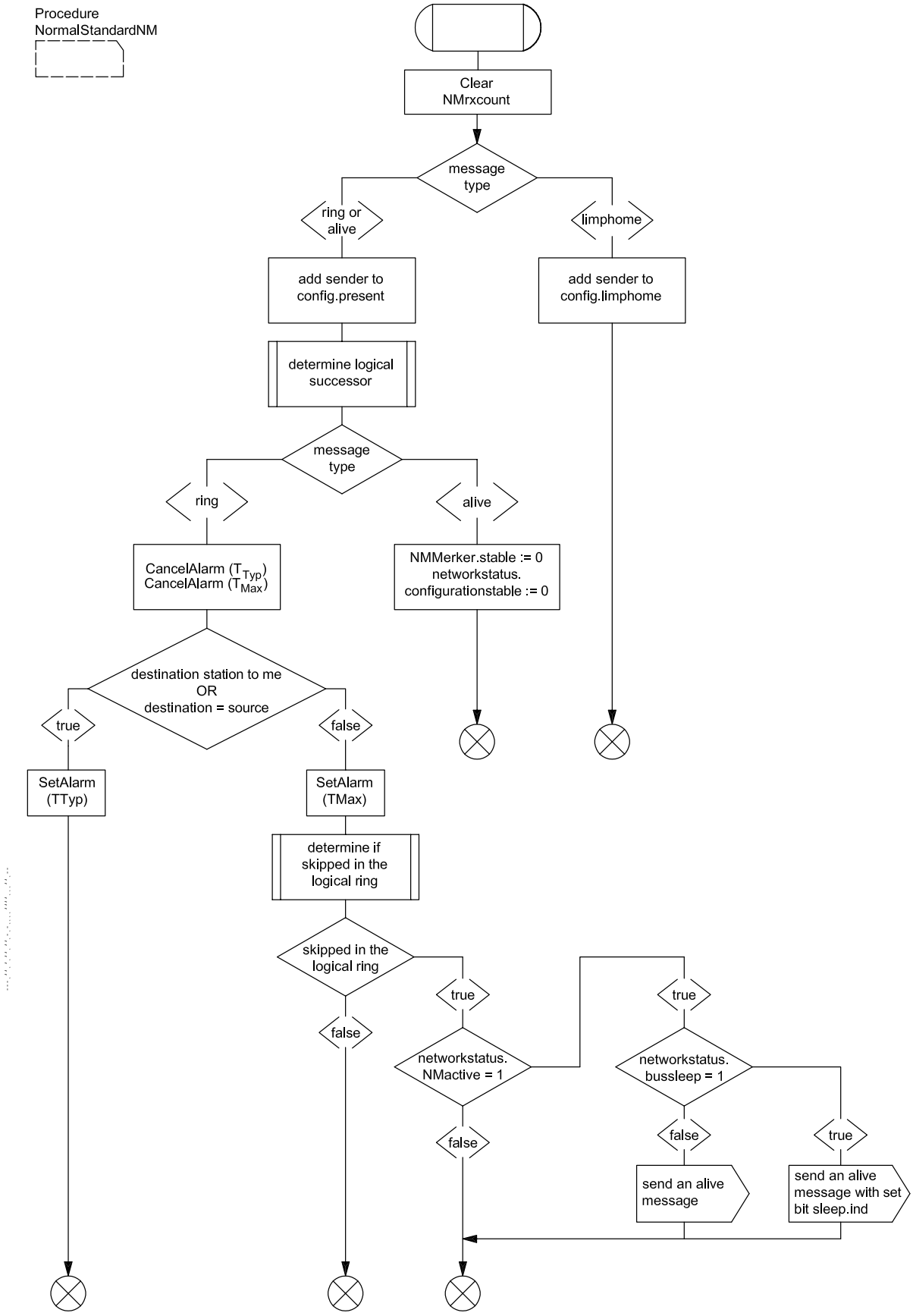


Figure 38 — Actions during NMNormalStandard

2.2.8.2.11 DLL transmit rejection, GotoMode(Awake) and GotoMode(BusSleep)

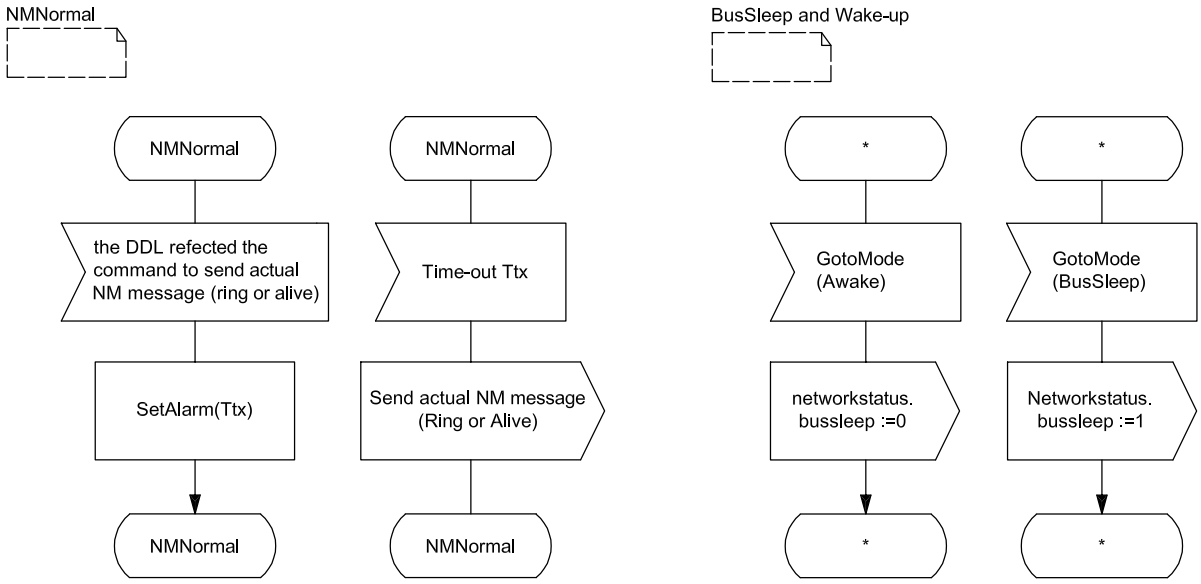


Figure 39 — DLL transmit rejection and GotoMode(Awake/BusSleep)

2.2.8.2.12 Indication of ring data, configuration and status

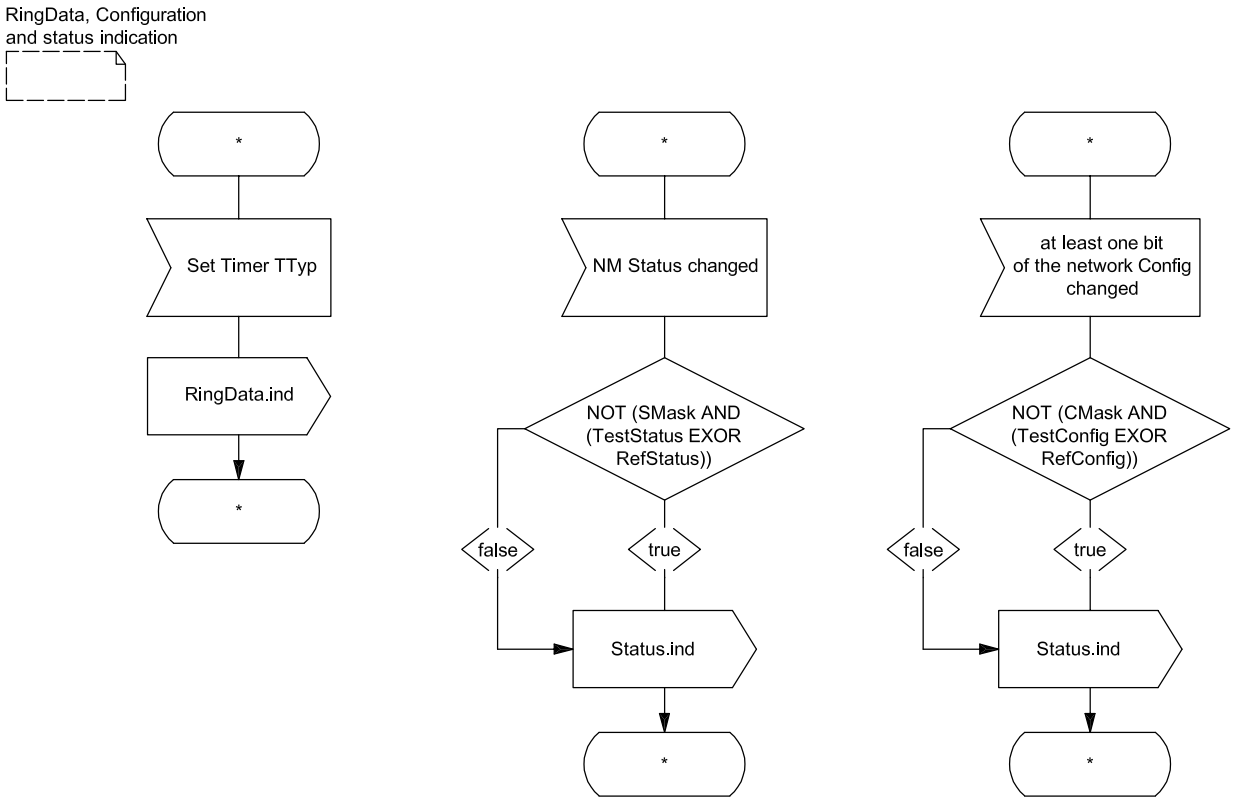


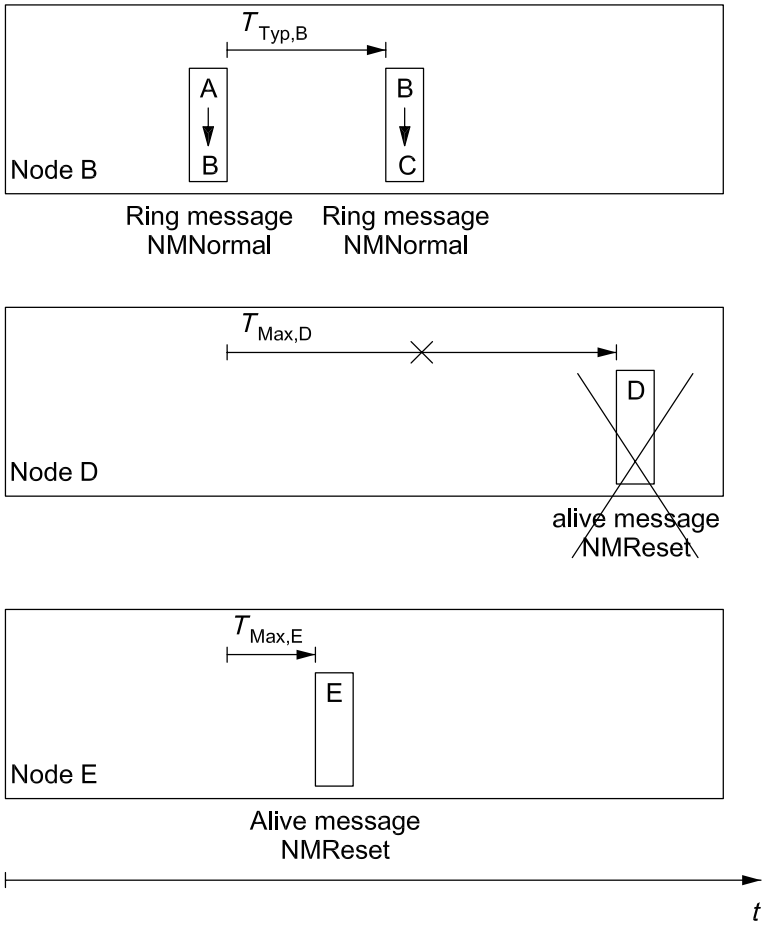
Figure 40 — Indication of ring date, configuration and network status

2.2.9 Alarms inside the Network Management

2.2.9.1 Rules to design the alarms T_{Typ} and T_{Max}

The definition of the logical ring requires that no alarm T_{Max} may run down if a ring message is passed delayed with T_{Typ}. This derives a requirement to the precision of the alarms inside a networked system (the transmission time of a message and the runtime of the software are not taken into consideration):

$$\left(T_{\text{Max}}\right)\Big|_K \quad > \quad \left(T_{\text{Typ}}\right)\Big|_J \qquad K,J \in \left[0;N-1\right]$$



Effect of the condition
$\left(T_{\text{Max}}\right)\Big|_K > \left(T_{\text{Typ}}\right)\Big|_J \quad K,J \in \left[0,N-1\right]$.

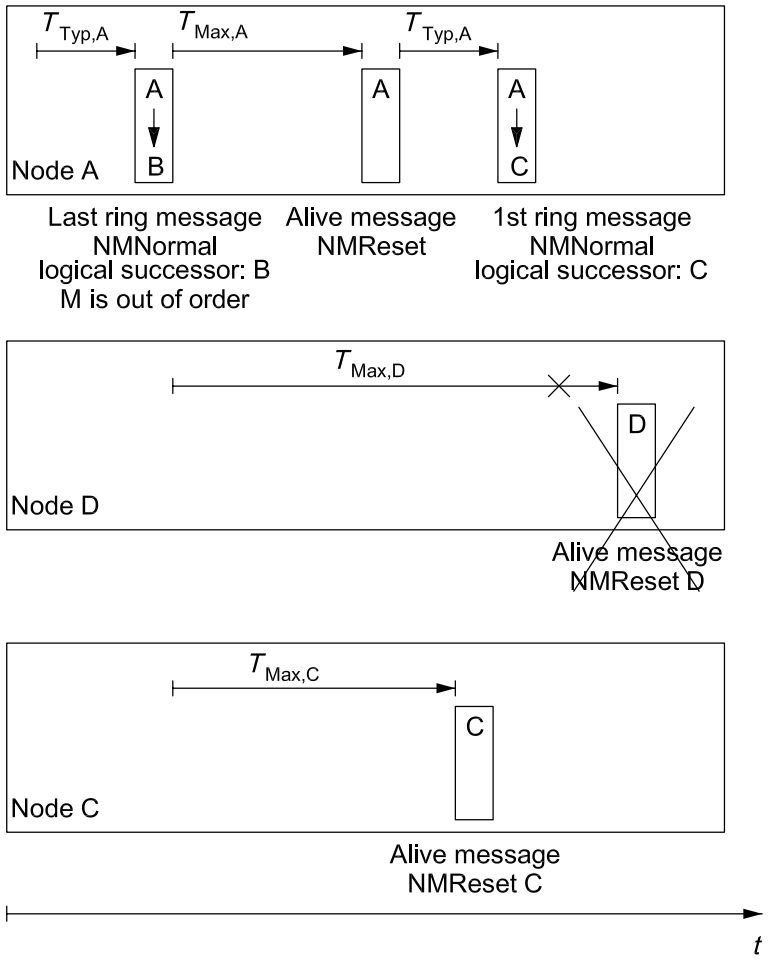
condition TRUE:
The Node D recognizes the correct running of the logical ring.

condition FALSE:
The node E recognizes the failure of another node although the ring is running perfectly.

Figure 41

The failure of a monitored node shall be recognized by all the other nodes inside the logical ring. All nodes shall be in NMNormal again, when the first ring message is transmitted after NMReset. This derives a requirement to the precision of the alarms inside a networked system (the transmission time of a message and the runtime of the software are not taken into consideration):

$$\left(T_{\text{Max}} + T_{\text{Typ}}\right)\Big|_K \quad > \quad \left(T_{\text{Max}}\right)\Big|_J \qquad K,J \in \left[0;N-1\right]$$



Key

1 Condition FALSE: The node D does not recognize the failure of the node B.

2 Condition TRUE: The node C recognizes the failure of the node B.

Figure 42 — Effect of the condition $\left(T_{\text{Max}} + T_{\text{Typ}}\right)\Big|_K \overset{!}{>} \left(T_{\text{Max}}\right)\Big|_J \quad K,J \in \left[0, N-1\right]$

Each of these alarms shall be provided with a tolerance (... $|_{\text{min}}$ and ... $|_{\text{max}}$) for every node. Inside a network all nodes shall meet both requirements:

$$\left(T_{\text{Max}}\big|_{\text{min}} + T_{\text{Typ}}\big|_{\text{min}}\right)\big|_K \rangle \left(T_{\text{Max}}\big|_{\text{max}}\right)\big|_J \quad K,J \in \left[0; N-1\right]; \text{ and}$$

$$\left(T_{\text{Max}}\big|_{\text{min}}\right)\big|_K \rangle \left(T_{\text{Typ}}\big|_{\text{max}}\right)\big|_J \quad K,J \in \left[0; N-1\right].$$

2.2.9.2 Rules to design the alarm T_{Error}

No important requirements for the alarm T_{Error}, which should be taken into consideration, exist. A useful value of the alarm T_{Error} is the value of T_{Typ} multiplied by 10. Tolerance calculations are insignificant.

2.2.9.3 Rules to design the alarm $T_{WaitBusSleep}$

After the successful transmission of the ring message with a set bit sleep.ack, there can still be user messages in the transmit queues. Nodes in the state limp-home are transmitting limp-home messages

delayed by T_{Error}. Several limp-home messages can be received in this time period; thus, a transition in the state NMBusSleep is possible without trouble.

The timer $T_{WaitBusSleep}$ is defined in addition to the timer T_{Error}. $T_{WaitBusSleep}\big|_{min} \geqslant T_{Error}\big|_{max}$ should be valid network wide. TWaitBusSleep is selected typically to 1,5 times of T_{Error}.

2.2.9.4 Design of a system

System requirements result from the requirements of the single alarms:

— recognizing a node failure:

$$\Delta T_{Max} = T_{Typ}\big|_{min} f_S \qquad 0 < f_S < 1;$$

— recognizing the logical ring:

$$T_{Typ}\big|_{max} = T_{Max}\big|_{min} f_R \qquad 0 < f_R < 1.$$

The tolerances of both alarms should be adapted to each other:

— with the precision:

$$\Delta T_{Typ} = \Delta T_{Max} f_\Delta .$$

The solution to determine the system requirements is:

$$T_{Typ}\big|_{max} = T_{Typ}\big|_{min} \left(1 + f_S f_\Delta\right);$$

$$T_{Max}\big|_{min} = T_{Typ}\big|_{min} \frac{1 + f_S f_\Delta}{f_R}; \text{ and}$$

$$T_{Max}\big|_{max} = T_{Typ}\big|_{min} \left(f_S + \frac{1 + f_S f_\Delta}{f_R} \right).$$

The designer of a system shall fix the values $T_{Typ}\big|_{min}, f_S, f_R, f_\Delta$ inside the whole network.

2.2.9.4.1 Worst Case

The worst case design points out the limit of the logical ring. The tolerances are selected for the perfect running of the logical ring in case of ideal communication system (e.g. the transmission time of a message and the runtime of the software disappears):

— recognizing a node failure:

$$\Delta T_{Max} = T_{Typ}\big|_{min} \qquad \Rightarrow f_S = 1;$$

— recognizing the logical ring:

$$T_{Typ}\big|_{max} = T_{Max}\big|_{min} \qquad \Rightarrow f_R = 1;$$

— with the precision:

$$\Delta T_{\text{Typ}} = \Delta T_{\text{Max}}\ f_{\Delta}\,.$$

The worst case system requirements are:

$$T_{\text{Typ}}\big|_{\text{max}} = T_{\text{Typ}}\big|_{\text{min}}\ (1+f_{\Delta})\,;$$

$$T_{\text{Max}}\big|_{\text{min}} = T_{\text{Typ}}\big|_{\text{min}}\ (1+f_{\Delta})\,;\ \text{and}$$

$$T_{\text{Max}}\big|_{\text{max}} = T_{\text{Typ}}\big|_{\text{min}}\ (2+f_{\Delta})\,.$$

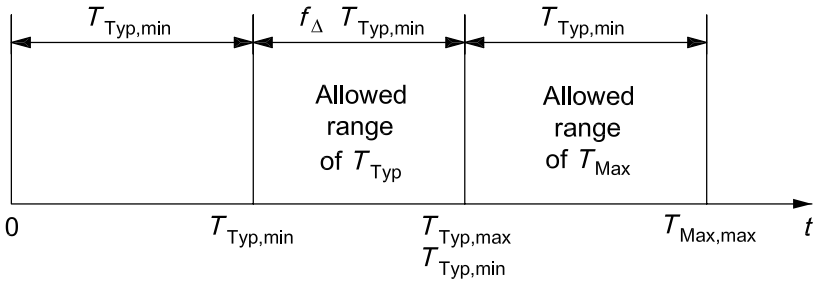


Figure 43 — Worst case system design of the alarms inside the NM

2.2.9.4.2 Example

For this example, the designer of the system fixed the values for $T_{\text{Typ}}\big|_{\text{min}}, f_S, f_R, f_{\Delta},$ as:

$$T_{\text{Typ}}\big|_{\text{min}} = 70ms \qquad f_S = 0{,}92 \qquad f_R = 0{,}5 \qquad f_{\Delta} = 0{,}62$$

The system-wide minimum and maximum values of the alarms T_{Typ} and T_{Max} result from the following fixed values for $T_{\text{Typ}}\big|_{\text{min}}, f_S, f_R, f_{\Delta}$:

$$T_{\text{Typ}}\big|_{\text{max}} = 110ms \qquad T_{\text{Max}}\big|_{\text{min}} = 220ms \qquad T_{\text{Max}}\big|_{\text{max}} = 284ms$$

Every node shall guarantee that their alarms remain inside the fixed limits.

EXAMPLE System design

T_{Typ} = 70ms ... 110ms
T_{Max} = 220ms ... 284ms
T_{Error} = ... 1s ...
$T_{\text{WaitBusSleep}}$ = ... 1,5s ...

3 Indirect Network Management

3.1 General

According to system design aspects, direct monitoring of the nodes may be impossible or nondesirable. This could be the case for example for very simple or time-critical applications.

Therefore, the mechanism of indirect monitoring is introduced. This Network Management is based on the use of monitored application messages. Indirect monitoring is thus limited to nodes that periodically send messages in the course of normal operation.

In this case, a node emitting such a periodical message is monitored by one or more other nodes receiving that message. Nodes whose normal functionality is limited to receiving shall send a dedicated periodic message in order to be monitored.

3.2 Concept

3.2.1 Node monitoring

3.2.1.1 General

Indirect Network Management uses monitoring of periodic application messages to determine states of nodes connected to the network. It does not make use of dedicated Network Management messages.

3.2.1.2 Node states

3.2.1.2.1 Emitter states

For a given node i, emitter states are used to check that node i, which is supposed to emit information on the bus, is indeed able to transmit:

— node is not mute → specific application message transmitted;

— node is mute → specific application message not transmitted during a time-out.

Node state “mute” can be extended to several state types (see 3.2.1.3).

3.2.1.2.2 Receiver states

A given node i monitors a subset of k nodes on the network; node i monitors only **source nodes**, from which it receives cyclic application messages. Therefore, node i will maintain a set of k receivers states, where k is the number of source nodes monitored by node i. Receiver states are used to check that node i, which is supposed to receive information from its k other source nodes, indeed receives information from each of its sources:

— node is present → specific application message received;

— node is absent → specific application message not received during a time-out.

Node state “absent” can be extended to several state types (see 3.2.1.3).

3.2.1.3 Extended node states

3.2.1.3.1 Extended emitter states

These include:

— node is not mute statically → specific application message transmitted;

— node is mute statically → specific application message not transmitted during a “long” time (several time-outs).

3.2.1.3.2 Extended receiver states

These include:

— node is present statically → specific application message received;

— node is absent statically → specific application message not received during a "long" time (several time-outs).

3.2.2 Configuration-management

3.2.2.1 Configuration

3.2.2.1.1 General

The configuration puts together the node states of all the monitored nodes determined by the NM.

3.2.2.1.2 Target configuration

The application recognizes node failures by comparison of the configuration (determined by the NM) with a target configuration. This target configuration may normally change depending on vehicle operation (e.g. nodes can appear and disappear from the network depending on ignition switch position).

NOTE The target configuration is not located inside NM. Several target configurations and several masks can be preprogrammed in the application. By using these masks depending on vehicle operation, the application is then able to filter information provided by NM by itself and recognize node failures.

3.2.2.2 Extended configuration

The extended configuration puts together the extended node states of all the monitored nodes determined by the NM.

3.2.3 Standard task

3.2.3.1 Network status

The network status is a set of information relating to local node hardware interface operation and local NM internal operating states.

Table 8 — Encoding of the network status

Network status	Interpretation
Operating mode of network interface	0 No error[a] 1 Error, Bus blocked[b]
Operation modes	0 NMOn 1 NMOff
	0 no NMLimpHome 1 NMLimpHome
	0 no NMBusSleep 1 NMBusSleep
	0 no NMWaitBusSleep 1 NMWaitBusSleep
[a] Reception and transmission of application messages successful. [b] e.g. CAN-busoff.	

3.2.3.2 Extended network status

The extended network status is specific to the user.

Table 9 — Example of encoding of the extended network status

Network status	Interpretation
Operating mode of network interface	00 No error[a] 01 Error, Communication possible[b] 10 Error, Communication not possible[c] 11 reserved
a Successful reception and transmission of application messages. b Communication via one wire. c E.g. CAN-busoff for a “long” time.	

3.2.4 Monitoring mechanisms

In order to evaluate node states and network status, Indirect Network Management provides three non-exclusive mechanisms of monitoring:

— Transmission: Determination of the emitter states by using transmission monitoring scheme. Transmission problems are detected by checking local confirmations related to transmissions of a unique periodic application frame chosen among those to be sent. This local confirmation is used to set the emitter states accordingly.

EXAMPLE If a message is correctly transmitted in case of CAN, it is then acknowledged on the bus. If the transmission fails, there is no acknowledgement and after a time-out, node i is considered “mute” by the NM.

— Reception: Determination of the set of receiver states by using reception monitoring scheme. Node i checks the presence of all its source nodes by monitoring the reception of a chosen cyclic frame per each remote source. If the supervised message of node k is not received at least once by node i before a configurable time-out, node k is then considered absent.

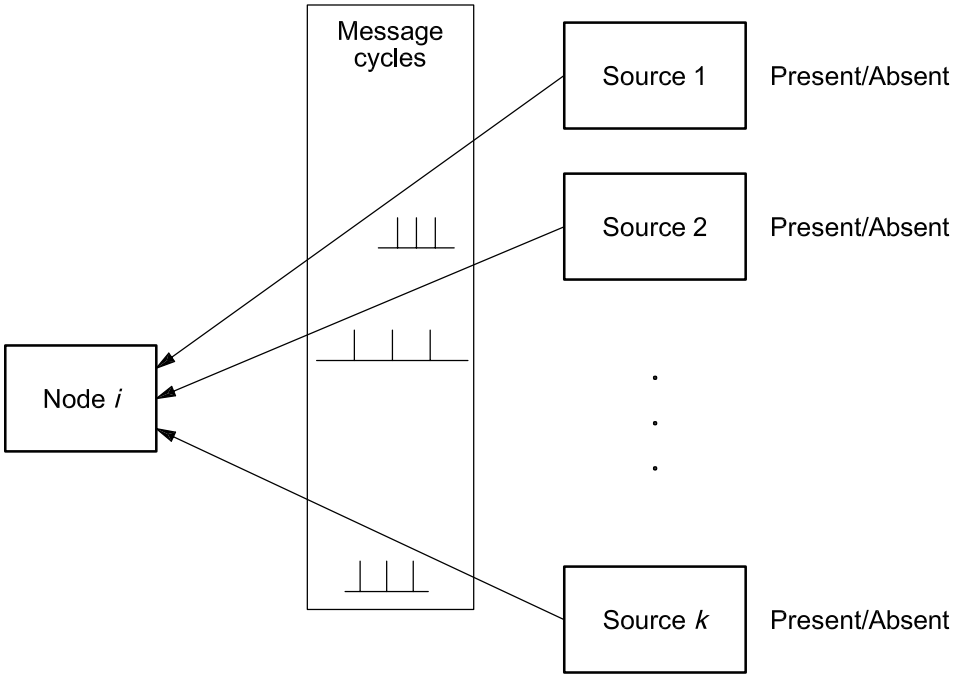


Figure 44 — Reception monitoring

— Status signal: Determination of Network Interface status by using controller indications from communication Data Link Layer, which itself uses low level controller or driver information.

EXAMPLE If the bus is blocked in case of CAN, controller indicates a “bus off” error to upper layers.

3.2.5 Monitoring time-outs

3.2.5.1 General

Indirect NM transmission and reception monitoring is based on two possible time-out monitoring mechanisms:

— All messages are monitored by one global time-out TOB (time-out for observation).

— Each message is monitored by its own dedicated time-out.

3.2.5.2 One global time-out

The global monitoring time-out is located inside NM and is used as a time-window observation:

— node present/not mute: at least one message has been transmitted or received from node k during the global time-out (time window observation);

— node absent/ mute: no message has been transmitted or received from node k during the global time-out (time window observation).

The monitoring time-out shall be adapted to the longest time requirement among all the monitored application messages.

NOTE The global time-window observation is handled in the SDL diagrams by a private configuration and a public configuration.

3.2.5.3 One monitoring time-out per message

In this case, Indirect NM uses “COM Deadline Monitoring” (see ISO 17356-4) mechanisms to monitor dedicated application messages. Time-outs are located at Interaction Layer level. NM is informed dynamically by COM each time a message has been correctly transmitted or received, or a time-out has expired for this message.

Each monitoring time-out can be adapted to the time requirements of each monitored application message.

3.2.5.4 Internal Network Management states

3.2.5.4.1 General

NM can enter the internal states listed hereafter:

— NMOff: NM is switched off;

— NMOn: NM is switched on;

NMOn:

— NMBusSleep: NM is in sleep mode;

— NMAwake: Active state of the NM;

NMAwake:

— NMNormal: Processing of indirect node monitoring;

— NMLimpHome: Handling of failure in own node;

— NMWaitBusSleep Synchronizing the network wide jump to the state BusSleep.

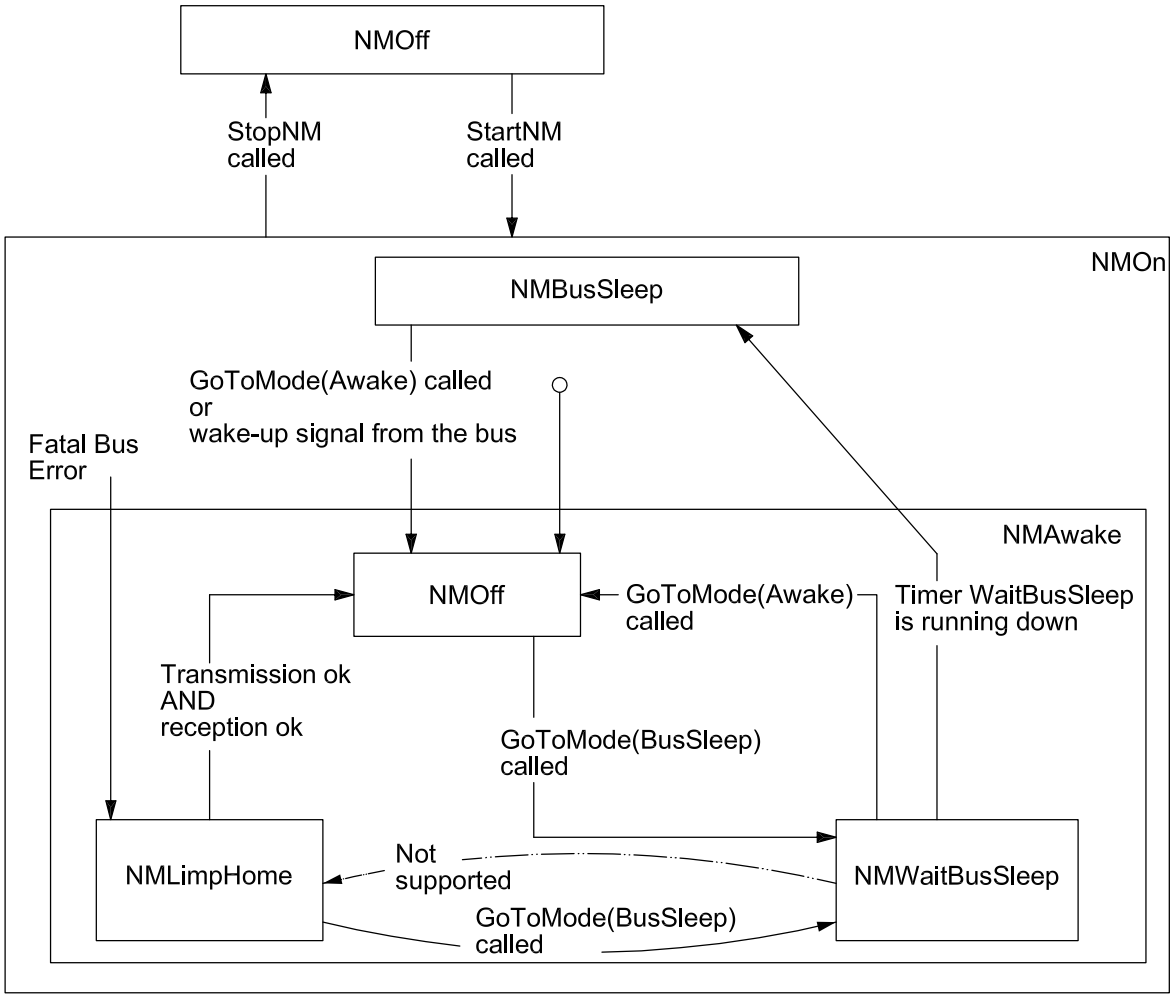


Figure 45 — Simplified state transition diagram of the indirect NM

3.2.5.4.2 NMLimpHome

This state is entered after a failure of the network communication interface, communication not being operational (e.g. Bus-Off failure for CAN).

Node states values (e.g. “node absent”) do not switch NM to the state NMLimphome. NM only performs monitoring actions, but has no knowledge about the expected target configuration. NM does not know if a missing node is a failure or not.

3.2.5.4.3 NMWaitBusSleep

This state is entered after the demand of the application for entering the BusSleep mode. It is a waiting state preparing for BusSleep mode. During this time, all other nodes shall receive the SleepMode command via their application (3.3.3.1) as well.

3.2.6 Operating modes

The NM does not manage application modes, but exclusively manages NM operating modes. NM distinguishes two main operating modes. The modes of the NM are directly mapped to internal NM states. These include:

— NMAwake: In NMAwake, the node monitors the selected application messages.

— NMBusSleep: If a node is in NMBusSleep, it does not monitor application messages. Depending on the hardware integrated in the networks, nodes can switch into some low power mode.

The NM provides services for selection of NM operation modes and indication of NM operating modes.

3.3 Algorithms and behaviour

3.3.1 Configuration management

The NM supports the configuration and the optional extended configuration management. The extended configuration is specified by monitoring application messages with a “long” time. This “long” time is realized by using counters.

3.3.2 Counter management

The states of the extended configuration are determined by decrementing and incrementing specific counters and by comparing the counters with a threshold. The functions used to increment and decrement shall avoid any overflow and underflow. From the point of view of the functionality, one of the values is redundant and can be selected statically. Therefore, NM sets the threshold to a constant value.

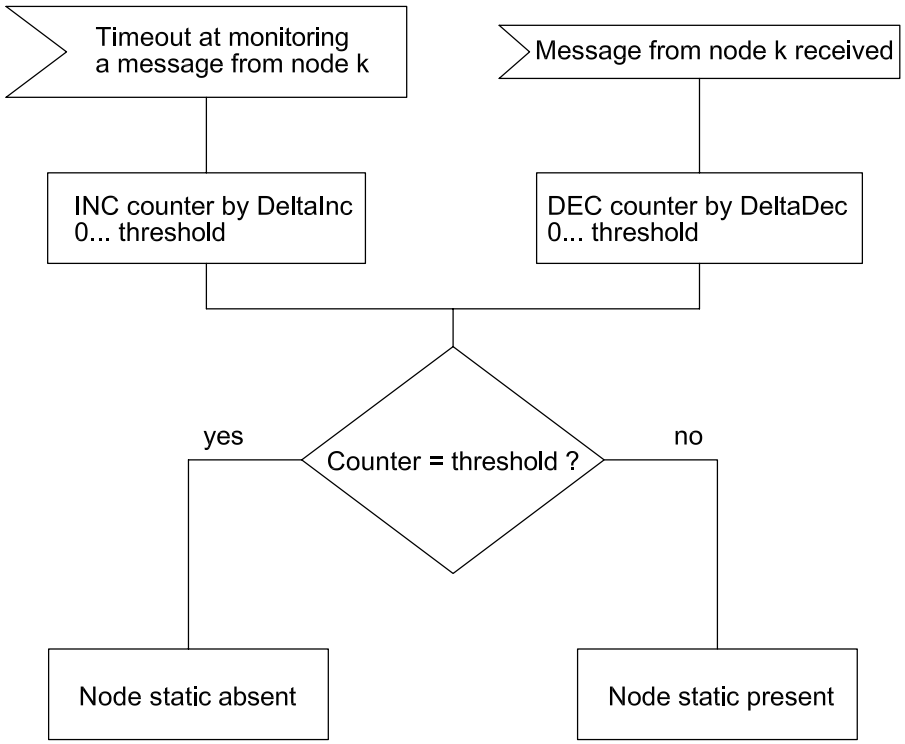


Figure 46 — Extended configuration illustrated at node k

Counter behaviour and corresponding states are illustrated by Figures 47-49.

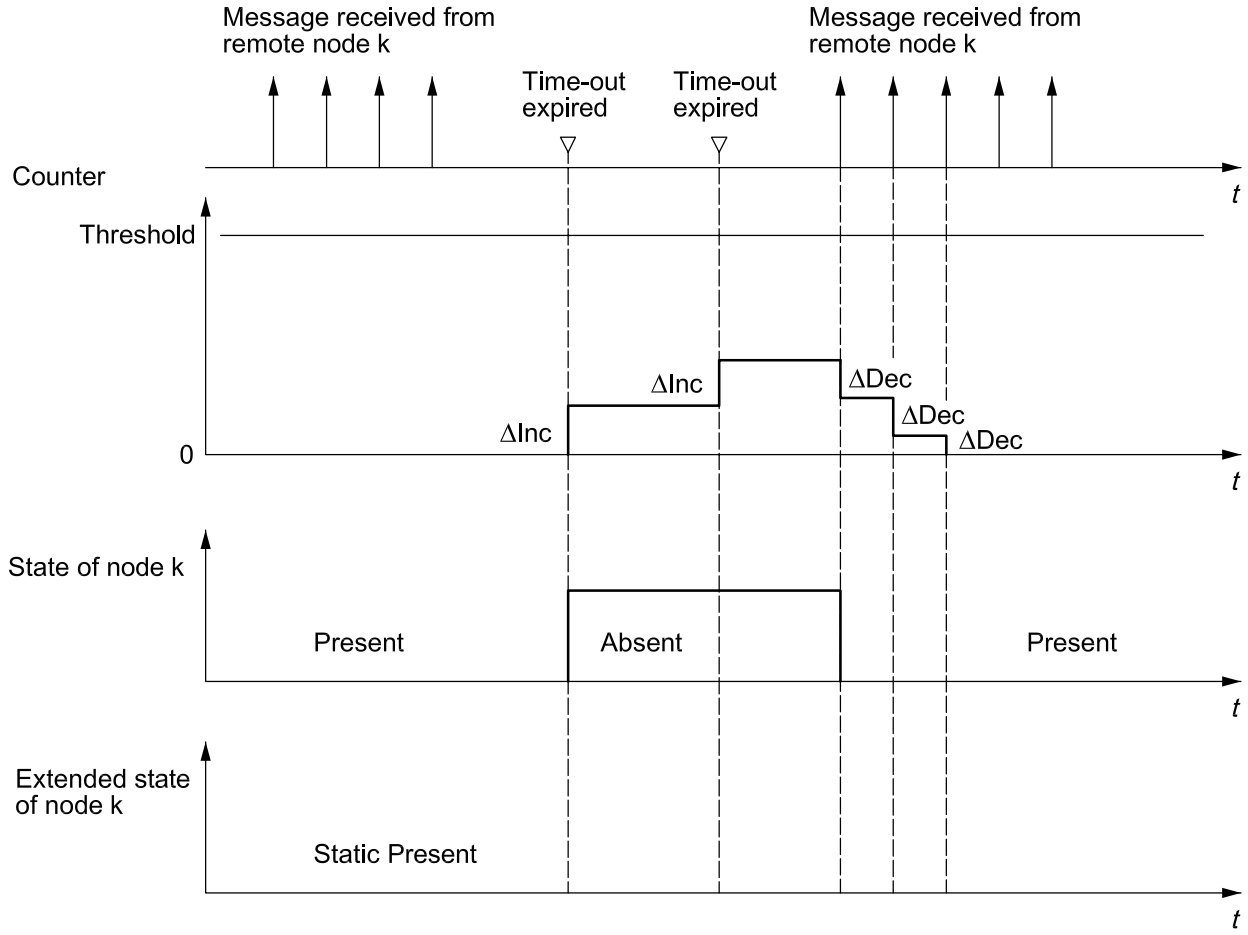


Figure 47 — Extended configuration illustrated at node k in the case of a very transient state of the node in which the state "Static Absent" will not be reached

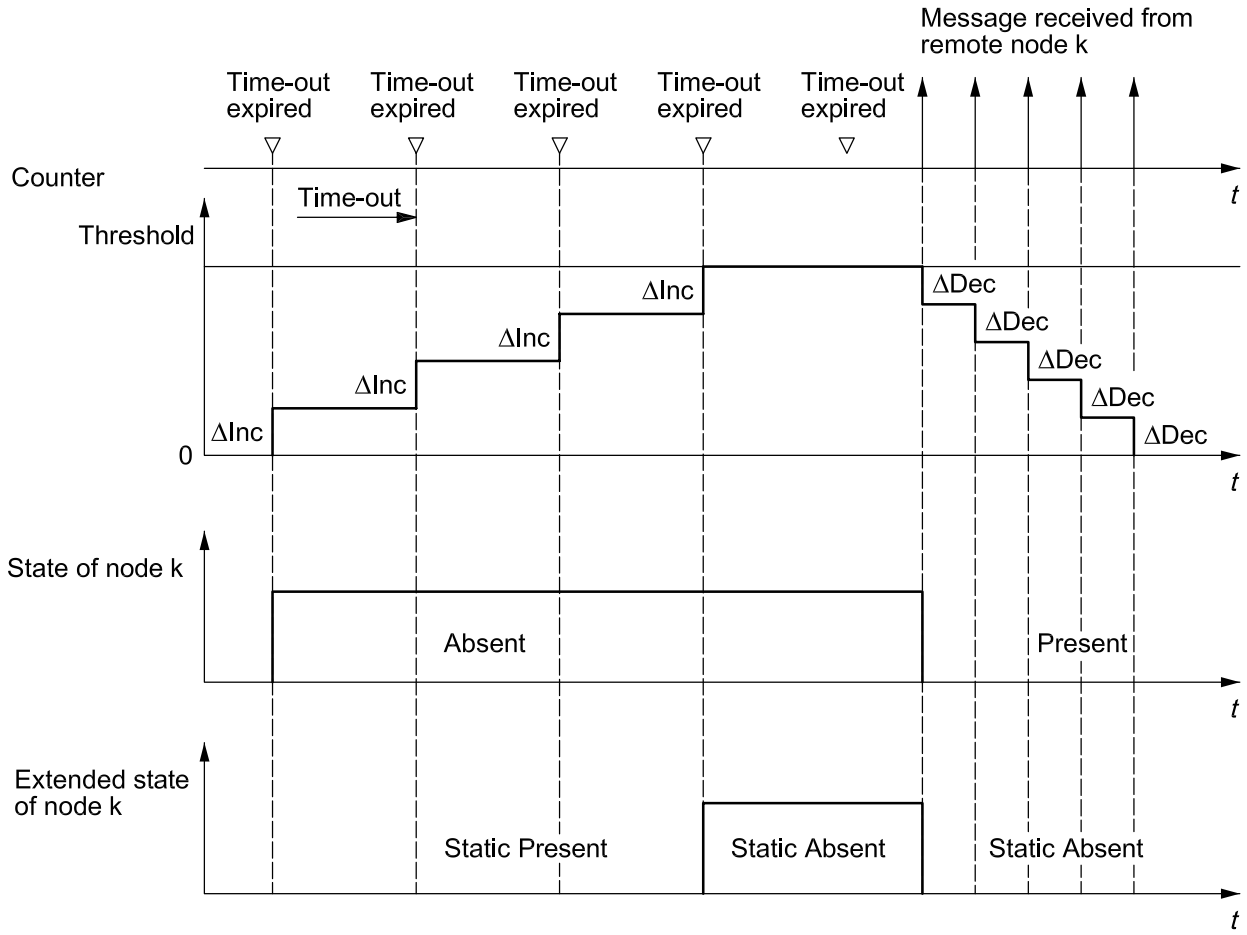


Figure 48 — Extended configuration illustrated at node k in the case of a permanent state of the node

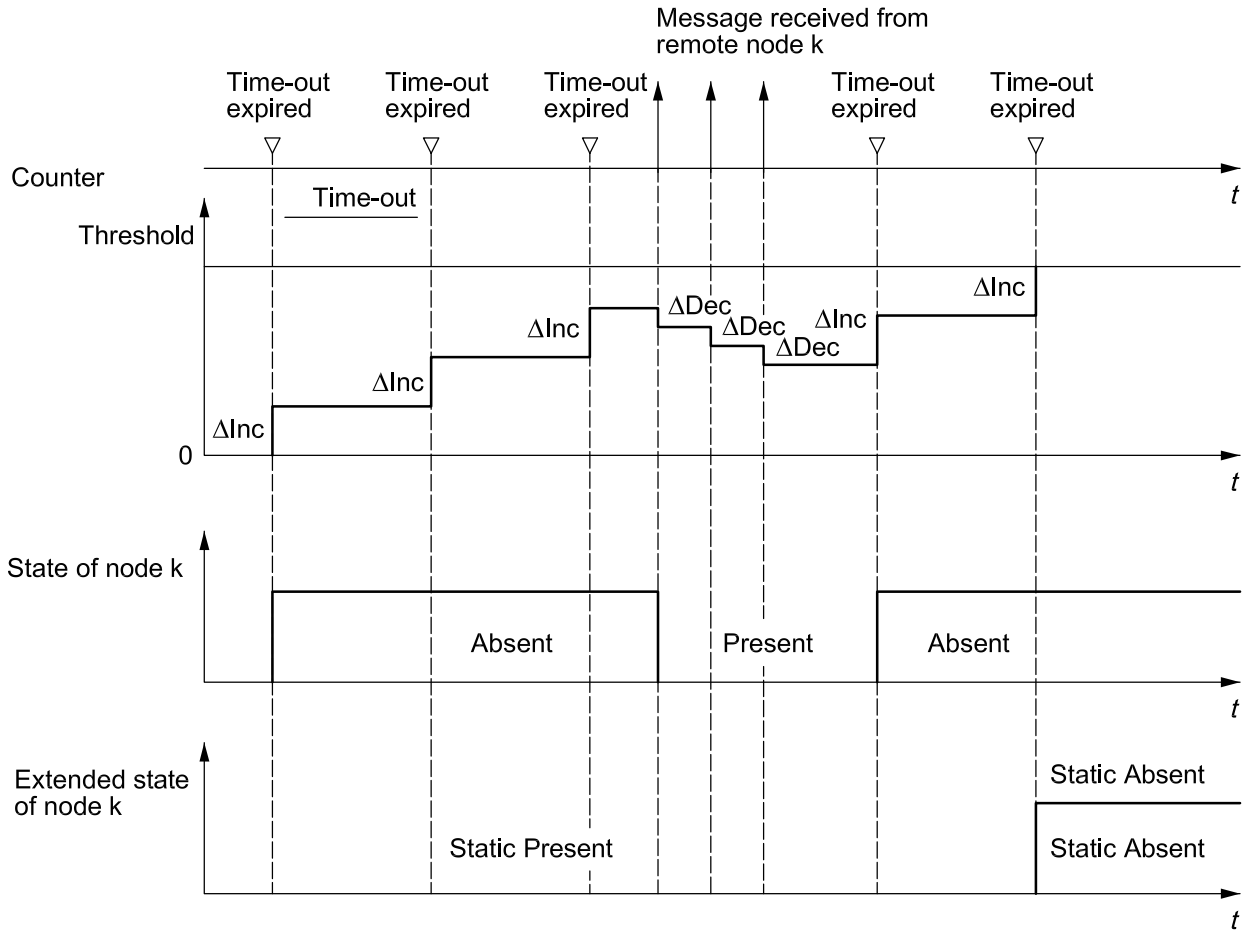


Figure 49 — Extended configuration illustrated at node k in case of a repetitive state of the node

Indirect NM static state detection algorithm is flexible and scaleable. It allows choosing different kinds of detection for static states by setting the parameters DeltaInc and DeltaDec at system generation time.

3.3.3 Operating mode

3.3.3.1 User Guide to handle BusSleep

The NM handles power down modes on demand of the user. Net-wide negotiations are not supported. Master-slave and multi-master behaviour can be realized by using the given services GotoMode(Awake) and GotoMode(BusSleep).

For example, in master-slave behaviour, the user reserves one bit in an application message which the master broadcasts to the slaves.

There are two possibilities:

— bit is set: the master requires the mode NMBusSleep from all slaves;

— bit is cleared: the master does not require the mode NMBusSleep from any slave.

Table 10 — Example of the application behaviour to handle NMAwake and NMBusSleep according to a master-slave approach

Application	NMAwake	NMBusSleep
In the master	set the reserved bit and send the corresponding message call GoToMode(BusSleep) after the message has been sent via the bus	call GoToMode(Awake) clear the reserved bit and send the corresponding message
In the slave	call GoToMode(BusSleep) when receiving the set bit call GoToMode(Awake) when receiving the cleared bit	-

NOTE The master and the slave behaviours can be supported by a single implementation of the indirect NM.

3.3.4 State machine in SDL

3.3.4.1 SDL model for one global time-out TOB

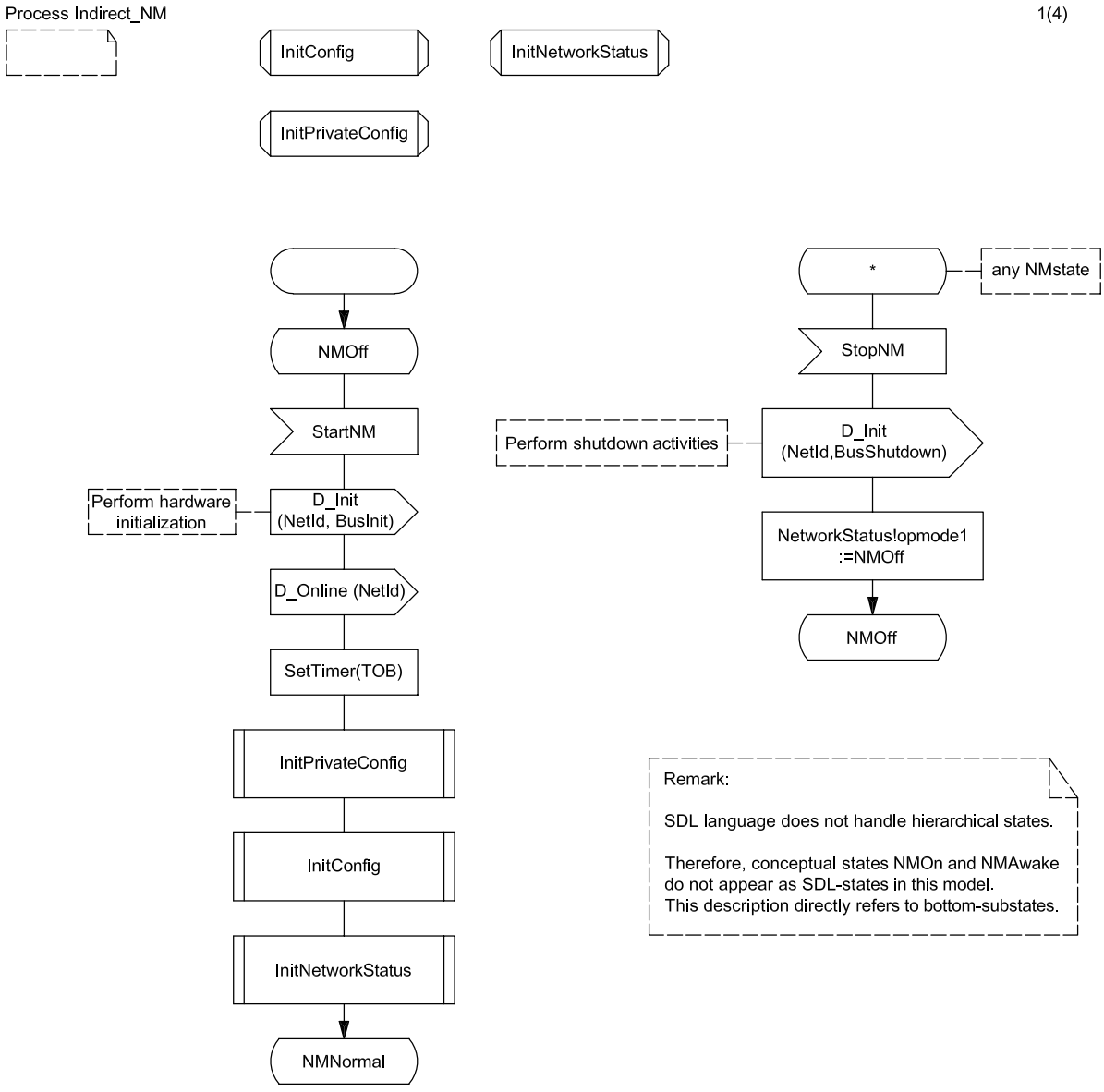


Figure 50 — Handling of the services StartNM and StopNM

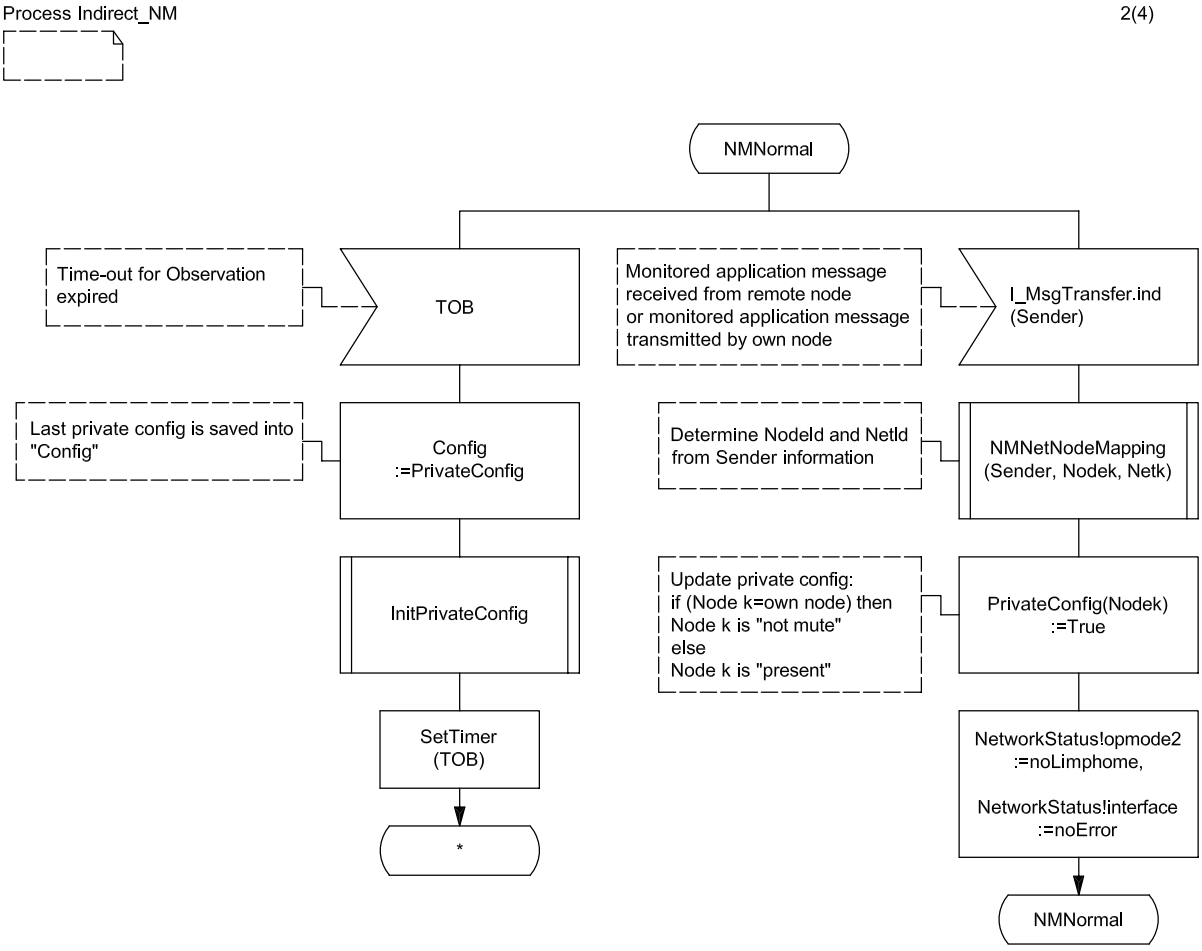


Figure 51 — Handling of the events “TOB” and “message received” during state NMNormal

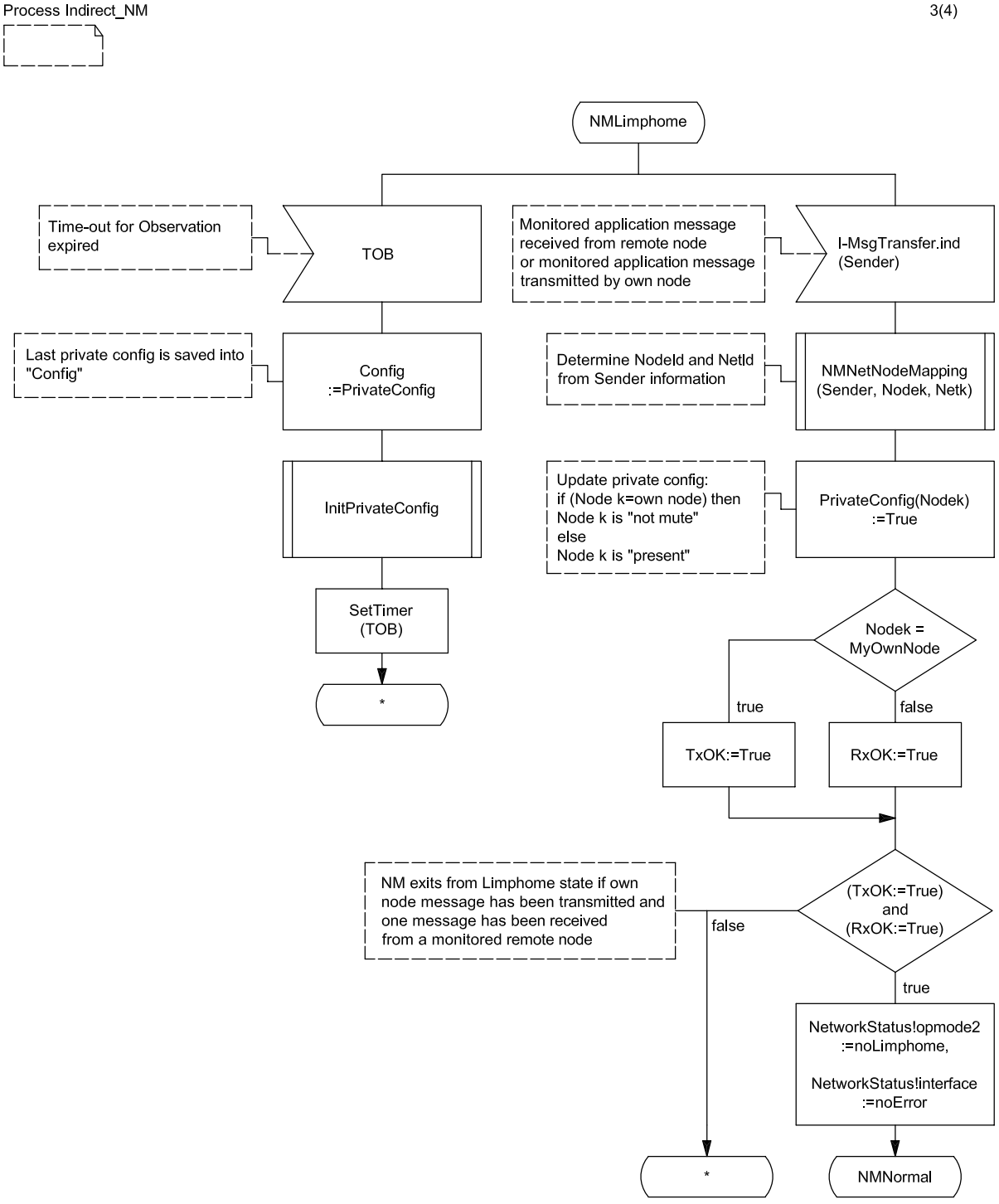


Figure 52 — Handling of the events “TOB” and “message received” during NMLimpHome

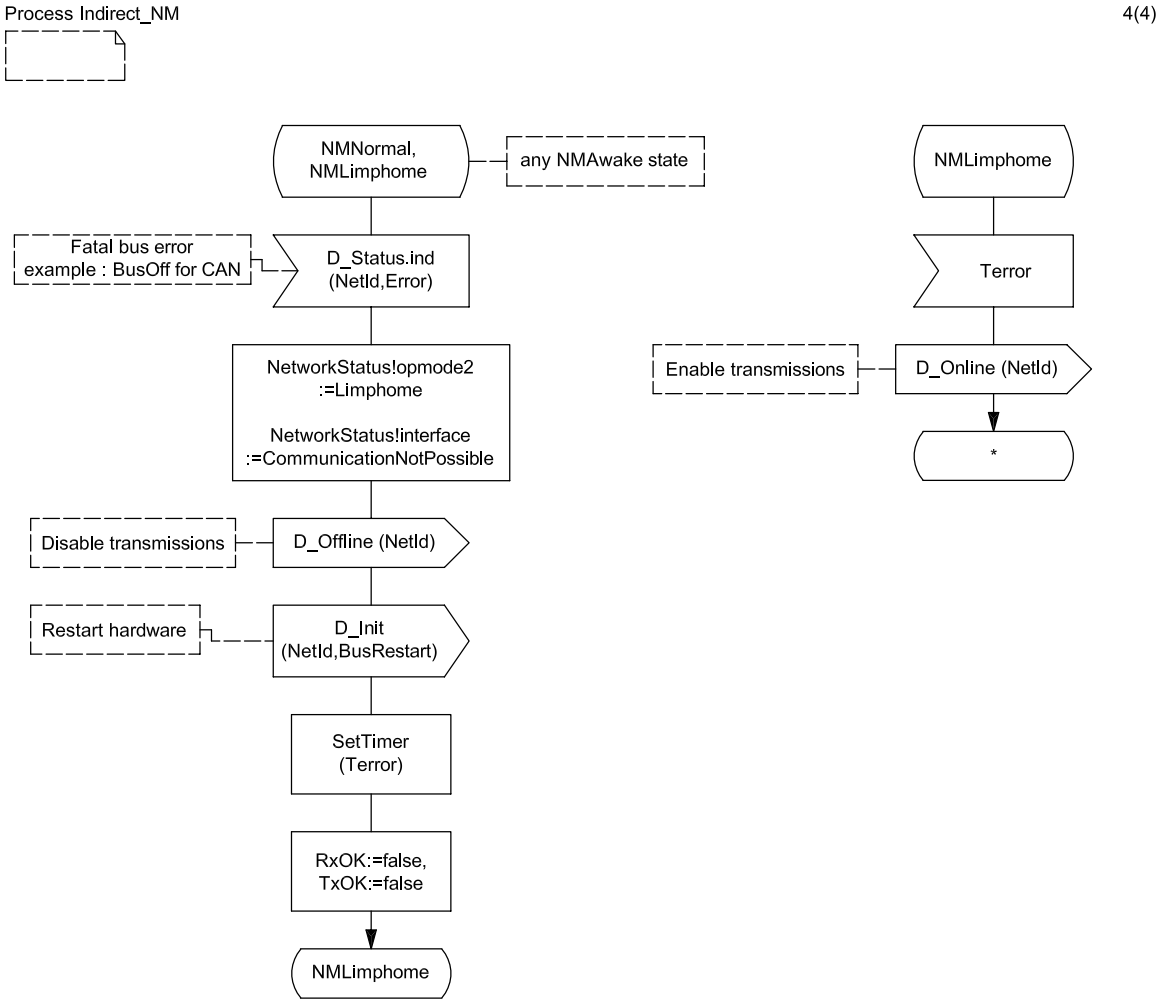


Figure 53 — Handling of a fatal bus error

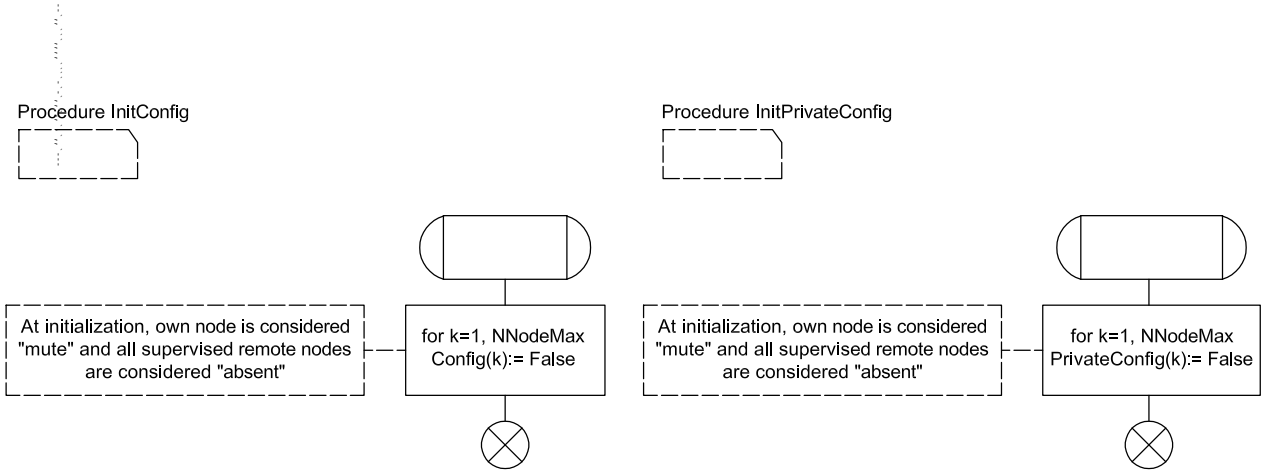


Figure 54 — Initialization of the configuration

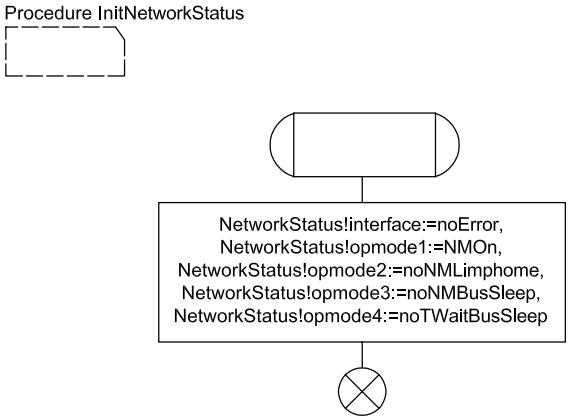


Figure 55 — Initialization of the NM status

3.3.4.2 SDL model for one monitoring time-out per message

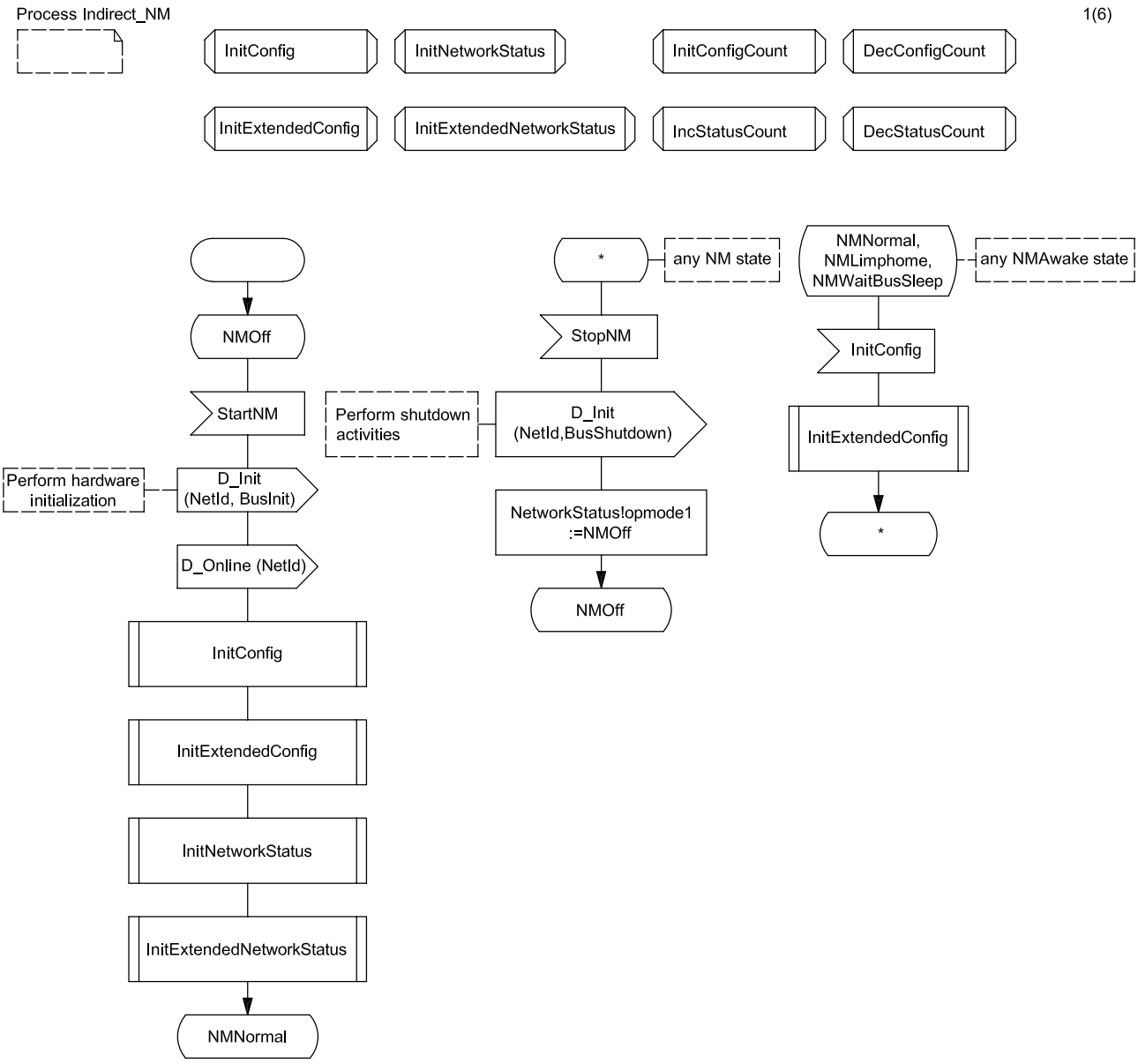


Figure 56 — Handling of the services StartNM, StopNM and InitConfig

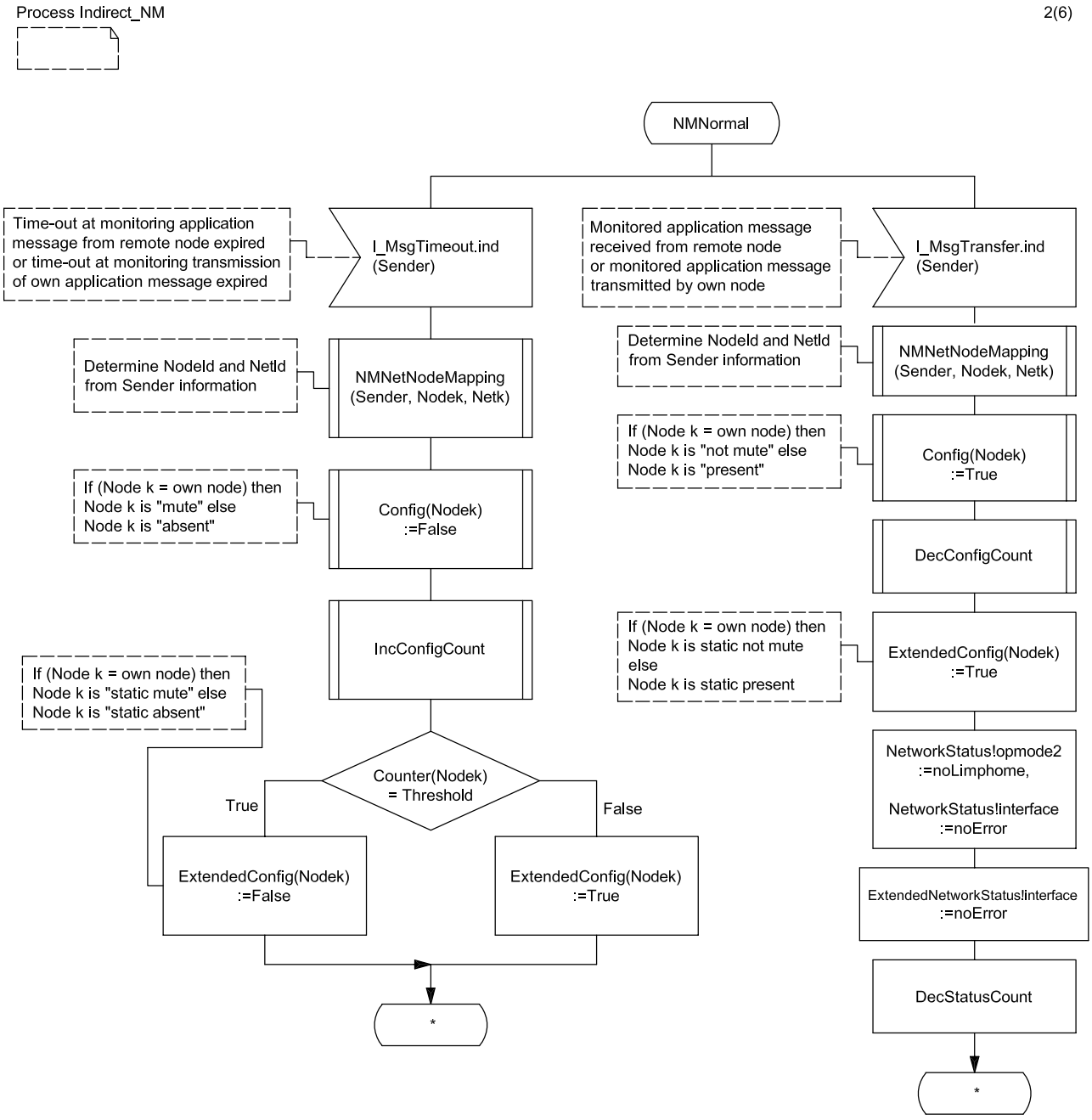


Figure 57 — Handling of the events “timeout for message” and “message received” during state NMNormal

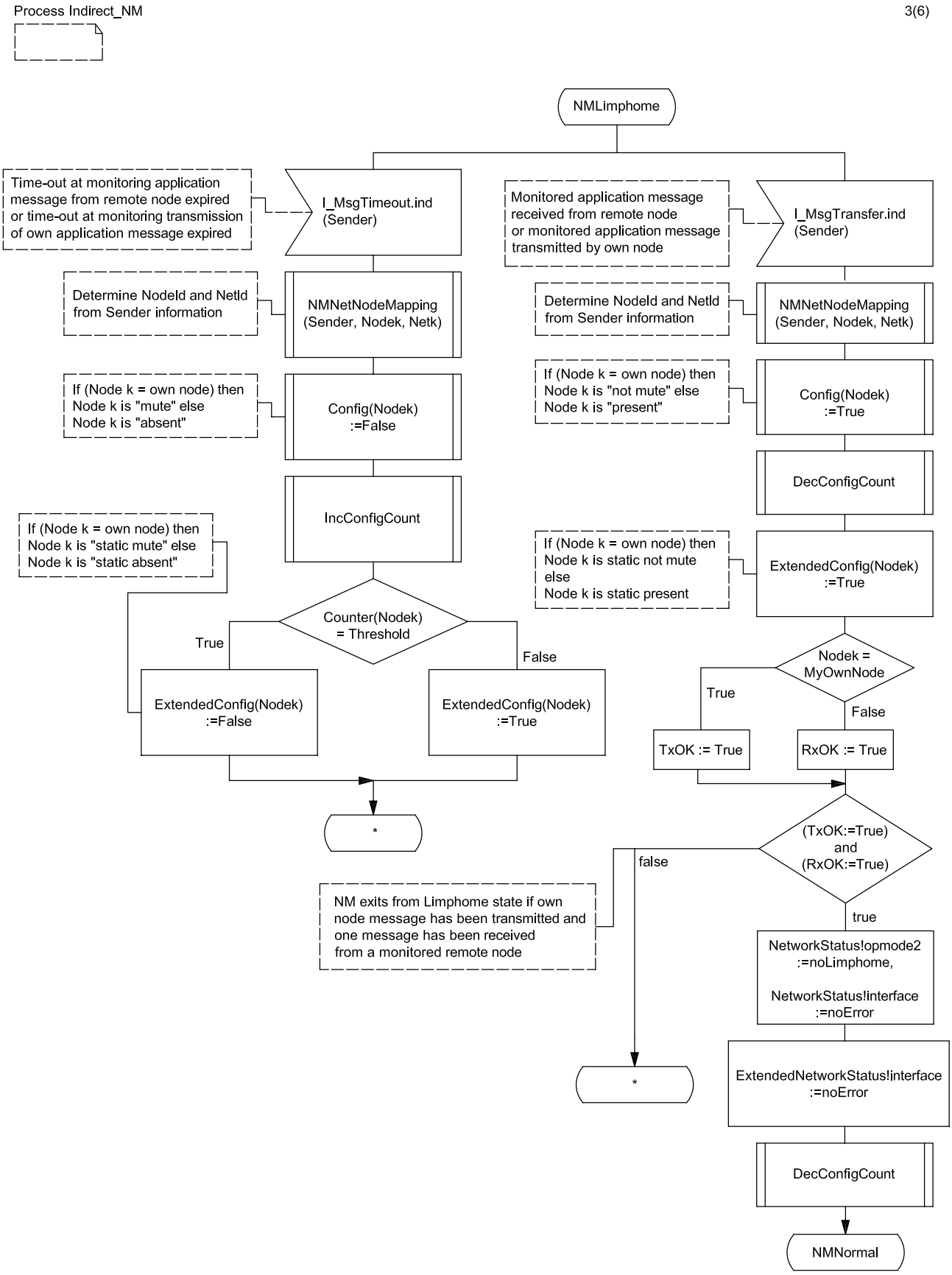


Figure 58 — Handling of the events “timeout for message” and “message received” during state NMLimpHome

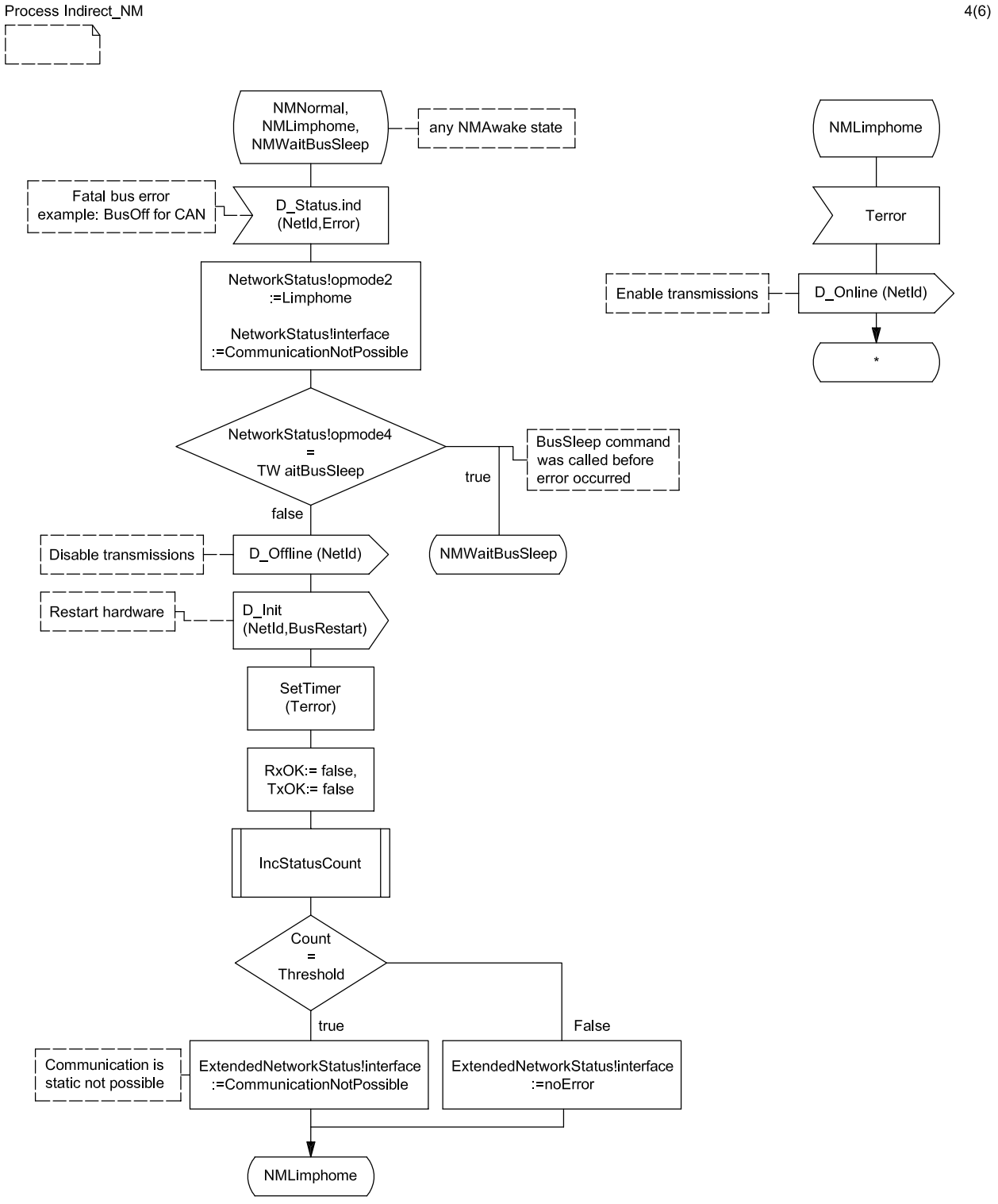


Figure 59 — Handling of a fatal bus error

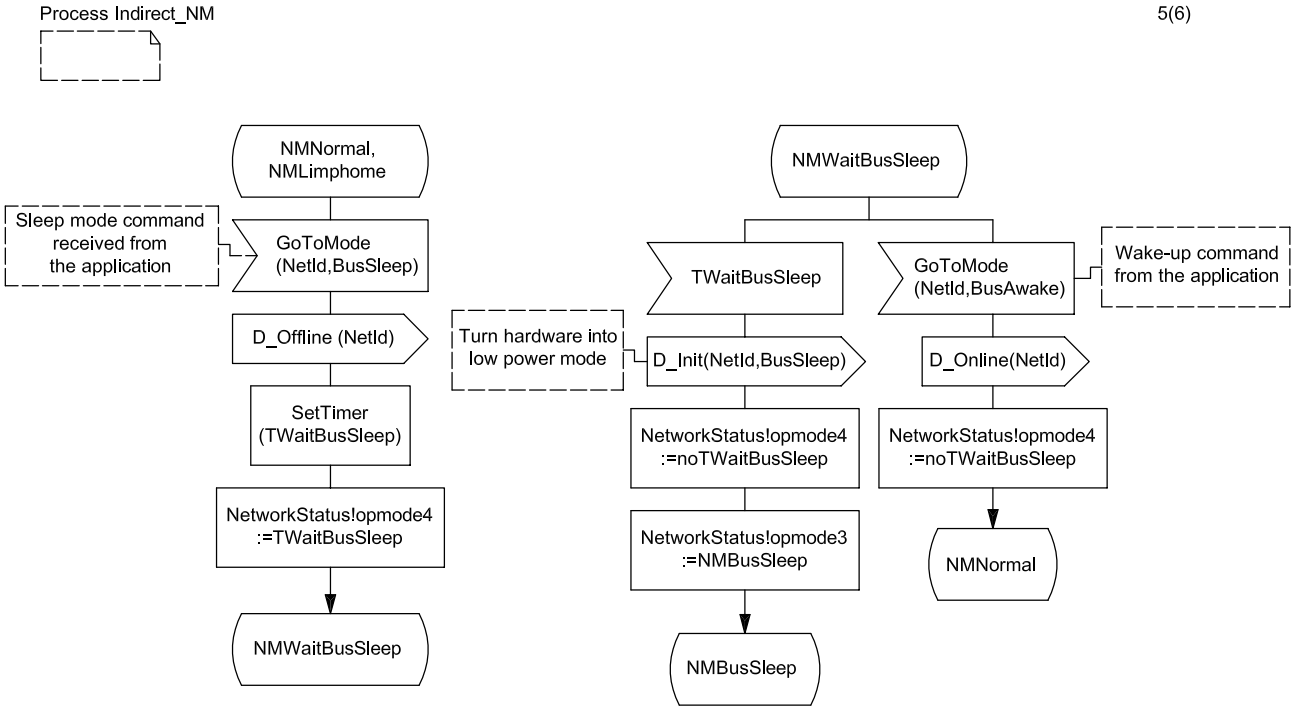


Figure 60 — Handle the transition to the state NMBusSleep

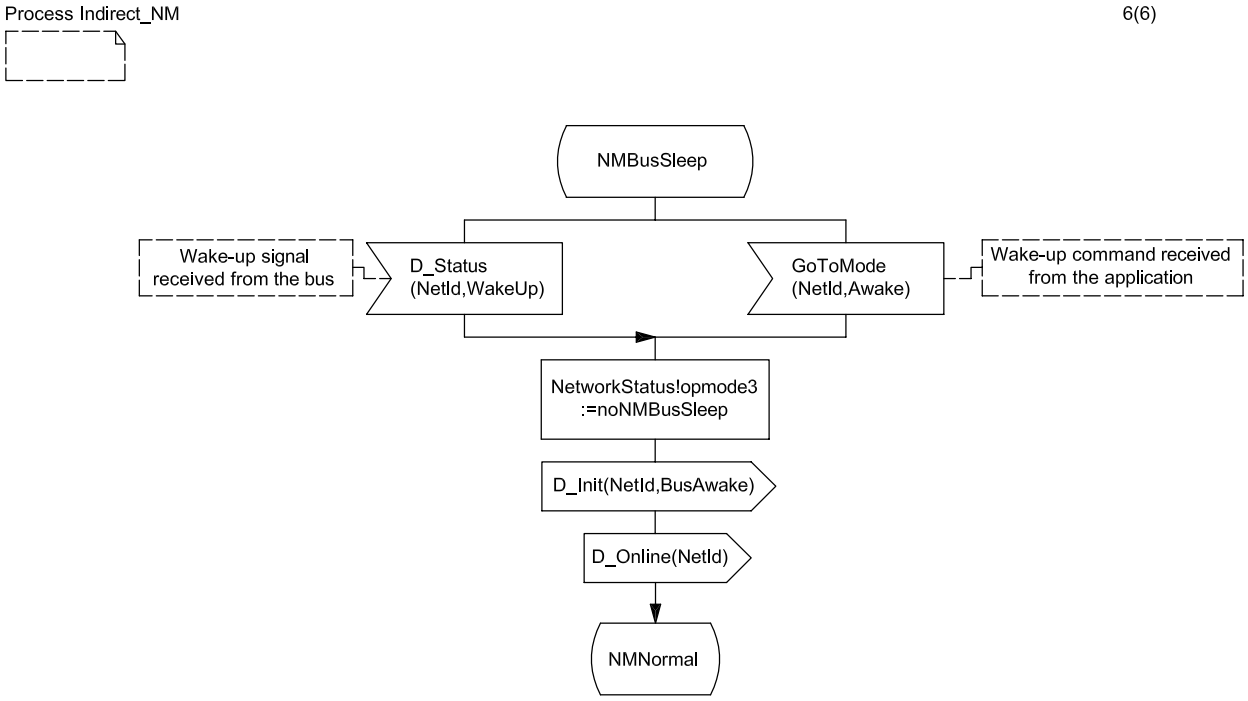


Figure 61 — Handle the transition from NMBusSleep into NMNormal

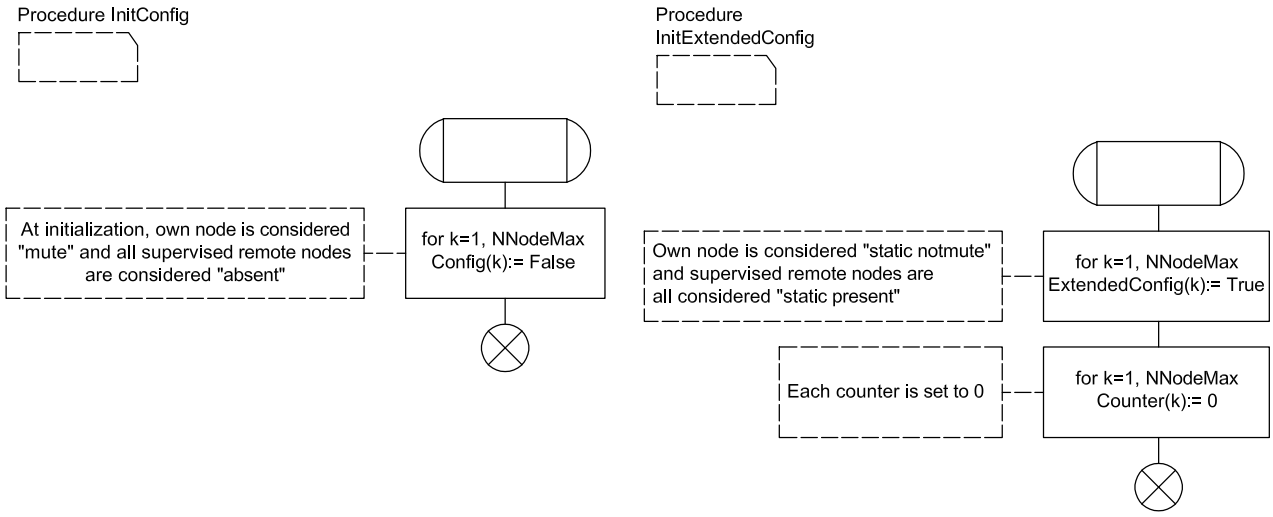


Figure 62 — Initialization of the configuration

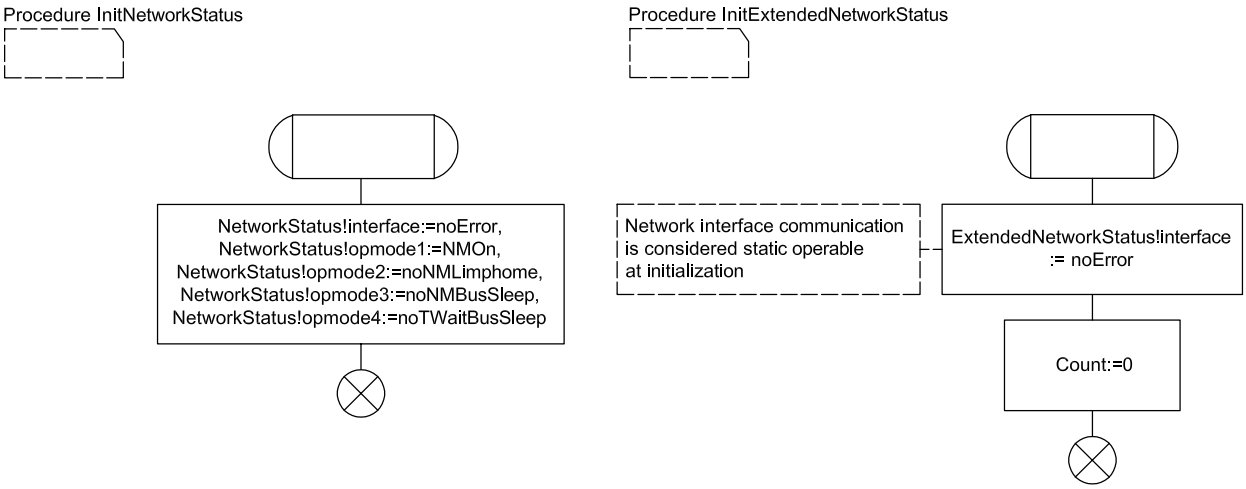


Figure 63 — Initialization of the NM status

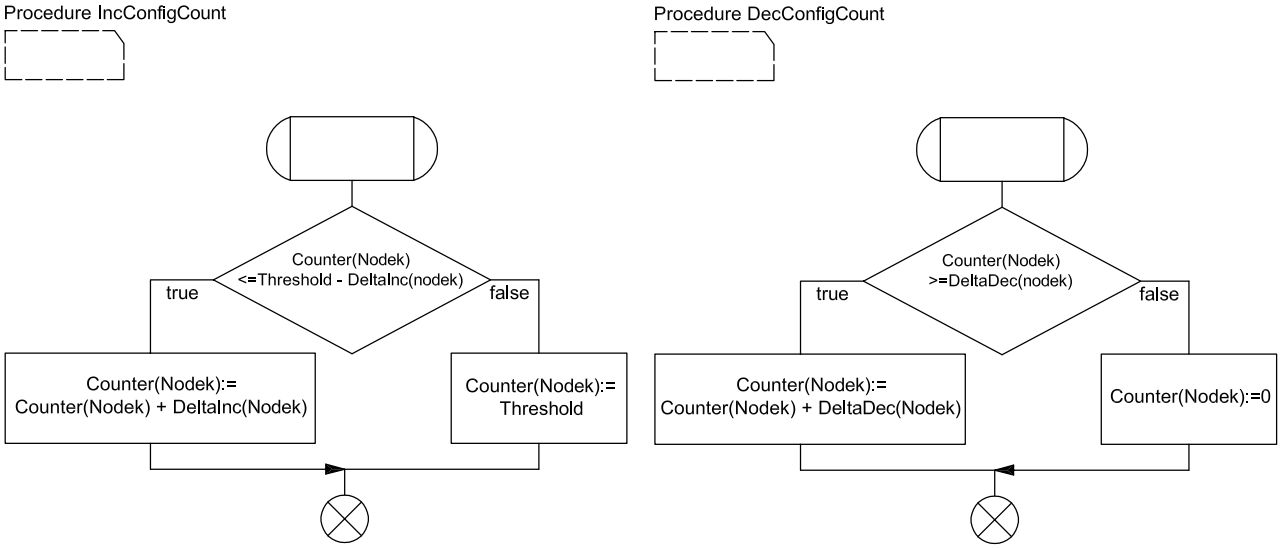


Figure 64 — Decrement and increment procedures for the extended configuration

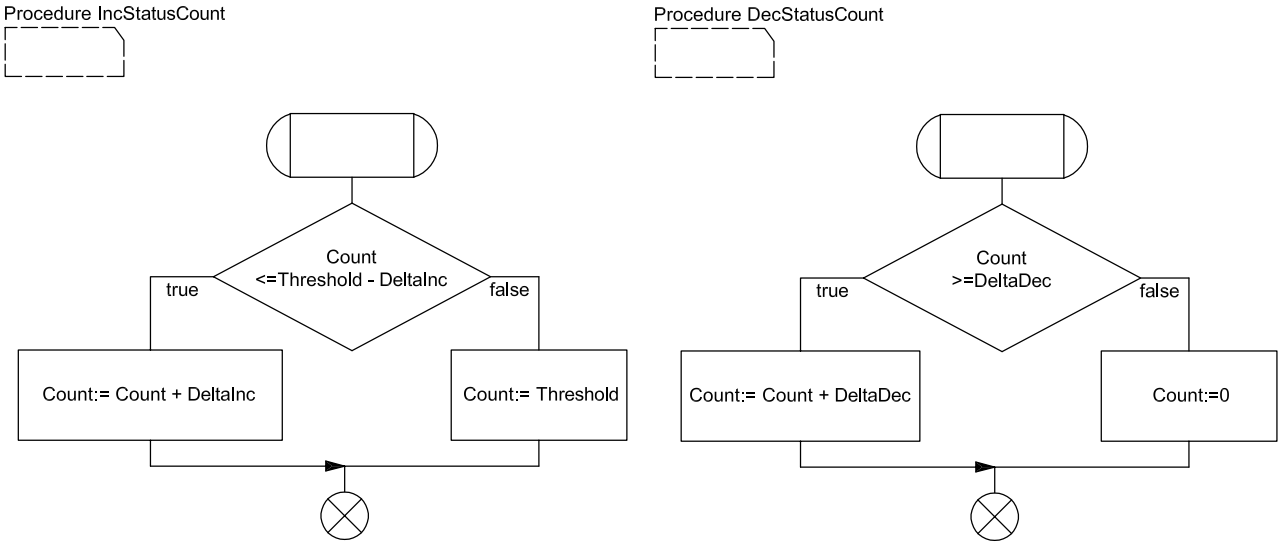


Figure 65 — Decrement and increment procedures for the extended network status

4 System generation and API

4.1 Overview

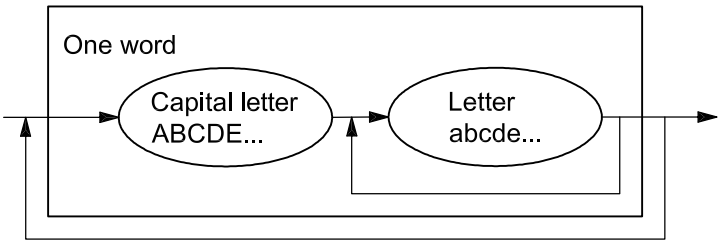


Figure 66 — Overview of an NM service (example GetConfig)

Table 11 — Breakdown of NM API-services into core services and optional services

	Service Call	dir. NM	ind. NM	ISL	Hook Level					Task Level	Core or Optional Service
				ISR	ErrorHook	PreTaskHook	PostTaskHook	StartHook	ShutdownHook	Task	
Configuration management											
Initialization of static parameter	**InitDirectNMParams** **InitIndirectNMParams**	✓	✓								OPT
Initialization of individual configurations masks	**InitCMaskTable**	✓	✓								OPT
Initialization of individual target configurations	**InitTargetConfigTable**	✓	✓								OPT
Indication of configuration change	**InitIndDeltaConfig**	✓	✓								OPT
Select the indication sensitivity	**SelectDeltaConfig**	✓	✓	✓						✓	OPT
Start or restart the configuration management	**InitConfig**	✓	✓	✓						✓	OPT
Make current configuration available	**GetConfig**	✓	✓	✓						✓	OPT
Comparison of two configurations	**CmpConfig**	✓	✓	✓						✓	OPT
Operating mode management											
Initialization of individual status masks	**InitSMaskTable**	✓	✓								OPT
Initialization of individual target states	**InitTargetStatusTable**	✓	✓								OPT
Indication of status change	**InitIndDeltaStatus**	✓	✓								OPT
Start of NM, i.e. transition to NM mode “NMON”.	**StartNM**	✓	✓	✓						✓	CORE
Stop of NM, i.e. transition to NM mode “NMShutDown” and finally to “NMOff”	**StopNM**	✓	✓	✓						✓	CORE
Transition to NM mode “NMPassive” without network-wide notification	**SilentNM**	✓		✓				✓		✓	OPT
Transition to NM mode “NMActive” after a previous call of SilentNM	**TalkNM**	✓		✓				✓		✓	OPT
Transition to a new operating mode (e.g. BusSleep, Awake)	**GotoMode**	✓	✓	✓				✓		✓	OPT
Select the indication sensitivity	**SelectDeltaStatus**	✓	✓	✓						✓	OPT
Get status information (network, node)	**GetStatus**	✓	✓	✓						✓	OPT
Comparison of two states	**CmpStatus**	✓	✓	✓						✓	OPT
Data Field management											
Indication of received data	**InitIndRingData**	✓									OPT
Transmit data	**TransmitRingData**	✓		✓						✓	OPT
Read received data	**ReadRingData**	✓		✓						✓	OPT

✓ Call to the NM service is allowed in this level (Interrupt level ISL, Hook level and Task level).

4.2 Conventions for service description

4.2.1 System generation

Within NM, all system objects shall be determined statically by the user (fixed at compile time). There are no system services available to dynamically create system objects.

System objects shall be defined or declared for usage in the application programs' source using specific calls.

The design of system objects may require additional specific tools. They enable the user to add or to modify values which have been specified. Consequently, the system generation and the tools are also implementation-specific.

4.2.2 Type of calls

System services are called according to the ANSI-C syntax. The implementation is normally a function call, but may also be solved differently, as required by the implementation, e.g. by C-pre-processor macros.

4.2.3 Error characteristics

All system services return a status to the user. The return status is E_OK if it has been possible to execute the system service without any restrictions. If the system recognizes an exceptional condition which restricts execution of any system service, a different status is returned.

If it is possible to exclude some real errors before run time, the run time version may omit the checking of these errors. If the only possible return status is E_OK, the implementation is free not to return a status.

To keep the system efficient and fast, NM does not check for all possible errors. NM assumes debugged applications and the correct usage of the system services. It is expected that undetected errors in the application result in undefined system behaviour.

All return values of a system service are listed under the individual descriptions. The return status distinguishes between the "Standard" and the "Extended" status. The "Standard" version fulfils the requirements of a debugged application system as described before. The "Extended" version is considered to support testing of not yet fully debugged applications. It comprises extended error checking compared to the standard version.

The sequence of error checking within the NM module is not specified. If multiple errors occur, the status returned depends on the implementation.

In case of fatal errors, the system service does not return to the application. Fatal error treatment is performed by the operating system.

4.2.4 Structure of the description

4.2.4.1 General

The descriptions of NM services are logically grouped. A coherent description is provided for all services of the configuration management, the management of operating modes and data field management.

The description of each of these logical groups starts with a description of the data types defined for the group. That section is followed by a description of the group specific system generation support and subsequently the run time services are described.

4.2.4.2 System generation support

The description of system generation actions comprises the following fields:

Name: Name of system generation action

Syntax: Call interface in C syntax

Parameter (In): List of all input parameters

Description: Explanation of the function

Particularities: Explanation of restrictions relating to the utilization

4.2.4.3 Service descriptions

A service description comprises the following fields:

Service name: Name of NM service

Syntax: Interface in ANSI-C syntax. The return value of the service is always of data type StatusType.

Parameter (In): List of all input parameters

Parameter (Out): List of all output parameters. Strictly speaking, transfers via the memory use the memory reference as input parameter and the memory contents as output parameter. To clarify the description, the reference is already specified with the output parameters.

Description: Explanation of the functionality of NM service

Particularities: Explanation of restrictions related to the use of NM service

Status: List of possible return values:

- Standard: List of return values provided in NM standard version;
- Extended: List of additional return values in NM extended version.

4.3 General data types

Table 12 — General data types

General data types	Remark
NodeIdType	Type for references to several nodes
NetIdType	Type for references to several communication networks
RoutineRefType	Type for references to low level routines
EventMaskType	Type for references to event masks
SignallingMode	Unique name defining the mode of signalling. Legal names are: “Activation”, "Event".
StatusType	Type[a] of returned status information
TaskRefType	References to tasks
TickType	This type represents count values in ticks
[a] Using the common definition from ISO 17356 parts 3,4 and 5.	

4.4 Common services

4.4.1 Standard functionality

4.4.1.1 System generation support

In general, the system designer shall select a NM which fits his needs. The selected NM can be scaled and shall be parameterized.

EXAMPLE The system designer selects a special implementation of the direct NM which guarantees a minimal calculating power demand. He decides to do it without using any scaling features. He concludes by fixing the parameter of the NM.

The services to support the system designer are the reflection of the know-how of a software vendor. The following proposals should give an idea how system generation could be handled.

Name: InitNMType

Syntax: InitNMType (NetIdType <NetId>,
NMType <NMType>

Parameter (In):
NetId: Addressed communication network

NMType: selected NM (e.g. direct or indirect)

Description: *InitNMType* is a directive to select a NM from a given set of NM implementations.

Particularities: none

Name: InitNMScaling

Syntax: InitNMScaling (NetIdType <NetId>,
ScalingParamType <ScalingParams>

Parameter (In):
NetId: Addressed communication network

ScalingParams: Set of parameter to scale the given NM

Description: *InitNMScaling* is a directive for scaling the given NM of the referenced net (e.g. the state NMBusSleep is supported or the state NMBusSleep is not supported).

Particularities: none

Name: SelectHWRoutines

Syntax: SelectHWRoutines (NetIdType <NetId>,
RoutineRefType <BusInit>,
RoutineRefType <BusAwake>,
RoutineRefType <BusSleep>
RoutineRefType <BusRestart>
RoutineRefType <BusShutDown>

Parameter (In):
NetId: Addressed communication network

BusInit: Referenced routine to initialize the bus hardware once at the start of the network

BusAwake: Referenced routine to reinitialize the bus hardware to leave the power down mode

BusSleep: Referenced routine to initialize the power down mode of the bus hardware

BusRestart: Referenced routine to restart the bus hardware in the case of a fatal bus error

BusShutDown: Referenced routine to shut down the bus hardware

Description: *SelectHWRoutines* is a directive to select routines from a given set of routines to drive the bus hardware.

Particularities: none

Figure 67 shows the routines to initialize, restart and shut down the bus hardware. The routines depend on the given hardware design and on the behaviour of the NM which the application requires.

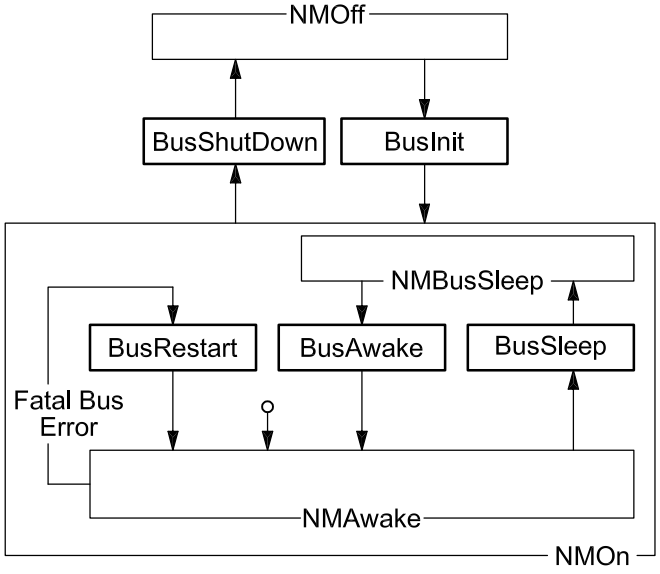


Figure 67 — Routines to initialize, restart and shut down the bus hardware

4.4.2 Configuration management

4.4.2.1 Data types

Table 13 — Special data types of the configuration management

NM data types	Remark
ConfigRefType	This data type represents the reference of a configuration.
ConfigKindName	Unique name defining the requested kind of configuration: “Normal” — supported by direct and indirect NM “Normal extended” — only supported by indirect NM "LimpHome" — only supported by direct NM
ConfigHandleType	This data type represents a handle to reference values of the type ConfigRefType.

4.4.2.2 System generation support

Name: InitCMaskTable

Syntax: InitCMaskTable (NetIdType <NetId>,
ConfigKindName <ConfigKind>,
ConfigRefType <CMask>

Parameter (In):

NetId: Addressed communication network,

ConfigKind: Kind of configuration,

CMask: Configuration mask (list of relevant nodes).

Description: *InitCMaskTable* is a directive for initializing an element of a table of relevant configuration masks to be used by the signalling of changed configurations.

Particularities: none

Name: InitTargetConfigTable

Syntax: InitTargetConfigTable (NetIdType <NetId>,
ConfigKindName <ConfigKind>,
ConfigRefType <TargetConfig>

Parameter (In):

NetId: Addressed communication network,

ConfigKind: Kind of configuration,

TargetConfig: Target Configuration.

Description: *InitTargetConfigTable* is a directive for initializing an element of a table of relevant target configurations to be used by the signalling of changed configurations.

Particularities: none

Name: InitIndDeltaConfig

Syntax:
```
InitIndDeltaConfig (    NetIdType <NetId>,
                        ConfigKindName <ConfigKind>,
                        SignallingMode <SMode>,
                        TaskRefType <TaskId>,
                        EventMaskType <EMask>)
```

Parameter (In):

NetId: Addressed communication network

ConfigKind: Kind of configuration

SMode: Mode of signaling

TaskId: Reference to the task to be signalled

EMask: Mask of the events to be set

Description: *InitIndDeltaConfig* is a directive for specifying the indication of configuration changes. The concerned configuration is specified by <ConfigKind>.

The parameter <SMode> specifies whether task activation (SMode = Activation) or event signalling (SMode = Event) is used for indication.
In case of task activation, <TaskId> contains a reference of the task to be activated if the configuration <ConfigKind> has changed.
In case of event signalling <EMask> specified the event to be set for task <TaskId>, if the configuration <ConfigKind> has changed.

Particularities: none

Name: InitSMaskTable

Syntax:
```
InitSMaskTable      (NetIdType <NetId>,
                     StatusRefType <SMask>
```

Parameter (In):

NetId: Addressed communication network

SMask: status mask (list of relevant network states)

Description: *InitSMaskTable* is a directive for initializing an element of a table of relevant status masks to be used by the signalling of changed network states.

Particularities: none

Name: InitTargetStatusTable

Syntax:
```
InitTargetStatusTable (    NetIdType <NetId>,
                           StatusRefType <TargetStatus>
```

Parameter (In):

NetId: Addressed communication network

TargetStatus: Target network status

Description: *InitTargetStatusTable* is a directive for initializing an element of a table of relevant target network states to be used by the signalling of changed network states.

Particularities: none

Name: InitIndDeltaStatus

Syntax: InitIndDeltaStatus (NetIdType <NetId>,
SignallingMode <SMode>,
TaskRefType <TaskId>,
EventMaskType <EMask>)

Parameter (In):

NetId	Addressed communication network
SMode	Mode of signalling
TaskId	Reference to the task to be signalled
EMask	Mask of the events to be set

Description: *InitIndDeltaStatus* is a directive for specifying the indication of status changes.

The parameter <SMode> specifies whether task activation (SMode = Activation) or event signalling (SMode = Event) is used for indication.
In case of task activation, <TaskId> contains a reference of the task to be activated if the status has changed.
In case of event signalling <EMask> specified the event to be set for task <TaskId>, if the status has changed.

Particularities: none

The extended network status is not supported by the proposed system generation.

4.4.2.3 Services

Service name: InitConfig

Syntax: StatusType InitConfig (NetIdType <NetId>)

Parameter (In):

NetId: Addressed communication network

Parameter (Out):

Description: This service makes the NM start or restart the configuration management. The service only works if the NM is in the state NMNormal. The service makes the NM leave the state NMNormal.

Particularities:

Status:

Standard: E_OK, no error

Extended: none

Service name: GetConfig

Syntax: StatusType GetConfig (NetIdType <NetId>,
ConfigRefType <Config>,
ConfigKindName <ConfigKind>)

Parameter (In):

NetId: Addressed communication network

ConfigKind: Kind of configuration

Parameter (Out):

Config: Configuration inquired

Description: This service provides the actual configuration of the kind specified by <ConfigKind>.

Particularities: The application shall provide the memory to transfer the configuration.

Status:

Standard: E_OK, no error

Extended: none

Service name: CmpConfig

Syntax: StatusType CmpConfig (NetIdType <NetId>,
ConfigRefType <TestConfig>,
ConfigRefType <RefConfig>,
ConfigRefType <CMask>)

Parameter(In):

NetId: Addressed communication network

TestConfig: Test configuration

RefConfig: Reference configuration

CMask: List of relevant nodes

Parameter (Out): none

Description: The test configuration <TestConfig> is compared to the specified reference configuration <RefConfig> taking account of the mask <CMask>.

The presence of a node in the network is identified within the test configuration and the reference configuration by TRUE. The relevance of the result of the comparison (<TestConfig> EXOR <RefConfig>) of the node within the network is identified within the <CMask> by TRUE.

Status = NOT (<CMask> AND
(<TestConfig> EXOR <RefConfig>))

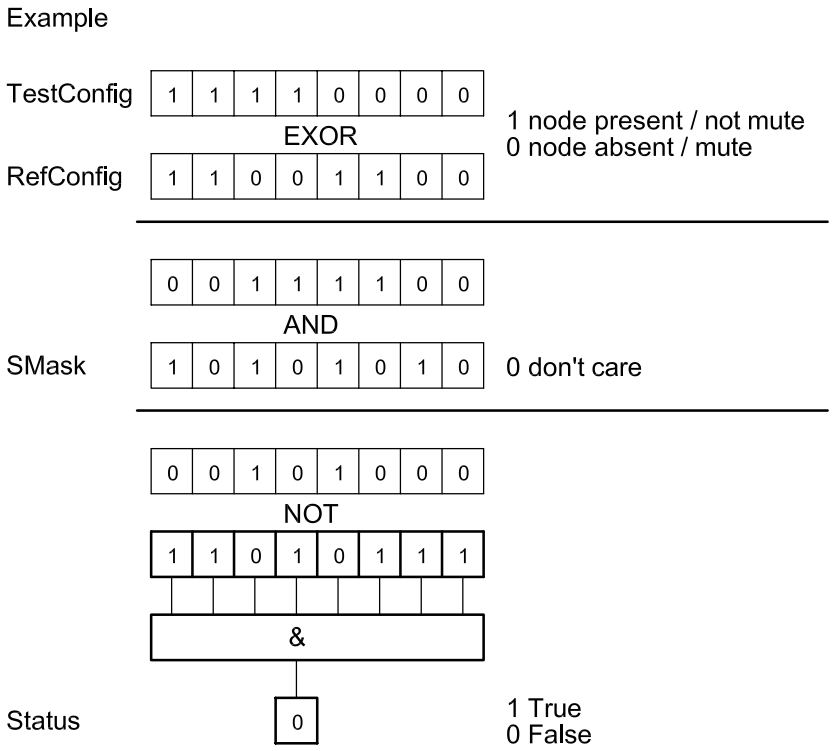


Figure 68 — Example of CmpConfig

Status:

Standard:

TRUE, test condition for specified mask exists

FALSE, else

Extended: none

Service name: SelectDeltaConfig

Syntax: StatusType SelectDeltaConfig (NetIdType <NetId>,
ConfigKindName <ConfigKind>),
ConfigHandleType <ConfigHandle>,
ConfigHandleType <CMaskHandle>

Parameter(In):

NetId: Addressed communication network

ConfigKind: Kind of configuration

ConfigHandle: Referenced target configuration

CMaskHandle: Referenced configuration mask

Parameter (Out): none

Description: A set of predefined parameter is selectable to drive the signalling of changed configurations.

Status: none

4.4.3 Operating modes and operating mode management

4.4.3.1 Data types

Table 14 — Special data types of the operating mode management

NM data types	Remark
NMModeName	Unique name defining the NM operational modes. Legal names are: “BusSleep” and “Awake”.
NetworkStatusType	Type of network status.
StatusHandleType	This data type represents a handle to reference values of the type StatusRefType.

4.4.3.2 Services

Service name: StartNM

Syntax: StatusType StartNM (NetIdType <NetId>)

Parameter (In):

NetId Addressed communication network

Parameter (Out): none

Description: *StartNM* starts the local Network Management. This causes the state transition from NMOff to NMOn.

Particularities: none

Status:

Standard: E_OK, no error

Extended: none

Service name: StopNM

Syntax: StatusType StopNM (NetIdType <NetId>)

Parameter (In):

NetId: Addressed communication network

Parameter (Out): none

Description: *StopNM* stops the local Network Management. This causes the state transition from NMOn to NMShutDown and after processing of the shutdown activities to NMOff.

Particularities: none

Status:

Standard: E_OK, no error

Extended: none

Service name: GotoMode

Syntax: StatusType GotoMode (NetIdType <NetId>,
NMModeName <NewMode>)

Parameter (In):

NetId: Addressed communication network

NewMode: NM operating mode to be set (only BusSleep, Awake)

Parameter (Out): none

Description: *GotoMode* serves to set the NM operating mode specified by <NewMode>. Operating modes to be set globally are recognized by the local NM and treated accordingly.

NOTE If a global operating mode has been set, the application, depending on the task specified by InitIndDeltaStatus, is informed accordingly.

Particularities: none

Status:

Standard: E_OK, no error

Extended: none

Service name: GetStatus

Syntax: StatusType GetStatus (NetIdType <NetId>,
NetworkStatusRefType <NetworkStatus>)

Parameter (In):

NetId: Addressed communication network

Parameter (Out):

NetworkStatus: requested Status of the node

Description: This service provides the current status of the network.

Particularities: none

Status:

Standard: E_OK, no error

Extended: none

Service name: CmpStatus

Syntax: StatusType CmpStatus (NetIdType <NetId>,
StatusRefType <TestStatus>,
StatusRefType <RefStatus>,
StatusRefType <SMask>)

Parameter(In):

NetId: Addressed communication network

TestStatus: Test status

RefStatus: Reference status

SMask: List of relevant states

Parameter (Out): none

Description: The test status <TestStatus> is compared to the specified reference status <RefStatus> taking account of the mask <SMask>.

Status= NOT (<SMask> AND
(<TestStatus> EXOR <RefStatus>))

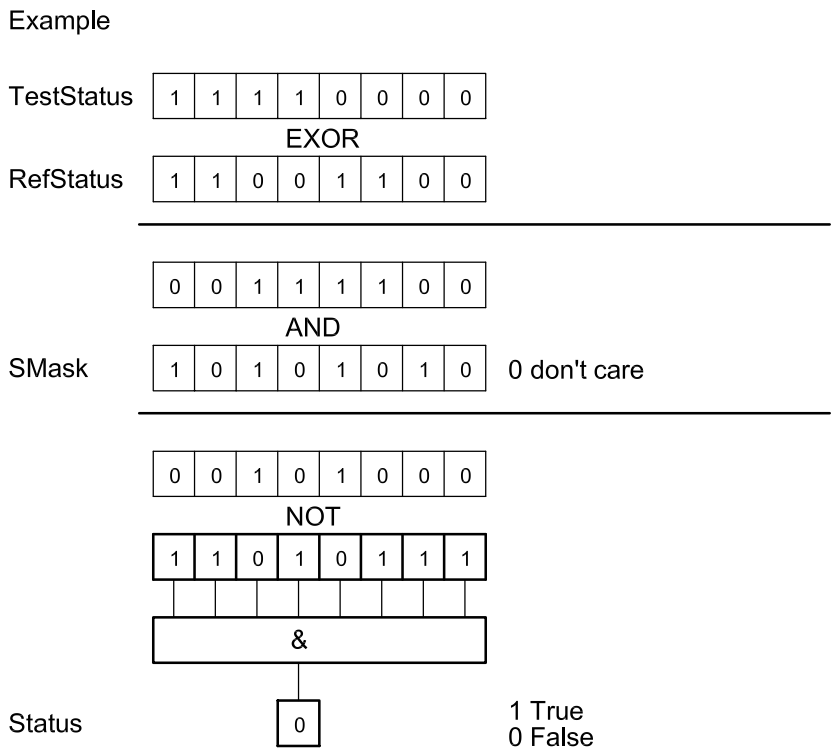


Figure 69 — Example of CmpConfig

Status:

Standard: TRUE, test condition for specified mask exists

FALSE, else

Extended: none

Service name: SelectDeltaStatus

Syntax: StatusType SelectDeltaStatus (NetIdType <NetId>,
StatusHandleType <StatusHandle>,
StatusHandleType <SMaskHandle>

Parameter(In):

NetId Addressed communication network

StatusHandle Referenced target network status

SMaskHandle Referenced network status mask

Parameter (Out): none

Description: A set of predefined parameter is selectable to drive the signaling of changed states.

Status: none

4.5 Services for direct NM

4.5.1 Standard functionality

4.5.1.1 System generation support

Name: InitDirectNMParams

Syntax: InitDirectNMParams (NetIdType <NetId>,
NodeIdType <NodeId>,
TickType <TimerTyp>,
TickType <TimerMax>,
TickType <TimerError>,
TickType <TimerWaitBusSleep>
TickType <TimerTx>

Parameter (In):

NetId: Addressed communication network

NodeId: Relative identification of the node-specific NM messages

TimerTyp: Typical time interval between two ring messages

TimerMax: Maximum time interval between two ring messages

TimerError: Time interval between two ring messages with NMLimpHome identification

TimerWaitBusSleep: Time the NM waits before transmission into the state NMBusSleep

TimerTx: Delay to repeat transmission requests

Description: *InitDirectNMParams* is a directive for initializing the parameters of the direct NM.

Particularities: none

4.5.2 Operating modes and operating mode management

4.5.2.1 Services

Service name: SilentNM

Syntax: StatusType SilentNM (NetIdType <NetId>)

Parameter (In):

NetId Addressed communication network

Parameter (Out): none

Description: *SilentNM* disables the communication of the NM. This causes the state transition from NMActive to NMPassive.

Particularities: none

Status:

Standard: E_OK, no error

Extended: none

Service name: TalkNM

Syntax: StatusType TalkNM (NetIdType <NetId>)

Parameter (In):

NetId: Addressed communication network

Parameter (Out): none

Description: *TalkNM* enables the communication of the NM again, after a previous call of *SilentNM*. This causes the state transition from NMPassive to NMActive.

Particularities: After a call of *StartNM* the NM is always in state NMActive.

Status:

Standard: E_OK, no error

Extended: none

4.5.3 Data Field Management

4.5.3.1 Data Types

Table 15 — Special data types of the data field management

NM data types	Remark
RingDataType	Type of data field in the NMPDU

4.5.3.2 System Generation Support

Service name: InitIndRingData

Syntax: InitIndRingData (NetIdType <NetId>,
SignallingMode <SMode>,
TaskRefType <TaskId>,
EventMaskType <EMask>)

Parameter (In):

NetId: Addressed communication network

SMode: Mode of signalling

TaskId: Reference to the task to be signalled

EMask: Mask of the events to be set

Description: *InitIndRingData* is a directive for specifying the indication of received data in the data field of a ring message, which is addressed to this node.

The parameter <SMode> specifies whether task activation (SMode = Activation) or event signaling (SMode = Event) is used for indication.
In case of task activation, <TaskId> contains a reference of the task to be activated if the NM received ring data.
In case of event signaling, <EMask> specified the event to be set for task <TaskId> if the NM received ring data.

Particularities: none

4.5.3.3 Services

Service name: ReadRingData

Syntax: StatusType ReadRingData (NetIdType <NetId>,
RingDataType <RingData>)

Parameter (In):

NetId: Addressed communication network

Parameter (Out):

RingData: Contents of the data field are within the network management that contains the data either received by the last NM message or written to by TransmitRingData.

Description: *ReadRingData* enables the application to read the data that has been received by a ring message.

Particularities: none

Status:

Standard: E_OK, no error

E_NotOK:
the NM does not pass a ring message currently;
the logical ring does not run in a stable state.

Service name: TransmitRingData

Syntax: StatusType TransmitRingData (NetIdType <NetId>,
RingDataType <RingData>)

Parameter (In):

RingData: Data which is written to the data field to be transmitted with the next ring message

NetId: Addressed communication network

Parameter (Out): none

Description: This service enables the application to transmit data via the ring message.

Particularities: none

Status:

Standard: E_OK, no error

E_NotOK:
the NM does not pass a ring message currently;
the logical ring does not run in a stable state.

4.6 Services for indirect NM

4.6.1 Standard functionality

4.6.1.1 System generation support

Name: InitIndirectNMParams

Syntax: InitIndirectNMParams (NetIdType <NetId>,
NodeIdType <NodeId>,
TickType <TOB>,
TickType <TimerError>,
TickType <TimerWaitBusSleep>

Parameter (In):

NetId: Addressed communication network

NodeId: Relative identification of the node-specific NM messages

TOB: Time to monitor a subset of nodes

TimerError: Time interval before reinitializing the bus hardware after an error which makes the NM shift to LimpHome

TimerWaitBusSleep: Time the NM waits before transmission in NMBusSleep

Description: *InitIndirectNMParams* is a directive for initializing the parameters of the indirect NM.

Particularities: none

4.6.2 Configuration management

4.6.2.1 System generation support

The determination of the monitored messages which are used by the indirect NM is located and described by the system generation of COM.

Name: InitExtNodeMonitoring

Syntax: InitExtNodeMonitiring (NetIdType <NetId>,
NodeIdType <NodeId>,
Int < DeltaInc>
Int < DeltaDec>

Parameter (In):

NetId: Addressed communication network

NodeId: Relative identification of the node-specific NM messages

DeltaInc: Value to increment the node status counter when a message is not received during a given time

DeltaDec: Value to decrement the node status counter when a message is received

Description: *InitExtNodeMonitoring* is a directive for initializing a set of parameters to monitor one node with an individual time-out. The (redundant) parameter “threshold” is hidden.

Particularities: none

5 Impacts upon OS, COM and the data link layer

5.1 Error codes

The NM supports several mechanisms to pass errors inside the NM to the application:

— return value (see “Error codes” in ISO 17356-2) of the API-services;

— indications which are activated by the NM if required configurations or network states are not recognized by the NM.

The NM does not support centralized error handling by the application:

— not any error-hook specific to NM;

— not a common error-hook used by ISO 17356, parts 3, 4 and 5;

— services to pass kernel errors to the application, which are not supported;

— the handling of NM kernel errors, which are up to the implementers.

5.2 Common impacts

5.2.1 Requirements of the data link layer

5.2.1.1 D_Init

From the NM point of view, five services to initialize the DLL are needed in general. Parameters are adjusted according to the following examples:

— baud rate;

— sample point;

— sample algorithm;

— synchronization mechanism;

— bit timing;

— Sleep Mode of the protocol circuit;

— Sleep Mode of the physical layer;

— Standby Mode of the physical layer;

— operation modes of the protocol circuit.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)
InitRoutine	BusInit	initialize the bus hardware once at the start of the network
	BusShutDown	shut down the bus hardware
	BusRestart	restart the bus hardware in the case of a fatal bus error
	BusSleep	initialize the power down mode of the bus hardware
	BusAwake	reinitialize the bus hardware to leave the power down mode

5.2.1.2 D_Status.ind

Indication of states of the data link layer (software and hardware) are according to the following examples:

— errors from the physical layer;
— errors from bus monitoring circuits;
— errors from the protocol circuit (CAN, e.g. bus off or error active/passive);
— errors from the DLL;
— wake-up signal.

EXAMPLE

parameter (out)	NetId	connected bus (not necessary when just one bus is connected)
	status	hardware specific status data

5.2.1.3 D_GetLayerStatus

Reading the status information of the data link layer according to the following examples:

— interrupt acknowledge to the protocol circuit;
— get the status of the protocol circuit, e.g. transmit, receive, overrun, bus off;
— get the status of the physical layer, e.g. transmission line error.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)
parameter (out)	status	hardware specific status data

5.2.1.4 D_Offline

This service allows the user transmission via the data link layer at least to be blocked.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)

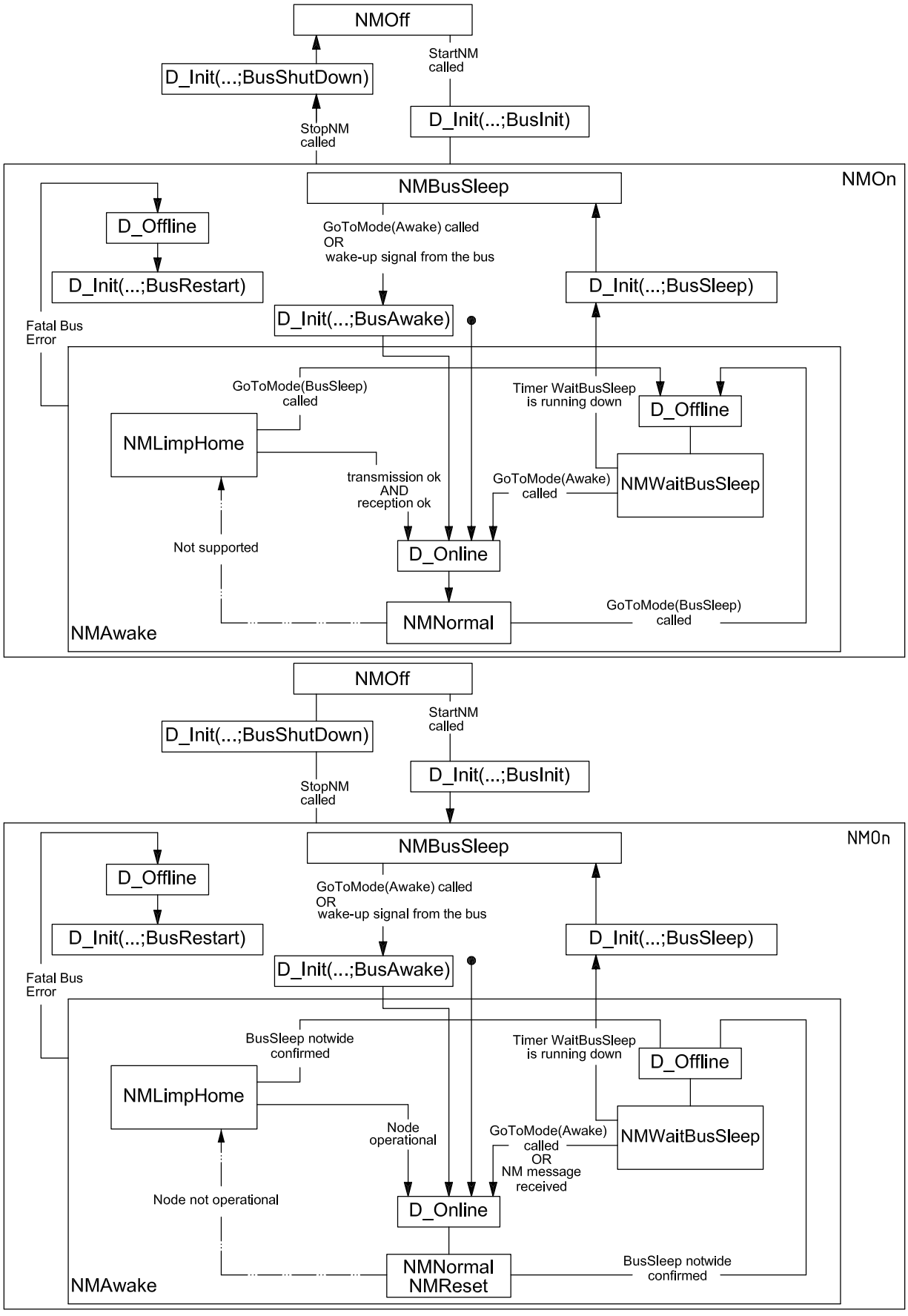
5.2.1.5 D_Online

This service enables the user communication on the data link layer, e.g. after a call of D_Offline.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)

The NM calls DLL services at the transition from one state to another state.



On the left is an indirect NM, on the right a direct NM.

Figure 70 — Use of DLL services by the NM

5.2.2 Requirements of OS

The operating system requirements for implementation of NM are listed below. The standard services for configuration management, management of operating modes and data field management are available at the lowest conformance class BCC1 of OS. This allows the implementation of NM on the basis of OS class BCC1. Additional features partly require higher conformance classes.

If NM uses the event triggering mechanism, then this feature is required from the operating system.

The implementation can also be based on a non-OS, which at least provides the functionality of OS services listed below:

— Alarm services: SetRelAlarm and CancelAlarm; and

— Task management: GetTaskState, DeclareTask, ActivateTask, TerminateTask and ChainTask.

5.3 Impacts from direct NM

5.3.1 Interface to the data link layer

5.3.1.1 General

From the NM point of view, the NM in a node shall transmit a NMPDU to the bus and shall receive every NMPDU from the NMs in all networked nodes. The structure of the NMPDU is fixed by the NM. However, the data representation inside a NMPDU and how to code/decode a NMPDU to a message is out of the scope of the NM. Annex A contains proposals to handle these tasks.

Table 16 — NMPDU – responsible

Topic	Responsible
Structure of the NMPDU EXAMPLE Address-Field (source and destination) control-Field (12 message types) optional Data-Field	NM
Data representation inside the NMPDU	out of the scope of NM
Coding and decoding of a NMPDU to a message	out of the scope of NM

In general, the interface between NM and the DLL to transmit and receive NMPDUs will be directly influenced by the agreement to fix the data representation inside an NMPDU and the coding/decoding to a message.

Based on the experiences according to the state of the art and the proposals given in Annex A, an interface between NM and the DLL can be suggested.

5.3.1.2 D_DefineWindow

Definition of the encoding/decoding algorithm to broadcast/receive the NMPDU via the bus. This action will be handled by a system generation tool. The system generator is responsible for the selected algorithm.

EXAMPLE

static parameter (in)	NetId	connected bus (not necessary when just one bus is connected)
	WindowMask	mask for filtering NM messages
	IdBase	base identification of an NM message
	SourceId	identification of the source of the NMPDU
	DataLengthTx	number of bytes of NMPDU to transmit (if data length is static)
	DataLengthRx	number of bytes of NMPDUs to receive (if data length is static)

5.3.1.3 D_Window_Data_req

Service to transmit a NMPDU to the network.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)
	NMPDU	except the source (static see the examples in D_Define_Window and DataLengthTx)
	DataLengthTx	number of bytes of the NMPDU to transmit (if data length is dynamic)

5.3.1.4 D_Window_Data_ind

Service to receive a NMPDU to the network.

EXAMPLE

parameter (in)	NetId	connected bus (not necessary when just one bus is connected)
parameter (out)	NMPDU	number of bytes referenced by the value DataLengthRx (static, see the example D_Define_Window)
	DataLengthRx	number of bytes NMPDUs to receive (if data length is dynamic)

5.4 Impacts from indirect NM

5.4.1 Interface to communication (COM)

5.4.1.1 General

When a monitored application message is received/transmitted by COM, indirect NM shall be informed. In case of using one dedicated time-out per message monitored, indirect NM shall be informed when a monitoring time-out expires.

For each of these situations, the indirect NM needs to know to which NetId and NodeId the monitored message refers. COM provides this information to NM via a parameter called “Sender”, corresponding to a combination of both NetId and NodeId.

Services needed between indirect NM and COM depend on the selected monitoring scheme (one global time-out/one dedicated time-out per monitored message).

Table 17 — Interface of indirect NM with COM

Interface to COM	Options	
	One global time-out	One dedicated time-out per monitored message
I_MessageTransfer.ind	core	core
I_MessageTimeOut.ind		core

5.4.1.2 I_MessageTransfer.ind

Indication from COM that a monitored message has been received from a remote node or that the local monitored message has been transmitted:

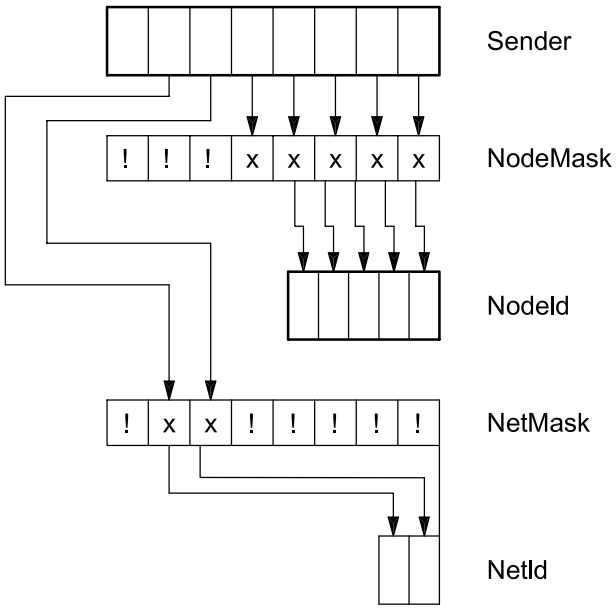
parameter (out) Sender combination of NodeId and NetId

I_MessageTimeOut.ind

Indication from COM that a time-out at monitoring a message from a remote node has expired or that the time-out at monitoring the local message transmission has expired:

parameter (out) Sender combination of NodeId and NetId

5.4.1.3 Mapping NodeId, NetId ⇔ Sender



Components
x don't care, take Message bit
! do not take this bit

Figure 71 — Encoding and decoding of sender to a NodeId and a NetId by using a mechanism with a mask

5.4.1.4 NMDefineNetNodeMapping

This is the definition of the algorithm to map a sender to a node and to a net. This action will be handled by a system generation tool. The system generator is responsible for the selected algorithm.

EXAMPLE

static parameter (in)	NetMask	mask for filtering NM messages
	NodeMask	mask for filtering NM messages

5.4.1.5 NMNetNodeMapping

This is the mapping of a given sender to the corresponding node and the corresponding net.

EXAMPLE

parameter (in)	sender		
parameter (out)	NodeId	node which corresponds to the referenced	identification
	NetId	connected bus (not necessary when just one bus is connected)	

Annex A
(informative)

Implementation proposal (direct NM)

A.1 Implementation proposal (direct NM)

A.1.1 Overview of internal activities

All the internal services of the NM begin with NM. All words of the service name begin with a capital letter.

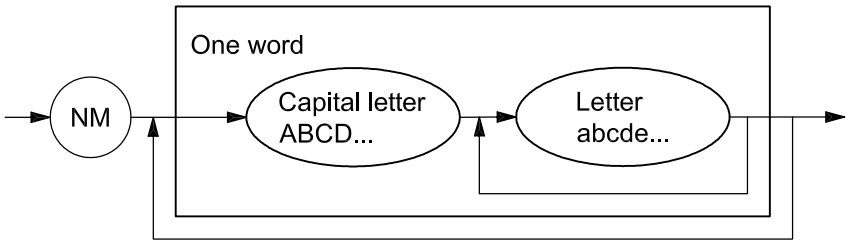


Figure A.1 — Syntax of the names of internal NM services (Example: NMShutDown)

Table A.1 — Breakdown of internal NM activities into core services and optional services: Reset, Normal or LimpHome

	Activity	Core or Optional
Shut-off of NM	**NMOff**	CORE
Save parameters, store history and shutdown NM	**NMShutDown**	CORE
Initialize NM and the resources pertaining to it	**NMInit**	CORE
NM in the state NMBusSleep	**NMBusSleep**	CORE
NM in the state NMActive	**NMActive**	CORE
NM in the state NMPassive	**NMPassive**	CORE
NM in the state NM~Standard	**NM~Standard**	CORE
NM in the state NM~Active	**NM~Active**	CORE
NM in the state NM~Passive	**NM~Passive**	CORE
NM in the state NM~ActivePrepBusSleep	**NM~ActivePrepBusSleep**	CORE
NM in the state NM~PassivePrepBusSleep	**NM~PassivePrepBusSleep**	CORE

The state transition diagrams (STD) listed hereafter define system hierarchy and general transition rules for the NM behaviour.

NM activities are performed by calls of the internal activities in the respective states of the STD and identified by the names of these dedicated internal activity. Internal activities are defined verbally in the referenced chapters according to the description of their characteristics. Consequently, they can be considered as macros which are generated at compile time, using (elementary) services which are defined otherwise.

Thus, there is neither an appropriate C syntax, nor specifications about input / output parameters or status of the internal activity.

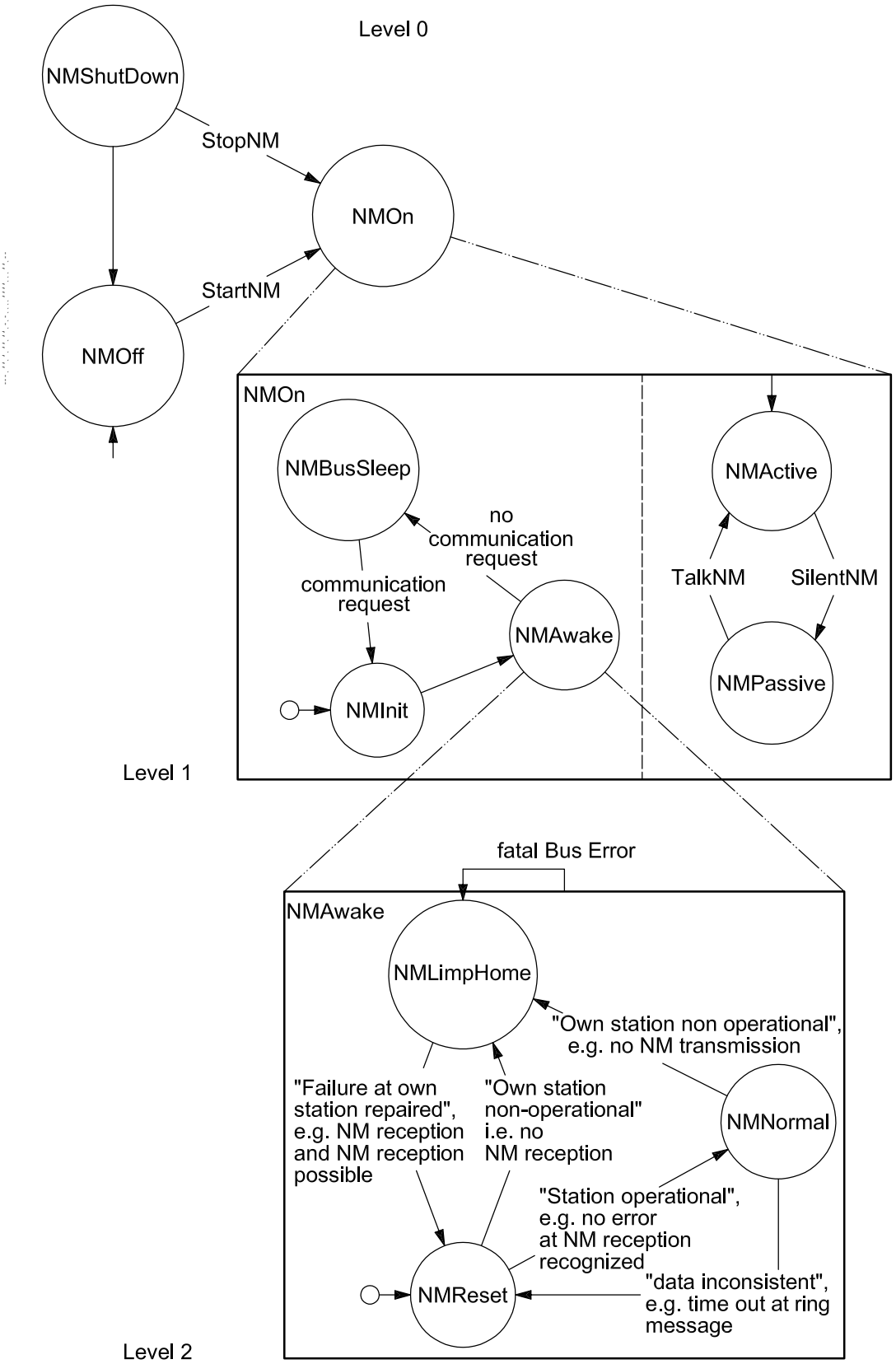


Figure A.2 — Simplified state transition diagram (STD) of the direct NM

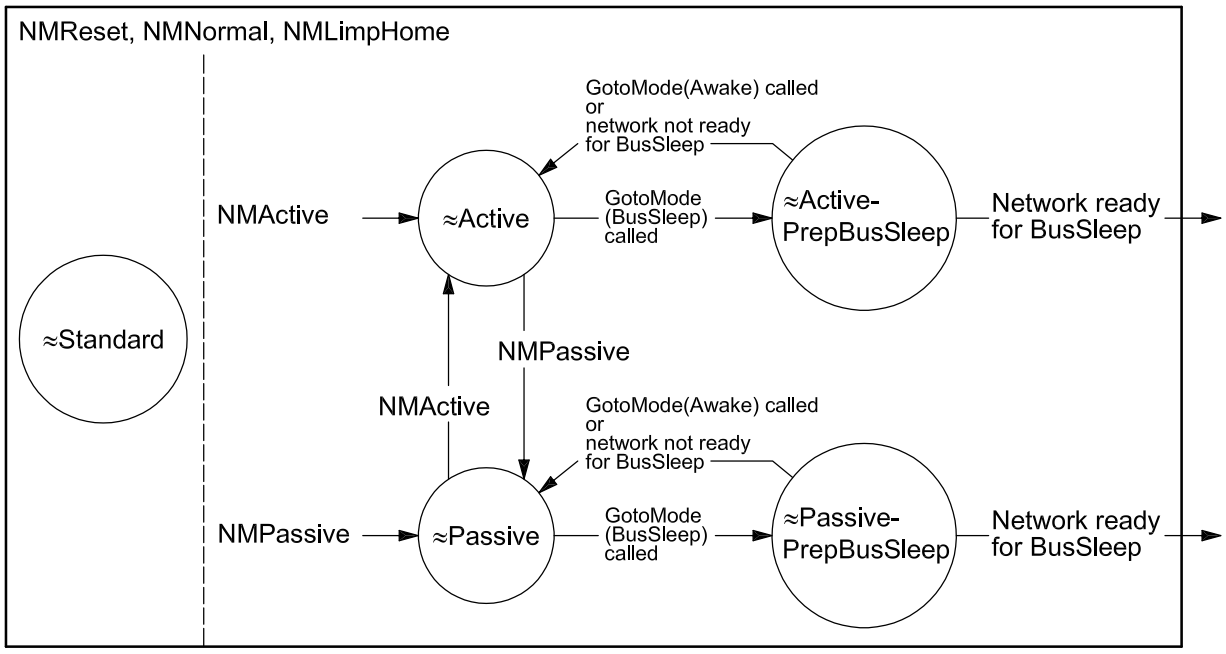


Figure A.3 — NM internal states "NMNormal" or "NMReset" or "NMLimpHome"

A.1.2 Specification of internal activities

Service name: NMOff

Description: NM of the node is shut-off.

Particularities: none

Service name: NMShutDown

Description: Consists of service for selective shut-off of NM entity. This includes all "clearing-up work" (see below) to be effected by NM.

This service is effected without confirmation throughout the whole network (see Figure A.2).

The tasks of this service comprise:

- Saving NM state, including the last valid network configuration, operating state, version number (optional, depending on system design).
- Releasing all resources assigned for NM.
- Resetting interface module.

Particularities: none

Service name: NMInit

Description: Service for initializing NM according to NM STD:

- Initialization of network interface.
- Assignment and initialization of NM resources.

Particularities: none

Service name: NMBusSleep

Description: The NM module of the node is mode NMBusSleep according to the NM STD (level 1).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMActive

Description: The NM module of the node is mode NMActive according to the NM STD (level 1).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMPassive

Description: The NM module of the node is mode NMPassive according to the NM STD (level 1).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMNormalActivePrepBusSleep

Description: The NM module of the node is mode NMNormalActivePrepBusSleep according to the NM STD (level 3).

The activities performed are according to the concept of NM the notification of a sleep request for the whole network to all nodes in the network and pending for confirmation.

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMLimpHomeActivePrepBusSleep

Description: The NM module of the node is mode NMLimpHomeActivePrepBusSleep according to the NM STD (level 3).

The activities performed are according to the concept of NM the notification of a sleep request for the whole network to all nodes in the network and pending for confirmation.

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMResetActivePrepBusSleep

Description: The NM module of the node is mode NMResetActivePrepBusSleep according to the NM STD (level 3).

The activities performed are according to the concept of NM the notification of a sleep request for the whole network to all nodes in the network and pending for confirmation.

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMNormalPassivePrepBusSleep

Description: The NM module of the node is mode NMNormalPassivePrepBusSleep according to the NM STD (level 3).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMLimpHomePassivePrepBusSleep

Description: The NM module of the node is mode NMLimpHomePassivePrepBusSleep according to the NM STD (level 3).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMResetPassivePrepBusSleep

Description: The NM module of the node is mode NMResetPassivePrepBusSleep according to the NM STD (level 3).

Particularities: Concrete procedures are specified by the respective system responsible.

Service name: NMNormalActive

Description: The NM module of the node is mode NMNormalActive according to the NM STD (level 3).

The procedure performed is to participate in the NM communication according to the logical ring concept and to assess the NMPDU.

Particularities: none

Service name: NMLimpHomeActive

Description: The NM module of the node is mode NMLimpHomeActive according to the NM STD (level 3).

The procedure performed is to participate in the NM communication according to the logical ring concept and to assess the NMPDU.

Particularities: none

Service name: NMResetActive

Description: The NM module of the node is mode NMResetActive according to the NM STD (level 3).

The procedure performed is to participate in the NM communication according to the logical ring concept and to assess the NMPDU.

Particularities: none

Service name: NMNormalPassive

Description: The NM module of the node is mode NMNormalPassive according to the NM STD (level 3).

Particularities: none

Service name: NMLimpHomePassive

Description: The NM module of the node is mode NMLimpHomePassive according to the NM STD (level 3).

Particularities: none

Service name: NMResetPassive

Description: The NM module of the node is mode NMResetPassive according to the NM STD (level 3).

Particularities: none

Service name: NMNormalStandard

Description: The NM module of the node is mode NMNormalStandard according to the NM STD (level 3).

Particularities: none

Service name: NMLimpHomeStandard

Description: The NM module of the node is mode NMLimpHomeStandard according to the NM STD (level 3).

Particularities: none

Service name: NMResetStandard

Description: The NM module of the node is mode NMResetStandard according to the NM STD (level 3).

Particularities: none

A.1.3 NMPDU

A.1.3.1 General

The NM implementation of direct node monitoring supports the implementation of NMPDU as listed hereafter.

Additional information for extended NM features, e.g. dedicated enhanced diagnosis support, could be mapped into the data field of the NM message. This is an optional feature in the responsibility of the respective system developer and it depends on the used bus protocol.

A.1.3.2 Implementation

The implementation features:

— support of a maximum number of 256 nodes;

— demand of 3 bytes.

Table A.2 — Implementation of NMPDU

Addressing field		Control field	Data field (optional)
8 bit	8 bit	8 bit	specific to the protocol (e.g. CAN)
Source Id	Destination Id	OpCode	Data

A.1.3.3 OpCode

Table A.3 — Table NMPDU

Address field		Control field									Data field
Source Id	Dest. Id	OpCode									Data
mandatory											optional
		Coding Example								Interpretation	
		0	0	0	0	0	**0**	**0**	**1**	Ring Message, cleared Bussleep.ack, cleared Bussleep.ind	0
		0	0	0	0	0	**1**	**0**	**1**	Ring Message, cleared Bussleep.ack, set Bussleep.ind	
		0	0	0	0	0	**x**	**1**	**1**	Ring Message, set Bussleep.ack	
		0	0	0	0	0	**0**	**1**	**0**	Alive Message, cleared Bussleep.ind	
		0	0	0	0	0	**1**	**1**	**0**	Alive Message, set Bussleep.ind	
		0	0	0	0	0	**0**	**0**	**0**	Limp Home Message, cleared Bussleep.ind	
		0	0	0	0	0	**1**	**0**	**0**	Limp Home Message, set Bussleep.ind	

The first five bits of the OpCode are reserved for future extensions. They should be initialized to logical zero. The data field should be initialized to logical zero.

A.1.3.4 Encoding and decoding

A.1.3.4.1 Addressing mechanisms

The following set-up is required for each node to implement the window mechanism with a broadcast behaviour:

— one node-specific transmit object;

— one or more global receive objects (windows) for all node-specific NM messages.

Under worst-case conditions, NM uses a range of message headers for network-wide communication. Such a range of messages can be mapped to one or more window objects. Each window object is identified by the values:

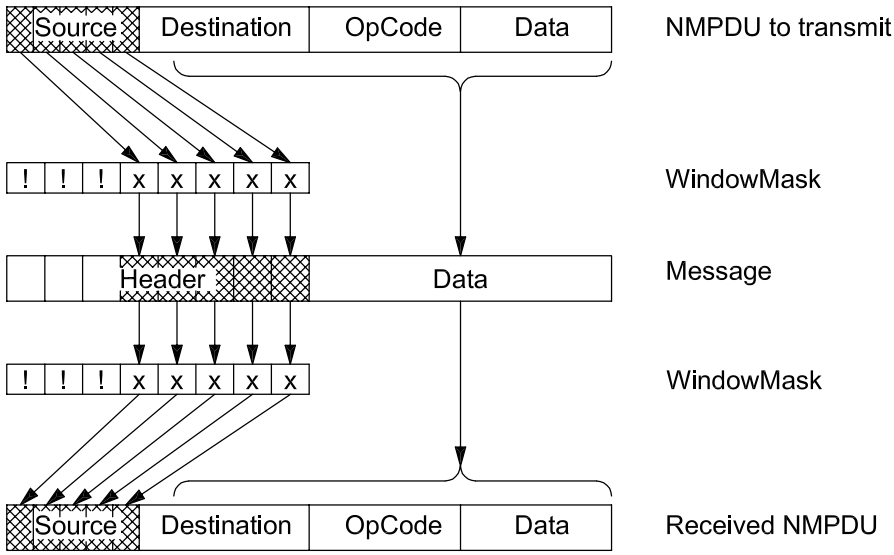
— IdBase: Base identification of any NM message header;

— WindowMask: Mask for filtering NM messages (acceptance).

EXAMPLE for acceptance filtering:

Reception is OK: IF(Id_of_Frame & WindowMask = = IdBase)

EXAMPLE for encoding and decoding of a NMPDU



Components
x don't care, take NMPDU bit
! take original bit of IdBase

Figure A.4 — Encoding and decoding of the NMPDU to a message by using the window mechanism with IdBase and WindowMask

The example shows that the receiving node can determine parts of the NMPDU, e.g. the identification of the transmitting node, from the transmitted frame.

A.1.3.4.2 Coherent allocation of NM message headers

A simple implementation results if the message headers for NM are selected in a coherent numeric range.

Two integers n and k shall be selected in order to enable straightforward acceptance filtering of NM messages.

Using the constant n, 2^n (WindowSize) directly addressable nodes are made available. The constant k defines the Base of the message header as an integer multiple of the maximum number of directly addressable nodes.

Table A.4 — Selection of message headers and NodeNumbers

Node identification	0 ... $2^n - 1$
IdBase	$k\ 2^n$
least message header	$k\ 2^n + 0$
greatest message header	$k\ 2^n + 2^n - 1$

EXAMPLE Addressing of 32 separate nodes is enabled. The NM message headers start with message identifier 600hex. This implies:

Selected parameters: 32 = (2^5) → n = 5
600hex = (48*32) → k = 48.

Node identification 0 ... 31 dec

Least header `110 000 0000` bin
`110 000` bin, corresponds to k

Greatest header (61Fhex) `110 0001 1111` bin

IdBase `110 0000 0000` bin

WindowMask `111 1110 0000` bin "`1`" : target
"`0`" :don't care

EXAMPLE For CAN, an NM message containing the NMPDU shall be mapped into diverse bus protocols. The figures below show a CAN realization example (i.e. max. 256 nodes can be addressed). Because CAN implementations do not allow unique message identifiers used by more than one transmitter, it is essential that all NM messages differ from each other. This can be achieved by e.g. encoding the NM Source Id into the CAN message Id.

CAN Identifier	DCL	CAN Data field			
11 (29) bit	4 bit	⩽ 64 bit			
	Addressing field			Control field	Data field
3 (21) bit	8 bit		8 bit	8 bit	48 bit
IdBase	Source Id		Dest. Id	OpCode	Data
x	S	8	D	x	x

Figure A.5 — Structure of NM message in case of CAN (6-byte data field)

CAN Identifier		DCL	CAN Data field
11 (29) bit		4 bit	16 bit
	Addressing field		Control field
3 (21) bit	8 bit		8 bit	8 bit
IdBase	Source Id		Dest. Id	OpCode
x	S	2	D	x

Figure A.6 — Structure of NM message in case of CAN (without data field)

Can Identifier	CAN Data field
11 (29) bit	3 - 8 byte

	Addressing field		Control field		Data field
3 (21) bit	8 bit	1 byte	1 byte	1 byte	⩽ 5 byte
	Source Id	Dest. Id	OpCode		Data

x	D hex	0	0	x	x	x	x	x	x	Ring messages
x	D hex	1	0	x	x	x	x	x	x	
x	D hex	0	1	x	x	x	x	x	x	
x	D hex	1	1	x	x	x	x	x	x	
x	E hex	0	0	x	x	x	x	x	x	Alive messages
x	E hex	1	0	x	x	x	x	x	x	
x	C hex	0	0	x	x	x	x	x	x	LimpHome messages
x	C hex	1	0	x	x	x	x	x	x	

Components

x reserved

Figure A.7 — Example of the mapping of the NMPDU to an NM message based on CAN, comparable to the DaimlerChrysler encoding

NOTE In principle, message headers required to implement the window can obviously be assigned in any order. Selecting the digits n and k according to the principle introduced above, the choice is automatically limited to powers of two and enables straightforward filtering for acceptance in the destination system. In the case of possible dynamic allocations, the window parameters can be coded using two bytes and can be transmitted with a message.

A.1.3.4.3 Non-coherent allocation of NM message headers

If the system design requires distribution, i.e. numerically separate arrangement, of the message headers, they can remain coherent within the software if an appropriate mask is used.

EXAMPLE Addressing of 32 separate nodes is enabled. The NM message headers 400hex to 40Fhex and 600hex to 60Fhex a used This implies:

—	Node identification	0 ... 31 dec
—	Least header (400hex)	`100 0000 0000` bin
—	Header 40Fhex	`100 0000 1111` bin
—	Header 600hex	`110 0000 0000` bin
—	Header 60Fhex	`110 0000 1111` bin
—	IdBase	`100 0000 0000` bin
—	WindowMask	`101 1111 0000` bin "`1`" : target "`0`" : don't care

A.1.3.4.4 Node identifications

The local node identifications of NM, and consequently the global node identifications, shall be allocated uniquely within the entire network.

In accordance with the determinations, numeric values in the range from 0 to (2^n – 1) are used for this purpose. Group addresses are provided for special applications by the system responsible. It depends on the selected transformation for node identification into message header, whether the local and global node identifications are equal.

Table A.5 — Determination of node identifications using the example n=8

Node identification	Interpretation
0	reserved
1 ... 254	node no. 1 up to node no. 254
255	Group “all nodes”

A.1.4 Scalability

In most control unit networks with a centralized structure, three node types are distinguished:

— Function master: Clearly defined node which performs all centralized and coordination functions.

— Potential function master: In case of failure of the function master, e.g. node breaks down, each of these back-up masters is capable of performing at least some of the master's functions.

— Function slaves

The individual nodes may feature broadly varying available computing power for implementation of NM. The decentralized NM can be scaled to save resources (requirements of RAM/ROM and computer time), resulting in two extreme NM types:

— Max_NM: Set of all NM functions according to direct node monitoring.

— Min_NM: Minimum set of required functionality enabling participation in direct node monitoring.

The choice of functions can be adapted to the nodes' performance.

Table A.6 — Functionality of the configuration algorithms of Max NM and Min NM

Task	Max NM	*scalable* →	Min NM
Store the present configuration	✓		-
Time-out monitoring to detect faulty node	✓		-
"Re-login" if skipped	✓		✓
Login	✓		-
Determine logical successor	✓		✓
Delayed transmission of NM message according to sequencing rule of the logical ring	✓		✓
Startup of the logical ring	✓		-

If necessary, the individual node types (Function master ... Function slave) can be supplied with subsets of the decentralized NM.

In a centrally structured network, the group of nodes consisting of function master and potential function masters can be considered as decently structured with regard to the configuration adjustment within the NM.

The dynamic concept of configuration determination enables integration of any function slaves performing Min NM and of any potential function master into the network.

NOTE For the sake of clarity, the implementation of identical NM modules is required in each node. In other words, the basic scalability of the decentralized NM should only be used in vital, exceptional cases.

A.2 Implementation proposal (indirect NM)

A.2.1 Scalability

According to system designer needs and to computing power performance of nodes (RAM/ROM and computer time), indirect NM can be scaled in NM types ranging from:

— Max_NM: Set of all NM functions including all extended features;

down to:

— Min_NM: Minimum set of required functionality enabling network communication.

Table A.7 — Example of functionality for different NM types

Function	Max NM	scalable →			Min NM
Hardware initialization, restart of hardware after a failure, bus shutdown	✓	✓	✓	✓	✓
Dynamic states monitoring	✓	✓	✓	✓	-
Static states monitoring	✓	-	✓	-	-
BusSleep	✓	✓	-	-	-

NOTE The implementation of identical indirect NM type is not required in each node. Choice of functions to be implemented is let to system designer.

A.2.2 Implementation hints

A.2.2.1 Choice one global time-out/one monitoring time-out per message

A.2.2.1.1 General

Implementing node monitoring functionality, the system designer can choose two monitoring schemes:

— all messages are monitored by one global time-out TOB (time-out for observation);

— each message is monitored by its own dedicated time-out.

A.2.2.1.2 One global time-out

— Advantage: This solution does not require much micro-controller CPU time resource.

— Drawback: If monitored messages have very different transmission period (for example, one 10ms message from a node and one 500ms message from another), the user shall choose the biggest value for timer TOB to be sure than each message has arrived before time-out expires. The resulting delay on the 10ms message monitoring may be unacceptable if this message is time-critical for the application.

A.2.2.1.3 One time-out per monitored message

— Advantage: Each message can be monitored regarding its time-criticality.

— Drawback: This solution requires more micro-controller CPU time resource.

A.2.2.2 Configuration of extended states detection algorithm

A.2.2.2.1 General

Extended states detection algorithm shall be configured at system generation time. Parameters to be set are:

— the Threshold value, which is the same for all counters; and

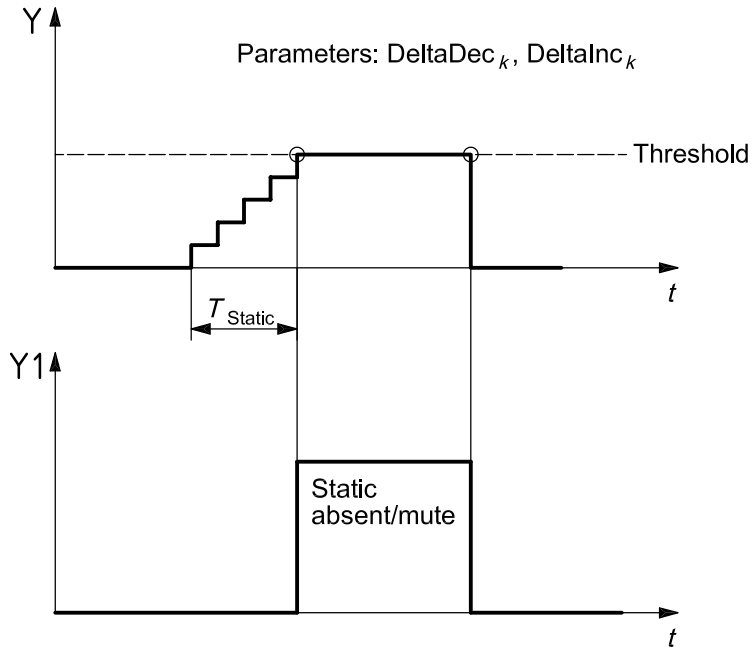
— a DeltaInc (increment of counter) and a DeltaDec (decrement of counter) values per monitored node.

Threshold value is usually set to 255; its value has no impact on the algorithm behaviour. DeltaInc and DeltaDec modify algorithm behaviour.

A.2.2.2.2 Examples

A.2.2.2.2.1 Example 1

If the system designer needs “static states” corresponding to states during a unique T_{Static} time value for every monitored node, although these nodes have different transmission periods and are monitored by different time, counters return directly to 0 when static states are left.



Y counter $_k$

Y1 extended state of the node k

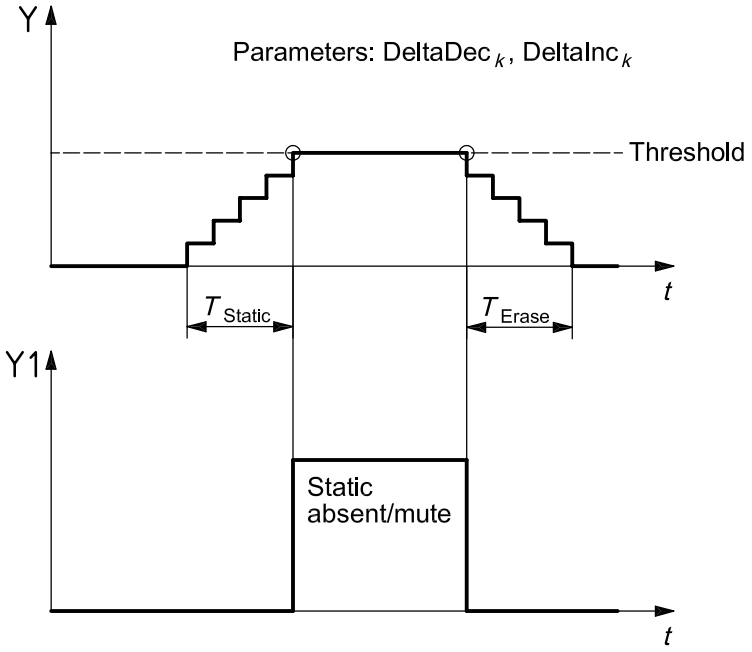
Figure A.8 — Extended state example one

Table A.8 — Calculation of DeltaInc and DeltaDec according to example one
TimeOut$_k$: Monitoring time-out for node k

Parameter of node k	Value
DeltaInc	$\frac{Threshold \times TimeOut_k}{T_{Static}}$ [a]
DeltaDec	Threshold
[a] TimeOut$_k$: Monitoring time-out for node k.	

A.2.2.2.2.2 Example 2

If the system designer needs “static states” corresponding to states during a unique T_{Static} time value for every monitored node, although these nodes have different transmission periods and are monitored by different time-outs, counters keep track of node states during a T_{Erase} time value after static states are left.



Y counter $_k$

Y1 extended state of the node k

Figure A.9 — Extended state example two

Table A.9 — Calculation of DeltaInc and DeltaDec according to example one

Parameter for node k	Value
DeltaInc	$\frac{Threshold \times TimeOut_k}{T_{Static}}$ [a]
DeltaDec	$\frac{Threshold - T_k}{T_{Erase}}$ [b]

[a] $TimeOut_k$: Monitoring time-out for node k.

[b] T_k: Period of the supervised message received from node k.

A.2.3 Summary of SDL state diagram graphical notation

The SDL graphical symbols used in the specification of the Indirect Network Management state machine are described in Figure A.10.

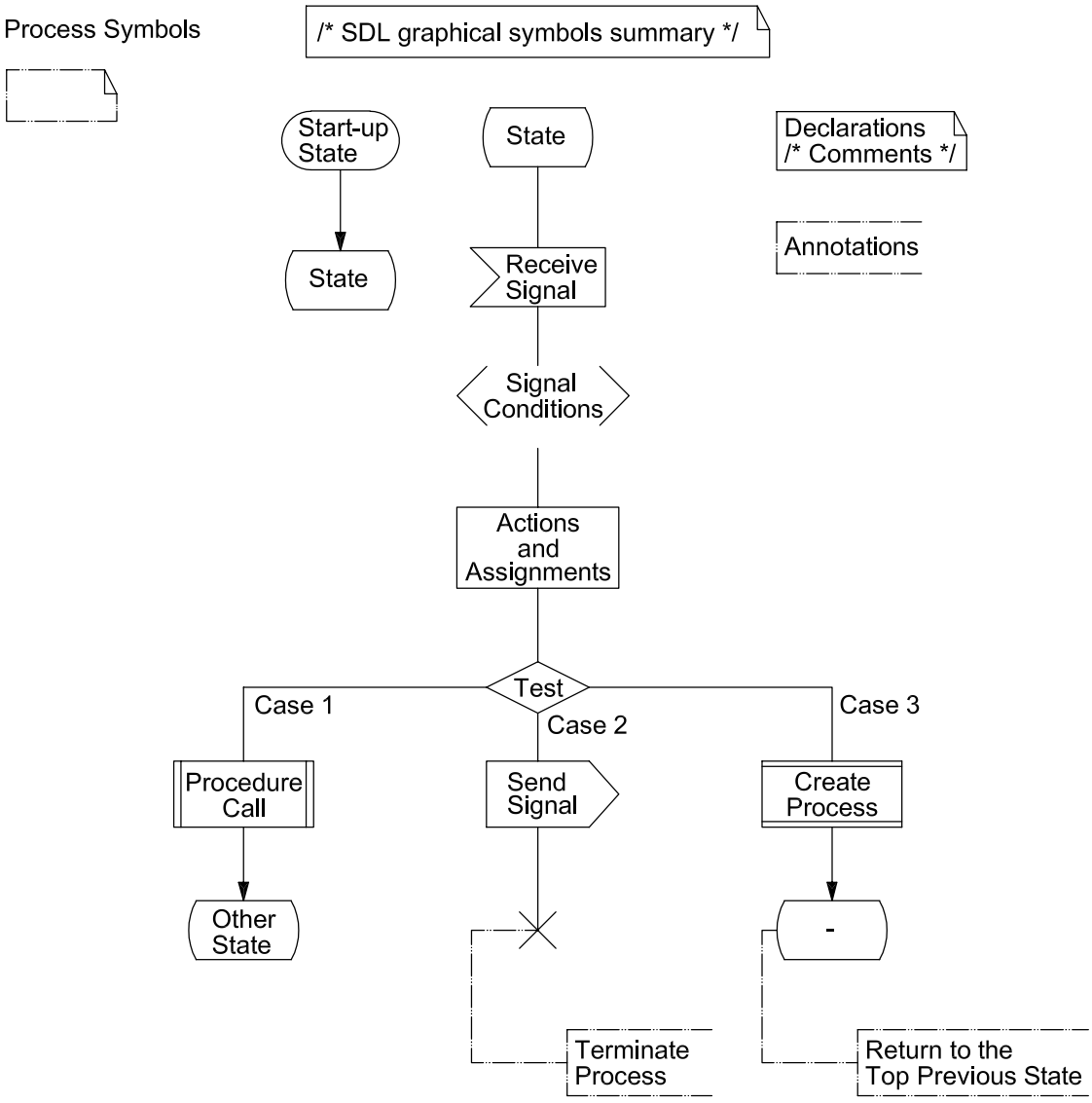


Figure A.10 — Summary of SDL state diagram graphical notation

Index

List of all Network Management services, data types and internal activities.

ICS 43.040.15

Price based on 117 pages