Draft Specification Proposal

CiA 302-2

Version 4.1.0

CANopen additional application layer functions

Part 2: Network management

February 02, 2009

HISTORY

Date	Changes
2006-08-08	*Publication of version 4.0* as draft standard proposal
	Based on CiA 302 version 3.3
	Split into parts, re-worded and re-chaptered
2009-02-02	*Publication of version 4.1* as draft standard proposal
	NOTE: This document has been converted into “docx format”. The conversion caused minor layout differences to the predecessor document in “doc format”. The technical content word-by-word is the very same.

General information on licensing and patents

CAN in AUTOMATION (CiA) calls attention to the possibility that some of the elements of this CiA specification may be subject of patent rights. CiA shall not be responsible for identifying any or all such patent rights.

Because this specification is licensed free of charge, there is no warranty for this specification, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holder and/or other parties provide this specification “as is” without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the correctness and completeness of the specification is with you. Should this specification prove failures, you assume the cost of all necessary servicing, repair or correction.

CiA is committed to use inclusive language in its specifications and technical reports. CiA documents are going to be updated with the inclusive terms. A listing of inclusive terms and terms they substitute is provided in the CiA house style document downloadable from the CiA website (https://can-cia.org/s/housestyle).

Trademarks

CANopen and CiA are registered community trademarks of CAN in Automation. The use is restricted for CiA members or owners of CANopen® vendor ID. More detailed terms for the use are available from CiA.

CAN in Automation e. V.
Kontumazgarten 3
DE - 90429 Nuremberg, Germany
Tel.: +49-911-928819-0
Fax: +49-911-928819-79
Url: www.can-cia.org
Email: headquarters@can-cia.org

CONTENTS

1 Scope

The definition of the network management includes the definition of the network startup behavior as well as definitions that are related to networks that operate without NMT master, networks with one CANopen device capable of the NMT master mode, and networks with more than one CANopen device capable of the NMT master mode (NMT flying master for higher availability).

These definitions are intended to be an add-on to the CANopen application layer and communication profile (see */CiA301/*).

2 References

/CiA301/ CIA 301, CANopen – Application layer and communication profile

/CiA302-1/ CiA 302, Additional application layer functions — Part 1: General definitions

/CiA305/ CiA 305, CANopen – Layer setting services and protocol (LSS)

3 Startup

3.1 NMT startup

CANopen managers shall behave according to the NMT slave state machine as defined in */CiA301/*. Before transition from NMT state *Pre-operational* to NMT state *Operational* of the CANopen manager all assigned NMT slaves shall be booted. The main flow chart for the procedure is defined in Figure 1 and Figure 2. The simplest flow chart for the startup is defined in Figure 3.

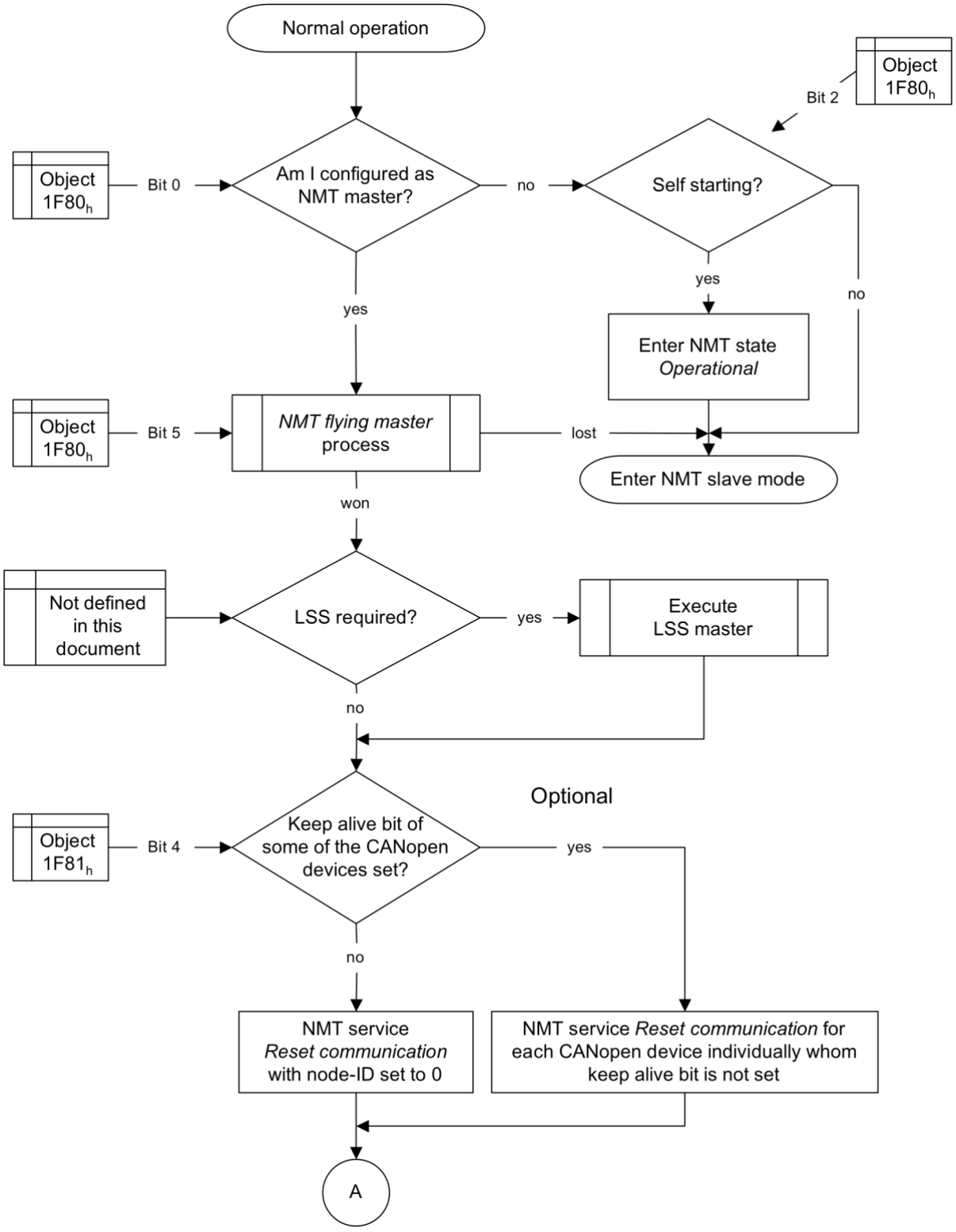

Figure 1 — NMT startup, part 1

The process NMT startup as shown in Figure 1 consists of the following basic steps[1]:

a) Bit 0 of object 1F80h (see 6.1) is used to decide upon whether this CANopen device shall be the NMT master. If the CANopen device is NMT master the process shall continue. If the CANopen device is configured as self-starting device, it shall enter the NMT state

[1] The process execute LSS master is not given in this specification

Operational automatically.
If the CANopen device is not the NMT master the process shall end.

b) Bit 5 of object $1F80_h$ (see 6.1) is used to decide upon whether this CANopen device shall participate in the service *NMT flying master negotiation* (see 5.3). If the CANopen device shall participate in the service *NMT flying master negotiation* and the CANopen device lost the service *NMT flying master negotiation* the CANopen device shall not become NMT master.

c) If LSS (see */CiA305/*) is required to set the node-ID and bit rate of other CANopen devices within the network the NMT master shall execute the LSS master services. The LSS master services may be executed at any time. The precise definition of the LSS master services falls not within the scope of this specification.

d) Bit 4 of all entries object $1F81_h$ (see 6.3) is used to decide upon whether the NMT master shall perform the NMT service *Reset communication* with node-ID set to 0 or if the NMT service *Reset communication* shall be performed for each CANopen device in the network individually.
If at least for one entry of object $1F81_h$ bit 4 is set to 1_b and the corresponding CANopen device is in NMT state *Operational* the NMT master shall not issue the NMT service *Reset communication* with node-ID set to 0. In this case each CANopen device shall be reset individually.
This shall also include all node-IDs that are not part of the slave list $1F81_h$.
NOTE: This will force potentially existing CANopen devices that are not configured in the slave list to transmit a boot-up message. By this the CANopen Manager will recognize un-configured CANopen devices via the boot-up handler (Figure 13).
The NMT service *Reset communication* shall not apply to the NMT master itself.

If after step d) the NMT service *Reset communication* is necessary, for any reason, then the NMT service *Reset communication* shall be executed with the node-ID of that NMT slave. This shall apply independently from the configuration for that NMT slave in object $1F81_h$.

The process NMT startup resumes in Figure 2.

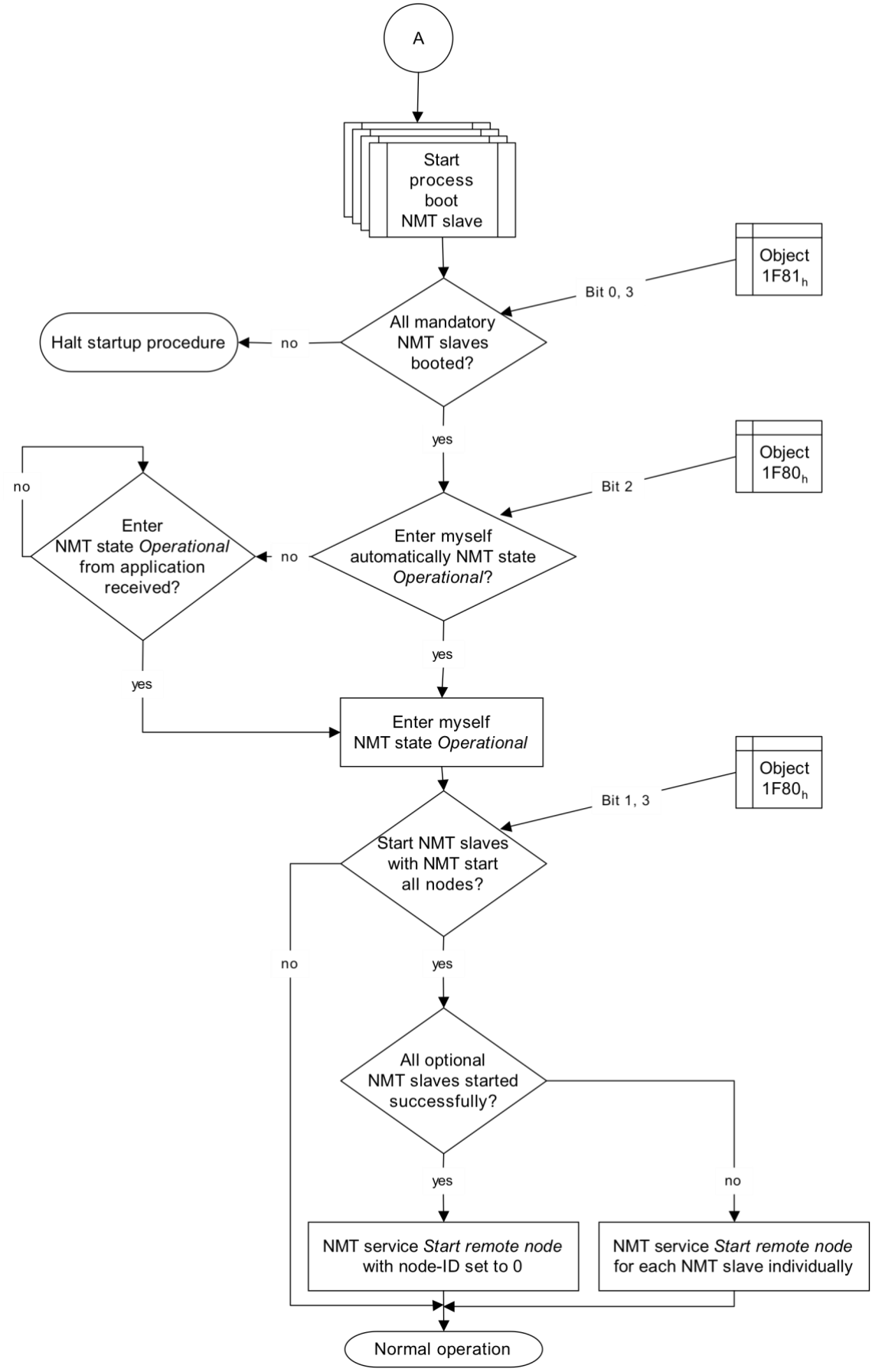

Figure 2 — NMT startup, part 2

e) The NMT master shall start the process *start process boot NMT slave* for all NMT slaves as shown in Figure 4. For all NMT slaves that are marked as mandatory (bit 0 and bit 3 of object $1F81_h$; see 6.3) the process *start process boot NMT slave* shall terminate successfully.

f) If an error is detected during process *boot slave* for the NMT slaves that are marked as mandatory the process *NMT startup* shall be stopped.

g) Bit 2 of object $1F80_h$ (see 6.1) is used to decide upon whether the NMT master shall enter the NMT state *Operational* automatically by itself or shall wait until it is requested by the application running on the very same CANopen device.

h) Under the following conditions
— Bit 3 of object $1F80_h$ (see 6.1) is set to 0_b,
— Bit 1 of object $1F80_h$ (see 6.1) is set to 1_b,
— and all NMT slaves listed in $1F81_h$ booted successfully
the NMT service *Start remote node* shall be performed with node-ID set to 0.

Under the following conditions
— Bit 3 of object $1F80_h$ (see 6.1) is set to 0_b,
— Bit 1 of object $1F80_h$ (see 6.1) is set to 1_b,
— and not all NMT slaves listed in $1F81_h$ booted successfully
the NMT service *Start remote node* shall be performed for each NMT slave individually.

i) The process *NMT startup* ended successfully and the NMT master shall proceed with normal operation.

Occurrence and detection of NMT slaves not listed in $1F81_h$ falls into the responsibility of the application.

3.2 NMT startup simple

Since nearly all objects and features are optional it is possible to implement a basic NMT master, which may make sense for some applications. Removing all optional parts in the definitions above results in the process *NMT startup simple* as shown in Figure 3.

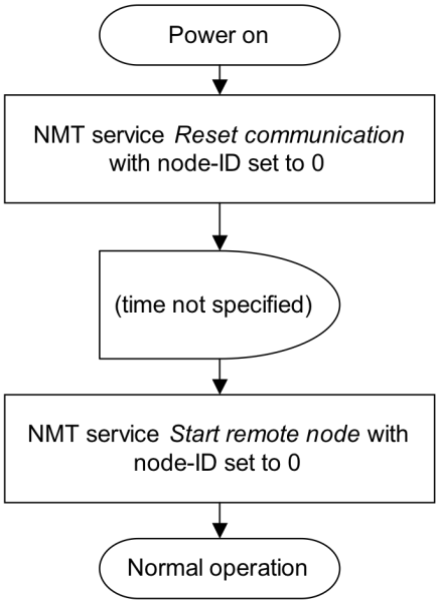

Figure 3 — NMT startup simple

3.3 Start process boot NMT slave

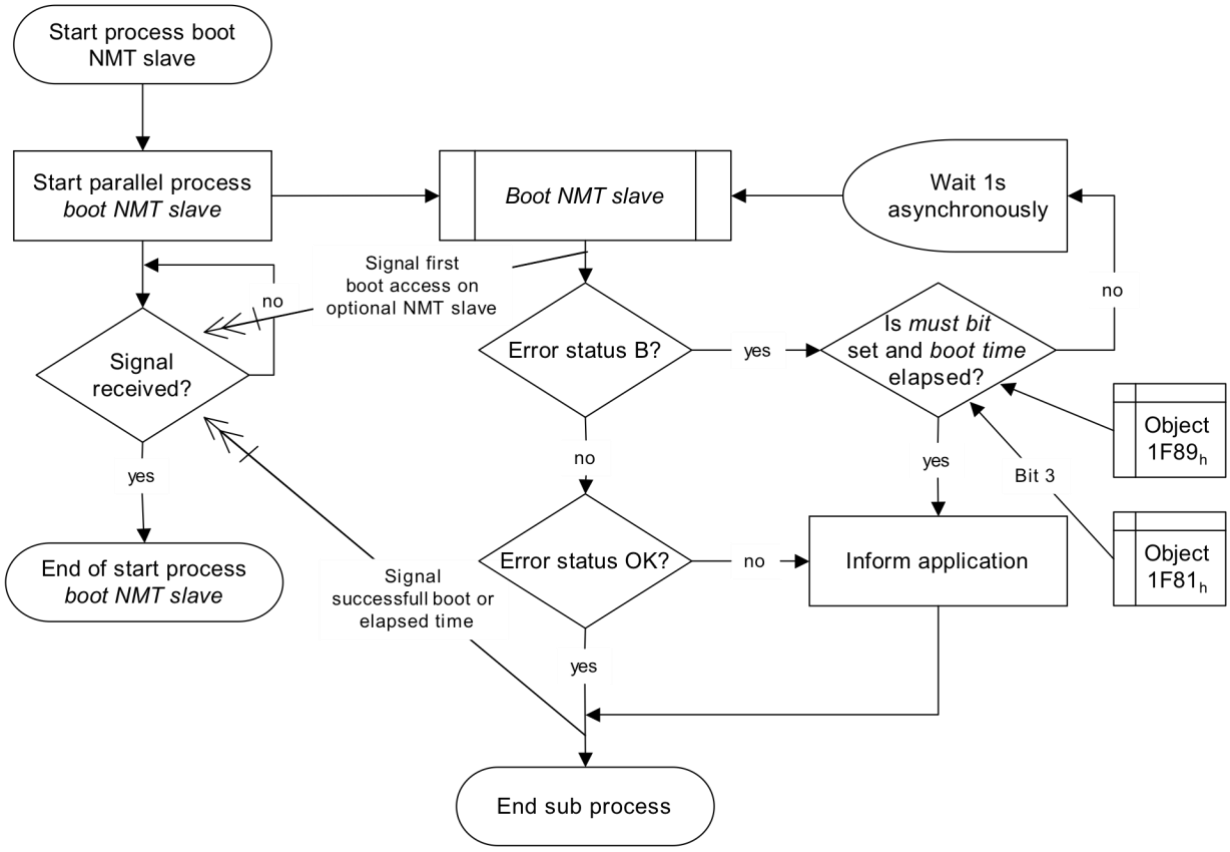

Figure 4 — Start process boot NMT slave

The process *start process boot NMT slave* as shown in Figure 4 shall include the following steps:

a) Start parallel process *boot NMT slave*.

b) Mandatory NMT slaves: wait for completion of the process *boot NMT slave*.
Optional NMT slaves: wait for signal that the process *Boot NMT slave* was performed.

The parallel process shall

a) Perform the process *boot NMT slave* (see 3.4).

b) Create signal for every try of the process *boot NMT slave*.

c) If the process *boot NMT slave* returned with status OK the process shall terminate.
This process shall run endlessly for any optional NMT slave until the process *Boot NMT slave* finishes with status OK. The recommended cycle time is 1 second for bit rate higher than 125 kbps.
If the process *boot NMT slave* returned with error status B for mandatory NMT slaves and the elapsed time is greater than the configured value of object $1F89_h$ (see 6.11) then the application shall be informed and this sub-process shall end.

The sub process of the process *start process boot NMT slave* shall run asynchronously to other processes.

3.4 Boot NMT slave

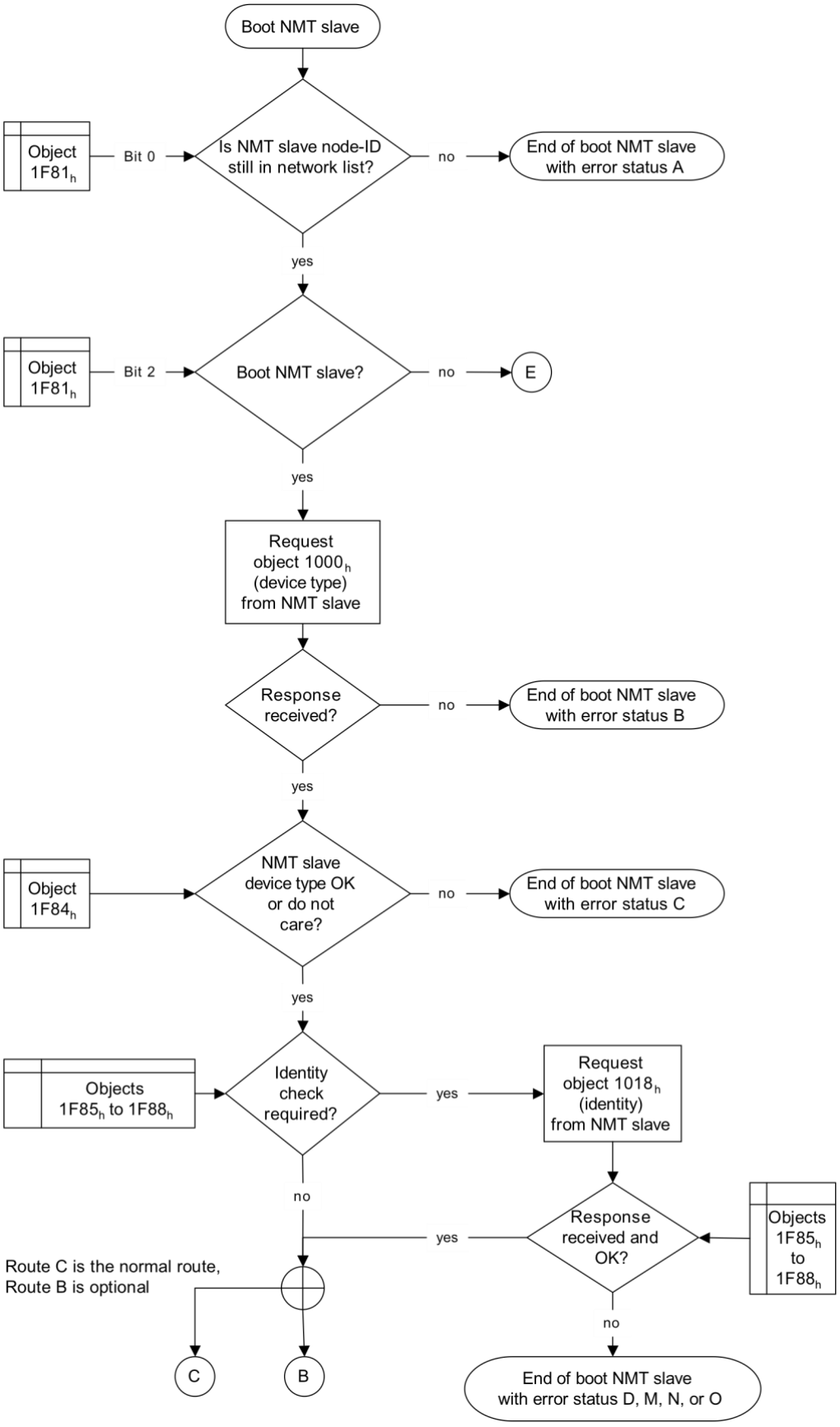

Figure 5 — Boot NMT slave, part 1

The process *boot NMT slave* as shown in Figure 5 consists of the following steps:

a) Bit 0 of object $1F81_h$ (see 6.3) is used to decide upon whether the NMT slave shall be processed or if the process shall terminate with error status.

b) Bit 2 of object $1F81_h$ (see 6.3) is used to decide upon whether the NMT slave shall be configured and started.

c) Upload object 1000_h from the NMT slave. In case no response is received the process shall terminate with error status.

d) In case the value of object $1F84_h$ (see 6.6) for the NMT slave is unequal 0 the value shall be checked against object 1000_h. In case both values are different the process shall terminate with error status.

e) In case the values of objects $1F85_h$ to $1F88_h$ (see 6.7 to 6.10) are unequal 0 the particular object values shall be checked against their corresponding values of the object 1018_h from the NMT slave. In case one of the values of object $1F85_h$ to $1F88_h$ is different to the corresponding values of the object 1018_h from the NMT slave the process shall terminate with error status.

The process *boot NMT slave* may continues with an optional part 2 (see Figure 6), which introduces two optional features:

- keeping alive CANopen devices initially in NMT state *Operational*, and
- managing application software versions including automatic update of the application software.

If the keep alive bit of a CANopen device is set the NMT master must not issue the NMT services *Reset node* and *Reset communication* for this CANopen device.

NOTE Such situations can happen when the NMT master encounters a failure with a subsequent re-start, e.g. power fail.

Software version control may be used in systems where the version of the application software running on the CANopen device is checked for correctness. The process check and update software version may be used to automatically download the most current version of the application software. CANopen devices that detect an error in their application software, for example by calculating a checksum during startup, may force a download of the most current application software by responding with wrong version information.

Both or only one of the two features may be implemented optionally.

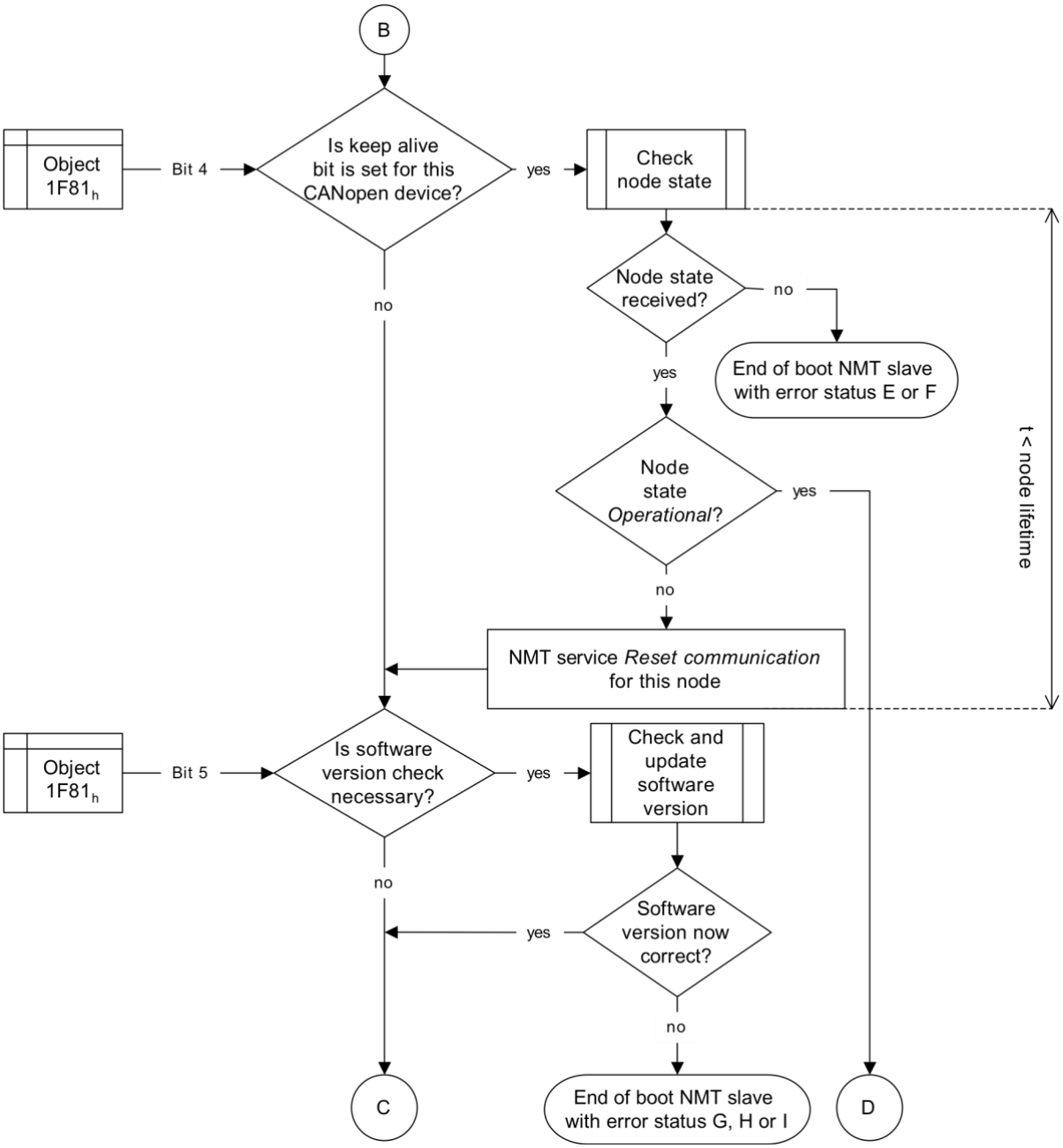

Figure 6 — Boot NMT slave, part 2

a) Bit 4 of object $1F81_h$ (see 6.3) is used to decide upon keep alive is requested.

   In case keep alive is requested the NMT master shall request[2] the current NMT state of the NMT slave for which the *boot NMT slave* is performed. In case no current NMT state is received the process shall finish with error status. In case the current NMT state is *Operational* the process shall continue with D. In case the current NMT state is not the NMT state *Operational* the NMT master shall perform the NMT service *Reset communication* for that NMT slave.

   The process check NMT state is defined in 3.6.

b) Bit 5 of object $1F81_h$ (see 6.3) is used to decide upon whether the application software verification shall be performed.

[2] In case the NMT slave supports Node and Life guarding only the NMT master has to take care that the time from requesting the current NMT state from the NMT slave to issuing the NMT service *Reset communication* is less than the life time as set in the NMT slave.

In case application software verification is requested the process check and update software version shall be performed.

In case the software version will stay incorrect the process Boot NMT slave shall end with error status.

The process *boot NMT slave* continues in part 3 (see

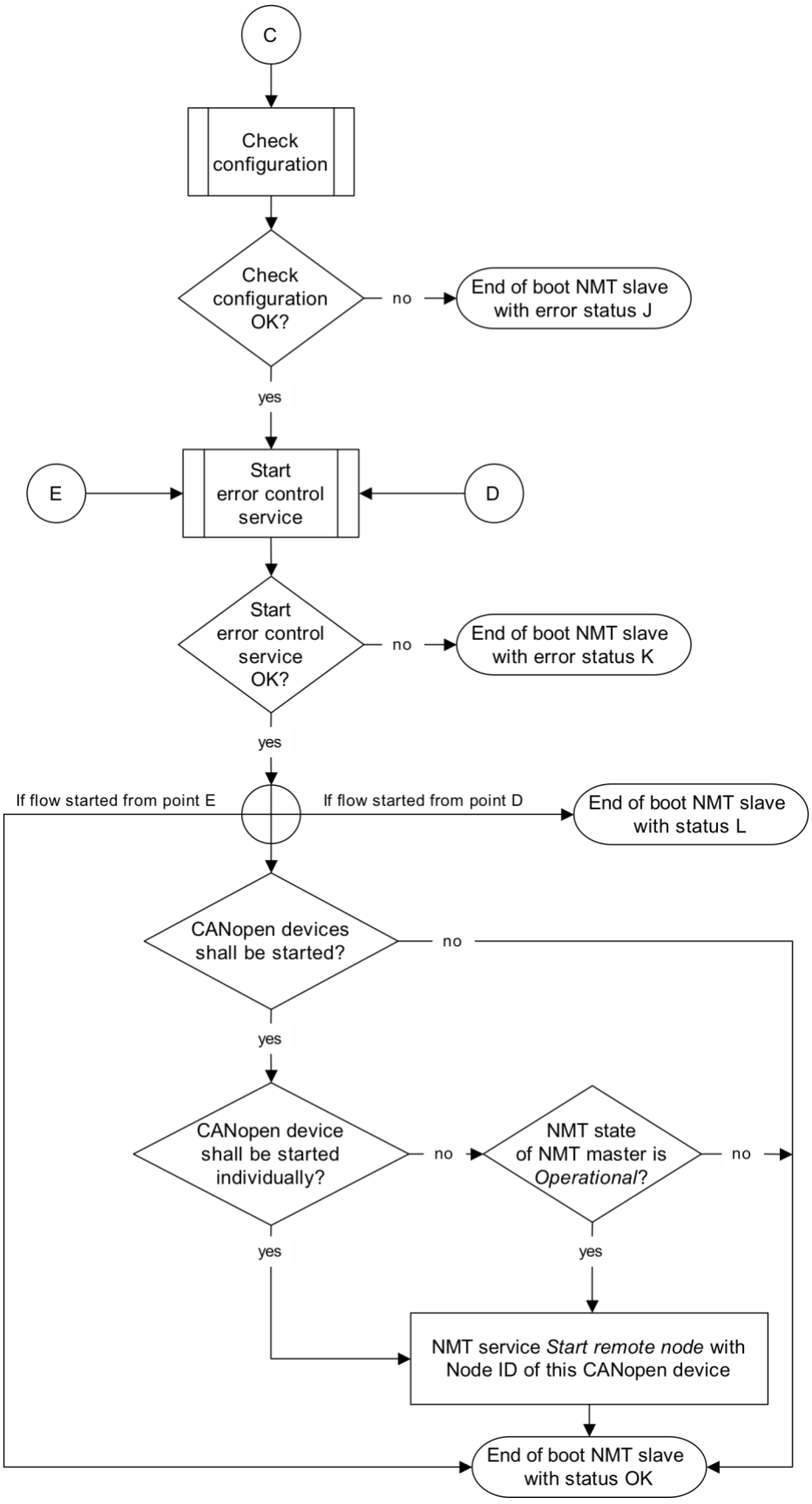

Figure 7) and is mandatory.

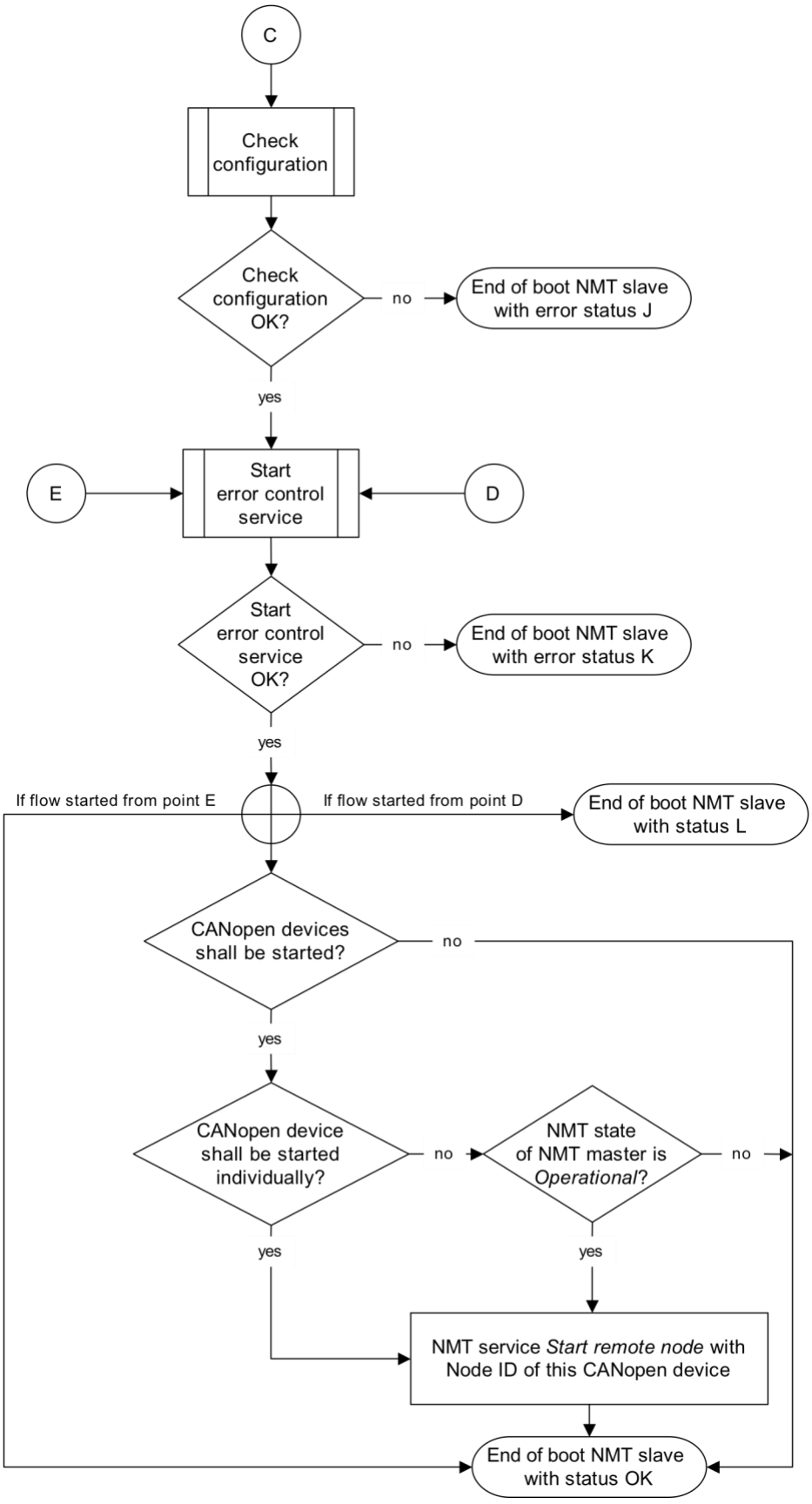

Figure 7 — Boot NMT slave, part 3

c) The process check configuration shall be performed.

d) In case the process check configuration finished unsuccessfully the process *boot NMT slave* shall end with an error status.

e) The process *start error control* (see 4.1) shall be performed.

f) In case the process *start error control* finished unsuccessfully the process *boot NMT slave* shall end with an error status.

g) In case the NMT slave is in NMT state *Operational* the process *boot NMT slave* shall finish successfully.

h) Bit 3 of object $1F80_h$ (see 6.1) is used to decide upon whether the NMT master shall execute the NMT service *Start remote node*.

i) In case the NMT master shall not execute the NMT service *Start remote node* the process *boot NMT slave* shall end successfully with status.

j) Bit 1 of object $1F80_h$ (see 6.1) is used to decide upon whether the NMT master shall execute the NMT service *Start remote node* with node-ID set to 0 or for each NMT slave in the network individually.

k) In case the NMT master shall execute the NMT service *Start remote node* with node-ID set to 0 and the NMT master is not in NMT state *Operational* the process *boot NMT slave* shall finish successfully with status OK.
NOTE: If the NMT master is in NMT state *Operational* it is regarded as an indication that the initial NMT startup has been completed prior to the startup of this NMT slave.

l) In case the NMT master shall execute the NMT service *Start remote node* for each NMT slave in the network individually or in case the NMT master shall execute the NMT service *Start remote node* with node-ID set to 0 and the NMT master is in NMT state *Operational* the NMT master shall execute the NMT service *Start remote node* with node-ID set to the appropriate value.

m) The process *boot NMT slave* shall finish successfully with status.

3.5 Check configuration

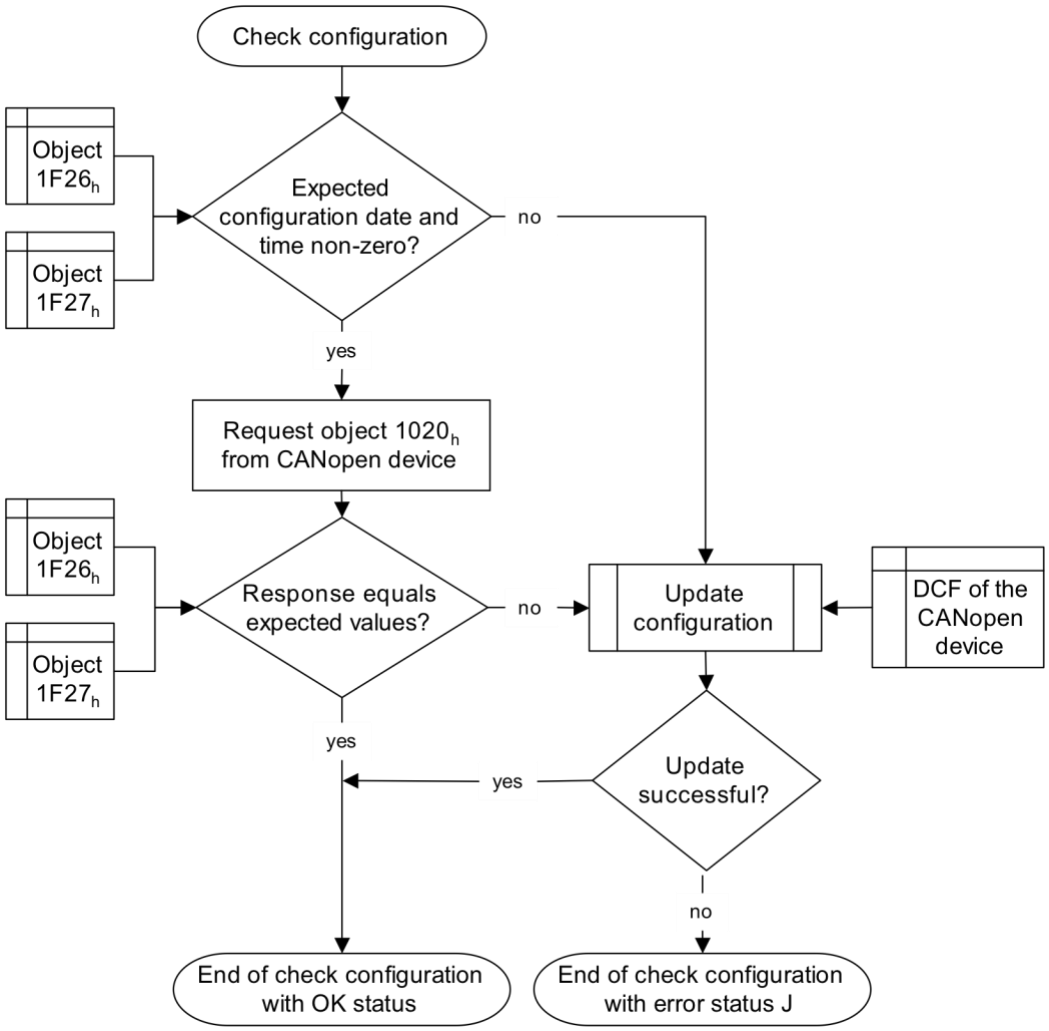

Figure 8 — Check configuration

The process *check configuration* as shown in Figure 8 consists of the following steps:

a) The according entries (sub-index shall equal node-ID) of object 1F26h and object 1F27h is used decide upon whether the CANopen devices configuration is verified. If the values are equals 0 the process shall go to step d).
If the values are not equals 0 the process shall go to step b).

b) Object 1020h shall be requested from the CANopen device.

c) The values of the according entries (sub-index shall equal node-ID) of object 1F26h and object 1F27h shall be compared to the values of the received object 1020h. If the values equals the process shall finish with status OK.
If the values does not equal the process shall go to step d).

d) The configuration shall be updated on the CANopen device.
Depending on the settings of 1F81h and 1F8Ah the configured restore operation shall be executed. Only after the restore operation the NMT service reset communication or NMT service reset node for the corresponding NMT slave shall be issued (see /CiA301/).
The configuration may be derived from any kind of DCF provided for that CANopen device. The DCF may be read from a local file system, object 1F20h, or object 1F22h on the configuration manager.

e) If the download is successful then the process shall finish with status OK. If the download is not successful then the process shall finish with error status.

The process *check configuration* may be skipped if the keep-alive is enabled for that CANopen device and the CANopen device is in the NMT state *Operational*. If this occurs the

process boot NMT slave shall finish with error status (L — "NMT slave was initially operational").

3.6 Check NMT state

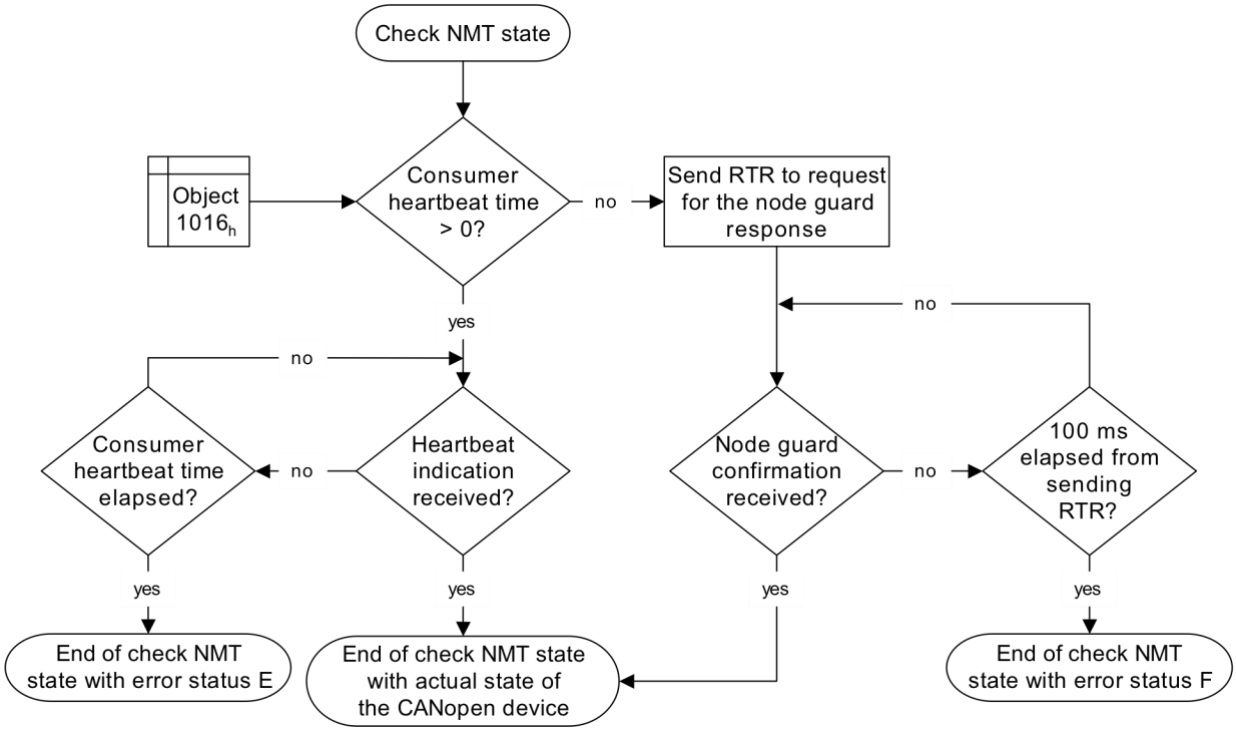

Figure 9 — Check NMT state[3]

The process *check NMT state* as shown in Figure 9 consists of the following steps:

a) Object 1016_h (see /CiA301/) is used to decide upon whether the CANopen device is set up for heartbeat.

b) In case the CANopen device is set up for heartbeat the NMT master shall check if it received a heartbeat indication in time (heartbeat consumer time is not elapsed). If the heartbeat indication is not received in time the process shall end with an error status. If the heartbeat indication is received the process shall end with the actual NMT state of the CANopen device that is checked.

c) In case the CANopen device that is checked is not set up for heartbeat it is assumed that the CANopen device is set up for CANopen device guarding. In this case the NMT master shall request the actual NMT state of the CANopen device that is checked, the NMT service CANopen device guarding shall be used. In case no confirmation is received and more than 100 milliseconds are elapsed the process shall end with an error status. In case a confirmation is received the process shall and with the actual NMT state of the CANopen device that is checked.

[3] Required, if bit 4 of object $1F81_h$ is set (keep alive supported).

3.7 NMT flying master startup

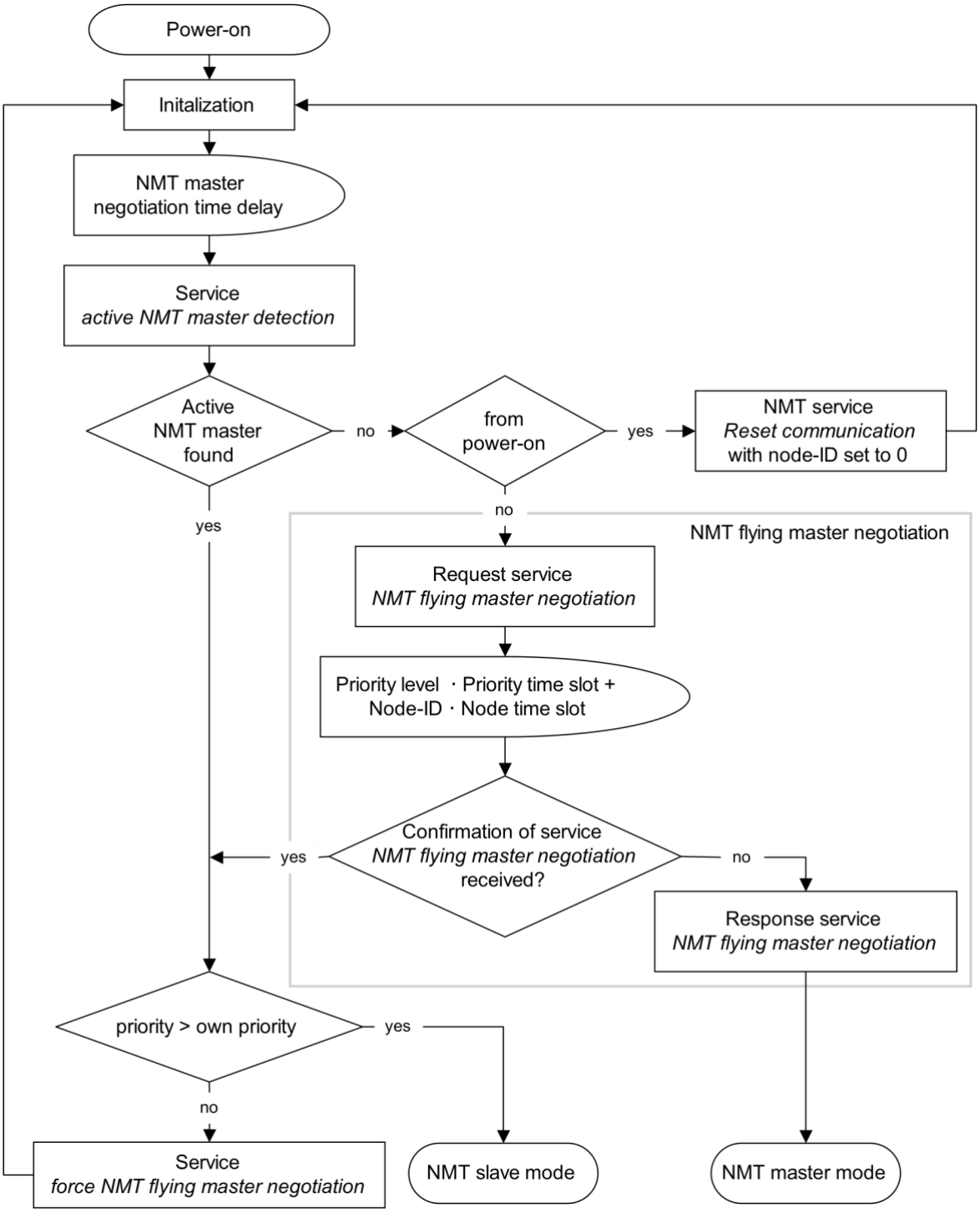

Figure 10 — NMT flying master startup

The process *NMT flying master startup* as shown in Figure 10 consists of the following steps:

a) After power-on the CANopen device shall perform initialization. Initialization is not within the scope of this document.

b) After initialization the CANopen device shall wait until the NMT master negotiation time delay (see object $1F90_h$ sub-index 02_h – see 6.12) has expired.

c) The CANopen device shall request the service *active NMT master detection*.

d) In case no confirmation for the service *active NMT master detection* is received and in case the CANopen device started from power-on the CANopen device shall request the NMT service *Reset communication* with node-ID set to 0.

e) In case no confirmation for the service *active NMT master detection* is received and in case the CANopen device started from the NMT service *Reset communication* the CANopen device shall request the service *NMT flying master negotiation*.

f) In case a confirmation for the service *active NMT master detection* is received and the priority of the active NMT master is greater then the own priority the CANopen device shall enter the NMT slave mode and becomes a NMT slave.

g) In case a confirmation for the service *active NMT master detection* is received and the priority of the active NMT master is lower then the own priority the CANopen device shall request the service *force NMT flying master negotiation*.

h) In case no confirmation for the service *NMT flying master negotiation* is received the CANopen device shall respond the service *NMT flying master negotiation* with its own priority and its own node-ID and shall enter the NMT master mode and becomes the NMT master.

i) In case a confirmation for the service *NMT flying master negotiation* is received and the priority of the active NMT master is lower then the own priority the CANopen device shall request the service *force NMT flying master negotiation*.

3.8 Error status

The errors as shown in Table 1 shall be signaled within the NMT master during startup.

Table 1 — Error status

Error status	Description
A	The CANopen device is not listed in object $1F81_h$.
B	No response received for upload request of object 1000_h.
C	Value of object 1000_h from CANopen device is different to value in object $1F84_h$ (Device type).
D	Value of object 1018_h sub-index 01_h from CANopen device is different to value in object $1F85_h$ (Vendor-ID).
E	Heartbeat event. No heartbeat message received from CANopen device.
F	Node guarding event. No confirmation for guarding request received from CANopen device.
G	Objects for program download are not configured or inconsistent.
H	Software update is required, but not allowed because of configuration or current status.
I	Software update is required, but program download failed.
J	Configuration download failed.
K	Heartbeat event during start error control service. No heartbeat message received from CANopen device during start error control service.
L	NMT slave was initially operational. (CANopen manager may resume operation with other CANopen devices)
M	Value of object 1018_h sub-index 02_h from CANopen device is different to value in object $1F86_h$ (Product code).
N	Value of object 1018_h sub-index 03_h from CANopen device is different to value in object $1F87_h$ (Revision number).
O	Value of object 1018_h sub-index 04_h from CANopen device is different to value in object $1F88_h$ (Serial number).

4 Error control

4.1 Start error control

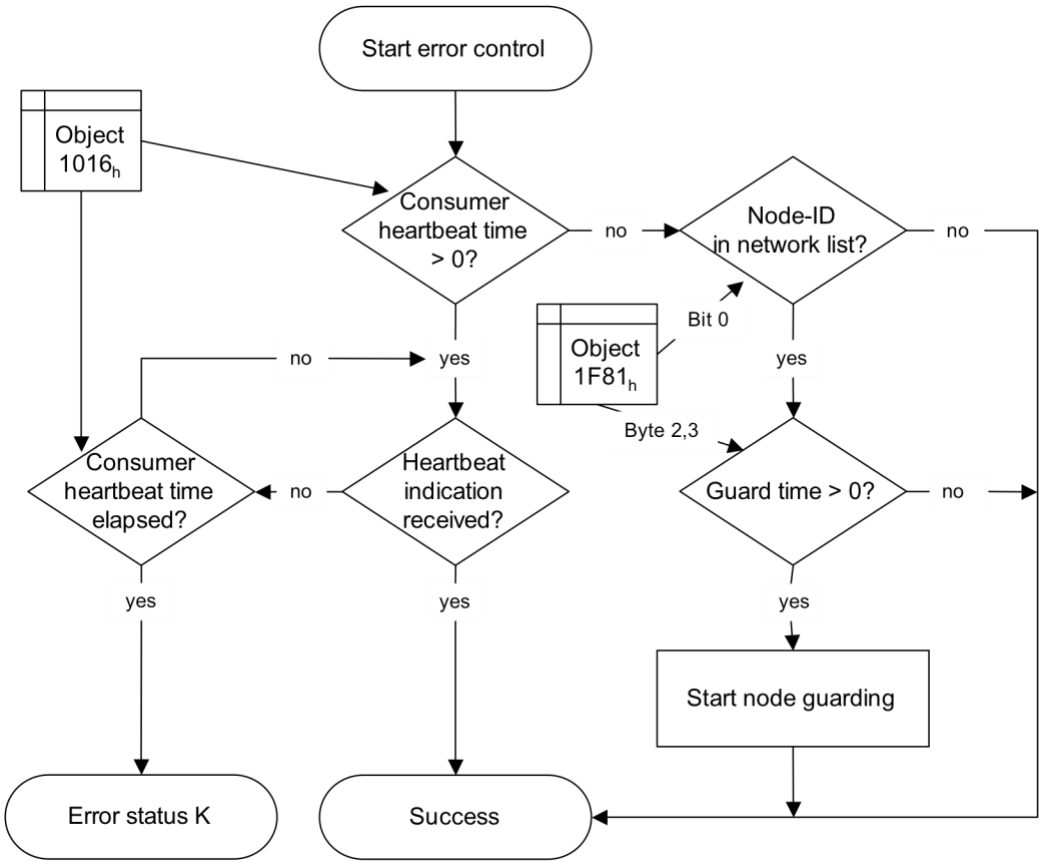

Figure 11 — Start error control

The process *start error control* as shown in Figure 11 consists of the following steps:

a) Object 1016_h (see /CiA301/) shall be checked to decide upon whether the CANopen device is set up for heartbeat.

b) In case the CANopen device is set up for heartbeat and no heartbeat indication is received in time the process shall end with an error status. The timer for timeout checking shall be started immediately with the process itself.

c) In case the CANopen device is set up for heartbeat and a heartbeat indication is received in time the process shall end successfully.

d) In case the CANopen device is not set up for heartbeat bit 0 of object $1F81_h$ is used to decide upon whether the CANopen device shall be guarded.

e) In case the CANopen device is not in the network list the process shall end successfully.

f) In case the CANopen device is in the network list the value of bytes 2 and 3 of object $1F81_h$ is used to decide whether upon the CANopen device shall be guarded.

g) In case the value is greater than 0 guarding shall be started for that CANopen device and the process shall end successfully.

h) In case the value equals 0 the process shall end successfully.

4.2 Error handler

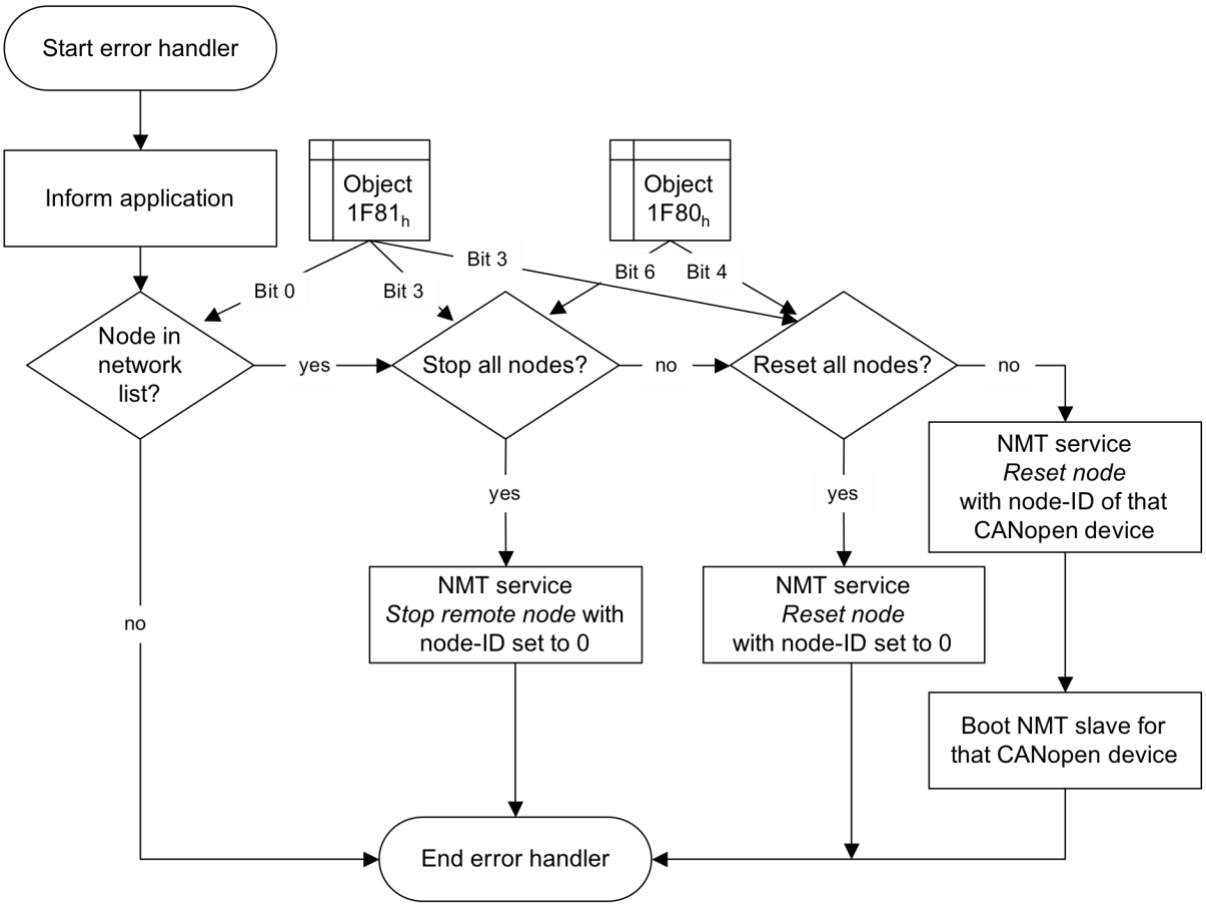

Figure 12 — Error handler

The process *error handler* as defined in Figure 12 shall be initiated any time an NMT error event occurs.

4.3 Bootup handler

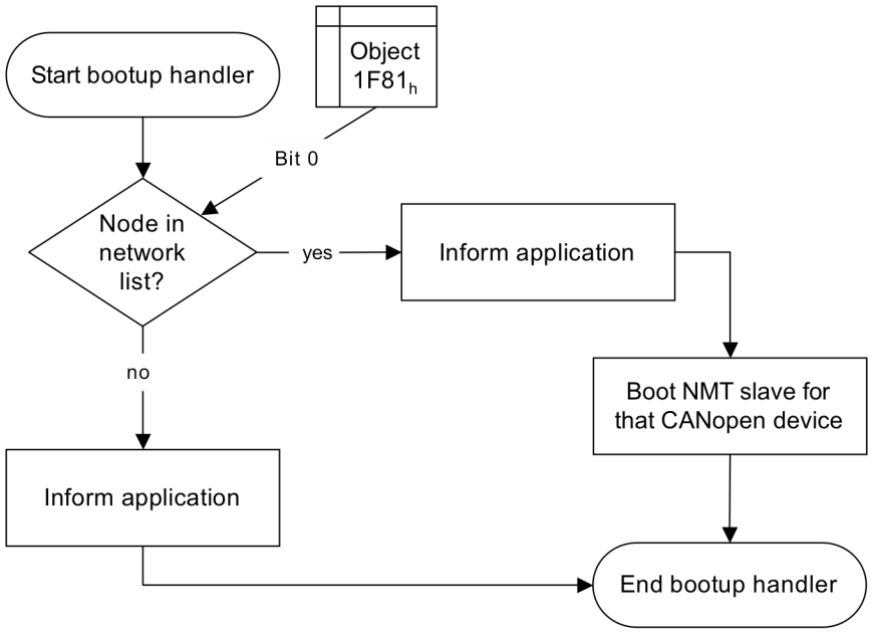

Figure 13 — Bootup handler

The process *bootup handler* as defined in Figure 13 shall be initiated any time a bootup event occurs (see */CiA301/*).

5 Additional NMT master services and protocols

5.1 NMT master detection

5.1.1 NMT master detection service

Through this service a CANopen device is able to check if there are CANopen devices in the network available that are capable of the NMT master mode. The service *NMT master detection* may be performed by one CANopen device or by many CANopen devices at the very same time. Table 2 defines the service parameters.

The service *NMT master detection* may be initiated by any CANopen device that is capable of the NMT master mode or that is able to self start (*see object 1F80$_h$*). The service shall be initiated by the CANopen device after the CANopen device switched into the NMT state *Pre-operational* and an additional time delay (*see object 1F91$_h$ sub-index 02$_h$*). The service shall be confirmed and the remote result parameter shall indicate the success of the request. The confirmation for this service shall be received within a specified time (*see object 1F91$_h$ sub-index 01$_h$*). In case no confirmation is received the CANopen device shall assume a failure (negative success) and may execute the NMT service *Start remote node* for each CANopen device separately or with node-ID set to 0 after a specified time, which is the result of the following calculation:

$$wait\ time = Node\text{-}ID \times Node\ time\ slot$$

The CANopen device time slot shall be given by configuration (*see object 1F91$_h$ sub-index 03$_h$*).

Table 2 — NMT master detection

Parameter	Request / Indication	Response / Confirmation
Argument none	Mandatory / Optional	
Remote result Success		Mandatory / Optional Mandatory

5.1.2 NMT master detection protocol

This protocol as defined in Figure 14 shall be used to implement the service *NMT master detection*.

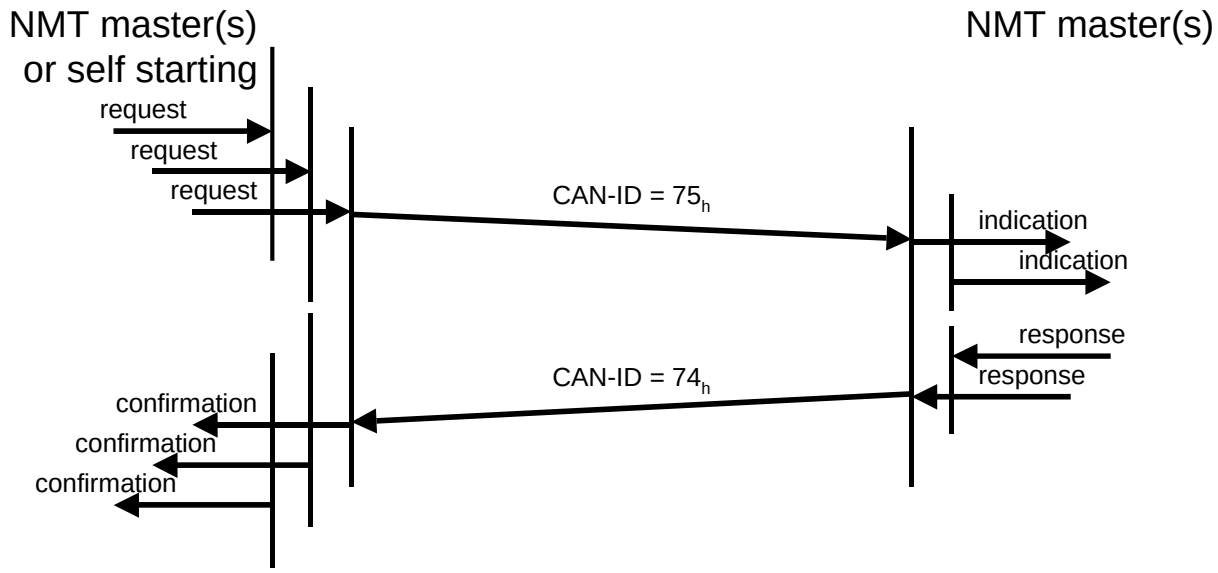

Figure 14 — NMT master detection

5.2 Active NMT master detection

5.2.1 Active NMT master detection service

Through this service a CANopen device is able to check if there is a CANopen device in the network available that is in NMT master mode. The service *active NMT master detection* may be performed by one CANopen device or by many CANopen devices at the very same time. Table 3 defines the service parameters.

Any CANopen device that is capable of the NMT master mode and able to participate the NMT flying master process may initiate the service *active NMT master detection*. The CANopen device may initiate the service at any time. The service shall be confirmed and the remote result parameter shall indicate the success of the request. In case of no failure (positive success) the node-ID of the actual NMT master and its priority level shall be given (*see object 1F90h sub-index 03h*). The confirmation for this service shall be received within a specified time (*see object 1F90h sub-index 01h*). In case a CANopen device that is capable of the NMT master mode detects that the priority level of the actual NMT master is lower than its own priority level this CANopen device may execute the service *force NMT flying master negotiation*. In case no confirmation is received the CANopen device shall assume a failure (negative success) and may execute the service *NMT flying master negotiation*.

Table 3 — Active NMT master detection

Parameter	Request / Indication	Response / Confirmation
Argument none	Mandatory / Optional	
Remote result Success Node-ID Priority level		Mandatory / Optional Mandatory Mandatory Mandatory

5.2.2 Active NMT master detection protocol

This protocol as defined in Figure 15 shall be used to implement the service *active NMT master detection*.

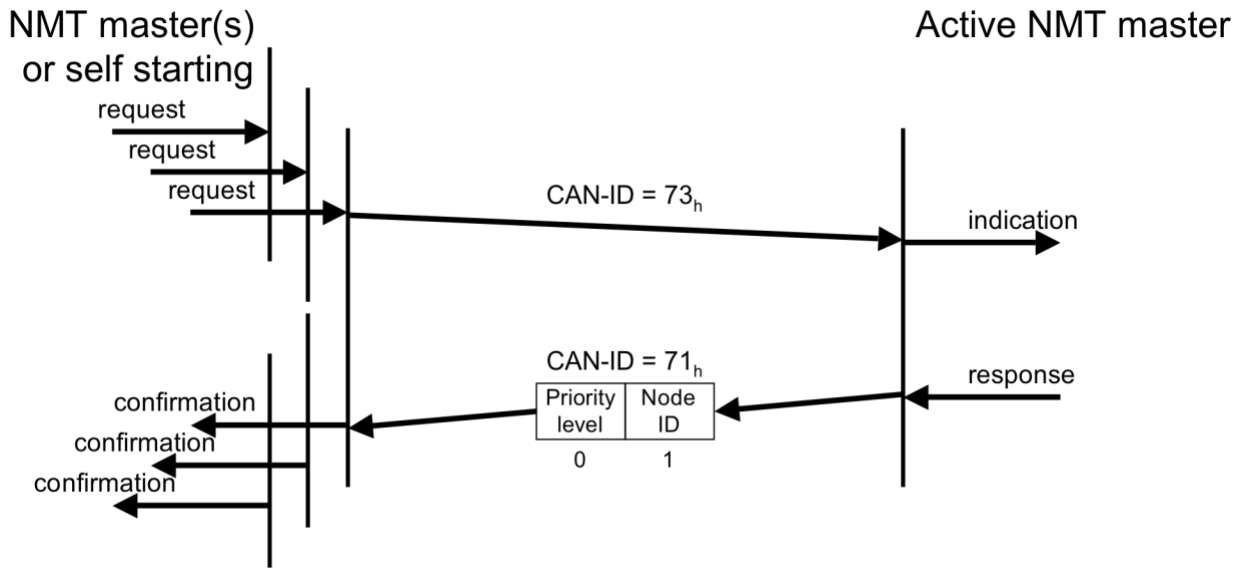

Figure 15 — Active NMT master detection

5.3 NMT flying master negotiation

5.3.1 NMT flying master negotiation service

Through this service it is evaluated which CANopen device in the network is capable of the NMT master mode, able to participate the NMT flying master process and is having the highest priority level, that means which CANopen device is the highest prior CANopen device and able to switch in NMT master mode. The service *NMT flying master negotiation* may be initiated by one CANopen device or by many CANopen devices at the very same time. In case a request is indicated other CANopen devices in the network shall immediately start their waiting time according to the result of the following formula:

$$waiting\,time = Priority\,level \times Priority\,time\,slot + Node\text{-}ID \times CANopen\,device\,time\,slot$$

Table 4 defines the service parameters.

Any CANopen device that is capable of the NMT master mode and able to participate the NMT flying master process may initiate the service *NMT flying master negotiation*. Any CANopen device may initiate the service after the unsuccessful completion of the service *active NMT master detection* or after the service *force NMT flying master negotiation*. The service shall be confirmed and the remote result parameter shall indicate the success of the request. In case of no failure (positive success) the node-ID of the new actual NMT master and its priority level shall be given. The confirmation shall be given after a specified waiting time calculated by the above given formula.

Table 4 — NMT flying master negotiation

Parameter	Request / Indication	Response / Confirmation
Argument none	Mandatory / Optional	
Remote result Success Node-ID Priority level		Mandatory / Optional Mandatory Mandatory Mandatory

5.3.2 NMT flying master negotiation protocol

This protocol as defined in Figure 16 shall be used to implement the service *NMT flying master negotiation*.

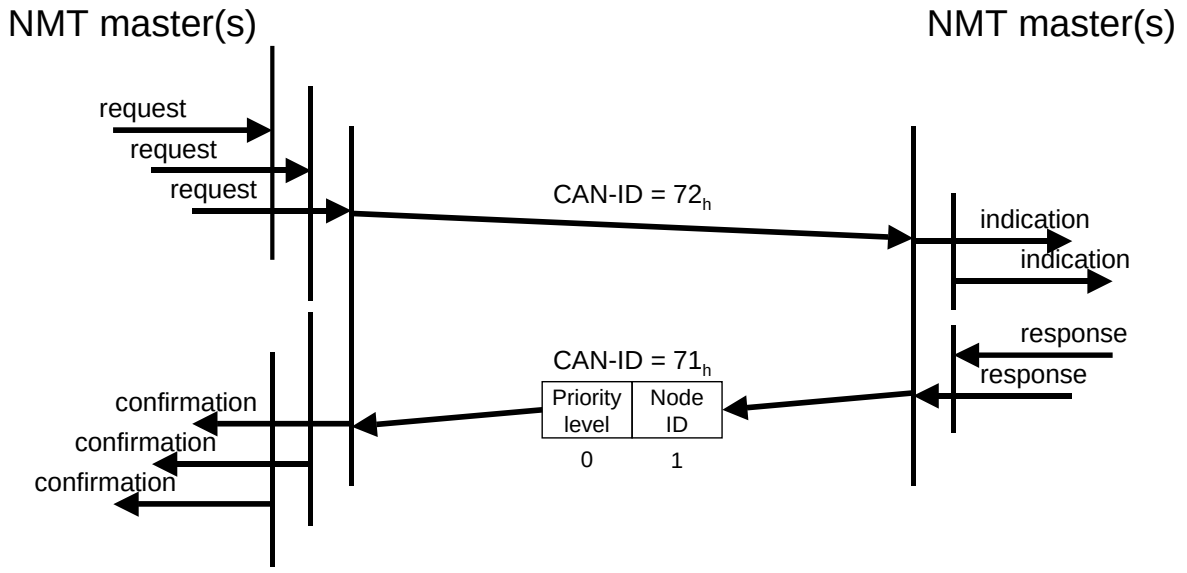

Figure 16 — NMT flying master negotiation

5.4 Force NMT flying master negotiation

5.4.1 Force NMT flying master negotiation service

Through this service the service *NMT flying master negotiation* shall be forced. Any CANopen device in the network that is capable of the NMT master mode and able to participate in the NMT flying master process at any time may initiate the service *force NMT flying master negotiation*. Normally a CANopen device initiates it after the service *active NMT master detection* and in case the CANopen device detected that its own priority level is higher than the priority level of the CANopen device identified as the active NMT master. Table 5 defines the service parameters.

After the NMT flying master negotiation service the active NMT master shall request the NMT service *Reset communication* for every CANopen device in the network or with node-ID set to 0 to be executed by the CANopen device that is in NMT master mode.

Table 5 — Active NMT master detection

Parameter	Request / Indication	Response / Confirmation
Argument none	Mandatory / Optional	

5.4.2 Force NMT flying master negotiation protocol

This protocol as defined in Figure 17 shall be used to implement the service *force NMT flying master negotiation*.

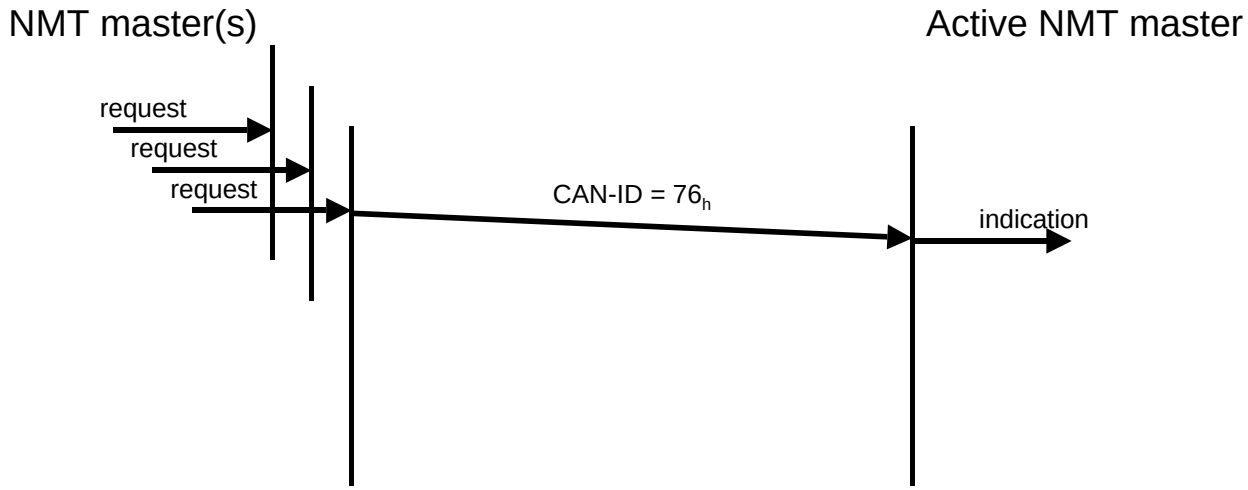

Figure 17 — Force NMT flying master negotiation

5.5 Detection of Failures

5.5.1 General

In a CANopen network only one active NMT master is allowed. Due to failure conditions problems will occur which shall be resolved automatically by the following measures:

- When the active NMT master fails, a backup NMT master shall detect the loss of the active NMT master and correct the situation.
- When two or more NMT master become active at the same time this shall be detected by the NMT masters and corrected. This situation can happen when the network was temporarily split, causing two NMT masters becoming active.

5.5.2 Detection of an NMT master failure

The loss of the active NMT master shall be detected by every NMT master capable CANopen device by monitoring the heartbeat of the active NMT master. The node-ID of the active NMT master shall be captured during the service *active NMT master detection*.

The NMT master negotiation shall be initiated by initiating the NMT service *Reset communication* with node-ID set to 0.

NOTE 1 It is not possible to use the service *force NMT flying master negotiation*, since there is no NMT master that is able to initiate the NMT service *Reset node*.

NOTE 2 When an NMT master supports the heartbeat consumer (1016_h), all NMT master capable CANopen devices shall be configured by a configuration tool. When an NMT master do not support this object, the NMT master shall configure the internal receive object for reception of the heartbeat message of the active NMT master.

5.5.3 Detection of multiple NMT masters

Active NMT masters shall cyclically initiate the service *force NMT flying master negotiation* without itself initiating the NMT service *Reset communication*. The cycle time period is configured in object $1F90_h$ sub-index 06_h of the NMT flying master timing parameters.

If multiple NMT flying masters exist, they will be forced into a reset by this mechanism. The very rarely case that both (or more) initiate the service at the same time will have the effect that they do not recognize it. In order to resolve this situation the cycle time periods of all NMT flying master shall be configured with different values.

6 Object dictionary

6.1 Object $102A_h$ – NMT inhibit time

This object shall indicate the configured inhibit time between two subsequent NMT messages. The outstanding NMT services shall be queued and shall be issued in order of their

occurrence respecting the configured inhibit time. Table 6 and Table 7 define the object description and the entry description.

The value shall be given in multiples of 100 µs. The value 0 shall disable the inhibit time.

Table 6 — Object description

Attribute	Value
Index	$102A_h$
Name	NMT inhibit time
Object code	VAR
Data type	UNSIGNED16
Category	Mandatory

Table 7 — Entry description

Attribute	Value
Sub-index	00_h
Access	rw; const, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	0

6.2 Object $1F80_h$ – NMT startup

This object shall configure the startup behavior of a CANopen device. Internal state transitions shall not change the value of this object. An attempt to change a bit of a functionality that is not supported by the CANopen device shall be responded with an abort message (abort code: $0800\ 0000_h$ or $0609\ 0030_h$). Figure 18 and Figure 19 defines the bit-oriented structure of the value. Table 8, Table 9, Table 10, Table 11, Table 12, Table 13, and Table 14 define the allowed values. Table 15 shows pre-definitions for flying NMT master devices. Table 16 defines exceptions for start-up capable devices. Table 17 and Table 18 define the object description and the entry description.

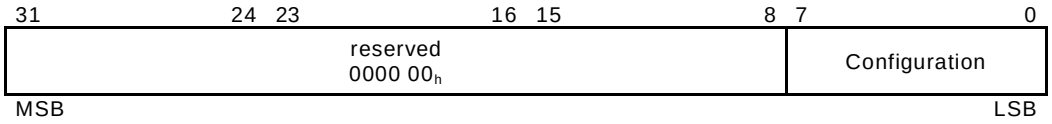

Figure 18 — object $1F80_h$ – bit structure of the value

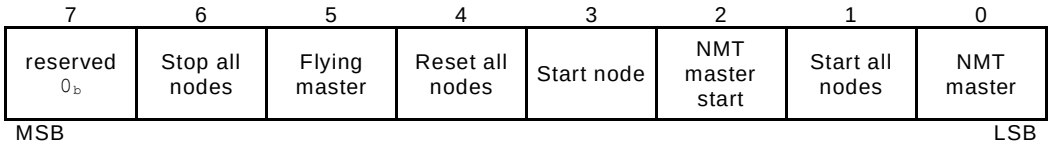

Figure 19 — object $1F80_h$ – bit structure of the configuration value

Table 8 — Value NMT master (bit: 0)

Value	Description
0_b	CANopen device is not NMT master The entries of the object $1F81_h$ shall be ignored. All other bits of object $1F80_h$ shall be ignored with the exceptions defined in Table 16.
1_b	CANopen device is the NMT master

Table 9 — Value Start all nodes (bit: 1)

Value	Description
0_b	NMT service *start remote node* for each node-ID
1_b	NMT service *start remote node* with node-ID = 0

Table 10 — Value NMT master start (bit: 2)

Value	Description
0_b	Shall switch into NMT state *Operational* in the process NMT startup (see Figure 2)
1_b	Shall not switch into the NMT state *Operational* by itself.

Table 11 — Value Start node (bit: 3)

Value	Description
0_b	The NMT master shall start the NMT slaves.
1_b	The NMT master shall not start the NMT slaves and the application may start the NMT slaves.

Table 12 — Reset all nodes (bit: 4)

Value	Description
0_b	In case of error control event of a CANopen device defined as mandatory (*see object 1F81$_h$*) the NMT service *reset node* with node-ID of the CANopen device that caused the error control event shall be executed.
1_b	In case of error control event of a CANopen device defined as mandatory (*see object 1F81$_h$*) the NMT service *reset node* with node-ID = 0 shall be executed.

Table 13 — Flying master (bit: 5)

Value	Description
0_b	CANopen device shall not participate the NMT flying master negotiation
1_b	CANopen device shall participate the NMT flying master negotiation (*see* 5.3)

Table 14 — Stop all nodes (bit: 6)

Value	Description
0_b	In case of error control event of a CANopen device defined as mandatory (*see object 1F81$_h$*) the action as defined by bit 4 shall be executed.
1_b	In case of error control event of a CANopen device defined as mandatory (*see object 1F81$_h$*) the NMT service *Stop remote node* with node-ID = 0 shall be executed. Bit 4 shall be ignored.

Table 15 — Pre-defined behaviors for flying NMT master

Value	Description
00000000 00000000 00000000 0x0xxxx1$_b$	NMT master without participating in the NMT flying master process
00000000 00000000 00000000 0x1xxxx1$_b$	NMT master participating in the NMT flying master process and bit 0 shall stay 1_b always, even in case it loses the flying NMT master negotiation. In that case bit 2 shall not be evaluated. The CANopen device shall store internally that it is not the current NMT master. If the CANopen device becomes NMT master it shall continue the NMT boot-up by evaluating all other bits.

Table 16 — Exceptions for NMT start-up capable devices

Value	Description
00000000 00000000 00000000 00001000$_b$	NMT slave that shall enter the NMT state *Operational* after the NMT state *Initialisation* autonomously (self starting)
00000000 00000000 00000000 00000010$_b$	NMT slave that shall execute the NMT service *start remote node* with node-ID set to 0

Table 17 — Object description

Attribute	Value
Index	1F80$_h$
Name	NMT startup
Object code	VAR
Data type	UNSIGNED32
Category	Conditional; Mandatory if the CANopen device is a CANopen manager or a start-up capable CANopen device.

Table 18 — Entry description

Attribute	Value
Sub-index	00$_h$
Access	rw; const, if rw is not supported (e.g. self starting)
PDO mapping	No
Value range	*see value definition*
Default value	No

6.3 Object 1F81$_h$ – NMT slave assignment

This object shall assign CANopen devices to the NMT master, the device that shall implement this object. Each sub-index of this object shall correspond to the node-ID of the according CANopen device in the network. The sub-index corresponding to its own node-ID shall be ignored. An attempt to change a bit of a functionality that is not supported by the CANopen device shall be responded with an abort message (abort code: 0800 0000$_h$ or 0609 0030$_h$).

Figure 20 and Figure 21 define the bit-oriented structure of the value. Table 19, Table 20, Table 21, Table 22, Table 23, Table 24, and Table 25 define the value contents. Table 26 and Table 27 define the object description and the entry description.

The value for the retry factor indicates the number of retries the NMT master shall issue in case of a node guarding event (see /CiA301/). The value 0 shall disable node guarding for the CANopen device.

The value for the guard time shall indicate the cycle time for the node guarding of the CANopen device. The value shall be indicated in multiples of ms. The value 0 shall disable node guarding for the CANopen device.

NOTE If the heartbeat consumer object is configured to a value unequal 0, then the heartbeat mechanism will have priority over node guarding.

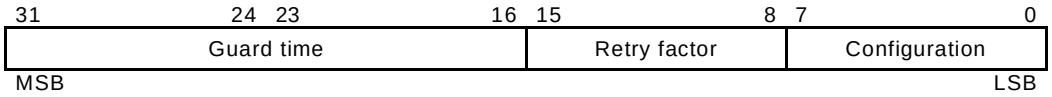

Figure 20 — object 1F81$_h$ – bit structure of the value

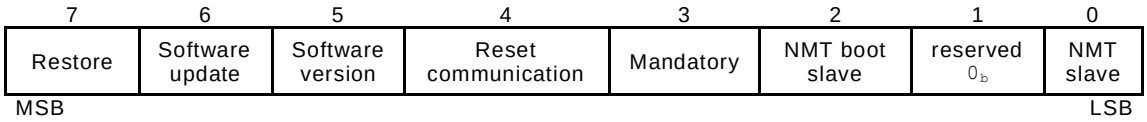

Figure 21 — object 1F81$_h$ – bit structure of the configuration value

Table 19 — NMT slave (bit: 0)

Value	Description
0_b	NMT master or not available in the network
1_b	NMT slave and available in the network

Table 20 — NMT boot slave (bit: 2)

Value	Description
0_b	Configuration and NMT service *Start remote node* shall not be allowed in case of error control event or NMT service *Bootup* NOTE The application is responsible for the NMT slave startup.
1_b	Configuration and NMT service *Start remote node* shall be performed in case of error control event or NMT service *Bootup*

Table 21 — Mandatory (bit: 3)

Value	Description
0_b	CANopen device may be present prior to network startup (CANopen device is optional)
1_b	CANopen device shall be present prior to network startup (CANopen device is mandatory)

Table 22 — Reset communication (bit: 4)

Value	Description
0_b	NMT service *Reset communication* may be executed for the CANopen device at any time
1_b	NMT service *Reset communication* shall not be executed for the CANopen device in case the CANopen device is in NMT state *Operational*

Table 23 — Software version (bit: 5)

Value	Description
0_b	Software version verification shall not be performed for the CANopen device
1_b	Software version verification shall be performed for the CANopen device

Table 24 — Software update (bit: 6)

Value	Description
0_b	Software update shall not be performed for the CANopen device
1_b	Software update shall be performed for the CANopen device

Table 25 — Restore (bit: 7)

Value	Description
0_b	CANopen device may be used without prior resetting.
1_b	CANopen device shall be reset to factory defaults by issuing a restore to defaults (object 1011_h)

Table 26 — Object description

Attribute	Value
Index	$1F81_h$
Name	NMT slave assignment
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 27 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw
PDO mapping	No
Value range	see Figure 20, Figure 21,Table 19, Table 20, Table 21, Table 22, Table 23, Table 24, and Table 25
Default value	$0000\ 0000_h$
to	

Attribute	Value
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw
PDO mapping	No
Value range	see Figure 20, Figure 21,Table 19, Table 20, Table 21, Table 22, Table 23, Table 24, and Table 25
Default value	$0000\ 0000_h$

6.4 Object $1F82_h$ – Request NMT

This object shall request a specific NMT service for a unique CANopen device in the network or for all CANopen devices in the network in case the CANopen device implementing this object is in NMT master mode. Normally, the request is issued by another CANopen device (e.g. configuration tool) in the network or by the application on the very same CANopen device (e.g. in an IEC 61131 environment).

This object shall indicate the current NMT state of a unique CANopen device in the network in case the CANopen device implementing this object is in NMT master mode. Normally, the NMT state is indicated to another CANopen device (e.g. configuration tool) in the network or to the application on the very same CANopen device (e.g. in an IEC 61131 environment).

The sub-index shall correspond to the node-ID of the CANopen devices in the network. These requests may apply for the NMT master itself. Table 28 defines the value contents. Table 29 and Table 30 define the object description and the entry description.

The values from 84_h to $8F_h$ require the knowledge of the node-ID of the requesting CANopen device. If the node-ID is unknown the service shall be aborted (abort code: $0800\ 0000_h$ or $0609\ 0030_h$).

An attempt to download a value that is reserved shall be responded with an abort message (abort code: $0800\ 0000_h$ or $0609\ 0030_h$).

NOTE The values from 00_h to $7F_h$ have to be applied carefully to not unintentionally affect the NMT master or the requesting CANopen device itself.

Table 28 — Value definition

Value	Description	
	on upload (read)	on download (write)
00_h	NMT state unknown	reserved
01_h	CANopen device missing	reserved
02_h	reserved	
03_h	reserved	
04_h	NMT state *Stopped*	NMT service *Stop remote node*
05_h	NMT state *Operational*	NMT service *Start remote node*
06_h	reserved	NMT service *Reset node*
07_h	reserved	NMT service *Reset communication*
08_h	reserved	
:::::	:::::	
$7E_h$	reserved	
$7F_h$	NMT state *Pre-operational*	NMT service *Enter pre-operational*

Value	Description	
	on upload (read)	on download (write)
80_h	reserved	
:::::	:::::	
83_h	reserved	
84_h	NMT state *Stopped*	NMT service *Stop remote node* (excluding NMT master and requesting CANopen device)
85_h	NMT state *Operational*	NMT service *Start remote node* (excluding NMT master and requesting CANopen device)
86_h	reserved	NMT service *Reset node* (excluding NMT master and requesting CANopen device)
87_h	reserved	NMT service *Reset communication* (excluding NMT master and requesting CANopen device)
88_h	reserved	
:::::	:::::	
$8E_h$	reserved	
$8F_h$	NMT state *Pre-operational*	NMT service *Enter pre-operational* (excluding NMT master and requesting CANopen device)
90_h	reserved	
:::::	:::::	
FF_h	reserved	

Table 29 — Object description

Attribute	Value
Index	$1F82_h$
Name	Request NMT
Object code	ARRAY
Data type	UNSIGNED8
Category	Optional

Table 30 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	80_h
Default value	80_h

Attribute	Value
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; ro or wo, if rw is not supported
PDO mapping	No
Value range	see Table 28
Default value	00_h
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; ro or wo, if rw is not supported
PDO mapping	No
Value range	see Table 28
Default value	00_h
Sub-index	80_h
Description	All nodes
Entry category	Mandatory
Access	wo
PDO mapping	No
Value range	see Table 28
Default value	00_h

6.5 Object $1F83_h$ – Request node guarding

This object shall request node guarding for a unique CANopen device in the network or for all CANopen devices in the network in case the CANopen device implementing this object is in NMT master mode and node guarding is enabled. Normally, the request is issued by another CANopen device (e.g. configuration tool) in the network or by the application on the very same CANopen device (e.g. in an IEC 61131 environment).

This object shall indicate the node guarding state for a unique CANopen device in the network in case the CANopen device implementing this object is in NMT master mode and node guarding is enabled. Normally, the node guarding state is indicated to another CANopen device (e.g. configuration tool) in the network or to the application on the very same CANopen device (e.g. in an IEC 61131 environment).

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored. Table 31 defines the value contents. Table 32 and Table 33 define the object description and the entry description.

An attempt to download a value that is reserved shall be responded with an abort message (abort code: $0800\ 0000_h$ or $0609\ 0030_h$).

NOTE Heartbeat consumer is covered by object 1016_h (see /CiA301/).

Table 31 — Value definition

Value	Description	
	on download (write)	on upload (read)
00_h	Stop node guarding	Node guarding stopped
01_h	Start node guarding	Node guarding started
02_h	reserved	
:::::	:::::	
FF_h	reserved	

Table 32 — Object description

Attribute	Value
Index	$1F83_h$
Name	Request node guarding
Object code	ARRAY
Data type	UNSIGNED8
Category	Optional

Table 33 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	80_h
Default value	80_h
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; ro or wo, if rw is not supported
PDO mapping	No
Value range	see Table 31
Default value	00_h

Attribute	Value
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; ro or wo, if rw is not supported
PDO mapping	No
Value range	see Table 31
Default value	00_h
to	
Sub-index	80_h
Description	All nodes
Entry category	Mandatory
Access	wo
PDO mapping	No
Value range	see Table 31
Default value	00_h

6.6 Object $1F84_h$ – Device type identification

This object is used for verification of the device type of the CANopen devices in the network.

The device type (object 1000_h – *see /CiA301/*) of the CANopen device in the network shall be matched against the value of this object in case the value is unequal 0. An error event shall be generated if the values mismatch. In case the value of this object is 0 the device type of the CANopen device in the network may not verified. Table 34 and Table 35 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

Table 34 — Object description

Attribute	Value
Index	$1F84_h$
Name	Device type identification
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 35 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1000_h – see */CiA301/*
Default value	$0000\ 0000_h$
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1000_h – see */CiA301/*
Default value	$0000\ 0000_h$

6.7 Object $1F85_h$ – Vendor identification

This object shall be used for verification of the vendor-ID of the CANopen devices in the network.

The vendor-ID (object 1018_h sub-index 01_h – *see /CiA301/*) of the CANopen device in the network shall be matched against the value of this object in case the value is unequal 0. An error event shall be generated if the values mismatch. In case the value of this object is 0 the vendor-ID of the CANopen device in the network may not verified. Table 36 and Table 37 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

Table 36 — Object description

Attribute	Value
Index	$1F85_h$
Name	Vendor identification
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 37 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 01_h – see */CiA301/*
Default value	$0000\ 0000_h$
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 01_h – see */CiA301/*
Default value	$0000\ 0000_h$

6.8 Object $1F86_h$ – Product code

This object shall be used for verification of the product code of the CANopen devices in the network.

The product code (object 1018_h sub-index 02_h – *see /CiA301/*) of the CANopen device in the network shall be matched against the value of this object in case the value is unequal 0. An error event shall be generated if the values mismatch. In case the value of this object is 0 the

product code of the CANopen device in the network may not verified. Table 38 and Table 39 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

Table 38 — Object description

Attribute	Value
Index	$1F86_h$
Name	Product code
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 39 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 02_h – see */CiA301/*
Default value	$0000\ 0000_h$
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 02_h – see */CiA301/*
Default value	$0000\ 0000_h$

6.9 Object $1F87_h$ – Revision number

This object shall be used for verification of the revision number of the CANopen devices in the network.

The revision number (object 1018_h sub-index 03_h – *see /CiA301/*) of the CANopen device in the network shall be matched against the value of this object in case the value is unequal 0. An error event shall be generated if the values mismatch. A mismatch is defined as:

- the major revision number is unequal to the expected major revision number, or
- the minor revision number is less than the expected minor revision number,

In case the value of this object is 0 the revision number of the CANopen device in the network shall not be verified. Table 40 and Table 41 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

Table 40 — Object description

Attribute	Value
Index	$1F87_h$
Name	Revision number
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 41 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 03_h – see */CiA301/*
Default value	$0000\ 0000_h$
to	

Attribute	Value
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 03_h – see */CiA301/*
Default value	$0000\ 0000_h$

6.10 Object $1F88_h$ – Serial number

This object shall be used for verification of the serial number of the CANopen devices in the network.

The serial number (object 1018_h sub-index 04_h – *see /CiA301/*) of the CANopen device in the network shall be matched against the value of this object in case the value is unequal 0. An error event shall be generated if the values mismatch. In case the value of this object is 0 the serial number of the CANopen device in the network may not verified. Table 42 and Table 43 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

Table 42 — Object description

Attribute	Value
Index	$1F88_h$
Name	Serial number
Object code	ARRAY
Data type	UNSIGNED32
Category	Optional

Table 43 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$

Attribute	Value
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 04_h – see */CiA301/*
Default value	$0000\ 0000_h$
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see value definition of object 1018_h sub-index 04_h – see */CiA301/*
Default value	$0000\ 0000_h$

6.11 Object $1F89_h$ – Boot time

The object defines the time out between start of the process *Start process boot NMT slave* and signaling of successful boot of all mandatory NMT slaves. The details are defined in 3.3.

The value shall be given in multiples of ms. The value 0 shall disable the timer.

Table 44 — Object description

Attribute	Value
Index	$1F89_h$
Name	Boot time
Object code	VAR
Data type	UNSIGNED32
Category	Optional

Table 45 — Entry description

Attribute	Value
Sub-index	00_h
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED32
Default value	0

6.12 Object $1F8A_h$ – Restore configuration

This object is used to define the allowed restore procedure for a CANopen device during startup.

The restore procedure configured in this object shall be used for restore during startup for the according CANopen device in the network (object 1011_h – *see /CiA301/*). Table 46 and Table 47 define the object description and the entry description.

The sub-index shall correspond to the node-ID of the CANopen devices in the network. The sub-index corresponding to its own node-ID shall be ignored.

The entry value (see Table 47) shall determine the sub-index of the object 1011_h of the corresponding CANopen device that is used to initiate the restore operation. If the entry value is 0, then no restore shall be sent to the CANopen device.

Table 46 — Object description

Attribute	Value
Index	$1F8A_h$
Name	Restore configuration
Object code	ARRAY
Data type	UNSIGNED8
Category	Optional

Table 47 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	01_h to $7F_h$
Default value	$7F_h$
Sub-index	01_h
Description	Node-ID 1
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	00_h to $7F_h$
Default value	manufacturer-specific
to	
Sub-index	$7F_h$
Description	Node-ID 127
Entry category	Conditional; Mandatory, according to the *SupportedNodeID* condition (see /CiA302-1/)
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	00_h to $7F_h$
Default value	manufacturer-specific

6.13 Object 1F90$_h$ – NMT flying master timing parameters

This object defines different parameters required to configure the behavior of the CANopen device capable of the NMT master mode and capable of participating the NMT flying master process. Table 49 and Table 50 define the object description and entry description. Table 48 defines the NMT master priority level.

The NMT master timeout, NMT master negotiation time delay, priority time slot, node time slot and NMT master detect cycle time shall be given in multiples of ms.

The following formula shall apply for the priority time slot:

$$\textit{Priority time slot} > 127 \cdot \textit{CANopen device time slot}$$

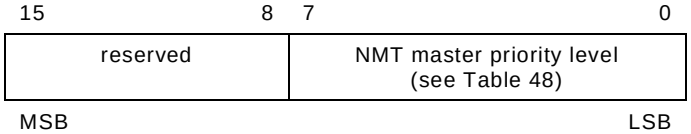

Figure 22 — NMT master priority

Table 48 — NMT master priority level

Value	Description
0000_h	Priority high
0001_h	Priority medium
0002_h	Priority low
0003_h	reserved
:::::	:::::
$00FF_h$	reserved
0100	
::::	::::
$FFFF_h$	reserved

Table 49 — Object description

Attribute	Value
Index	$1F90_h$
Name	NMT flying master timing parameters
Object code	ARRAY
Data type	UNSIGNED16
Category	Optional; Mandatory for CANopen devices supporting NMT flying master

Table 50 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	06_h
Default value	06_h
Sub-index	01_h
Description	NMT master timeout
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	100

Attribute	Value
Sub-index	02_h
Description	NMT master negotiation time delay
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	500
Sub-index	03_h
Description	NMT master priority
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	see Figure 22
Default value	Manufacturer-specific
Sub-index	04_h
Description	Priority time slot
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	1500
Sub-index	05_h
Description	CANopen device time slot
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	10
Sub-index	06_h
Description	Multiple NMT master detect cycle time
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	4000 + 10 • Node-ID

6.14 Object $1F91_h$ – Self starting nodes timing parameters

This object defines parameters that shall be configured to apply a determined behavior for self starting CANopen devices. Table 51 and Table 52 define the object description and entry description.

The NMT master detection timeout, NMT master request delay time and node time slot shall be given in multiples of ms.

Table 51 — Object description

Attribute	Value
Index	$1F91_h$
Name	Self starting nodes timing parameters
Object code	ARRAY
Data type	UNSIGNED16
Category	Optional; Mandatory for CANopen devices supporting self starting

Table 52 — Entry description

Attribute	Value
Sub-index	00_h
Description	Highest sub-index supported
Entry category	Mandatory
Access	const
PDO mapping	No
Value range	03_h
Default value	03_h
Sub-index	01_h
Description	NMT master detection timeout
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	100
Sub-index	02_h
Description	NMT master request delay time
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	500

Attribute	Value
Sub-index	03_h
Description	Node time slot
Entry category	Mandatory
Access	rw; wo, if rw is not supported
PDO mapping	No
Value range	UNSIGNED16
Default value	15

Annex A — Informative

A.1 General

An application that implements the NMT master mode may considers compatibility between CANopen version 3 and CANopen version 4. That means that some objects and service were not available in CANopen version 3. Nodes that are implemented according to CANopen version 3 maybe used within the very same network.

An application that implemented the NMT master mode may consider performing some of the process in parallel instead of process by process to speed-up the startup of the network.

A.2 Object dictionary layout

Table A.1 — Network management objects

Index	Object code	Name	Data type	Access	M/O/C
$102A_h$	VAR	NMT inhibit time	UNSIGNED16	rw	M
$1F80_h$	VAR	NMT startup	UNSIGNED32	rw	O
$1F81_h$	ARRAY	NMT slave assignment	UNSIGNED32	rw	O
$1F82_h$	ARRAY	Request NMT	UNSIGNED8	-	O
$1F83_h$	ARRAY	Request guarding	UNSIGNED8	-	O
$1F84_h$	ARRAY	Device type identification	UNSIGNED32	rw	O
$1F85_h$	ARRAY	Vendor identification	UNSIGNED32	rw	O
$1F86_h$	ARRAY	Product code	UNSIGNED32	rw	O
$1F87_h$	ARRAY	Revision number	UNSIGNED32	rw	O
$1F88_h$	ARRAY	Serial number	UNSIGNED32	rw	O
$1F89_h$	VAR	Boot time	UNSIGNED32	rw	O
$1F8A_h$	ARRAY	Restore configuration	UNSIGNED8	rw	O
$1F90_h$	ARRAY	NMT flying master timing parameters	UNSIGNED16	rw	C
$1F91_h$	ARRAY	Self starting CANopen device timing parameters	UNSIGNED16	rw	C

