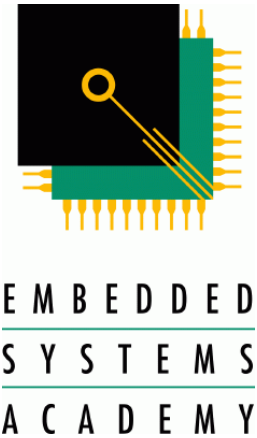
CANopenIA

Serial Remote Access to CANopen

for firmware version 1.0 or higher



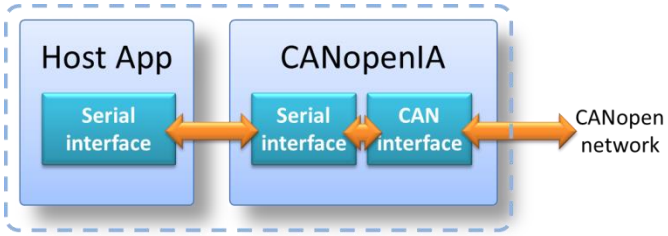
Published by
Embedded Systems Academy GmbH
Bahnhofstraße 17
D-30890 Barsinghausen, Germany
www.esacademy.com

Product based on CANgine by
Embedded Systems Solutions GmbH
Industriestraße 15
D-76829 Landau, Germany
www.essolutions.de

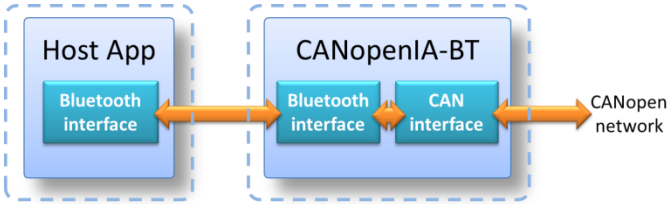
1 Contents

2 Overview

The CANopen coprocessor (447izer if in CiA447 mode) implements a CANopen device. A host system can communicate with the CANopenIA coprocessor via a regular serial channel. The protocol used is ESAcademy's CANopen remote access protocol described in this document. The CANopenIA coprocessor handles all CANopen communication.

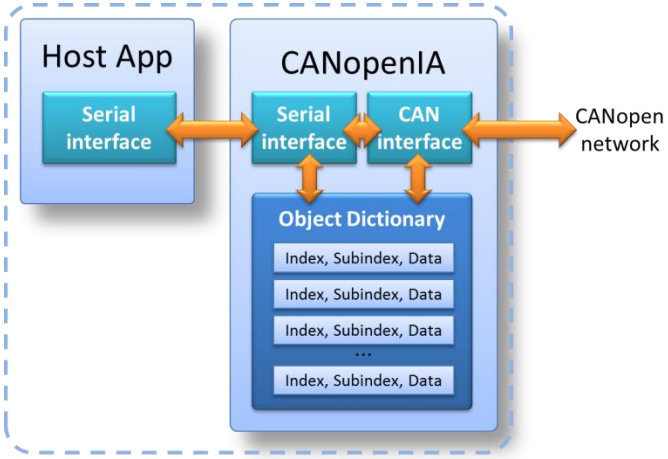


The serial interface can either be directly connected (wired), or it can be a wireless connection, for example using a serial Bluetooth connection.



As required by any CANopen device, the CANopenIA/447izer implements a CANopen Object Dictionary (OD) that contains all configurations of the chip itself as well as all the process data communicated. This OD is available to the CANopen network as well as to the host. Which OD entries are present in the CANopenIA depends on its configuration. The configuration file can be generated using the CANopen Architect EDS Editor and transferred into the flash memory of the CANopenIA.

Once the CANopenIA device is up and running (CANopen state operational), it may also send CANopen SDO (Service Data Object) read and write requests to the nodes connected to the CANopen network. This gives the host application read and write access to all the Object Dictionaries of all connected nodes.



Note that in regular CANopen this means that this device uses the regular SDO client channels used by a CANopen Manager. DO NOT use this mode, when a CANopen Manager is present and using these channels at the same time.

The Software also implements a generic low-level access mode. If this mode is activated, then any CAN message can be transmitted by the host and CAN messages received are reported back to the host. An optional CAN message ID filter allows selecting the CAN messages that should be received.

3 Hardware options

3.1 CANgine Light

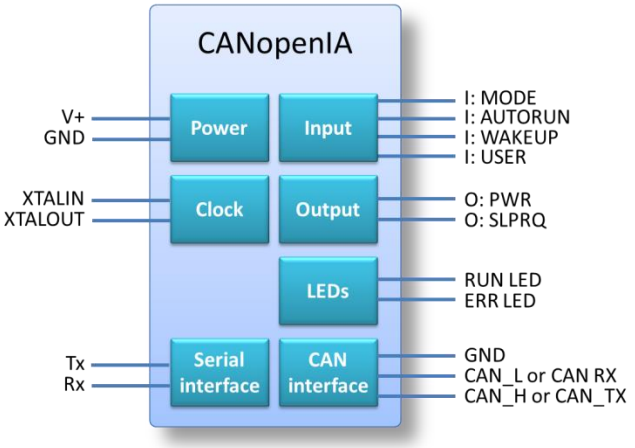
The CANgine Light by Embedded Systems Solutions GmbH is a small device with a CAN connector on one side and a RS232 (DB9) connector on the other. Power is supplied via the CAN connector. The RS232 side can be directly connected to most USB-RS232 converters.

3.2 CANgineII BT

The CANgineII BT by Embedded Systems Solutions GmbH is a small device with a CAN connector and an internal Bluetooth module. Power is supplied via the CAN connector. On the connecting Bluetooth device, the CANgineII BT appears like a generic serial device.

3.3 Module or Chip

The CANopenIA Coprocessor is available as a module or chip for direct integration into your hardware. The number of pins used is minimal, the input, output and LEDs are optional.

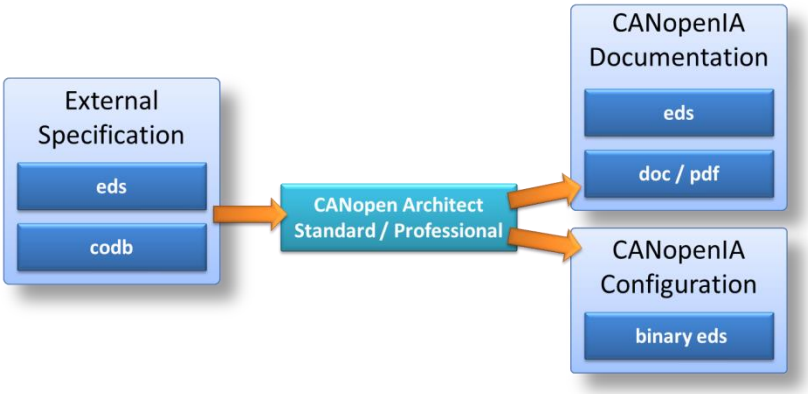


CANopenIA coprocessor implementations are available for the following microcontrollers:

- NXP LPC11C24 (uses internal transceiver)
- St-Microelectronics STM32F091 or STM32F042 (requires external transceiver)

4 System configuration

The CANopenIA implementation is configured by the local Object Dictionary. This is stored in Flash memory and can be re-loaded. The format used is ESAcademy's binary EDS file format, which is exported by the CANopen Architect EDS utility for Editing CANopen Electronic Data Sheets (EDS).



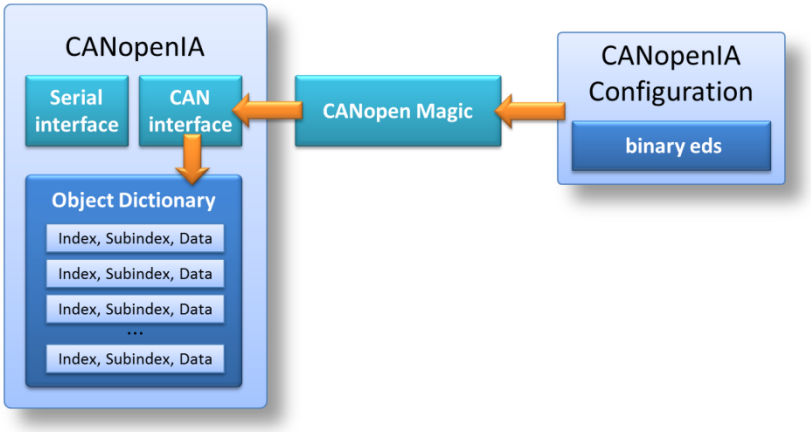
If an EDS or codb (CiA format for Object Dictionary definition) already exists specifying the configuration, then this can be imported into the CANopen Architect software. After editing/modifying the configuration, a current eds and binary eds are exported. The binary eds is directly loaded into the CANopenIA device, module or chip.

4.1 Bitrate and node ID selection

The bitrate and node ID settings are also made through the binary EDS. The host system cannot change these settings. If the configured node ID is zero, then LSS (Layer Setting Services) are used to get a node ID assigned by the LSS Master.

4.2 Loading a binary EDS

The Object Dictionary entries of the CANopenIA device are accessible with a CANopen Manager or configuration tool like CANopen Magic. The binary EDS configuration is stored at entry [1F50h,2]. A version dependent activation sequence is required to set the CANopenIA implementation in the mode required to accept a new configuration.



[Details on version depending sequence to be added later]

4.3 Step-by-step configuration example

[Version specific details to be added later]

5 The Remote Access Protocol

This chapter specifies the commands for controlling the CANopenIA Coprocessor via a serial interface. The protocol is suitable for tunneling through other networks such as a Bluetooth or TCP connection as well as for communication between a CANopen task and a host task within one system.

The communication between the host and the CANopenIA is based on messages with binary content and a check sum.

5.1 Definitions

Byte or UNSIGNED8:
8-bit, unsigned value

UNSIGNED16:
16-bit, unsigned value

UNSIGNED32:
32-bit, unsigned value

Host:
The processor or application controlling the CANopen CANopenIA via the interface specified in this document

Command:
Message from host to CANopenIA with a request to execute a command.

Response:
Message from CANopenIA to host in response to a command. Every command triggers a response. Some responses may take longer as CANopen communication might be involved. As a result one or multiple Indications might occur before receiving a response.

Indication:
Message from CANopenIA to host indicating the host that an event occurred.

Max data size:
In this version, the maximum user data size is 28 bytes. Including overhead, this results in a maximum serial packet size of 35 bytes.

Message Definition
Any message exchanged between Host and the CANopen node use the following structure (all Bytes):

<start character><length><command/response/indication><checksum>

Multi-Byte values are transmitted in little-endian format.

<start character> (Byte) default: 11h

1. Bits 0 to 3 indicate the network number, the value of zero is reserved, the default is one.
2. Bit 4 indicates if a checksum is used or not. If set, checksum is used, the default is one, using a checksum.
3. Bit 5 indicates if the length value has 8 or 16 bit. If set, 16 bits are used, the default is zero, using 8 bits for the length value.
4. Bits 6 to 7 are reserved.

<length> (Byte or UNSIGNED16, see Bit 5 of start character)

The total length of the command/response/indication in bytes.

<command/response/indication>

The data transferred in this packet can be a command, a response or an indication. For details see specifications below.

<checksum> (UNSIGNED16 or not used, see Bit 4 of start character)

A 16-bit CRC calculated with the Polynomial x16 + x15 + x2 + 1. The checksum calculation does not involve the start character.

5.2 Error Codes

Most of the responses contain an error code field. A value of zero means "no error". The bits in the error code field have the following meanings:

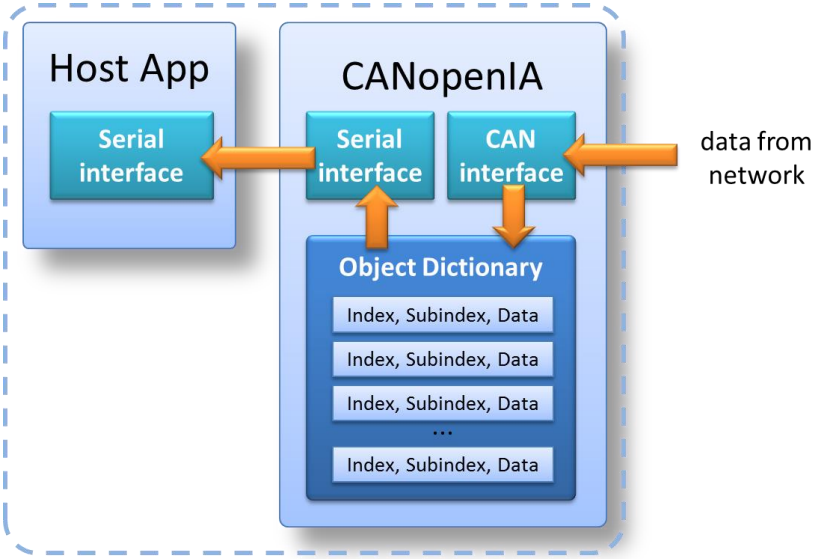
Bit	Meaning
0	Object Dictionary entry not found
1	Invalid command length
2	Invalid command
3	Busy (e.g. SDO client is currently in use)
4	No resources (e.g. internal problem obtaining an SDO
5	Transmit buffer is full
6	Transfer was aborted
7	Receive buffer size was too small
8	SDO toggle error
9	SDO timeout
10	Unknown/miscellaneous error
11	Not supported
12	Non-volatile memory write failure

6 Commands, Responses and Indications

6.1 Access to local Object Dictionary

The commands, responses and indications of this section are used to access the local object dictionary of the CANopenIA Coprocessor.

Indication "D": New process data written to local Object Dictionary



Automatic indication of new data written to Object Dictionary

New process data arrived from the CANopen network and was written to a local Object Dictionary entry. The node ID of the sender, the Object Dictionary entry in question and the new data is part of this indication. Data size is indicated via length field of lower communication layer.

Per default, in CiA 447 mode ALL CiA 447 defined PDOs are received and cause an indication. The host software can write to the

Syntax:

D<nodeid><index><subindex><data>

<nodeid> (UNSIGNED8):

The node ID of the device sending this data.

<index> (UNSIGNED16):

The index of the Object Dictionary entry.

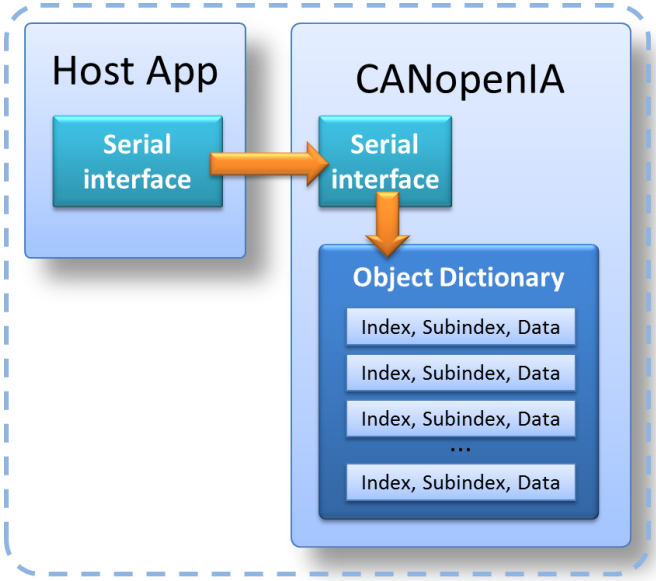
<subindex> (UNSIGNED8):

The subindex of the Object Dictionary entry.

<data> (UNSIGNED8):

The data of the Object Dictionary entry.

Command "W": Write to a local Object Dictionary entry



Read or write command to local Object Dictionary

Writes data to one Object Dictionary entry. Data size is indicated via length field of lower communication layer (see message definition).

Syntax:

W<index><subindex><data>

<index> (UNSIGNED16):

The index of the Object Dictionary entry.

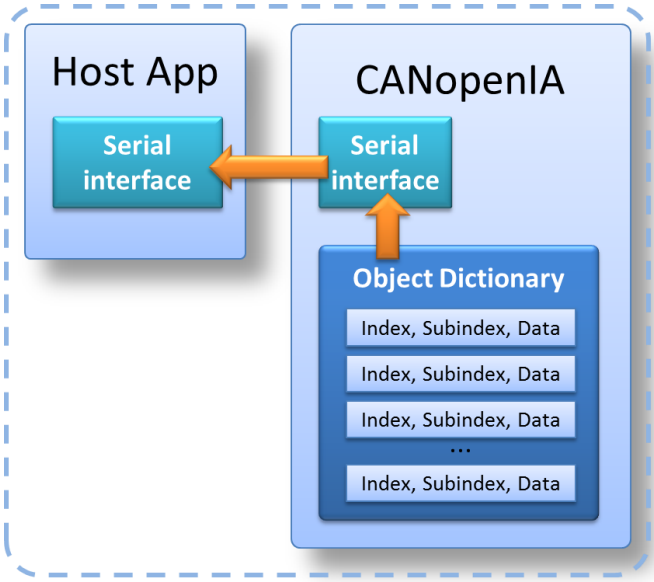
<subindex> (UNSIGNED8):

The subindex of the Object Dictionary entry.

<data> (one or multiple UNSIGNED8):

The data to be written to the Object Dictionary entry.

Response "W": Write (local) response



Read or write response from local Object Dictionary

The following message is a response from the CANopen device to every “W” message processed.

Syntax:

W<index><subindex><err>

<index> (UNSIGNED16):

The index of the Object Dictionary entry.

<subindex> (UNSIGNED8):

The subindex of the Object Dictionary entry.

<err> (UNSIGNED16):

Error code or zero for no error.

Command "R": Read from a local Object Dictionary entry

Request to read data from one Object Dictionary entry. Data size is indicated via length field of lower communication layer.

Syntax:

R<index><subindex>

<index> (UNSIGNED16):

The index of the Object Dictionary entry.

<subindex> (UNSIGNED8):

The subindex of the Object Dictionary entry.

Response "R": Read (local) response

The following message is a response from the CANopen device to every “R” message processed. Data size is indicated via length field of lower communication layer (see message definition).

Syntax:

R<index><subindex><err><data>

<index> (UNSIGNED16):

The index of the Object Dictionary entry.

<subindex> (UNSIGNED8):

The subindex of the Object Dictionary entry.

<err> (UNSIGNED16):

Error code or zero for no error.

<data> (one or multiple UNSIGNED8):

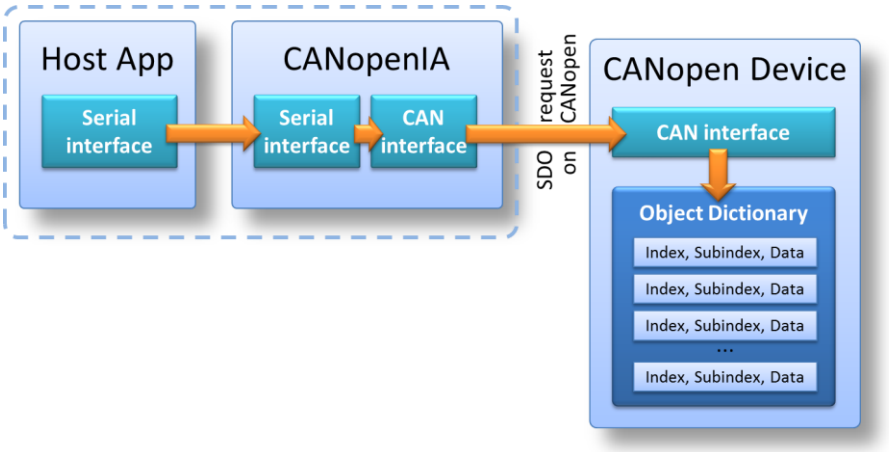
The data read from the Object Dictionary entry.

6.2 Access to other nodes

The commands, responses and indications of this section are used to access object dictionary entries of any node on the network. In CANopen terminology these use SDO clients to communicate with the nodes addressed.

These commands require SDO clients which are only availble when the Manager or CiA 447 functionality is enabled.

Command "S": Write to a remote Object Dictionary entry



Read or write request to a remote Object Dictionary of a node on the network

Writes data to one Object Dictionary entry of a remote node (using SDO client access). Data size is indicated via length field of lower communication layer.

Syntax:

S<nodeid><index><subindex><data>

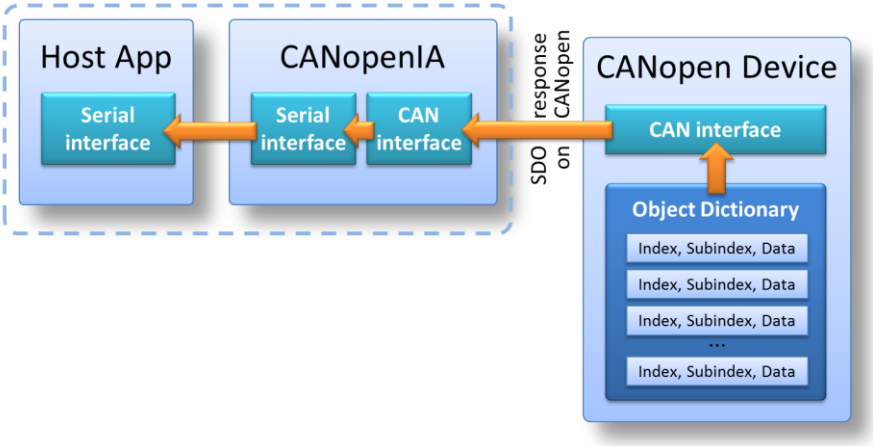
<nodeid> (UNSIGNED8):

The ID of the node to write to.

All other parameters are the same as with the write command.

Note: only one remote SDO operation can take place at a time. This applies to read and writes. An attempt to start a new SDO operation while one is still completing will generate an error.

Response "S": Write (remote) response



Read or write response from a remote Object Dictionary of a node on the network

The following message is a response from the CANopen device to every “S” message processed.

Syntax:

S<sdo><index><subindex><err>

<sdo> (UNSIGNED8):

The SDO channel number used. In CiA447 the channel number equals the node ID of the remote node addressed.

<err> (UNSIGNED16):

Error code or zero for no error.

All other parameters are the same as with the local write response.

Command "U": Read from a remote Object Dictionary entry

Request to read data from a remote Object Dictionary entry (using SDO client access, upload).

Syntax:

U<nodeid><index><subindex>

<nodeid> (UNSIGNED8):

The ID of the node to read from.

All other parameters are the same as with the write command.

Note: only one remote SDO operation can take place at a time. This applies to read and writes. An attempt to start a new SDO operation while one is still completing will generate an error.

Response "U": Read (remote) response

The following message is a response from the CANopen device to every “U” message processed. Data size is indicated via length field of lower communication layer.

Syntax:

U<sdo><index><subindex><err><data>

<sdo> (UNSIGNED8):

The SDO channel number used. In CiA447 the channel number equals the node ID of the remote node addressed.

<err> (UNSIGNED16):

Error code or zero for no error.

All other parameters are the same as with the local write response.

7 Object Dictionary entries in the manufacturer specific area

The manufacturer specific area of the Object Dictionary provides direct access to configuration data. These can be accessed using the read and write local commands, indications are sent automatically for writes to 5F00h and 5F04h.

Syntax used in listing below:

[index,subindex] (data type, access type) name

7.1 Device status

[5F00h,01h] (UNSIGNED8,RO) Device status: own node ID

[5F00h,02h] (UNSIGNED8,RO) Device status: own NMT state

[5F00h,03h] (UNSIGNED8,RO) Device status: own HW state

Bit: 0: INIT - set to 1 after a completed initialization

1: CERR - set to 1 if a CAN bit or frame error occurred

2: ERPA - set to 1 if a CAN "error passive" occurred

3: RXOR - set to 1 if a receive queue overrun occurred

4: TXOR - set to 1 if a transmit queue overrun occurred

5: Reserved

6: TXBSY - set to 1 if Transmit queue is not empty

7: BOFF - set to 1 if a CAN "bus off" error occurred

7.2 Device Control

[5F01h,01h] (UNSIGNED8,WO) Device control: reset (set to 1 to force reset)

[5F01h,02h] (UNSIGNED8,WO) Device control: sleep objection (set to 1 to objct)

[5F01h,03h] (UNSIGNED32,WO) Device control: Ignore PDOs from VD.

If a bit is set in this value, then PDOs coming from the corresponding virtual device (see vdfg number in CiA-447) are ignored. For example: set bit 7 to ignore all PDOs coming from GPS devices.

7.3 Status of all nodes

[5F04h,01h] (UNSIGNED8,RO) Last known state of Node 1

[5F04h,02h] (UNSIGNED8,RO) Last known state of Node 2

…

The following values are defined:

NODESTATUS_BOOT	0x00
NODESTATUS_STOPPED	0x04
NODESTATUS_OPERATIONAL	0x05
NODESTATUS_PREOP	0x7F
NODESTATUS_EMCY_NEW	0x80
NODESTATUS_EMCY_OVER	0x81
NODESTATUS_HBACTIVE	0x90
NODESTATUS_HBLOST	0x91
NODESTATUS_SCANCOMPLETE	0xA0
NODESTATUS_SCANABORTED	0xA8
NODESTATUS_RESETAPP	0xB0
NODESTATUS_RESETCOM	0xB1
NODESTATUS_SLEEP	0xF0

7.4 Automatic Node Scan

In CiA 447 mode, the device automatically scans nodes found on the network for their entries Device Type, Vendor ID and Virtula Device Info. This data is available, as soon as a node's state is reported as NODESTATUS_SCANCOMPLETE.

[5F08h,00h] (UNSIGNED16,RO) One bit for each of the 16 nodes.

Corresponding bit is set, if the node has been scanned.

[5F11h,01h] (UNSIGNED32,RO) Scanned data: [1000h,00h] of Node 1

[5F11h,02h] (UNSIGNED32,RO) Scanned data: [1018h,01h] of Node 1

[5F11h,03h] (UNSIGNED32,RO) Scanned data: [6000h,01h] of Node 1

[5F12h,01h] (UNSIGNED32,RO) Scanned data: [1000h,00h] of Node 2

[5F12h,02h] (UNSIGNED32,RO) Scanned data: [1018h,01h] of Node 2

[5F12h,03h] (UNSIGNED32,RO) Scanned data: [6000h,01h] of Node 2

…

[5F20h,01h] (UNSIGNED32,RO) Scanned data: [1000h,00h] of Node 16

[5F20h,02h] (UNSIGNED32,RO) Scanned data: [1018h,01h] of Node 16

[5F20h,03h] (UNSIGNED32,RO) Scanned data: [6000h,01h] of Node 16