CANopen User Guide

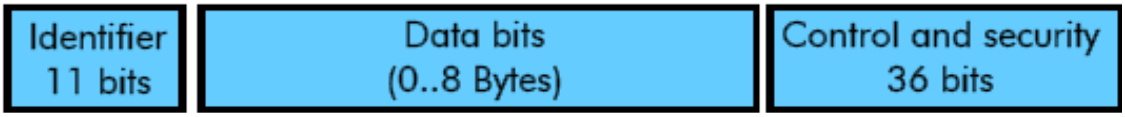
MAN0900-01

Table of Contents

Chapter 1: Introduction

CANopen is a communication protocol and device profile specification for embedded systems used in automation. In terms of the OSI model, CANopen implements the layers above and including the network layer. The CANopen standard consists of an addressing scheme, several small communication protocols and an application layer defined by a device profile. The communication protocols have support for network management, device monitoring and communication between nodes, including a simple transport layer for message segmentation/desegmentation. The lower level protocol implementing the data link and physical layers is usually Controller Area Network (CAN), although devices using some other means of communication (such as EtherCAT) can also implement the CANopen device profile.

CANopen is a CAN-based higher layer protocol. It is developed as a standardized embedded network with highly flexible configuration capabilities. CANopen is designed for motion-oriented machine control networks, such as handling systems. It is used in various fields, such as medical equipment, off-road vehicles, maritime electronics, public transportation, building automation, etc



The CANopen communication profiles are based on the CAN Application Layer (CAL) protocol. Version 4 of CANopen (CiA DS 301) is standardized as EN 50325-4.The CANopen specification cover application layer and communication profile (CiA DS 301), as well as a framework for programmable devices (CiA 302), recommendations for cables and connectors (CiA 303-1) and SI units and prefix representations (CiA 303-2). The application-layer as well as the CAN-based profiles is implemented in software.

1.1 Elements

CANopen Services, Protocols and Communication objects are as follows

- Device Model and Object Dictionary
- Process Data Objects (PDO) → Explained in ***Section 4.2.5***
- Service Data Objects (SDO) → Explained in ***Section 4.2.4***
- Network Management Objects (NMT)
- Special functions objects (Sync, Emcy, Time) → Explained in ***Section 4.2.2***
- Error control → Explained in ***Section 4.2.3***

CANopen unburdens the developer from dealing with CAN-specific details such as bit-timing and implementation-specific functions. It provides standardized communication objects for real-time data (Process Data Objects, PDO), configuration data (Service Data Objects, SDO), and special functions (Time Stamp, Sync message, and Emergency message) as well as network management data (Boot-up message, NMT message, and Error Control).

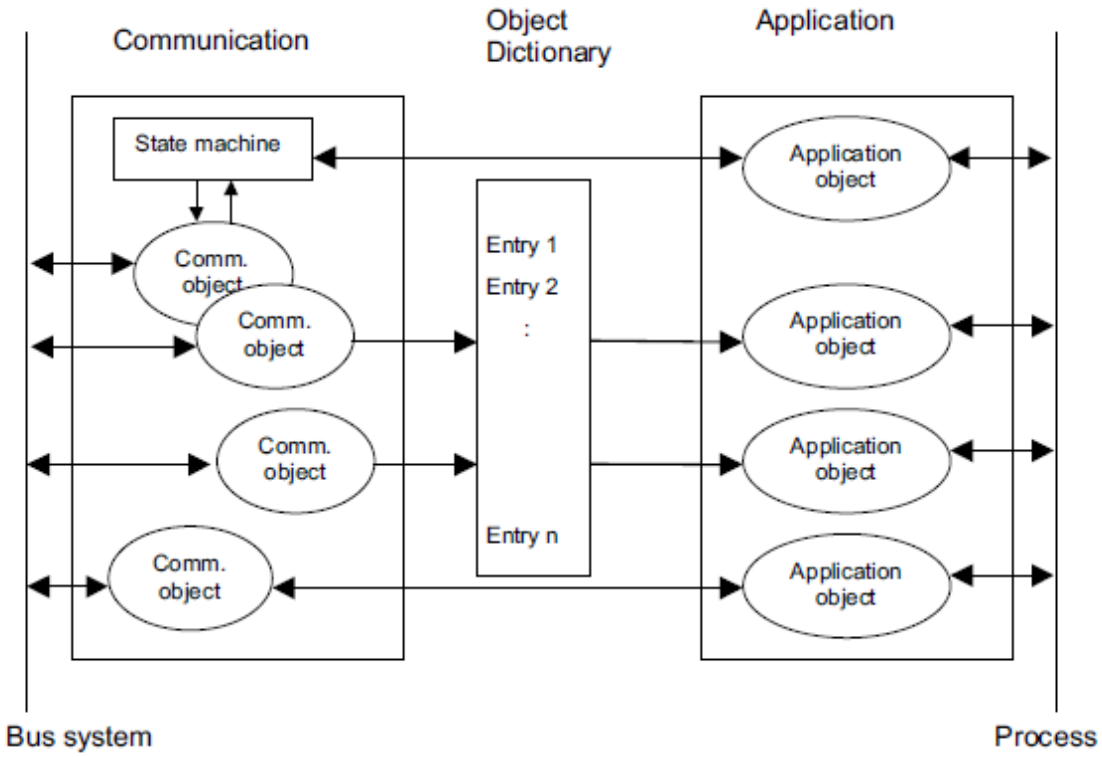
Device Model:

The Device Model is structured as shown in figure below. It consists of Communication, Object Dictionary and Application.

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

Object Dictionary: The Object Dictionary is a collection of all the data items which have an influence on the behavior of the application objects, the communication objects and the state machine used on this device.

Application: The application comprises the functionality of the device with respect to the interaction with the process environment.



Object Dictionary:

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

The General structure of the object dictionary is as follows:

Index (hex)	Object (Symbolic Name)	Name	Type	Attrib.	M/O

Index: The Index column denotes the objects position within the Object Dictionary. This acts as a kind of address to reference the desired data field. The sub-index is not specified here. The sub-index is used to reference data fields within a complex object such as an array or record.

Object: The Object column contains the Object Name and is used to denote what kind of object is at that particular index within the Object Dictionary.

Name: The name column provides a simple textual description of the function of that particular object.

Type: The type column gives information as to the type of the object. Eg: Boolean, Floating number, Unsigned Integer, Signed Integer etc.

Attribute: The Attribute column defines the access rights for a particular object. Eg: rw (read and write access), wo (write only), ro (read only), Const (read only and value is constant).

M/O: The M/O column defines whether the object is **M**andatory or **O**ptional.

The standard object dictionary structure is as shown below:

Index (hex)	Object
0000	not used
0001-001F	Static Data Types
0020-003F	Complex Data Types
0040-005F	Manufacturer Specific Complex Data Types
0060-007F	Device Profile Specific Static Data Types
0080-009F	Device Profile Specific Complex Data Types
00A0-0FFF	Reserved for further use
1000-1FFF	Communication Profile Area
2000-5FFF	Manufacturer Specific Profile Area
6000-9FFF	Standardised Device Profile Area
A000-BFFF	Standardised Interface Profile Area
C000-FFFF	Reserved for further use

Network Management (NMT)

The Network Management (NMT) is node oriented and follows a master-slave structure. NMT objects are used for executing NMT services. Through NMT services, nodes are initialized, started, monitored, resetted or stopped. All nodes are regarded as NMT slaves. An NMT Slave is uniquely identified in the network by its Node-ID, a value in the range of [1 to127].

NMT requires that one device in the network fulfils the function of the NMT Master.

NMT Services:

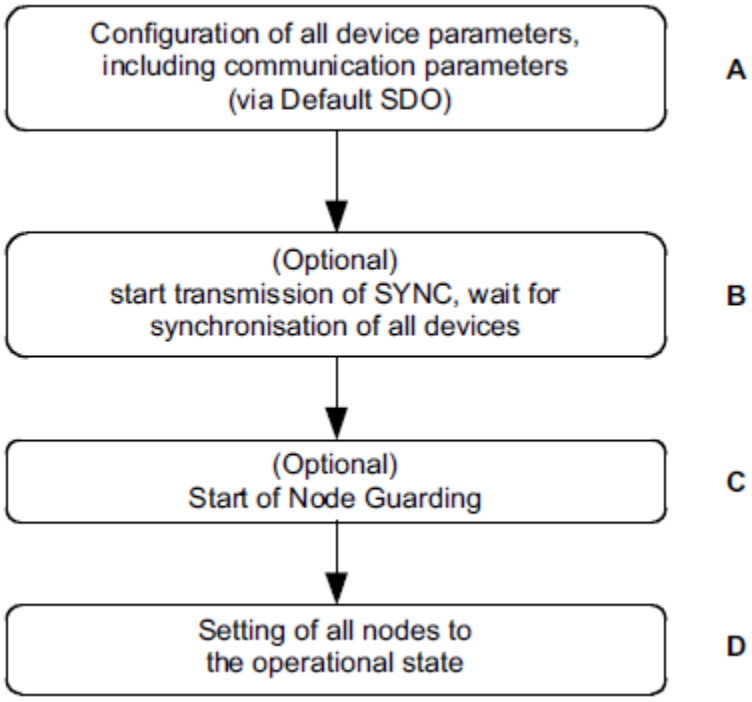
1. **Module Control Services:** Through Module Control Services, the NMT master controls the state of the NMT slaves. The state attribute is one of the values {STOPPED, PRE-OPERATIONAL, OPERATIONAL, INITIALISING}. The Module Control Services can be performed with a certain node or with all nodes simultaneously.

2. **Error Control Service:** Through Error control services the NMT detects failures in a CAN-based Network. Local errors in a node may e.g. lead to a reset or change of state. Error Control services are achieved principally through periodically transmitting of messages by a device. There exist two possibilities to perform Error Control i.e., Node Guard and Heart Beat Error Control. (Explained in ***Section 4.2.3***)

3. **Bootup Service:** Through this service, the NMT slave indicates that a local state transition occurred from the state INITIALISING to the state PRE-OPERATIONAL.

Network Initialisation Process:

The general flow chart of the network initialisation process, controlled by a NMT Master Application or Configuration Application is shown:



In step A the devices are in the node state PRE-OPERATIONAL which is entered automatically after power-on. In this state the devices are accessible via their Default-SDO using identifiers that have been assigned according to the Predefined Connection Set. In this step the configuration of device parameters takes place on all nodes which support parameter configuration.

This is done from a Configuration Application or Tool which resides on the node that is the client for the default SDOs. For devices that support these features the selection and/or configuration of PDOs, the mapping of application objects (PDO mapping), the configuration of additional SDOs and optionally the setting of COB-IDs may be performed via the Default-SDO objects.

In many cases a configuration is not even necessary as default values are defined for all application and communication parameters.
If the application requires the synchronisation of all or some nodes in the network, the appropriate mechanisms can be initiated in the optional Step B. It can be used to ensure that all nodes are synchronised by the SYNC object before entering the node state OPERATIONAL in step D. The first transmission of SYNC object starts within 1 sync cycle after entering the PRE-OPERATIONAL state.

In Step C Node guarding can be activated (if supported) using the guarding parameters configured in step A. With step D all nodes are enabled to communicate via their PDO objects.

States and Communication Object Relations:

Table below shows the relation between communication states and communication objects. Services on the listed communication objects may only be executed if the devices involved in the communication are in the appropriate communication states.

	INITIALISING	PRE-OPERATIONAL	OPERATIONAL	STOPPED
PDO			X	
SDO		X	X	
Synchronisation Object		X	X	
Time Stamp Object		X	X	
Emergency Object		X	X	
Boot-Up Object	X			
Network Management Objects		X	X	X

Default CAN message identifiers and an example for Node ID 5 for messages in CANopen network is as shown in table below:

CAN ID			**If CANopen Node ID = 5**
From	**To**	**Communication Objects**	**Message Identifier**
0h		NMT Service	0h
80h		SYNC Message	80h
81h	FFh	Emergency Messages	85h
100h		Time Stamp Messages	100h
181h	1FFh	Transmit PDO 1	185h
201h	27Fh	Receive PDO 1	205h
281h	2FFh	Transmit PDO 2	285h
301h	37Fh	Receive PDO 2	305h
381h	3FFh	Transmit PDO 3	385h
401h	47Fh	Receive PDO 3	405h
481h	5FFH	Transmit PDO 4	485h
501h	57Fh	Receive PDO 4	505h
581h	5FFh	Transmit SDO	585h
601h	67Fh	Receive SDO	605h
701h	77Fh	NMT Error Control	705h

1.2 OCS models supporting CANopen

In general XL Series, NX Series and QX Series support CANopen. Following is the present list of models supporting CANopen:

XLe Series	XLt Series	XL6 Series	NX Series	QX Series	
				BP41	BP43
XE200	XT200	XL200	NX220	QX451	QX451
XE202	XT202	XL202	NX221	QX551	QX551
XE203	XT203	XL203	NX222	QX651	QX651
XE204	XT204	XL204	NX250	QX751	QX751
XE205	XT205	XL205	NX251		
			NX252		

NOTE: Contact Tech Support for a specific model enquiry.

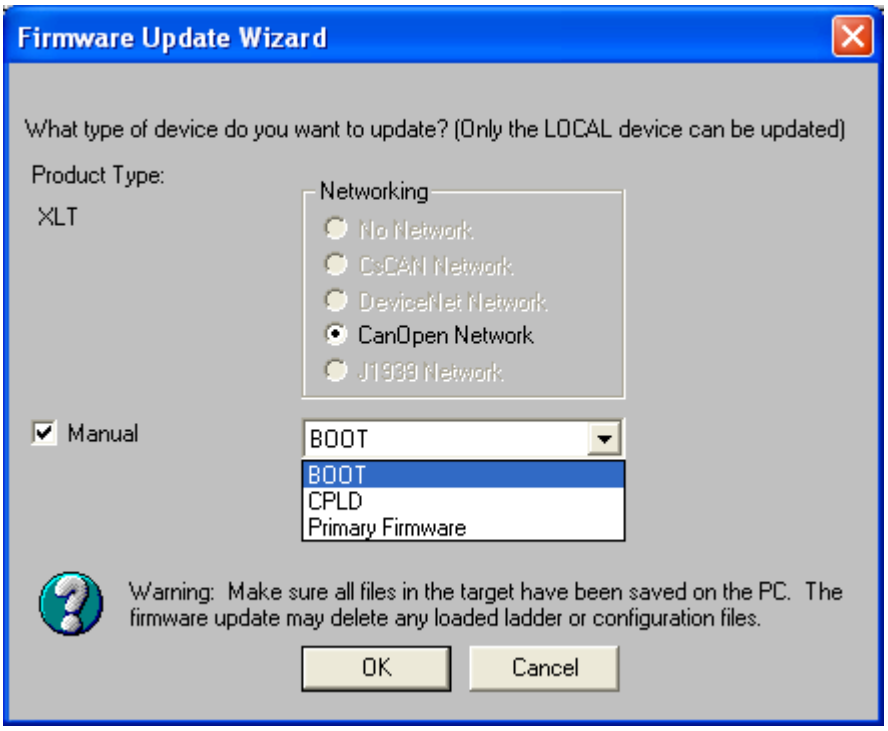
1.3 Features

	XLe/t	XL6, QX, NX
Baud Rate supported	125K, 250K, 500K & 1M bps	
COB IDs supported	11 Bits	
Modes	Master & Slave Modes	
Process Data Objects	Transmit Process Data Objects (TPDO) : supports 64 PDO's Receive Process Data Objects (RPDO) : supports 64 PDO's	
PDO Mapping	Static PDO : supports 64 PDO's Dynamic PDO : supports 16 PDO's	
PDO Modes	**Asynchronous**: On change of Data, On Event Time, On Trigger Bit and Remote requested **Synchronous**: On change of Data, On Sync count and Remote requested Inhibit time supported in case of Asynchronous PDOs	
PDO communication parameters	COB-ID, Transmission Type & Timing Parameters	
SDO Services	Expedited Upload, Expedited Download & Segmented Upload	
Emergency Message	Supported only in Generic Error	
NMT Manager	Supports up to 127 slaves	
Application Layer supported	CiA 301 v 4.02	
Profiles supported	CiA 405 v 2.00	
CsCAN Modes	Pass through mode and Remote mode **Note:** If Node is busy with boot up reconfiguration then CsCAN pass through mode will not work.	
Slave Nodes	Allows addition of 16 slave nodes	Allows addition of 126 slave nodes
Error Control Protocols	Node Guard Protocol: 16 Nodes Heart Beat Protocol : 16 Nodes	Node Guard Protocol:127 Nodes Heart Beat Protocol : 127 Nodes
Service Data objects (SDO) Modes	Server: 16 Client: 16 Supported only in Master	Server: 128 Client: 128 Supported only in Master
Sync Message Generation	N/A	Supported

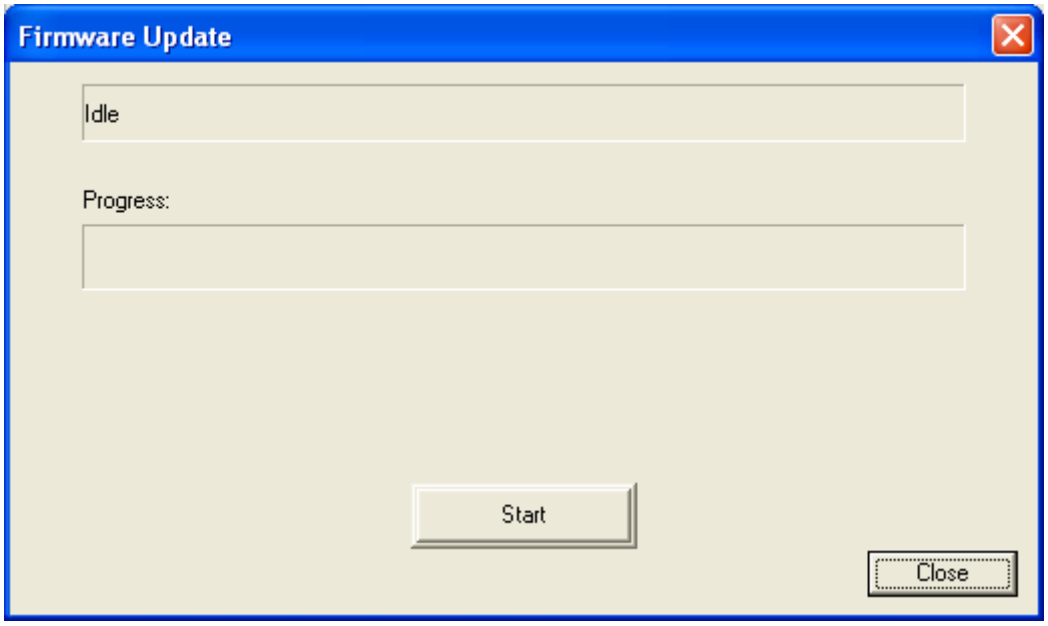
Chapter 2: Firmware update

2.1 Firmware update for CANopen Models

1. Download the appropriate firmware from www.heapg.com (For CANopen, the firmware files usually end with "2" or "c". For Eg: xlt002.s19 or XL6000c.S19).
2. Ensure that the firmware is in '\Cscape\Firmware' folder.
3. Start Cscape.
4. If the user needs to update the firmware to the already connected OCS unit then go to **step 6**.
5. If the user needs to update firmware to a different OCS unit then user needs to disconnect the serial cable from the OCS unit, then select File → Firmware Update Wizard, Select the '*Product Type*' from the dropdown. Select **CANopen Network** in the '*Networking field*' and connect the serial cable back to the OCS unit. Go to **step 7**.
6. Select File → Firmware Update Wizard, the following window pops up.



7. The following two options are available for firmware update:
   a. 'Manual' checkbox disabled (Unchecked): - This will do a full update firmware including BOOT, CPLD and Primary Firmware (according to the files present in the Firmware folder).
   b. 'Manual' checkbox enabled (Checked): - When manual checkbox is selected, the drop down shows the possible options for download. Select the option required to be downloaded and press OK button.
8. The firmware update dialog will be shown. Press **Start** to begin the update process. Wait for the dialog to indicate that update process is complete.


Firmware Update
Idle
Progress:
Start
Close

Chapter 3: CANopen Configuration

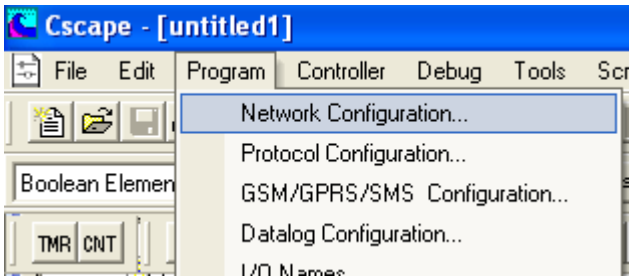
3.1 Launching CANopen Configurator

Master/Slave relationship

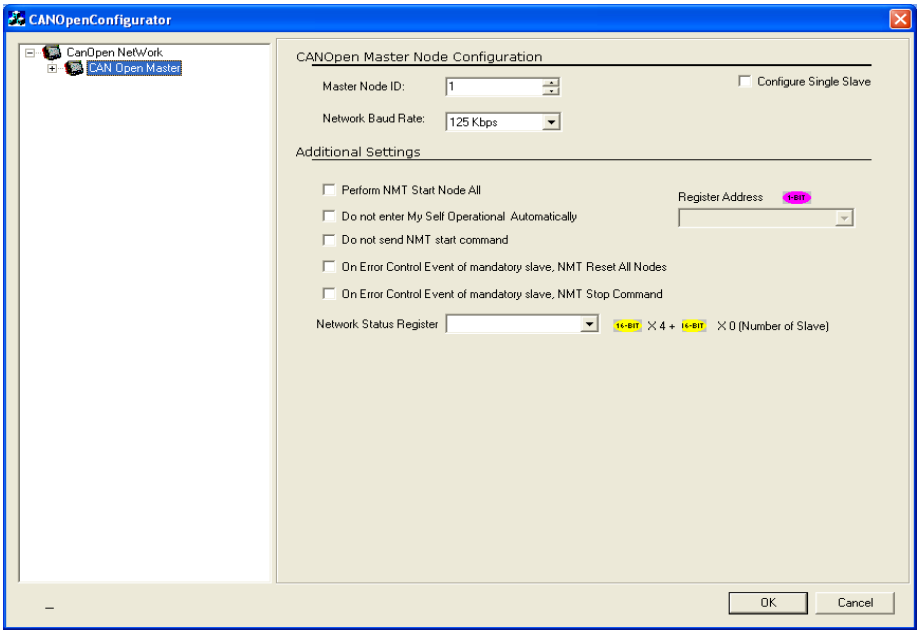
At any time there is exactly one device in the network serving as a master for a specific functionality. All other devices in the network are considered as slaves. The master issues a request and the addressed slave(s) respond(s) if the protocol requires this behavior.

Following are the steps to launch the CANopen Configurator from Cscape:

1. Select any model controller from **I/O Configuration** that supports CANopen.
2. Select Program → Network Configuration.



3. It will launch a Master Node CANopen Configurator as shown



Chapter 4: Master Configuration

4.1 Minimum and Full Configuration

The table below gives the minimum master configuration and full master configuration.

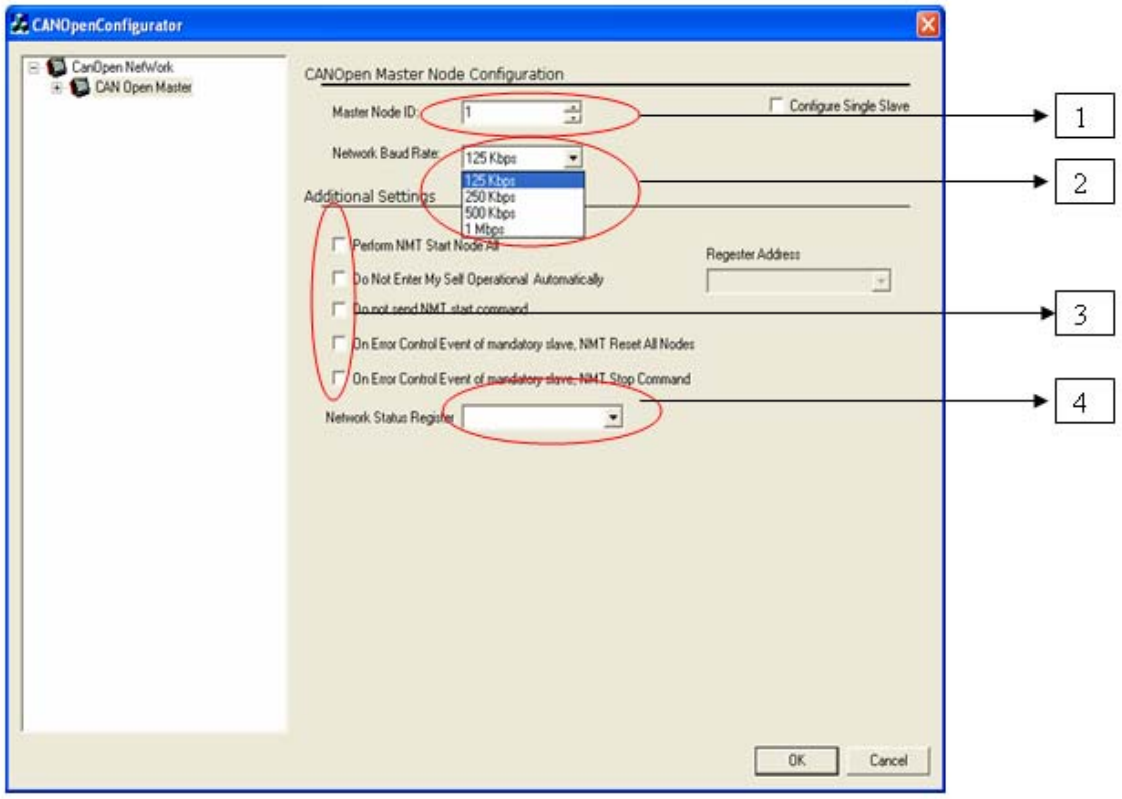
NOTE:

1. If user is connecting pre-configured slave in the Network, then Minimum configuration would be required in the Master configuration.
2. By default, without configuration download, the OCS will act as slave node

Eg: If user has 2 nodes (Assume both are Horner OCS) in the CANopen network, one is Master and other is slave. If slave is configured using single slave Configuration and connected to network, then while configuring Master node user need not configure all slave parameters for that slave, only Error control Parameter configuration is required.

Minimum Configuration (Master does not reconfigures the slave node)	**Full Configuration**
Configure '*Master Node ID*' and '*Network Baud Rate*'.	Configure '*Master Node ID*' and '*Network Baud Rate*'.
Configure various '*Additional Settings*' check boxes	Configure various '*Additional Settings*' check boxes
Select and configure '*Error Control Object*'	Configure '*Network Status Register*'
Configure '*Process Data Objects(PDO)*'	Select and configure '*Error control Object*'
Add various slave nodes which need to be controlled by Master	Configure '*Process Data Objects(PDO)*'
Configure only '*Error control protocol*' parameter of each added slave node.	Add various slave nodes which need to be controlled and configured by Master, with EDS import
	Select type of slave i.e. '*Mandatory or Non-Mandatory*' and other check boxes in the slave configuration as per the need.
	Select Check Boxes in '*Node Bootup sequence Configuration*' for each node as per the network requirement and features supported by the individual nodes
	Configure '*Error Control Object*', '*Special Function Objects*', '*Service Data Objects (SDO)*' and '*Process Data Objects (PDO)*'.

4.2 CANopen Master



1. CANopen Network ID: 1- 127.
2. CANopen Network Baud Rate
3. Additional Master Settings
4. CANopen Network status register

CANopen Network ID: Unique ID provided to each node, no two nodes can have same ID. User can provide network ID in the range of 1 - 127.

CANopen Baud Rate: User can select any of the 4 supported baud rate, for the given network each slave node has to be set to same baud as that of Master node. Different nodes at different baud rate cannot communicate with each other.

Note: Using System Menu user can modify Baud Rate and Node-ID, with the change in Node-ID default SDO-ID will change. But after power reset Node-ID and Baud rate will set to value configured using Configuration tool.

Additional Master Settings:

*1. Perform "**NMT Start Node all**":* Checking this option Master will send a single start command over the CANopen Network to start all the slaves.

*2. Do Not Enter "**My Self Operational Automatically**"*: Checking this option will ask for trigger bit to start Master node, which will inturn start all other slave nodes in the network.

*3. Do Not Send "**NMT Start Command**":* Checking this option master node does not send any start command over the network, each slave must be configured for self start.

*4. On Error Control Event Of Mandatory Slave, "**NMT Reset All Nodes**":* Checking this option master node resets CANopen communication (it self and all nodes on the network) in case of error in any of the mandatory slave.

*5. On Error Control Event Of Mandatory Slave "**NMT Stop Command**":* Checking this option master node sends stop command over the network to stop the CANopen communication in case of error in any of the mandatory slave.

NOTE: In points 1 & 3, point 3 takes higher precedence over point 1 i.e., if user selects both options point 3 will be executed. Similarly between points 4 and 5, point 4 has higher precedence over point 5.

Network Status Register: It indicates the status of CANopen Network in Master. Master and slave will have the same Network Status Register but master Node will have additional status of each Slave Node following 64bit long Status register. One 16bit register will be dedicated to indicate status of single slave node.

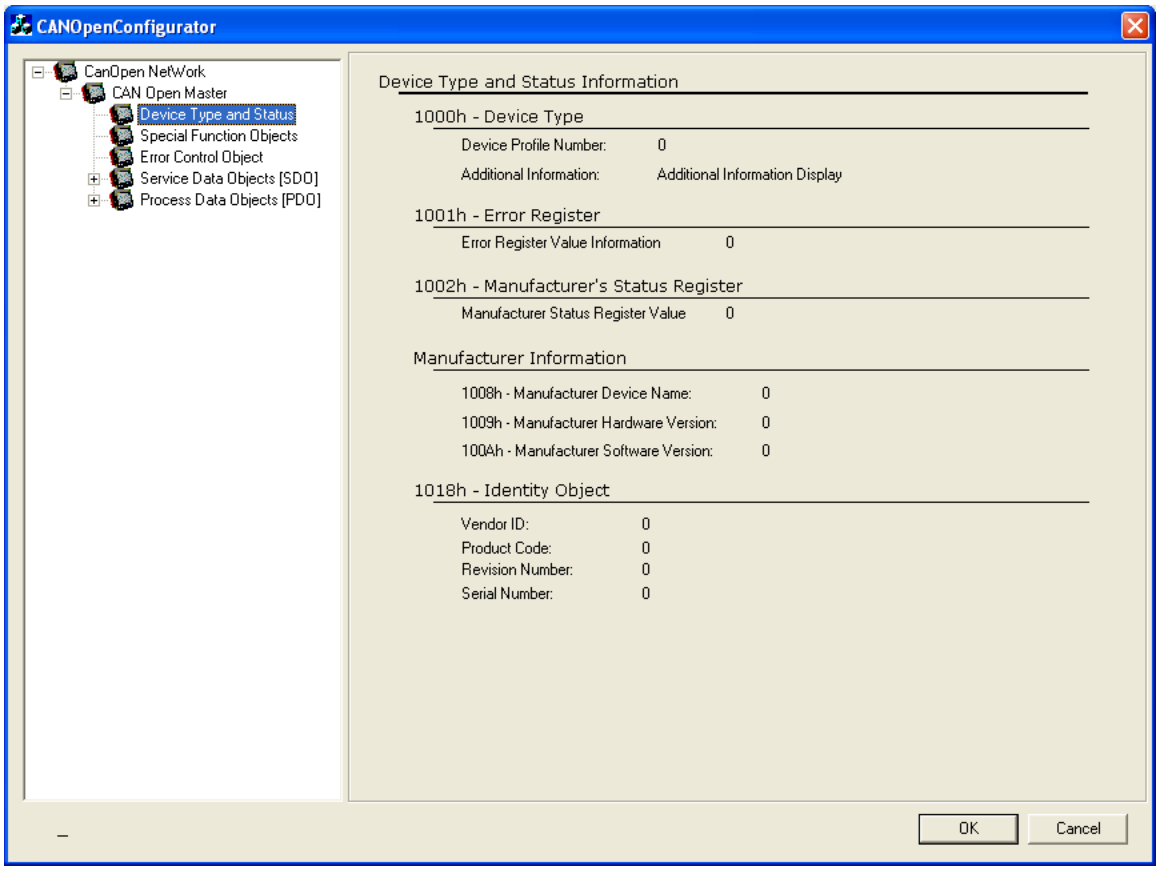
For details on Status Error Bits refer *Chapter 7*

NOTE: After Configuration download, firmware takes at least 10 sec to start CANopen communication with new configuration. Node will switch to pre-operational state during new CANopen configuration download; if node is master then it will send NMT command to all slave nodes to switch to pre-operational state.

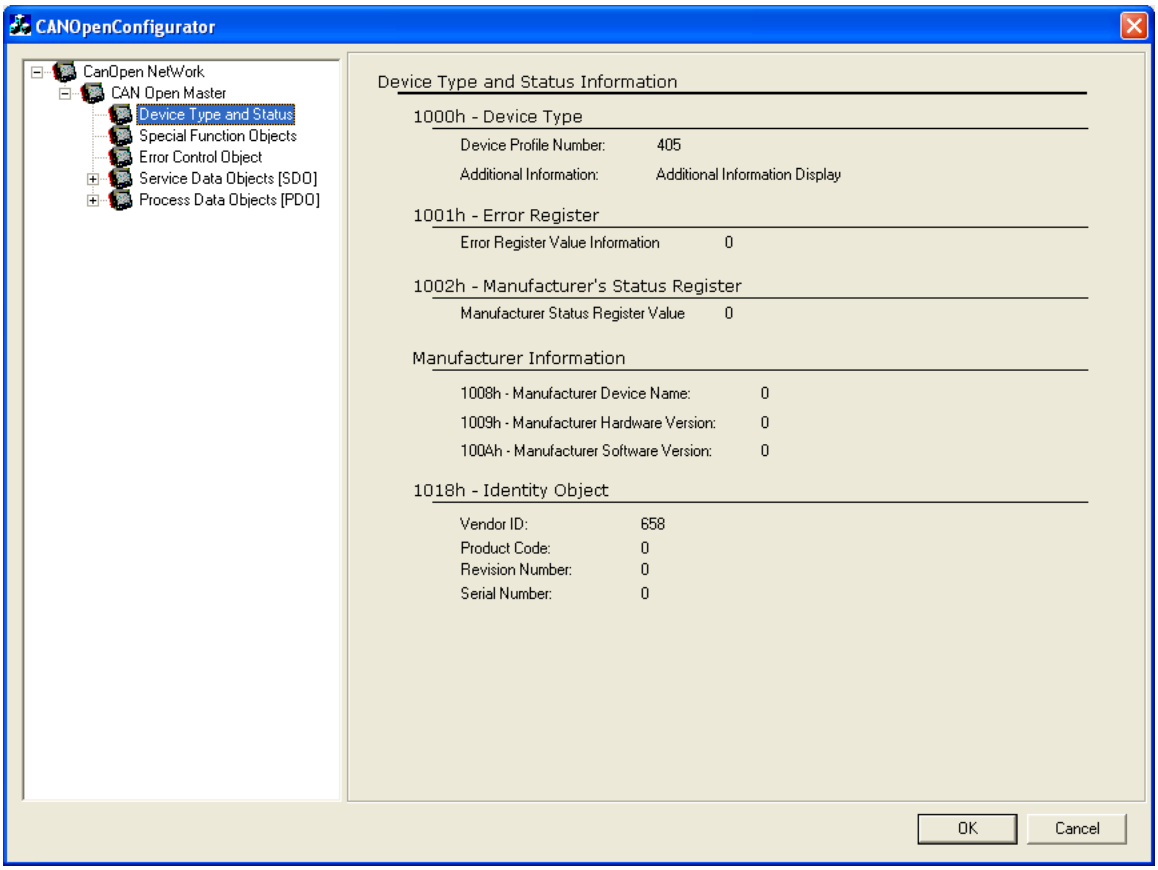
4.2.1 Device Type and Status

The figure below displays the device type information which is loaded from EDS file in the CANopen Configurator.

Before loading the EDS file all the fields are shown as "ZERO".



After the EDS file is loaded it takes the values from the EDS file and displays the values as shown below.



Device Type:

It contains information about the device type. The object at index 1000h describes the type of device and its functionality. It is composed of a 16-bit field which describes the device profile that is used and a second 16-bit field which gives additional information about optional functionality of the device. The Additional Information parameter is device profile specific.

Error Register:

This object is an error register for the device. The device can map internal errors in this byte. This entry is mandatory for all devices. It is a part of an Emergency object.

The table below gives the structure of the error register

Bit	M/O	Meaning
0	M	Generic error
1	O	Current
2	O	Voltage
3	O	Temperature
4	O	Communication error (overrun, error state)
5	O	Device profile specific
6	O	Reserved (always 0)
7	O	Manufacturer specific

Manufacturer's Status Register:

This object is a common status register for manufacturer specific purposes.

Manufacturer Information:

This object contains the manufacturer device name, manufacturer hardware version description and manufacturer software version description.

Identity Object:

This object contains general information of the device.

> **Vendor ID**: contains a unique value allocated to each manufacturer.
> **Product Code**: It is the manufacturer-specific Product code (sub-index 2h) which identifies a specific device version.
> **Revision Number**: It is the manufacturer-specific Revision number (sub-index 3h) which consists of a major revision number and a minor revision number. The major revision number identifies a specific CANopen behavior. If the CANopen functionality is expanded, the major revision has to be incremented. The minor revision number identifies different versions with the same CANopen behavior.
> **Serial Number**: It is the manufacturer-specific Serial number (sub-index 4h) which identifies a specific device.

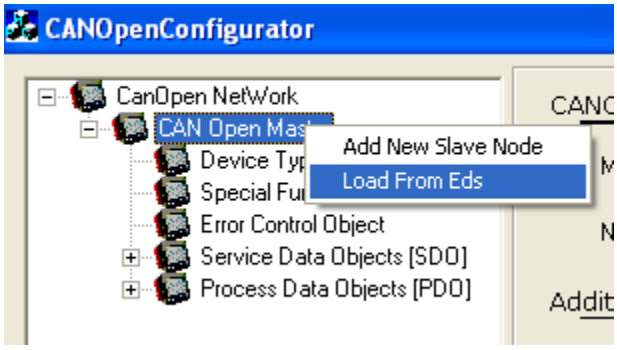
4.2.1.1 Loading EDS File

CANopen EDS (Electronic Data Sheet) file serves as template for different configurations for one device type.

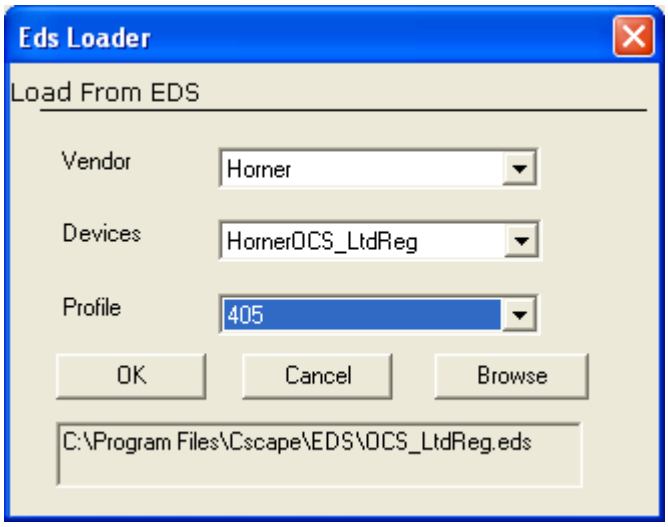
EDS file describes
- Communication functionality and objects as defined in /CiA301/ and Application Frameworks.
- Device specific objects as defined in the device profiles

User must ensure that EDS files are present in the '/Cscape/EDS' folder.

Loading EDS file: Right click on CANopen Master Node → Select Load from EDS, will open EDS Loader.



Select the Vendor, Devices and Profile from the drop down of corresponding fields. Click on OK. EDS file will be loaded.



NOTE:
Horner supplies the following EDS files with the configuration Tool.

a. OCS_FullReg.eds
b. OCS_LtdReg.eds

OCS_FullReg.eds:- This file contains full registers i.e. about 15,500 registers and it takes about 1 - 2 mins for loading EDS file. The details of the registers range are as follows:

%R - %R1 to %R9999
%M - %M1 to %M2048
%AQ - %AQ1 to %AQ512
%Q - %Q1 to %Q2048
%AI - %AI1 to %AI512
%I - %I1 to %I2048
%AQG - %AQG1 to %AQG32
%QG - %QG1 to %QG64
%AIG - %AIG1 to %AIG32
%IG - %IG1 to %IG64
%T - %T1 to %T2048
%S - %S1 to %S16
%K - %K1 to %K7
%D - %D1 to %D1023
%SR - %SR1 to %SR192

OCS_LtdReg.eds:- This file contains limited number of registers i.e., 254 registers of each type and it takes 2 - 3 secs for loading EDS file. The details of the registers range are as follows:

%R - %R1 to %R254
%M - %M1 to %M254
%AQ - %AQ1 to %AQ254
%Q - %Q1 to %Q254
%AI - %AI1 to %AI254
%I - %I1 to %I254
%AQG - %AQG1 to %AQG32
%QG - %QG1 to %QG64
%AIG - %AIG1 to %AIG32
%IG - %IG1 to %IG64

%T - %T1 to %T254
%S - %S1 to %S16
%K - %K1 to %K7
%D - %D1 to %D254
%SR - %SR1 to %SR192

4.2.2 Special Functions Objects

CANopen defines three special function protocols such as sync protocol, time-stamp Protocol and Emergency Protocol.

4.2.2.1 Synchronization Objects (Sync) protocol

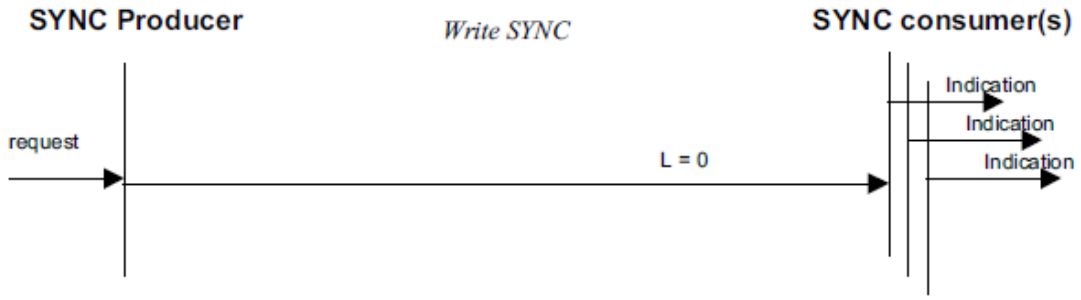
The Sync Object is broadcasted periodically by the Sync Producer. The Sync-Producer provides the synchronization-signal for the Sync-Consumer. When the Sync-Consumer receives the signal they start carrying out their synchronous tasks.

The time period between Sync messages is defined by the communication Cycle Period, which may be reset by a configuration tool to the application devices during the boot-up process. The Sync message is mapped to a single CAN frame with the identifier 128 by default. The Sync message does not carry any data.

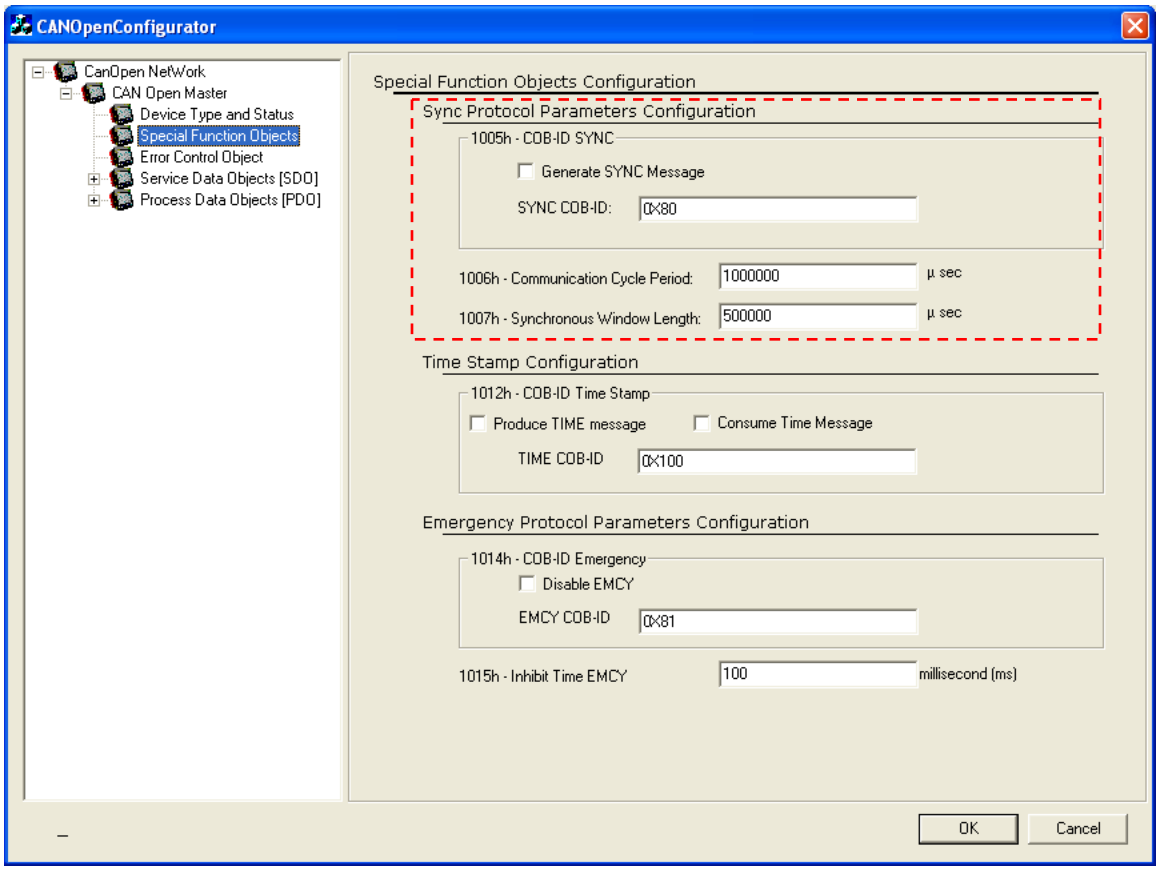
Devices which operate synchronously may use the SYNC object to synchronise their own timing with that of the Synchronisation Object producer.

The below figure shows a SYNC Protocol:

The SYNC transmission follows the producer/consumer push model.



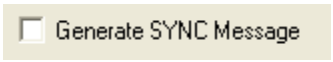
Sync Protocol Parameters Configuration



The Box marked in RED in above figure is for Sync Protocol Parameters Configuration

The Synchronization Object is broadcasted periodically by the SYNC producer. This SYNC provides the basic network clock

Generate SYNC Message: Selecting this option will produce SYNC messages i.e. Node will act as SYNC producer. Available only in QX, XL6 and NX series, but XLe/t can act as SYNC consumer. In the given network only one node can be SYNC producer and other nodes will act as SYNC consumer.



SYNC COB-ID: Default COB-ID for Sync message is 0x80; user can change as per his requirement.

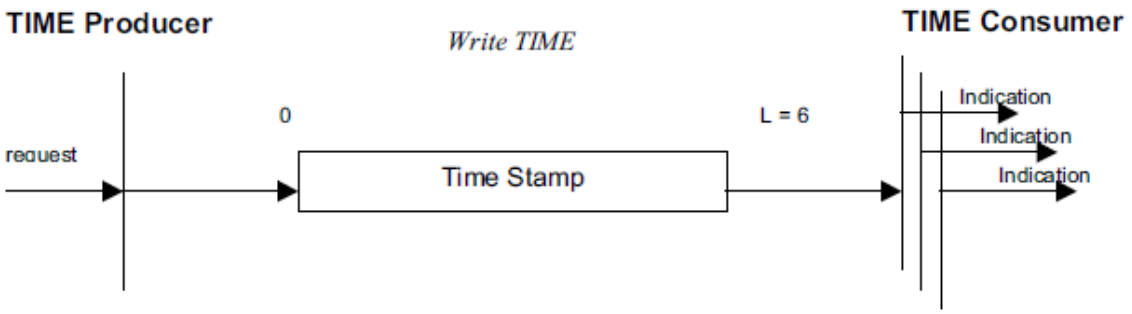
Communication Cycle Period: Enter required SYNC object cycle period in microseconds. Zero if not used.

Synchronous Window Length: Enter the length of the time window for synchronous PDOs in microseconds. It is Zero if not used.

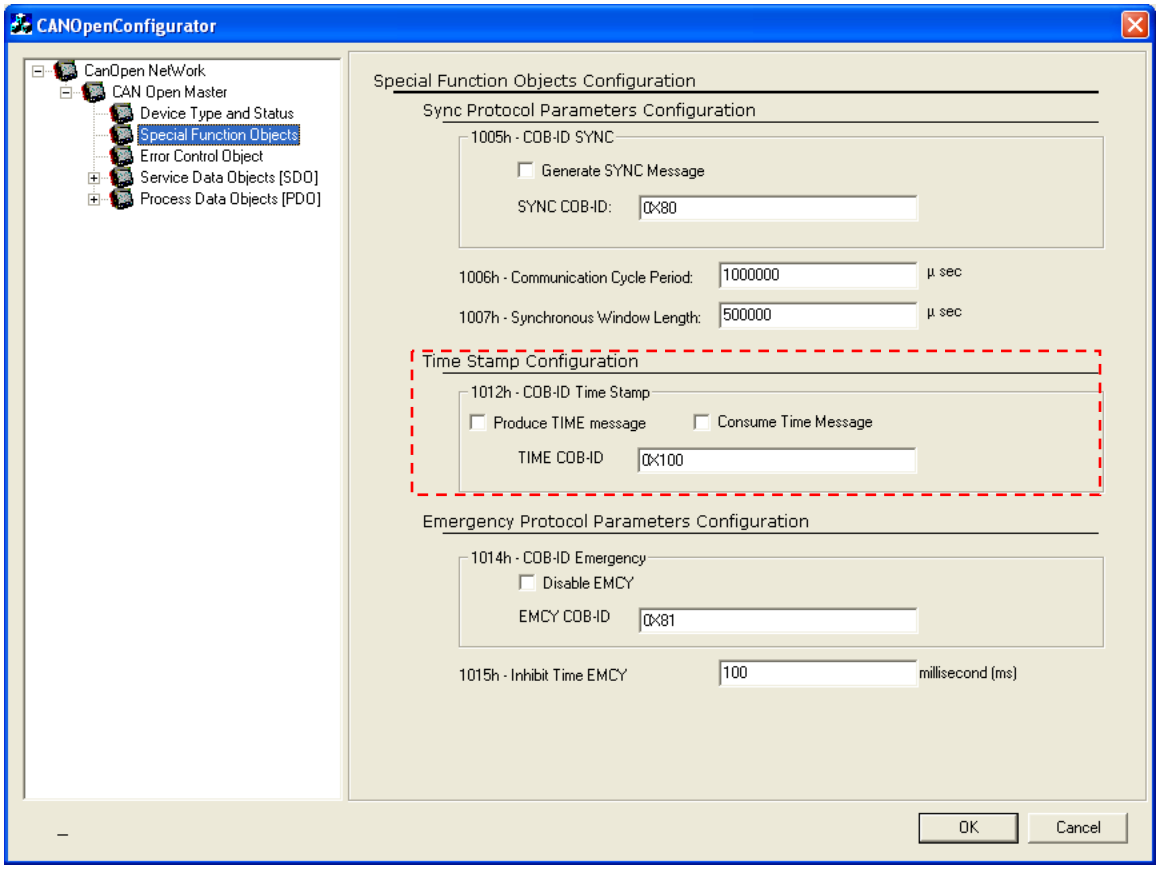
4.2.2.2 Time Stamp Protocol

By means of Time-Stamp, a common time frame reference is provided to application devices. It contains a value of the type Time-of-Day. This object transmission follows the producer/consumer push model. The associated CAN frame has the pre-defined identifier 256 and a data field of 6-byte length.

The below figure shows a Time Stamp Protocol:



Time Stamp Configuration



The Box marked in RED in above figure is for Time Stamp Protocol Configuration.

By means of Time stamp object a common time frame reference is provided to devices. It contains a value of the type TIME_OF_DAY. The identifier of the Time object is located at object index 1012h.

Time COB-ID: Default COB ID for Time stamp is 0X100. User can change as per his requirements.

Produce TIME Message: It is used to send the time of a controller to other devices.

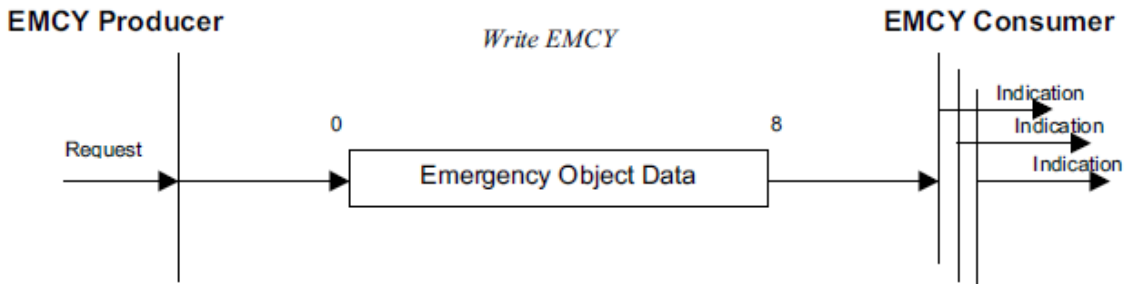
Consume TIME Message: Used to accept the time from a device.

4.2.2.3 Emergency Protocol

The Emergency message is triggered by the occurrence of a device internal error situation and is transmitted from an Emergency producer on the concerned application device. This makes them suitable for interrupt type error alerts. Emergency message is transmitted only once per 'error event'. As long as no new errors occurs on a device, no further Emergency message shall be transmitted. Emergency object may be received by zero or more Emergency consumers.

The reaction of the Emergency consumer is application-specific. CANopen defines several Emergency Error Codes to be transmitted in the Emergency message, which is a single CAN frame with 8 data byte.

The below figure shows a Emergency Object Protocol:

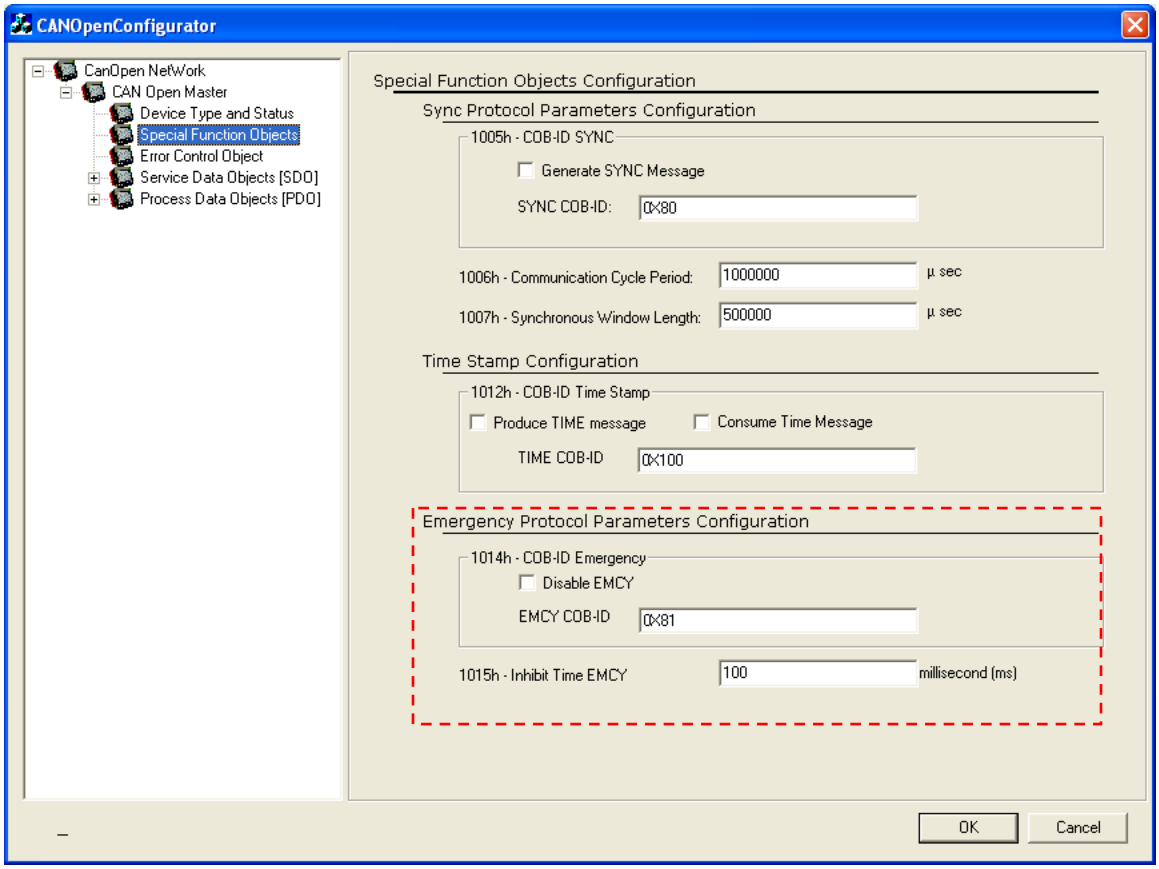


It is not allowed to request an Emergency Object by a remote transmission request (RTR).

Emergency Protocol Parameters Configuration

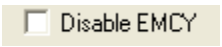
The Box marked in RED in below figure is for Emergency Protocol Parameters Configuration.

Master node will consume Emergency Messages generated from several nodes



EMCY COB-ID: Default COB-ID for EMCY message is 0x80 + NODE ID, user can change as per his requirement.

Disable EMCY: Selecting this option will disable generation of Emergency messages.



Inhibit Time EMCY: Inhibit Time of a data object defines the minimum time that has to elapse between two consecutive invocations of a transmission service for that data object. It mainly avoids sending of too many messages at a time. Enter required inhibit time for the EMCY message.

4.2.3 Error Control Object

There are two types of error control protocol: Node Guarding Protocol and Heartbeat protocol.

4.2.3.1 Node Guarding Protocol

The guarding is achieved through transmitting guarding requests (Node guarding protocol) by the NMT Master. If a NMT Slave has not responded within a defined span of time (node life time) or if the NMT Slave's communication status has changed, the NMT Master informs its NMT Master Application about that event.

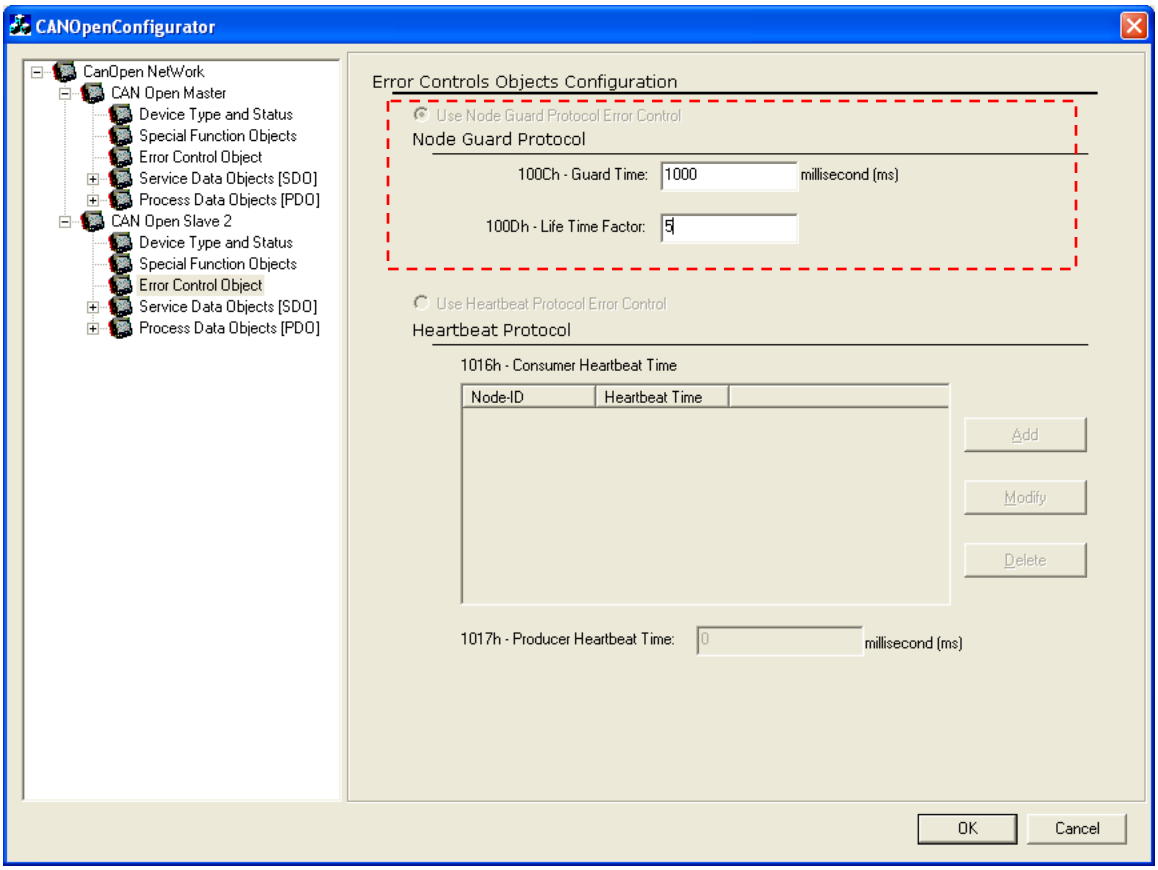
If Life guarding (NMT slave guarded NMT master) is supported, the slave uses the guard time and lifetime factor from its Object Dictionary to determine the node life time. If the NMT Slave is not guarded within its life time, the NMT Slave informs its local Application about that event. Guarding starts for the slave when the first remote-transmit-request for its guarding identifier is received. This may be during the boot-up phase or later

1. This protocol is used to detect remote errors in the network.
2. Each NMT Slave uses one remote COB for the Node Guarding Protocol.
3. The NMT Master polls each NMT Slave at regular time intervals. This time-interval is called the guard time and may be different for each NMT Slave.
4. The node life time is given by the guard time multiplied by the life time factor. The node life time can be different for each NMT Slave.
5. If the NMT Slave has not been polled during its life time, a remote node error is indicated through the 'Life Guarding Event' service.

Node Guard Protocol Configuration

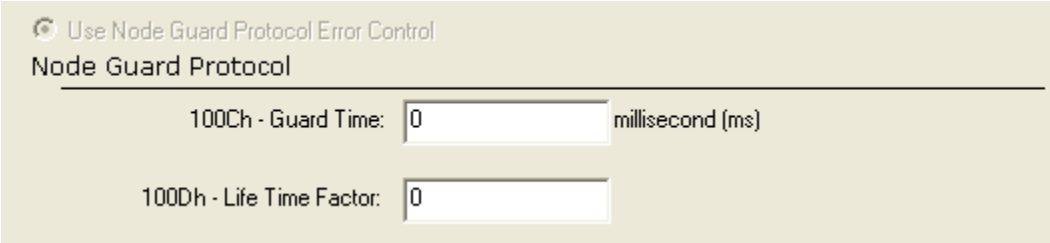
Note: In a CANopen Network, select either of the following two options for the error control Protocol. The choice for selecting an option of error control protocol will be available only in the master node or in case of a Single slave Configuration.

1. Select '*Use Node Guard Protocol Error Control*' in master node.
2. Configure Guard time and Life time factor for individual slave. Master node does not require configuration of Guard time and Life time factor.



NOTE:

Error Control protocol can be disabled by selecting Node Guard type and entering 'Guard Time' & 'Life Time factor' value as Zero.



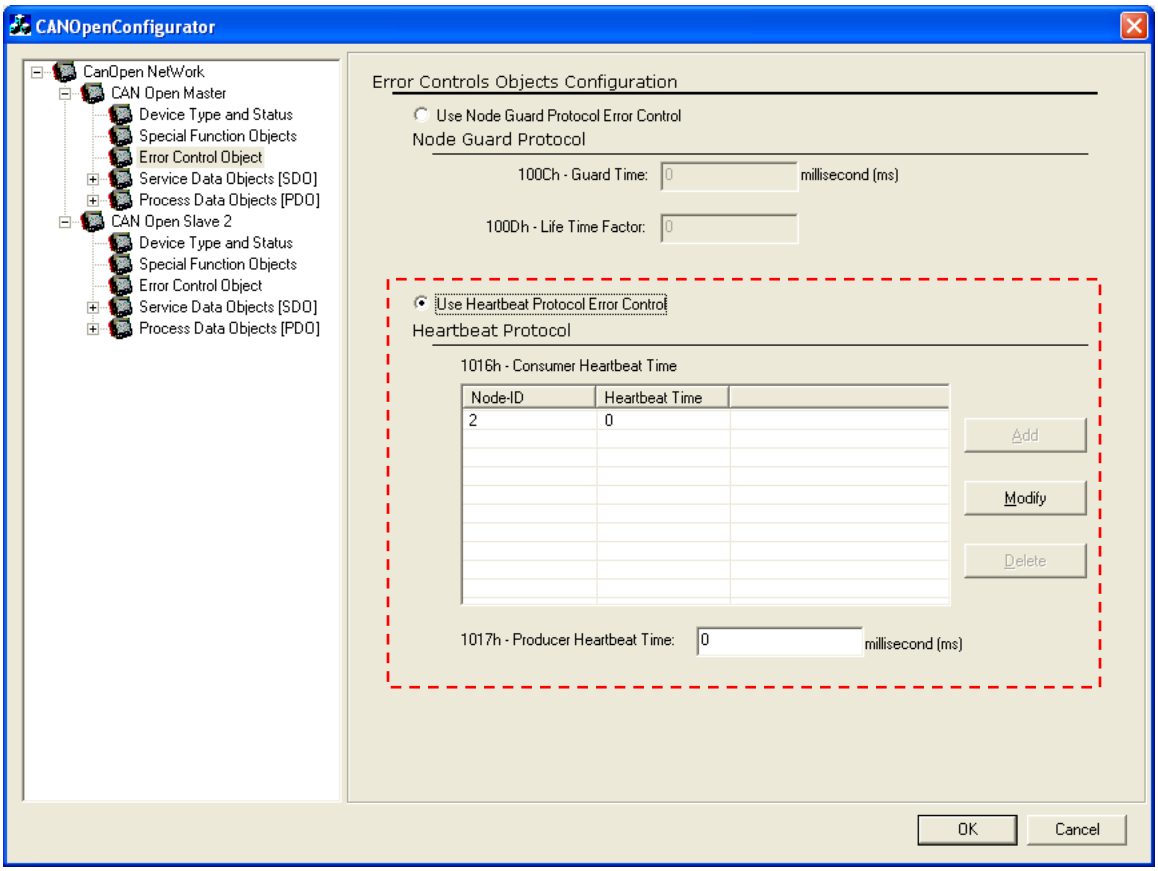
4.2.3.2 Heart Beat Protocol

The Heartbeat protocol is used to monitor the nodes in the network and verify that they are alive. A heartbeat producer (usually a slave device) periodically sends a message with binary function code of 1110 and its node id (COB ID = 0x700 + node id). The data part of the frame contains a byte indicating the node status. The heartbeat consumer reads these messages. If the messages fail to arrive within a certain time limit (defined in the object directory of the devices) the consumer can take action to, for example, reset the device or indicate an error. Frame format is: COBID + DATA (status of node)

A Heartbeat Producer transmits a Heartbeat message cyclically. One or more Heartbeat Consumer receives the message. The relationship between producer and consumer is configurable via the object dictionary.

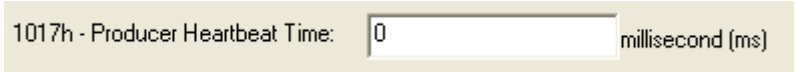
The Heartbeat Consumer guards the reception of the Heartbeat within the Heartbeat Consumer Time. If the Heartbeat is not received within the Heartbeat Consumer Time a Heartbeat Event will be generated.

Heart Beat Protocol Configuration

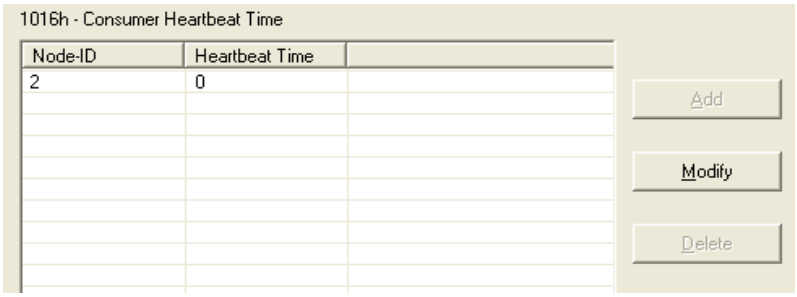


Select '*Use Heart Beat Protocol Error Control*'.

Configure Producer Heartbeat Time if given node is Heartbeat producer.



Configure Consumer Heartbeat time for individual node if node is Heartbeat consumer.



NOTE:

1. Master node by default will act as heartbeat consumer.
2. Given Node can be both producer and consumer
3. The consumer time configured must be greater than the producer time.

4.2.4 Service Data Object (SDO)

The SDO protocol is used to set and read values from the object directory of a remote device. The device whose object directory is accessed is the SDO server and the device accessing the remote device is the SDO client. The communication is always initiated by the SDO client. In CANopen terminology, communication is viewed from the SDO server, so that a read from an object directory results in an SDO upload and a write to directory is an SDO download.

The client can control which data set is to be transferred. The contents of the data set are defined within the Object Dictionary.

Basically SDO can transfer data in two types: **1. Expedited Transfer** and **2. Block Transfer**

Expedited Transfer:

Basically a SDO is transferred as a sequence of *segments*. Prior to transferring the segments there is an initialization phase where client and server prepare themselves for transferring the segments. For SDOs, it is also possible to transfer a data set of up to four bytes during the initialization phase. This mechanism is called an *expedited* transfer.

Block Transfer:

SDO can be transferred as a sequence of *blocks* where each block is a sequence of up to 127 segments containing a sequence number and the data. Prior to transferring the blocks there is an initialization phase where client and server can prepare themselves for transferring the blocks and negotiating the number of segments in one block. After transferring the blocks there is a finalization phase where client and server can optionally verify the correctness of the previous data transfer by comparing checksums derived from the data set. This transfer type mentioned above is called a *block* transfer which is faster than the segmented transfer for a large set of data.

Features of SDO

1. Used for Point-To-Point communication between a configuration tool or master/manager and the nodes.
2. It allows complete access to all entries in the Object Dictionary of a node
    a. Includes all process data
3. Confirmed communication mode
    a. Each request gets a response
4. Supports transfer of long data blocks such as upload or download of code blocks

a. Up to 7 bytes per message, every message confirmed by receiver
b. Special “block transfer mode” allows transmitting blocks of messages with just one confirmation

SDO Configuration

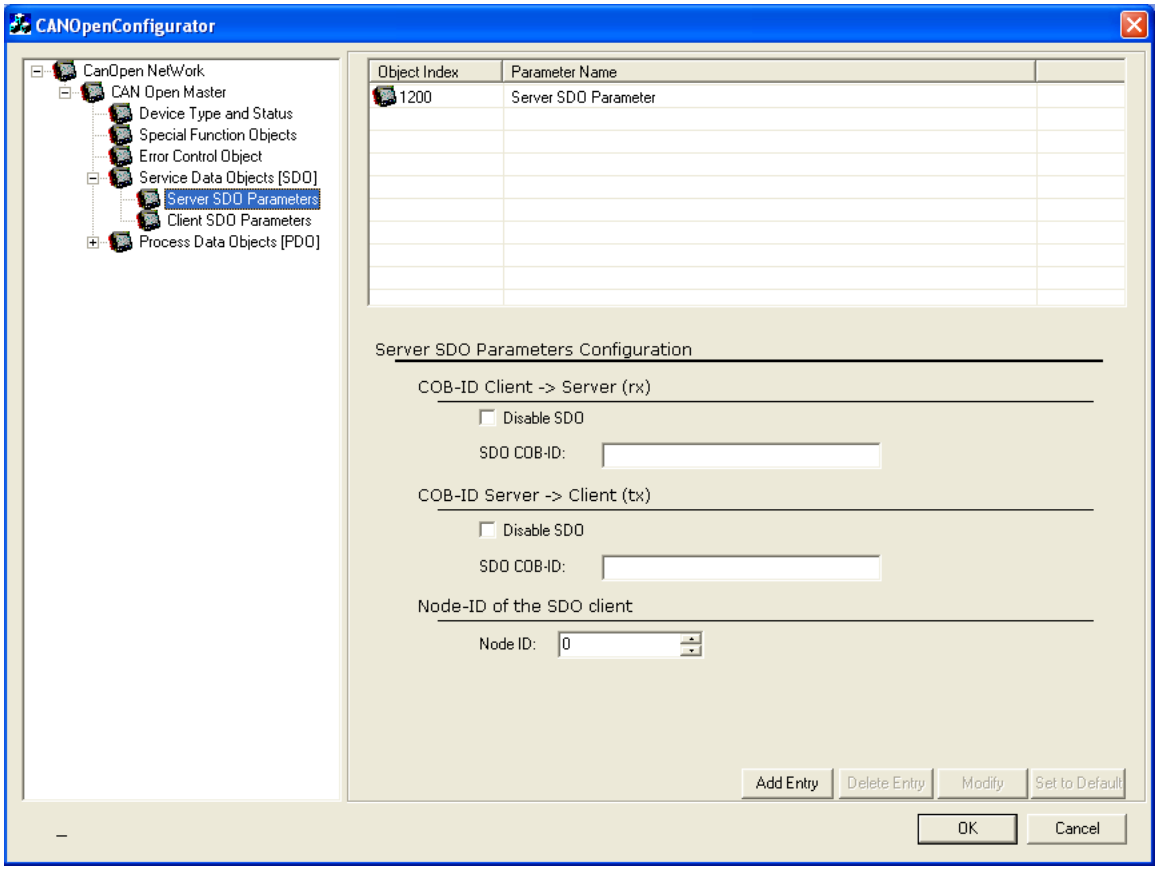
Service data objects are designed to access entries in device object dictionary. By default SDOs are configured automatically. Each node will support single Server SDO and additional server SDOs can be configured. Client SDOs are supported only in Master.

NOTE:
For every new slave device added in Master Configuration a new SDO client gets created in Master and default server SDO gets created in Slave Node. These SDO creation happen by default in configuration and user need not take care anything here. These default SDO clients and server are created for Slave reconfiguration purpose

Server SDO / Client SDO Parameters

Server SDO: The Server SDO receives the SDO messages from the corresponding Client SDO and responds to each SDO message or a block of SDO messages (SDO block transfer).

Client SDO: The Client SDO initiates the SDO communication by means of reading or writing to dictionary of the SDO server device.



Server SDO and Client SDO will have the following buttons with same functionality. Except 'Add Entry' button all the other 3 buttons will remain disable till the user select any object in the list.

Add Entry: To add an entry in the list.

Delete Entry: To delete an entry in the list

Modify: To modify an entry already existing in the list.

Set to Default: This button will set the selected object in the list to its default value. This will search for the default value in the EDS file selected for this node (In this case master Node). If the EDS file is not selected or the selected EDS file does not have any entry corresponding to the object selected in the list, then it will not work.

Note: Add Entry is disabled in *Client SDO Parameters*

COB ID Client → Server (rx)

Disable SDO: Select this option to disable configured SDO

SDO COB-ID: Configure required 11 bit COB-ID in this field, click on 'Set default' to read from EDS file.

COB ID Server → Client (tx)

Disable SDO: Select this option to disable configured SDO

SDO COB-ID: Configure required 11 bit COB-ID in this field, click on 'Set default' to read from EDS file.

Node ID: Enter the node ID of the SDO Client.

4.2.5 Process Data Object (PDO)

"Process Data Object" protocol is used to process real time data among various nodes. It can transfer up to 8 bytes (64bits) data in one PDO either from or to the device. One PDO can contain multiple object dictionary entries. The objects within one PDO are configurable using the mapping parameter object dictionary entries. Process Data Objects (PDOs) are mapped to a single CAN frame using up to 8 bytes of the data to transmit application objects. Each PDO has a unique identifier and is transmitted by only one node, but it can be consumed by more than one node.

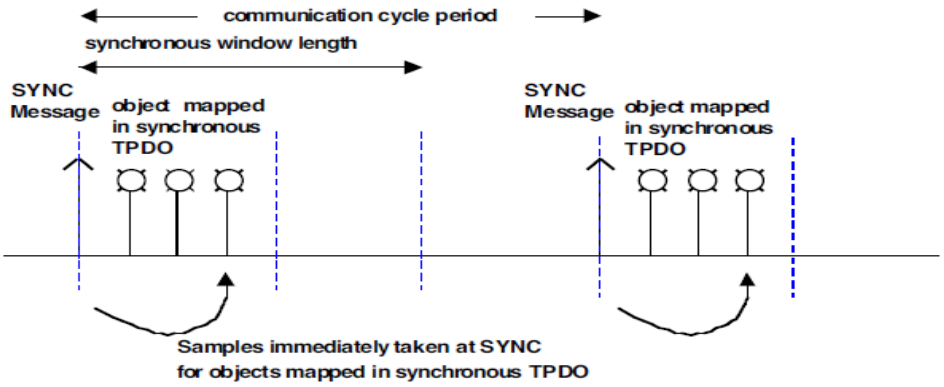
There are two kinds of PDOs:

TPDO (Transmit PDO): for data coming from the device
RPDO (Receive PDO): for data going to the device

I.e. with RPDO user can send data to the device and with TPDO user can read data from the device. PDOs can be sent synchronously or asynchronously.

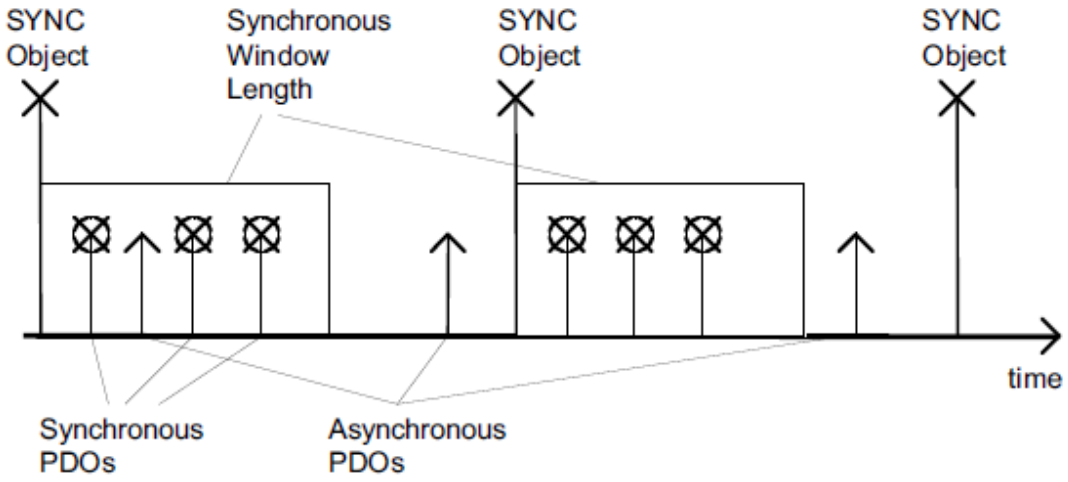
PDO transmissions may be driven by an internal event, by an internal timer, by remote requests or by the Sync message received.

a. **Event- or timer-driven**: An event (specified in the device profile) triggers message transmission. An elapsed timer additionally triggers the periodically transmitting nodes.
b. **Remotely requested**: Another device may initiate the transmission of an asynchronous PDO by sending a remote transmission request (remote frame)
c. **Synchronous transmission**:
   - In order to initiate simultaneous sampling of input values of all nodes, a periodically transmitted Sync message (Sync Object) is required. Synchronous transmission of PDOs occurs in Cyclic and Acyclic transmission modes.
     - o In Cyclic transmission the node waits for the Sync message, after which it sends its measured values. Its PDO transmission type number (1 to 240) indicates the Sync rate it listens to (how many Sync messages the node waits before the next transmission of its values)
     - o Acyclically transmitted synchronous PDOs are triggered by a defined application-specific event. The node transmits its values with the next Sync message but will not transmit again until another application-specific event has occurred
     - o In case of Synchronous RPDO messages, values are updated only after reception of Sync message. i.e., In first sync, PDO messages will be sent from the transmitter and on next sync, PDO message values will be updated in the receiver though it might have received PDO message much before arrival of synch message



d. **Asynchronous** messages are sent after internal or external trigger and Synchronous PDOs are sent after the SYNC message. For example, you can make a request to a device to transmit TPDO that contains data you need by sending empty TPDO with RTR flag (if the device is configured to accept TPDO requests).
With RPDOs user can, for example, start two devices simultaneously. User only need to map the same RPDO into two or more different device and make sure those RPDOs are mapped with the same COB ID

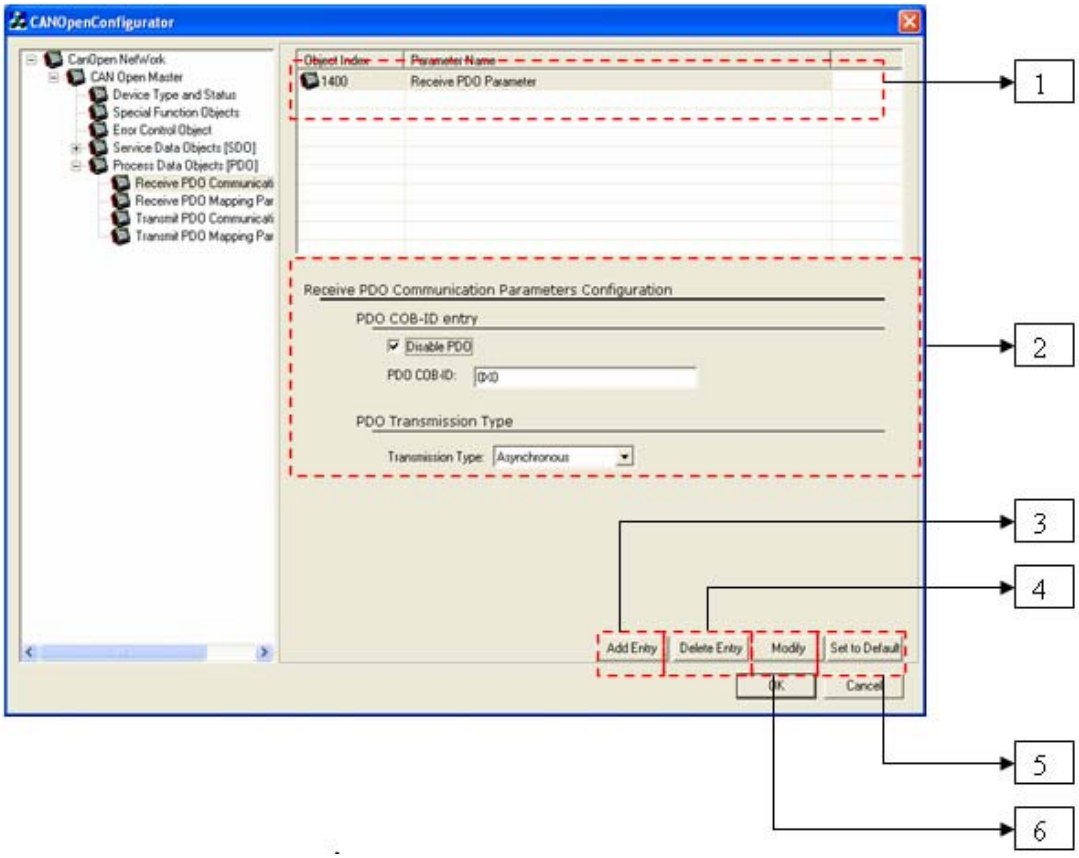
The below figure shows the Asynchronous and Synchronous transmission types:



PDO Configuration

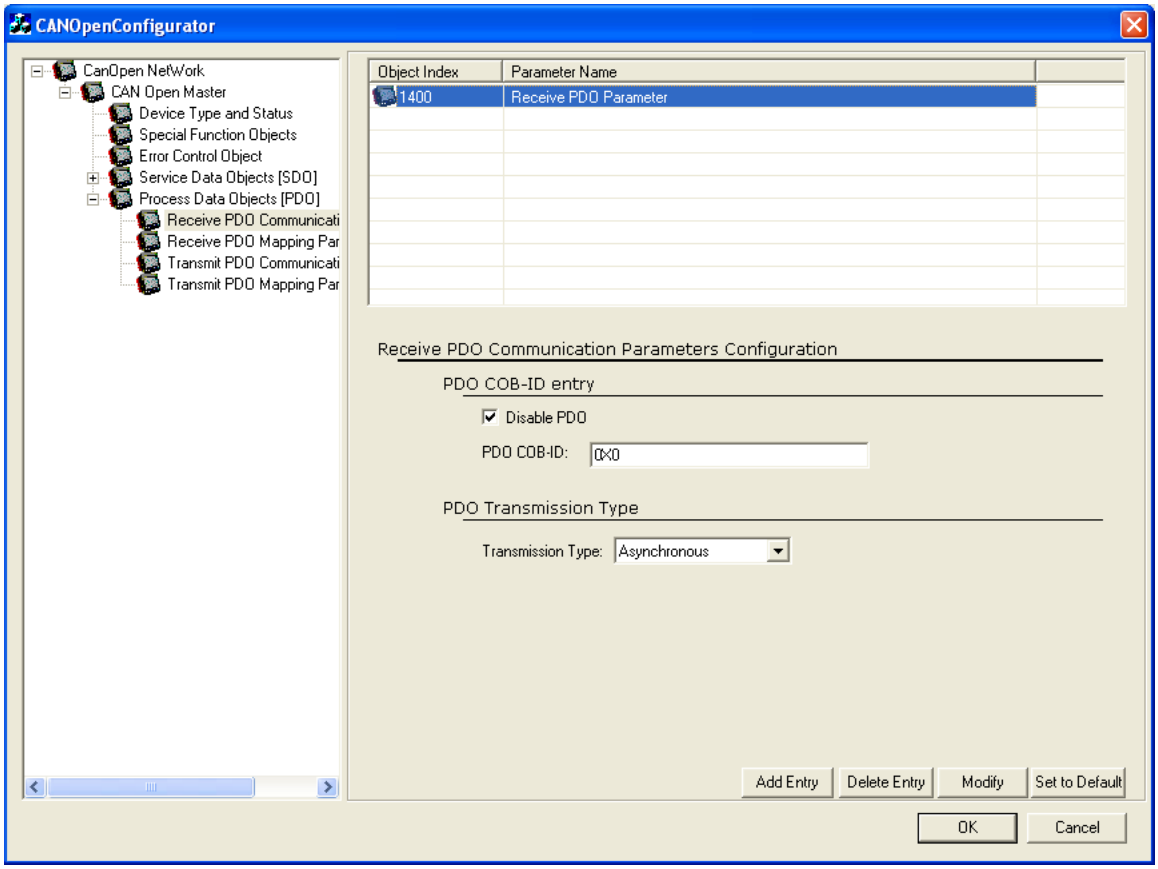
The real-time data transfer is performed by means of "Process Data Objects (PDO)".

PDO Communication Parameter Configuration:



1. Configured PDO list.
2. PDO configuration field
3. Click to add new PDO entry
4. Click to delete Selected entry
5. Click to take default information for selected entry from EDS
6. Click to modify the selected entry

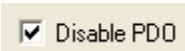
4.2.5.1 Receive PDO Communication Parameters (RPDO)



PDO COB-ID Entry

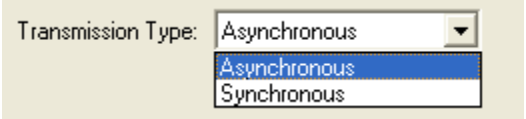
PDO COB-ID: Configure required 11 bit COB-ID in this field, click on 'Set default' to read from EDS file

Disable PDO: Select this option to disable configured PDO



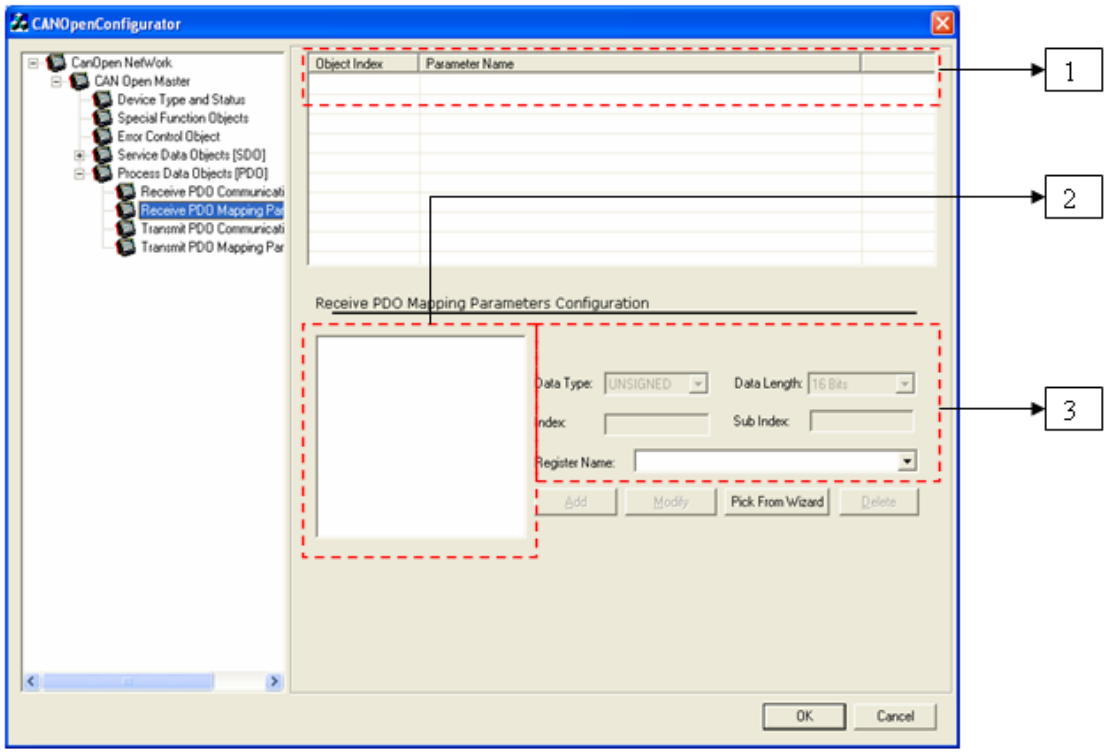
PDO Transmission Type: Select type of PDO receive method (Depends upon actual transmission type)

Asynchronous: Messages are transmitted after any data changes.
Synchronous: Messages are transmitted after reception of Sync message.



NOTE:
In case of Synchronous RPDO messages, values are updated only after reception of Sync message. i.e., in first sync, PDO messages will be sent from the transmitter and on next sync, PDO message values will be updated in the receiver though it might have received PDO message much before arrival of synch message.

4.2.5.2 Receive/Transmit PDO Mapping parameters



1. Configured PDO Mapping list
2. List box displaying mapped object for given PDO
3. To view and edit mapped object Characteristics

For Receive PDO Mapping parameters and Transmit PDO Mapping parameters, the entries in the list on the top will get automatically added when the user adds an entry in the Receive PDO Communication or in the Transmit PDO Communication respectively.

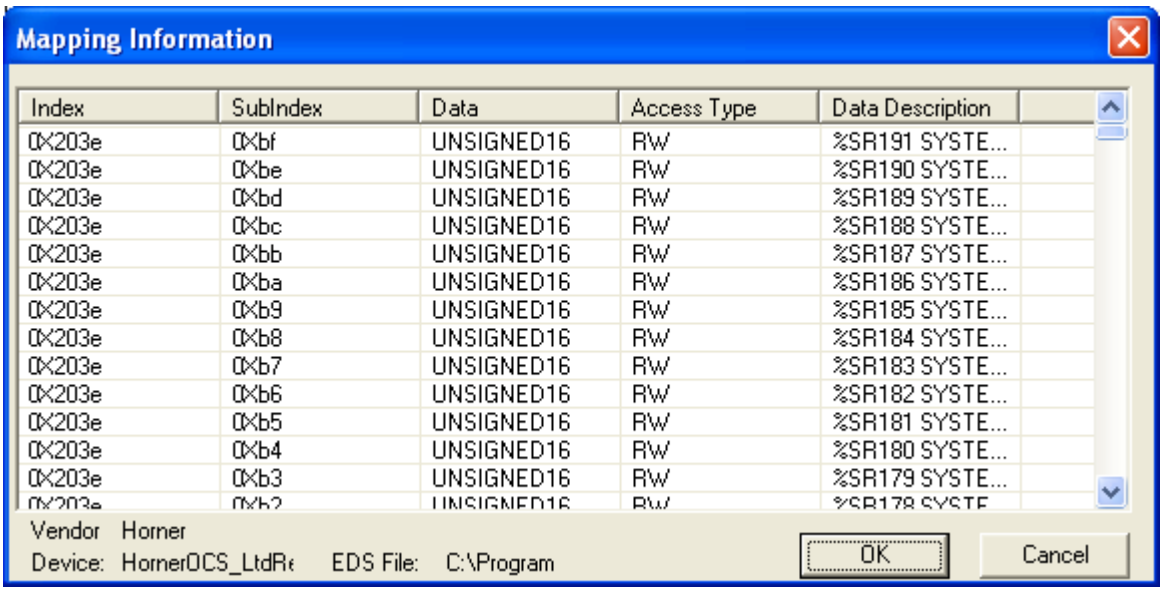
Receive PDO Mapping and Transmit PDO Mapping will have same buttons with same functionality.

Register Name: This field will enable only if EDS files supplied by Horner.

Add: User can add mapping Data in the box immediate left to add button. This will add an entry against the item selected in the list on the top

Modify: This button is used to modify any existing Block information (Mapped Data). This button will work only if an item in the box is selected

Pick From Wizard: This button will launch a dialog box containing different values that can be used as valid entry for mapping data. Before using this user needs to select the EDS file by right clicking on the root node (in this case it is "CANopen Master"). All the available information will be populated in the list box as shown in the Figure below. Once the user selects an object from the list box Mapping Information and clicks OK the values will automatically populate in the index and sub Index edit box of the PDO mapping parameter configurations. Then the user can directly click the add button to add the entry in the mapping Data.

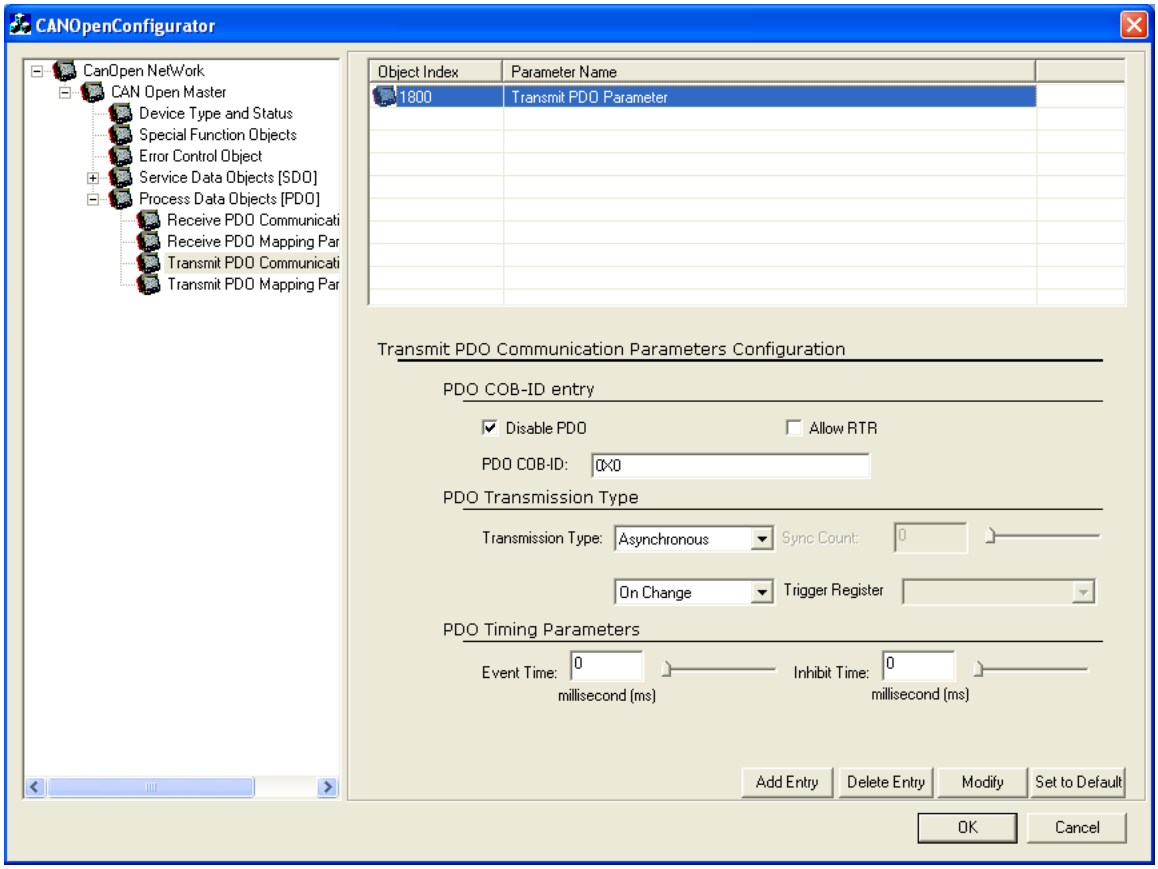


NOTE:

1. Single PDO can take up to 4 -16 bit objects, or 8 – 8 bit Objects or 2-32 Objects and vice versa.
2. User can directly map OCS register to PDOs, conversion of OCS native register index to CANopen index and vice versa is done by the Configurator

Delete: It is used to delete existing Block information. This button will work only if an item in the box is selected.

4.2.5.3 Transmit PDO Communication Parameters



PDO COB-ID Entry

PDO COB-ID: Configure required 11 bit COB-ID in this field, click on 'Set default' to read from EDS file

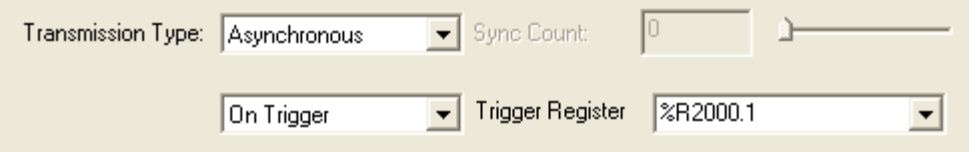
Disable PDO: Select this option to disable configured PDO

Allow RTR: Select this option if PDO needs to be transmitted on Remote request.

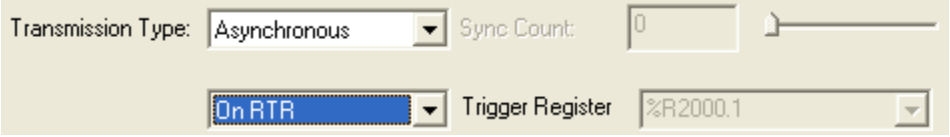
PDO Transmission Type: Select type of PDO transmission method, select on change, on trigger or On RTR type. Trigger type requires trigger bit to be configured

Asynchronous On Change Configuration: On change, messages are transmitted as soon as message data changes

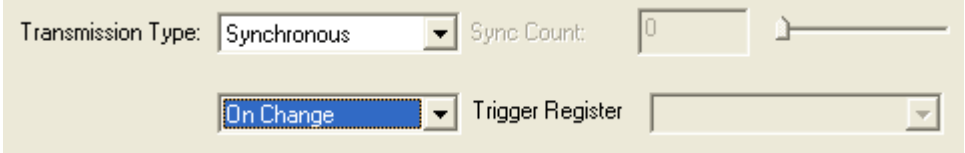
Asynchronous On Trigger Configuration: When trigger bit is set then PDO is transmitted and trigger bit is reset



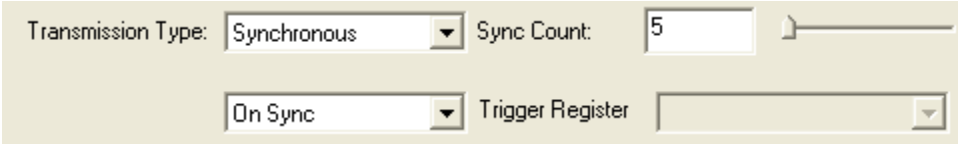
Asynchronous On RTR Configuration: PDO is transmitted on RTR (PDO must be enabled for RTR).



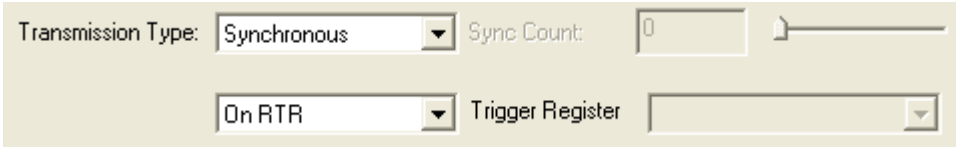
Synchronous On Change Configuration: PDO is transmitted on change of message data and after reception of Sync message within sync window time



Synchronous On Sync Count Configuration: PDO is transmitted on reach of 'n' sync count within sync window time

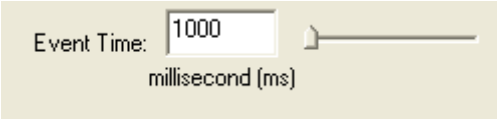


Synchronous On RTR Configuration: PDO is transmitted on RTR (PDO must be enabled for RTR) and after reception of Sync message within sync window time

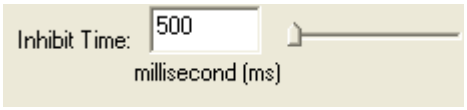


PDO Timing Parameter:

Event Time: Configure event time for asynchronous messages in ms. zero if not used



Inhibit Time: Inhibit time of a data object defines the minimum time that has to elapse between two consecutive invocations of a transmission service for that data object. Configure inhibit time for asynchronous messages in ms. zero if not used

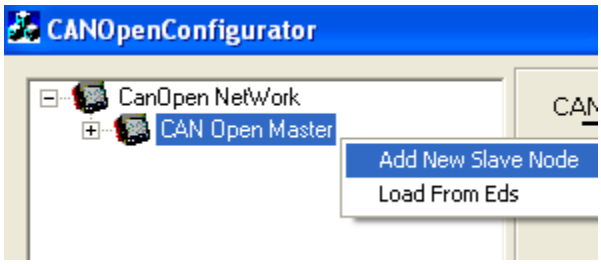


Chapter 5: Slave Configuration

For Minimum and Full Configuration of Slave refer ***Chapter 4***.

By default, without configuration download, the OCS will act as slave node

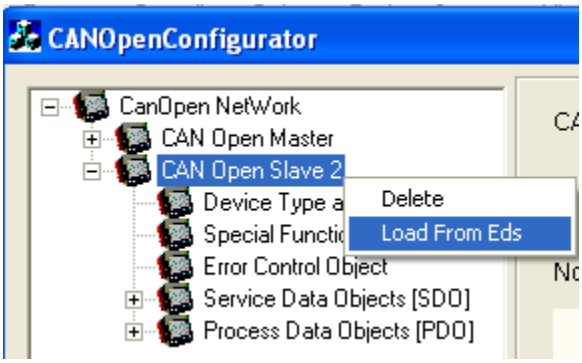
Right click on Master Node tree to add slaves (as shown)



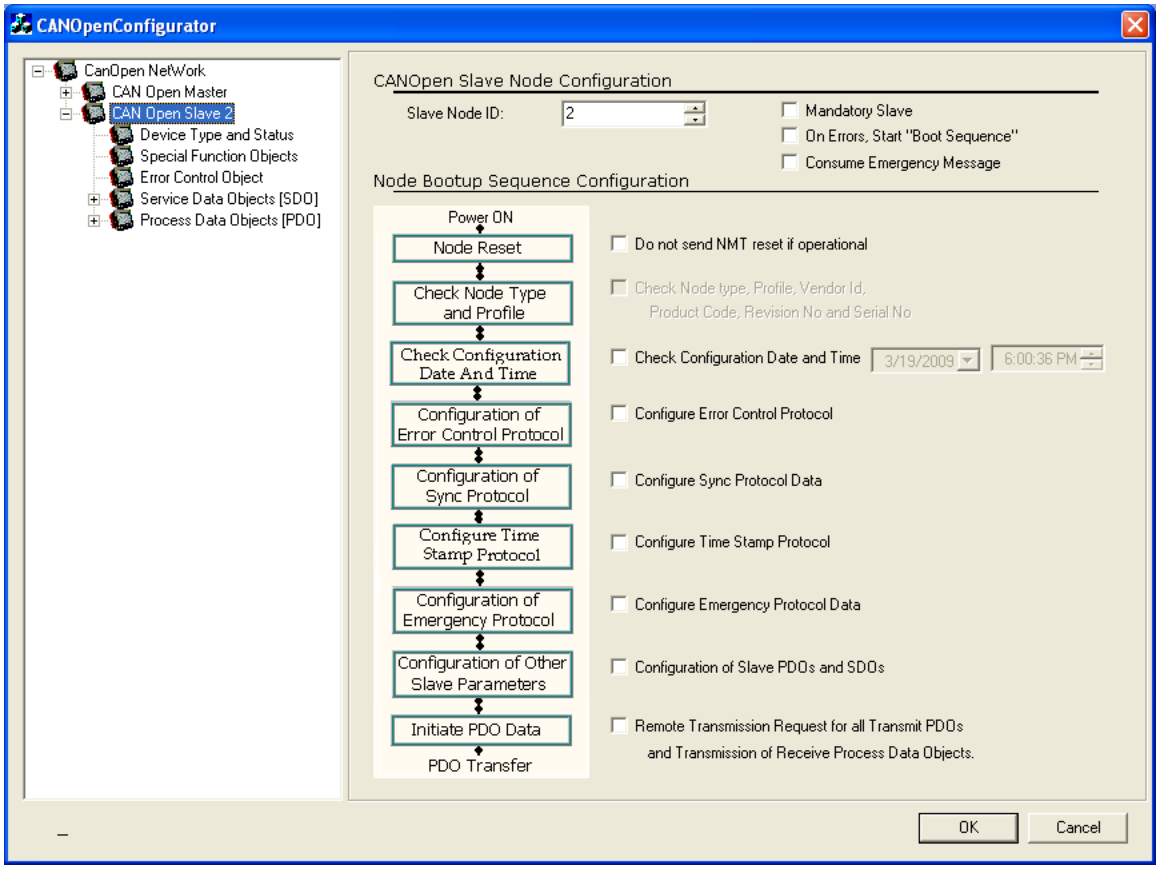
NOTE: - It is possible to configure upto 64 PDO's from the Cscape whereas; only 16 PDO's can be reconfigured by master during RUN time.

Load Slave Device specific EDS file (as shown)

Right click on slave node and select Load from Eds.

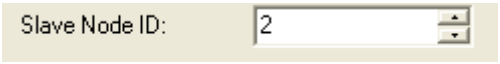


The below figure shows a slave node



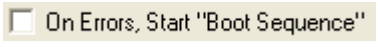
Master node needs '*Node Bootup Sequence Configuration*' information to know when to configure the slave node and what all parameters need to be checked before configuring the particular slave. This is not necessary while configuring single slave, since there is no master configuration done in case of single slave configuration.

Configure required **Slave node ID**

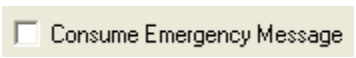


Mandatory Slave: Select this option if slave node is Mandatory

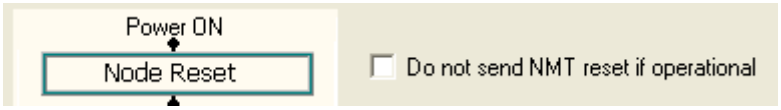
On Error, Start "Boot Sequence": Select this option if given node has to be rebooted on error



Consume Emergency Message: Select this option if master node has to consume emergency message from this node.

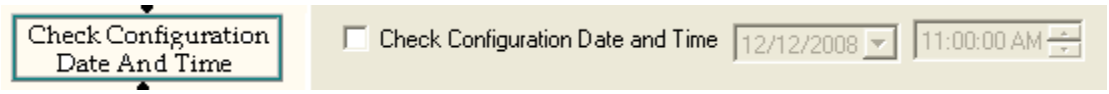


Do not send NMT reset if operational: If selected Master will not send reset command on boot up if node is in operational state



Check Node type, Profile, Vendor Id, Product Code, Revision No and Serial No.: This option gets enabled only after loading the EDS file. Selecting this option master will check these information on network Boot-up. If any of these parameters doesn't match then master sets error and stops configuration of given node.

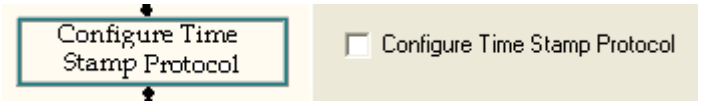
Check Configuration Date and Time: Selecting this option Master node sends SDO requests to upload Date and Time of Last Configuration downloaded on network Boot-up. If Current Date and Time is different from that returned from slave, then Master Node continues with reconfiguration of selected parameters.



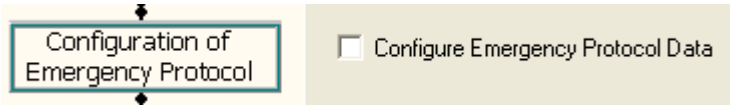
Configure Error Control Protocol: Selecting this option Master Node reconfigures Error Control protocol on network Boot-up. (If supported in case of 3rd party slaves)

Configure Sync Protocol Data: Selecting this option Master Node reconfigures Sync Protocol data on network Boot-up. (If supported in case of 3rd party slaves)

Configure Time Stamp Protocol: Selecting this option Master Node reconfigures Time stamp Protocol data on network Boot-up. (If supported in case of 3rd party slaves)

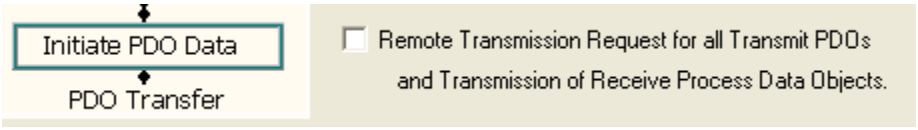


Configure Emergency Protocol data: Selecting this option Master Node reconfigures Emergency Protocol data on network Boot-up. (If supported in case of 3rd party slaves)

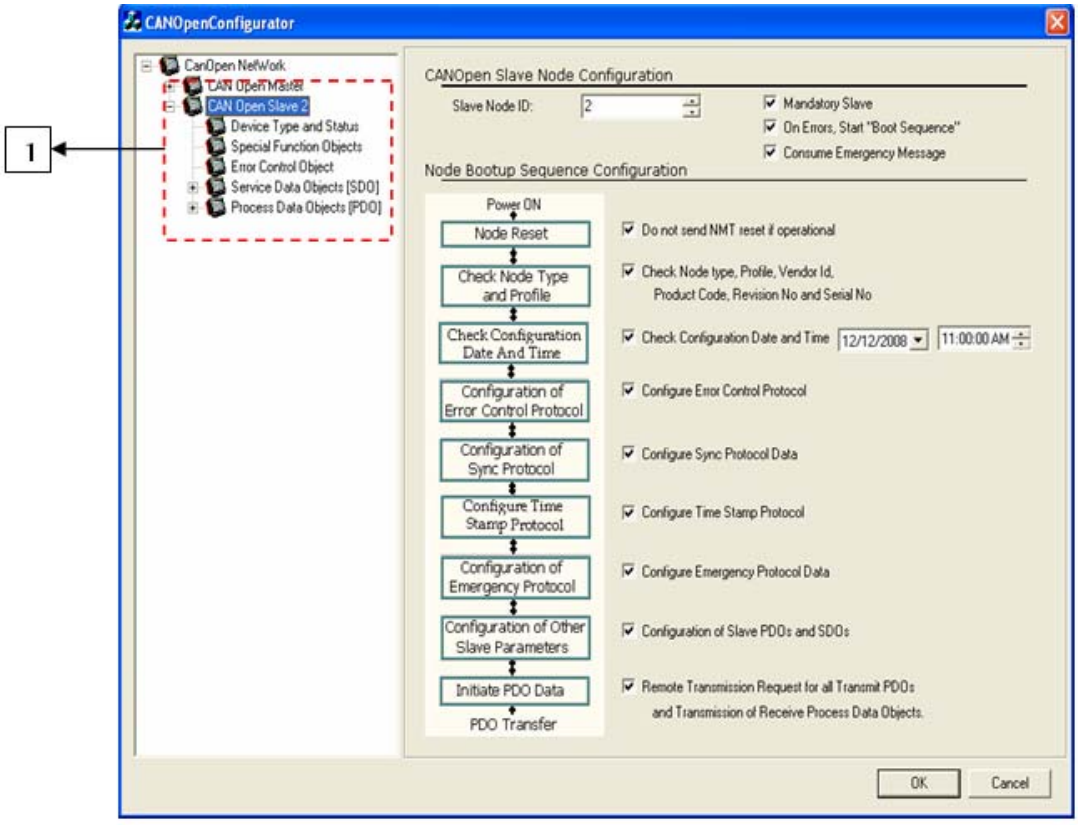


Configuration of Slave PDOs and SDOs: Selecting this option Master Node Reconfigures SDOs and PDO data.

Remote Transmission Request for all Transmit PDOs and Transmission of Receive Process Data Objects: Selecting this option Master Node sends RTR request for transmit PDOs of given slave and transmits all receive PDOs on network Boot-up.



Configure other CANopen parameters for given slaves as shown



1. Configuration of these parameters are same as explained in CANopen Master Configuration. (Refer to ***Chapter 4***)

Chapter 6: Single Slave Configuration

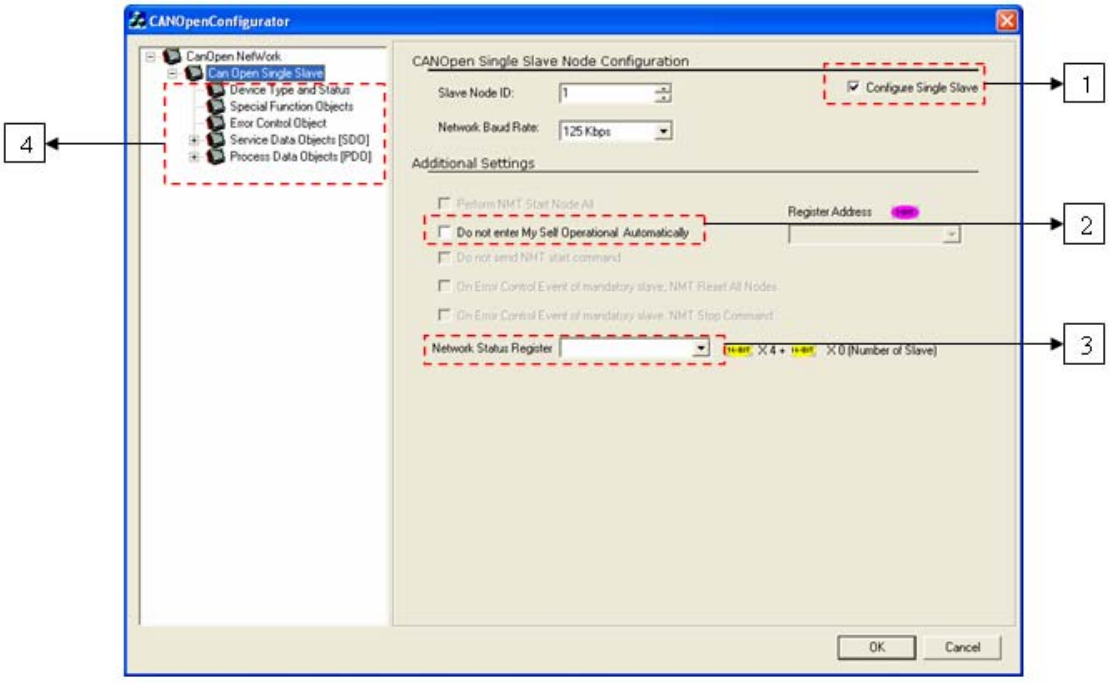
Master node needs '*Node Bootup Sequence configuration*' information to know when to configure the slave node and what all parameters need to be checked before configuring the particular slave. This is not necessary while configuring single slave, since there is no master configuration done in case of single slave configuration.

After Configuration download, firmware takes at least 10 sec to start CANopen communication with new configuration. Node will switch to pre-operational state during new CANopen configuration download; if node is master then it will send NMT command to all slave nodes to switch to pre-operational state.

The table below shows the minimum and full configuration for a single slave / standalone slave

6.1 Minimum and Full Configuration

Minimum Configuration	Full Configuration
Configure '*Slave Node ID*' and '*Network Baud Rate*'	Configure '*Slave Node ID*' and '*Network Baud Rate*'
Configure "*Do Not Enter My Self Operational Automatically*" flag	Configure "*Do Not Enter My Self Operational Automatically*" flag
	Configure '*Network Status Register*'
	Configure '*Error Control Object*'
	Configure '*Process Data Object (PDO)*'



1. Select ‘Configure Single Slave’.
2. Select this option if operational state transition is controlled by Master
3. Configure Network Status register address
4. Configure other CANopen network parameters (Similar to Slave Configuration)

Chapter 7: Status Register

7.1 Master and Slave Status Registers

CANopen Master and Slave status registers is 64 bit and details are as below:

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Mas-ter	Sla-ve
1	Object Dictionary Error.	Invalid or corrupt CANopen configuration can cause this error. Configuration of any COB-ID with value '0' (with or without disable option selection) can also cause this error to happen.	No CANopen communication at all, status register will be updated only if firmware finds configured status register address is valid.	Download new configuration with COB-ID other than zero.	N/A	✓	✓
2	Node ID Error (Invalid Node ID).	This error will be flagged if firmware finds invalid Node-id value, i.e. zero or greater than 127. The corrupt configuration can also cause this error to happen.	Firmware will ignore downloaded configuration and will refer default internal slave configuration	Download new configuration	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
3	Error Control Protocol is Not configured	If any of (i.e. Node Guard or Heartbeat) error control protocol is not configured and node configured is single slave.	CANopen communication will work as normal but Master node will not be able to detect some of the slave failures.	User can configure any of Error control protocol.	This error will not trigger EMCY Object.	X	✓
4	TX Error.	Baud rate mismatch, CAN network without proper terminating resistor, improper CAN network cabling can cause this error.	CANopen communication might not work properly.	Verify configured baud rate, check for proper terminating resistor and cabling.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
5	RPDO Object Mapping error.	This error is disabled and indicated by other bits (collectively indicated by Bits 6, 7, 8 & 9).	N/A.	N/A.	N/A.	N/A.	

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Mas-ter	Sla-ve
6	RPDO Set Data error.	Firmware is not able to set the configured internal register value.	RPDO data will not get updated in the register. The RPDO index value which is having error will be updated in status register.	Verify configured index value, index value should be within supported range and with read/write access.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
7	RPDO Invalid Object Index.	Configured RPDO object index value is out of range or not supported by the firmware.	RPDO data will not get updated in the register. The RPDO index value which is having error will be updated in status register.	Verify configured index value, index value should be within supported range.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
8	RPDO DLC Error.	Configured RPDO object count (or data length) does not match with received RPDO message. The received RPDO might have less number of objects (or different data length object) compared to configured one.	RPDO data will not get updated in the register. The RPDO index value which is having error will be updated in status register.	Verify RPDO object count and data length of each object with that of actual RPDO message on bus.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
9	RPDO Mapped Object Count Error.	RPDO is configured without any objects.	RPDO data will not get updated in the register. The RPDO index value which is having error will be updated in status register.	Verify RPDO object mapping and configure as per the requirement.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
10	TPDO Object Mapping error.	This error is disabled and indicated by other bits (Collectively indicated by Bits 11, 12, 13 & 14).	N/A	N/A	N/A	N/A	

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Mas-ter	Sla-ve
11	TPDO Get Data error.	Firmware is not able to get the configured internal register value.	Configured TPDO will not be sent. The TPDO index value which is having error will be updated in status register.	Verify configured index value, index value should be within supported range and with read access.	This error will trigger valid EMCY Object with status register value in 'Manufactur-er Specific Error Field' of message.	✓	✓
12	TPDO Compose Error.	Firmware is not able compose the configured TPDO.	Configured TPDO will not be sent. The TPDO index value which is having error will be updated in status register.	Verify configured index value, index value should be within supported range and with read access. Also corrupt TPDO configuration can cause this error, in such case reload the CANopen configuration.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
13	TPDO Invalid Object Index.	Configured TPDO object index value is out of range or not supported by the firmware.	Configured TPDO will not be sent. The TPDO index value which is having error will be updated in status register.	Verify configured index value, index value should be within supported range.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
14	TPDO Mapped Object Count Error.	TPDO is configured without any objects.	Configured TPDO will not be sent. The TPDO index value which is having error will be updated in status register.	Verify TPDO object mapping and configure as per the requirement.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Mas-ter	Sla-ve
15	SDO DLC Error.	Received SDO message is with invalid byte count, i.e. count is not equal to 8.	SDO request or response will not be processed.	Verify SDO message generated by the node, if byte count is not equal to 8 then the node is not valid CANopen device.	This error will not trigger EMCY Object.	✓	✓
16	NMT DLC Error.	Received NMT message is with invalid byte count, i.e. count is not equal to 2.	NMT request will not be processed.	Verify NMT message generated by the node, if byte count is not equal to 2 then the node is not valid CANopen device.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	X	✓
17	Invalid Status Register Address.	Status register address is not configured or address is invalid.	CANopen node status will not be available.	Configure valid status register address.	This error will not trigger EMCY Object.	✓	✓
18	Time Out for Node Guard message from Master.	Node guard message request from Master is not received within configured time. It is also called as 'Node Life Time Error'.	Slave will set error, but continues with normal operation.	Verify node guard time configured in Master Configuration.	This error will trigger valid EMCY Object with status register value in 'Manufactur-er Specific Error Field' of message.	X	✓
19	Consumer heartbeat time expired.	Node is configured for Heartbeat message consumption and heartbeat message from producer is not received within configured consume time.	Node will set error, but continue with normal operation. But in case of Master node action to be taken can be configured.	Verify Heartbeat consume time configured in the node, it should be greater than the producer time. Check whether producer node is configured for Heartbeat message production as required interval.	This error will trigger valid EMCY Object with status register value in 'Manufactur-er Specific Error Field' of message.	✓	✓

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Master	Slave
20	Slave Error.	One of configured non mandatory slave is failed.	Master will set error, but continue with normal operation. If all slaves in the network are failed then master node doesn't allow NMT state transition. Master node can be configured to reinitialize boot up process of slave node on error.	Check CAN cabling, Error control protocol configuration in slave and master node and power status of slave node etc.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	X
21	Mandatory slave Error.	One of configured mandatory slave is failed.	Master will set error and try to reconfigure entire network or stop entire network based upon user configuration	Check CAN cabling, Error control protocol configuration in slave and master node and power status of slave node etc.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	X
22	CAN Bus Overrun.	Number of CAN messages received per second is more than the limit of CAN hardware and firmware.	CANopen communication is not guaranteed.	Check the CAN bus load, it should be around 80%. Also check CAN cable wiring and terminating resistor.	This error will trigger valid EMCY Object with status register value in 'Manufacturer Specific Error Field' of message.	✓	✓
23	CAN Bus Off Error.	One of CAN controller error state entered when it detects more than 256 CAN errors.	No CANopen communication	Check for proper terminating resistor and CAN wiring. Requires power reset to start new CANopen communication	N/A	✓	✓

Bit	Error	Reason	Indication	Remedy	EMCY Object	Applicable	
						Mas-ter	Sla-ve
24	CAN Bus Passive Error.	One of CAN controller error state entered when it detects more than 127 CAN errors, but less than 256. Unplugging CAN network cable can cause this error.	No CANopen communicat-ion	Check for proper terminating resistor , CAN wiring and firm connection CAN connector to device	N/A	✓	✓
25 - 32	NMT Sate	The 8 bit displays CANopen NMT state of device. It can have following different values. 127(Decimal) – 0x7F (Hex) - Preoperational State 005(Decimal) – 0x05 (Hex) - Operational State 004(Decimal) – 0x04 (Hex) - Stop State	N/A	N/A	N/A	N/A	
33-48	Failed TPDO array Index	Failed TPDO array Index (Updated only in case of any TPDO errors and array index will start with value 0).	N/A	N/A	N/A	N/A	
49-64	Failed RPDO array Index	Failed RPDO array Index (Updated only in case of any RPDO errors and array index will start with value 0).	N/A	N/A	N/A	N/A	

7.2 Status of Single Slave node

Master Node will have additional status of each Slave Node following 64bit long Status register. One 16bit register will be dedicated to indicate status of single slave node.

7.2.1 Format

Bit 0 to 7: Error Code
Bit 8 to 15: Node-ID

7.2.2 Error codes

Error Code Values	Error Description
0	No error.
1	The slave no longer exists in the Network list
2	No response on access to Actual Device Type (object 1000h) received
3	Actual Device Type (object 1000h) of the slave node did not match with the expected Device Type Identification in object 1F84h
4	Actual Vendor ID (object 1018h) of the slave node did not match with the expected Vendor ID
5	Slave node did not respond with its state during Check node state - process. Slave is a heartbeat producer
6	Slave node did not respond with its state during Check node state - process. Slave is a Node Guard slave (NMT slave)
7	It was requested to verify the application software version, but the expected version date and time values were not configured
8	Actual application software version Date or Time did not match with the expected date and time values. Automatic software update was not allowed
9	Actual application software version Date or Time did not match with the expected date and time values and automatic software update failed
10	Automatic configuration download failed
11	The slave node did not send its heartbeat message during Start Error Control Service although it was reported to be a heartbeat producer
12	Slave was initially operational. (CANopen manager may resume operation with other nodes)
13	Actual Product Code (object 1018h) of the slave node did not match with the expected Product Code
14	Actual Revision Number (object 1018h) of the slave node did not match with the expected Revision Number
15	Actual Serial Number (object 1018h) of the slave node did not match with the expected Serial Number in object
244	Error Configuring Error Control Protocol (Either Node Guard or Heart Beat) parameters
245	Error Configuring SYNC Protocol parameters
246	Error Configuring Time-Stamp Protocol parameters
247	Error Configuring Emergency (EMCY) protocol parameters
248	Error Configuring RPDO communication parameters
249	Error Configuring RPDO mapping parameters
250	Error Configuring TPDO communication parameters
251	Error Configuring TPDO Mapping parameters
252	Error Configuring SDO protocol parameters.
253	Invalid NMT state (Mismatch between Master NMT state and that slave NMT state)
254	Received Emergency Object
255	Unknown Error/ Master Reconfiguration is Active

Glossary:

CAN: Controller Area Network is a standardized serial bus system.

COB (Communication Object): A unit of transportation in a CAN network. Data must be sent across a CAN Network inside a COB. There are 2048 different COB's in a CAN network. A COB can contain at most 8 bytes of data.

COB-ID: Each COB is uniquely identified in a CAN network by a number called the COB Identifier (COB-ID). The COB-ID determines the priority of that COB for the MAC sub-layer.

MAC (Medium Access Control): One of the sub-layers of the Data Link Layer in the CAN Reference Model that controls who gets access to the medium to send a message.

Device Profile: A device profile defines the device-specific communication services including the configuration services in all details.

EDS (Electronic Data Sheet): Electronic data sheets describe the functionality of a device in a standardized manner. CANopen and DeviceNet use different EDS formats.

NMT (Network Management): One of the service elements of the application layer in the CAN Reference Model. The NMT serves to configure, initialize, and handle errors in a CAN network

Node ID: The Node-ID of the NMT Slave has to be assigned uniquely.

PDO (Process Data Object): Process Data Object protocol is used to process real time data among various nodes. It can transfer up to 8 bytes (64bits) data in one PDO either from or to the device

RPDO (Receive PDO): RPDO is used for sending data to a device.

SDO (Service Data Object): The SDO protocol is used to set and read values from the object directory of a remote device. The device whose object directory is accessed is the SDO server and the device accessing the remote device is the SDO client.

Sub Index: 8-bit sub-address to access the sub-objects of arrays and records.

SYNC (Synchronization Object): The Sync Object is broadcast periodically by the Sync Producer. The Sync-Producer provides the synchronization-signal for the Sync-Consumer. When the Sync-Consumer receives the signal they start carrying out their synchronous tasks.

TPDO (Transmit PDO): TPDO is used for reading data from a device.