PM-311x CPS Series
Compact Power Meter CANopen Application
User's Manual

Warranty

　　All products manufactured by ICP DAS are under warranty regarding defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

　　ICP DAS assumes no liability for damages resulting from the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

　　Copyright @2013 by ICP DAS. All rights are reserved.

Trademark

　　The names used for identification only may be registered trademarks of their respective companies.

Table of Contents

1. Introduction

1.1   Overview

CANopen, a kind of communication protocols, is an intelligent field bus (CAN bus). It has been developed as a standard embedded network with a high flexible configuration. It provides a standard communication protocol transmitting real-time data in PDO (**P**rocess **D**ata **O**bjects), configuration data in SDO (**S**ervice **D**ata **O**bjects), and network management data (NMT message, and Error Control), even supports the special functions (Time Stamp, Sync message, and Emergency message).These features can improve the network reliability and transmission efficiency. Nowadays, CANopen is used on many applications and in specific fields, such as medical equipment, off-road vehicles, maritime electronics, public transportation, automation and so on. The PM-311x-CPS power meter is built-in the CANopen interface. Therefore, users can easily apply in any CANopen applications via the power meter.

2. DIP Switch of PM-311x-CPS

2.1 The Node-ID of CANopen Application

Node ID   **Baud rate**
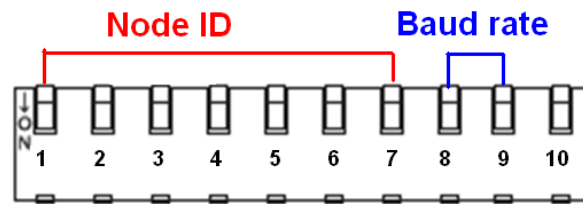
There are hardware Node-ID and software Node-ID in the PM-311x-CPS. The hardware Node-ID can be represented by DIP-Switch binary value with the first 7 pins. The relationship between the Node-ID and the DIP-Switch status is shown below.

Node-ID and Dip-Switch 1 ~ 7 digital.

Node ID	Pin 1	Pin2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7
1	OFF	OFF	OFF	OFF	OFF	OFF	OFF
2	ON	OFF	OFF	OFF	OFF	OFF	OFF
3	OFF	ON	OFF	OFF	OFF	OFF	OFF
4	ON	ON	OFF	OFF	OFF	OFF	OFF
…							
126	ON	OFF	ON	ON	ON	ON	ON
127	OFF	ON	ON	ON	ON	ON	ON
128	ON	ON	ON	ON	ON	ON	ON

Table 2-1

2.2   The baud rate of CANopen Application
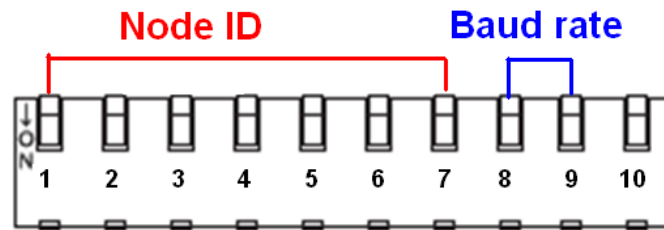
The DIP-Switch from pin 8 to pin 9 represents the CAN bus baud rate of the PM-311x-CPS. The mapping table is shown as Table 2-2. The default baud rate value is 125K, and the corresponding DIP-Switch value from pin 8 to pin 9 is (OFF) (OFF).

CAN baud rate and Dip-Switch8~ 9.

CAN baud-rate	Pin 8	Pin9
125kbps	OFF	OFF
250kbps	ON	OFF
500kbps	OFF	OFF
1000kbps	ON	ON

Table 2-2

2.3   The LED State

RUN LED of CANopen" is an indicator LED of CAN bus in the PM-311x-CPS. It shows whether the CAN communication is normal or incorrect. The following figure shows the LED.
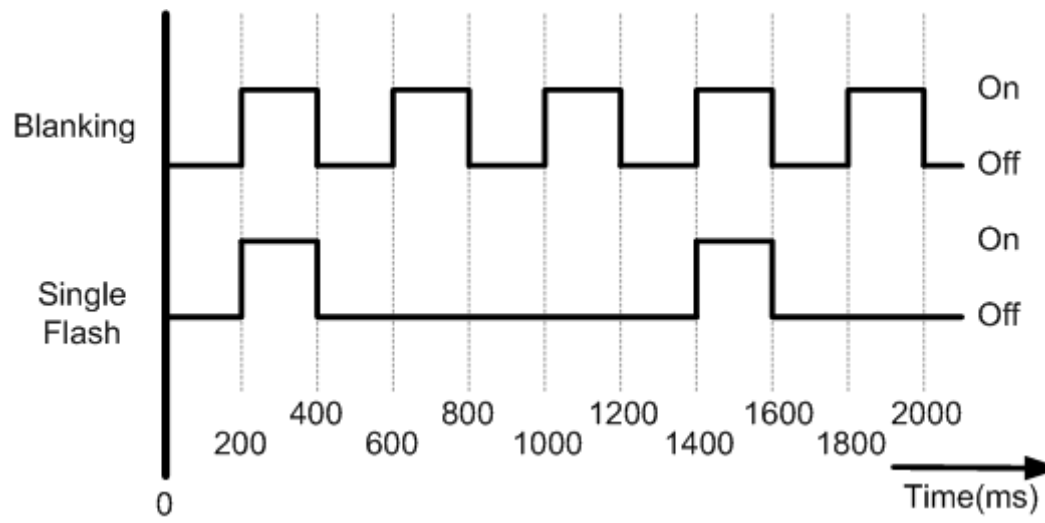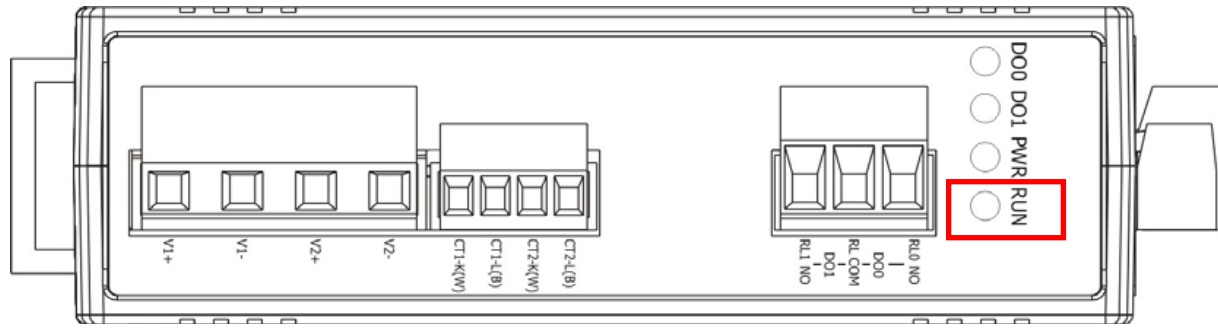
No.	Signal	State	Description
1	No Light	Non-operation	Malfunction or Power Supply/Connection not ready
2	Single Flash	Stopped	The device is in Stopped state
3	Blinking	Pre-operation	The device is in the pre-operation state
4	Continuing Light	Operation	The device is in the operational state

Table 2-3

3.   CANopen Protocol

The CANopen is a kind of network protocols evolving from the CAN bus, used on car control system in early days, and has been greatly used in various applications, such as vehicles, industrial machines, building automation, medical devices, maritime applications, restaurant appliances, laboratory equipment & research.

3.1   CANopen Introduction

CANopen provides not only the broadcasting function but also the peer-to-peer data exchange function between every CANopen node. The network management function instructed in the CANopen simplifies the program design. In addition, users can also implement and diagnose the CANopen network, including network start-up, and error management by standard mechanisms (CANopen device), i.e. the CANopen device can effectively access the I/O values and detect node states of other devices in the same network. Generally, a CANopen device can be modeled into three parts.

- Communication
- Object Dictionary
- Application program

The functions and general concepts for each part are shown as follows.

Bus System   Process
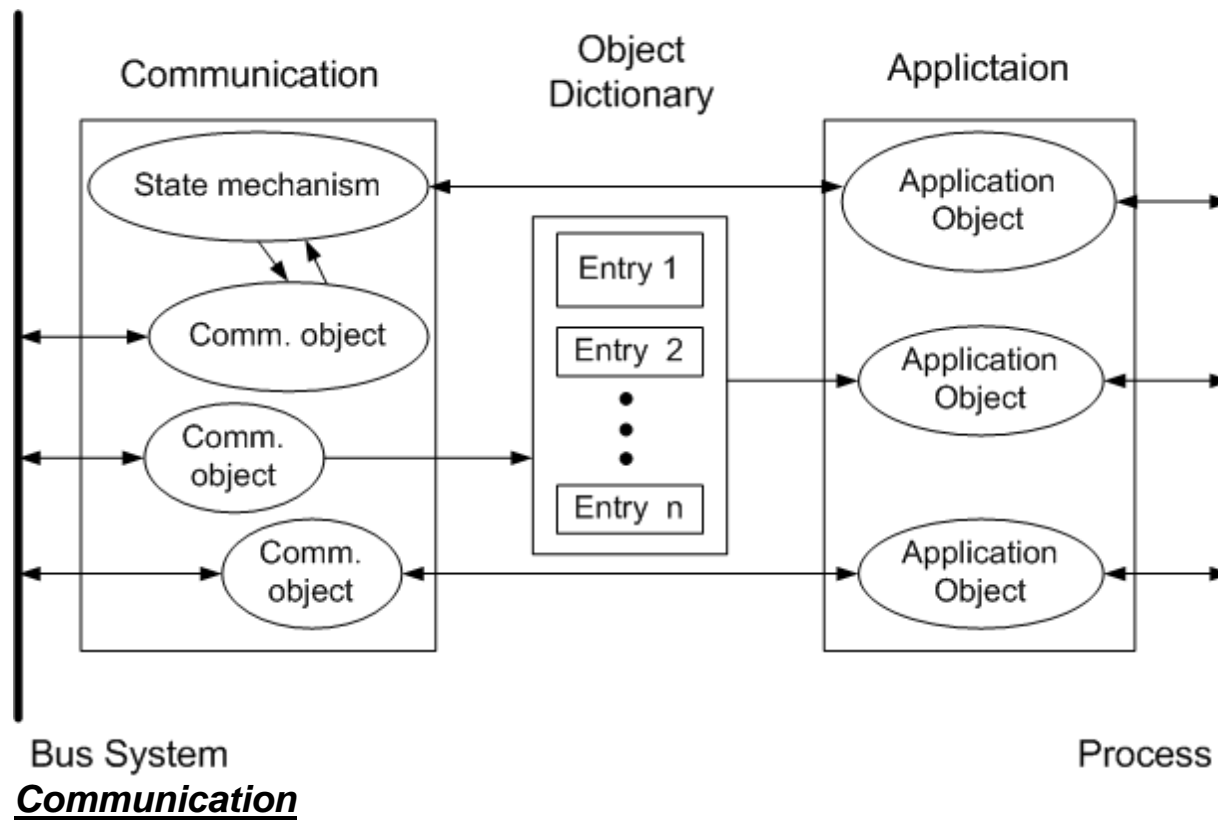
Communication

The communication part provides several communication objects and appropriate functionalities to transmit CANopen messages via the network structure. These objects include PDO (Process Data Object), SDO (Service Data Object), NMT (Network Management Objects), SYNC (Synchronous Objects)…etc. Each communication object has its relative communication model and functionality. For example, the communication objects for accessing the device object dictionary is SDO, using the Client/Server structure as its communication model (section 3.2). Real-time data or I/O values can be accessed quickly without any protocol by means of PDO communication objects. The PDO's communication model follows the Producer/Consumer structure. It is also named the Push/Pull model (section 3.3). NMT communication objects are used for controlling and supervising the state of the nodes in the CANopen network, and it follows a Master/Slave structure (section 3.4). No matter which kind of communication object is used, the transmitted message will comply with the data frame defined in the CAN 2.0A spec. Generally, it looks like the following table.

ID	RTR	Data Length	8-byte Data

11-bit data is limited in the ID field. It is useful in the arbitration mechanism. The RTR, limited in 1-bit data, is used for remote-transmitting requests as the value is set to 1. The data length, limited in 4-bit data, shows the valid data number stored in the 8-byte data field. The last field, 8-byte data, is applied to store the message data.

In the CANopen specifications the 4-bit function code and 7-bit node ID are assumed to combine the 11-bit ID of CAN message, and named the communication object ID (COB-ID). The COB-ID structure is displayed below.

bit 10   bit0

| Function Code | Node ID |
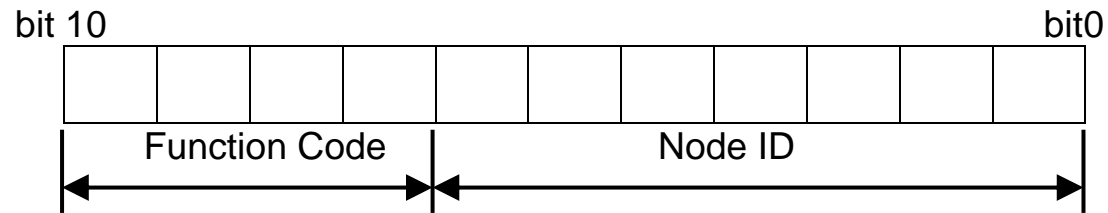
The COB-IDs are used for recognizing where the message comes from or where the message is sent to, as well deciding the priority of the message transmission around node network. According to the arbitration mechanism rule of the CAN bus, the CAN message with the lower COB-ID will get the higher priority to be transmitted. In the CANopen specifications some COB-IDs are reversed for specific communication objects, and can't be defined arbitrarily by users. The following list shows these reversed COB-IDs.

Reversed COB-ID (Hex)	Used by object
0	NMT
1	Reserved
80	SYNC
81~FF	EMERGENCY
100	TIME STAMP
101~180	reversed
581~5FF	Default Transmit-PDO
601~67F	Default Receive-PDO
6E0	reversed
701~77F	NMT Error Control
780~7FF	reversed

In addition, the other COB-IDs shown in the following table can be used if necessary.

(Bit10~Bit7) (Function Code)	(Bit6~Bit0)	Communication object Name
0000	0000000	NMT
0001	0000000	SYNC
0010	Node ID	TIME STAMP

0001	Node ID	EMERGENCY
0011/0101/0111/1001	Node ID	TxPDO1/2/3/4
0100/0110/1000/1010	Node ID	RxPDO1/2/3/4
1011	Node ID	SDO for transmission (TxSDO)
1100	Node ID	SDO for reception (RxSDO)
1110	Node ID	NMT Error Control

Note: For the PM-311x-CPS, all communication objects are supported except the TIME STAMP and RxPDO.

Object Dictionary

The object dictionary collects a lot of important information which can affect device's reaction, such as the data accessing through I/O channels, the communication values and the network states. Essentially, the object dictionary consists of a group of entry objects, and these entries can be accessed via the node network in a pre-defined method. Each object entry within the object dictionary has its own function, for example communication parameters, device profile, data type (ex. 8-bit Integer, 8-bit unsigned…), and access type (read only, write only …). All of them are addressed in a 16-bit index and an 8-bit sub-index. The overall profile of the standard object dictionary is shown below.

Index	Object
0x0000	Reserved
0x0001 – 0x001F	Static Data Types
0x0020 – 0x003F	Complex Data Types
0x0040 – 0x005F	Manufacturer Specific Complex Data Types
0x0060 – 0x007F	Device Profile Specific Static Data Types
0x0080 – 0x009F	Device Profile Specific Complex Data Types
0x00A0 – 0x0FFF	Reserved for further use
0x1000 – 0x1FFF	Communication Profile Area
0x2000 – 0x5FFF	Manufacturer Specific Profile Area
0x6000 – 0x9FFF	Standardized Device Profile Area
0xA000 – 0xBFFF	Standardized Interface Profile Area
0xC000 – 0xFFFF	Reserved for further use

3.2   SDO Introduction

In order to access the entries in a device object dictionary, service data objects (SDOs) are provided. By means of the SDO communication method, a peer-to-peer communication bridge between two devices is established, and its transmission follows the client-server relationship. The general concept is shown in the figure below.
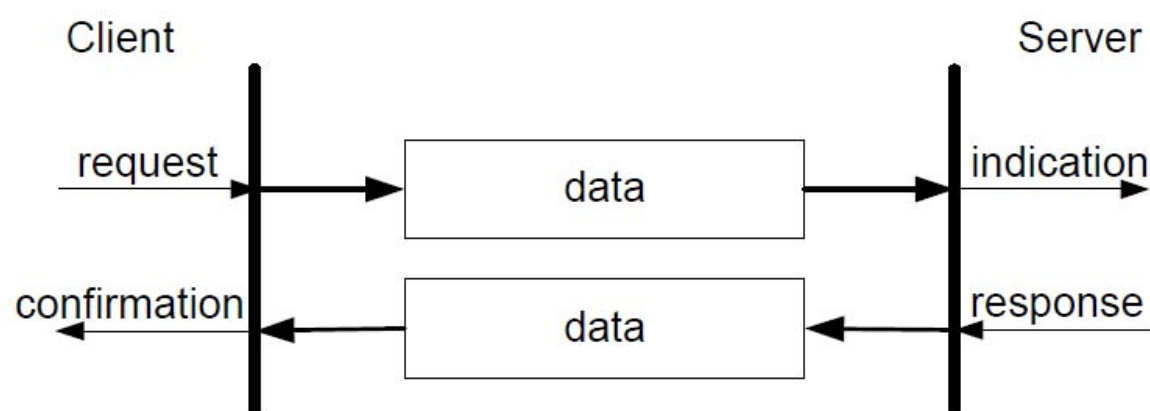
The SDO has two kinds of the COB-IDs, RxSDOs and TxSDOs. They can be viewed in the CANopen device. For example, users send a SDO message to the PM-311x-CPS by using RxSDO. On the contrary, the device, PM-311x-CPS, transmits a SDO message by using TxSDOS.

Before the SDO has been used, only the client can take the active requirement for a SDO transmission. When the SDO client starts to transmit a SDO, it is necessary to choose a proper protocol.

If the SDO client has to get the information from the device object dictionary and from the SDO server, the segment upload protocol or block upload protocol will be applied.

It is worth to be mentioned, the front protocol is used for transmitting fewer data; the latter protocol is used for transmitting larger data. Both the segment download protocol and block download protocol will work when the SDO client wants to modify the object dictionary to the SDO server. The differences between the segment download protocol and the block download protocol are similar to the differences between the segment upload protocol and the block upload protocol. Because of the different access types in the object dictionary, not all accessing action of the object dictionary via the SDO transmission is allowed. If the SDO client trends to modify the entries of the SDO server object dictionary which uses the read-only access type, the abort SDO transfer protocol will be given, and the SDO transmission will also be stopped.

The PM-311x-CPS only supports the SDO server. Therefore, it can be passive and wait for requests from clients. The general concept figure of the upload and download protocol with the PM-311x-CPS is shown as follows.

SDO Client   SDO Server
                                              (PM-311x-CPS)

Request the data of the object dictionary →

Response the data of the object dictionary
or the abort SDO message ←

Upload protocol

SDO Client   SDO Server
                                              (PM-311x-CPS)

Write the data of the object dictionary →

Response access OK
or the abort SDO message ←

Download protocol
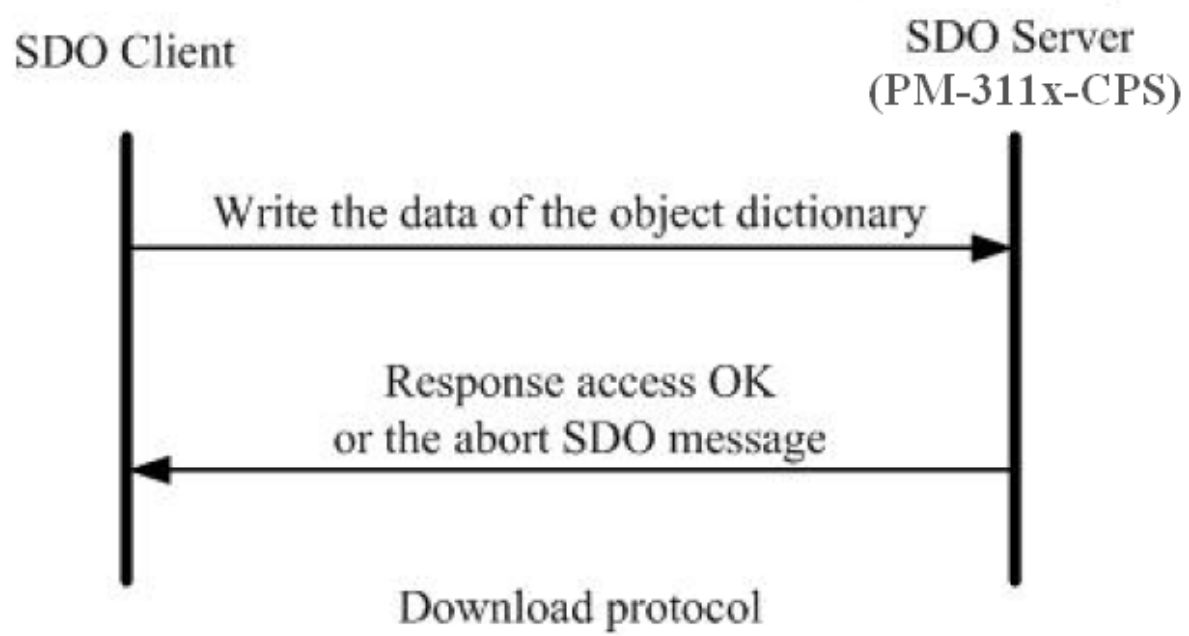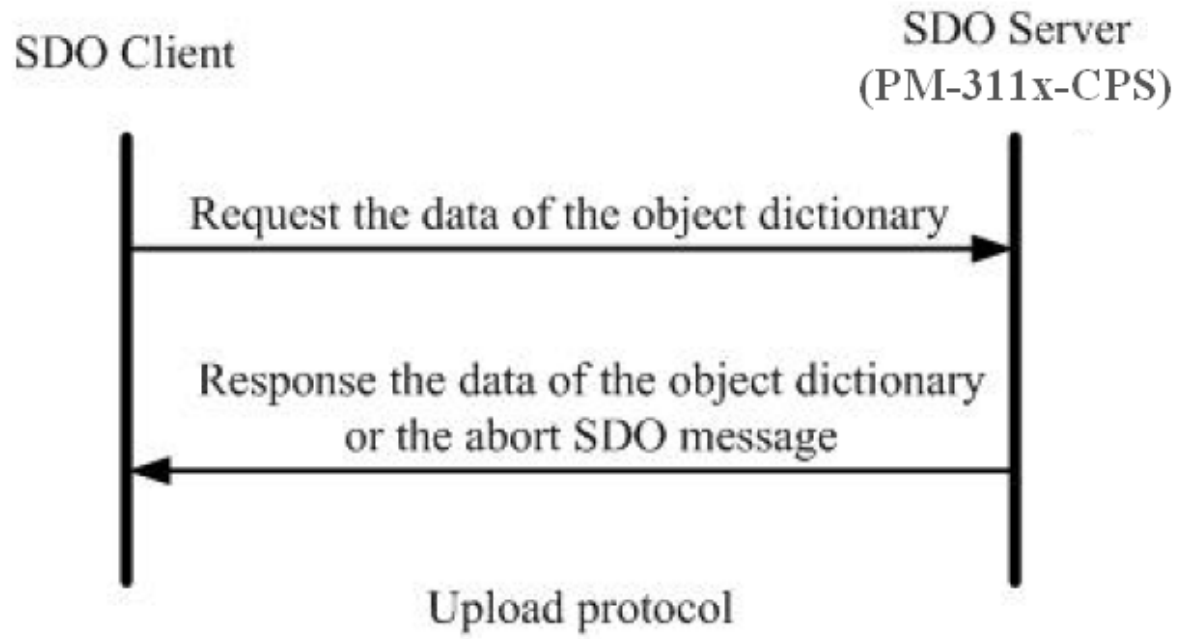
3.3 PDO Introduction

Based on the transmission data format of the CAN bus, the PDO can transmit eight bytes of process data at one time. Because of the PDO messages without overheads, it is more efficient than other communication objects within CANopen and therefore used for real-time data transfer, such as DI, DO, AI, AO, etc.

Communication Modes for the PDO

PDO reception or transmission is implemented via the producer/consumer communication model (also called the push/pull model). When starting to communicate in the PDO push mode, it needs one CANopen device to play the role of PDO producer, and non device or more than one device to play the role of PDO consumer.

The PDO producer sends out the PDO message after it reached the CAN bus arbitration. Afterwards, each PDO consumer will receive this PDO message respectively, and then message is processed by each device to check whether it is needed or not (be dropped). In the PDO pull mode, one of the PDO consumers needs to send out a remote transmit request to the PDO producer. According to this remote request message, the PDO producer responds the corresponding PDO message for each PDO consumer in the CAN bus. The PDO communication structure figure is shown below.
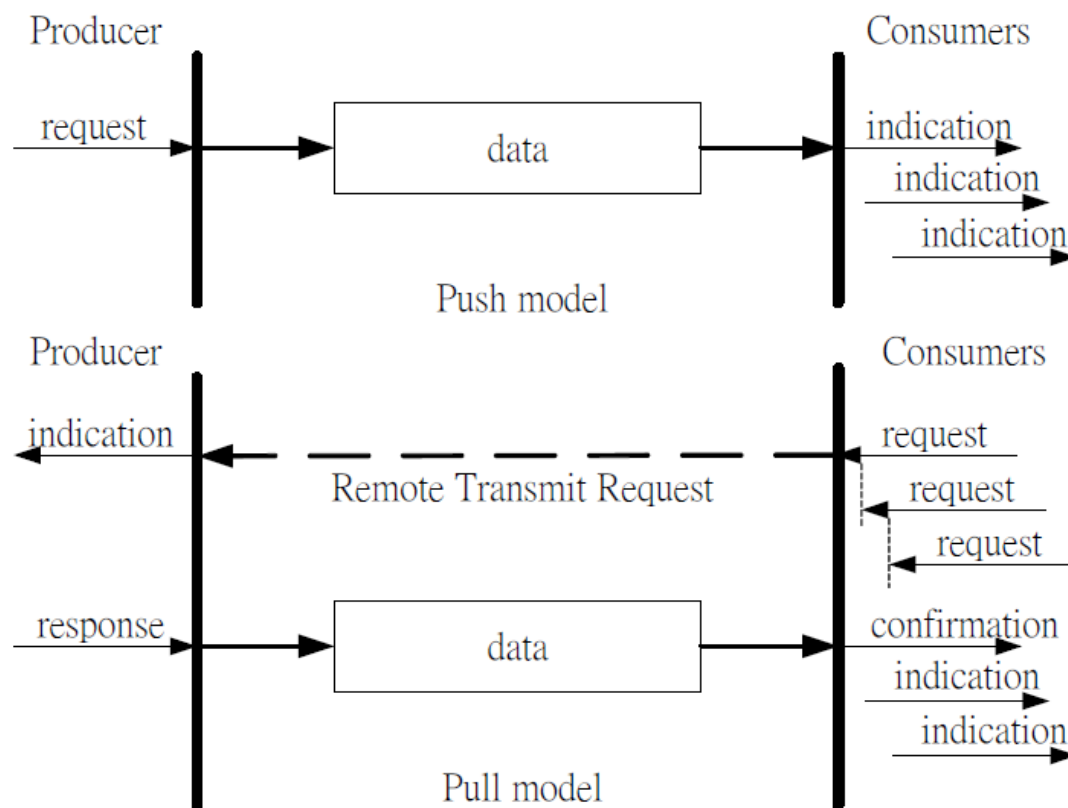
For the CANopen device, the TxPDO specializes in transmitting data, and is usually applied on DI/AI channels. The COB-ID of the PDO for receiving data is RxPDO COB-ID, and it is usually applied on DO/AO channels. Take the PM-311x-CPS as an example. The PM-311x-CPS only supports TxPDO. When some PDO consumers send remote transmit requests to the PM-311x-CPS, it must use the TxPDO COB-ID of the PM-311x-CPS because it is a PDO transmission action viewed from the PM-311x-CPS.

Trigger Modes Of PDO

For PDO producers, PDO transmission messages can be trigged by three conditions. They are the event driven, timer driven and remote request conditions. All of them are described below.

Event Driven
PDO transmission can be triggered by a specific driven event, including the following conditions. Under the cyclic synchronous transmission type, the event is driven by the expiration of the specified transmission period, synchronized by the reception of the SYNC message. Moreover, under the acyclic synchronous or asynchronous transmission type, the PDO transmission can also be triggered or driven by a device-specified event in the CANopen specification DS-401 v2.1, i.e. by following this spec, the PDO will be triggered by any change in the DI-channel states when the transmission type of this PDO is set to acyclic synchronous or asynchronous.
Note: PM-311x-CPS doesn't have the trig condition.

Timer Driven
PDO transmissions are also triggered by a specific time event, even if a specified time elapsed without occurrence of an event. For example, the PDO transmission of the PM-311x-CPS can be triggered by the event timer of the PDO communication parameters, which is set by users.

Remote Request
The PDO transmission can be triggered by receiving a remote request from any other PDO consumer with under the asynchronous or RTR setting.

PDO Transmission Types

Generally there are two kinds of PDO transmission modes, synchronous and asynchronous. For the PDO in a synchronous mode, it must be triggered by the reception of a SYNC message. The synchronous mode can be further distinguished into three kinds of transmission(s), acyclic synchronous, cyclic synchronous and RTR-only synchronous. The acyclic synchronous can be triggered by both the reception of a SYNC message and the driven event mentioned above.

Acyclic synchronous
For the TxPDO object, after receiving an object from the SYNC producer, the PM-311x-CPS will respond with a pre-defined TxPDO message to the PDO consumers. F The following figures indicate how the acyclic synchronous transmission type works on the TxPDO.

acyclic synchronous TxPDO
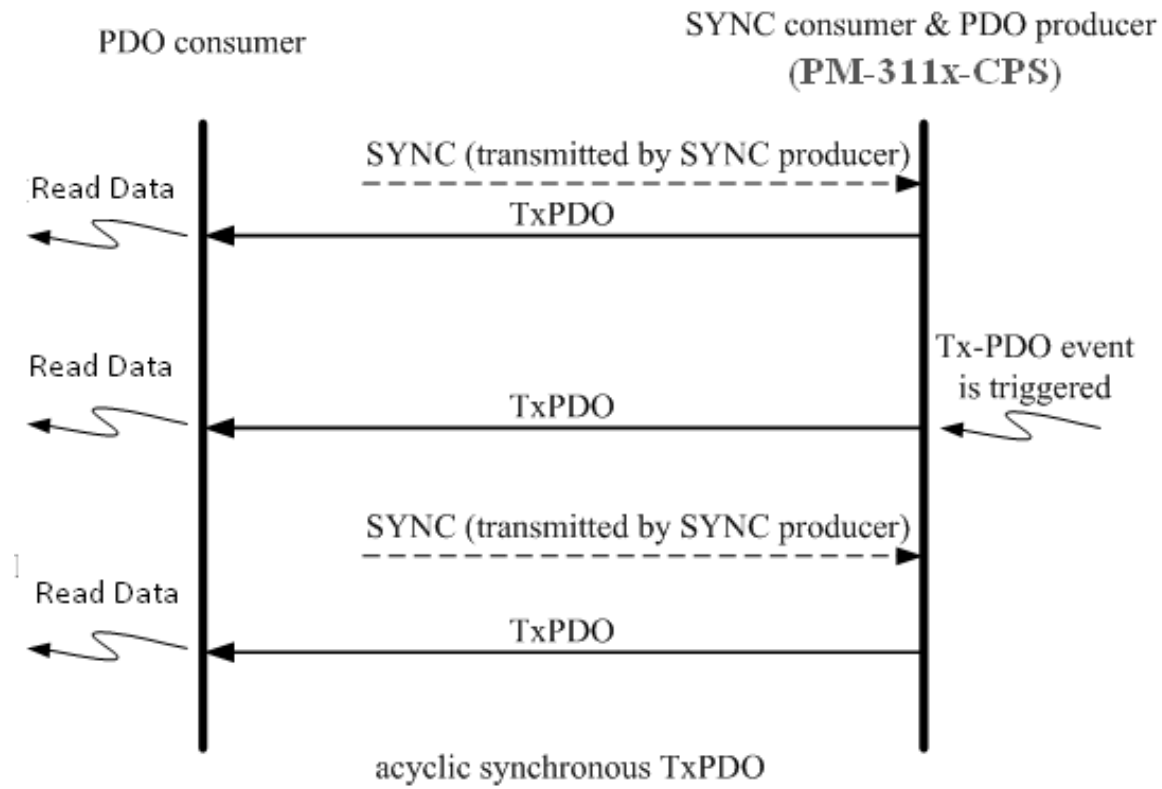
Cyclic synchronous

Inversely, the cyclic synchronous transmission mode is triggered by the reception of an expected number of SYNC objects, and the max number of expected SYNC objects can be 240. For example, if the TxPDO is set to response when receiving 3 SYNC objects, the PM-311x-CPS will feed back the TxPDO object according to the set. The concept is shown in the figure below.

cyclic synchronous TxPDO
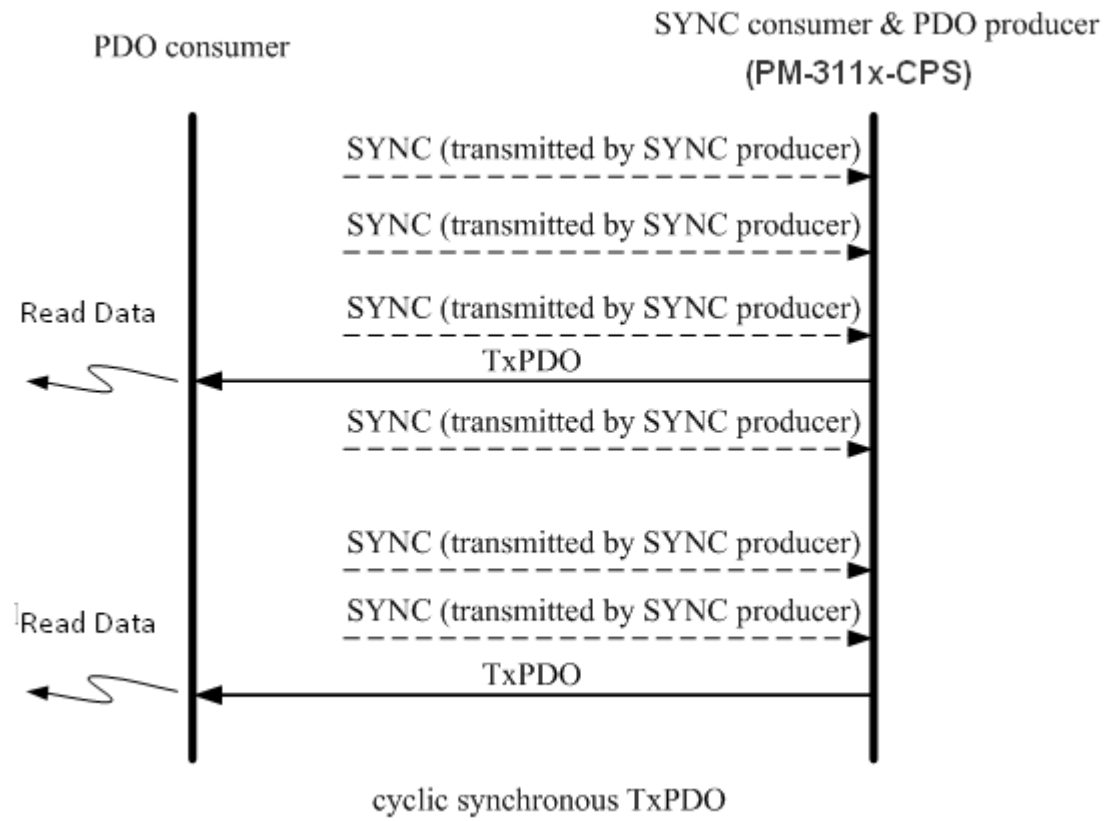
RTR-only synchronous

The RTR-only synchronous mode is activated when receiving a remote-transmit-request message, i.e. SYNC objects. This transmission type is only useful for TxPDO. In this situation, the consumer will update the DI/AI value when receiving any SYNC object. And, if the RTR object is received, the producer will respond to the TxPDO object. The following figure shows the mechanism of this transmission type.
Note: PM-311x-CPS doesn't support the type.

RTR-only synchronous TxPDO
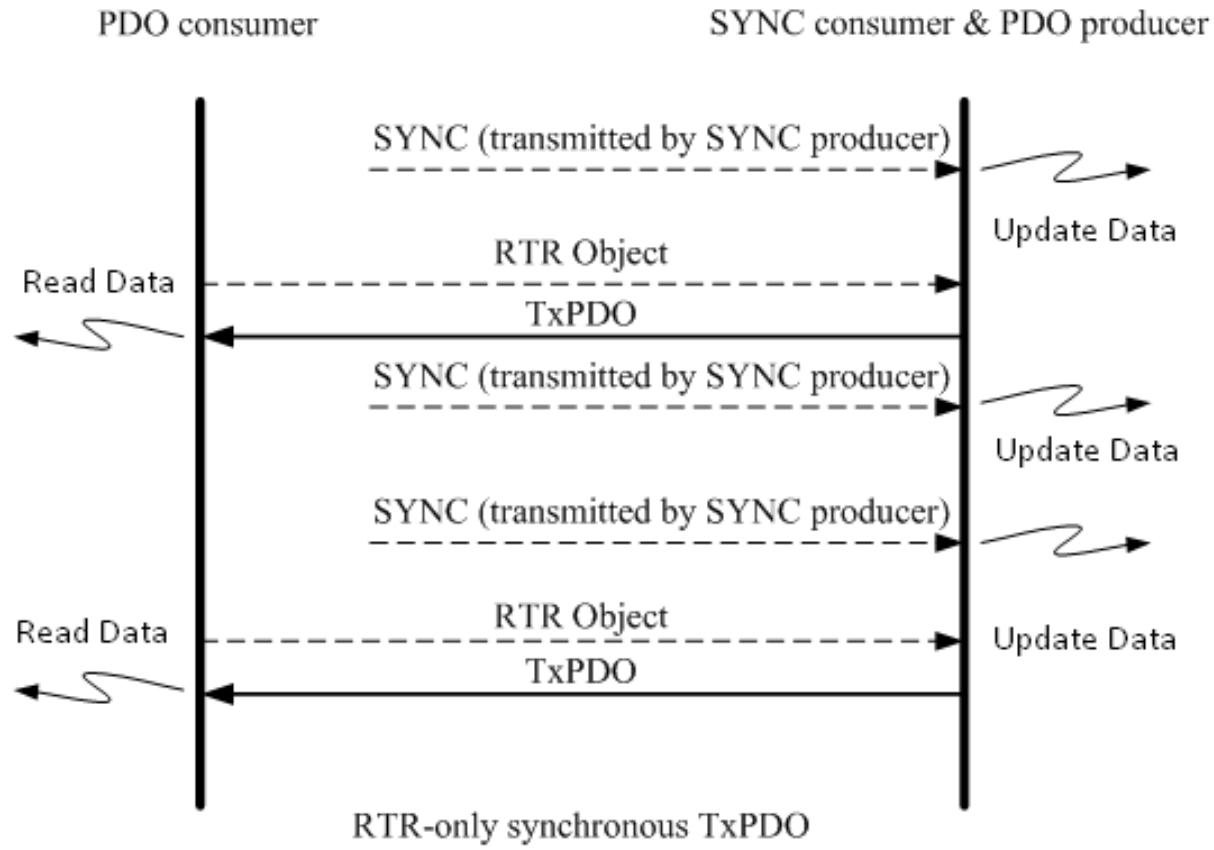
RTR-only asynchronous

The asynchronous mode is independent of the SYNC object. This mode can also be divided into two parts. There are RTR-only asynchronous transmission type and asynchronous transmission type. The RTR-only transmission type is only for supporting TxPDO transmissions, only triggered by receiving the RTR object from the PDO consumer. This action is depicted below.

RTR-only asynchronous TxPDO
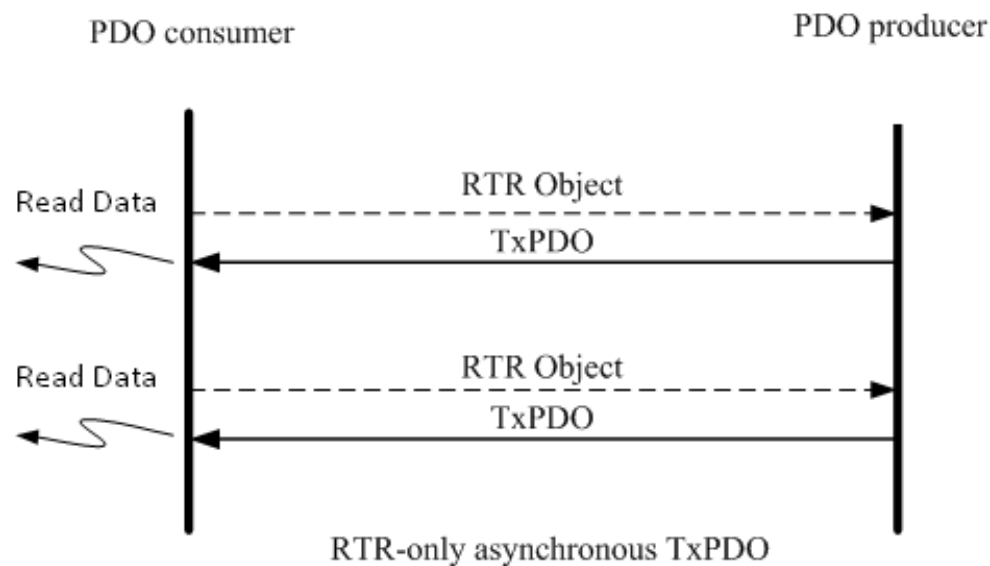
Asynchronous

The other part is the asynchronous transmission type. Under this type, the TxPDO message can be triggered by receiving the RTR object and the device-specified event mentioned in the event driven paragraph. Furthermore, the DO/AO channels can act directly by receiving the RxPDO object.
Note: PM-311x-CPS doesn't support the type.

Inhibit Time

Because of the arbitration mechanism of the CAN bus, the CANopen communication object ID in small size has a higher transmission priority than the bigger one. For example, there are two nodes on the CAN bus, the one needs to transmit the CAN message with the COB-ID 0x181, and the other has to transmit the message with COB-ID 0x182. When these two nodes transmit the CAN message to the CAN bus simultaneously, only the message containing COB-ID 0x181 can be successfully sent to the CAN bus because of the higher transmission priority. So the message with COB-ID 0x182 will be held to transmit until the message with COB-ID 0x181 is successfully transmitted. This arbitration mechanism can guarantee the successful transmission for one node when a transmission conflict occurs.

However, if the message with COB-ID 0x181 is continually transmitted, the message with COB-ID 0x182 will be postponed to be

transmitted. In order to avoid the occupation of the transmission privilege by the message with the lower COB-ID, the inhibit time parameters for each of the PDO objects are supported to define a minimum time interval between each PDO message transmission, which has a multiple of 100us. During this time interval, the PDO message will be inhibited from transmission.

Event Timer

This parameter setting on the event timer is only used for TxPDO. If the parameter of the event timer is not equal to 0 under the transmission type in asynchronous mode, the expiration of this time value can be just considered to be an event. This event will cause the TxPDO message transmission. The event timer parameter is defined as a multiple of 1ms.

PDO Mapping Objects

The PDO mapping objects are provided to the interface which is for PDO messages and real I/O data in the CANopen device. They define the meanings for each byte in the PDO message, and may be changed by using a SDO message. All of the PDO mapping objects are arranged in the Communication Profile Area. In the CANopen spec (see DS 401), RxPDO and TxPDO default mapping objects will specify something as follows:

- There shall be up to 4 TxPDO mapping objects and up to 4 RxPDO mapping objects with default mappings.
- The 1st RxPDO and TxPDO mapping objects are used for digital outputs and inputs to each other.
- The 2nd, 3rd, and 4th RxPDO and TxPDO mapping objects are respectively assigned to record the value of analog outputs and inputs.

Before applying the PDO communications, the PDO producer and the PDO consumers must have mutual PDO mapping information. On the one hand, the PDO producers need PDO mapping information to decide how to assign the expected practical I/O data to PDO messages. Besides, PDO consumers need the PDO mapping information to recognize each byte of received PDO message, i.e. when a PDO producer transmits a PDO object to PDO consumers, the consumers will contrast this PDO message with PDO mapping entries, previously obtained from the PDO producer, and then interpret the meanings of these values from the received PDO object.

3.4   NMT Introduction

The Network Management (NMT) follows the node-oriented structure

and the master-slave relationship. In the same CAN bus network, only one CANopen device is allowed to execute the function of NMT master. Each CANopen node is regarded as a unique NMT slave identified by its node ID from 1 to 127.

The NMT service supplies two protocols, the module control protocol and the error control protocol. Through the module control protocol, the nodes can be controlled to several kinds of status, such as installing, pre-operational, operational, and stopped. According to the NMT slave can present in different statuses, it has different privileges to carry out the communication protocol. Through the error control protocol, users are able to detect the remote error in the network in order to confirm whether the node still works or not.

3.4.1   Module Control Protocols

Before introducing the modules control protocols, the architecture of the NMT state mechanism needs to be mentioned. The diagram shows the process and the relationships among each NMT state and the mechanism.
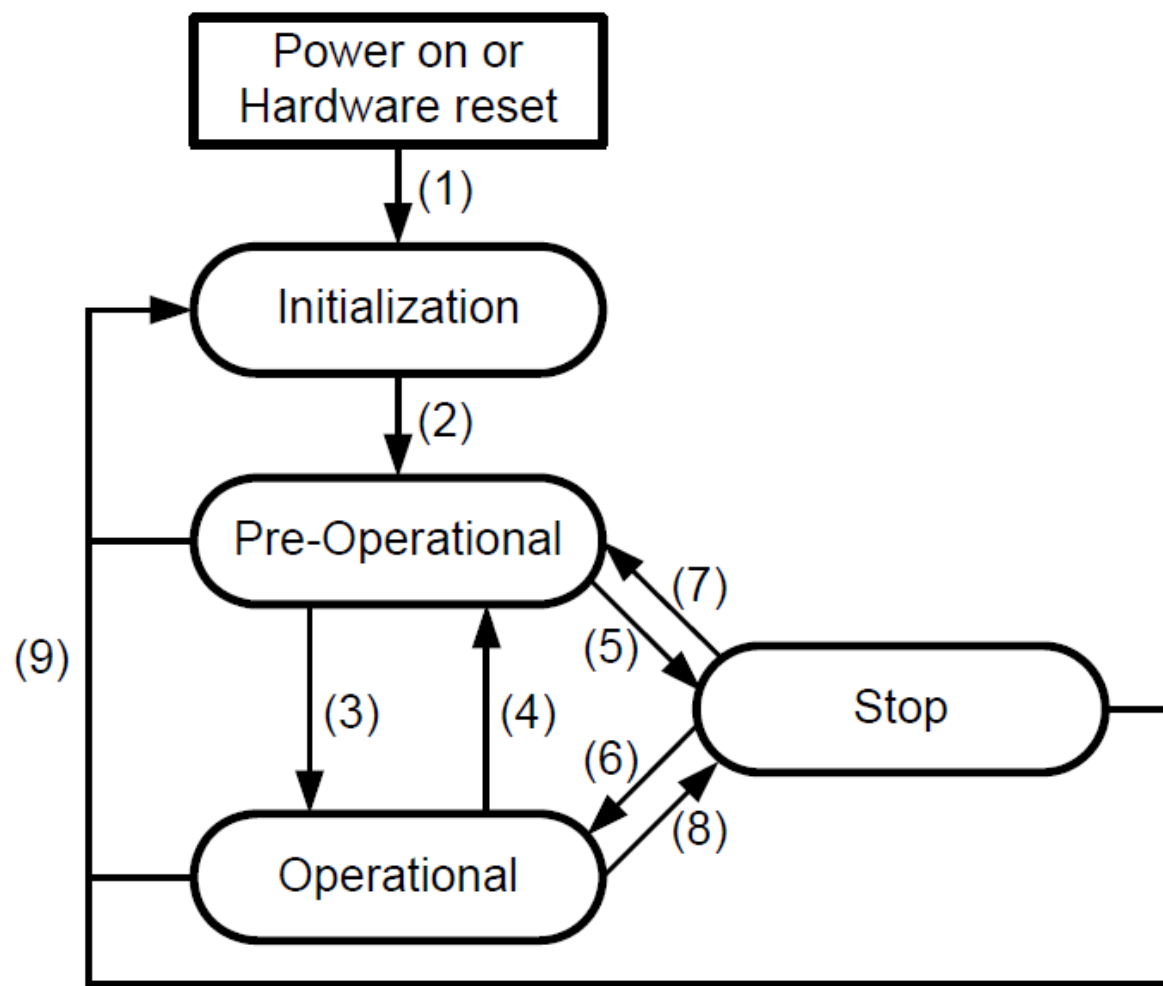
State Mechanism Diagram

(1)	Under "Power on" or "Hardware Reset", the initialization state will be loaded automatically.
(2)	As the Initialization accomplished, Pre-Operational state will be entered automatically
(3), (6)	Indication of starting remote node
(4), (7)	Indication of entering Pre-Optional State
(5), (8)	Indication of stopping remote node
(9)	Indication of the "Reset Node" or the "Reset Communication"

    Devices will directly lead to the Pre-Operational state after finishing the device initialization. Then, the nodes will be switched into different state by receiving a specific indication. By the way, each different NMT state will consider a specific communication method. For example, the PDO message can only do the transmission and receiving in the operational state. In the following table, the relationship among each NMT state and communication objects is given.

	Installing	Pre-operation	Operational	Stopped
PDO			O	
SDO		O	O	
SYNC		O	O	
Time Stamp Object		O	O	
EMCY Object		O	O	
Boot-Up Object	O			
NMT		O	O	O

3.4.2 Error Control Protocols

There are two kinds of protocols defined in the error control protocol. According to the CANopen spec, one device is not allowed to use the following error control mechanisms at the same time, Node Guarding Protocol and Heartbeat Protocol. In addition, the PM-311x-CPS provides the salve function of the Node Guarding Protocol for practical applications. Therefore, only node guarding protocols will be highlighted here, and described below.

Node Guarding Protocol

The Node Guarding Protocol follows the Master/Slave relationship. It helps users monitoring the node in the CAN bus. The communication method of node guarding protocol is defined as follows.
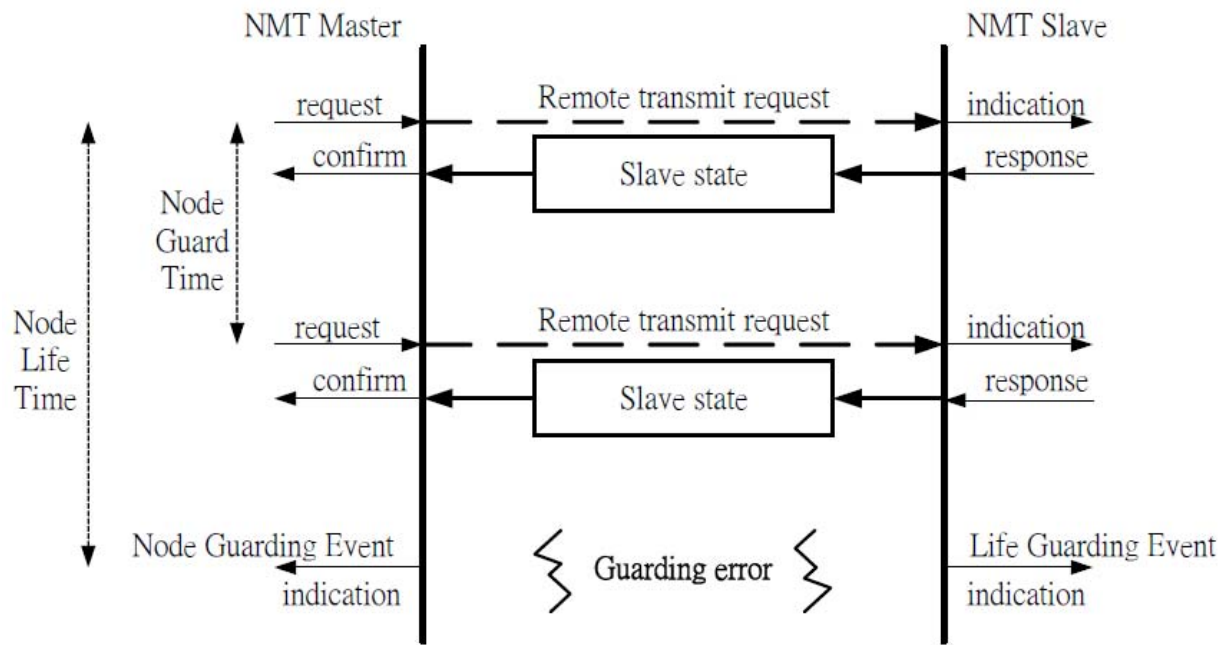
The NMT master will inspect each NMT slave at regular time intervals. This time-interval is called the node guard time, given by the "guard time * life time factor", and may be different from each NMT slave. And the response of the NMT slave contains the state of that NMT slave, which may be in a "**Stopped**", "**Operational**", or "**Pre-operational**" state. The node life time factor can also be different for each NMT slave. If the NMT slave has not been inspected during its life time, a remote node error will be given, and indicate through the "Life Guarding Event" service.

Heartbeat Protocol

The Heartbeat Protocol follows the Producer/Consumer relationship. It provides a way to help uses monitor the node in the CAN bus. The communication method of heartbeat protocol is defined as follows.
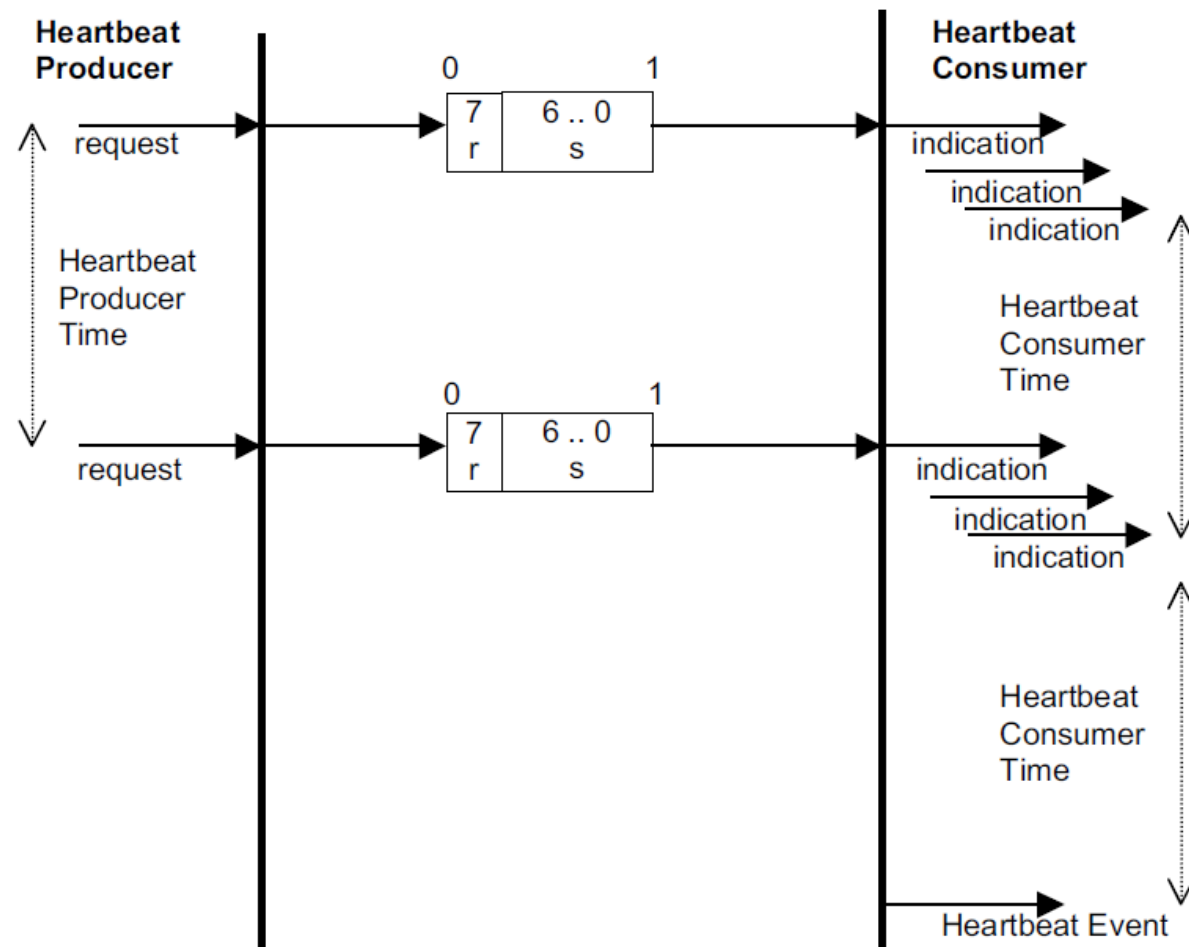
The Heartbeat Protocol defines an Error Control Service without need for remote frames. A Heartbeat Producer transmits a Heartbeat message cyclically. One or more Heartbeat Consumer receives the indication. The relationship between producer and consumer is configurable via the object dictionary. The 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.

4. CANopen Protocol Examples

4.1 SDO Communication Set

4.1.1 Upload SDO Protocol

Initiate SDO Upload Protocol

Before transferring the SDO segments, the client and server need to communicate with each other by using the initiate SDO upload protocol. Via the initiate SDO upload protocol, the SDO client will inform the SDO server what object the SDO client wants to request. As well, the initiate SDO upload protocol is permitted to transmit up to four bytes of data. Therefore, if the data length of the object, which the SDO client can read, is equal to or less than the permitted data amount, the SDO communication will be finished only by using the initial SDO upload protocol, i.e. if the data upload is less enough to be transmitted in the initiate SDO upload protocol, then the upload SDO segment protocol will not be used. The communication process of this protocol is shown as follows.

Initiate SDO Upload Protocol
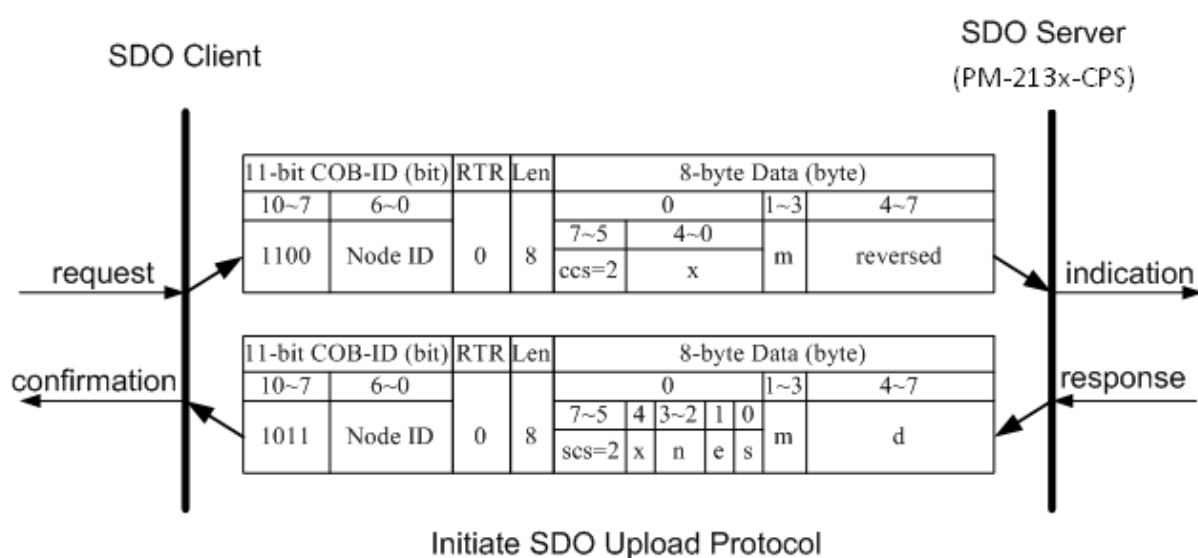
ccs   : client command specified
              2: initiate upload request
scs   : server command specified
              2: initiate upload response
n   : Only valid if **e** = 1 and **s** = 1, otherwise 0. If valid, it
              indicates the number of bytes in **d** that do not contain
              data. Bytes [8-**n**, 7] do not contain segment data.

e : transfer type
0: normal transfer
1: expedited transfer
If the **e**=1, it means that the data of the object are equal or less than 4 bytes, and only initiate SDO upload protocol is needed. If **e**=0, the upload SDO segment protocol is necessary.

s : size indicator
0: Data set size is not indicated.
1: Data set size is indicated.

m : multiplexer
It represents the index/sub-index of the data to be transfer by the SDO. The first two bytes are the index value and the last byte is the sub-index value.

d : data
e=0, **s**=0: **d** is reserved for further use.
e=0, **s**=1: **d** contains the number of bytes to be uploaded, and byte 4 contains the least significant bit, and byte 7 contains the most significant bit.
e=1, **s**=1: **d** contains the data of length 4-**n** to be uploaded, the encoding depends on the type of the data referenced by index and sub-index.
e=1, **s**=0: **d** contains unspecified number of bytes to be uploaded.

x : not used, always 0
reserved : reserved for further use , always 0

Upload SDO Segment Protocol

When the upload data length is over 4 bytes, the upload SDO segment protocol will be needed. After finishing the transmission of the initiate SDO upload protocol, the SDO client will start to upload the data. The upload SDO segment protocol will comply with the process shown below.

Upload SDO Segment Protocol
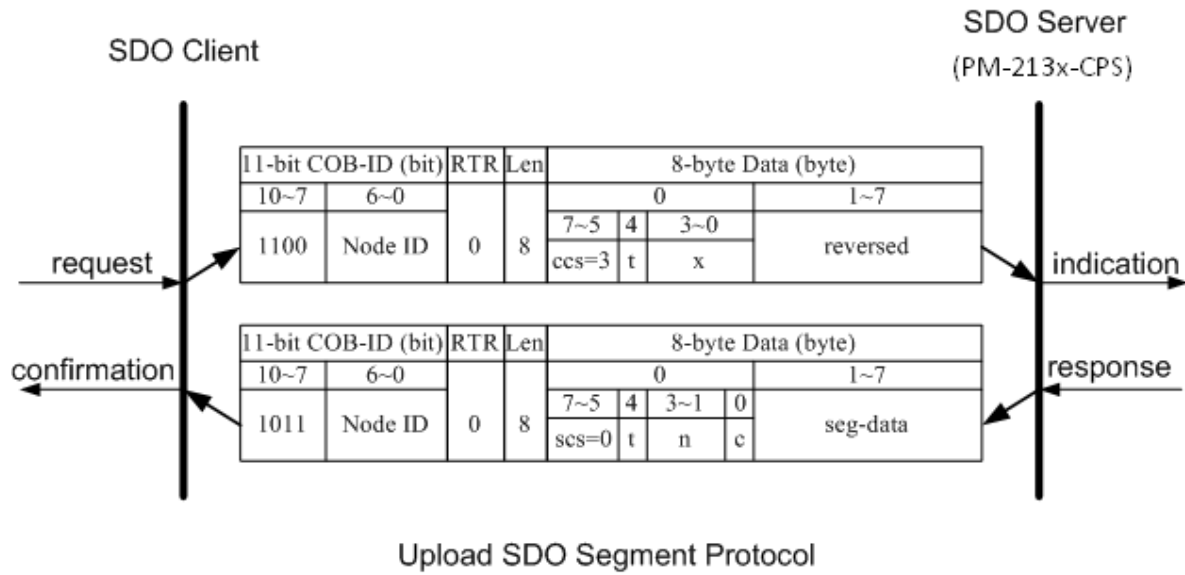
ccs	: client command specified
	3: upload segment request
scs	: server command specified
	0: upload segment response
t	: toggle bit.
	This bit must alternate for each subsequence segment that is uploaded. The first segment will have the toggle bit set to 0. The toggle bit will be equal for the request and response message.
c	: indicates whether where are still more segments to be uploaded
	0: more segments to be uploaded .
	1: no more segment to be uploaded.
seg-data	: It is at most 7 bytes of segment data to be uploaded. The encoding depends on the type of the data referenced by index and sub-index.
n	: It indicates the number of bytes in **seg-data** that do not contain segment data. Bytes [8-**n**, 7] do not contain segment data. **n** = 0 if no segment size is indicated.
x	: not used, always 0
reserved	: reserved for further use , always 0

SDO Upload Example

The practical application of the SDO upload is illustrated as below.
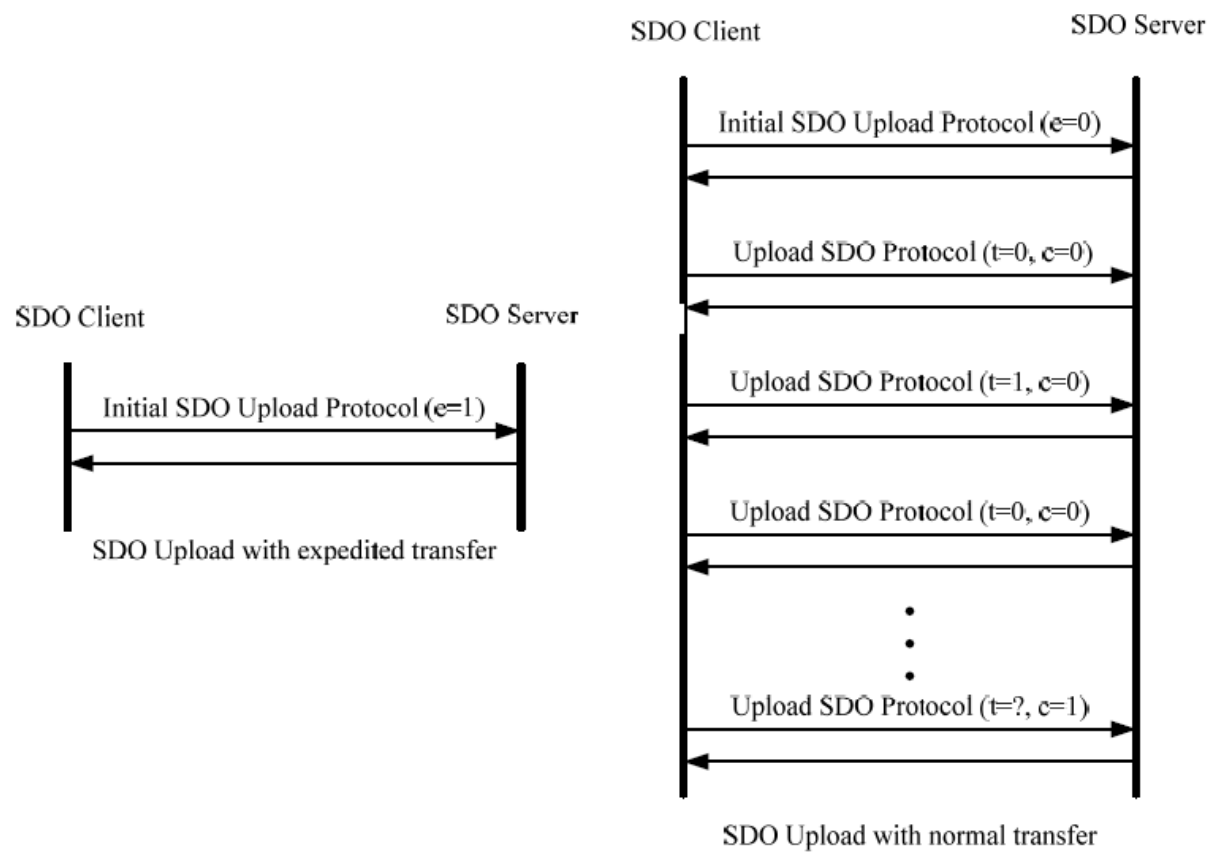
In the following paragraph, both expedited transfer and normal transfer are given according to the procedure described above. In addition, the method of how to get the value stored in the object dictionary is also presented. As to the initiate SDO upload protocol, users can obtain how many sub-indexes the object with index 0x1400 can support. This information is in the object with index 0x1400 with sub-index 00. As well, users can get the string in the object with index 0x1008 via the initiate SDO upload protocol and the upload SDO segment protocol.

● **Example for expedited transfer**

Step 1. SDO message will be sent to the PM-311x-CPS to obtain the object entry with index 0x1800 and sub-index 00 stored in the communication profile area. The message structure is as follows. Moreover, the node ID of the PM-311x-CPS is set to 1, and the information about the object entry with index 0x1800 will be described in the chapter 5.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	00	18	00	00	00	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**
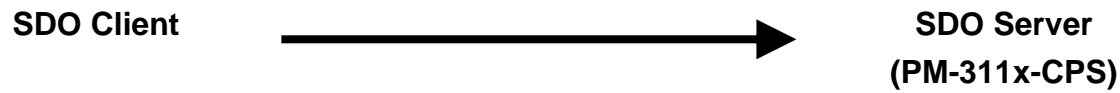
ccs : 2
m : 00 18 00
According to the low byte has the higher transferred sequence, the first byte "00" will get the priority than the second byte "18". Here the last byte "00" means the sub-index 00.

Step 2. The PM-311x-CPS will reply to the data stored in the object entry with index 0x1800 and sub-index 00.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	4F	00	18	00	05	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

scs : 2
n : 3
e : 1
s : 1
m : 00 18 00
d : 05 00 00 00
Because of the **n**=3, only the 4th byte is valid. Therefore, the feedback value is 05.

- **Example for normal transfer**

Step 1. Send the RxSDO message to the PM-311x-CPS to obtain the object entry with index 0x1008 and sub-index 00 stored in the communication profile area. The message structure is as follows. Moreover, the node ID for the PM-311x-CPS is set to 1, and the information about object entry with index 0x1008 will be described in the chapter 5.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	08	10	00	00	00	00	00

SDO Client → **SDO Server (PM-311x-CPS)**

ccs   : 2
m   : 08 10 00

Step 2. The PM-311x-CPS will respond to the SDO message with the indication of how many bytes will be uploaded from the PM-311x-CPS.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	41	08	10	00	08	00	00	00

SDO Client ← **SDO Server (PM-311x-CPS)**
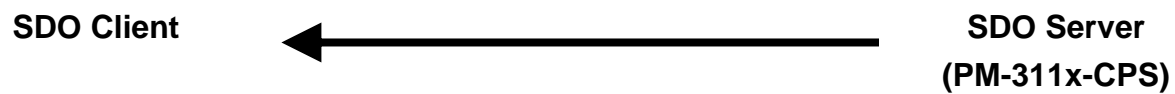
SCS   :   2
n   :   0
e   :   0
s   :   1
m   :   08 10 00
d   :   08 00 00 00
        Because of the **e**=0 and **s**=1, the **d** means how many data users will upload from the PM-311x-CPS. The byte "08" is the lowest byte in the data length with a long format. Therefore, the data "08 00 00 00" means that users will upload 8 bytes data from PM-311x-CPS.

Step 3. The PM-311x-CPS is requested to start the data transmission.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	60	00	00	00	00	00	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**
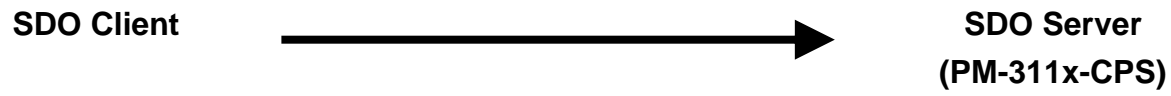
ccs : 3
t : 0

Step 4. The PM-311x-CPS will respond to the first 7 bytes in the index 0x1008 and sub-index 00 object entries.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	00	32	31	33	78	2D	43	50

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

scs : 0
t : 0
n : 0
c : 0
seg-data : 32 31 33 78 2D 43 50
Users can check the chapter 5 to know that the object entry with index 0x1008 and sub index 00 has the data type "VISIBLE_STRING". Therefore, users need to transform these data values into the corresponding ASCII character. After transformation, they become "213x-CP".

Step 5. The PM-213x is requested to transmit the rest of the data.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	70	00	00	00	00	00	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**
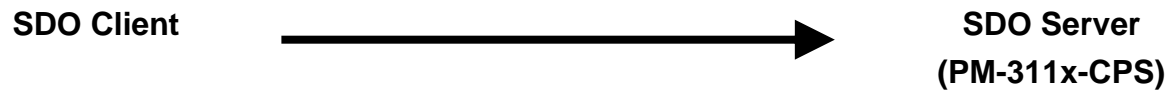
ccs	:	3
t	:	1

Step 6. Tthe rest of the data will be received from the SDO server.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID																
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	1D	53	00	00	00	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

scs	:	0
t	:	1
n	:	6
c	:	1
seg-data	:	53 00 00 00 00 00 00
		Because of the **n**=6, and only the first byte is valid, the value of 0x53 will be transferred to the corresponding ASCII character. After transformation, it became "S ".

4.1.2 Download SDO Protocol

Initiate SDO Download Protocol

The download modes are similar to the upload modes, but different in some parameters of the SDO messages. They are also separated into two steps. If the download data length is less than 4 bytes, the download action will finish in the download initialization protocol. Otherwise, the download segment protocol will be needed. These two protocols are shown below.

Initiate SDO Download Protocol
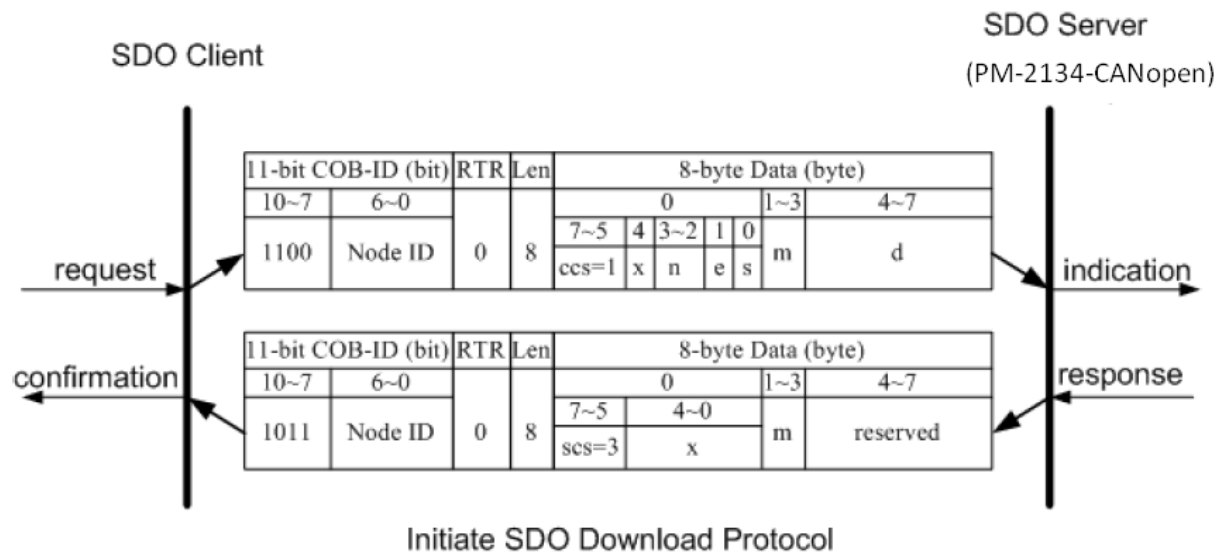
ccs	: client command specified	
	1: initiate download request	
scs	: server command specified	
	3: initiate download response	
n	: Only valid if **e** = 1 and **s** = 1, otherwise 0. If valid, it indicates the number of bytes in **d** that do not contain data. Bytes [8-**n**, 7] do not contain segment data.	
e	: transfer type	
	0: normal transfer	
	1: expedited transfer	
	If the **e**=1, it means that the data of the object are equal or less than 4 bytes, and only initiate SDO download protocol is needed. If **e**=0, the download SDO segment protocol is necessary.	
s	: size indicator	
	0: Data set size is not indicated.	
	1: Data set size is indicated.	
m	: multiplexer	

It represents the index/sub-index of the data to be transfer by the SDO. The first two bytes are the index value and the last byte is the sub-index value.

d   : data

     e=0, **s**=0: **d** is reserved for further use.

     e=0, **s**=1: **d** contains the number of bytes to be downloaded, and byte 4 contains the least significant bit, and byte 7 contains the most significant bit.

     e=1, **s**=1: **d** contains the data of length 4-**n** to be downloaded, the encoding depends on the type of the data referenced by index and sub-index.

     e=1, **s**=0: **d** contains unspecified number of bytes to be downloaded.

x   : not used, always 0

reserved   : reserved for further use , always 0

Download Segment Protocol

Download SDO Segment Protocol
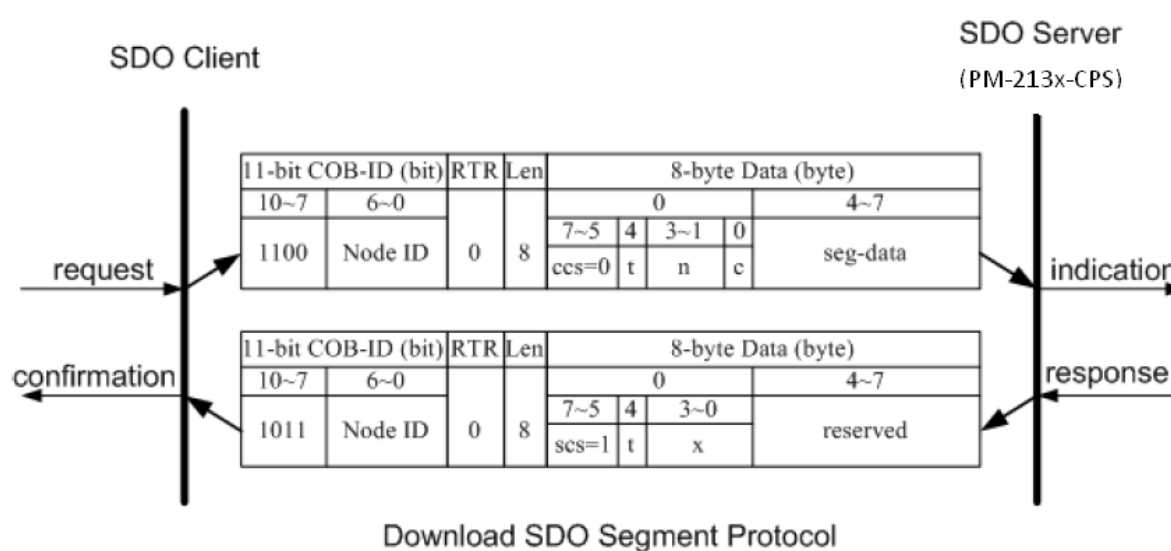
ccs   : client command specified
         0: download segment request

scs   : server command specified
         1: download segment response

seg-data   : It is at most 7 bytes of segment data to be downloaded. The encoding depends on the type of the data referenced by index and sub-index.

n   : It indicates the number of bytes in **seg-data** that do not contain segment data. Bytes [8-**n**, 7] do not contain segment data. **n** = 0 if no segment size is indicated.

c   : It indicates whether there are still more segments to be downloaded.
0:more segments to be downloaded.
1:no more segments to be downloaded.

t   : toggle bit
This bit must alternate for each subsequent segment that is downloaded. The first segment will have the toggle-bit set to 0.The toggle bit will be equal for the request and the response message.

x   : not used, always 0

reserved   : reserved for further use , always 0

SDO Download Example

When the SDO download example has been applied, the procedure in the below figure may be applied.
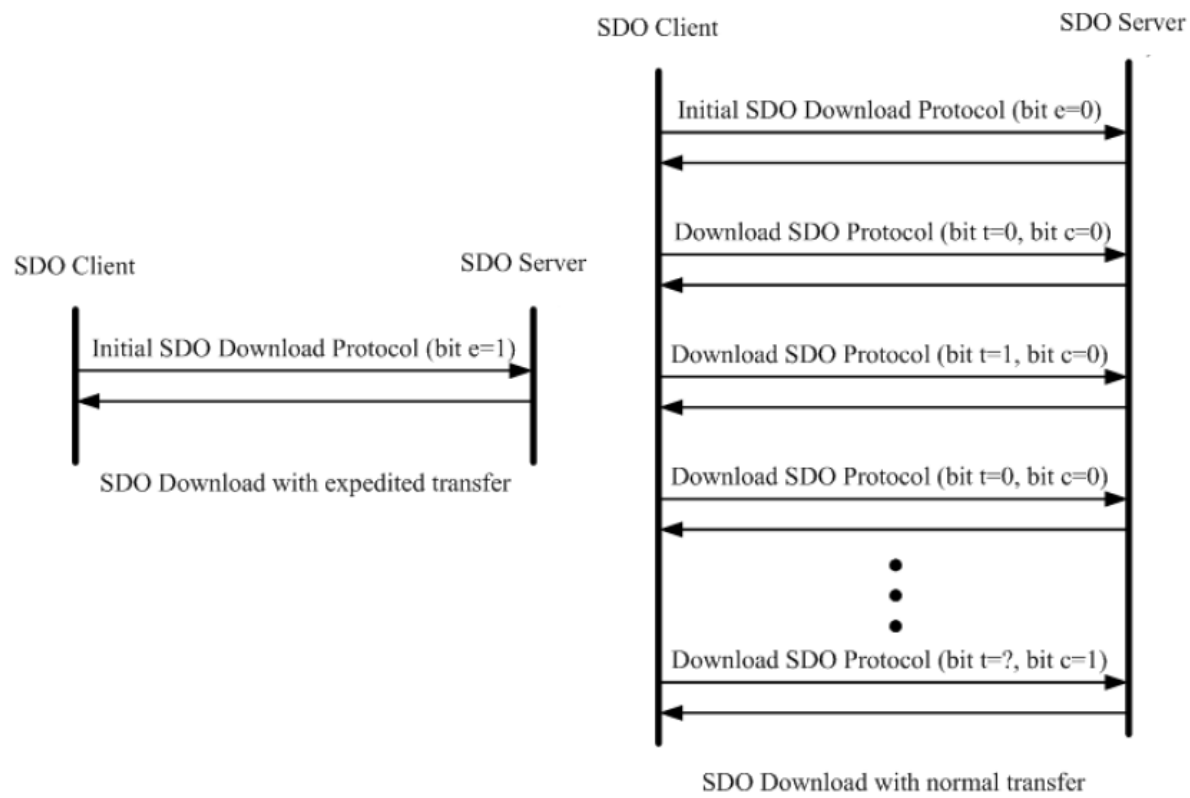
Since all of those object entries, which can be written, in the PM-311x-CPS are equal or less than 4 bytes, we can only provide the example for expedited transfer.

● **Example for expedited transfer**

Step 1. The RxSDO message is sent to the PM-311x-CPS to access the object entry with index 0x1800 and sub-index 02 stored in the communication profile area. For example, the value of this object entry is changed to 5, as the node ID for the PM-311x-CPS is set to 1.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	00	18	02	05	00	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**

ccs	: 1
n	: 3
e	: 1
s	: 1
m	: 00 18 02
d	: 05 00 00 00

According to the low byte has the higher transferred sequence, the first byte "00" will get the priority than the second byte "18". Here the last byte "00" means the sub-index 00.

Step 2. The PM-311x-CPS will reply with the message to finish the data download. Then, users can use the upload methods to read back the value.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	00	18	02	00	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

scs	:	3
m	:	00 18 02

4.1.3   Abort SDO Transfer Protocol

In some conditions, the SDO client or SDO server will terminate the SDO transmission. For example, the value of entries that users want to modify does not exist or is read-only, even users wouldn't continue the uncompleted SDO protocol under some special situations. When these conditions occur, both the client and the server can be activated to send the Abort SDO Transfer message. The Abort SDO Transfer protocol is shown below.

SDO Client   SDO Server
                                                       (PM-213x-CPS)

11-bit COB-ID (bit)		RTR	Len	8-byte Data (byte)			
10~7	6~0			0		1~3	4~7
				7~5	4~0		
1100	Node ID	0	8	cs=4	x	m	d

request →   → indication

Abort SDO Transfer Protocol

SDO Client   SDO Server
                                                       (PM-213x-CPS)

11-bit COB-ID (bit)		RTR	Len	8-byte Data (byte)			
10~7	6~0			0		1~3	4~7
				7~5	4~0		
1011	Node ID	0	8	cs=4	x	m	d

← indication   ← request

Abort SDO Transfer Protocol
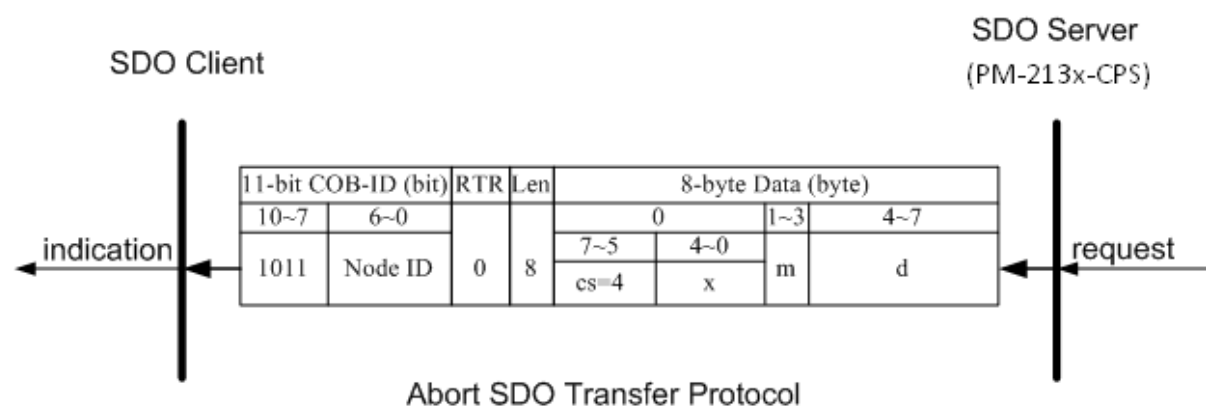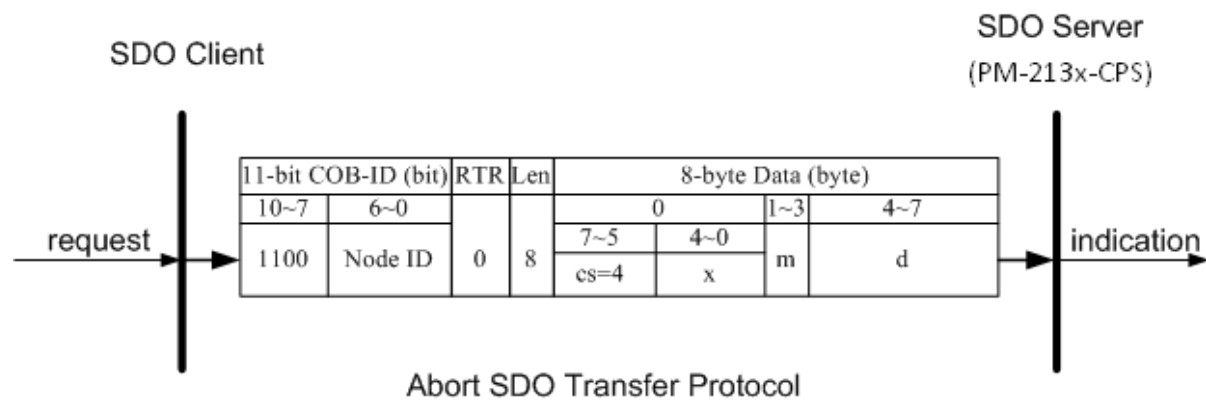
cs   :   command specified
              4: abort transfer request
x   :   not used, always 0
m   :   multiplexer
              It represents index and sub-index of the SDO
d   :   contains a 4-byte "Abort Code" about the reason for
              the abort.

Abort Code	Description
0503 0000h	Toggle bit not alternated.
0504 0000h	SDO protocol timed out.
0504 0001h	Client/server command specified not valid or unknown.
0504 0002h	Invalid block size (block mode only).
0504 0003h	Invalid sequence number (block mode only).
0504 0004h	CRC error (block mode only).
0504 0005h	Out of memory.
0601 0000h	Unsupported access to an object.
0601 0001h	Attempt to read a write only object.
0601 0002h	Attempt to write a read only object.
0602 0000h	Object does not exist in the object dictionary.
0604 0041h	Object cannot be mapped to the PDO.
0604 0042h	The number and length of the objects to be mapped would exceed PDO length.
0604 0043h	General parameter incompatibility reason.
0604 0047h	General internal incompatibility in the device.
0606 0000h	Access failed due to an hardware error.
0607 0010h	Data type does not match, length of service parameter does not match
0607 0012h	Data type does not match, length of service parameter too high
0607 0013h	Data type does not match, length of service parameter too low
0609 0011h	Sub-index does not exist.
0609 0030h	Value range of parameter exceeded (only for write access).
0609 0031h	Value of parameter written too high.
0609 0032h	Value of parameter written too low.
0609 0036h	Maximum value is less than minimum value.
0800 0000h	General error.
0800 0020h	Data cannot be transferred or stored to the application.
0800 0021h	Data cannot be transferred or stored to the application because of local control.
0800 0022h	Data cannot be transferred or stored to the application because of the present device state.
0800 0023h	Object dictionary dynamic generation fails or no object dictionary is present (e.g. object dictionary is generated from file and generation fails because of an file error).

Abort SDO Transfer Example

The object index 0x1008 doesn't support the sub-index 01 entry. Therefore, if users read the object entry with index 0x1008 and sub-index 01, the PM-311x-CPS will reply the Abort SDO Transfer message. The example is figured as follows.

Step 1. The RxSDO message will be sent to the PM-311x-CPS in order to get the object entry with index 0x1008 and sub-index 01. The following example is assumed that the node ID for the PM-311x-CPS is set to 1.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	40	08	10	01	00	00	00	00

SDO Client ➡ **SDO Server (PM-311x-CPS)**

ccs : 2
m : 08 10 01

Step 2. The PM-311x-CPS will reply to the Abort SDO message as shown below.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	80	08	10	01	11	00	09	06

SDO Client ⬅ **SDO Server (PM-311x-CPS)**

cs : 4
m : 08 10 01
d : 11 00 09 06

According to the low byte data have the transferring priority, the data will be converted to "06 09 00 11". Therefore, after searching the Abort Code table described above, this Abort Code can be interpreted as "Sub-index does not exist".

4.2   PDO Communication Set

4.2.1   PDO COB-ID Parameters

Before the real-time data are transmitted by the PDO, it is necessary to check the COB-ID parameter of this PDO in the PDO communication objects. This parameter setting controls the COB-ID of the PDO communication, which is in 32 bits, and each bit with its meaning is given in the table follow.

Bit Number	Value	Meaning
31(MSB)	0	PDO exits (PDO is valid)
	1	PDO does not exist (PDO is not valid)
30	0	RTR allowed on this PDO
	1	No RTR allowed on this PDO
29	0	11-bit ID (CAN 2.0A)
	1	29-bit ID (CAN 2.0B)
28-11	0	If bit 29=0
	X	If bit 29=1: 28-11 bits of 29-bit COB-ID
10-0(LSB)	X	10-0 bits of COB-ID

Note: Only PM-311x-CPS supports CAN 2.0A.

In the following table, it's regarding the default PDO COB-ID parameters.

Number of PDO	Default COB-ID	
	Bit10~Bit7 (Function Code)	Bit6~Bit0
TxPDO1	0011	Node-ID
TxPDO2	0101	Node-ID
TxPDO3	0111	Node-ID
TxPDO4	1001	Node-ID
RxPDO1	0100	Node-ID
RxPDO2	0110	Node-ID
RxPDO3	1000	Node-ID
RxPDO4	1010	Node-ID

Note:
1. Users can also define the PDO COB-ID by themselves. Actually, all COB-ID can be defined by users except the reserved COB-ID described in the table of the section 3.1. It is important to avoid the conflict with the defined COB-ID used in the same node.
2. The PDO COB-ID parameters cannot be changed if the PDO is valid (bit 31 =0).

4.2.2 Transmission Type

The transmission type is one of the several parameters defined in PDO communication objects with sub-index 02. Each PDO has its own transmission type. The transmission type can indicate the transmission or reception character for its corresponding PDO. The following table describes the relationship between the value of the transmission type and the PDO character. For example, if users used transmission type 0 for the first TxPDO, the CANopen device will follow the rule of the acyclic and synchronous PDO transmission.

Transmission Type	PDO Transmission Method				
	cyclic	acyclic	synchronous	Asynchronous	RTR only
0		O	O		
1-240	O		O		
241-251	----------------------------Reserved----------------------------------				
252			O		O
253				O	O
254				O	
255				O	

Note:
1. The transmission type 1-240 indicates how many SYNC objects the TxPDO will be triggered. The RxPDO is always triggered by the following SYNC upon reception of data independent of the transmission types 0-240.
2. The transmission type 252 and 253 are only used for TxPDO. The transmission type 252 means that the data is updated (but not sent) immediately after reception of the SYNC object. For these two transmission types, the PDO is only transmitted on remote transmission requests.
3. For the transmission types 254 and 255, the event timer will be used in the TxPDO. The PDO, including the DI value, will be sent when the DI value is changed. And both transmission types will directly trigger an update of the mapped data when receiving the RxPDO.
4. The PM-311x-CPS doesn't support RxPDO.

4.2.3 PDO Communication Rule

The PDO related objects are indicated from index 0x1400 to 0x1BFF. For the PM-311x-CPS, RxPDO communication objects are not used. The ranges of the TxPDO communication objects and the mapping objects are from index 0x1800 to index 0x1813 and from index 0x1A00 to index 0x1A13 respectively. Moreover, each PDO communication object has its own PDO mapping object.

For example, the first TxPDO communication object is stored in the entry with index 0x1800, and the corresponding mapping object is stored in an entry with index 0x1A00. The object with index 0x1801 and the object with index 0x1A01 are a group, and so on. Therefore, before users access the practical data via PDO communication, each parameter for the PDO communications and mapping objects must be controlled.

Besides, only PDO communications can be used in the NMT operational state. Users can use the NMT module control protocol to change the NMT state of the PM-311x-CPS. It is described in the section 3.4. Besides, during communication via the PDO messages, the data length of the PDO message must match with the PDO mapping object. If the data length 'L' of the PDO message exceeds the total bytes 'n' of the PDO mapping object entries, only the first 'n' bytes of the PDO message are used by the PDO consumer. If 'L' is less than 'n', the PDO message will not be disposed by the PDO consumer, and an Emergency message with error code 8210h will be transmitted to the PDO producer. The PDO communication set is shown as follows.
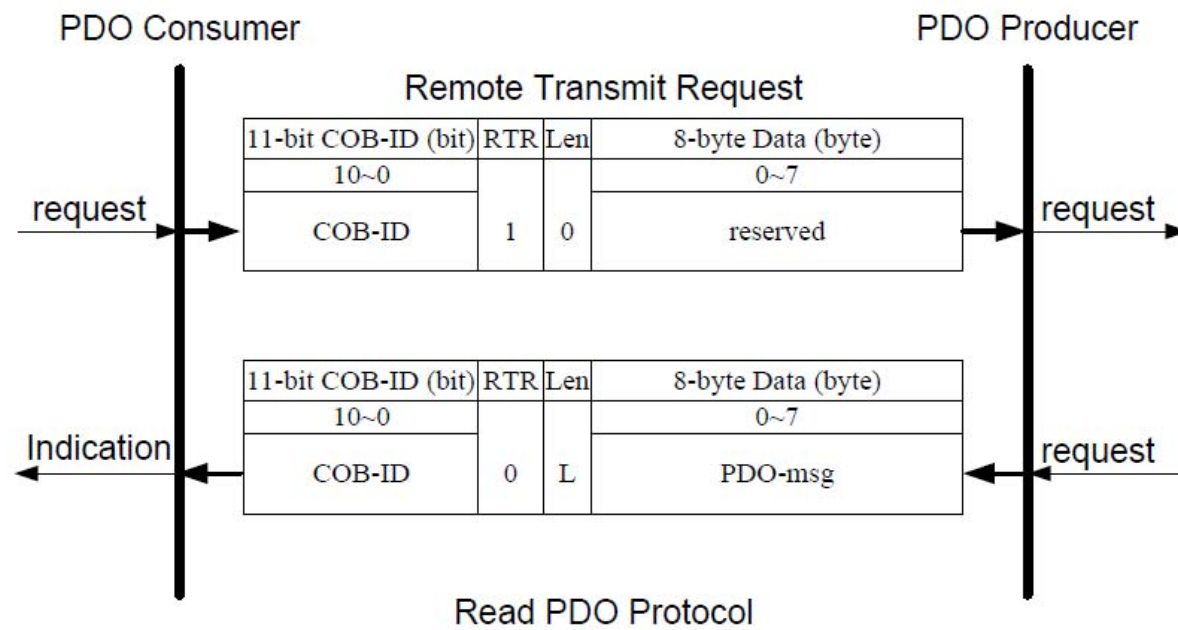
PDO Consumer ... PDO Producer

Remote Transmit Request

Read PDO Protocol

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

PDO Communication Example

Before describing the example, the step0 must be checked. And the default COB-ID for each communication object is assumed to be being used.

Step0: The following message must be sent in order to change the NMT state of the PM-311x-CPS first, because only the PDO communication can run under the NMT Operational state.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
0	0	0	0	0	0	0	0	0	0	0	0	8	01	01	00	00	00	00	00	00

NMT master ⟶ **NMT slave (PM-311x-CPS)**

cs : 1
Node ID : 1

Step 1. Users need to use the RTR message from the 2nd TxPDO to read back the power meter data in the PM-311x-CPS.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
0	1	0	1	0	0	0	0	0	0	1	1	0	00	00	00	00	00	00	00	00
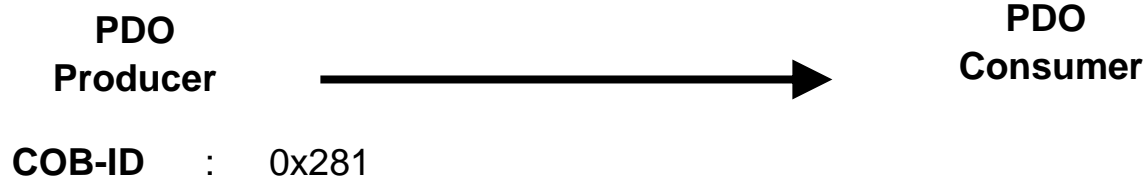
**PDO
Producer**   ⟶   **PDO
Consumer**

COB-ID : 0x281

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
0	1	0	1	0	0	0	0	0	0	1	0	8	91	64	8A	BC	76	20	06	C0

**PDO
Producer**   ⟵   **PDO
Consumer**

COB-ID : 0x281
L   : 8
PDO-msg : 91 64 8A BC 76 20 06 C0
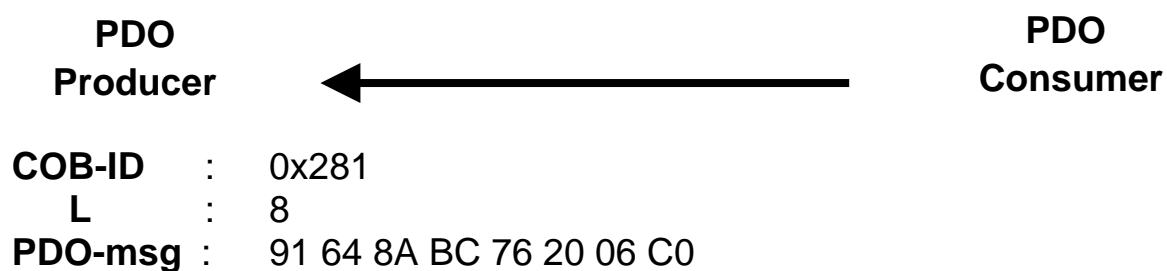
● **Event Timer Functionality**

Step 2. Users can use the SDO to change the event timer of the 2nd TxPDO to 1000, stored in index 0x1801 with sub-index 5. In addition, the value 1000 means 1 second according to the event timer is ms,

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	01	18	05	E8	03	00	00

SDO Client   ⟶   **SDO Server
(PM-311x-CPS)**

ccs	: 1
n	: 2
e	: 1
s	: 1
m	: 01 18 05
d	: E8 03 00 00

The value 0x03E8 is equal to 1000. Because the **n**=2, the last two bytes "00 00" is useless.

Step 3. The PM-311x-CPS will response the message to finish the data download.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	01	18	05	00	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

scs : 3
m : 01 18 05

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	58	84	8A	BC	1F	46	06	C0

PDO Producer ⟵ **PDO Consumer**

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	A9	2F	8A	BC	33	46	06	C0

	PDO		PDO
	Producer	←	**Consumer**

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	1	0	1	0	0	0	0	0	0	1	0	8	31	1F	8A	BC	47	46	06	C0

	PDO		PDO
	Producer	←	**Consumer**

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	01	18	05	00	00	00	00

SDO Client ———————————————▶ **SDO Server (PM-311x-CPS)**
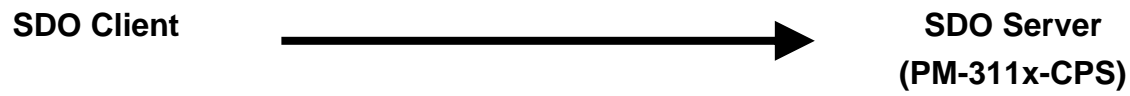
ccs : 1
n : 2
e : 1
s : 1
m : 01 18 05
d : 00 00 00 00

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0										
1	0	1	1	0	0	0	0	0	0	1	0	8	60	01	18	05	00	00	00	00

SDO Client ◀——————————————— **SDO Server (PM-311x-CPS)**

SCS : 4
m : 01 18 05

● **Dynamic PDO Mapping for other data**

Step 4. Users can use the 5th TxPDO to create a new PDO communication with PDO COB-ID 0x182, which is unused in the PM-311x-CPS. Before setting the COB-ID of a PDO, users have to check the bit 31 of the COB-ID first. Only the COB-ID with the value 0 on the bit 31 can be changed. So the COB-ID can be configured directly according to the 5th TxPDO is invalid.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	23	05	18	01	82	01	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**
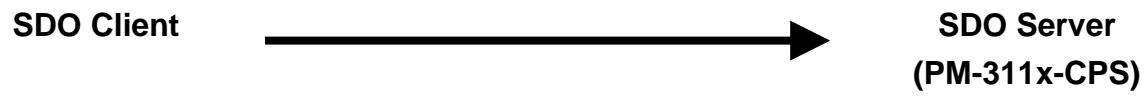
ccs : 1
n : 0
e : 1
s : 1
m : 05 18 01
d : 82 01 00 00

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	05	18	01	00	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**

SCS : 4
m : 05 18 01

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	5	1	0			0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	8	01	01	00	00	00	00	00	00

NMT master ⟶ **NMT slave (PM-311x-CPS)**

cs : 1
Node ID : 1

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	0	2	1	0	00	00	00	00	00	00	00	00

**PDO**
Producer ⟶ **PDO**
Consumer

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	0	2	0	8	3F	55	DC	42	3F	CA	66	3E

PDO
Producer ⟵ **PDO**
Consumer

COB-ID : the default PDO COB-ID, or the PDO COB-ID defined by users

L : the data length about how many bytes the PDO message has

PDO-msg : the real-time data or the data which can be mapped into the PDO mapping objects

4.3   NMT Communication Set

4.3.1   Module Control Protocol

The NMT communication set can be applied for changing the NMT slave status. The following figure shows how to change the different NMT statuses for the PM-311x-CPS.

Start Remote Node Protocol

Start Remote Node Protocol
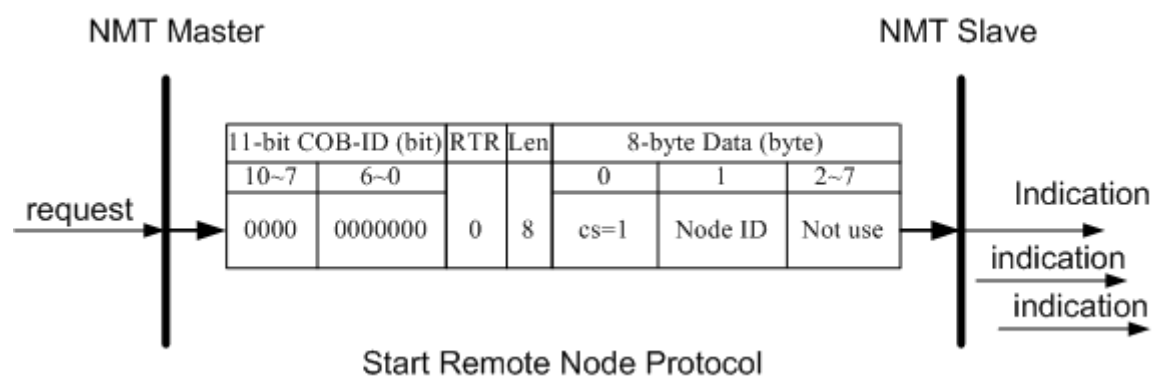
cs   : NMT command specified
            1: start
Node ID : the node ID of the NMT slave device

Stop Remote Node Protocol

Stop Remote Node Protocol
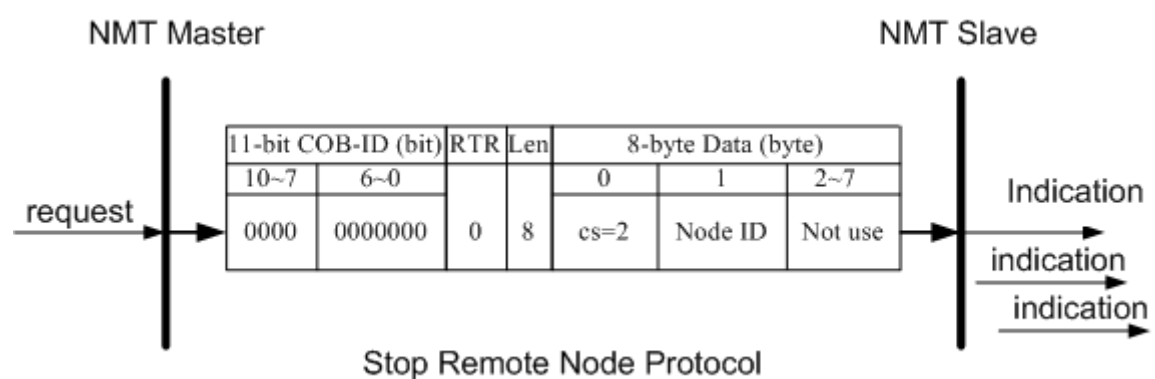
cs   : NMT command specified
            2: stop
Node ID : the node ID of the NMT slave device

Enter Pre-Operational Protocol

Enter Pre-Operational Protocol
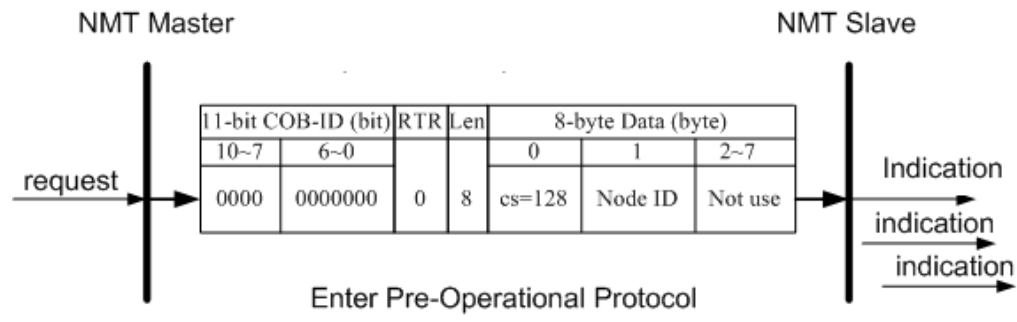
cs : NMT command specified
128: enter PRE-OPERATIONAL
Node ID : the node ID of the NMT slave device

Reset Node Protocol

Reset Node Protocol
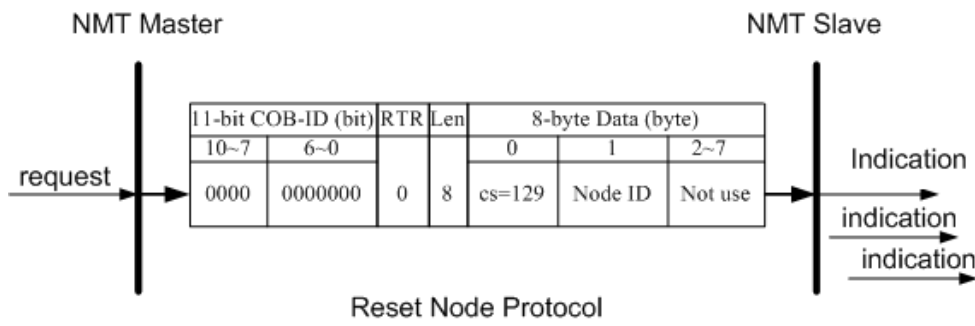
cs : NMT command specified
129: Reset_Node
Node ID : the node ID of the NMT slave device

Reset Communication Protocol

Reset Communication Protocol
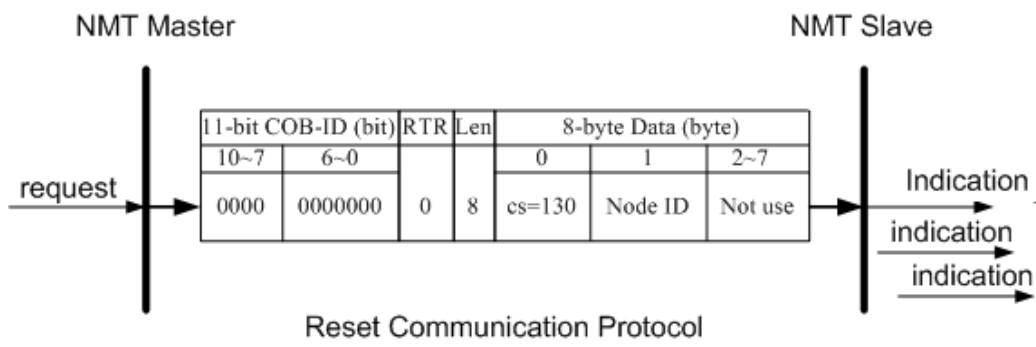
cs : NMT command specified
130: Reset_Communication
Node ID : the node ID of the NMT slave device

Module Control Protocol Example

If the PM-311x-CPS node ID is set to 5 as an example, the following steps would be....

Step1. Turn off the PM-311x-CPS.

Step2. Then, turn it on. After the initialization, the PM-311x-CPS will automatically enter the Pre_Operational state. Users will note the RUN LED flashing twice per second.

Step3. Users can send the NMT module control protocol, and control the PM-311x-CPS to enter the operational state.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	0	8	01	05	00	00	00	00	00	00
0	0	0	0	0	0	0	0	0	0	0										

NMT master ➡️ **NMT slave
(PM-311x-CPS)**

      cs   : 1
Node ID : 5

4.3.2   Error Control Protocol

Error Control Protocol is a kind of the solution to check whether the CANopen device is still alive or not. And its related objects include 0x100C and 0x100D. The 0x100C is the guard time, and the 0x100D is the life time factor. The node life time is the guard time multiplied by the life time factor. The Node Guarding timer of the PM-311x-CPS will start to count after receiving the first RTR message for the guarding identifier. The communication set of the Error Control protocol is displayed below.

Node Guarding Protocol
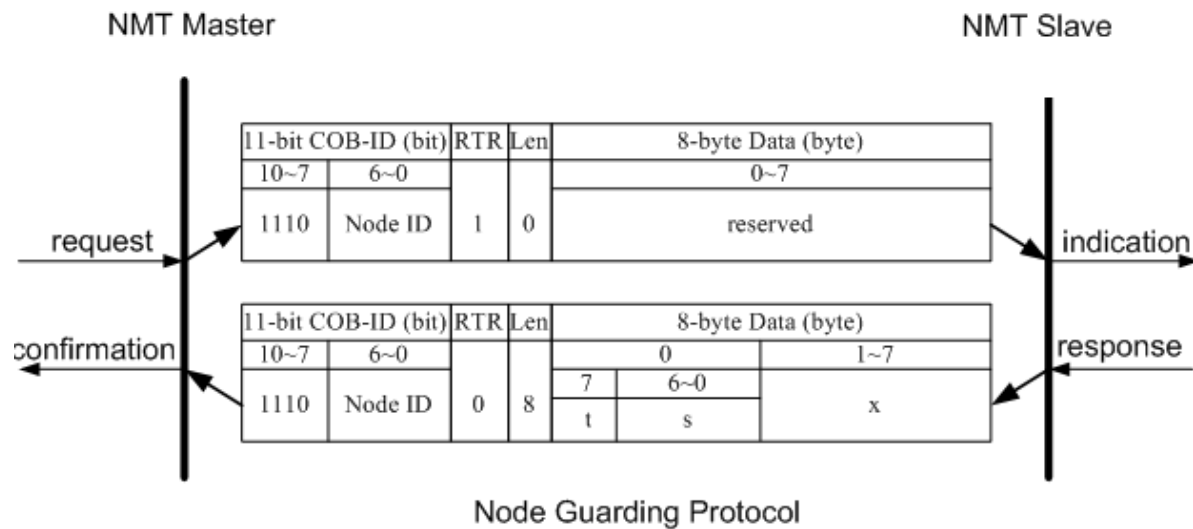
t : toggle bit
            The value of this bit will be alternatively changed between two consecutive responses from the NMT slave. After the Node Guarding protocol becomes active, the value of the toggle-bit of the first response will be 0.
s : the state of the NMT Slave
            4: STOPPED
            5: OPERATIONAL
            127: PRE_OPERATIONAL

Error Control Protocol Example

The default EMCY function code and the node ID 1 for the PM-311x-CPS are used as an example on the error control protocol. The steps will be as follows.

Step 1. Turn off the PM-311x-CPS. Then, turn it on. The PM-311x-CPS will be in the Pre_Operational state.

Step 2. Users can set the guard time value to 250. This value will be

stored in index 0x100C with sub-index 00.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2B	0C	10	00	FA	00	00	00

SDO Client ⟶ **SDO Server (PM-311x-CPS)**
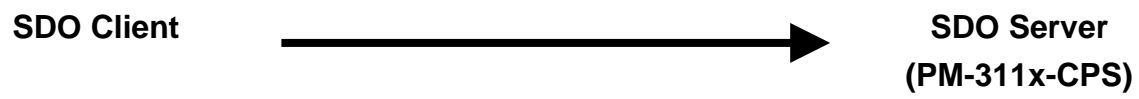
ccs : 1
n : 2
e : 1
s : 1
m : 0C 10 00
d : FA 00 00 00

Step 3. The PM-311x-CPS will reply with the ending message.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	0C	10	00	00	00	00	00

SDO Client ⟵ **SDO Server (PM-311x-CPS)**
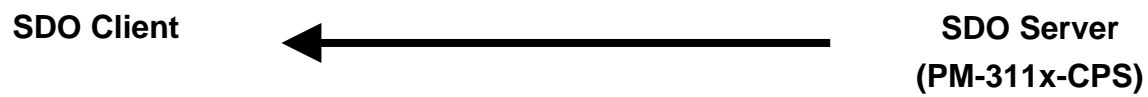
scs : 3
m : 0C 10 00

Step 4. Users can set the life-time factor value to 4. This value will be stored in the index 0x100D with sub-index 00. Then, the ending message from PM-311x-CPS will be received.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	2F	0D	10	00	04	00	00	00

SDO Client ————————————▶ **SDO Server**
(PM-311x-CPS)

ccs	: 1
n	: 3
e	: 1
s	: 1
m	: 0D 10 00
d	: 04 00 00 00

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	0D	10	00	00	00	00	00

SDO Client ◀———————————— **SDO Server**
(PM-311x-CPS)

scs	:	3
m	:	0D 10 00

Step 5. Users can send the node guarding protocol to start the mechanism of the node guard. The life time here is equal to 1000 ms (guard time * life time factor =250*4=1000),

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0	0	0	1	1	0	00	00	00	00	00	00	00	00

NMT master ————————————▶ **NMT slave**
(PM-311x-CPS)

COB-ID : 0x701

Step 6. Then, users will receive the message, recording the NMT state of the PM-311x-CPS. For the reason that life time is equal to 1000 ms (guard time * life time factor =250*4=1000), users will transmit the node guarding protocol again.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0										
1	1	1	0	0	0	0	0	0	0	1	0	8	7F	00	00	00	00	00	00	00

NMT master ⟶ **NMT slave (PM-311x-CPS)**

COB-ID : 0x701
t : 1
s : 7F
The value 7F means that the PM-311x-CPS is in the NMT Pre_Operational state.

4.4   Special Functions for PM-311x-CPS

4.4.1   Power Meter Data Table

The PM-311x-CPS Manufacturer in the Specific Profile Area defines some entries, which are used for the power meter data. The objects with index 0x3200~0x3208 will map to the PDOs as below table. The D0 to D7 represent the CANopen message from Data0 to Data7.Below is an PDO mapping example of PM-3114-CPS.

No.(PDO)	COB-ID	Data Length	D0~D3	D4~D7
1	0x180+Node-ID	8	kW(Kw_a)	kWh_a
2	0x280+Node-ID	8	kW(Kw_b)	kWh_b
3	0x380+Node-ID	8	kW(Kw_c)	kWh_c
4	0x480+Node-ID	8	kW(Kw_d)	kWh_d
5	---	8	Volt(V_a)	Amp(I_a)
6	---	8	Volt(V_b)	Amp(I_b)
7	---	8	Volt(V_c)	Amp(I_c)
8	---	8	Volt(V_d)	Amp(I_d)
9	---	8	kvar(kvar_a)	kVA(Kva_a)
10	---	8	kvar(kvar_b)	kVA(Kva_b)
11	---	8	kvar(kvar_c)	kVA(Kva_c)
12	---	8	kvar(kvar_d)	kVA(Kva_d)
13	---	8	PF_a	kVAh_a
14	---	8	PF_b	kVAh_d
15	---	8	PF_c	kVAh_c
16	---	8	PF_d	kVAh_d
17	---	4	kvarh_a	---
18	---	4	kvarh_b	---
19	---	4	kvarh_c	---
20	---	4	kvarh_d	---

User can read PDO1~PDO4 by using default COB-ID. Suppose the node be set to 1.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	0	8	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	8	01	01	00	00	00	00	00	00

NMT master ⟶ **NMT slave (PM-311x-CPS)**

cs : 1
Node ID : 1

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	1	0	0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	0	1	1	0	00	00	00	00	00	00	00	00

PDO Producer ⟶ **PDO Consumer**

COB-ID : 0x181

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	0	8	0	1	2	3	4	5	6	7
0	0	1	1	0	0	0	0	0	0	1	0	8	B1	CD	8C	BC	6A	1A	F0	BF

PDO Producer ⟵ **PDO Consumer**

COB-ID : 0x181
L : 8
PDO-msg : B1 CD 8C BC 6A 1A F0 BF
The D0~D3 is data of Kw_a and the D4~D7 is the data of kWh_a.

It is necessary to use dynamic PDO if user want to get date PDO5~PDO20. For example, read data of PDO 9.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	1	0	0	0	0	0	0	0	0	1	0	8	23	08	18	01	82	01	00	00

SDO Client ————————————————▶ **SDO Server**
(PM-311x-CPS)

ccs : 1
n : 0
e : 1
s : 1
m : 08 18 01
d : 82 01 00 00

Step 3. The PM-311x-CPS will reply with the ending message.

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
1	0	1	1	0	0	0	0	0	0	1	0	8	60	08	18	01	00	00	00	00

SDO Client ◀———————————————— **SDO Server**
(PM-311x-CPS)

SCS : 3
m : 08 18 01

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0			0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	8	01	01	00	00	00	00	00	00

NMT master ————————————————▶ **NMT slave**
(PM-311x-CPS)

cs : 1
Node ID : 1

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	1	0	00	00	00	00	00	00	00	00
0	0	1	1	0	0	0	0	0	0	2										

**PDO
Producer** ⟶ **PDO
Consumer**

COB-ID   : 0x182

11-bit COB-ID (bit)											RTR	Data Length	8-byte Data (byte)							
Func Code				Node ID									0	1	2	3	4	5	6	7
10	9	8	7	6	5	4	3	2	1	0	0	8	E0	26	CC	3C	65	42	95	3C
0	0	1	1	0	0	0	0	0	0	2										

**PDO
Producer** ⟵ **PDO
Consumer**

COB-ID   : 0x182
L   : 8
PDO-msg : E0 26 CC 3C 65 42 95 3C
         The D0~D3 is data of Kvar_a and the D4~D7 is the data of
         kva_a.

5. Object Dictionary of PM-311x-CPS

5.1 Communication Profile Area

The following tables are regarding each entry of the communication profile area is defined in PM-311x-CPS. Taking PM-3114-CPS as example in the below. For the convenient purpose, all communication entries are divided into several tables. They are "General Communication Entries", "TxPDO Communication Entries", and "TxPDO Mapping Communication Entries".

Please note that in the table header with "Idx", "Sidx" and "Attr" represent "index", "sub-index", and "attribute" respectively. The sign "---" in the default field means that the default is not defined or can be defined conditionally by the firmware built in PM-311x-CPS. In the table, the number accompanying letter "h" indicates that this value is in the hex format.

General Communication Entries

Idx	Sidx	Description	Type	Attr	Default
1000h	0h	device type	UNSIGNED 32	RO	---
1001h	0h	error register	UNSIGNED 8	RO	---
1003h	0h	largest sub-index supported for "predefine error field"	UNSIGNED 8	RO	0h
	1h	actual error (the newest one)	UNSIGNED 32	RO	---
	…	…	…	…	---
	5h	actual error (the oldest one)	UNSIGNED 32	RO	---
1005h	0h	COB-ID of Sync message	UNSIGNED 32	RW	80h
1008h	0h	manufacturer device name	VISIBLE_STRING	RO	
1009h	0h	manufacturer hardware version	VISIBLE_STRING	RO	---
100Ah	0h	manufacturer software version	VISIBLE_STRING	RO	---
100Ch	0h	guard time	UNSIGNED 16	RW	0
100Dh	0h	life time factor	UNSIGNED 8	RW	0
1014h	0h	COB-ID of EMCY	UNSIGNED 32	RW	80h+Node-ID
1015h	0h	Inhibit time of EMCY	UNSIGNED 16	RW	0
1018h	0h	largest sub-index supported for "identity object"	UNSIGNED 8	RO	1
	1h	vender ID	UNSIGNED 32	RO	---

SDO Communication Entries

Idx	Sidx	Description	Type	Attr	Default
1200h	0h	largest sub-index supported for "server SDO parameter"	UNSIGNED 8	RO	2
	1h	COB-ID form client to server (RxSDO)	UNSIGNED 32	RO	600h+Node-ID
	2h	COB-ID form server to client(TxSDO)	UNSIGNED 32	RO	580h+Node-ID

TxPDO Communication Entries

Idx	Sidx	Description	Type	Attr	Default
1800h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	180h+Node-ID
	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0
1801h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	280h+Node-ID
	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0
1802h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	380h+Node-ID
	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0
1803h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	480h+Node-ID
	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0
1804h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	80000000h

	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0
…	…	…	…	…	…
1813h	0	largest sub-index supported for "receive PDO parameter"	UNSIGNED 8	RO	5
	1	COB-ID used by PDO (Tx)	UNSIGNED 32	RW	80000000h
	2	transmission type	UNSIGNED 8	RW	FFh
	3	inhibit time	UNSIGNED 16	RW	0
	4	Reversed	…	…	…
	5	event timer	UNSIGNED 16	RW	0

TxPDO Mapping Communication Entries

Idx	Sidx	Description	Type	Attr	Default
1A00h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kW(Kw_a) data	INTEGER 32	RO	3200 0120h
	2	read kWh_a data	INTEGER 32	RO	3201 0120h
1A01h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kW(Kw_b) data	INTEGER 32	RO	3200 0220h
	2	read kWh_b data	INTEGER 32	RO	3201 0220h
1A02h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kW(Kw_c) data	INTEGER 32	RO	3200 0320h
	2	read kWh_c data	INTEGER 32	RO	3201 0320h
1A03h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kW(Kw_d) data	INTEGER 32	RO	3200 0420h
	2	read kWh_d data	INTEGER 32	RO	3201 0420h
1A04h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read Volt(V_a) data	INTEGER 32	RO	3202 0120h
	2	read Amp(I_a) data	INTEGER 32	RO	3203 0120h

1A05h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read Volt(V_b) data	INTEGER 32	RO	3202 0220h
	2	read Amp(I_b) data	INTEGER 32	RO	3203 0220h
1A06h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read Volt(V_c) data	INTEGER 32	RO	3202 0320h
	2	read Amp(I_c) data	INTEGER 32	RO	3203 0320h
1A07h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read Volt(V_d) data	INTEGER 32	RO	3202 0420h
	2	read Amp(I_d) data	INTEGER 32	RO	3203 0420h
1A08h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kvar(kvar_a) data	INTEGER 32	RO	3204 0120h
	2	read kVA(Kva_a) data	INTEGER 32	RO	3205 0120h
1A09h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kvar(kvar_b) data	INTEGER 32	RO	3204 0220h
	2	read kVA(Kva_b) data	INTEGER 32	RO	3205 0220h
1A0Ah	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kvar(kvar_c) data	INTEGER 32	RO	3204 0320h
	2	read kVA(Kva_c) data	INTEGER 32	RO	3205 0320h
1A0Bh	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read kvar(kvar_d) data	INTEGER 32	RO	3204 0420h
	2	read kVA(Kva_d) data	INTEGER 32	RO	3205 0420h
1A0Ch	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read PF_a data	INTEGER 32	RO	3206 0120h
	2	read kVAh_a data	INTEGER 32	RO	3207 0120h
1A0Dh	0	inhibit time	UNSIGNED 8	RO	2

	1	read PF_b data	INTEGER 32	RO	3206 0220h
	2	read kVAh_b data	INTEGER 32	RO	3207 0220h
1A0Eh	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read PF_c data	INTEGER 32	RO	3206 0320h
	2	read kVAh_c data	INTEGER 32	RO	3207 0320h
1A0Fh	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	2
	1	read PF_d data	INTEGER 32	RO	3206 0420h
	2	read kVAh_d data	INTEGER 32	RO	3207 0420h
1A10h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	1
	1	read kvarh_a data	INTEGER 32	RO	3208 0120h
1A11h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	1
	1	read kvarh_b data	INTEGER 32	RO	3208 0220h
1A12h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	1
	1	read kvarh_c data	INTEGER 32	RO	3208 0320h
1A13h	0	largest sub-index supported for "transmit PDO mapping"	UNSIGNED 8	RO	1
	1	read kvarh_d data	INTEGER 32	RO	3208 0420h

Idx	Sidx	Description	Type	Attr	Default
6200h	0	largest sub-index supported for object	UNSIGNED 8	RW	1
	1	Write digital output 1h to 2h	UNSIGNED 8	RW	-

5.2 Manufacturer Specific Profile Area

In the following table, there is information about some special functions for the PM-311x-CPS. The index from 0x3200 to 0x3208 records the power meter measurement parameters which is taking PM-3114-CPS as example. The number of these entries will be automatically updated when the PM-311x-CPS boot up. 0x3209 stores two meter parameters, PT Ratio and RT Ratio. PT Ratio means potential transformer ratio, the default value is 100 and the unit is 0.1. RT Ratio means current transformer ratio, the default value is 1 and the unit is 1. 0x320A is used to clear energy data, reset setting and set frequency.

Idx	Sidx	Description	Type	Attr	Default
3200h	0	largest sub-index supported for "kW"	UNSIGNED 8	RO	4
	1	kW(Kw_a)	INTEGER32	RO	0
	2	kW(Kw_b)	INTEGER32	RO	0
	3	kW(Kw_c)	INTEGER32	RO	0
	4	kW(Kw_d)	INTEGER32	RO	0
3201h	0	largest sub-index supported for "kWh"	UNSIGNED 8	RO	4
	1	kWh_a	INTEGER32	RO	0
	2	kWh_b	INTEGER32	RO	0
	3	kWh_c	INTEGER32	RO	0
	4	kWh_d	INTEGER32	RO	0
3202h	0	largest sub-index supported for "Volt"	UNSIGNED 8	RO	4
	1	Volt(V_a)	INTEGER32	RO	0
	2	Volt(V_b)	INTEGER32	RO	0
	3	Volt(V_c)	INTEGER32	RO	0
	4	Volt(V_b)	INTEGER32	RO	0
3203h	0	largest sub-index supported for "Amp"	UNSIGNED 8	RO	4
	1	Amp(I_a)	INTEGER32	RO	0
	2	Amp(I_b)	INTEGER32	RO	0
	3	Amp(I_c)	INTEGER32	RO	0
	4	Amp(I_d)	INTEGER32	RO	0
3204h	0	largest sub-index supported for "kvar"	UNSIGNED 8	RO	4
	1	kvar(kvar_a)	INTEGER32	RO	0

	2	kvar(kvar_b)	INTEGER32	RO	0
	3	kvar(kvar_c)	INTEGER32	RO	0
	4	kvar(kvar_d)	INTEGER32	RO	0
3205h	0	largest sub-index supported for "kVA"	UNSIGNED 8	RO	4
	1	kVA(Kva_a)	INTEGER32	RO	0
	2	kVA(Kva_b)	INTEGER32	RO	0
	3	kVA(Kva_c)	INTEGER32	RO	0
	4	kVA(Kva_d)	INTEGER32	RO	0
3206h	0	largest sub-index supported for "PF"	UNSIGNED 8	RO	4
	1	PF_a	INTEGER32	RO	0
	2	PF_b	INTEGER32	RO	0
	3	PF_c	INTEGER32	RO	0
	4	PF_d	INTEGER32	RO	0
3207h	0	largest sub-index supported for "kVAh"	UNSIGNED 8	RO	4
	1	kVAh_a	INTEGER32	RO	0
	2	kVAh_b	INTEGER32	RO	0
	3	kVAh_c	INTEGER32	RO	0
	4	kVAh_d	INTEGER32	RO	0
3208h	0	largest sub-index supported for "kvarh"	UNSIGNED 8	RO	4
	1	kvarh_a	INTEGER32	RO	0
	2	kvarh_b	INTEGER32	RO	0
	3	kvarh_c	INTEGER32	RO	0
	4	kvarh_d	INTEGER32	RO	0
3209h	0	largest sub-index supported for object	UNSIGNED 8	RO	2
	1	PT Ratio	UNSIGNED 16	RW	100
	2	RT Ratio	UNSIGNED 16	RW	1
320Ah	0	largest sub-index supported for object	UNSIGNED 8	RO	3
	1	Set Energy to Zero	UNSIGNED 16	RW	0x0055
	2	Reset to Factory Settings	UNSIGNED 16	RW	0x0055
	3	Default Frequency	UNSIGNED 16	RW	0x0055

5.3 Application Object

The users can write the object the value 65766173h to object with index 1010h and subindex 1 to save the application setting, or write the value 64616F6Ch to object with index 1011h and subindex 1 and reboot the module to load the factory default.