

---

# PISO-CAN Series

(PISO-CAN / PEX-CAN / PCM-CAN)

## CANopen Master Library

### User's Manual

#### **Warranty**

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

#### **Warning**

ICP DAS assume no liability for damages consequent to 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, nor for any infringements of patents or other rights of third parties resulting from its use.

#### **Copyright**

Copyright 2005 by ICP DAS. All rights are reserved.

#### **Trademark**

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

---

## Tables of Content

<b>1</b>	<b>General Information.....</b>	<b>5</b>
1.1	CANopen Introduction.....	5
1.2	CANopen Applications .....	6
1.3	CANopen Master Library Characteristics .....	7
1.4	Features.....	9
<b>2</b>	<b>Software Installation.....</b>	<b>10</b>
2.1	Installation Driver Step by Step .....	11
<b>3</b>	<b>Function Description.....</b>	<b>19</b>
3.1	DLL Function Definition and Description .....	19
3.2	Function Return Code .....	22
3.3	CANopen Master Library Application Flowchart.....	24
3.4	Communication Services Introduction .....	26
3.5	Function Description .....	29
3.5.1	CPM_GetCANDriverVer.....	29
3.5.2	CPM_GetVersion .....	30
3.5.3	CPM_TotalBoard.....	31
3.5.4	CPM_GetCardPortNum .....	32
3.5.5	CPM_GetBoardInf.....	33
3.5.6	CPM_GetCANStatus.....	34
3.5.7	CPM_SetFunctionTimeout.....	35
3.5.8	CPM_InitMaster .....	36
3.5.9	CPM_ShutdownMaster.....	37
3.5.10	CPM_MasterSendBootupMsg .....	38
3.5.11	CPM_SetMasterMode.....	39
3.5.12	CPM_GetMasterMode .....	40
3.5.13	CPM_EDS_Load .....	41
3.5.14	CPM_AddNode .....	43
3.5.15	CPM_RemoveNode .....	45
3.5.16	CPM_RemoveAndResetNode.....	46
3.5.17	CPM_DelayAndResponseTimeout.....	47
3.5.18	CPM_ScanNode.....	48
3.5.19	CPM_GetNodeList.....	49
3.5.20	CPM_NMTChangeState .....	50
3.5.21	CPM_NMTGetState .....	51
3.5.22	CPM_NMTGuarding.....	52
3.5.23	CPM_NMTHeartbeat .....	54

---

3.5.24	CPM_SDOReadData .....	56
3.5.25	CPM_SDOReadFile.....	57
3.5.26	CPM_SDOWriteData .....	58
3.5.27	CPM_SDOAbortTransmit .....	60
3.5.28	CPM_PDOWrite.....	61
3.5.29	CPM_PDOWrite_Fast .....	62
3.5.30	CPM_PDORemote .....	63
3.5.31	CPM_PDORemote_Fast .....	64
3.5.32	CPM_SetPDORemotePolling .....	65
3.5.33	CPM_GetPDOLastData.....	66
3.5.34	CPM_GetMultiPDOData.....	67
3.5.35	CPM_GetRxPDOID .....	68
3.5.36	CPM_GetTxPDOID.....	69
3.5.37	CPM_InstallPDO .....	70
3.5.38	CPM_DynamicPDO.....	71
3.5.39	CPM_RemovePDO.....	73
3.5.40	CPM_ChangePDOID.....	74
3.5.41	CPM_GetPDOMapInfo.....	75
3.5.42	CPM_InstallPDO_List.....	76
3.5.43	CPM_RemovePDO_List .....	78
3.5.44	CPM_PDOWUseEntry.....	79
3.5.45	CPM_PDOTxType .....	80
3.5.46	CPM_PDOWEventTimer .....	81
3.5.47	CPM_PDOWInhibitTime.....	82
3.5.48	CPM_ChangeSYNCCID.....	83
3.5.49	CPM_SetSYNC_List .....	84
3.5.50	CPM_GetSYNCCID.....	85
3.5.51	CPM_SendSYNCCMsg .....	86
3.5.52	CPM_GetCyclicSYNCCInfo .....	87
3.5.53	CPM_ChangeEMCCYID .....	88
3.5.54	CPM_SetEMCCY_List .....	89
3.5.55	CPM_GetEMCCYID .....	90
3.5.56	CPM_ReadLastEMCCY.....	91
3.5.57	CPM_GetBootUpNodeAfterAdd .....	92
3.5.58	CPM_GetEMCCYData .....	93
3.5.59	CPM_GetNMTErrror.....	94
3.5.60	CPM_InstallBootUpISR .....	95
3.5.61	CPM_RemoveBootUpISR .....	96

---

---

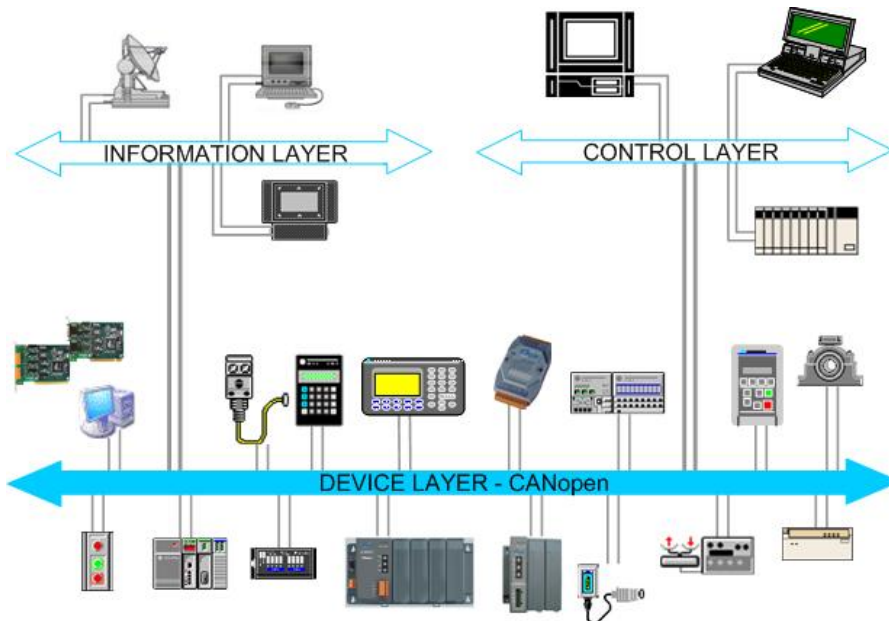
3.5.62	CPM_InstallEMCYISR.....	97
3.5.63	CPM_RemoveEMCYISR .....	98
3.5.64	CPM_InstallNMTErrISR .....	99
3.5.65	CPM_RemoveNMTErrISR .....	100
3.5.66	CPM_GetMasterReadSDOEvent.....	101
3.5.67	CPM_GetMasterWriteSDOEvent .....	102
3.5.68	CPM_ResponseMasterSDO .....	103
3.5.69	CPM_InstallReadSDOISR.....	104
3.5.70	CPM_RemoveReadSDOISR .....	105
3.5.71	CPM_InstallWriteSDOISR .....	106
3.5.72	CPM_RemoveWriteSDOISR.....	107
3.5.73	CPM_GetMasterRemotePDOEvent .....	108
3.5.74	CPM_GetMasterRxPDOEvent.....	109
3.5.75	CPM_ResponseMasterPDO .....	110
3.5.76	CPM_InstallRxPDOISR.....	111
3.5.77	CPM_RemoveRxPDOISR .....	112
3.5.78	CPM_InstallRemotePDOISR .....	113
3.5.79	CPM_RemoveRemotePDOISR.....	114
4	Demo Programs .....	115
4.1	Brief of the demo programs.....	115
4.1.1	Listen_Mode .....	116
4.1.2	NMT_Protocol.....	117
4.1.3	PDO_Parameter .....	118
4.1.4	PDO_Protocol.....	119
4.1.5	Scan_Node.....	120
4.1.6	SDO_PDO_ISR.....	121
4.1.7	SDO_Read.....	122
4.1.8	SDO_Write.....	123
4.1.9	SYNC_Protocol.....	124
4.1.10	PDO_MultiData .....	125

---

# 1 General Information

## 1.1 CANopen Introduction

The CAN (Controller Area Network) is a kind of serial communication protocols, which efficiently supports distributed real-time control with a very high level of security. It is an especially suited for networking intelligent devices as well as sensors and actuators within a system or sub-system. In CAN networks, there is no addressing of subscribers or stations in the conventional sense, but instead, prioritized messages are transmitted. CANopen is one kind of the network protocols based on the CAN bus and it is applied in a low level network that provides the connections between simple industrial devices (sensors, actuators) and higher-level devices (controllers), as shown in the Figure 1.1.



**Figure 1.1 Example of the CANopen network**

CANopen was developed as a standardized embedded network with highly flexible configuration capabilities. It provides standardized communication objects for real-time data (Process Data Objects, PDO), configuration data (Service Data Objects, SDO), network management data (NMT message, and Error Control), and special functions (Time Stamp, Sync message, and Emergency message). Nowadays, CANopen is used in many various application fields, such as medical equipment, off-road vehicles, maritime electronics, public transportation, building automation and so on.

---

## 1.2 CANopen Applications

CANopen is the standardized network application layer optimized for embedded networks. Its specifications cover the standardized application layer, frameworks for the various applications (e.g. general I/O, motion control system, maritime electronics and so forth) as well as device, interface, and application profiles.

The main CANopen protocol and products are generally applied in the low-volume and mid-volume embedded systems. The following examples show some parts of the CANopen application fields. (For more information, please refer to the web site, <http://www.can-cia.org>):

- Truck-based superstructure control systems
- Off-highway and off-road vehicles
- Passenger and cargo trains
- Maritime electronics
- Factory automation
- Industrial machine control
- Lifts and escalators
- Building automation
- Medical equipment and devices
- Non-industrial control
- Non-industrial equipment



---

## 1.3 CANopen Master Library Characteristics

ICP DAS CANopen Master DLL Library provides users to establish CANopen communication network rapidly. It is special for ICPDAS PISO-CAN, PEX-CAN and PCM-CAN series PCI interface cards (**hereinafter referred to as the PISO-CAN series cards**). Using the library, most of the CANopen communication protocols will be handled by the library function automatically. Therefore, it can help users reduce the complexity of developing a CANopen master interface, and let users to ignore the CANopen protocol detail technology information. The library mainly supports the predefined master-slave connection set, which include some useful functions to control the CANopen slave device in the CANopen network. The general application architecture is demonstrated as the Figure 1.2.

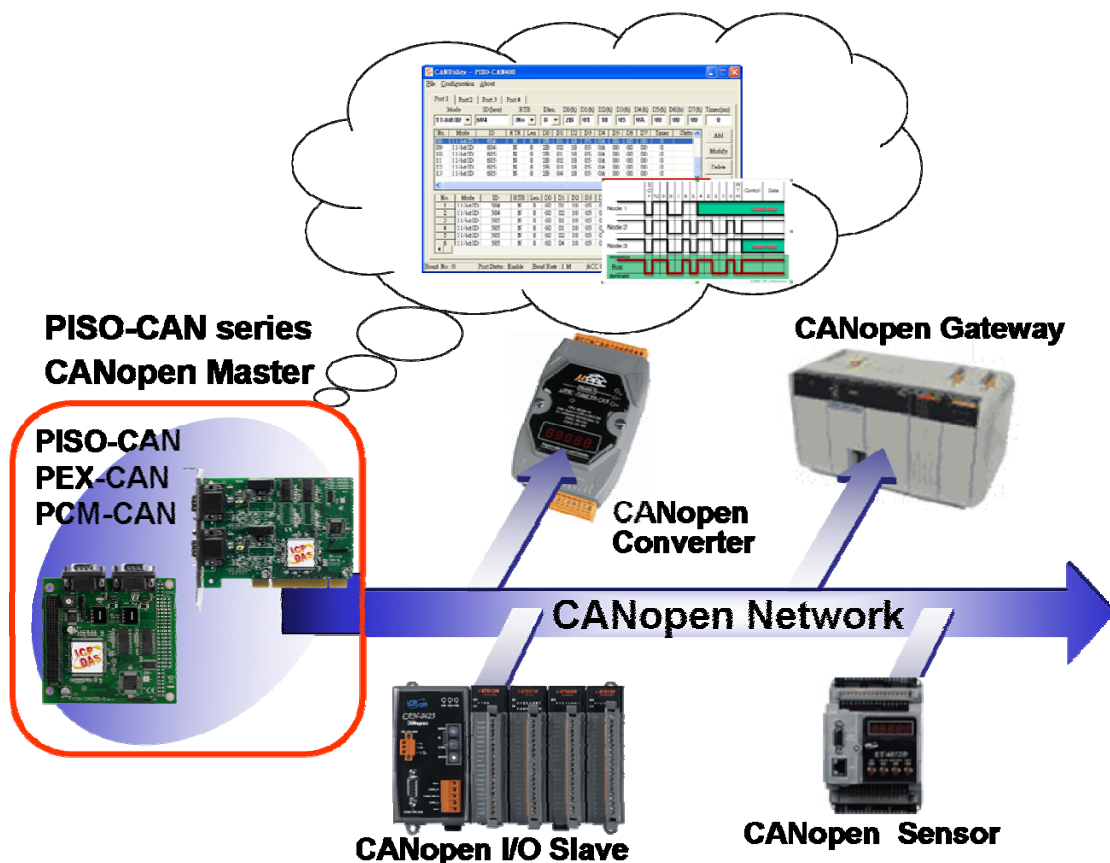


Figure 1.2 Application architecture

The CANopen Master Library follows the CiA CANopen specification DS-301 V4.02, and supports the several CANopen features. The CANopen communication general concept is shown as Figure 1.3.

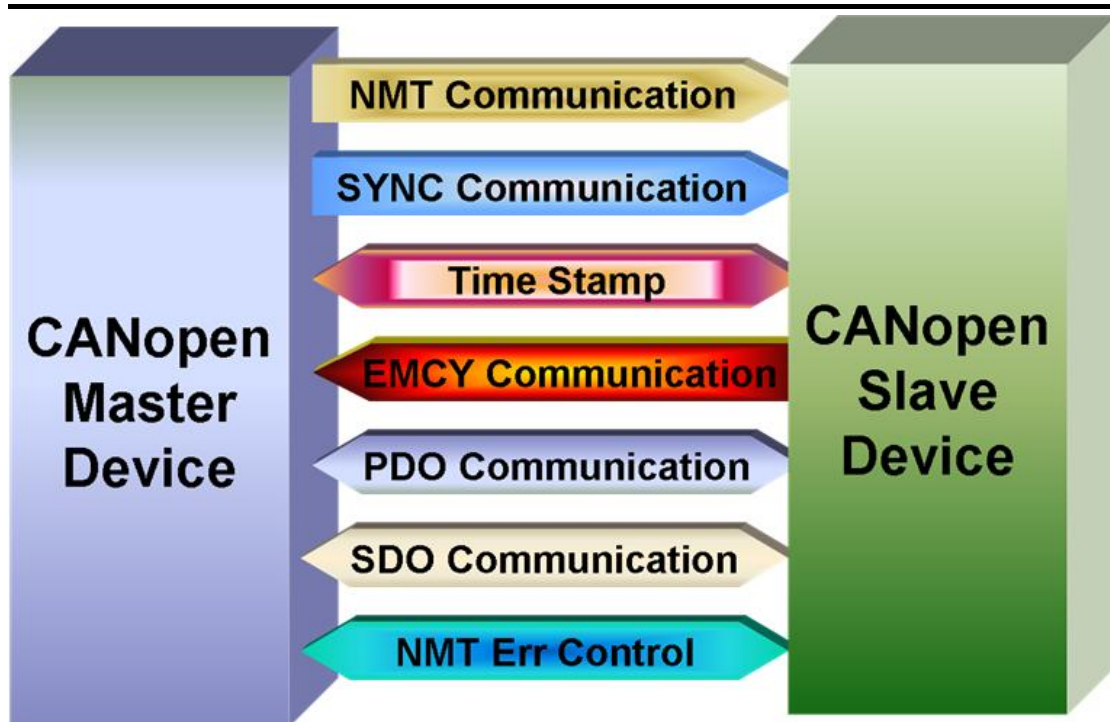


Figure 1.3 CANopen communication general concept

- **Node Manager (NMT Master)**
  - Provide function to change the slave device state
  - Node Guarding Protocol for error control
  - Support Emergency (EMCY) message
- **SDO Manager**
  - Expedited, segmented and block methods for SDO download and upload
- **PDO Manager**
  - Support transmission type and event timer
- **SYNC Manager**
  - SYNC messages production
  - SYNC cycles of 1ms resolution
- **EMCY Manager**
  - EMCY messages consumer

For more information about the CANopen functions described above, please refer to the function descriptions and demo programs shown in the chapter 3 and chapter 4.



---

## 1.4 Features

- Driver supported for Windows 2K/XP/Vista/Win7
- Follow CiA DS-301 V4.02.
- Support 8 kinds baud: 10 kbps, 20 kbps, 50 kbps, 125 kbps, 250 kbps, 500 kbps, 800 kbps, and 1 Mbps.
- Support Node Guarding protocol
- Support the node id range from 1 ~ 127.
- Support upload and download SDO Segment.
- Support Node Guarding protocol and Heartbeat protocol.
- Provide 5 sets of SYNC cyclic transmission.
- Support EDS file.
- Support EMCY protocol.
- Provide Listen Mode to listen the slave status of the CANopen network.
- Block-function and non-block-function selected.
- Demos and utility are provided.
- Library provides VC++, C#.Net2005, and VB.Net2005 developments.

---

## 2 Software Installation

The CANopen DLL driver is the CANopen specification function collections for the PISO-CAN series cards used in Windows 2000/XP/Vista/7 systems. The application structure is presented in the following figure. The users' CANopen master application programs can be developed by the following program development tools: VB, VC++, VB.net, and C#. When users use these program development tools to develop the CANopen master interface, the PISOCANCPM.DLL must be used. Because the PISOCANCPM.DLL calls the function of PISO-CAN driver, the PISO-CAN driver must be installed before using the PISOCANCPM.DLL driver. The driver architecture is shown in the following Figure.

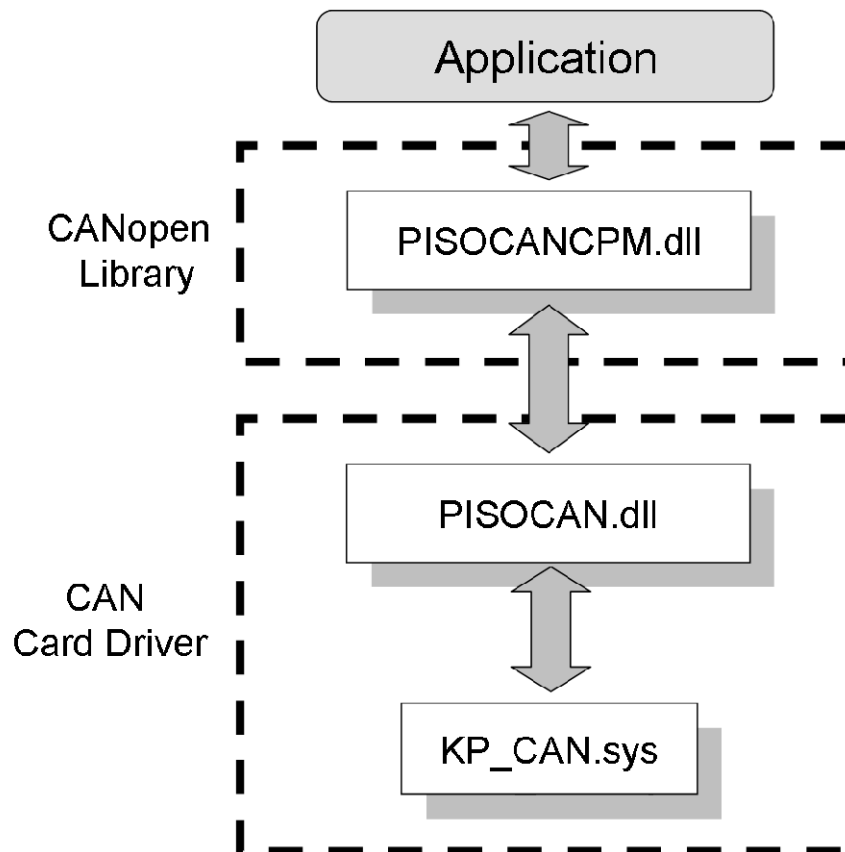


Figure 2.1 Driver concept of CANopen library

---

## 2.1 Installation Driver Step by Step

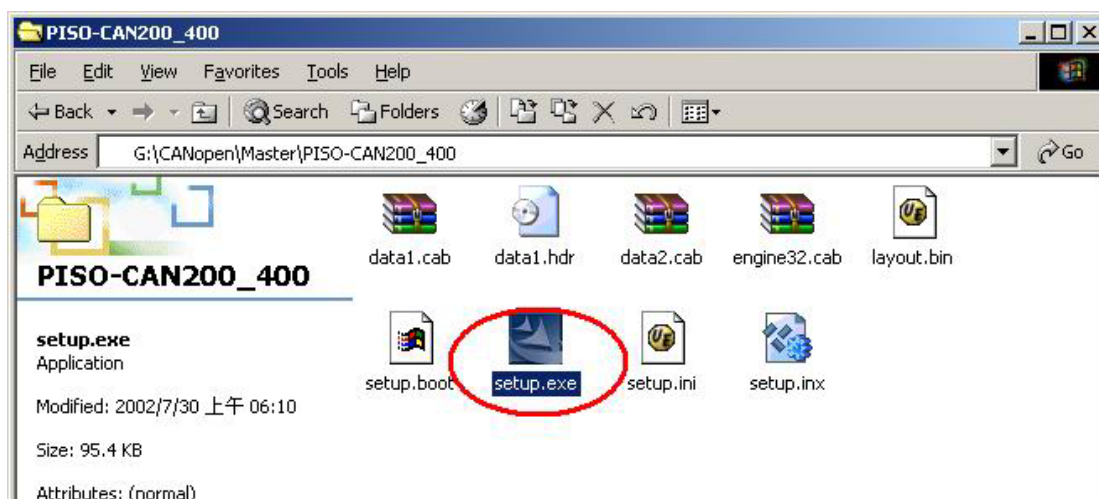
When users want to use the CANopen specification DLL driver, the PISO-CAN series CAN card driver must be installed firstly. Afterwards, users should install the CANopen Master Library. After finishing the installation process, the demo programs may be a good reference for users to build their CANopen master interface by using VC++, C# and VB.Net. The demo programs also give a simple interface to show the basic functions of master/slave connection and CANopen master program architectures. It is very helpful for users to understand how to use these functions and how to develop their CANopen master application. The following description displays the step-by-step procedures about how to install the PISO-CAN driver and PISOCANCPM.DLL driver.

### **Install and Remove the PISO-CAN series CAN card driver**

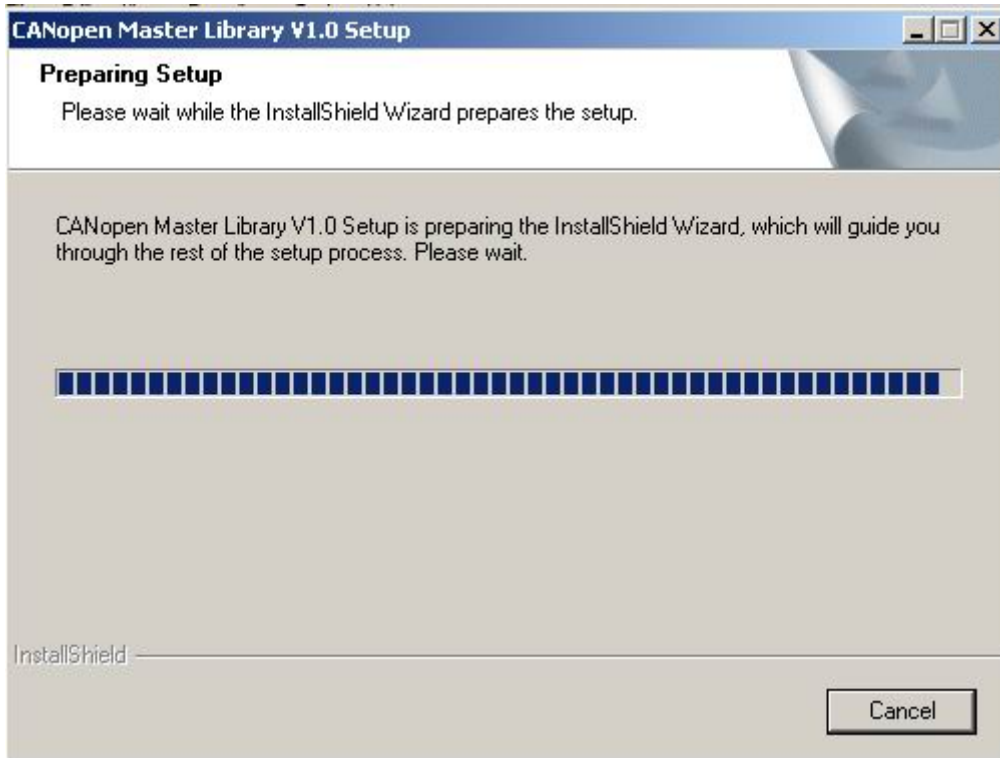
Please refer to the manual `piso-can_user_manual.pdf`. It can be found in the web path [http://ftp.icpdas.com/pub/cd/fieldbus\\_cd/can/pci/pcm\\_piso-can\\_series/](http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/pcm_piso-can_series/) and the CD path `\CAN\PCI\PCM_PISO-CAN_Series\`.

### **Install the CANopen Master Library**

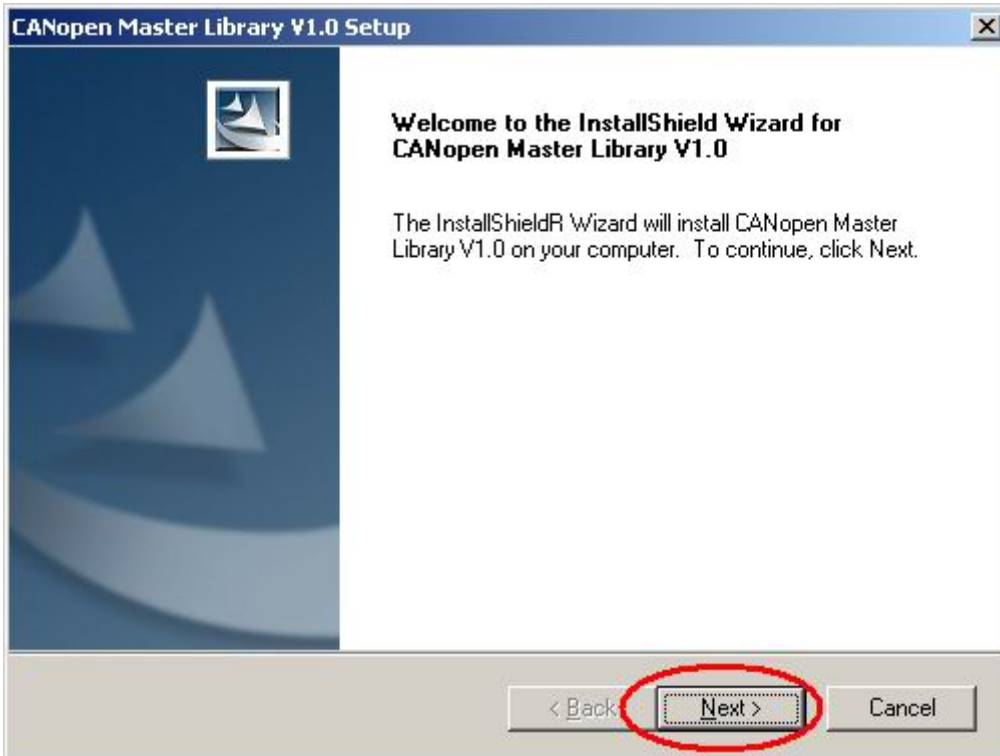
Step 1: Insert the product CD into the CD-ROM and find the path `\CANopen\Master\PISO-CAN_Series`. Then execute the `setup.exe` to install the CANopen Master Library.



Step 2: Wait until the install wizard has prepared.



Step 3: Click "Next" to start the CANopen Master Library installation.



---

Step 4: Select the folder where the CANopen Master Library setup will be installed and click “Next” button to continue.

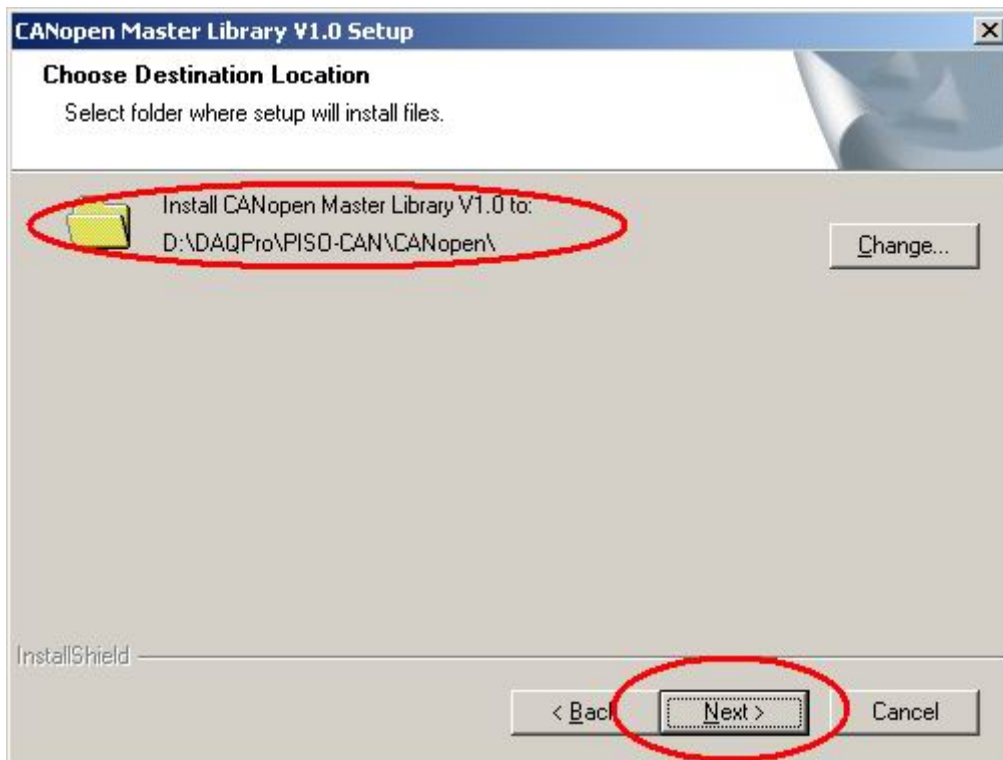
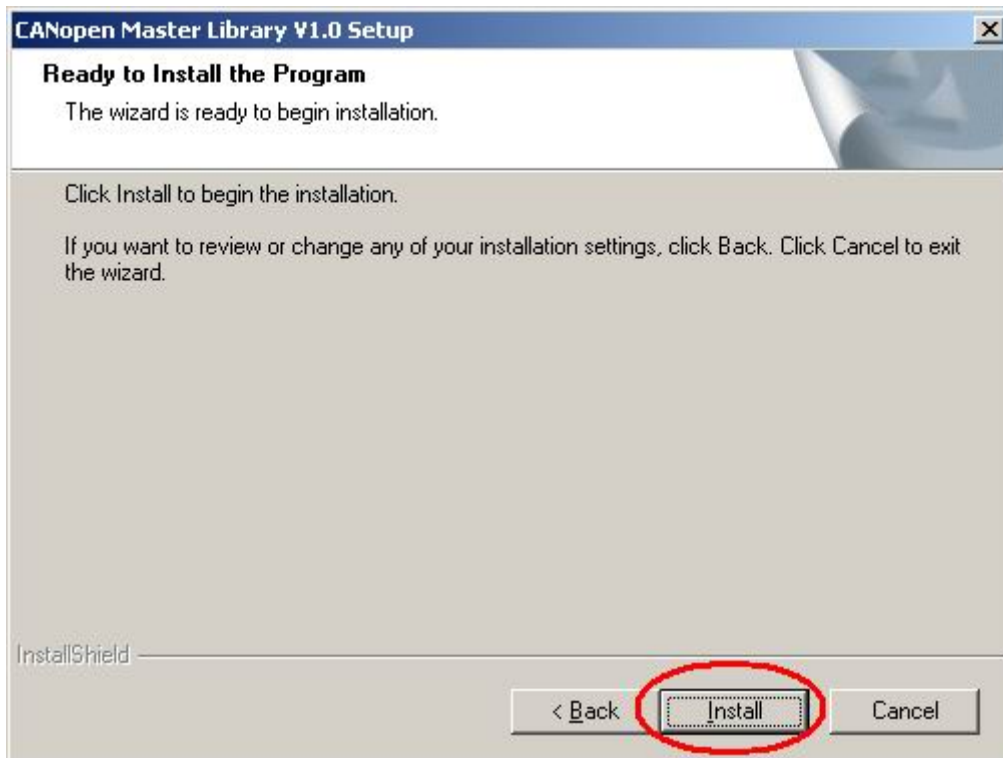


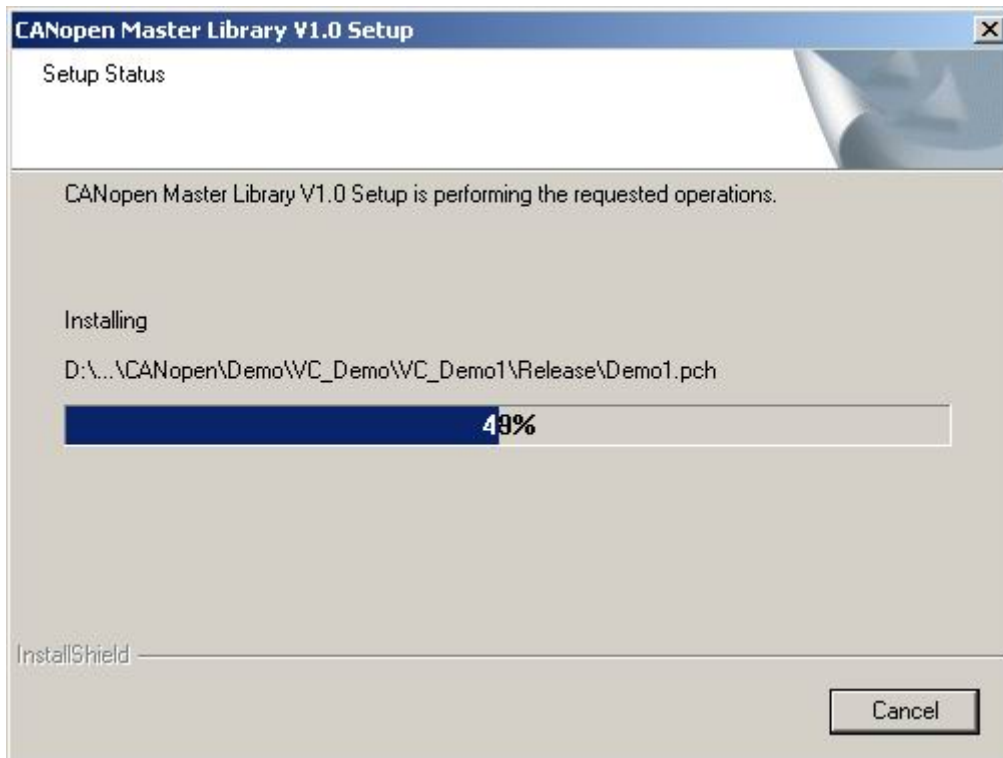
Figure 2.5

Step 5: Click the button “Install” to continue.

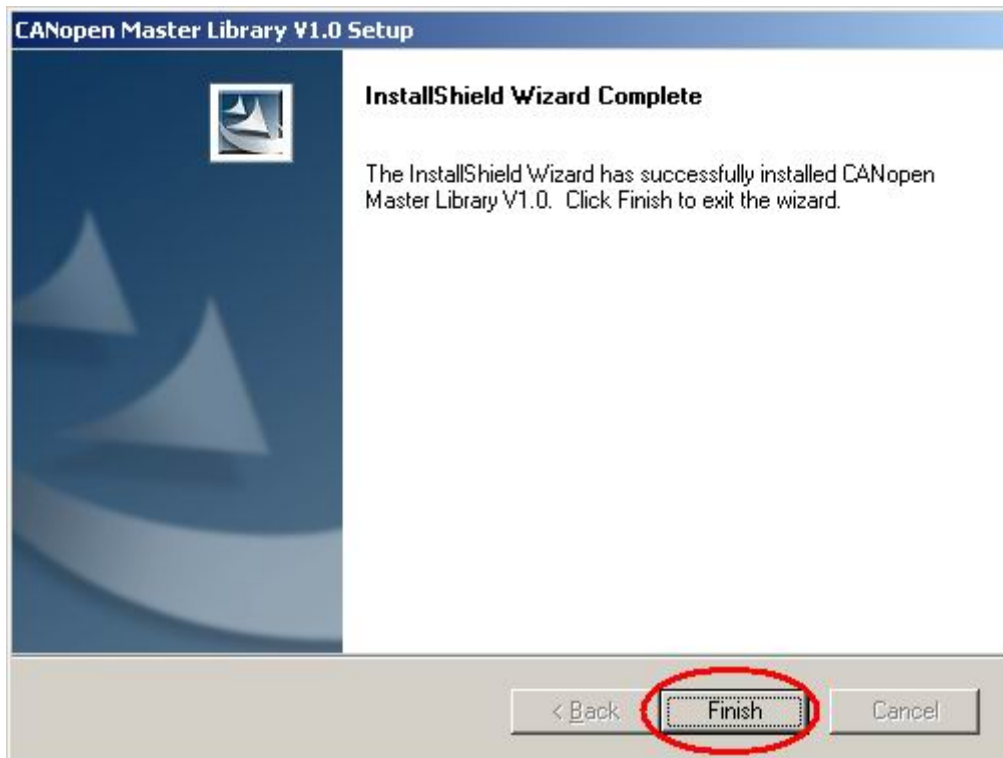


---

Step 6: Wait for the CANopen Master Library installation.

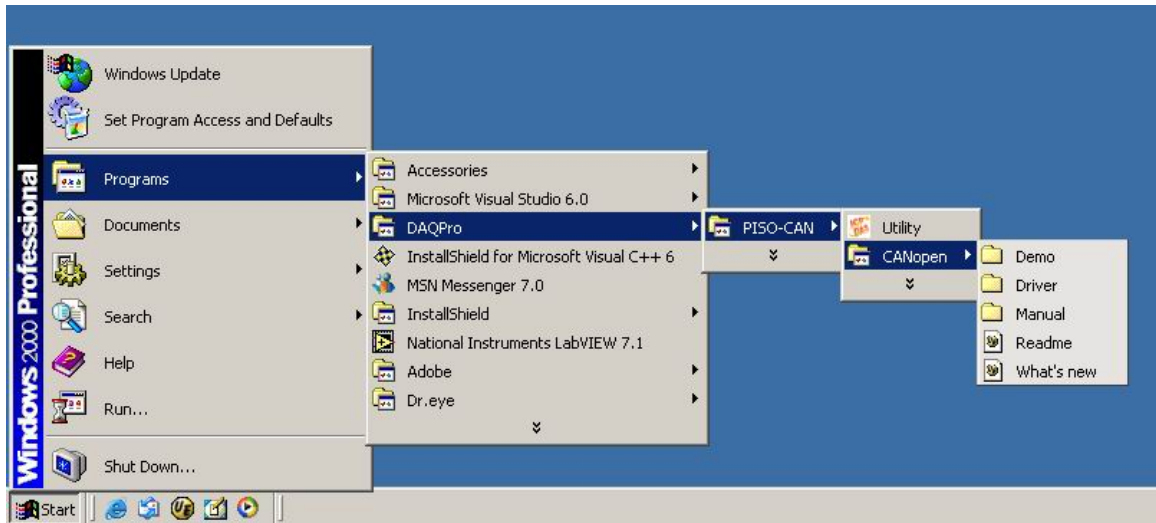


Step 7: After finishing the process, click "Finish" button to complete the installation.



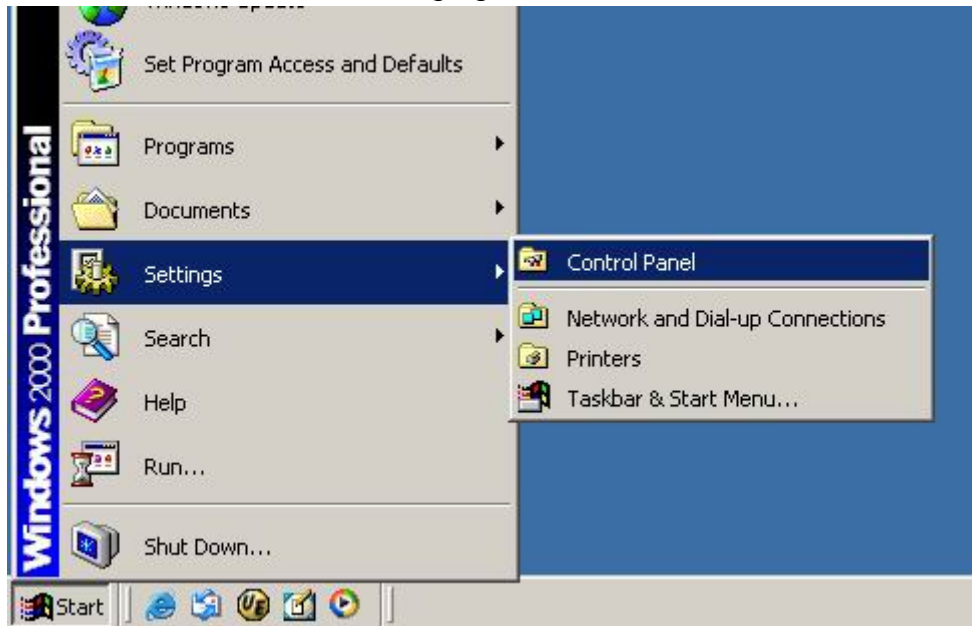
---

When finishing the CANopen Master Library installation. The CANopen folder will be found at the Start menu shown as below.



### **Remove the CANopen Master Library**

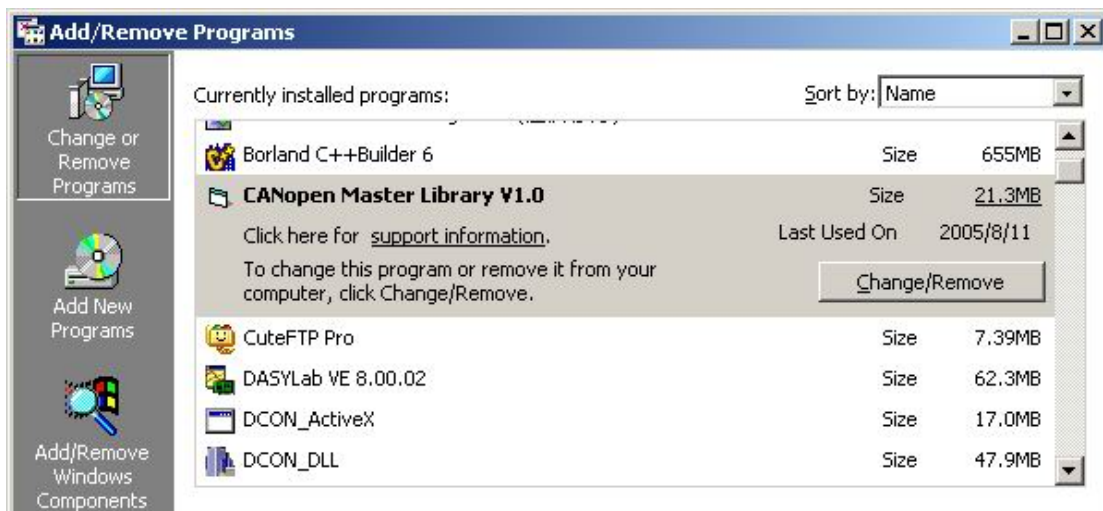
Step 1: Click “Start” in the task bar. Select the Setting/Control Panel as shown in the following figure.



Step 2: Click the “Add/Remove Programs” icon to open the dialog.



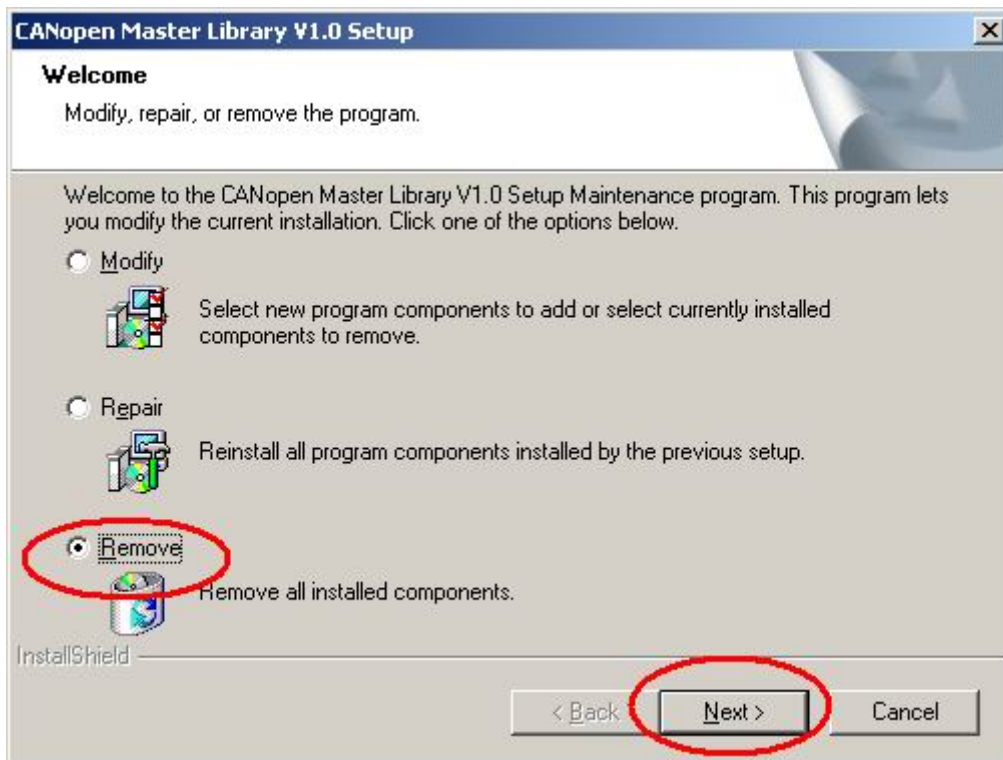
Step 3: Find out the CANopen Master Library, and click the button “Change/Remove”.



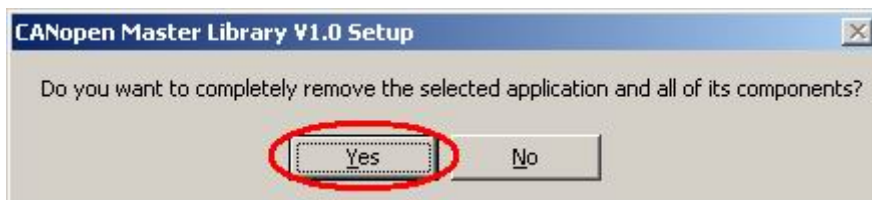


---

Step 4: Choose “Remove” option and click the button “Next” to remove the software.

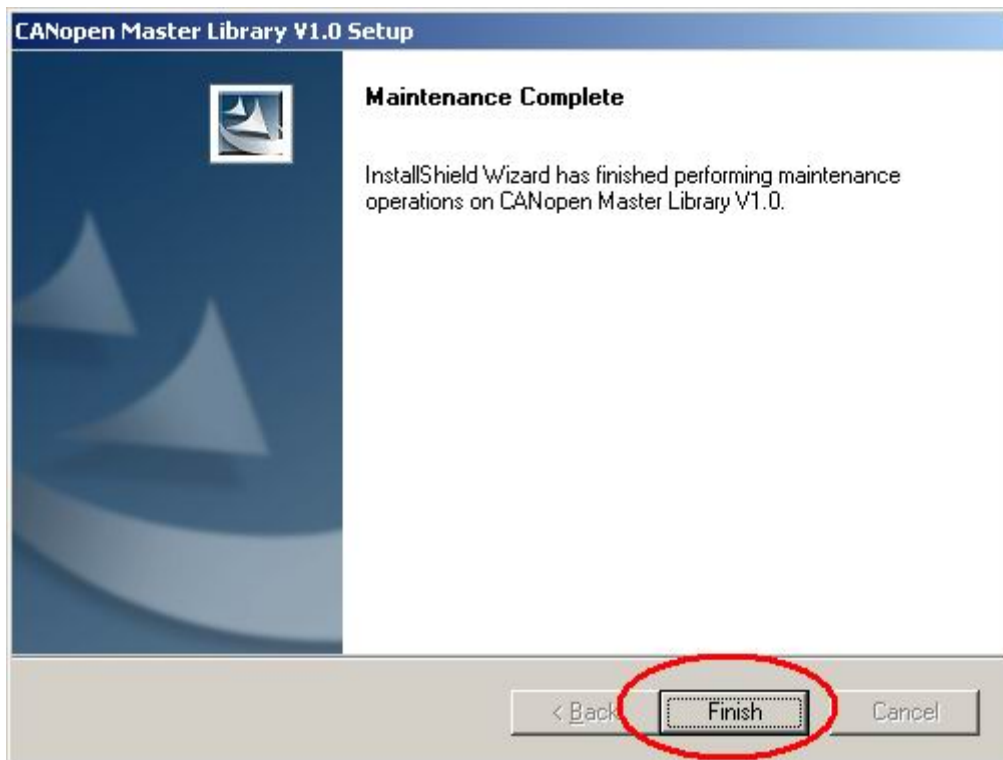


Step 5: Click the button “Yes” to remove the software.



---

Step 6: Finally, click the button “Finish” to finish the uninstall process.



---

## 3 Function Description

### 3.1 DLL Function Definition and Description

All the functions provided by the PISOCANCPM.dll are listed in the following table. The detail information for each function is presented in the section 3.5. In order to make the descriptions more simply and clear, the attributes for the both input and output parameter functions are given as [input] and [output] respectively. They are shown in the table 3.1.

Keyword	Set parameter by user before calling this function?	Get the data from this parameter after calling this function?
[ input ]	Yes	No
[ output ]	No	Yes

Table 3.1

#### Functions Table

Listen Mode	Function Name	Description
O	CPM_GetCANDriverVer	Get the version of the PISO-CAN driver
O	CPM_GetVersion	Get the version of the CPM library
O	CPM_TotalBoard	Get the total number of PISO-CAN series card
O	CPM_GetCardPortNum	Get the CAN port number of the board
O	CPM_GetBoardInf	Get board information of the PISO-CAN card
O	CPM_GetCANStatus	Obtain the status of the CAN controller
O	CPM_SetFunctionTimeout	Set the max. timeout value of all functions
O	CPM_InitMaster	Activate the CAN port of the PISO-CAN card
O	CPM_ShutdownMaster	Remove all nodes and stop the CAN port master
X	CPM_MasterSendBootupMsg	Let the Master sent a boot up message
O	CPM_SetMasterMode	Set the master to normal mode or listen mode
O	CPM_GetMasterMode	Get operation (normal/listen) mode of the master
O	CPM_EDS_Load	Add a slave node from the EDS file
O	CPM_AddNode	Add a slave node into the master manager
O	CPM_RemoveNode	Remove a node from the master manager

X	CPM_RemoveAndResetNode	Remove a node from the master manager and then reset it
O	CPM_DelayAndResponseTimeout	Change minimum time interval and response timeout value of CAN messages
X	CPM_ScanNode	Scan all nodes on the CANopen network
O	CPM_GetNodeList	Get the node list of the master manager
X	CPM_NMTChangeState	Change the CANopen node state
O	CPM_NMTGetState	Get the CANopen node state
O	CPM_NMTGuarding	Start to the node guarding function
O	CPM_NMTHeartbeat	Start to the node heartbeat function
X	CPM_SDOReadData	Read data by using the upload SDO protocol
X	CPM_SDOReadFile	Read the huge SDO data for specific slave node
X	CPM_SDOWriteData	Write data by using the download SDO protocol
X	CPM_SDOAbortTransmit	Send the SDO abort message
X	CPM_PDOWrite	Use the PDO to write data to the CANopen node
X	CPM_PDOWrite_Fast	Only send out the RxPDO but not do any check
X	CPM_PDORemote	Use the PDO to get data from the CANopen node
X	CPM_PDORemote_Fast	Only send the Rtr message but don't check response
X	CPM_SetPDORemotePolling	Set PDO polling list table and poll them
O	CPM_GetPDOLastData	Get the last input or output PDO data
O	CPM_GetMultiPDOData	Get the several input or output PDO data once
O	CPM_GetRxPDOID	Get all COB-ID of the RxPDO of the specific slave
O	CPM_GetTxPDOID	Get all COB-ID of the TxPDO of the specific slave
X	CPM_InstallPDO	Install and enable a specific PDO
X	CPM_DynamicPDO	New or change a PDO mapping to the slave
X	CPM_RemovePDO	Remove a specific PDO mapping entry or object
X	CPM_ChangePDOID	Change the PDO COB-ID of a specific slave
O	CPM_GetPDOMapInfo	Obtain all the PDO-related information
O	CPM_InstallPDO_List	Install a PDO manually without check if it exists
O	CPM_RemovePDO_List	Remove PDO object without check real states
X	CPM_PDOWUseEntry	Change the valid entry number of the PDO objects
X	CPM_PDOWTxType	Set transmission type of the specific TxPDO
X	CPM_PDOWEventTimer	Set event timer of the specific TxPDO
X	CPM_PDOWInhibitTime	Set inhibit time of the specific TxPDO
X	CPM_ChangeSYNCOID	Change the SYNC COB-ID
O	CPM_SetSYNC_List	Set the SYNC COB-ID without check if it exists
O	CPM_GetSYNCOID	Get the SYNC COB-ID

X	CPM_SendSYNCMsg	Send the SYNC message
X	CPM_GetCyclicSYNCInfo	Get all the cyclic sending SYNC information
X	CPM_ChangeEMCYID	Change the EMCY COB-ID
O	CPM_SetEMCY_List	Set the EMCY COB-ID and don't check if it exists
O	CPM_GetEMCYID	Get the EMCY COB-ID
O	CPM_ReadLastEMCY	Get the last EMCY message of the slave
O	CPM_GetBootupNodeAfterAdd	Get boot up message of node slave that had added in node list of the Master.
O	CPM_GetEMCYData	Get the EMCY message from the EMCY buffer
O	CPM_GetNMTErr	Get the NMT Error event from the NMT event buffer
O	CPM_InstallBootUpISR	Install user-defined slave boot up process
O	CPM_RemoveBootUpISR	Remove the slave boot up process
O	CPM_InstallEMCYISR	Install user-defined EMCY process
O	CPM_RemoveEMCYISR	Remove the EMCY process
O	CPM_InstallNMTErrISR	Install user-defined Guarding/Heartbeat event process
O	CPM_RemoveNMTErrISR	Remove the Guarding/Heartbeat event process
O	CPM_GetMasterReadSDOEvent	Get the read SDO message sent to the Master
O	CPM_GetMasterWriteSDOEvent	Get the write SDO message sent to the Master
X	CPM_ResponseMasterSDO	Response the SDO message to the SDO sender
O	CPM_GetMasterRemotePDOEvent	Get the remote PDO message send to the Master
O	CPM_GetMasterRxPDOEvent	Get the RxPDO RTR sent to the Master
X	CPM_ResponseMasterPDO	Response the RxPDO RTR to the RTR sender
O	CPM_InstallRxSDOISR	Install the user-defined Master SDO process
O	CPM_RemoveRxSDOISR	Remove the master SDO process
O	CPM_InstallRxPDOISR	Install the user-defined Write Master PDO process
O	CPM_RemoveRxPDOISR	Remove the Write Master PDO process
O	CPM_InstallTxPDOISR	Install the user-defined Remote Master PDO process
O	CPM_RemoveTxPDOISR	Remove the Remote Master PDO process

**Table 3.2 Description of functions**

---

## 3.2 Function Return Code

The following table interprets all the return code returned by the CANopen Master Library function.

Return Code	Error ID	Description
0	CPM_NoError	OK
1	CPM_DriverError	Kernel driver is not opened
3	CPM_BoardNumberErr	There is no PISO-CAN on the specific board No
4	CPM_PortNumberErr	The CAN port number is over the range
5	CPM_ConfigErr	The PISO-CAN hasn't been configured successfully
6	CPM_MasterInitErr	The Master (CAN port) initialization error
7	CPM_MasterNotInit	The Master (CAN port) hasn't been initialized
8	CPM_ListenMode	The Master is in listen mode now
9	CPM_NodeErr	Set node number error
10	CPM_NodeExist	The node had been added to the Master
11	CPM_AddModeErr	The mode of AddNode function is invalid
12	CPM_TxBusy	Tx buffer is busy, please wait a minute to send again
15	CPM_DataEmpty	There is no data to receive
16	CPM_MemAllocErr	Driver has not enough memory
17	CPM_SendCycMsgErr	Cyclic message send error
18	CPM_StatusErr	NMT state of CANopen slave is error
20	CPM_SetGuardErr	Set Guarding and LifeTime parameter error
21	CPM_SetHbeatErr	Set Heartbeat parameter error
22	CPM_SegLenErr	SDO Segment receive error length
23	CPM_SegToggleErr	SDO Segment receive error toggle
24	CPM_SegWriteErr	SDO write segment error
25	CPM_Abort	The return message is an Abort message
26	CPM_PDOLenErr	PDO output data error
27	CPM_COBIDErr	The COB-ID isn't exist or isn't correct one
28	CPM_PDOLenErr	Install the PDO object error
29	CPM_PDODynaErr	The PDO mapping data is setting error
30	CPM_PDONumErr	The PDO number and COB-ID is not match
31	CPM_PDOSetErr	PDO parameter is setting error
32	CPM_PDOLenErr	The PDO entry parameter is more then useful entry
33	CPM_SetCobIdErr	The EMCY or SYNC COB-ID is setting error

34	CPM_CycFullErr	There are already 5 cyclic message running
35	CPM_Timeout	Message response timeout
36	CPM_DataLenErr	Data length setting error
40	CPM_Wait	Command is uncompleted (only for non-block mode)
41	CPM_Processing	Command is running (only for non-block mode)
50	CPM_LoadEDSErr	Loading the EDS file fails
51	CPM_EDSFormatErr	The format of the EDS file is incorrect

**Table 3.3 Description of return code**

---

### 3.3 CANopen Master Library Application Flowchart

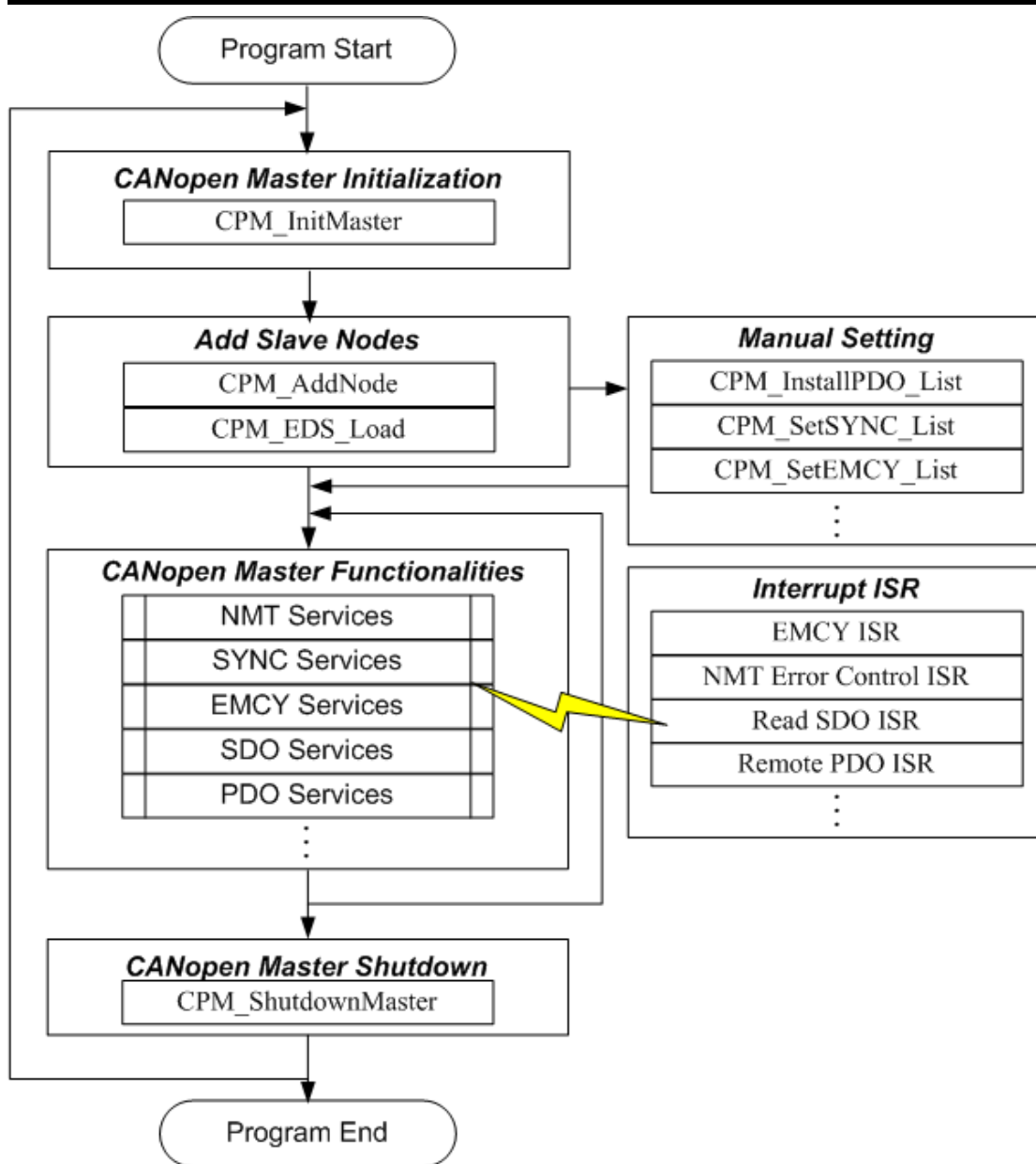
In this section, it describes that the operation procedure about how to use the CANopen Master Library to build users applications. This information is helpful for users to apply the CANopen Master Library easily. Besides, the CANopen operation principles must be obeyed when build a CANopen master application. For example, if the CANopen node is in the pre-operational status, the PDO communication object is not allowed to use. For more detail information, please refer to the demo programs in the section 4.

When users' programs apply the CANopen Master Library functions, the function `CPM_InitMaster` must be called first. The functions is used to initialize PISO-CAN card and configure the CAN port.

After initializing the CAN interface successfully, users need to use the function `CPM_AddNode` or `CPM_EDS_Load` to install at least one CANopen device into the node list. The function `CPM_AddNode` can install slave node automatically or manually. When the user applies the function to install node manually, the PDO ID, SYNC ID, and EMCY ID of the node must be added manually by the functions `CPM_InstallPDO_List`, `CP_SetSYNC_List`, and `CPM_SetEMCY_List`.

If the function `CPM_InitMaster` and `CPM_AddNode/ CPM_EDS_Load` have been executed, the communication services (NMT, SYNC, EMCY, SDO, and PDO services) can be used at any time before calling the functions `CPM_ShutdownMaster`, because the `CPM_ShutdownMaster` would stop all process created by the function `CPM_InitMaster`.





**Figure 3.1 Main programming sequences**

---

## 3.4 Communication Services Introduction

### NMT Services

The CANopen Master Library provides several NMT services functions, such as the functions CPM\_AddNode, CPM\_EDS\_Load, CPM\_RemoveNode, CPM\_NMTChangeState, CPM\_NMTGetState, CPM\_NMTHeartbeat, and CPM\_NMTGuarding. As the prerequisite for the master, the slave nodes have to be registered by the function CPM\_AddNode or CPM\_EDS\_Load with providing its Node-ID. The registered slave nodes can be individually removed from the node list by the function CPM\_RemoveNode. Through NMT services, the NMT Master controls the state of the slaves. Table 3.4 is the command value and corresponding NMT command for the input parameters of the CPM\_NMTChangeState function. When using the function CPM\_NMTGetState, the slave status value and their descriptions are shown in the table 3.5. The Node Guarding and Heartbeat protocol are implemented via the function CPM\_NMTGuarding and the function CPM\_NMTHeartbeat. If the slave nodes are in the node list, users can change the node guarding or heartbeat parameters defined in the slave nodes by calling the function CPM\_NMTGuarding or CPM\_NMTHeartbeat.

Command Value	Description
1 (0x01)	Enter Operational
2 (0x02)	Stop
128 (0x80)	Enter Pre-Operational
129 (0x81)	Reset_Node
130 (0x82)	Reset_Communication

**Table 3.4 NMT Command Specifier**

State of Slave	Description
4 (0x04)	STOPPED
5 (0x05)	OPERATIONAL
127 (0x7F)	PRE-OPERATIONAL

**Table 3.5 State of the Slave**

---

## **SDO Services**

“Initiate SDO download protocol” or “Initiate SDO upload protocol” is used when the object data length  $\leq$  4 bytes. If the object data length  $>$  4 bytes, “Download SDO segment protocol” or “Upload SDO segment protocol” will be used. Calling these two functions, CPM\_SDOReadData and CPM\_SDOWriteData, the initial protocol and segment protocol will be selected automatically according to the object data length.

CPM\_SDOAbortTransmit function can abort a pending SDO transfer at any time. Applying the abort service will have no confirmation from the slave.

## **PDO Services**

After using the CPM\_AddNode to add a slave, the default TxPDOs and RxPDOs (TxPDO 1 ~ 10, RxPDO 1 ~ 10) will also be added to the Master’s control list. If there are PDOs other than the default setting, the function CPM\_InstallPDO is used for enabling these TxPDOs or RxPDOs object. After installing the PDOs, the function CPM\_DynamicPDO can add or change the PDOs’ mapping objects. Each PDO object supports 0~8 application objects. These application objects defined in the CANopen specification CiA401, and they are mapped to the relative parameters of the DI/DO/AI/AO channels. After calling the function CPM\_InstallPDO and CPM\_DynamicPDO, the PDO communication object will be mapped and activated. If the PDO communication object is not needed no more, use the function CPM\_RemovePDO to remove it.

When you want to output data via PDO, the function CPM\_PDOWriteData is useful. This function can write all PDO 8-byte data or write some parts of PDO 8-byte data. Calling this function will send the specific data to the corresponding node via PDO protocol, and put the output data into PDO buffer at the same time. Therefore, you can check the output history of the PDO. But, if the connection between the Master and the slave is lost, the history output information may be not the same with the real status on the slave.

In CANopen specification, you can get the TxPDOs data by applying the remote transmit request CAN frame. The function CPM\_PDORemote is needed in this case. Or you can use CPM\_GetPDOLastData to get the last RxPDO and TxPDO data from the PDO buffer.

---

## **SYNC Services**

Calling the function `CPM_SendSYNCMsg` starts the SYNC object transmission. This function supports single SYNC message transmission and cyclic SYNC message transmission. The parameter "Timer" of the function `CPM_SendSYNCMsg` can adjust the cyclic period of the SYNC COB-ID sent by master. And the parameter "Times" can set the sending times of the SYNC message. If the timer parameter is set to 0, the SYNC object transmission will be stopped. When the times parameter is set to non-zero value, the function will send the SYNC message until the timer is set to 0 or the parameter "Times" is achieved.

## **EMCY Services**

The Emergency messages are triggered by the occurrence of a device internal error situation. Users can call the function `CPM_ReadLastEMCY` to receive the EMCY message or the function `CPM_InstallEMCYISR` to install user-defined EMCY interrupt process. In this case, if there are CAN slaves sending the EMCY, these EMCY messages will be processed by the user-defined EMCY interrupt process.



## 3.5 Function Description

### 3.5.1 CPM\_GetCANDriverVer

- **Description:**

This function is used to obtain the version information of the PISO-CAN driver.

- **Syntax:**

**WORD** CPM\_GetCANDriverVer (**void**)

- **Parameter:**

None

- **Return:**

PISO-CAN driver version information.



---

### 3.5.2 CPM\_GetVersion

- **Description:**

This function is used to obtain the version information of the PISOCANCPM.dll library.

- **Syntax:**

**WORD** CPM\_GetVersion(**void**)

- **Parameter:**

None

- **Return:**

LIB library version information.



### 3.5.3 CPM\_TotalBoard

- **Description:**

Obtain the total board number of PISO-CAN series plugged in the PCI bus.

- **Syntax:**

**WORD** CPM\_TotalBoard(**void**)

- **Parameter:**

None

- **Return:**

Return the scanned total board number of PISO-CAN series.



### 3.5.4 CPM\_GetCardPortNum

- **Description:**

Obtain the CAN port number of the PISO-CAN card.

- **Syntax:**

**WORD** CPM\_GetCardPortNum(**BYTE** BoardNo, **BYTE** \*GetPortNum)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**\*GetPortNum:** [output] To get the CAN port number of the PISO-CAN board.





### 3.5.5 CPM\_GetBoardInf

- **Description:**

Obtain the information of PISO-CAN board, which includes vendor ID, device ID and the interrupt number.

- **Syntax:**

```
WORD CPM_GetBoardInf(BYTE BoardNo, DWORD *dwVID,  
                    DWORD *dwDID, DWORD *dwSVID,  
                    DWORD *dwSDID, DWORD *dwSAuxID,  
                    DWORD *dwIrqNo)
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**\*dwVID:** [output] The address of a variable which is used to receive the vendor ID.

**\*dwDID:** [output] The address of a variable used to receive device ID.

**\*dwSVID:** [output] The address of variable applied to receive sub-vendor ID.

**\*dwSDID:** [output] The address of variable applied to receive sub-device ID.

**\*dwSAuxID:** [output] The address of a variable used to receive sub-auxiliary ID.

**\*dwIrqNo:** [output] The address of a variable used to receive logical interrupt number.



### 3.5.6 CPM\_GetCANStatus

- **Description:**

Obtain the status of the CAN controller of the specific CAN board.

- **Syntax:**

```
int CPM_GetCANStatus(BYTE BoardNo, BYTE Port, BYTE *bStatus)
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**\*bStatus:** [output] The address of a variable is applied to get the status value of CAN controller. The bit interpretation of the bStatus parameter is as below.

Bit	Name	Value	Status
Bit 7	Bus Status	1	Bus-off
		0	Bus-on
Bit 6	Error Status	1	Error
		0	OK
Bit 5	Transmit Status	1	Transmit
		0	Idle
Bit 4	Receive Status	1	Receive
		0	Idle
Bit 3	Transmission Complete Status	1	Complete
		0	Incomplete
Bit 2	Transmit Buffer Status	1	Release
		0	Locked
Bit 1	Data Overrun Status	1	Overrun
		0	Absent
Bit 0	Receive Buffer Status	1	Not Empty
		0	Empty



---

### 3.5.7 CPM\_SetFunctionTimeout

- **Description:**

Sometimes, some function cost more time to complete its task, such as CPM\_ScanNode. If some API of the CPM library never gets the feedback until timeout value goes by, the error code “CPM\_Timeout” will be returned. Through this function, the user can adjust the suitable maximum timeout value of all functions for application. The default timeout value is 1 second.

- **Syntax:**

```
void CPM_SetFunctionTimeout(BYTE BoardNo, BYTE Port,  
                            WORD Timeout)
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Timeout:** [input] Maximum timeout value of all functions (ms).



### 3.5.8 CPM\_InitMaster

- **Description:**

The function must be applied when configuring the CAN controller and initialize the CAN board. It must be called once before using other functions of the CPM library.

- **Syntax:**

**WORD** CPM\_InitMaster(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **BYTE** BaudRate, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Master's node ID. If the parameter is 0, the master will be a normal master, and no other master can control it. If the parameter is 1 ~ 127 (different from other slave), other master can do some control to it through some ISR function.

**BaudRate:** [input] The baud rate of the CAN port.

Value	0	1	2	3	4	5	6	7
<b>Baud rate (bps)</b>	10k	20k	50k	125l	250k	500k	800k	1M

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.9 CPM\_ShutdownMaster

- **Description:**

The function CPM\_ShutdownMaster removes all the slaves that had added to master and stop all the functions of the CAN port. The function must be called before exit the users' application programs.

- **Syntax:**

**WORD** CPM\_ShutdownMaster (**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

### 3.5.10 CPM\_MasterSendBootupMsg

- **Description:**

To use the function CPM\_MasterSendBootupMsg can let the Master sends a boot up message after CPM\_InitMaster is called.

**Note:** The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.

- **Syntax:**

**WORD** CPM\_MasterSendBootupMsg (**BYTE** BoardNo, **BYTE** Port, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.11 CPM\_SetMasterMode

- **Description:**

This function can configure if the master is into listen mode or normal mode (default mode). In listen mode, the Master can't send any CANopen message, and some functions will be useless in this mode. User can select normal mode or listen mode at any time after calling function CPM\_InitMaster.

- **Syntax:**

**WORD** CPM\_SetMasterMode(**BYTE** BoardNo, **BYTE** Port,  
**BYTE** Mode, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Mode:** [input] 1 is listen mode, and others are normal mode (default mode).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.12 CPM\_GetMasterMode

- **Description:**

If user want to know what operation mode of the master is, call the function to get it.

- **Syntax:**

**WORD** CPM\_GetMasterMode(**BYTE** BoardNo, **BYTE** Port,  
**BYTE** \*Mode, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**\*Mode:** [output] 0 is normal mode (default mode), and 1 is listen mode.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".





### 3.5.13 CPM\_EDS\_Load

- **Description:**

The function CPM\_EDS\_Load can let users load EDS file for adding a CANopen slave with specified Node ID into the master node list. Using this function will not send any message to check if the slave is existent or not.

- **Syntax:**

**WORD** CPM\_EDS\_Load (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **char** \*FilePath, **WORD** DelayTime, **WORD** ResTimeout, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*FilePath:** [input] Absolute or relative file path of the EDS file.

**DelayTime:** [input] The parameter defines the time interval between sending two messages from the PISO-CAN. It can avoid the master frequently sending messages that may overrun the buffer of the slave. Too large value of this parameter reduces the performance of the PISO-CAN. The unit of the parameter is ms. This parameter will be applied to the specified slave after finishing the ESD loading.

**ResTimeout:** [input] The timeout value of the response of the CANopen slaves. When the master sends a CANopen message to the slave, it will wait the response until timeout if there is a response. The unit of this parameter is millisecond. This parameter will be applied to the specified slave after finishing the ESD loading.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 is block-function. If set this parameter to 1, the running procedure of the users' application is held until finishing this function. If 0, this function returns "CPM\_ Processing" directly. While users apply it



---

with the same “BoardNo”, “Port” again, it returns the process status.  
If the procedure is still not complete, it returns “CPM\_Wait”.



### 3.5.14 CPM\_AddNode

- **Description:**

The function CPM\_AddNode can add a CANopen slave with specified Node ID into the master node list. There are three mode of the function. Mode 1 is adding node automatically, mode 2 is adding node manually, and mode 3 is allowing the automatic adding the node while it boots up. In the automatic mode, after calling this function to add a slave, the slave will be into the operational state directly. And master will scan the TxPDO1 ~ TxPDO10 and RxPDO1 ~ RxPDO10 and install them into the Master if the slave supports these PDO objects. In the manual mode, this function will add a CANopen slave into the master node list only, and will not send any message to check if the slave is existent or not. Besides, the manual mode doesn't install the SYNC ID, EMCY ID, and any PDO object into the Master. Users must call CPM\_SetSYNC\_List, CPM\_SetEMCY\_List, and CPM\_InstallPDO\_List to complete the object installation to finish the adding node process.

The added node can be removed from the master node list by the function CPM\_RemoveNode.

- **Syntax:**

**WORD** CPM\_AddNode(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **BYTE** AddMode, **WORD** DelayTime, **WORD** ResTimeout, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**AddMode:** [input] 1: Automatic mode. 2: Manual mode. 3: Wait Boot-up mode.

**DelayTime:** [input] The parameter is the shortest time interval between sending two messages from the PISO-CAN. It can avoid the master to send too much CANopen messages in a



---

short time that may let some slaves occur the errors. But if the delay time is too long, the performance of the PISO-CAN is down. The unit of the parameter is ms.

**ResTimeout:** [input] The timeout value of the responded messages from the CANopen slaves. When the master sends a CANopen message to the slave, it will wait the feedback until timeout if there is a feedback. The unit of this parameter is millisecond.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.15 CPM\_RemoveNode

- **Description:**

The function CPM\_RemoveNode removes the slave with the specified Node-ID from node list of the master. It requires a valid Node-ID, which has installed by the function CPM\_AddNode before.

- **Syntax:**

**WORD** CPM\_RemoveNode(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.16 CPM\_RemoveAndResetNode

- **Description:**

The function CPM\_RemoveAndResetNode removes the slave with the specified Node-ID from node list of the master and reset the slave. It requires a valid Node-ID, which has installed by the function CPM\_AddNode before.

- **Syntax:**

**WORD** CPM\_RemoveAndResetNode (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".

### 3.5.17 CPM\_DelayAndResponseTimeout

- **Description:**

Call the function to change CAN message minimum interval time and response timeout value of the slave node at any time.

- **Syntax:**

**WORD** CPM\_DelayAndResponseTimeout (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** DelayTime, **WORD** ResTimeout, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**DelayTime:** [input] The parameter is the same as “DelayTime” of the function CPM\_AddNode.

**ResTimeout:** [input] The parameter is the same as “ResTimeout” of the function CPM\_AddNode.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.



### 3.5.18 CPM\_ScanNode

- **Description:**

User can use the function CPM\_ScanNode to know how many slave nodes are on the CANopen network.

- **Syntax:**

**WORD** CPM\_ScanNode(**BYTE** BoardNo, **BYTE** Port, **BYTE** S\_Node, **BYTE** E\_Node, **BYTE** \*NodeList, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**S\_Node:** [input] Start node ID.

**E\_Node:** [input] End node ID. This function will scan node ID from S\_Node to E\_Node. If S\_Node >= E\_Node, it will scan all slave node ID (1 ~ 127) on the CANopen network.

**\*NodeList:** [output] This is a 16-byte array parameter. Each bit of the parameter means a node ID, the bit is 0 means that the node ID doesn't exist and 1 means the node ID is on the CANopen network. For example, if the NodeList[0] is 0x16 (0001 0110), it means that the nodes with ID 1, 2, and 4 exist, and the nodes with ID 0, 3, 5, 6, and 7 don't exist. If the NodeList[1] is 0x41 (0100 0001), it means that the nodes with ID 8 and 14 exist, and the nodes with ID 9, 10, 11, 12, 13, and 15 don't exist.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".





### 3.5.19 CPM\_GetNodeList

- **Description:**

User can use the function CPM\_GetNodeList to know how many slave nodes are added to the node list of the CAN Master.

- **Syntax:**

**WORD** CPM\_GetNodeList(**BYTE** BoardNo, **BYTE** Port,  
**BYTE** \*NodeList, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**\*NodeList:** [output] This is a 16-byte array parameter. Each bit of the parameter means a node ID, the bit is 0 means the node ID not exist and 1 means the node ID now on the CANopen bus. For example, if the NodeList[0] is 0x16 (0001 0110), it means that the nodes with ID 1, 2, and 4 have been added into node list, and the nodes with ID 0, 3, 5, 6, and 7 don't. If the NodeList[1] is 0x41 (0100 0001), it means that the nodes with ID 8 and 14 have been added into node list, and the nodes with ID 9, 10, 11, 12, 13, and 15 don't.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.20 CPM\_NMTChangeState

- **Description:**

The function CPM\_NMTChangeState is used to change the NMT state of a slave. If the node parameter of this function is set to 0, the state of all nodes on the same CANopen network will be changed.

- **Syntax:**

**WORD** CPM\_NMTChangeState(**BYTE** BoardNo, **BYTE** Port,  
**BYTE** Node, **BYTE** State, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127). Set this parameter to 0 to indicate all slave devices.

**State:** [input] NMT command specifier.

**1:** start

**2:** stop

**128:** enter PRE-OPERATIONAL

**129:** Reset\_Node

**130:** Reset\_Communication

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.21 CPM\_NMTGetState

- **Description:**

The function CPM\_NMTGetState can get the NMT state from slaves.

- **Syntax:**

**WORD** CPM\_NMTGetState(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**BYTE** \*State, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*State:** [output] The NMT state of the slave.

**4:** STOPPED

**5:** OPERATIONAL

**127:** PRE-OPERATIONAL

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.22 CPM\_NMTGuarding

- **Description:**

Use the function CPM\_NMTGuarding to set guard time and life time factor of the slave with the specified node ID. Then, the master will automatically send the guarding message to slave device according to the “GuardTime” parameters of this function. If the master doesn’t receive the confirmation of guarding message from the slave, the CAN Master will produce a Node\_Guarding\_Event event to users. Users may use the function CPM\_InstallNMTErrISR to install a user-defined process to get the guarding fail event and process the guarding fail procedure. However, if the slave doesn’t receive the guarding message during the Node Life time period (Node Life time = “GuardTime” \* “LifeTime”), it will be triggered to send the EMCY message. It is recommended that “LifeTime” value is set to more than 1.

Take a note that following the CANopen specification, it is not allowed for one slave device to use both error control mechanisms Guarding Protocol and Heartbeat Protocol at the same time.

- **Syntax:**

**WORD** CPM\_NMGuarding(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** GuardTime, **BYTE** LifeTime, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**GuardTime:** [input] Guard Time (1 ~ 65535 ms).

**LifeTime:** [input] Life Time Factor (1 ~ 255).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users’ application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status



---

while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.



### 3.5.23 CPM\_NMTHeartbeat

- **Description:**

Use the function CPM\_NMTHeartbeat to set heartbeat time of the slave with the specified node ID and consume time with the Master. Then, the slave will automatically send the heartbeat message to master according to the “ProduceTime” parameters of this function. If the master doesn’t receive the heartbeat message from the slave until the “ConsumeTime” time (unit is millisecond) is up, the Master will produce a “Heartbeat\_Event” event to users. Users may use the function CPM\_InstallNMTErrISR to install a user-defined process to get the heartbeat fail event and process the heartbeat fail procedure. It is recommended that “ConsumeTime” value is set to bigger than the “ProduceTime” value.

Take a note that following the CANopen specification, it is not allowed for one slave device to use both error control mechanisms Guarding Protocol and Heartbeat Protocol at the same time.

- **Syntax:**

**WORD** CPM\_NMHartbeat(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** ProduceTime, **WORD** ConsumeTime, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**ProduceTime:** [input] Produce Time of slave device (1 ~ 65535 ms).

**ConsumeTime:** [input] Consume Time of master (1 ~ 65535 ms).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users’ application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status



---

while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.



### 3.5.24 CPM\_SDOReadData

- **Description:**

The function CPM\_SDOReadData is useful to the SDO upload from a specified slave. When users use this function, pass the slave device node ID, object index and object subindex into this function. This function supports both expedition mode (less than 4-byte data length) and segment mode (more than 4-byte data length).

- **Syntax:**

**WORD** CPM\_SDOReadData (**BYTE** BoardNo, **BYTE** Port,  
**BYTE** Node, **WORD** Index, **BYTE** SubIndex,  
**DWORD** \*RDLen, **BYTE** \*RData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Index:** [input] Object index of object dictionary of slave devices.

**SubIndex:** [input] Object subindex of object dictionary of slave devices.

**\*RDLen:** [output] Total data length.

**\*RData:** [output] SDO data respond from the specified slave device.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".





---

### 3.5.25 CPM\_SDOReadFile

- **Description:**

The function CPM\_SDOReadFile is useful as uploading the SDO data more than 1024 bytes. While users use the CPM\_ReadData to read the SDO data and the return data length is more than 1024 byte. The SDO data are stored in a file. Users can use the function CPM\_SDOReadFile for reading the SDO data from the file.

- **Syntax:**

**WORD** CPM\_SDOReadFile (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Index, **BYTE** SubIndex, **DWORD** Start, **DWORD** Len, **DWORD** \*RLen, **BYTE** \*RData)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Index:** [input] Object index of object dictionary of slave devices.

**SubIndex:** [input] Object subindex of object dictionary of slave devices.

**Start:** [input] Start position for reading the SDO data from file. The range is from 0 to maximum length.

**Len:** [input] Data length for reading the SDO data.

**\*RDLen:** [output] Returned data length in reality.

**\*RData:** [output] The SDO data respond from the specified slave device.



### 3.5.26 CPM\_SDOWriteData

- **Description:**

The function CPM\_SDOWriteData can send out a SDO message to the specified slave device. This procedure is also called download SDO protocol. The parameter node of the function CPM\_SDOWriteData is used to point which slave device will receive this SDO message. Because the data length of each object stored in object dictionary is different, users need to know the data length when writing the object of the object dictionary of the specified slave devices. This function supports both expedition mode (less than 4-byte data length) and segment mode (more than 4-byte data length)

- **Syntax:**

**WORD** CPM\_SDOWriteData (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Index, **BYTE** SubIndex, **DWORD** TDLen, **BYTE** \*TData, **WORD** \*RDLen, **BYTE** \*RData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Index:** [input] The index value of object of the object dictionary.

**SubIndex:** [input] The subindex value of object of the object dictionary.

**TDLen:** [input] Total data size to be written.

**\*TData:** [input] The SDO data written to slave device.

**\*RLen:** [output] Total data size of responded data.

**\*RData:** [output] SDO data responded from the specified slave device.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status



---

while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.



### 3.5.27 CPM\_SDOAbortTransmit

- **Description:**

Call the function CPM\_SDOAbortTransmit to cancel the SDO transmission. The parameter node of this function is used to specify which SDO communication will be terminated between the master and the specified slave device.

- **Syntax:**

**WORD** CPM\_SDOAbortTransmit(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Index, **BYTE** SubIndex, **DWORD** \*AData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Index:** [input] The object index value of the object dictionary.

**SubIndex:** [input] The object subindex value of the object dictionary.

**\*AData:** [input] Abort data to be send to slave.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.28 CPM\_PDOWrite

- **Description:**

Call the function CPM\_PDOWrite to send out a PDO message to the specified slave device. Before using this function, users need to use the function CPM\_InstallPDO to install the PDO object into the Master if users want to use non-default PDO. Then, change the NMT state of the target slave device to operational mode by using the function CPM\_NMTChangeState if the slave is not in the operational mode. Use the parameter offset to set the start position of the PDO data, and use the parameters “\*TData” and “DLen” to point the data and data length. Then, this function will follow the data length to cover the slave PDO buffer of the Master with the data from the specified start position, and send the data to the specified slave via PDO protocol at the same time.

- **Syntax:**

**WORD** CPM\_PDOWrite (**BYTE** BoardNo, **BYTE** Port, **WORD** Cobid, **BYTE** Offset, **BYTE** DLen, **BYTE** \*TData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Offset:** [input] The start byte position of PDO data (0 ~ 7).

**DLen:** [input] data size (dlen + offset ≤ 8 (total length of the PDO)).

**\*TData:** [output] The data pointer point to the PDO data.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Cobid” again. If the procedure is still not complete, it will return “CPM\_Wait”.



---

### 3.5.29 CPM\_PDOWrite\_Fast

- **Description:**

The function is like CPM\_PDOWrite but does not check whether the PDO message really sends to CAN bus or not. So CPM\_PDOWrite\_Fast is about twice as faster than CPM\_PDOWrite at high speed baud rate (greater than or equal to 250kbps).

- **Syntax:**

**WORD** CPM\_PDOWrite\_Fast (**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **BYTE** Offset, **BYTE** DLen,  
**BYTE** \*TData)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Offset:** [input] The start byte position of PDO data (0 ~ 7).

**DLen:** [input] data size (dlen + offset  $\leq$  8 (total length of the PDO)).

**\*TData:** [output] The data pointer point to the PDO data.

### 3.5.30 CPM\_PDORemote

- **Description:**

Use the function CPM\_PDORemote to send a RTR (remote-transmit-request) PDO message to the slave device.

- **Syntax:**

**WORD** CPM\_PDORemote (**BYTE** BoardNo, **BYTE** Port, **WORD** Cobid, **BYTE** \*DLen, **BYTE** \*TData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**\*DLen:** [output] The data length of the RTR PDO message.

**\*TData:** [output] The PDO message received from the slave device.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".

### 3.5.31 CPM\_PDORemote\_Fast

- **Description:**

Use the function CPM\_PDORemote\_Fast only to send a RTR (remote-transmit-request) PDO message to the slave device but not wait for the response message.

**Note:** This function usually is used with CPM\_GetPDOLastData.

- **Syntax:**

**WORD** CPM\_PDORemote\_Fast (**BYTE** BoardNo, **BYTE** Port, **WORD** Cobid)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.





### 3.5.32 CPM\_SetPDORemotePolling

- **Description:**

If the CANopen slaves do not support the event timer function of the TxPDOs, using the function CPM\_SetPDORemotePolling can config the most 125 TxPDO objects into the remote polling list. Then, the PISO-CAN will poll the configured TxPDOs and update the data into buffer automatically. Users can use CPM\_GetMultiPDOData to get these TxPDOs data from the buffer faster and easily.

- **Syntax:**

**WORD** CPM\_SetPDORemotePoling (**BYTE** BoardNo, **BYTE** Port, **BYTE** PDOCnt, **WORD** \*Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**PDOPCnt:** [input] The number of the \*Cobid array.

**\*Cobid:** [input] COB-ID array used by the TxPDO objects.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.33 CPM\_GetPDOLastData

- **Description:**

Using the function CPM\_GetPDOLastData can get the last data of the RxPDO and TxPDO from the PDO data buffer. The last PDO data is saved in PDO buffer, so it may not be the same with the real situation.

- **Syntax:**

**WORD** CPM\_GetPDOLastData (**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **BYTE** \*IsNew, **BYTE** \*DLen,  
**BYTE** \*RData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**\*IsNew:** [output] Check the data if had been read before. 0 is been read before, and 1 is new one.

**\*DLen:** [output] The data length of the PDO message.

**\*RData:** [output] The PDO message gets from the PDO buffer.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.34 CPM\_GetMultiPDOData

- **Description:**

This can get the last data of the RxPDO and TxPDO from the PDO data buffer such as the function CPM\_GetPDOLastData. But the difference between these two functions is that user can use the function CPM\_GetMultiPDOData to get maximum 50 PDO data at the same time.

- **Syntax:**

**WORD** CPM\_GetMultiPDOData (**BYTE** BoardNo, **BYTE** Port, **BYTE** PDOCnt, **WORD** \*Cobid, **BYTE** \*IsNew, **BYTE** \*DLen, **BYTE** \*RData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**PDOPCnt:** [input] Total PDO number that want to get.

**\*Cobid:** [input] Maximum 50 COB-ID used by the PDO objects.

**\*IsNew:** [output] Check these PDO data if they have been read before.  
0 is to be read before, and 1 is new one.

**\*DLen:** [output] The total data length obtained from the PDO buffer.

**\*RData:** [output] The PDO messages get from the PDO buffer.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".

### 3.5.35 CPM\_GetRxPDOID

- **Description:**

Use the function CPM\_GetRxPDOID to get all the RxPDO COB-IDs of the specified slave, which have been installed to the master.

- **Syntax:**

**WORD** CPM\_GetRxPDOID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **BYTE** \*PDO\_Cnt, **WORD** \*ID\_List, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*PDO\_Cnt:** [output] The number of installed RxPDO.

**\*ID\_List:** [output] The RxPDO COB-ID list.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".

### 3.5.36 CPM\_GetTxPDOID

- **Description:**

Use the function CPM\_GetTxPDOID to get all the TxPDO COB-IDs of the specified slave, which have been installed to the master.

- **Syntax:**

**WORD** CPM\_GetTxPDOID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **BYTE** \*PDO\_Cnt, **WORD** \*ID\_List, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*PDO\_Cnt:** [output] The number of installed TxPDO.

**\*ID\_List:** [output] The TxPDO COB-ID list.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.37 CPM\_InstallPDO

- **Description:**

After calling the CPM\_InstallPDO function, a PDO COB-ID will be installed in the PDO object list of the CANopen Master Library stack. If the slave device has defined the default PDO object in RxPDO1 ~ RxPDO10 and TxPDO1 ~ TxPDO10, in this case, these default PDO will be installed automatically while using the function CPM\_AddNode with automatic mode. Otherwise, the TxPDOs or RxPDOs need to be installed manually by calling the function CPM\_InstallPDO.

- **Syntax:**

**WORD** CPM\_InstallPDO(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**WORD** Cobid, **BYTE** RxTx,  
**WORD** PDO\_No, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the PDO object.

**RxTx:** [input] PDO type (0: RxPDO, 1: TxPDO).

**PDO\_No:** [input] PDO mapping object No (1 ~ 512).

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.38 CPM\_DynamicPDO

- **Description:**

This function can modify the mapping data of PDO object in the PDO object list of the CANopen Master Library stack. Take a note that before calling this function user must check if the PDO had been installed in the Master.

- **Syntax:**

**WORD** CPM\_DynamicPDO(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** RxTx, **BYTE** Entry, **DWORD** EntryData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the PDO object.

**RxTx:** [input] PDO type (0: RxPDO, 1: TxPDO).

**Entry:** [input] PDO mapping object subindex value (1 ~ 8).

**EntryData:** [input] A Double Word (4-byte) information of mapped application object. Users need to look up the user manual of the CANopen slave device to find the information of the application object, and obey the following example format to fill this parameter.

**For Example, EntryData = 0x64110310:** Mapping to index 0x6411 and subindex 0x03 with data length 0x10 bit (2-byte).

If the function parameters are as following, **Cobid = 0x333**, **RxTx = 0**, **Entry = 2**, **EntryData = 0x64110310**. This example will map the 16-bit data of index 0x6411 and subindex 0x03 object to the 2<sup>nd</sup> entry of COB-ID 0x333 RxPDO.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status



---

while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.





### 3.5.39 CPM\_RemovePDO

- **Description:**

The function CPM\_RemovePDO can remove a TxPDO or RxPDO object had installed by the CPM\_InstallPDO or CPM\_AddNode. This function also can remove single object mapped in TxPDO or RxPDO.

- **Syntax:**

**WORD** CPM\_RemovePDO(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**WORD** Cobid, **BYTE** Entry,  
**BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the PDO object.

**Entry:** [input] PDO mapping object subindex value (0 ~ 8). If this value is set to 0, the specified PDO object will be removed. If others (1 ~ 8), the specified subindex content of the PDO will be removed.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.40 CPM\_ChangePD OID

- **Description:**

Use the function CPM\_ChangePD OID to change the PDO COB-ID from old “Old\_Cobid” to new “New\_Cobid” of a slave device.

- **Syntax:**

**WORD** CPM\_ChangePD OID (**BYTE** BoardNo, **BYTE** Port,  
**WORD** Old\_Cobid, **WORD** New\_Cobid,  
**BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Old\_Cobid:** [input] Old COB-ID used by the PDO object.

**New\_Cobid:** [input] New COB-ID used by the PDO object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Old\_Cobid” again. If the procedure is still not complete, it will return “CPM\_Wait”.



### 3.5.41 CPM\_GetPDOMapInfo

- **Description:**

The function CPM\_GetPDOMapInfo can get the mapping data information of the PDO object.

- **Syntax:**

**WORD** CPM\_GetPDOMapInfo (**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **BYTE** \*RxTx, **BYTE** \*Tx\_Type,  
**WORD** \*Event\_Timer, **BYTE** \*Entry\_Cnt,  
**DWORD** \*Map\_Data, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**\*RxTx:** [output] PDO type (0: RxPDO, 1: TxPDO).

**\*Tx\_Type:** [output] Transmission type.

**\*Event\_Timer:** [output] PDO event timer.

**\*Entry\_Cnt:** [output] Useful PDO entry number of the PDO object.

**\*Map\_Data:** [output] Double Word array parameter. Response the mapping data of the PDO object's every useful entry.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.42 CPM\_InstallPDO\_List

- **Description:**

This function is similar with the CPM\_InstallPDO function. It can install the old or new PDO object in the PDO object list of the Master. It is the same as CPM\_InstallPDO. But the CPM\_InstallPDO\_List doesn't send any message to check if the PDO object exists in the real slave. It just changes the list in the memory of the Master. It means that user can use this function to install PDO and change PDO mapping data arbitrarily without disturbing the CANopen network. After using this function, the Master will process the slave PDOs which have the same IDs configured by the function CPM\_InstallPDO\_List. It is very useful when the Master is running in listen mode. User can use the function CPM\_RemovePDO\_List to remove the PDO object which is installed by CPM\_InstallPDO\_List.

- **Syntax:**

**WORD** CPM\_InstallPDO\_List(**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** RxTx, **WORD** PDO\_No, **BYTE** EntryUse, **DWORD** \*EntryData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127). If the "Node" parameter is the CPM100 Node-ID, the parameters, EntryUse & EntryData, will be useless. In this case, users can use the function CPM100\_InstallRxPDOISR or CPM100\_InstallRemotePDOISR to install the users' callback function to process the received PDOs of the CPM100.

**Cobid:** [input] COB-ID used by the PDO object.

**RxTx:** [input] PDO type (0: RxPDO, 1: TxPDO).

**PDO\_No:** [input] PDO mapping object No (1 ~ 512).

**EntryUse:** [input] Total entry of mapping object that will be installed.



---

**\*EntryData:** [input] Double Word array information of mapped application object. For example:

If the configuration is “**Cobid = 0x333, RxTx = 0, PDO\_No = 10, Entry = 2, EntryData[0] = 0x64110310, EntryData[1] = 0x62000108**”, it will map the RxPDO10 with COB-ID 0x333. The 1<sup>st</sup> entry is 16-bit data of index 0x6411 and subindex 0x03 object and the 2<sup>nd</sup> entry is 8-bit data of index 0x6200 and subindex 0x01 object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Node” again. If the procedure is still not complete, it will return “CPM\_Wait”.

### 3.5.43 CPM\_RemovePDO\_List

- **Description:**

The function CPM\_RemovePDO\_List can remove a TxPDO or RxPDO object had installed by the CPM\_InstallPDO\_List. This function also can remove single object mapped in the TxPDO or RxPDO.

- **Syntax:**

**WORD** CPM\_RemovePDO\_List(**BYTE** BoardNo, **BYTE** Port,  
**BYTE** Node, **WORD** Cobid, **BYTE** Entry,  
**BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the PDO object.

**Entry:** [input] PDO mapping object subindex value (0 ~ 8). If this parameter is set to 0, the specified PDO object will be removed. If others (1 ~ 8), the specified subindex content of the PDO will be removed.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.44 CPM\_PDUseEntry

- **Description:**

Use this function to change the useful object mapping entry of PDO object. The useful entry starts from 1 to the parameter Entry. Therefore, if the parameter Entry is 0, it means that the PDO have no useful object mapping entry.

- **Syntax:**

**WORD** CPM\_PDUseEntry(**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **BYTE** Entry, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Entry:** [input] Useful entry number of PDO mapping object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".

### 3.5.45 CPM\_PDOTxType

- **Description:**

Use this function to change transmission type of TxPDO. The default transmission type is 255.

- **Syntax:**

**WORD** CPM\_PDOTxType(**BYTE** BoardNo, **BYTE** Port, **WORD** Cobid, **BYTE** Tx\_Type, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Tx\_Type:** [input] Transmission type of TxPDO (0 ~ 255).

**Description of transmission type**

transmission type	PDO transmission				
	cyclic	acyclic	synchronous	asynchronous	RTR only
0		<b>X</b>	<b>X</b>		
1-240	<b>X</b>		<b>X</b>		
241-251	<b>- reserved -</b>				
252			<b>X</b>		<b>X</b>
253				<b>X</b>	<b>X</b>
254				<b>X</b>	
255				<b>X</b>	

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".





### 3.5.46 CPM\_PDSEventTimer

- **Description:**

Use this function to change the event timer of the TxPDO. The default event timer is 0. When the event timer of the PDO object of the slave is more than 0, the PDO will be sent to master due to the parameter “Timer” automatically.

- **Syntax:**

**WORD** CPM\_PDSEventTimer(**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **WORD** Timer, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Timer:** [input] Event timer of TxPDO. The unit is millisecond.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users’ application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Cobid” again. If the procedure is still not complete, it will return “CPM\_Wait”.



---

### 3.5.47 CPM\_PDOInhibitTime

- **Description:**

Use this function to set the inhibit time to the TxPDO. This time is a minimum interval for PDO transmission.

- **Syntax:**

**WORD** CPM\_PDOInhibitTime(**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **WORD** Time, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the PDO object.

**Time:** [input] Inhibit time of TxPDO.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.48 CPM\_ChangeSYNCID

- **Description:**

Use the function CPM\_ChangeSYNCID to change the SYNC COB-ID of a slave device.

- **Syntax:**

**WORD** CPM\_ChangeSYNCID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the SYNC object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Cobid" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.49 CPM\_SetSYNC\_List

- **Description:**

If the user uses CPM\_AddNode function to add the slave with manual mode, the function CPM\_SetSYNC\_List must be called while the SYNC ID of the slave needs to be changed or be set. The function CPM\_SetSYNC\_List can only change the SYNC COB-ID in the COB-ID list of the CPM100, the real value stored in the slave device may be different from the configuration which is set by the function CPM\_SetSYNC\_List. The users need to confirm that by themselves.

- **Syntax:**

**WORD** CPM\_SetSYNC\_List (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the SYNC object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.50 CPM\_GetSYNCID

- **Description:**

This function can get the SYNC ID from the COB-ID list of the Master.

- **Syntax:**

**WORD** CPM\_GetSYNCID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**WORD** \*Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*Cobid:** [output] Return the COB-ID used by the SYNC object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.51 CPM\_SendSYNCMsg

- **Description:**

Use the function CPM\_SendSYNCMsg to send a SYNC message with specified COB-ID cyclically. If the parameter “Timer” is 0, the SYNC message will be stopped. If the parameter “Timer” is more than 0, the function will send SYNC message per “Timer” millisecond until finish the parameter “Times”. When the “Times” is set to 0, the function will send SYNC message continuously until set “Timer” to 0. Users can set at most 5 SYNC messages with different ID to be sent cyclically.

- **Syntax:**

**WORD** CPM\_SendSYNCMsg(**BYTE** BoardNo, **BYTE** Port,  
**WORD** Cobid, **WORD** Timer, **DWORD** Times,  
**BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Cobid:** [input] COB-ID used by the SYNC object.

**Timer:** [input] SYNC message transmission period. If the timer is 0, the SYNC message will be stopped.

**Times:** [input] SYNC message transmission times. If the time is 0, the SYNC message will be sending until “Timer” is set to 0.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users’ application will be held in the function until return. If set to 0, this function will return “CPM\_Processing” directly. This function will return its process status while users apply it with the same “BoardNo”, “Port” and “Cobid” again. If the procedure is still not complete, it will return “CPM\_Wait”.



---

### 3.5.52 CPM\_GetCyclicSYNCInfo

- **Description:**

This function can get at most 5 SYNC messages information which have been configured by the function CPM\_SendSYNCMsg. User can know what SYNC ID had been set.

- **Syntax:**

**WORD** CPM\_GetCyclicSYNCInfo(**BYTE** BoardNo, **BYTE** Port,  
**WORD** \*Cobid, **WORD** \*Timer,  
**DWORD** \*Times, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**\*Cobid:** [input] COB-ID array. Return most 5 SYNC ID.

**\*Timer:** [input] 5 WORD array. Each value is the cyclic period of the SYNC message.

**\*Times:** [input] 5 Double WORD array. Each one is the SYNC message sending times that set before.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.53 CPM\_ChangeEMCYID

- **Description:**

Use the function CPM\_ChangeEMCYID to change the EMCY COB-ID of a specific slave device.

- **Syntax:**

**WORD** CPM\_ChangeEMCYID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the EMCY object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.54 CPM\_SetEMCY\_List

- **Description:**

If the user uses CPM\_AddNode function to add the slave with manual mode, the function CPM\_SetEMCY\_List must be called while the EMCY ID of the slave needs to be changed or be set. CPM\_SetEMCY\_List only can change the EMCY COB-ID in the COB-ID list of the Master. Afterwards, the Master processes the EMCY messages with the specific EMCY COB-ID which is configured by the function CPM\_SetEMCY\_List.

- **Syntax:**

**WORD** CPM\_SetEMCY\_List (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node, **WORD** Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**Cobid:** [input] COB-ID used by the EMCY object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



---

### 3.5.55 CPM\_GetEMCYID

- **Description:**

This function can get the EMCY ID from the COB-ID list of the Master.

- **Syntax:**

**WORD** CPM\_GetEMCYID (**BYTE** BoardNo, **BYTE** Port, **BYTE** Node,  
**WORD** \*Cobid, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*Cobid:** [output] Return the COB-ID used by the EMCY object.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.56 CPM\_ReadLastEMCY

- **Description:**

This function can check if one slave had produced EMCY. If yes, this function will return the last EMCY message of the specific slave.

- **Syntax:**

**WORD** CPM\_ReadLastEMCY (**BYTE** BoardNo, **BYTE** Port,  
**BYTE** Node, **BYTE** \*IsNew,  
**BYTE** \*RData, **BYTE** BlockMode)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**Node:** [input] Slave device Node-ID (1~127).

**\*IsNew:** [output] Check the data if had been read before. 0 is been read before, and 1 is new one.

**\*RData:** [output] 8-byte EMCY message gets from the EMCY buffer.

**BlockMode:** [input] 0 means this function is non-block-function, and 1 means this function is block-function. If set this parameter to 1, the running procedure of the users' application will be held in the function until return. If set to 0, this function will return "CPM\_Processing" directly. This function will return its process status while users apply it with the same "BoardNo", "Port" and "Node" again. If the procedure is still not complete, it will return "CPM\_Wait".



### 3.5.57 CPM\_GetBootUpNodeAfterAdd

- **Description:**

If users don't know which slave node occur the boot-up message and which Master received it. Users can use the function CPM\_GetBootUpNodeAfterAdd. This function can get not only the slave node ID but also the slot number of the PISO-CAN. The parameters, BoardNo and Port, indicate the number of the PISO-CAN which receives the boot-up message. The parameter, Node, is the ID of the slave node which produces the boot-up message. The CPM\_GetBootUpNodeAfterAdd function is usually applied with the function CPM\_InstallBootUpISR. But note that, this function can only get the slave node ID that has already been added (CPM\_AddNode) to the Master.

- **Syntax:**

**WORD** CPM\_GetBootUpNodeAfterAdd (**BYTE** \*BoardNo, **BYTE** \*Port,  
**BYTE** \*Node)

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

**Node:** [input] Get the slave node ID of the received boot-up message.



---

### 3.5.58 CPM\_GetEMCYData

- **Description:**

If users don't know that which slave node occurred the EMCY message and which Master received it. Users can use the function CPM\_GetEMCYData. This function can get not only the EMCY message with the slave node ID but also the board number of the PISO-CAN. The parameters, BoardNo and Port, indicate the number of the PISO-CAN which receives the EMCY message. The parameter, Node, is the ID of the slave node which produces the EMCY. The parameter, \*RData, is the 8-bytes EMCY message data. The function CPM\_GetEMCYData is usually applied with the function CPM\_InstallEMCYISR. About the demo please refer to the section 4.1.2 NMT\_Protocol.

- **Syntax:**

**WORD** CPM\_GetEMCYData (**BYTE** \*BoardNo, **BYTE** \*Port,  
**BYTE** \*Node, **BYTE** \*RData)

- **Parameter:**

- \***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.
- \***Port:** [output] Get the CAN port No. of the PISO-CAN board.
- \***Node:** [output] Get the slave node ID of the received EMCY message.
- \***RData:** [output] 8-byte EMCY message obtained from the EMCY buffer.



### 3.5.59 CPM\_GetNMTErr

- **Description:**

User can use the function CPM\_GetNMTErr to check if the Master gets NMT Error Event for any slave node. The parameters of the function indicate that which Master gets the NMT Error Event, which node produces this event, and what kind of event it is. The parameters, BoardNo and Port, indicate which Master indicates the Heartbeat\_Event or Node\_Guarding\_Event. The parameter, Node, is the ID of the slave node which responds the heartbeat protocol or guarding protocol. The parameter, NMTErr, is the NMTErr event mode. If the NMTErr event is Node\_Guarding\_Event, the NMTErr parameter is CPM\_Node\_Guarding\_Event or else the CPM\_Heartbeat\_Event is obtained. The function CPM\_GetNMTErr is usually applied with the function CPM\_InstallNMTErrISR. About the demo please refer to the section 4.1.2 NMT\_Protocol.

- **Syntax:**

**WORD** CPM\_GetNMTErr (**BYTE** \*BoardNo, **BYTE** \*Port,  
**BYTE** \*Node, **BYTE** \*NMTErr)

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

\***Node:** [output] Get the slave node ID of the NMT Error Event.

\***NMTErr:** [output] The value CPM\_Node\_Guarding\_Event indicates the Node Guarding Event, and the CPM\_Heartbeat\_Event is the Heartbeat Event.



---

### 3.5.60 CPM\_InstallBootUpISR

- **Description:**

This function allows the user to apply the slave boot-up IST (interrupt service thread). When the user puts his boot-up process into this function, all the boot-up triggered by the slaves will go to the boot-up IST. If the boot-up message of a slave which has been added to the Master is happen, the Master will go into the boot-up process to do some specified mechanism which follows the user's boot-up process.

- **Syntax:**

**WORD** CPM\_InstallBooUpISR (**BYTE** BoardNo, **BYTE** Port,  
**void** (\*BOOTISR)( ))

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*BOOTISR)( ):** [input] The pointer which points a function with format "void XXX( )". The XXX is the function name of the user's boot-up process. This process is usually applied with the function CPM\_GetBootUpNodeAfterAdd.



---

### 3.5.61 CPM\_RemoveBootUpISR

- **Description:**

When the user doesn't need the boot-up IST function, call this function to remove the user's IST.

- **Syntax:**

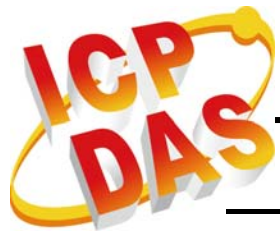
**WORD** CPM\_RemoveBootUpISR (**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.





---

### 3.5.62 CPM\_InstallEMCYISR

- **Description:**

This function allows the user to apply the EMCY IST (interrupt service thread). When the user puts his EMCY process into this function, all the EMCY triggered by the slaves will go to the EMCY IST. If the EMCY of a slave is happen, the Master will go into the EMCY process to do some security mechanism which follows the user's EMCY process.

- **Syntax:**

**WORD** CPM\_InstallEMCYISR(**BYTE** BoardNo, **BYTE** Port,  
**void** (\*EMCYISR)( ))

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*EMCYISR)( ): [input]** The pointer which points a function with format "void XXX( )". The XXX is the function name of the user's EMCY process. This process is usually applied with the function CPM\_GetEMCYData.



---

### 3.5.63 CPM\_RemoveEMCYISR

- **Description:**

When the user doesn't need the EMCY IST function, call this function to remove the user's IST.

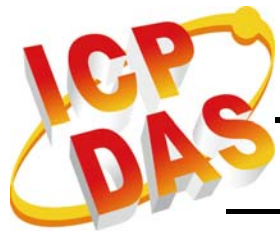
- **Syntax:**

**WORD** CPM\_RemoveEMCYISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

### 3.5.64 CPM\_InstallNMTErrISR

- **Description:**

This function allows the user to apply NMTErr IST (interrupt service thread). When the user puts his NMTErr process into this function, all the Heartbeat\_Event and Node\_Guarding\_Event triggered by the slaves will go to the IST. If the user had used the CPM\_NMTGuarding to enable the guarding protocol or had used the CPM\_Heartbeat to enable the heartbeat protocol, the Master will go into the NMTErr IST to do the user's NMTErr process while the guarding confirms or heartbeat indicator doesn't be received.

- **Syntax:**

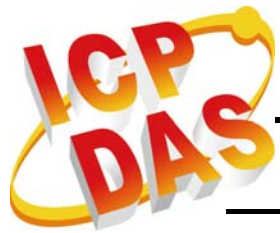
**WORD** CPM\_InstallNMTErrISR(**BYTE** BoardNo, **BYTE** Port,  
**void** (\*NMTErrISR)( ))

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*NMTErrISR)( ):** [input] The pointer which points a function with format "void XXX( )". The XXX is the function name of the user's process. This process is usually applied with the function CPM\_GetNMTError.



---

### 3.5.65 CPM\_RemoveNMTErrISR

- **Description:**

When the user doesn't need the NMTErr IST function, call this function to remove the user's IST.

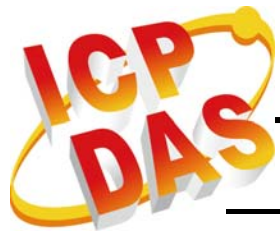
- **Syntax:**

**WORD** CPM\_RemoveNMTErrISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

### 3.5.66 CPM\_GetMasterReadSDOEvent

- **Description:**

Using this function can get all the read SDO messages sent to the specific node ID of the Master. For example, the Master is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the Master for reading an object, users can use the function CPM\_GetMasterReadSDOEvent for obtaining this SDO message, and respond some information to the SDO sender. The parameters, BoardNo and Port, indicate which Master receives the read SDO message. The parameters, Index and SubIndex, are the object indicator. The function CPM\_GetMasterReadSDOEvent is usually applied with the function CPM\_InstallReadSDOISR. About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

**WORD** CPM\_GetMasterReadSDOEvent(**BYTE** \*BoardNo, **BYTE** \*Port, **WORD** \*Index, **BYTE** \*SubIndex)

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

\***Index:** [output] Get the object index of the SDO message.

\***SubIndex:** [output] Get the object subindex of the SDO message.



---

### 3.5.67 CPM\_GetMasterWriteSDOEvent

- **Description:**

Using this function can get all the write SDO messages sent to the specific node ID of the Master. For example, the Master is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the Master for writing an object, users can use the function CPM\_GetMasterWriteSDOEvent for obtaining this SDO message. The parameters, BoardNo and Port, indicate which Master receives the write SDO message. The parameters, Index and SubIndex, are the object indicator. The parameter WLen is the data length of the parameter \*WData. The function CPM\_GetMasterWriteSDOEvent is usually applied with the function CPM\_InstallWriteSDOISR. About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

```
WORD CPM_GetMasterWriteSDOEvent(BYTE *BoardNo, BYTE *Port,  
                                WORD *Index, BYTE *SubIndex,  
                                BYTE *WLen, BYTE *WData)
```

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

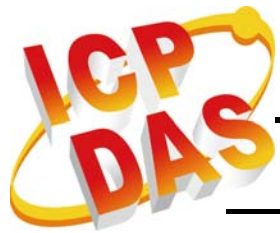
\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

\***Index:** [output] Get the object index of the SDO message.

\***SubIndex:** [output] Get the object subindex of the SDO message.

\***WLen:** [output] The data length of the write data.

\***WData:** [output] Return 0~4 bytes of the SDO write data.



---

### 3.5.68 CPM\_ResponseMasterSDO

- **Description:**

Using this function can reply the SDO messages to the SDO sender. For example, the Master is initialized with node ID 2. If someone sends a SDO message with the COB-ID 0x602 for reading or writing the object of the Master, the Master need to reply the corresponding SDO message, use the function CPM\_ResponseMasterSDO to do it. When users implement the function CPM\_ResponseMasterSDO, the Master will send a SDO message with COB-ID 0x582 to the CANopen network. This function is usually applied with the SDO ISR series function .About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note1: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

**Note2: If the Master want to reply a SDO Abort message, please use the function CPM\_SDOAbortTransmit (section 3.5.20) to do it.**

- **Syntax:**

**WORD** CPM\_ResponseMasterSDO (**BYTE** BoardNo, **BYTE** Port, **BYTE** ResType, **WORD** Index, **BYTE** SubIndex, **BYTE** Len, **BYTE** \*Data)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

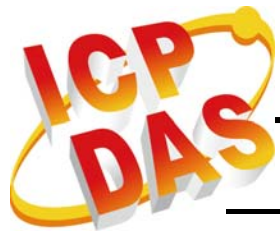
**ResType:** [input] Response type of SDO message, 0 for replying the read SDO message and 1 for the write SDO message.

**Index:** [input] Object index of object dictionary of slave devices.

**SubIndex:** [input] Object subindex of object dictionary of slave devices.

**Len:** [input] The data length of the response data. If the ResType is 1 (write type), the Len and \*Data parameter is useless.

**\*Data:** [input] Return 0~4 bytes of the SDO response data.



---

### 3.5.69 CPM\_InstallReadSDOISR

- **Description:**

This function allows the user to apply the ReadSDO IST (interrupt service thread) of the Master. When the user puts his read SDO process into this function, all the read SDO messages sent to the specified Master will trigger the IST. For example, the Master is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 for reading the object of the Master, the Master will go into the IST if the user has installed it.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

```
WORD CPM_InstallReadSDOISR(BYTE BoardNo, BYTE Port,  
                           void (*RSDOISR)( ))
```

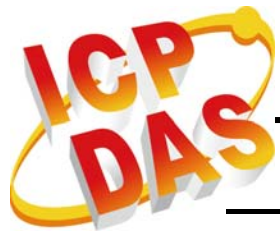
- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*RSDOISR)( ): [input]** The pointer which points a function with format "void XXX( )". The XXX is the function name of the user's process. This process is usually used with the function CPM\_GetMasterReadSDOEvent.





---

### 3.5.70 CPM\_RemoveReadSDOISR

- **Description:**

When the user doesn't need the ReadSDO IST function, call this function to remove the user IST.

- **Syntax:**

**WORD** CPM\_RemoveReadSDOISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

### 3.5.71 CPM\_InstallWriteSDOISR

- **Description:**

This function allows the user to apply the WriteSDO IST (interrupt service routine) of the Master. When the user puts the process into this function, all the written SDO messages sent to the specified Master will trigger the IST. For example, the Master is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the Master for writing an object, the Master will go into the IST if the user has installed it.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

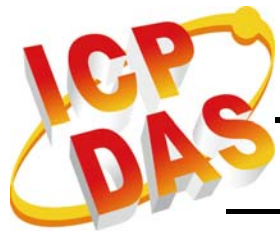
```
WORD CPM_InstallWriteSDOISR(BYTE BoardNo, BYTE Port,  
void (*WSDOISR)( ))
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*WSDOISR)( ): [input]** The pointer which points a function with format "void XXX( )". The XXX is the function name of user's process. This process is usually used with the function CPM\_GetMasterWriteSDOEvent.



---

### 3.5.72 CPM\_RemoveWriteSDOISR

- **Description:**

When the user doesn't need the WriteSDO IST function, call this function to remove the user IST.

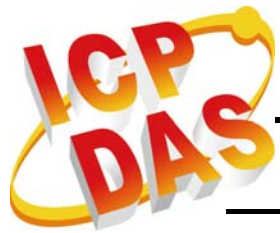
- **Syntax:**

**WORD** CPM\_RemoveWriteSDOISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

### 3.5.73 CPM\_GetMasterRemotePDOEvent

- **Description:**

Using this function can get all the Remote PDO messages sent to the Master. For example, the Master has used the function CPM\_InstallPDO\_List to install a TxPDO object with the COB-ID 0x444. If someone sends a Remote PDO message with the COB-ID 0x444 to the Master, users can use CPM\_GetMasterRemotePDOEvent to get this PDO message. The parameters, BoardNo and Port, indicate which Master receives the Remote PDO message. The parameter, Cobid, is the PDO COB-ID sent to the Master. This function is usually used with the function CPM\_InstallRemotePDOISR. About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

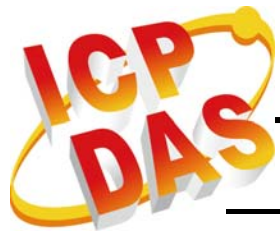
**WORD** CPM\_GetMasterRemotePDOEvent (**BYTE** \*BoardNo,  
**BYTE** \*Port, **WORD** \*CobId)

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

\***CobId:** [output] Return the COB-ID of the Remote PDO message.



---

### 3.5.74 CPM\_GetMasterRxPDOEvent

- **Description:**

Using this function can get all the RxPDO messages sent to the Master. For example, the Master has used the function CPM\_InstallPDO\_List to install an RxPDO object with the COB-ID 0x333. If someone sends an RxPDO message with the COB-ID 0x333 to the Master, users can use this function to get this RxPDO message. The parameters, BoardNo and Port, indicate which Master receives the RxPDO message. The parameter, Cobid, is the RxPDO COB-ID ID. The two parameters, \*WLen and \*WData, are the data length and contents of the RxPDO message. This function is usually applied with the function CPM\_InstallRxPDOISR. About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

**WORD** CPM\_GetMasterRxPDOEvent (**BYTE** \*BoardNo, **BYTE** \*Port,  
**WORD** \*CobId, **BYTE** \*WLen, **BYTE** \*WData)

- **Parameter:**

\***BoardNo:** [output] Get the board No. of the PISO-CAN which receives the NMT Error Event.

\***Port:** [output] Get the CAN port No. of the PISO-CAN board.

\***CobId:** [output] Return the RxPDO COB-ID.

\***WLen:** [output] The data length of the RxPDO data.

\***WData:** [output] Return 0~4 bytes of the RxPDO data.



---

### 3.5.75 CPM\_ResponseMasterPDO

- **Description:**

Using this function can reply the Remote PDO messages to the sender. For example, the Master has used CPM\_InstallPDO\_List to install a TxPDO object with the COB-ID 0x444. If someone sends a Remote PDO message with the COB-ID 0x444 to the Master, and the Master needs to reply a TxPDO message, users can use this function to do it. When users implement CPM\_ResponseMasterPDO, the Master will send a TxPDO message to the CANopen network. This function is usually used with the CPM\_InstallRemotePDOISR function. About the demo please refer to the section 4.1.6 SDO\_PDO\_ISR.

**Note: The function is valid while the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

**WORD** CPM\_ResponseMasterPDO (**BYTE** BoardNo, **BYTE** Port,  
**WORD** CobId, **BYTE** Len, **BYTE** \*Data)

- **Parameter:**

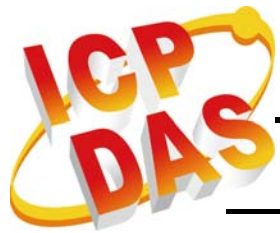
**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**CobId:** [input] TxPDO COB-ID for replying the PDO message.

**Len:** [input] The data length of the response data.

**\*Data:** [input] Return the COB-ID of the TxPDO PDO message.



---

### 3.5.76 CPM\_InstallRxPDOISR

- **Description:**

This function allows the user to apply the RxPDO IST (interrupt service routine) of the Master. When the user puts his process into this function, all the RxPDO messages with the Master's PDO objects will trigger the IST. For example, the Master has used CPM\_InstallPDO\_List to install a PDO object with the COB-ID 0x333 of the Master. If someone sends a PDO message with the COB-ID 0x333 to the Master, the Master will go into the IST if the user had installed it.

**Note: The function will usefully when the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

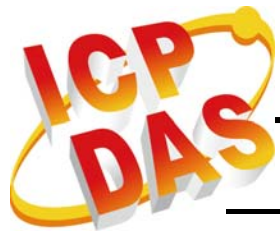
```
WORD CPM_InstallRxPDOISR(BYTE BoardNo, BYTE Port,  
void (*RXPDOISR)( ))
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*RXPDOISR)( ): [input]** The pointer which points a function with format "void XXX( )". The XXX is the function name of user's process. This process is usually used with the function CPM\_GetMasterRxPDOEvent.



---

### 3.5.77 CPM\_RemoveRxPDOISR

- **Description:**

When the user doesn't need the RxPDO IST function, call this function to remove the user IST.

- **Syntax:**

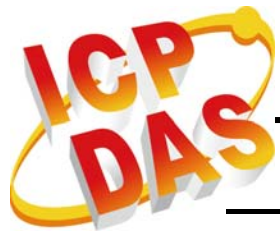
**WORD** CPM\_RemoveRxPDOISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.





---

### 3.5.78 CPM\_InstallRemotePDOISR

- **Description:**

This function allows the user to apply the RemotePDO IST (interrupt service routine) of the Master. When the user puts his process into this function, all the Remote PDO messages of the Master's PDO objects will trigger the IST. For example, the Master has used CPM\_InstallPDO\_List to install a TxPDO object with the COB-ID 0x444 of the Master. If some one sends a Remote PDO message with the COB-ID 0x444 to the Master, the Master will go into the IST if the user had installed it.

**Note: The function will usefully when the Node parameter of the function CPM\_InitMaster is > 0.**

- **Syntax:**

```
WORD CPM_InstallRemotePDOISR(BYTE BoardNo, BYTE Port,  
                             void (*REMOTEPDOISR)( ))
```

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.

**(\*REMOTEPDOISR)( ): [input]** The pointer which points a function with format "void XXX( )". The XXX is the function name of user's process. This process is usually used with the function CPM\_GetMasterRemotePDOEvent.



---

### 3.5.79 CPM\_RemoveRemotePDOISR

- **Description:**

When the user doesn't need the RemotePDO IST function, call this function to remove the user IST.

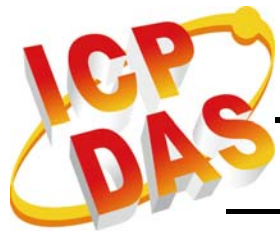
- **Syntax:**

**WORD** CPM\_RemoveRemotePDOISR(**BYTE** BoardNo, **BYTE** Port)

- **Parameter:**

**BoardNo:** [input] PISO-CAN board No.

**Port:** [input] CAN port No. of the PISO-CAN board.



---

## 4 Demo Programs

The PISO-CAN CANopen Master Library provides 10 demos of the various applications for NMT protocol, SDO protocol, PDO protocol, NMT Error IST...etc. All these demos support VC++, VB.net 2005, and C# 2005. Users can find these demos in the fieldbus CD or on the web site.

The path of CAN CD

**fieldbus cd://canopen/master/pcm\_piso-can\_series**

The address of the web site is

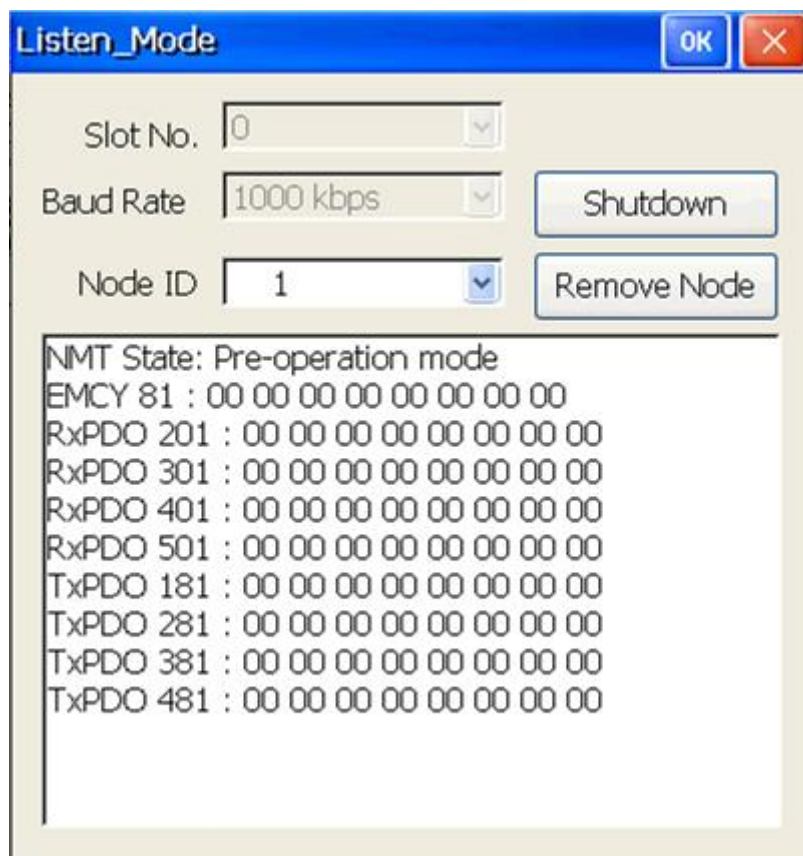
**[http://ftp.icpdas.com.tw/pub/cd/canopen/master/pcm\\_piso-can\\_series/](http://ftp.icpdas.com.tw/pub/cd/canopen/master/pcm_piso-can_series/)**

### 4.1 Brief of the demo programs

These demo programs are developed for demonstrating how to use the CANopen master library to apply the general CANopen communication protocol. These demo programs provide the SDO, PDO, NMT, SYNC communication applications. Each demo program includes some functions of the CANopen master library. The relationship between CANopen master library functions and demo programs are displayed in the following description.

### 4.1.1 Listen\_Mode

Initialize the CANopen Master with the “listen mode” and add slave nodes with the “manual mode” or EDS file, then the Master listens CANopen messages only and does not send any message to the CANopen network. In this demo, the Master will listen NMT state, 4 TxPDO messages (with the COB ID 0x180+Node ID, 0x280+Node ID, 0x380+Node ID, and 0x480+Node ID), 4 RxPDO messages (with the COB ID 0x200+Node ID, 0x300+Node ID, 0x400+Node ID, and 0x500+Node ID), and the EMCY messages.




Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_EDS\\_Load](#), [CPM\\_SetMasterMode](#), [CPM\\_NMTGetState](#),  
[CPM\\_InstallPDO\\_List](#), [CPM\\_GetPDOLastData](#), [CPM\\_GetRxPDOID](#),  
[CPM\\_GetTxPDOID](#), [CPM\\_SetEMCY\\_List](#), [CPM\\_GetEMCYID](#),  
[CPM\\_ReadLastEMCY](#).

## 4.1.2 NMT\_Protocol

This is a NMT network control demo. The demo not only tells users how to control the NMT status of a specific slave node, but also how to protect the slave through the “Guarding” and “Heartbeat” functions.



The screenshot shows a software window titled "NMT\_Protocol" with a blue title bar and "OK" and "X" buttons. The window contains several controls:

- Slot No.: A dropdown menu showing "0".
- Baud Rate: A dropdown menu showing "1000 kbps".
- Node: A dropdown menu showing "1".
- Node State: A dropdown menu showing "Operation".
- Guarding Time: A text input field containing "1000".
- Life: A text input field containing "2".
- Heartbeat (ms): A text input field containing "1000".
- Consumer (ms): A text input field containing "2500".
- EMCY: A text area containing "00 00 00 00 00 00 00 00".

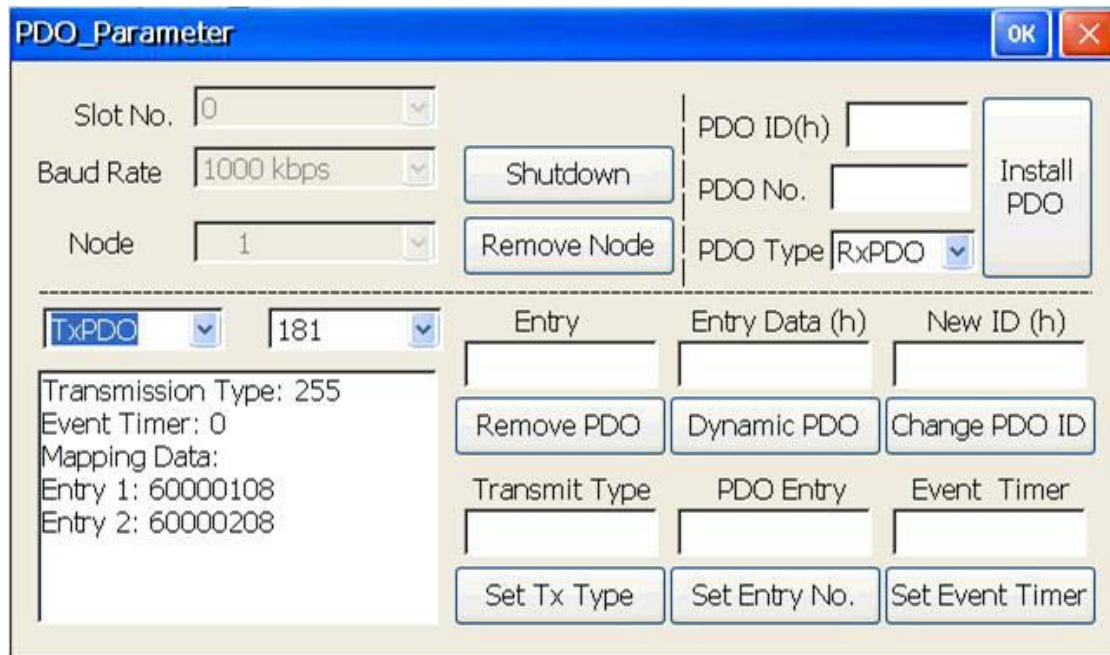
Buttons on the right side of the window include: "Shutdown", "Remove Node", "Set Status", "Set Guarding", "Set Heartbeat", and "Clear".

Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_NMTChangeState](#), [CPM\\_NMTGuarding](#), [CPM\\_NMTHeartbeat](#),  
[CPM\\_GetEMCYData](#), [CPM\\_GetNMTErr](#), [CPM\\_InstallNMTErrISR](#),  
[CPM\\_RemoveNMTErrISR](#), [CPM\\_InstallEMCYISR](#), [CPM\\_RemoveEMCYISR](#).

### 4.1.3 PDO\_Parameter

Sometimes, the default PDO configuration can't satisfy users. Users need to change the configuration of the PDO related parameters such as transmission type, PDO ID, event timer, dynamic PDO, and so forth. This demo will demonstrate how to change settings of these PDO parameters and show the configuration result.

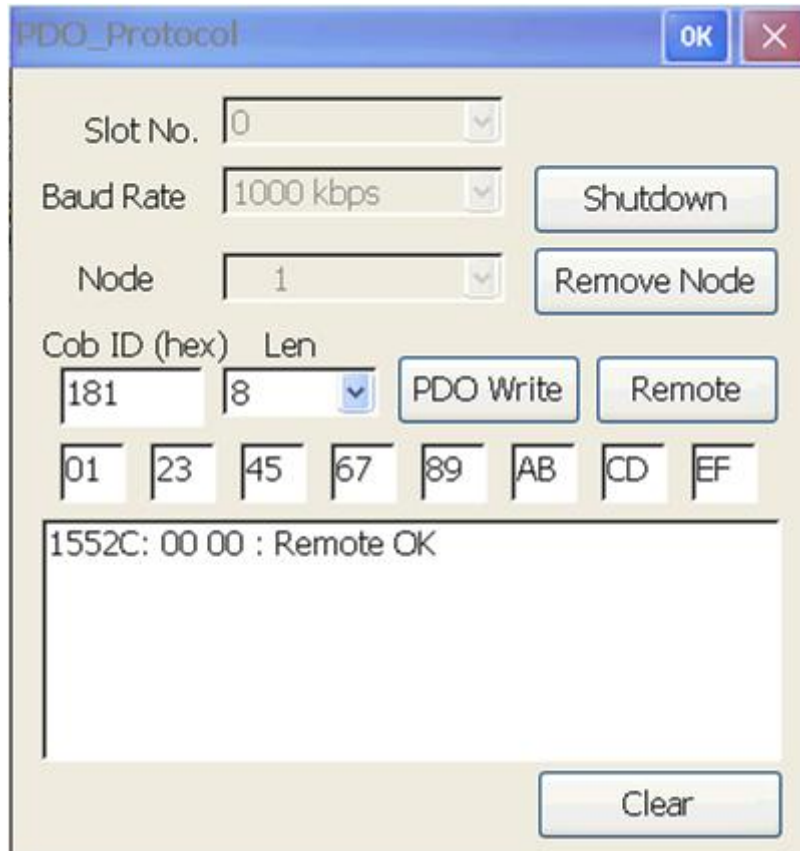


Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_InstallPDO](#), [CPM\\_RemovePDO](#), [CPM\\_DynamicPDO](#),  
[CPM\\_ChangePDOID](#), [CPM\\_PDOTxType](#), [CPM\\_PDOWUseEntry](#),  
[CPM\\_PDOWEventTimer](#), [CPM\\_GetTxPDOID](#), [CPM\\_GetRxPDOID](#),  
[CPM\\_GetPDOWMapInfo](#).

#### 4.1.4 PDO\_Protocol

The PDO protocol is the main protocol to control the I/O of the specific slave device in the CANopen network. This demo shows how to read and write data to the slave device with the PDO functions.

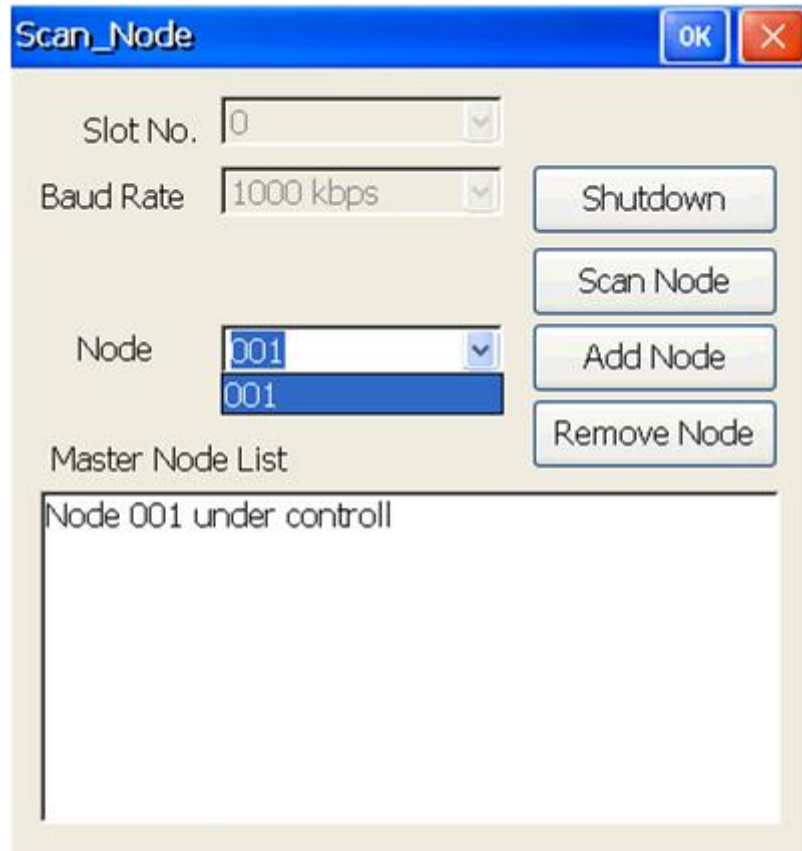


Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_PDOWrite](#), [CPM\\_PDORemote](#), [CPM\\_GetPDOLastData](#)

## 4.1.5 Scan\_Node

When users want to know which slave nodes exist on the CANopen network or which slave nodes are under the control of the CPM100, this demo will is useful.



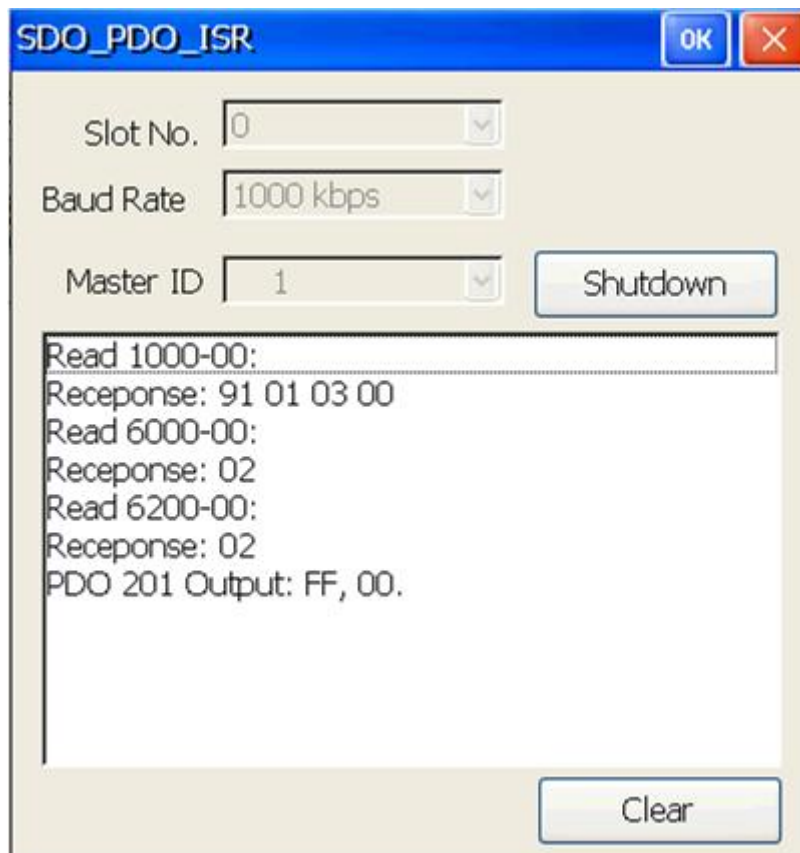
Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_SetFunctionTimeout](#), [CPM\\_ScanNode](#), [CPM\\_GetNodeList](#)



## 4.1.6 SDO\_PDO\_ISR

In this demo, it is allowed to configure the PISO-CAN as a CANopen slave. Users can use another CANopen master to read/write the users' defined object dictionary of the PISO-CAN by SDO protocol or to get/set the DIO status by PDO protocol when the PISO-CAN, the I-8053W DI module, and the I-8057W DO module are plugged in the same MCU. If the user has an application like this, this demo may be a good reference.

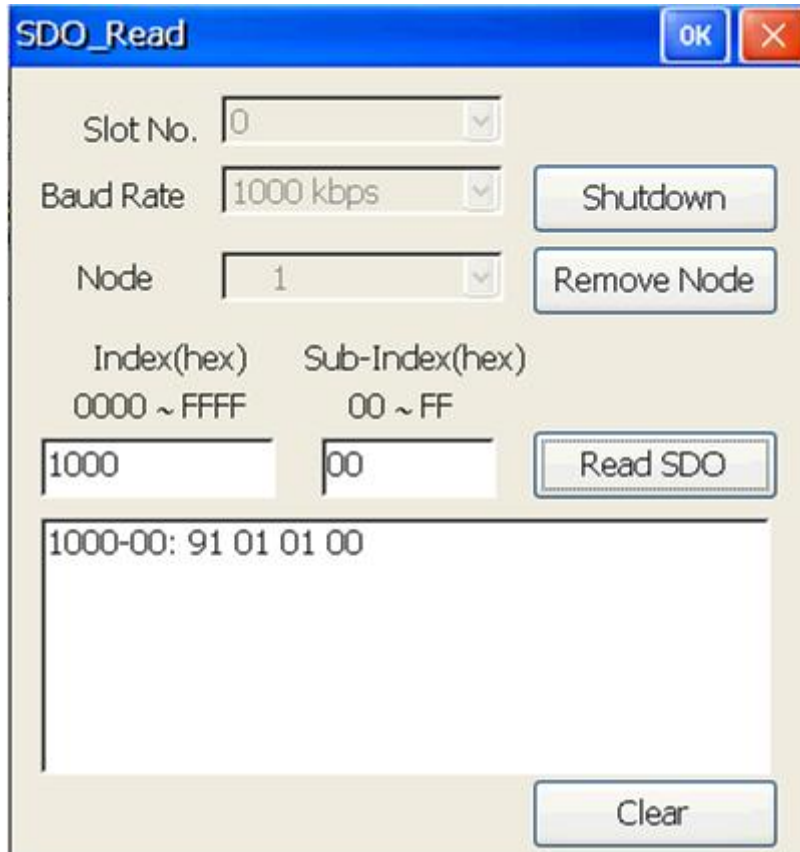


Applied function list:

[CPM\\_InitMaster](#),      [CPM\\_Shutdown](#),      [CPM\\_GetMasterReadSDOEvent](#),  
[CPM\\_GetMasterWriteSDOEvent](#),      [CPM\\_GetMasterRemotePDOEvent](#),  
[CPM\\_GetMasterRxPDOEvent](#),      [CPM\\_ResponseMasterSDO](#),  
[CPM\\_ResponseMasterPDO](#),      [CPM\\_InstallPDO\\_List](#),  
[CPM\\_InstallReadSDOISR](#),      [CPM\\_InstallWriteSDOISR](#),  
[CPM\\_InstallRxPDOISR](#), [CPM\\_InstallRemotePDOISR](#).

## 4.1.7 SDO\_Read

SDO protocol is a kind of the communication functions used to read/write CANopen object dictionary. You can read any object data of the object dictionary through the object address (index and sub-index) by SDO protocol. This demo is a good model to do that.



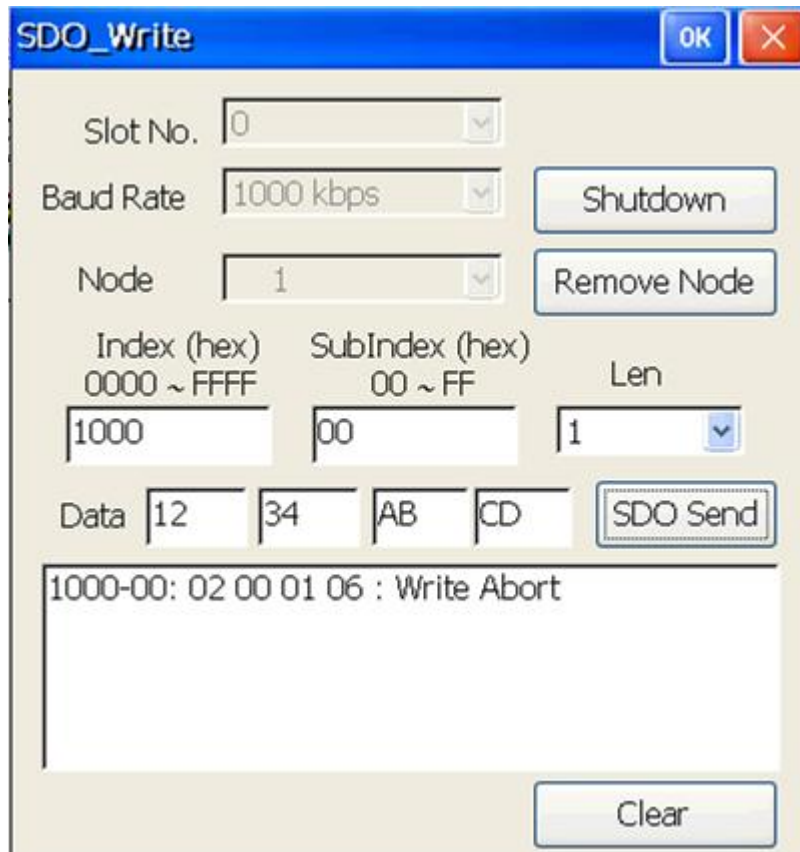
The screenshot shows a software dialog box titled "SDO\_Read". It contains several input fields and buttons. The "Slot No." field is set to 0. The "Baud Rate" field is set to 1000 kbps. The "Node" field is set to 1. The "Index(hex)" field is set to 1000, and the "Sub-Index(hex)" field is set to 00. Below these fields, there is a text area displaying the address "1000-00: 91 01 01 00". Buttons for "Shutdown", "Remove Node", "Read SDO", and "Clear" are visible. The dialog box also has "OK" and "X" buttons in the top right corner.

Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_SDOReadData](#).

## 4.1.8 SDO\_Write

SDO protocol is a kind of the communication functions used to read/write CANopen object dictionary. You can write any data to the specific object of the object dictionary through the object address (index and sub-index) by SDO protocol. This demo is a good model to do that.



SDO\_Write

Slot No. 0

Baud Rate 1000 kbps

Node 1

Index (hex) 0000 ~ FFFF: 1000

SubIndex (hex) 00 ~ FF: 00

Len: 1

Data: 12 34 AB CD

1000-00: 02 00 01 06 : Write Abort

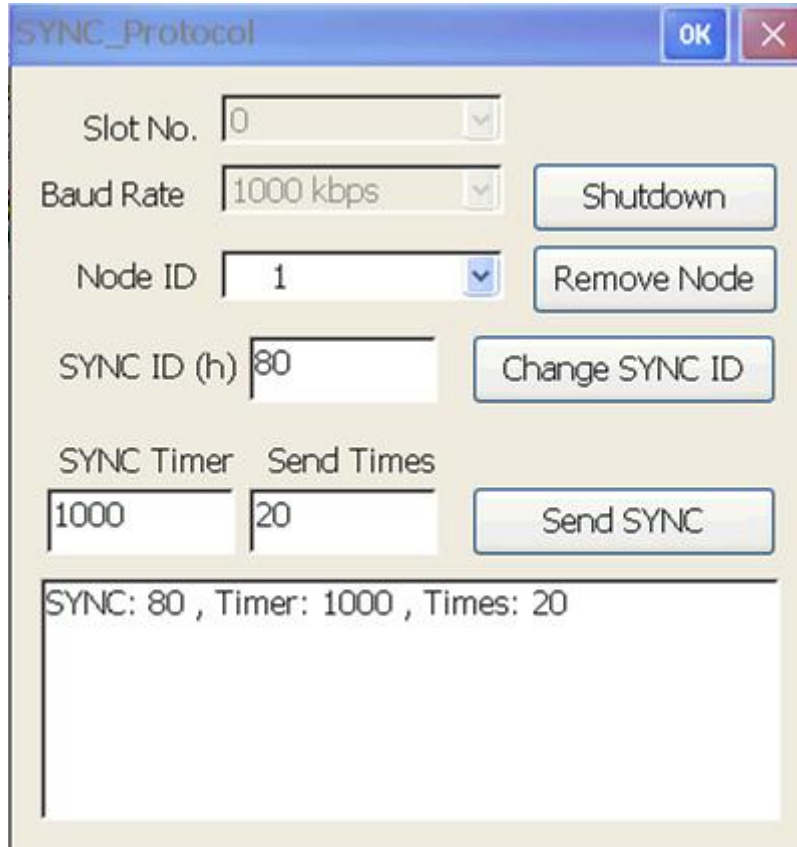
Clear

Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#), [CPM\\_SDOWriteData](#).

## 4.1.9 SYNC\_Protocol

SYNC protocol is a synchronous function of the PDO communication. It is always used with the transmission type of the PDO communication. In this demo, users can know how to use the SYNC related functions.



The screenshot shows a software window titled "SYNC\_Protocol" with standard Windows window controls (OK, X). The window contains the following configuration options:

- Slot No.: 0
- Baud Rate: 1000 kbps
- Node ID: 1
- SYNC ID (h): 80
- SYNC Timer: 1000
- Send Times: 20

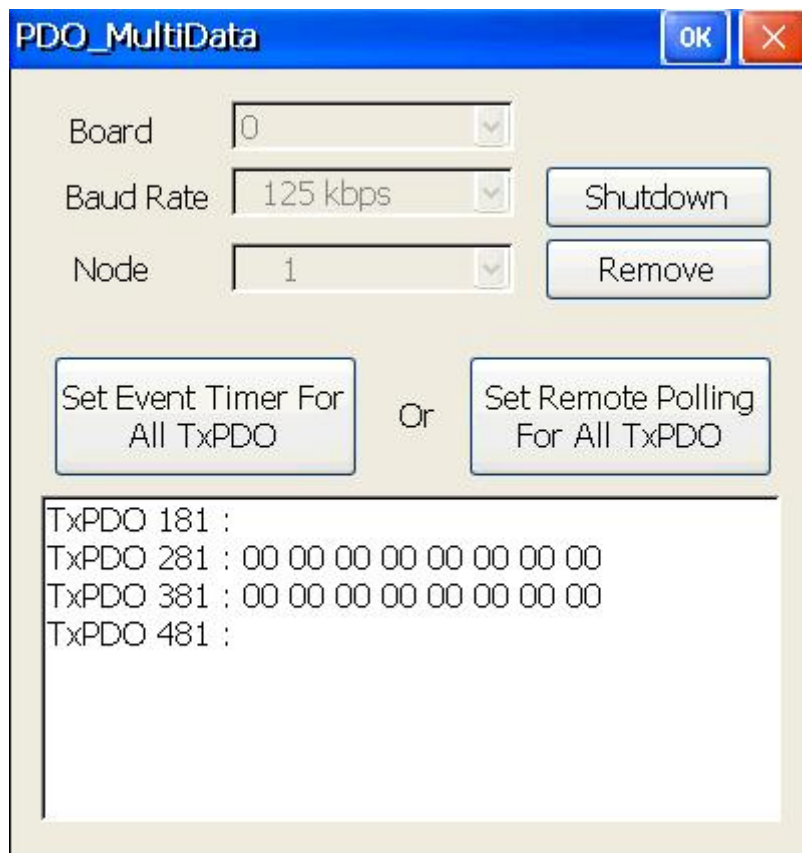
Buttons available in the window include "Shutdown", "Remove Node", "Change SYNC ID", and "Send SYNC". A status box at the bottom of the window displays the current configuration: "SYNC: 80 , Timer: 1000 , Times: 20".

Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_ChangeSYNCID](#), [CPM\\_GetSYNCID](#), [CPM\\_SendSYNCMsg](#),  
[CPM\\_GetCyclicSYNCInfo](#).

#### 4.1.10 PDO\_MultiData

Sometimes, users want to poll several PDO objects data at the same time for increasing the performance. But it is slower than sending the Remote PDO to poll each PDO data one by one. So users can set event timer or remote list for these PDO. When the PDO data are polled by the Master or are replied from slave automatically, then use the CPM\_GetMultiPDOData function to obtain these PDO data from the buffer at the same time.



Applied function list:

[CPM\\_InitMaster](#), [CPM\\_Shutdown](#), [CPM\\_AddNode](#), [CPM\\_RemoveNode](#),  
[CPM\\_GetTxPDOID](#), [CPM\\_SetPDORemotePolling](#), [CPM\\_PDODoEventTimer](#),  
[CPM\\_GetMultiPDOData](#)