PISO-CPM100U-D/T
PCM-CPM100-D
CANopen Master PCI/ PCI-104 Card

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.

Trademark

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

Tables of Content

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.

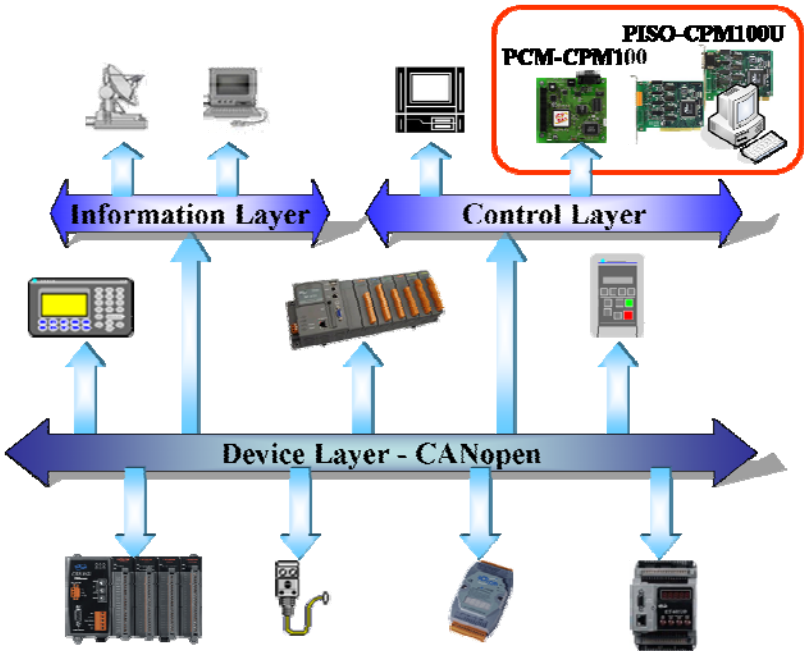


Figure 1.1 Example of the CANopen network

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 for 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. CPM100 Library Characteristics

In order to use the PCI CAN board of PISO-CPM100(U) and the PCI-104 CAN board of PCM-CPM100, we provide CPM100 library for VC, VB.net and C# development, and users can use it to establish the CANopen communication network rapidly. Most of the CANopen communication protocols, such as PDO, SDO and NMT, would be handled by the library automatically. Therefore, it is helpful to reduce the complexity of developing a CANopen master interface, and let users ignore the detail CANopen protocol technology. This library mainly supports connection sets of master-slave architecture, which include some useful functions to control the CANopen slave device in the CANopen network. The following figure describes the general application architecture of CPM100 series.

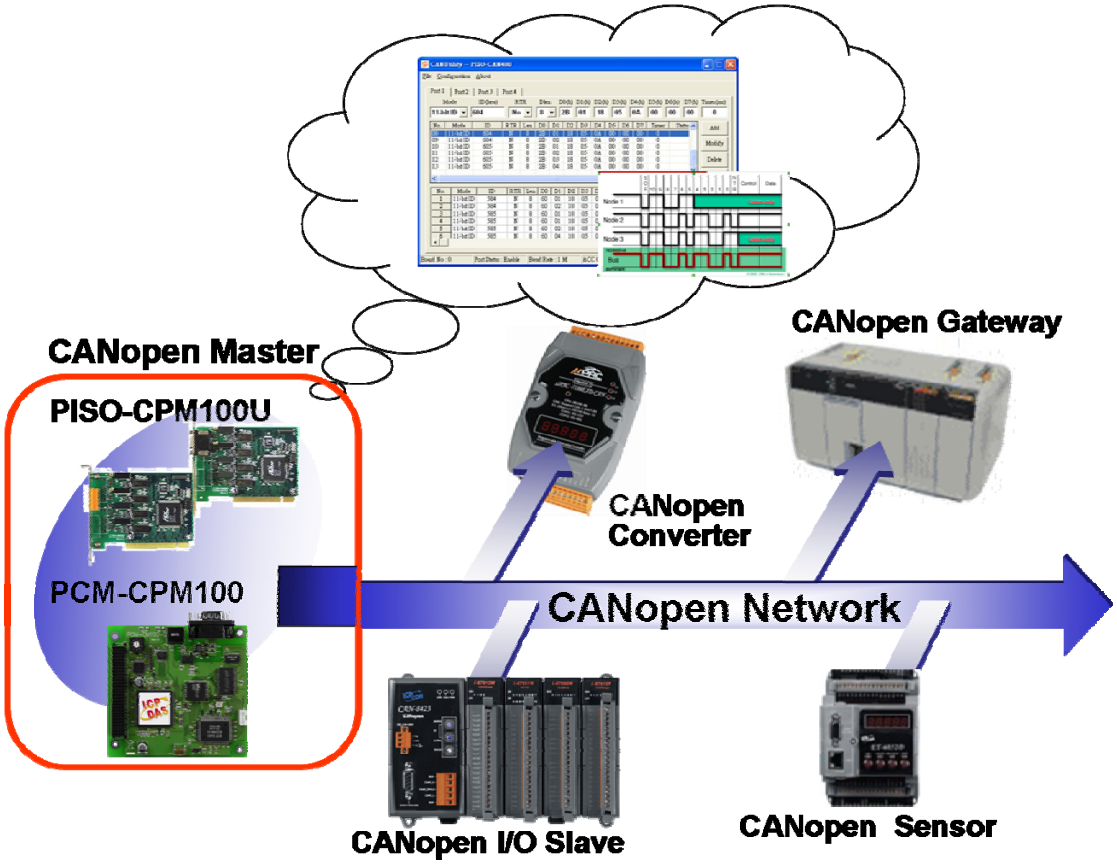


Figure 1.2 Example of application architecture

The CPM100 series 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.

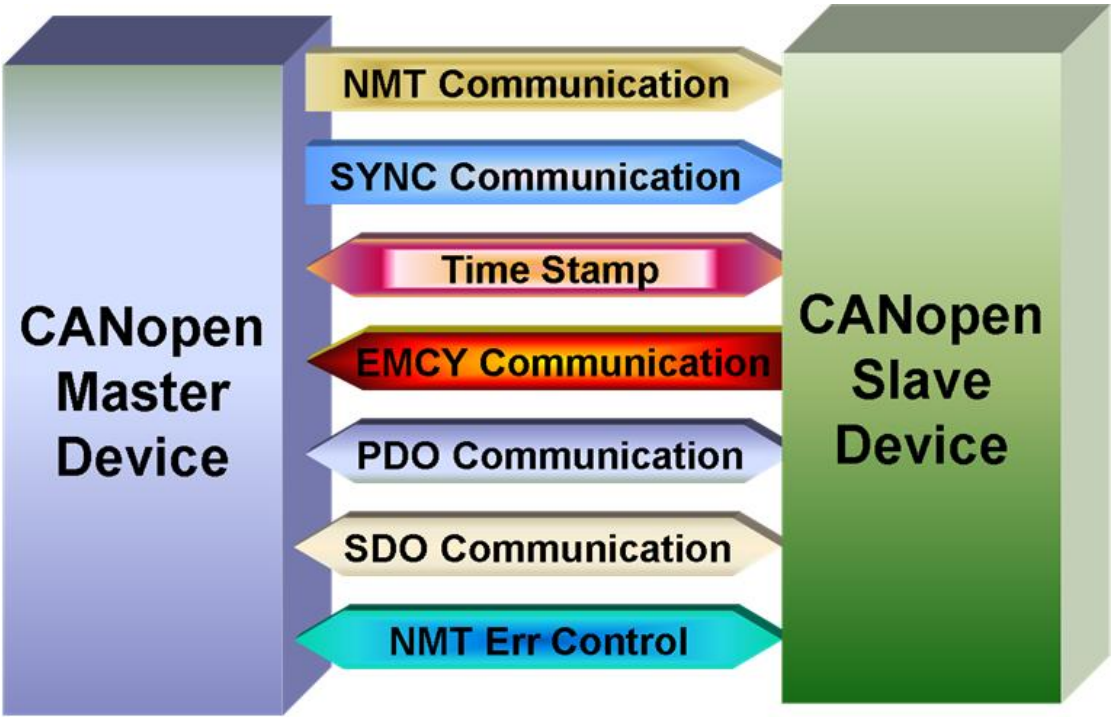


Figure 1.3 CANopen communication general concept

- **Node Manager (NMT Master)**
  - Functions for changing the slave device state
  - Node Guarding and Heartbeat Protocol for error control
  - Support Emergency (EMCY) messages
- **SDO Manager**
  - Expedited, segmented and block methods for SDO download and upload
- **PDO Manager**
  - Support all transmission types and event timer
- **SYNC Manager**
  - SYNC message production
  - SYNC cycles of 0.1ms resolution
- **EMCY Manager**
  - EMCY message 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.

Specifications

- PISO-CPM100-D/T:
  - 33 MHz 32bit 5 V PCI bus (V2.1) plug and play technology.
  - Connector: 5-pin screw terminal or 9-pin D-sub male connector.
- PISO-CPM100U-D/T:
  - Universal PCI card supports both 5 V and 3.3 V PCI bus.
  - Connector: 5-pin screw terminal or 9-pin D-sub male connector.
- PCM-CPM100-D:
  - PCI-104 card supports both 5 V and 3.3 V PCI bus.
  - Connector: 9-pin D-sub male connector.
- CPU: 80186 compactable CPU, 80 MHz.
- CAN controller: NXP SJA1000T with 16 MHz.
- CAN transceiver: NXP 82C250.
- CAN bus interface: Follow ISO 11898-2 specification.
- Isolation voltage: 2500 Vrms photo-isolation protection on CAN side.
- Power requirements: 5 V@ 400 mA.
- Operating Temperature: 0 ~ +60 ℃.
- Storage Temperature: -20 ~ +80 ℃.
- Humidity: 0 ~ 90% non-condensing.
- Dimensions: please refer to section 2.1.

Features

- One CAN communication port.
- 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 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.
- Timestamp of CAN message with at least ±1ms precision.
- Jumper select 120Ω terminator resistor for CANopen network.
- Support firmware update.
- Two indication LEDs (Tx/Rx and Err LEDs).

- 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. Hardware Configuration

This section would describe the hardware setting of the PISO-CPM100(U) and PCM-CPM100. This information includes the wire connection and terminal resistance configuration for the CAN network.

2.1. Board Layout

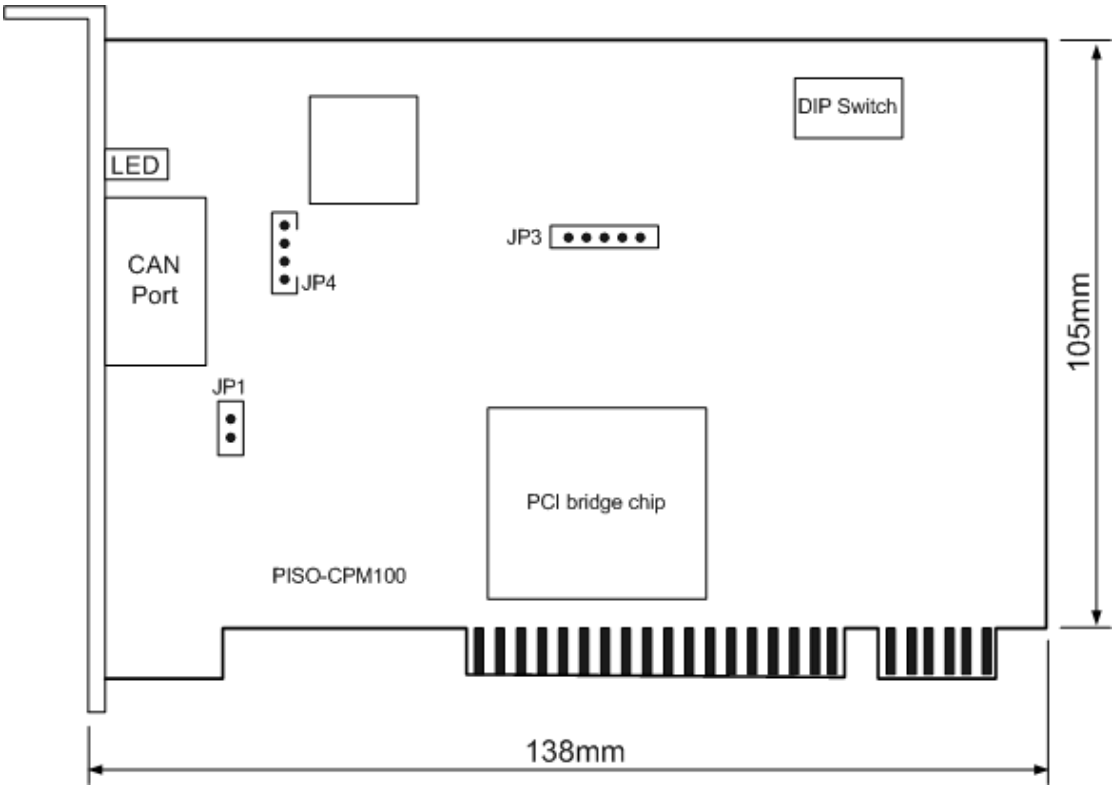


Figure 2.1 PISO-CPM100 board layout

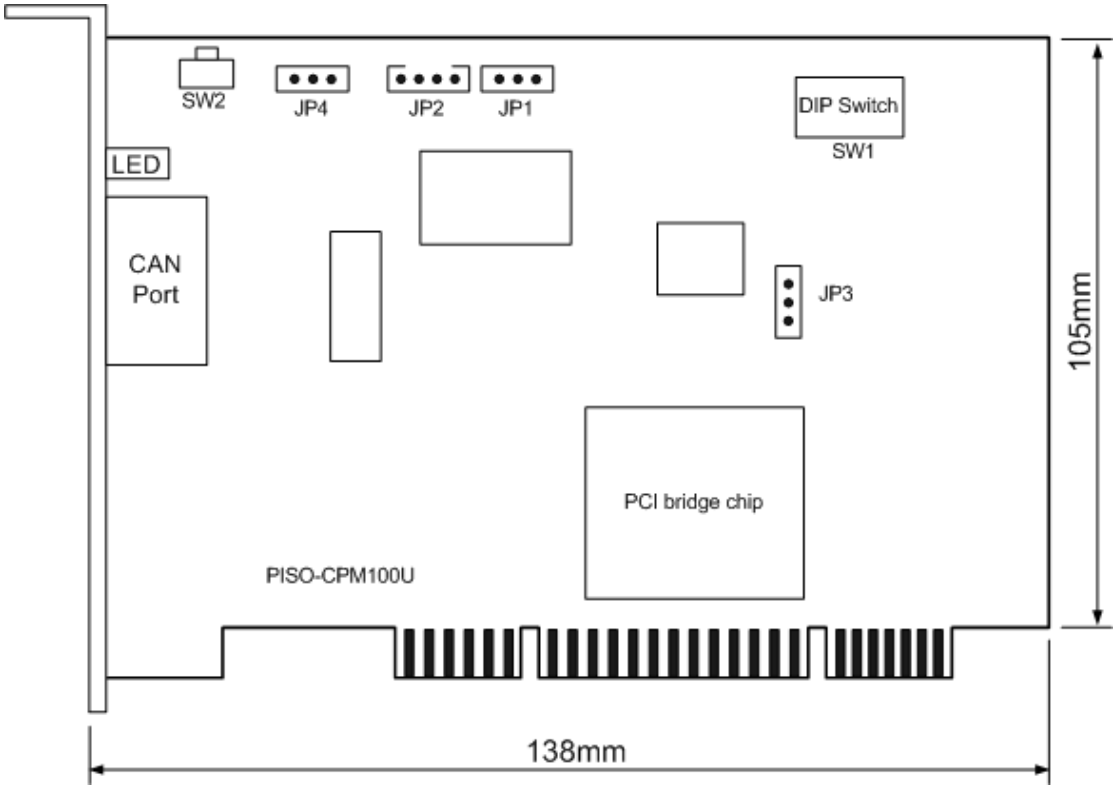


Figure 2.2 PISO-CPM100U board layout

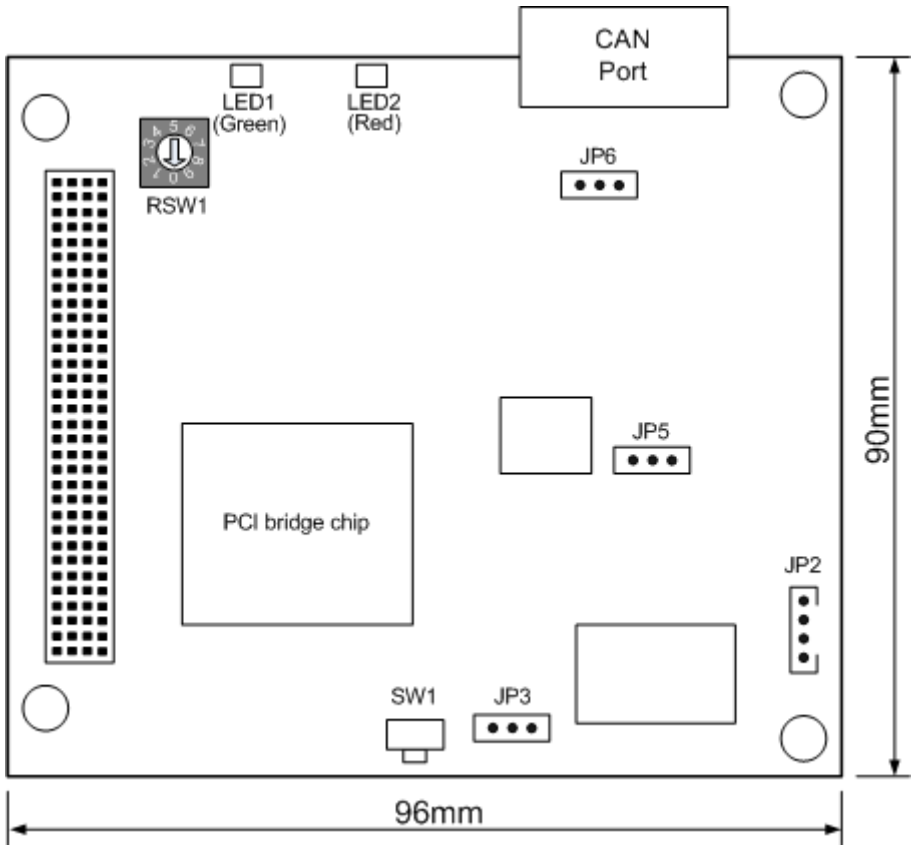


Figure 2.3 PCM-CPM100 board layout

Jumper Selection

The following table shows the definition of jumpers and DIP switch. Users need to refer to this table to configure the CPM100 series hardware.

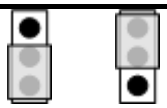
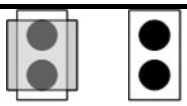
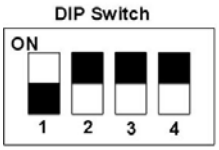
Name	Jumper	Status	Description
PISO-CPM100U	JP3	 Enable Disable	The Flash protection jumper. If users would like to protect the data of the Flash. Enable this jumper. In this case, the firmware can't be updated from the utility.
PCM-CPM100	JP5	 Enable Disable	
PISO-CPM100	JP1	 Enable Disable	The 120Ω terminal resistance of the CAN bus.
PISO-CPM100U	JP4	 Enable Disable	
PCM-CPM100	JP6	 Enable Disable	
PISO-CPM100	JP3	 Enable Disable	Reset button / pin for download error. If users want to update firmware but the process is fail, users can enable this jumper to reset CPM100 into download mode.
PISO-CPM100U	SW2		
PCM-CPM100	SW1		
PISO-CPM100	DIP switch	 The situation indicates the board No. 1.	The DIP switch is for the PISO-CM100U board No. If the left-hand-side switch (bit No. 1) is ON, the board No. is set to 1. The range of the board No. is from 0 to 15.
PISO-CPM100U			
PCM-CPM100			

Table 2.1 Jumper or DIP switch selections

Connector Pin Assignment

The CPM100 series have two kinds of connector. One is 5-pin screw terminal connector and the other is 9-pin D-sub male connector for wire connection of the CANopen network. The connector's pin assignment is specified as follows:

2.1.1. 5-pin screw terminal connector

The 5-pin screw terminal connector of the CAN bus interface is shown in figure 2.2. The details for the pin assignment are presented in the following table.

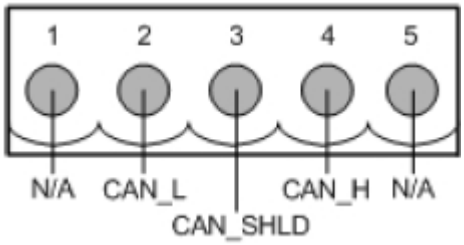


Figure 2.2 5-pin screw terminal connector

Pin No.	Signal	Description
1	N/A	No use
2	CAN_L	CAN_L bus line (dominant low)
3	CAN_SHLD	Optional CAN shield
4	CAN_H	CAN_H bus line (dominant high)
5	N/A	No use

Table 2.2 Pin assignment of 5-pin screw terminal connector

2.1.2. 9-pin D-sub male connector

The 9-pin D-sub male connector of the CAN bus interface is shown in figure 2.3 and the corresponding pin assignments are given in following table.

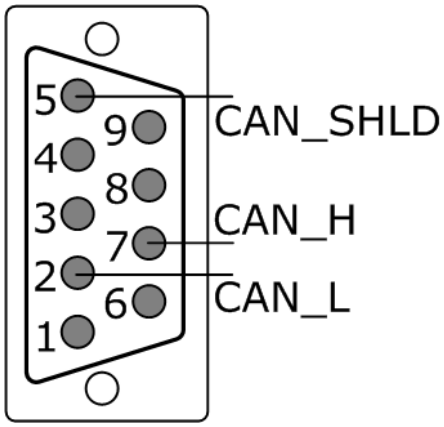


Figure 2.3 9-pin D-sub male connector

Pin No.	Signal	Description
1	N/A	No use
2	CAN_L	CAN_L bus line (dominant low)
3	N/A	No use
4	N/A	No use
5	CAN_SHLD	Optional CAN Shield
6	N/A	No use
7	CAN_H	CAN_H bus line (dominant high)
8	N/A	No use
9	N/A	No use

Table 2.3 Pin assignment of the 9-pin D-sub male connector

2.1.3. Wire Connection

In order to minimize the reflection effects on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminal resistances as in the following figure. According to the ISO 11898-2 spec, each terminal resistance is 120Ω(or between 108Ω~132Ω). The length related resistance should have 70mΩ/m. Users should check the resistances of the CAN bus, before they install a new CAN network.

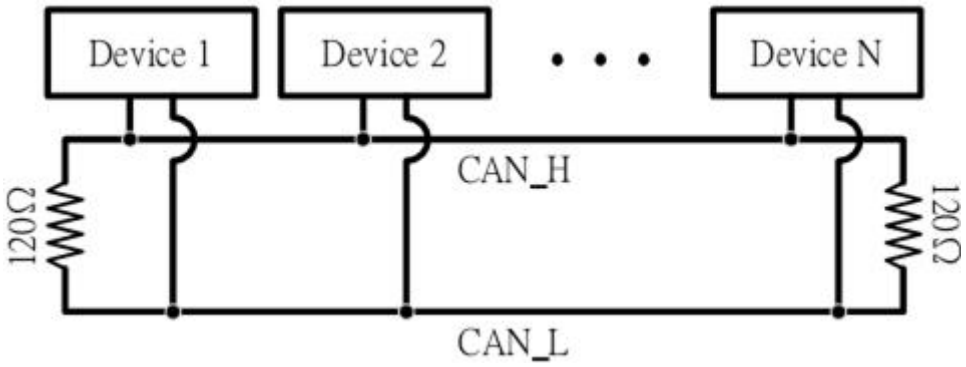


Figure 2.4 CANopen network topology

Moreover, to minimize the voltage drop over long distances, the terminal resistance should be higher than the value defined in the ISO 11898-2. The following table can be used as a good reference.

Bus Length (meter)	Bus Cable Parameters		Terminal Resistance (Ω)
	Length Related Resistance (mΩ/m)	Cross Section (Type)	
0~40	70	0.25(23AWG)~0.34 mm^2 (22AWG)	124 (0.1%)
40~300	<60	0.34(22AWG)~0.6 mm^2 (20AWG)	127 (0.1%)
300~600	<40	0.5~0.6mm^2 (20AWG)	150~300
600~1K	<20	0.75~0.8mm^2 (18AWG)	150~300

Table 2.4 Relationship between cable feature and terminal resistance

2.2. Green LED

After CPM100 has been activated, the greed LED would be flashed once when CPM100 receives or transmits one message to CAN bus successfully. If the bus loading is heavy, the green LED would turn on always.

2.3. Red LED

When some error occurs, the red LED would turn on until the error has been solved. Users can use CPM100_GetCANStatus function to get the situation except buffer status.

2.4. Hardware Installation

When users want to use CPM100, the hardware installation needs to be finished as following steps.

1. Shutdown your personal computer.
2. Configure the DIP switch for the board No. and the terminal resistance. The more detail information could be found on the section 2.1.
3. Find an empty PCI / PCI-104 slot for the CPM100 on the mother board of the personal computer and plug the configured CPM100 into this empty PCI / PCI-104 slot.
4. Plug the CAN bus cable(s) into the 5-pin screw terminal connector or the 9-pin D-sub connector.

When the procedure described above is completed, turn on the PC.

3. Software Installation

3.1. Software Structure

The CPM DLL driver is the CANopen specification function collections for the CPM100 cards used in Windows 2000/XP/Vista/7 with 32-bit 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: VC++, VB.net, and C#. The driver architecture is shown in the following Figure.

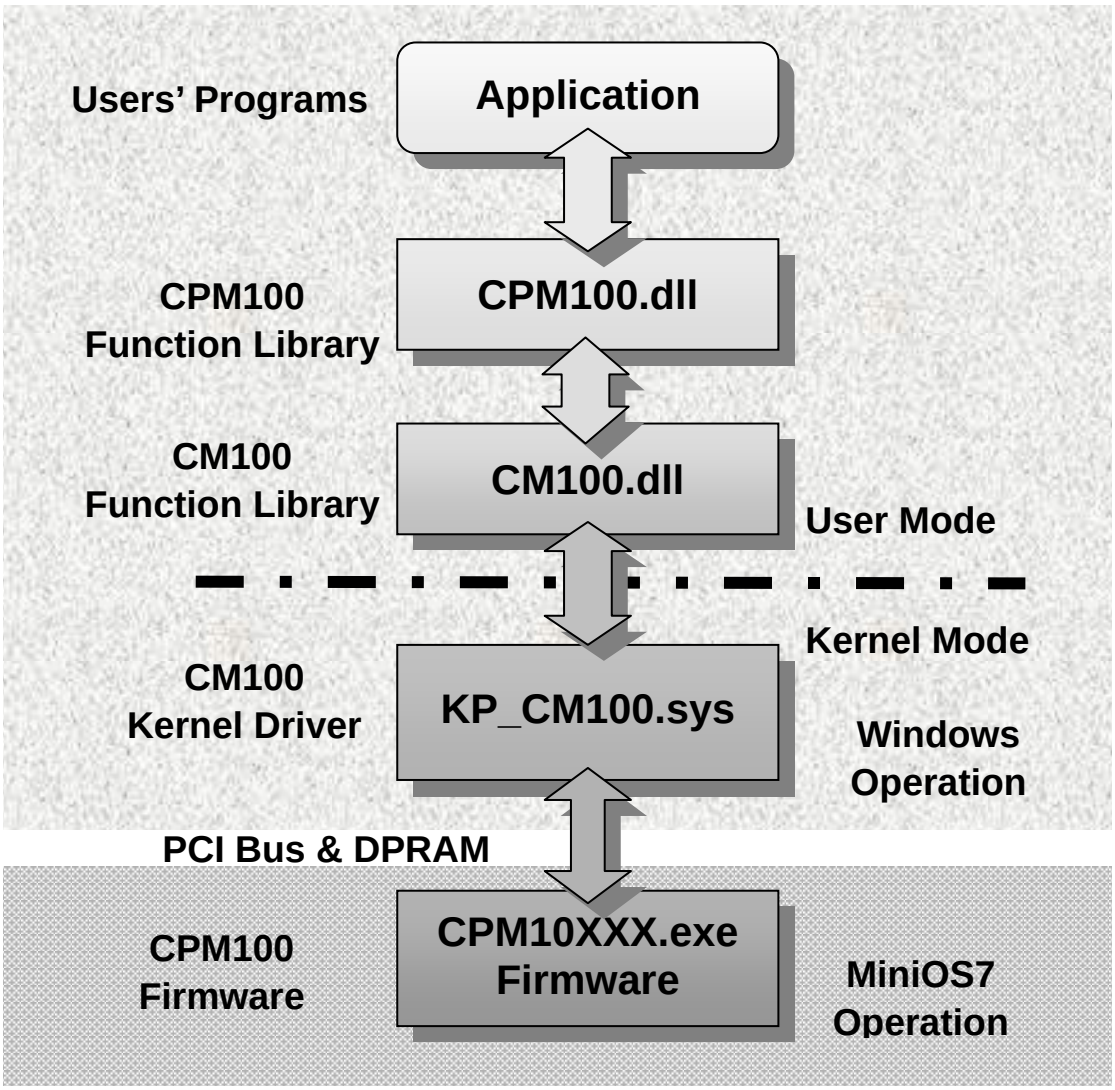


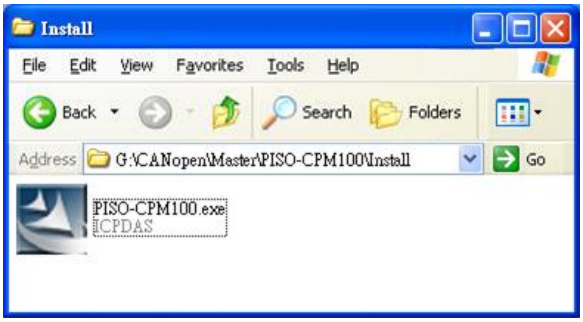
Figure 3.1 Driver concept of CPM100

3.2. Installation Driver Step by Step

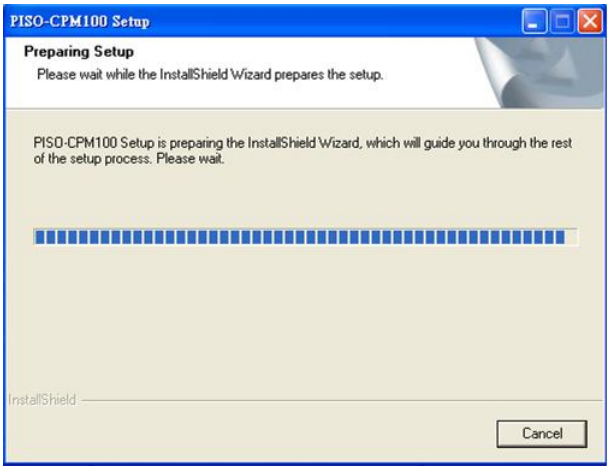
When users want to use the CPM100 CAN card, the CPM100 driver must be installed firstly. After finishing the installation process, the CPM_Utility and the demo programs would also be installed to the PC. The demo programs may be a good reference for users to build their CANopen master interface by using 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 develop their CANopen master application. If users do not want to develop this application by themselves, the CPM_Utility can be used to be an easily CANopen master program. The following description displays the step-by-step procedures about how to install the CPM100 driver.

Install the CPM100 CAN card driver

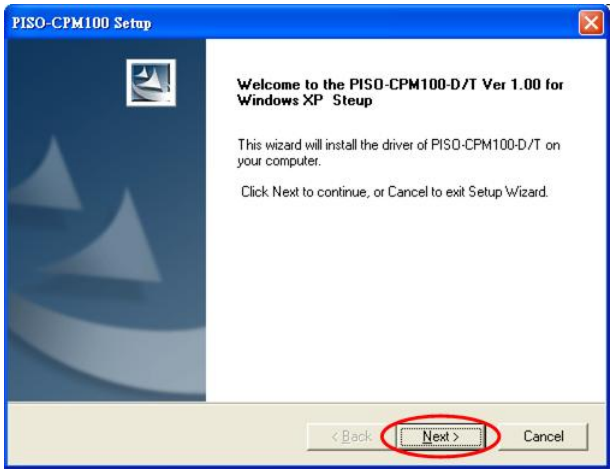
Step 1: Insert the product CD into the CD-ROM and find the path \CANopen\Master\PISO-CPM100\Install\. Then execute the PISO-CPM100.exe to install the CPM100 CAN card driver.



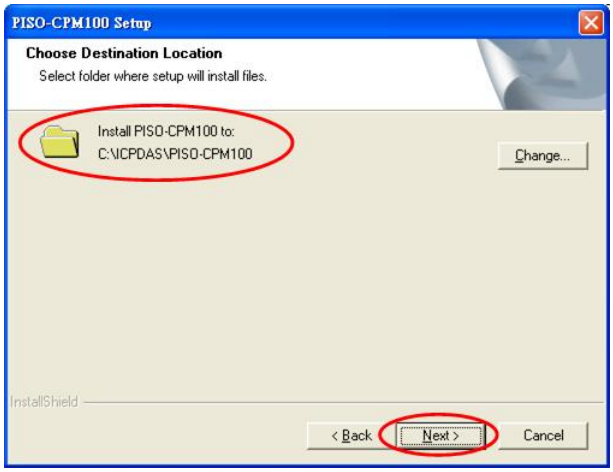
Step 2: Wait until the install wizard has prepared.



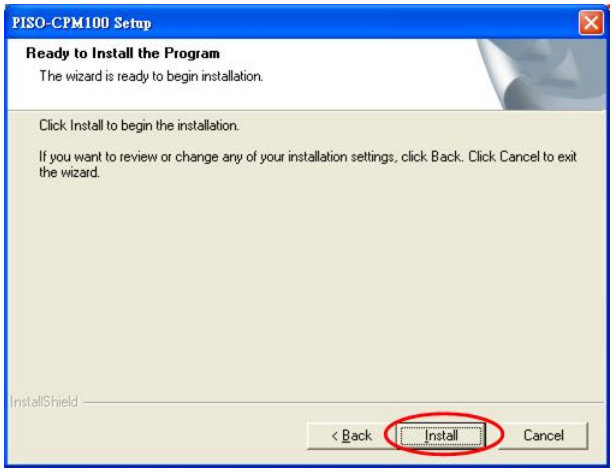
Step 3: Click “Next” to start the CPM100 installation.



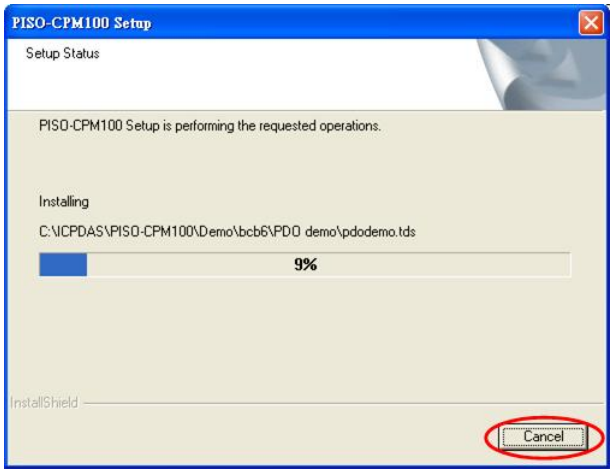
Step 4: Select the folder where the CPM100 setup would be installed and click “Next” button to continue.



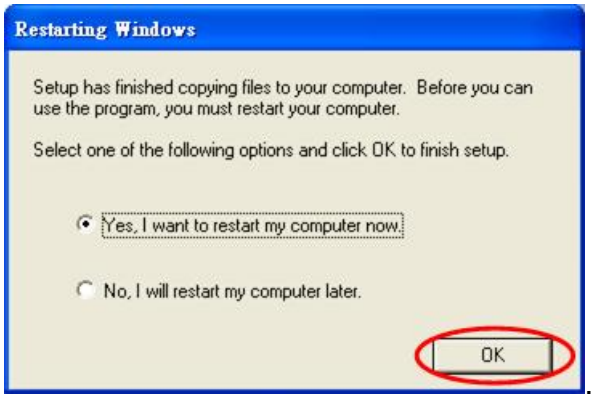
Step 5: Click the button “Install” to continue.



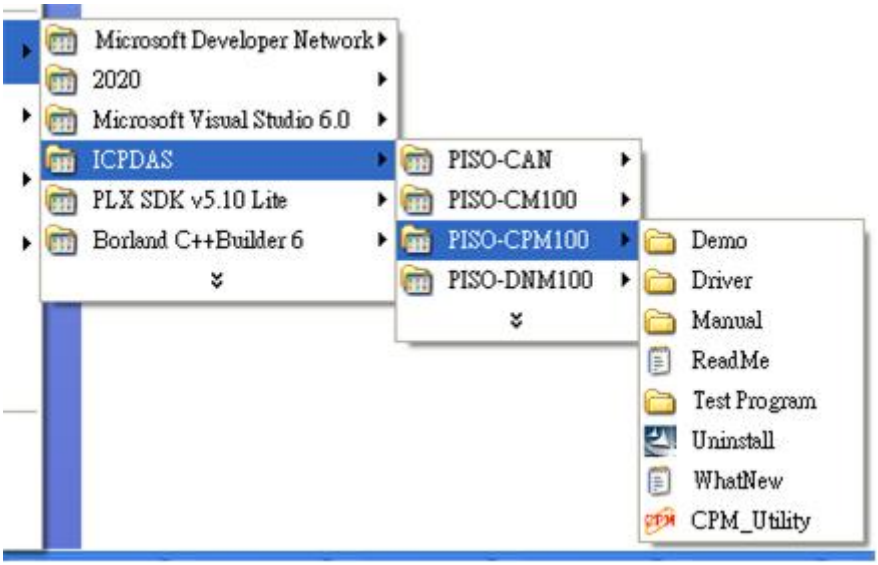
Step 6: Wait for finishing the CPM100 installation.



Step 7: Finally, restart the computer to complete the installation.

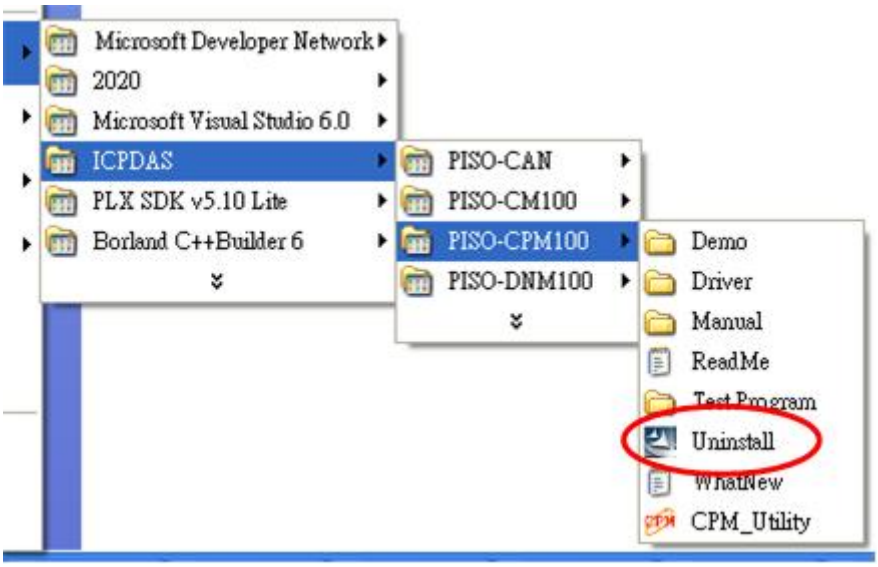


Step 8: When finishing the installation. The CPM100 folder would be found at the Start menu shown as below.



Remove the CPM100 driver

If the CPM100 driver is not used any more, users can click the "Uninstall" to remove the CPM100 driver below.



4. CPM100 Function Library

4.1. Function List

In order to use the CPM100 more easily, we provide some useful and easy-to-use functions in CPM100 library. There are three function libraries for different compiler, such as VC, VB and BCB. Users can use these functions to control the CPM100 by using the functions. The following table shows all functions provided by the CPM100 library.

Listen Mode	Function Name	Description
O	CPM100_GetVersion	Get the version of the CPM100 library
O	CPM100_TotalBoard	Get the total number of CPM100
O	CPM100_GetBoardSwitchNo	Get the board number from the dip switch number of the CPM100
O	CPM100_GetBoardInf	Get board information of the CPM100
O	CPM100_GetCANStatus	Obtain the status of the CAN controller
O	CPM100_SetFunctionTimeout	Set the max. timeout value of all functions
O	CPM100_InitMaster	Activate the CPM100 module
O	CPM100_ShutdownMaster	Remove all nodes and stop the master
X	CPM100_MasterSendBootupMsg	Let CPM100 sent a boot up message
O	CPM100_SetMasterMode	Set the master to normal mode or listen mode
O	CPM100_GetMasterMode	Get operation mode of the master
O	CPM100_GetFirmwareVersion	Get the version of the CPM100 firmware
O	CPM100_EDS_Load	Add a slave node from the EDS file
O	CPM100_AddNode	Add a slave node into the CPM100 master manager
O	CPM100_RemoveNode	Remove a node from the CPM100 master manager
X	CPM100_RemoveAndResetNode	Remove a node from the CPM100 master manager and then reset it
O	CPM100_DelayAndResponseTimeout	Change minimum interval time and response timeout value of CAN messages
X	CPM100_ScanNode	Scan all nodes on the CANopen network
O	CPM100_GetNodeList	Get the node list of the CPM100 master manager
X	CPM100_NMTChangeState	Change the CANopen node state
O	CPM100_NMTGetState	Get the CANopen node state

O	CPM100_NMTGuarding	Start to the node guarding function
O	CPM100_NMTHeartbeat	Start to the node heartbeat function
X	CPM100_SDOReadData	Read data by using the upload SDO protocol
X	CPM100_SDOReadFile	Read the huge SDO data for specific slave node
X	CPM100_SDOWriteData	Write data by using the download SDO protocol
X	CPM100_SDOAbortTransmit	Send the SDO abort message
X	CPM100_PDOWrite	Use the PDO to write data to the CANopen node
X	CPM100_PDOWrite_Fast	Only send out the RxPDO but not do any check
X	CPM100_PDORemote	Use the PDO to get data from the CANopen node
X	CPM100_PDORemote_Fast	Only send the Rtr message but don't check response
X	CPM100_SetPDORemotePolling	Set PDO polling list table and poll them
O	CPM100_GetPDOLastData	Get the last input or output PDO data
O	CPM100_GetPDOLastData_Fast	Get the last input or output PDO data in fast mode
O	CPM100_GetMultiPDOData	Get the several input or output PDO data once
O	CPM100_GetMultiPDOData_Fast	Get the several input or output PDO data once in fast mode
O	CPM100_GetRxPDOID	Get all COB-ID of the RxPDO of the specific slave
O	CPM100_GetTxPDOID	Get all COB-ID of the TxPDO of the specific slave
X	CPM100_InstallPDO	Install and enable a specific PDO
X	CPM100_DynamicPDO	New or change a PDO mapping to the slave
X	CPM100_RemovePDO	Remove a specific PDO mapping entry or object
X	CPM100_ChangePDOID	Change the PDO COB-ID of a specific slave
O	CPM100_GetPDOMapInfo	Obtain all the PDO-related information
O	CPM100_InstallPDO_List	Install a PDO manually without check if it exists
O	CPM100_RemovePDO_List	Remove PDO object without check real states
X	CPM100_PDOUseEntry	Change the valid entry number of the PDO objects
X	CPM100_PDOTxType	Set transmission type of the specific TxPDO
X	CPM100_PDOEventTimer	Set event timer of the specific TxPDO
X	CPM100_PDOInhibitTime	Set inhibit time of the specific TxPDO
X	CPM100_ChangeSYNCID	Change the SYNC COB-ID
O	CPM100_SetSYNC_List	Set the SYNC COB-ID without check if it exists
O	CPM100_GetSYNCID	Get the SYNC COB-ID
X	CPM100_SendSYNCMsg	Send the SYNC message
X	CPM100_GetCyclicSYNCInfo	Get all the cyclic sending SYNC information
X	CPM100_ChangeEMCYID	Change the EMCY COB-ID

O	CPM100_SetEMCY_List	Set the EMCY COB-ID and don't check if it exists
O	CPM100_GetEMCYID	Get the EMCY COB-ID
O	CPM100_ReadLastEMCY	Get the last EMCY message of the slave
O	CPM100_GetBootupNodeAfterAdd	Get boot up message of node slave that had added in CPM100 node list.
O	CPM100_GetEMCYData	Get the EMCY message from the EMCY buffer
O	CPM100_GetNMTError	Get the NMT Error event from the NMT event buffer
O	CPM100_InstallBootUpISR	Install user-defined slave boot up process
O	CPM100_RemoveBootUpISR	Remove the slave boot up process
O	CPM100_InstallEMCYISR	Install user-defined EMCY process
O	CPM100_RemoveEMCYISR	Remove the EMCY process
O	CPM100_InstallNMTErrISR	Install user-defined Guarding/Heartbeat event process
O	CPM100_RemoveNMTErrISR	Remove the Guarding/Heartbeat event process
O	CPM100_GetMasterReadSDOEvent	Get the read SDO message sent to the CPM100
O	CPM100_GetMasterWriteSDOEvent	Get the write SDO message sent to the CPM100
X	CPM100_ResponseMasterSDO	Response the SDO message to the SDO sender
O	CPM100_GetMasterRemotePDOEvent	Get the remote PDO message send to the CPM100
O	CPM100_GetMasterRxPDOEvent	Get the RxPDO RTR sent to the CPM100
X	CPM100_ResponseMasterPDO	Response the RxPDO RTR to the RTR sender
O	CPM100_InstallRxSDOISR	Install the user-defined Master SDO process
O	CPM100_RemoveRxSDOISR	Remove the master SDO process
O	CPM100_InstallRxPDOISR	Install the user-defined Write Master PDO process
O	CPM100_RemoveRxPDOISR	Remove the Write Master PDO process
O	CPM100_InstallTxPDOISR	Install the user-defined Remote Master PDO process
O	CPM100_RemoveTxPDOISR	Remove the Remote Master PDO process

Table 4.1 Description of functions

4.2. Function Return Code

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

Return Code	Error ID	Description
0	CPM_NoError	OK
1	CPM_DriverError	Kernel driver is not opened
3	CPM_BoardNumberErr	There is no CPM100 on the specific board No.
5	CPM_ConfigErr	The CPM100 hasn't been configured successfully
6	CPM_MasterInitErr	The CPM100 initialization error.
7	CPM_MasterNotInit	The CPM100 hasn't been initialized
8	CPM_ListenMode	The CPM100 is in listen mode now
9	CPM_NodeErr	Set node number error
10	CPM_NodeExist	The node had been added to the Master
12	CPM_TxBusy	Tx buffer is busy, please wait a minute to send again
13	CPM_UnknowCmd	This version of firmware doesn't support the function
14	CPM_CmdReceErr	CPM100 receive command of wrong length
15	CPM_DataEmpty	There is no data to receive
16	CPM_MemAllocErr	CPM100 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_PDOInstErr	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_PDOEntryErr	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 4.2 Description of return code

4.3. 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 5.

When users' programs apply the CANopen Master Library functions, the function CPM100_InitMaster must be called first. The functions is used to initialize CPM100 and configure the CAN port.

After initializing the CAN interface successfully, users need to use the function CPM100_AddNode or CPM100_EDS_Load to install at least one CANopen device into the node list. The function CPM100_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 CPM100_InstallPDO_List, CPM100_SetSYNC_List, and CPM100_SetEMCY_List.

If the function CPM100_InitMaster and CPM100_AddNode/ CPM100_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 CPM100_ShutdownMaster, because the CPM100_ShutdownMaster would stop all process created by the function CPM100_InitMaster.

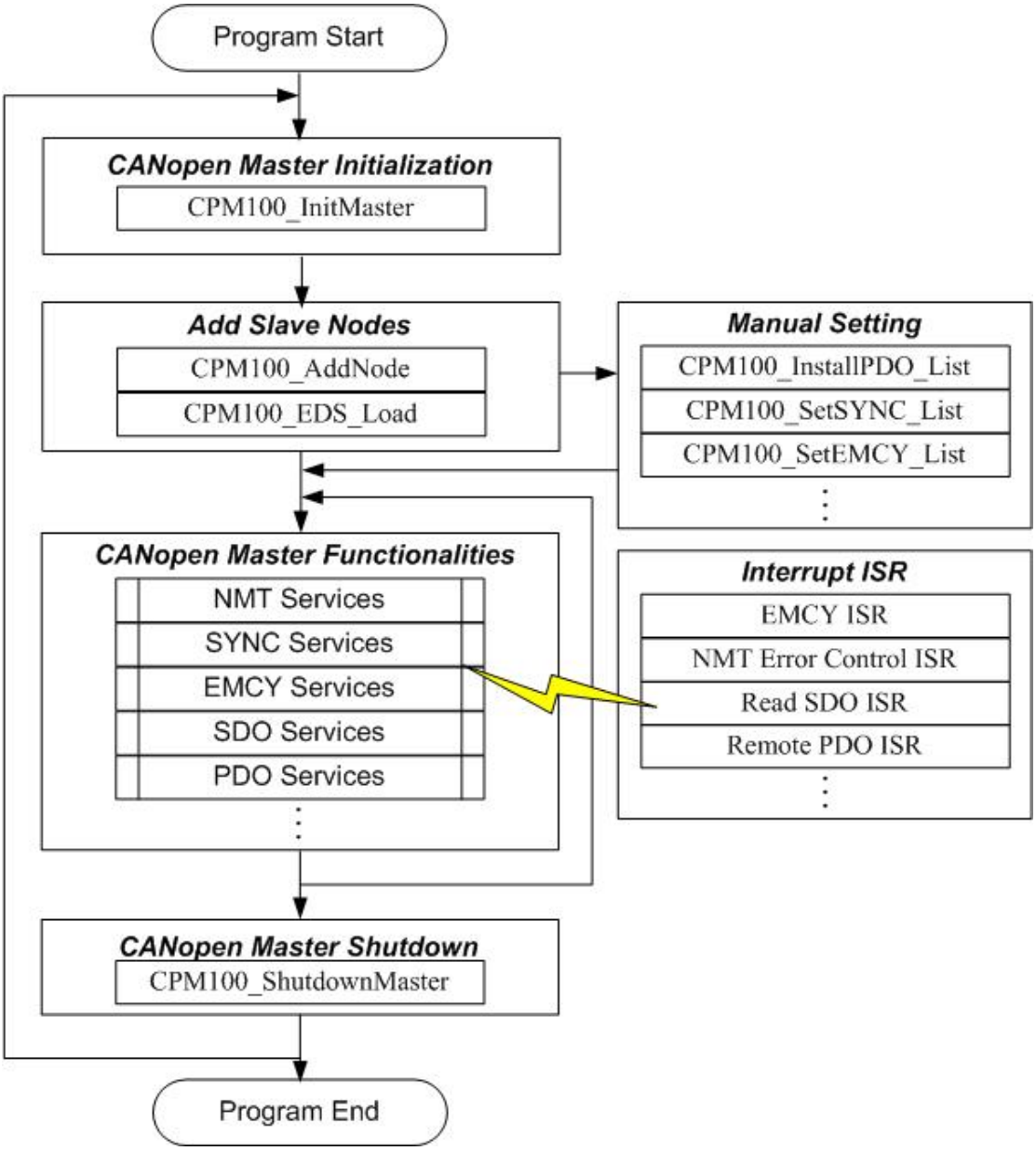


Figure 4.1 Main programming sequences

4.4. Communication Services Introduction

NMT Services

The CANopen Master Library provides several NMT services functions, such as the functions CPM100_AddNode, CPM100_EDS_Load, CPM100_RemoveNode, CPM100_NMTChangeState, CPM100_NMTGetState, CPM100_NMTHeartbeat, and CPM100_NMTGuarding. As the prerequisite for the master, the slave nodes have to be registered by the function CPM100_AddNode or CPM100_EDS_Load with providing its Node-ID. The registered slave nodes can be individually removed from the node list by the function CPM100_RemoveNode. Through NMT services, the NMT Master controls the state of the slaves. Table 4.3 is the command value and corresponding NMT command for the input parameters of the CPM100_NMTChangeState function. When using the function CPM100_NMTGetState, the slave status value and their descriptions are shown in the table 4.4. The Node Guarding and Heartbeat protocol are implemented via the function CPM100_NMTGuarding and the function CPM100_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 CPM100_NMTGuarding or CPM100_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 4.3 NMT Command Specifier

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

Table 4.4 State of the Slave

SDO Services

"Initiate SDO download protocol" or "Initiate SDO upload protocol" is used when the object data length ≦ 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, CPM100_SDOReadData and CPM100_SDOWriteData, the initial protocol and segment protocol will be selected automatically according to the object data length.

CPM100_SDOAbortTransmit function can abort a pending SDO transfer at any time. Applying the abort service will have no confirmation from the salve.

PDO Services

After using the CPM100_AddNode to add a slave, the default TxPDOs and RxPDOs (TxPDO 1 ~ 10, RxPDO 1 ~ 10) will also be added to the CPM100's control list. If there are PDOs other than the default setting, the function CPM100_InstallPDO is used for enabling these TxPDOs or RxPDOs object. After installing the PDOs, the function CPM100_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 DS401, and they are mapped to the relative parameters of the DI/DO/AI/AO channels. After calling the function CPM100_InstallPDO and CPM100_DynamicPDO, the PDO communication object will be mapped and activated. If the PDO communication object is not needed no more, use the function CPM100_RemovePDO to remove it.

When you want to output data via PDO, the function CPM100_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 CPM100 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 CPM100_PDORemote is needed in this case. Or you can use CPM100_GetPDOLastData to get the last RxPDO and TxPDO data from the PDO buffer.

SYNC Services

Calling the function CPM100_SendSYNCMsg starts the SYNC object transmission. This function supports single SYNC message transmission and cyclic SYNC message transmission. The parameter “Timer” of the function CPM100_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 CPM100_ReadLastEMCY to receive the EMCY message or the function CPM100_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.

4.5. Function Description

4.5.1. CPM100_GetVersion

- **Description:**

  This function is used to obtain the version information of the CPM100.lib library.

- **Syntax:**

  WORD CPM100_GetVersion(**void**)

- **Parameter:**

  None

- **Return:**

  LIB library version information.

4.5.2. CPM100_TotalBoard

- **Description:**

  Obtain the total board number of CPM100 plugged in the PCI bus.

- **Syntax:**

  WORD CPM100_TotalBoard(**void**)

- **Parameter:**

  None

- **Return:**

  Return the scanned total CPM100 number.

4.5.3. CPM100_GetBoardSwitchNo

- **Description:**

  Obtain the DIP switch No. of CPM100.

- **Syntax:**

  WORD CPM100_GetBoardSwitchNo(**BYTE** BoardCntNo,
  BYTE *BoardNo)

- **Parameter:**

  BoardCntNo: [input] The number of specified CPM100. For example, if the first CPM100 is applied, this value is 0. If the second board is applied, this value is 1.

  ***BoardNo:** [output] The address of a variable used to get the DIP switch No. of CPM100.

4.5.4. CPM100_GetBoardInf

- **Description:**

  Obtain the information of CPM100, which includes vendor ID, device ID and the interrupt number.

- **Syntax:**

  WORD CPM100_GetBoardInf(**BYTE** BoardNo, **DWORD** *dwVID,
  DWORD *dwDID, **DWORD** *dwSVID,
  DWORD *dwSDID, **DWORD** *dwSAuxID,
  DWORD *dwIrqNo)

- **Parameter:**

  BoardNo: [input] Switch No. of CPM100 DIP. The value is from 0 to 15.
  ***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.

4.5.5. CPM100_GetCANStatus

- **Description:**

  Obtain the status of the CAN controller of the specific CPM100 board.

- **Syntax:**

  int CPM100_GetCANStatus(**BYTE** BoardNo, **BYTE** *bStatus)

- **Parameter:**

  BoardNo: [input] CPM100 DIP switch No.(0~15).

  ***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

4.5.6. CPM100_SetFunctionTimeout

- **Description:**

  Sometimes, some function cost more time to complete its task, such as CPM100_ScanNode. If some API of the CPM100 library never gets the feedback from the CPM100 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. Default timeout value is 1 second.

- **Syntax:**

  void CPM100_SetFunctionTimeout(**BYTE** BoardNo, **WORD** Timeout)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  Timeout: [input] Maximum timeout value of all functions (ms).

4.5.7. CPM100_InitMaster

- **Description:**

  The function must be applied when configuring the CAN controller and initialize the CPM100. It must be called once before using other functions of the CPM100.lib.

- **Syntax:**

  WORD CPM100_InitMaster(**BYTE** BoardNo, **BYTE** Node, **BYTE** BaudRate, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  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 baudrate of the CPM100

Value	Baud rate
0	10Kbps
1	20Kbps
2	50Kbps
3	125Kbps
4	250Kbps
5	500Kbps
6	800Kbps
7	1Mbps

  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.8. CPM100_ShutdownMaster

- **Description:**

  The function CPM100_ShutdownMaster removes all the slaves that had added to master and stop all the functions of the CPM100. The function must be called before exit the users' application programs.

- **Syntax:**

  WORD CPM100_ShutdownMaster (**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.9. CPM100_ MasterSendBootupMsg

- **Description:**

  To use the function CPM100_MasterSendBootupMsg can let CPM100 sends a boot up message after CPM100_InitMaster is called.

  Note: The function is valid while the Node parameter of the function CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_MasterSendBootupMsg (**BYTE** BoardNo,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.10. CPM100_SetMasterMode

- **Description:**

  This function can configure if the master is into listen mode or normal mode (default mode). In listen mode, the CPM100 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 CPM100_InitMaster.

- **Syntax:**

  WORD CPM100_SetMasterMode(**BYTE** BoardNo, **BYTE** Mode,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.11. CPM100_GetMasterMode

- **Description:**

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

- **Syntax:**

  WORD CPM100_GetMasterMode(**BYTE** BoardNo, **BYTE** *Mode,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  ***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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.12. CPM100_GetFirmwareVersion

- **Description:**

  The function can let users know what is the version number of the CPM100's firmware.

- **Syntax:**

  WORD CPM100_GetFirmwareVersion (**BYTE** BoardNo,
  WORD *Fir_Ver, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  ***Fir_Ver:** [output] CPM100 firmware version information.
  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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.13. CPM100_EDS_Load

- **Description:**

  The function CPM100_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 CPM100_EDS_Load (**BYTE** BoardNo, **BYTE** Node,
  char *FilePath, **WORD** DelayTime,
  WORD ResTimeout, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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 CPM100. 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 CPM100. 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" again, it returns the process status. If the procedure is still not complete, it returns "CPM_Wait".

4.5.14. CPM100_AddNode

- **Description:**

The function CPM100_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 firmware of the CPM100 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 firmware of the CPM100. Users must call CPM100_SetSYNC_List, CPM100_SetEMCY_List, and CPM100_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 CPM100_RemoveNode.

- **Syntax:**

WORD CPM100_AddNode(**BYTE** BoardNo, **BYTE** Node,
BYTE AddMode, **WORD** DelayTime,
WORD ResTimeout, **BYTE** BlockMode)

- **Parameter:**

BoardNo: [input] CPM100 board number (0~7).
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 CPM100. 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 CPM100 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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.15. CPM100_RemoveNode

- **Description:**

  The function CPM100_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 CPM100_AddNode before.

- **Syntax:**

  WORD CPM100_RemoveNode(**BYTE** BoardNo, **BYTE** Node,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.16. CPM100_RemoveAndResetNode

- **Description:**

  The function CPM100_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 CPM100_AddNode before.

- **Syntax:**

  WORD CPM100_RemoveAndResetNode (**BYTE** BoardNo, **BYTE** Node, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.17. CPM100_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 CPM100_DelayAndResponseTimeout (**BYTE** BoardNo,
  BYTE Node, **WORD** DelayTime,
  WORD ResTimeout, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  Node: [input] Slave device Node-ID (1~127).
  DelayTime: [input] The parameter is the same as “DelayTime” of the function CPM100_AddNode.
  ResTimeout: [input] The parameter is the same as “ResTimeout” of the function CPM100_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” and “Node” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.18. CPM100_ScanNode

- **Description:**

  User can use the function CPM100_ScanNode to know how many slave nodes are on the CANopen network.

- **Syntax:**

  WORD CPM100_ScanNode(**BYTE** BoardNo, **BYTE** S_Node,
  BYTE E_Node, **BYTE** *NodeList, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.19. CPM100_GetNodeList

- **Description:**

  User can use the function CPM100_GetNodeList to know how many slave nodes are added to the node list of the CPM100.

- **Syntax:**

  WORD CPM100_GetNodeList(**BYTE** BoardNo, **BYTE** *NodeList,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  ***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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.20. CPM100_NMTChangeState

- **Description:**

  The function CPM100_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 CPM100_NMTChangeState(**BYTE** BoardNo, **BYTE** Node,
  BYTE State, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.21. CPM100_NMTGetState

- **Description:**

  The function CPM100_NMTGetState can get the NMT state from slaves.

- **Syntax:**

  WORD CPM100_NMTGetState(**BYTE** BoardNo, **BYTE** Node,
  BYTE *State, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.22. CPM100_NMTGuarding

- **Description:**

  Use the function CPM100_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 salve, the CPM100 will produce a Node_Guarding_Event event to users. Users may use the function CPM100_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 CPM100_NMGuarding(**BYTE** BoardNo, **BYTE** Node,
  WORD GuardTime, **BYTE** LifeTime, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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” and “Node” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.23. CPM100_NMTHeartbeat

- **Description:**

  Use the function CPM100_NMTHeartbeat to set heartbeat time of the slave with the specified node ID and consume time with the CPM100. 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 CPM100 will produce a “Heartbeat_Event” event to users. Users may use the function CPM100_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 CPM100_NMHartbeat(**BYTE** BoardNo, **BYTE** Node,
  WORD ProduceTime, **WORD** ConsumeTime,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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” and “Node” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.24. CPM100_SDOReadData

- **Description:**

  The function CPM100_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 then 4-byte data length) and segment mode (more then 4-byte data length).

- **Syntax:**

  WORD CPM100_SDOReadData (**BYTE** BoardNo, **BYTE** Node,
  WORD Index, **BYTE** SubIndex,
  DWORD *RDLen, **BYTE** *RData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.25. CPM100_SDOReadFile

- **Description:**

  The function CPM100_SDOReadFile is useful as uploading the SDO data more then 1024 bytes. While users use the CPM100_ReadData to read the SDO data and the return data length is more then 1024 byte. The SDO data are stored in a file. Users can use the function CPM100_SDOReadFile for reading the SDO data from the file.

- **Syntax:**

  WORD CPM100_SDOReadFile (**BYTE** BoardNo, **BYTE** Node,
  WORD Index, **BYTE** SubIndex,
  DWORD Start, **DWORD** Len,
  DWORD *RLen, **BYTE** *RData)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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.

4.5.26. CPM100_SDOWriteData

- **Description:**

  The function CPM100_SDOWriteData can send out a SDO message to the specified salve device. This procedure is also called download SDO protocol. The parameter node of the function CPM100_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 then 4-byte data length) and segment mode (more then 4-byte data length)

- **Syntax:**

  WORD CPM100_SDOWriteData (**BYTE** BoardNo, **BYTE** Node,
  WORD Index, **BYTE** SubIndex, **DWORD** TDLen,
  BYTE *TData, **WORD** *RDLen, **BYTE** *RData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.27. CPM100_SDOAbortTransmit

- **Description:**

  Call the function CPM100_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 salve device.

- **Syntax:**

  WORD CPM100_SDOAbortTransmit(**BYTE** BoardNo, **BYTE** Node,
  WORD Index, **BYTE** SubIndex, **DWORD** *AData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.28. CPM100_PDOWrite

- **Description:**

Call the function CPM100_PDOWrite to send out a PDO message to the specified slave device. Before using this function, users need to use the function CPM100_InstallPDO to install the PDO object into CPM100 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 CPM100_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 CPM100 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 CPM100_PDOWrite (**BYTE** BoardNo, **WORD** Cobid,
BYTE Offset, **BYTE** DLlen, **BYTE** *TData,
BYTE BlockMode)

- **Parameter:**

BoardNo: [input] CPM100 board number (0~7).
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 $\leqq$ 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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.29. CPM100_PDOWrite_Fast

- **Description:**

The function is like CPM100_PDOWrite but does not check whether the PDO message really sends to CAN bus or not. So CPM100_PDOWrite_Fast is about twice as faster than CPM100_PDOWrite at high speed baud rate (greater than or equal to 250kbps).

- **Syntax:**

WORD CPM100_PDOWrite_Fast (**BYTE** BoardNo, **WORD** Cobid,
BYTE Offset, **BYTE** DLlen, **BYTE** *TData)

- **Parameter:**

BoardNo: [input] CPM100 board number (0~7).
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.

4.5.30. CPM100_PDORemote

- **Description:**

  Use the function CPM100_PDORemote to send a RTR (remote-transmit-request) PDO message to the slave device.

- **Syntax:**

  WORD CPM100_PDORemote (**BYTE** BoardNo, **WORD** Cobid,
  BYTE *DLen, **BYTE** *TData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.31. CPM100_PDORemote_Fast

- **Description:**

  Use the function CPM100_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 CPM100_GetPDOLastData.

- **Syntax:**

  WORD CPM100_PDORemote_Fast (**BYTE** BoardNo, **WORD** Cobid)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  Cobid: [input] COB-ID used by the PDO object.

4.5.32. CPM100_SetPDORemotePolling

- **Description:**

  If the CANopen slaves do not support the event timer function of the TxPDOs, using the function CPM100_SetPDORemotePolling can config the most 125 TxPDO objects into the remote polling list. Then, the CPM100 will poll the configured TxPDOs and update the data into buffer automatically. Users can use CPM100_GetMultiPDOData to get these TxPDOs data from the buffer faster and easily.

- **Syntax:**

  WORD CPM100_SetPDORemotePoling (**BYTE** BoardNo,
  BYTE PDOCnt, **WORD** *Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  PDOCnt: [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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.33. CPM100_GetPDOLastData

- **Description:**

  Using the function CPM100_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 the same with the real situation.

- **Syntax:**

  WORD CPM100_GetPDOLastData (**BYTE** BoardNo, **WORD** Cobid,
  BYTE *IsNew, **BYTE** *DLen, **BYTE** *RData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.34. CPM100_GetPDOLastData_Fast

- **Description:**

  This function is like CPM100_GetPDOLastData. Although this function performance is better than CPM100_GetPDOLastData, but CPM100_GetPDOLastData_Fast can't check the getting data is new or not.

- **Syntax:**

  WORD CPM100_GetPDOLastData_Fast (**BYTE** BoardNo,
  WORD Cobid, **BYTE** *DLen,
  BYTE *RData)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  Cobid: [input] COB-ID used by the PDO object.
  ***DLen:** [output] The data length of the PDO message.
  ***RData:** [output] The PDO message gets from the PDO buffer.

4.5.35. CPM100_GetMultiPDOData

- **Description:**

  This can get the last data of the RxPDO and TxPDO from the PDO data buffer such as the function CPM100_GetPDOLastData. But the difference between these two functions is that user can use the function CPM100_GetMultiPDOData to get maximum 50 PDO data at the same time.

- **Syntax:**

  WORD CPM100_GetMuliPDOData (**BYTE** BoardNo, **BYTE** PDOCnt,
  WORD *Cobid, **BYTE** *IsNew,
  BYTE *DLen, **BYTE** *RData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  PDOCnt: [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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.36. CPM100_GetMultiPDOData_Fast

- **Description:**

  This function is like CPM100_GetMuliPDOData. Although this function performance is better than CPM100_GetMuliPDOData, but CPM100_GetMuliPDOData_Fast can't check the getting data is new or not.

- **Syntax:**

  WORD CPM100_GetMuliPDOData_Fast (**BYTE** BoardNo,
  BYTE PDOCnt, **WORD** *Cobid,
  BYTE *DLen, **BYTE** *RData)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  PDOCnt: [input] Total PDO number that want to get.
  ***Cobid:** [input] Maximum 50 COB-ID used by the PDO objects.
  ***DLen:** [output] The total data length obtained from the PDO buffer.
  ***RData:** [output] The PDO messages get from the PDO buffer.

4.5.37. CPM100_GetRxPDOID

- **Description:**

  Use the function CPM100_GetRxPDOID to get all the RxPDO COB-IDs of the specified slave, which have been installed to the master.

- **Syntax:**

  WORD CPM100_GetRxPDOID (**BYTE** BoardNo, **BYTE** Node,
  BYTE *PDO_Cnt, **WORD** *ID_List,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.38. CPM100_GetTxPDOID

- **Description:**

  Use the function CPM100_GetTxPDOID to get all the TxPDO COB-IDs of the specified slave, which have been installed to the master.

- **Syntax:**

  WORD CPM100_GetTxPDOID (**BYTE** BoardNo, **BYTE** Node,
  BYTE *PDO_Cnt, **WORD** *ID_List,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.39. CPM100_InstallPDO

- **Description:**

  After calling the CPM100_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 CPM100_AddNode with automatic mode. Otherwise, the TxPDOs or RxPDOs need to be installed manually by calling the function CPM100_InstallPDO.

- **Syntax:**

  WORD CPM100_InstallPDO(**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** RxTx,
  WORD PDO_No, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.40. CPM100_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 CPM100.

- **Syntax:**

  WORD CPM100_DynamicPDO(**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** RxTx, **BYTE** Entry,
  DWORD EntryData, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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 2nd 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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.41. CPM100_RemovePDO

- **Description:**

  The function CPM100_RemovePDO can remove a TxPDO or RxPDO object had installed by the CPM100_InstallPDO or CPM100_AddNode. This function also can remove single object mapped in TxPDO or RxPDO.

- **Syntax:**

  WORD CPM100_RemovePDO(**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** Entry,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.42. CPM100_ChangePDOID

- **Description:**

  Use the function CPM100_ChangePDOID to change the PDO COB-ID from old “Old_Cobid” to new “New_Cobid” of a slave device.

- **Syntax:**

  WORD CPM100_ChangePDOID (**BYTE** BoardNo, **WORD** Old_Cobid, **WORD** New_Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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” and “Old_Cobid” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.43. CPM100_GetPDOMapInfo

- **Description:**

  The function CPM100_GetPDOMapInfo can get the mapping data information of the PDO object.

- **Syntax:**

  WORD CPM100_GetPDOMapInfo (**BYTE** BoardNo, **WORD** Cobid,
  BYTE *RxTx, **BYTE** *Tx_Type,
  WORD *Event_Timer, **BYTE** *Entry_Cnt,
  DWORD *Map_Data, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.44. CPM100_InstallPDO_List

- **Description:**

  This function is similar with the CPM100_InstallPDO function. It can install the old or new PDO object in the PDO object list of the CPM100. It is the same as CPM100_InstallPDO. But the CPM100_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 CPM100. 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 CPM100 will process the slave PDOs which have the same IDs configured by the function CPM100_InstallPDO_List. It is very useful when the CPM100 is running in listen mode. User can use the function CPM100_RemovePDO_List to remove the PDO object which is installed by CPM100_InstallPDO_List.

- **Syntax:**

  WORD CPM100_InstallPDO_List(**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** RxTx, **WORD** PDO_No,
  BYTE EntryUse, **DWORD** *EntryData, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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 1st entry is 16-bit data of index 0x6411 and subindex 0x03 object and the 2nd 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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.45. CPM100_RemovePDO_List

- **Description:**

  The function CPM100_RemovePDO_List can remove a TxPDO or RxPDO object had installed by the CPM100_InstallPDO_List. This function also can remove single object mapped in the TxPDO or RxPDO.

- **Syntax:**

  WORD CPM100_RemovePDO_List(**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** Entry,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.46. CPM100_PDOUseEntry

- **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 CPM100_PDOUseEntry(**BYTE** BoardNo, **WORD** Cobid,
  BYTE Entry, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.47. CPM100_PDOTxType

- **Description:**

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

- **Syntax:**

  WORD CPM100_PDOTxType(**BYTE** BoardNo, **WORD** Cobid,
  BYTE Tx_Type, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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		X	X		
1-240	X		X		
241-251	**- reserved -**				
252			X		X
253				X	X
254				X	
255				X	

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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.48. CPM100_PDOEventTimer

- **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 CPM100_PDOEventTimer(**BYTE** BoardNo, **WORD** Cobid,
  WORD Timer, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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” and “Cobid” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.49. CPM100_PDOInhibitTime

- **Description:**

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

- **Syntax:**

  WORD CPM100_PDOInhibitTime(**BYTE** BoardNo, **WORD** Cobid,
  WORD Time, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.50. CPM100_ChangeSYNCID

- **Description:**

  Use the function CPM100_ChangeSYNCID to change the SYNC COB-ID of a slave device.

- **Syntax:**

  WORD CPM100_ChangeSYNCID (**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Cobid" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.51. CPM100_SetSYNC_List

- **Description:**

  If the user uses CPM100_AddNode function to add the slave with manual mode, the function CPM100_SetSYNC_List must be called while the SYNC ID of the slave needs to be changed or be set. The function CPM100_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 CPM100_SetSYNC_List. The users need to confirm that by themselves.

- **Syntax:**

  WORD CPM100_SetSYNC_List (**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.52. CPM100_GetSYNCID

- **Description:**

  This function can get the SYNC ID from the COB-ID list of the CPM100.

- **Syntax:**

  WORD CPM100_GetSYNCID (**BYTE** BoardNo, **BYTE** Node,
  WORD *Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.53. CPM100_SendSYNCMsg

- **Description:**

  Use the function CPM100_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 CPM100_SendSYNCMsg(**BYTE** BoardNo, **WORD** Cobid,
  WORD Timer, **DWORD** Times,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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” and “Cobid” again. If the procedure is still not complete, it will return “CPM_Wait”.

4.5.54. CPM100_GetCyclicSYNCInfo

- **Description:**

  This function can get at most 5 SYNC messages information which have been configured by the function CPM100_SendSYNCMsg. User can know what SYNC ID had been set.

- **Syntax:**

  WORD CPM100_GetCyclicSYNCInfo(**BYTE** BoardNo, **WORD** *Cobid,
  WORD *Timer, **DWORD** *Times,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  ***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" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.55. CPM100_ChangeEMCYID

- **Description:**

  Use the function CPM100_ChangeEMCYID to change the EMCY COB-ID of a specific slave device.

- **Syntax:**

  WORD CPM100_ChangeEMCYID (**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.56. CPM100_SetEMCY_List

- **Description:**

  If the user uses CPM100_AddNode function to add the slave with manual mode, the function CPM100_SetEMCY_List must be called while the EMCY ID of the slave needs to be changed or be set. CPM100_SetEMCY_List only can change the EMCY COB-ID in the COB-ID list of the CPM100. Afterwards, the CPM100 process the EMCY messages with the specific EMCY COB-ID which is configured by the function CPM100_SetEMCY_List.

- **Syntax:**

  WORD CPM100_SetEMCY_List (**BYTE** BoardNo, **BYTE** Node,
  WORD Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.57. CPM100_GetEMCYID

- **Description:**

  This function can get the EMCY ID from the COB-ID list of the CPM100.

- **Syntax:**

  WORD CPM100_GetEMCYID (**BYTE** BoardNo, **BYTE** Node,
  WORD *Cobid, **BYTE** BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.58. CPM100_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 CPM100_ReadLastEMCY (**BYTE** BoardNo, **BYTE** Node,
  BYTE *IsNew, **BYTE** *RData,
  BYTE BlockMode)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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" and "Node" again. If the procedure is still not complete, it will return "CPM_Wait".

4.5.59. CPM100_GetBootUpNodeAfterAdd

- **Description:**

  If users don't know which slave node occur the boot-up message and which CPM100 received it. Users can use the function CPM100_GetBootUpNodeAfterAdd. This function can get not only the slave node ID but also the slot number of the CPM100. The parameter, BoardNo, indicates the number of the CPM100 which receives the boot-up message. The parameter, Node, is the ID of the slave node which produces the boot-up message. The CPM100_GetBootUpNodeAfterAdd function is usually applied with the function CPM100_InstallBootUpISR. But note that, this function can only get the slave node ID that has already been added (CPM100_AddNode) to the CPM100.

- **Syntax:**

  WORD CPM100_GetBootUpNodeAfterAdd (**BYTE** *BoardNo,
  BYTE *Node)

- **Parameter:**

  BoardNo: [input] Get the board number of the CPM100 which receives the boot-up message.
  Node: [input] Get the slave node ID of the received boot-up message.

4.5.60. CPM100_GetEMCYData

- **Description:**

  If users don't know that which slave node occurred the EMCY message and which CPM100 received it. Users can use the function CPM100_GetEMCYData. This function can get not only the EMCY message with the slave node ID but also the board number of the CPM100. The parameter, BoardNo, indicates the number of the CPM100 which receives the EMCY message. The parameter, Node, is the ID of the slave node which produces the EMCY. The parameters, *RData, is the 8-bytes EMCY message data. The function CPM100_GetEMCYData is usually applied with the function CPM100_InstallEMCYISR. About the demo please refer to the section 4.1.2 NMT_Protocol.

- **Syntax:**

  WORD CPM100_GetEMCYData (**BYTE** *BoardNo, **BYTE** *Node,
  BYTE *RData)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the EMCY message.
  ***Node:** [output] Get the slave node ID of the received EMCY message.
  ***RData:** [output] 8-byte EMCY message obtained from the EMCY buffer.

4.5.61. CPM100_GetNMTError

- **Description:**

  User can use the function CPM100_GetNMTError to check if the CPM100 gets NMT Error Event for any slave node. The parameters of the function indicate that which CPM100 gets the NMT Error Event, which node produces this event, and what kind of event it is. The parameter, BoardNo, indicates the number of the CPM100 which 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 CPM100_GetNMTError is usually applied with the function CPM100_InstallNMTErrISR. About the demo please refer to the section 4.1.2 NMT_Protocol.

- **Syntax:**

  WORD CPM100_GetNMTErr (**BYTE** *BoardNo, **BYTE** *Node,
  BYTE *NMTErr)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the NMT Error Event.

  ***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.

4.5.62. CPM100_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 CPM100 is happen, the CPM100 will go into the boot-up process to do some specified mechanism which follows the user's boot-up process.

- **Syntax:**

  WORD CPM100_InstallBooUpISR (**BYTE** BoardNo,
  void (*BOOTISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetBootUpNodeAfterAdd.

4.5.63. CPM100_RemoveBootUpISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveBootUpISR (**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.64. CPM100_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 CPM100 will go into the EMCY process to do some security mechanism which follows the user's EMCY process.

- **Syntax:**

  WORD CPM100_InstallEMCYISR(**BYTE** BoardNo,
  void (*EMCYISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetEMCYData.

4.5.65. CPM100_RemoveEMCYISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveEMCYISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.66. CPM100_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 CPM100_NMTGuarding to enable the guarding protocol or had used the CPM100_Heartbeat to enable the heartbeat protocol, the CPM100 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 CPM100_InstallNMTErrISR(**BYTE** BoardNo,
  void (*NMTERRISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetNMTError.

4.5.67. CPM100_RemoveNMTErrISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveNMTErrISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.68. CPM100_GetMasterReadSDOEvent

- **Description:**

  Using this function can get all the read SDO messages sent to the specific node ID of the CPM100. For example, the CPM100 is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the CPM100 for reading an object, users can use the function CPM100_GetMasterReadSDOEvent for obtaining this SDO message, and respond some information to the SDO sender. The parameter, BoardNo, indicates the number of the CPM100 which receives the read SDO message. The parameters, Index and SubIndex, are the object indicator. The function CPM100_GetMasterReadSDOEvent is usually applied with the function CPM100_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 CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_GetMasterReadSDOEvent (**BYTE** *BoardNo,
  WORD *Index, **BYTE** *SubIndex)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the read SDO message.

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

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

4.5.69. CPM100_GetMasterWriteSDOEvent

- **Description:**

  Using this function can get all the write SDO messages sent to the specific node ID of the CPM100. For example, the CPM100 is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the CPM100 for writing an object, users can use the function CPM100_GetMasterWriteSDOEvent for obtaining this SDO message. The parameter, BoardNo, indicates the number of the CPM100 which 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 CPM100_GetMasterWriteSDOEvent is usually applied with the function CPM100_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 CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_GetMasterWriteSDOEvent (**BYTE** *BoardNo,
  WORD *Index, **BYTE** *SubIndex,
  BYTE *WLen, **BYTE** *WData)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the write SDO message.

  ***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.

4.5.70. CPM100_ResponseMasterSDO

- **Description:**

  Using this function can reply the SDO messages to the SDO sender. For example, the CPM100 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 CPM100, the CPM100 need to reply the corresponding SDO message, use the function CPM100_ResponseMasterSDO to do it. When users implement the function CPM100_ResponseMasterSDO, the CPM100 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 CPM100_InitMaster is > 0.

  Note2: If the CPM100 want to reply a SDO Abort message, please use the function CPM100_SDOAbortTransmit (section 3.5.20) to do it.

- **Syntax:**

  WORD CPM100_ResponseMasterSDO (**BYTE** BoardNo,
  BYTE ResType, **WORD** Index, **BYTE** SubIndex,
  BYTE Len, **BYTE** *Data)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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.

4.5.71. CPM100_InstallReadSDOISR

- **Description:**

  This function allows the user to apply the ReadSDO IST (interrupt service thread) of CPM100. When the user puts his read SDO process into this function, all the read SDO messages sent to the specified CPM100 will trigger the IST. For example, the CPM100 is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 for reading the object of the CPM100, the CPM100 will go into the IST if the user has installed it.

  Note: The function is valid while the Node parameter of the function CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_InstallReadSDOISR(**BYTE** BoardNo,
  void (*RSDOISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetMasterReadSDOEvent.

4.5.72. CPM100_RemoveReadSDOISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveReadSDOISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.73. CPM100_InstallWriteSDOISR

- **Description:**

  This function allows the user to apply the WriteSDO IST (interrupt service routine) of the CPM100. When the user puts the process into this function, all the written SDO messages sent to the specified CPM100 will trigger the IST. For example, the CPM100 is initialized with node ID 2. If someone sends an SDO message with the COB-ID 0x602 to the CPM100 for writing an object, the CPM100 will go into the IST if the user has installed it.

  Note: The function is valid while the Node parameter of the function CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_InstallWriteSDOISR(**BYTE** BoardNo,
  void (*WSDOISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetMasterWriteSDOEvent.

4.5.74. CPM100_RemoveWriteSDOISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveWriteSDOISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.75. CPM100_GetMasterRemotePDOEvent

- **Description:**

  Using this function can get all the Remote PDO messages sent to the CPM100. For example, the CPM100 has used the function CPM100_InstallPDO_List to install an CPM100's TxPDO object with the COB-ID 0x444. If someone sends a Remote PDO message with the COB-ID 0x444 to the CPM100, users can use the function CPM100_GetMasterRemotePDOEvent to get this PDO message. The parameter, BoardNo, indicates the number of the CPM100 which receives the Remote PDO message. The parameter, Cobid, is the PDO COB-ID sent to the CPM100. This function is usually used with the function CPM100_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 CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_GetMasterRemotePDOEvent (**BYTE** *BoardNo,
  WORD *CobId)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the Remote PDO message.

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

4.5.76. CPM100_GetMasterRxPDOEvent

- **Description:**

  Using this function can get all the RxPDO messages sent to the CPM100. For example, the CPM100 has used the function CPM100_InstallPDO_List to install an CPM100's RxPDO object with the COB-ID 0x333. If someone sends an RxPDO message with the COB-ID 0x333 to the CPM100, users can use this function to get this RxPDO message. The parameter, BoardNo, indicates the number of the CPM100 which 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 CPM100_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 CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_GetMasterRxPDOEvent (**BYTE** *BoardNo,
  WORD *CobId, **BYTE** *WLen, **BYTE** *WData)

- **Parameter:**

  ***BoardNo:** [output] Get the board number of the CPM100 which receives the RxPDO message.

  ***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.

4.5.77. CPM100_ResponseMasterPDO

- **Description:**

  Using this function can reply the Remote PDO messages to the sender. For example, the CPM100 has used CPM100_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 CPM100, and the CPM100 needs to reply a TxPDO message, users can use this function to do it. When users implement the function CPM100_ResponseMasterPDO, the CPM100 will send a TxPDO message to the CANopen network. This function is usually used with the CPM100_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 CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_ResponseMasterPDO (**BYTE** BoardNo,
  WORD CobId, **BYTE** Len, **BYTE** *Data)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).
  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.

4.5.78. CPM100_InstallRxPDOISR

- **Description:**

  This function allows the user to apply the RxPDO IST (interrupt service routine) of the CPM100. When the user puts his process into this function, all the RxPDO messages with the CPM100's PDO objects will trigger the IST. For example, the CPM100 has used CPM100_InstallPDO_List to install a PDO object with the COB-ID 0x333 of the CPM100. If some one sends a PDO message with the COB-ID 0x333 to the CPM100, the CPM100 will go into the IST if the user had installed it.

  Note: The function will usefully when the Node parameter of the function CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_InstallRxPDOISR(**BYTE** BoardNo,
  void (*RXPDOISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetMasterRxPDOEvent.

4.5.79. CPM100_RemoveRxPDOISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveRxPDOISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

4.5.80. CPM100_InstallRemotePDOISR

- **Description:**

  This function allows the user to apply the RemotePDO IST (interrupt service routine) of the CPM100. When the user puts his process into this function, all the Remote PDO messages of the CPM100's PDO objects will trigger the IST. For example, the CPM100 has used CPM100_InstallPDO_List to install a TxPDO object with the COB-ID 0x444 of the CPM100. If some one sends a Remote PDO message with the COB-ID 0x444 to the CPM100, the CPM100 will go into the IST if the user had installed it.

  Note: The function will usefully when the Node parameter of the function CPM100_InitMaster is > 0.

- **Syntax:**

  WORD CPM100_InstallRemotePDOISR(**BYTE** BoardNo,
  void (*REMOTEPDOISR)( ))

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

  (*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 CPM100_GetMasterRemotePDOEvent.

4.5.81. CPM100_RemoveRemotePDOISR

- **Description:**

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

- **Syntax:**

  WORD CPM100_RemoveRemotePDOISR(**BYTE** BoardNo)

- **Parameter:**

  BoardNo: [input] CPM100 board number (0~7).

5. Demo Programs

The CPM100 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. There is also a tool, CPMUtility, to control/monitor CANopen slaves with CPM100 easily and quickly. Users can find these demos and utility tool in the fieldbus CD or on the web site.

The path of CAN CD

fieldbus_cd://canopen/master/piso-cpm100

The address of the web site is

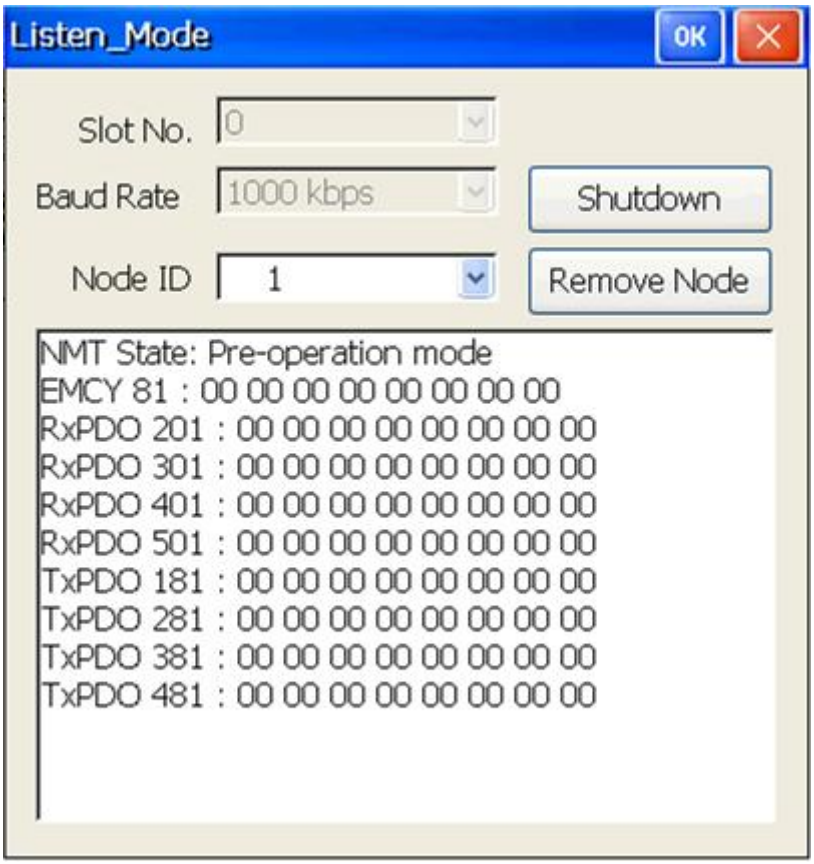
http://www.icpdas.com/products/Remote_IO/can_bus/piso-cpm100.htm

5.1. 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.

5.1.1. Listen_Mode

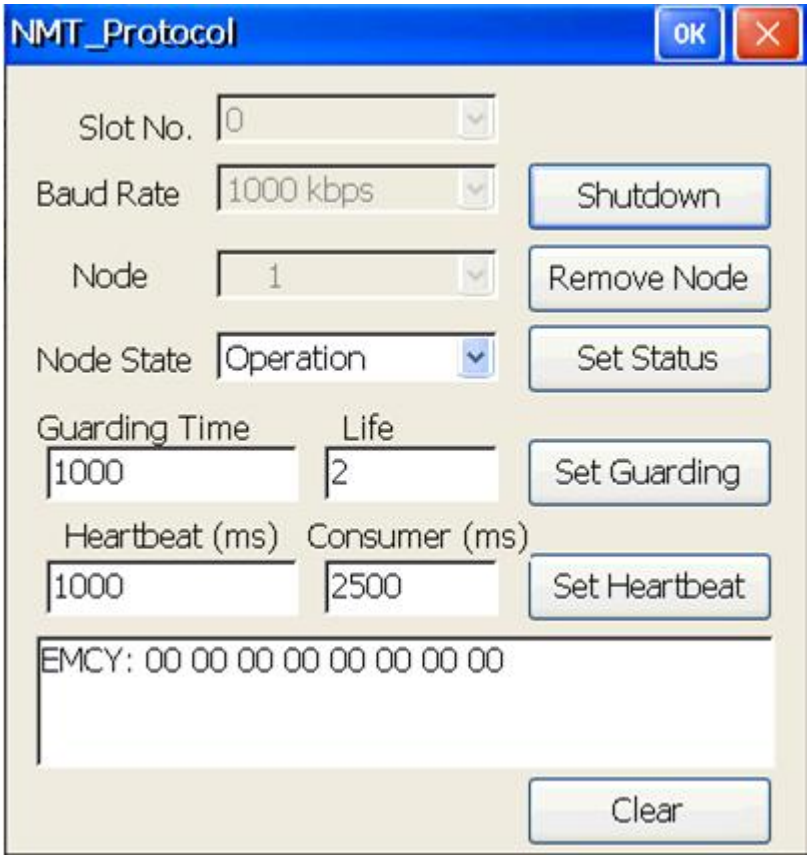
Initialize the CPM100 with the “listen mode” and add slave nodes with the “manual mode” or EDS file, then the CPM100 listens CANopen messages only and does not send any message to the CANopen network. In this demo, the CPM100 will listen NMT state, 4 TxPDO messages (with the COB ID 0x180+Node ID, 0x280+Node 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:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_EDS_Load, CPM100_SetMasterMode, CPM100_NMTGetState, CPM100_InstallPDO_List, CPM100_GetPDOLastData, CPM100_GetRxPDOID, CPM100_GetTxPDOID, CPM100_SetEMCY_List, CPM100_GetEMCYID, CPM100_ReadLastEMCY.

5.1.2. NMT_Protocol

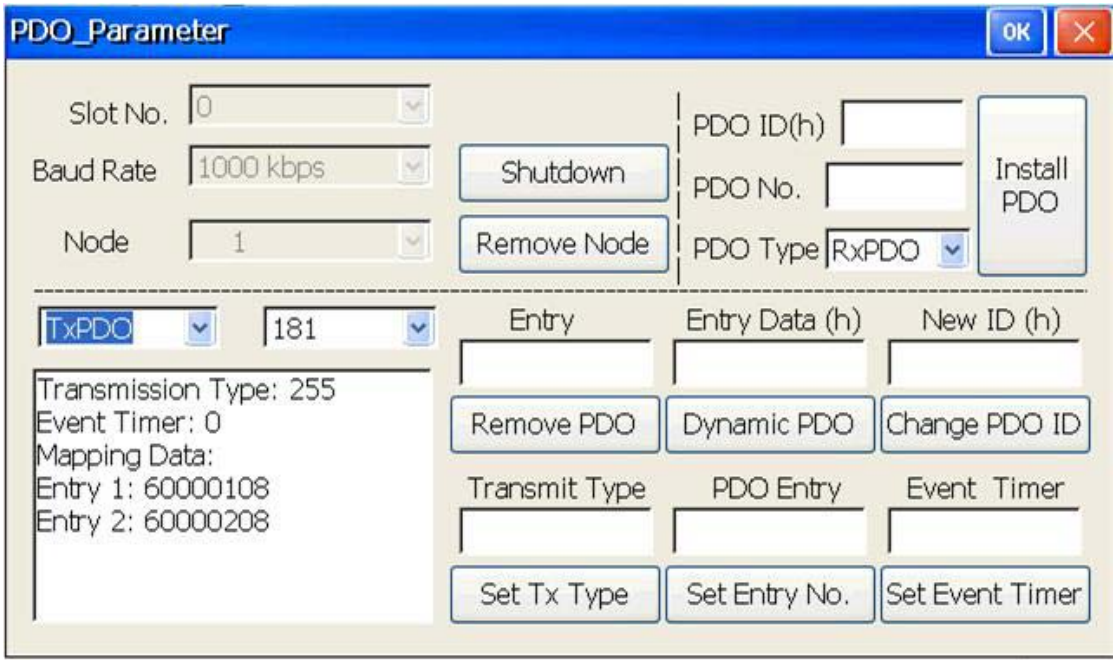
This is a NMT network control demo. The demo not only tells users how to control the NMT status of a specific slave node, but also how to protect the slave through the “Guarding” and “Heartbeat” functions.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_NMTChangeState, CPM100_NMTGuarding, CPM100_NMTHeartbeat, CPM100_GetEMCYData, CPM100_GetNMTError, CPM100_InstallNMTErrISR, CPM100_RemoveNMTErrISR, CPM100_InstallEMCYISR, CPM100_RemoveEMCYISR.

5.1.3. 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.

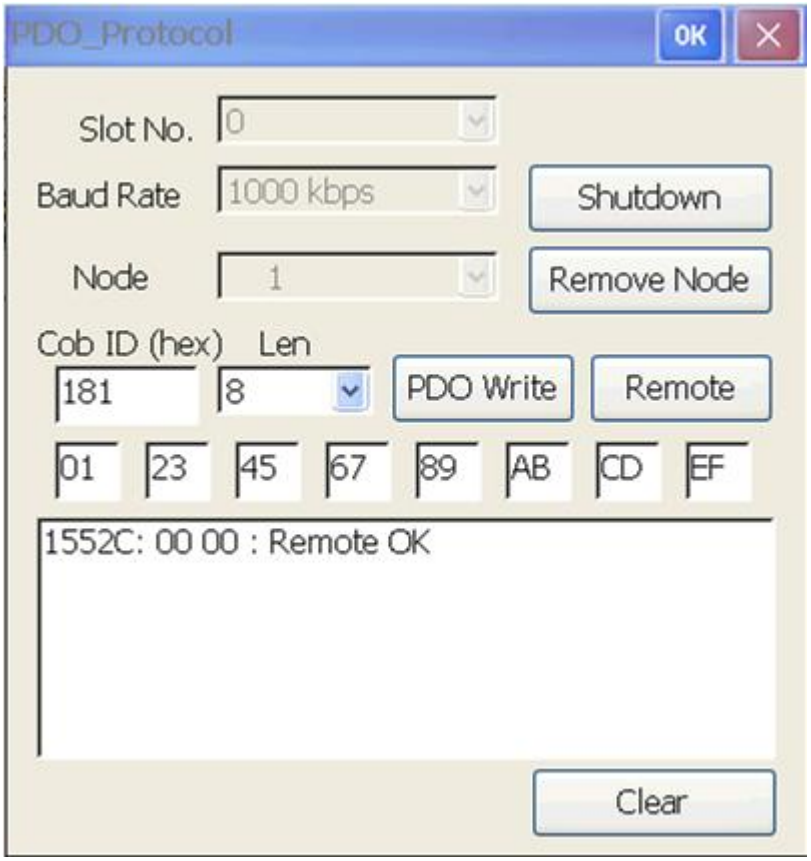


Applied function list:

CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_InstallPDO, CPM100_RemovePDO, CPM100_DynamicPDO, CPM100_ChangePDOID, CPM100_PDOTxType, CPM100_PDOUseEntry, CPM100_PDOEventTimer, CPM100_GetTxPDOID, CPM100_GetRxPDOID, CPM100_GetPDOMapInfo.

5.1.4. 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.

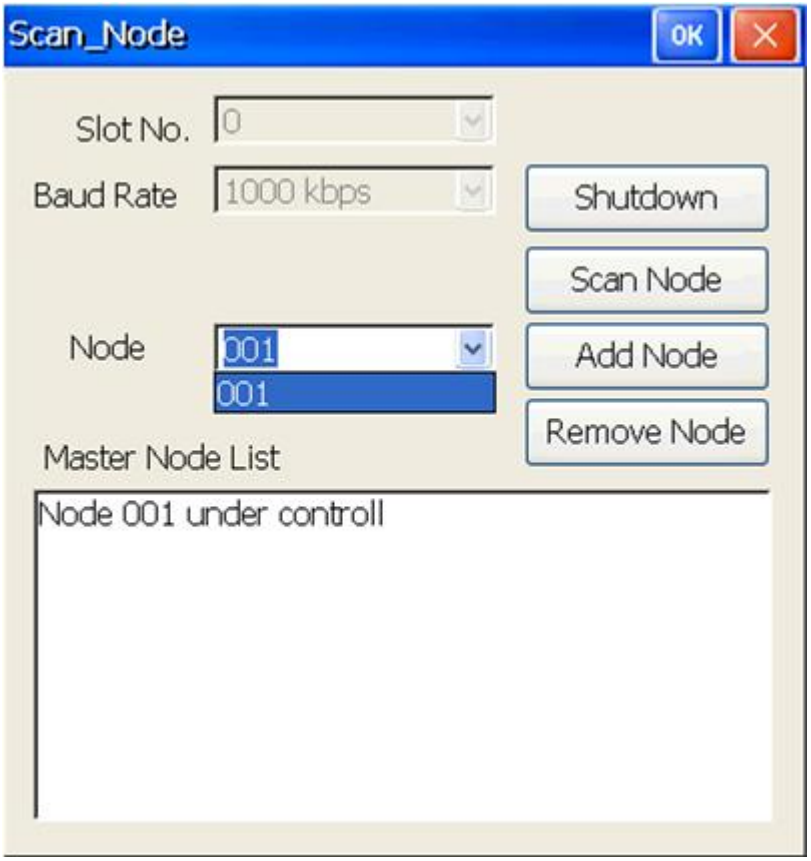


Applied function list:

CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_PDOWrite, CPM100_PDORemote, CPM100_GetPDOLastData

5.1.5. Scan_Node

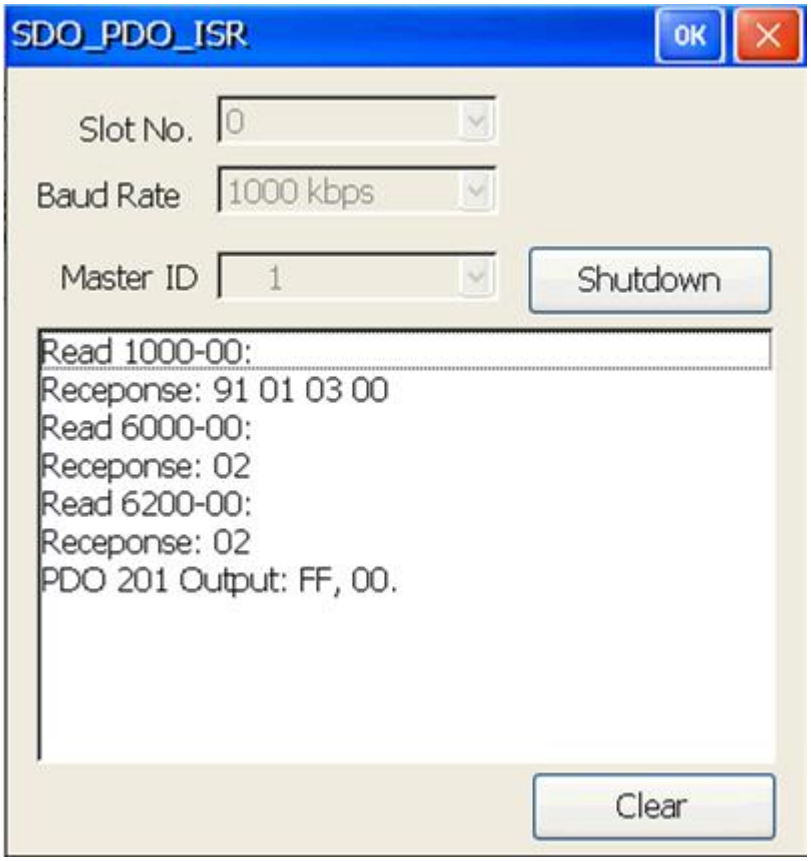
When users want to know which slave nodes exist on the CANopen network or which slave nodes are under the control of the CPM100, this demo will is useful.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_SetFunctionTimeout, CPM100_ScanNode, CPM100_GetNodeList

5.1.6. SDO_PDO_ISR

In this demo, it is allowed to configure the CPM100 as a CANopen slave. Users can use another CANopen master to read/write the users' defined object dictionary of the CPM100 by SDO protocol or to get/set the DIO status by PDO protocol when the CPM100, 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.

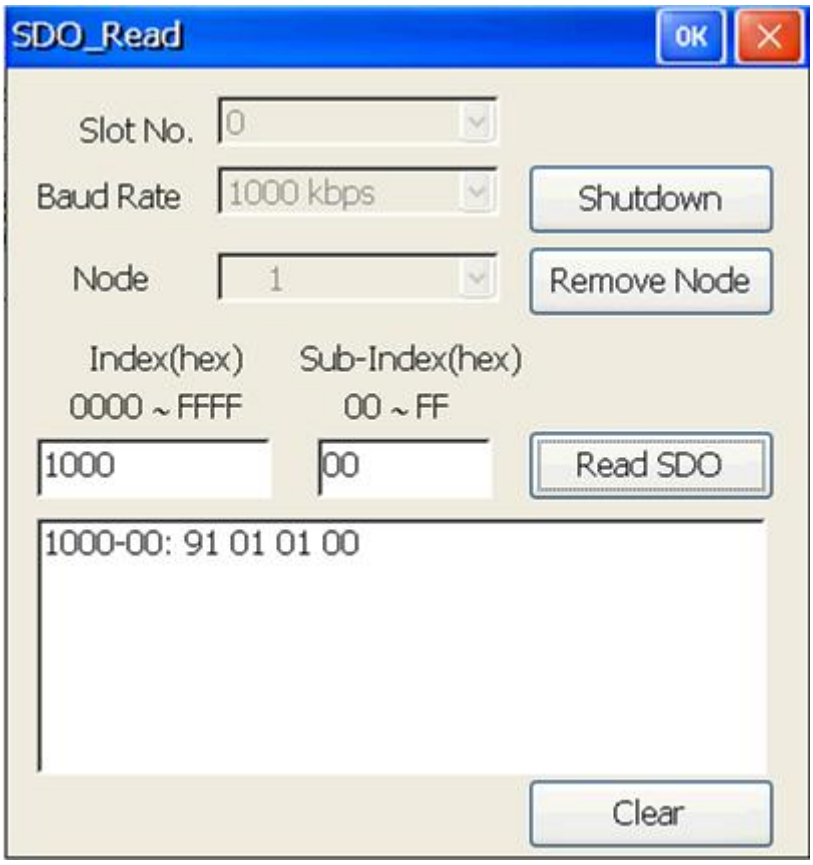


Applied function list:

CPM100_InitMaster, CPM100_Shutdown, CPM100_GetMasterReadSDOEvent, CPM100_GetMasterWriteSDOEvent, CPM100_GetMasterRemotePDOEvent, CPM100_GetMasterRxPDOEvent, CPM100_ResponseMasterSDO, CPM100_ResponseMasterPDO, CPM100_InstallPDO_List, CPM100_InstallReadSDOISR, CPM100_InstallWriteSDOISR, CPM100_InstallRxPDOISR, CPM100_InstallRemotePDOISR.

5.1.7. SDO_Read

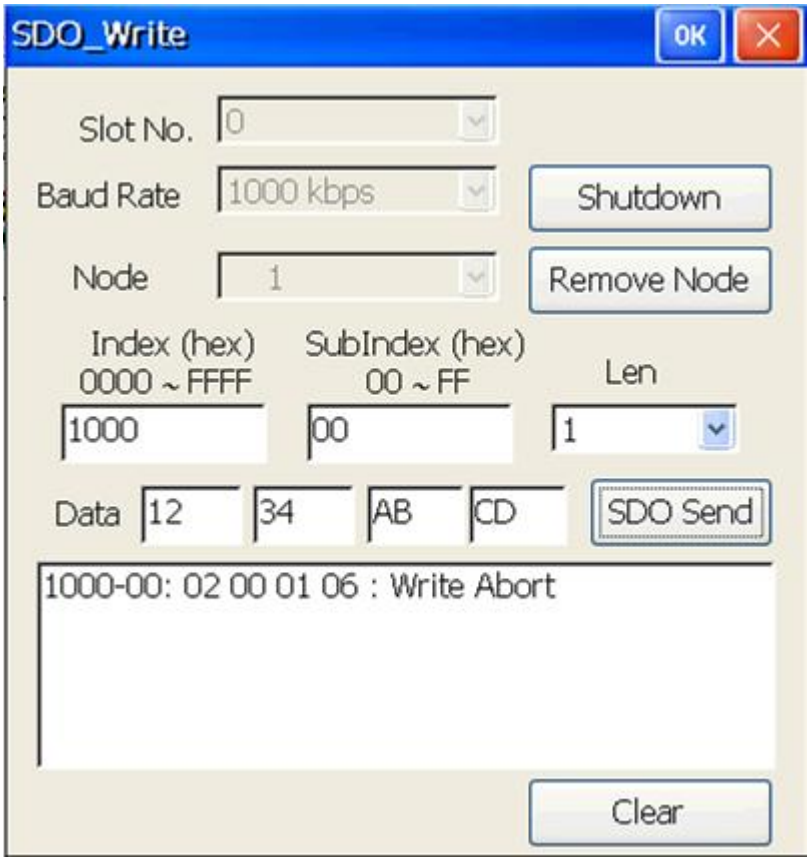
SDO protocol is a kind of the communication functions used to read/write CANopen object dictionary. You can read any object data of the object dictionary through the object address (index and sub-index) by SDO protocol. This demo is a good model to do that.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_SDOReadData.

5.1.8. SDO_Write

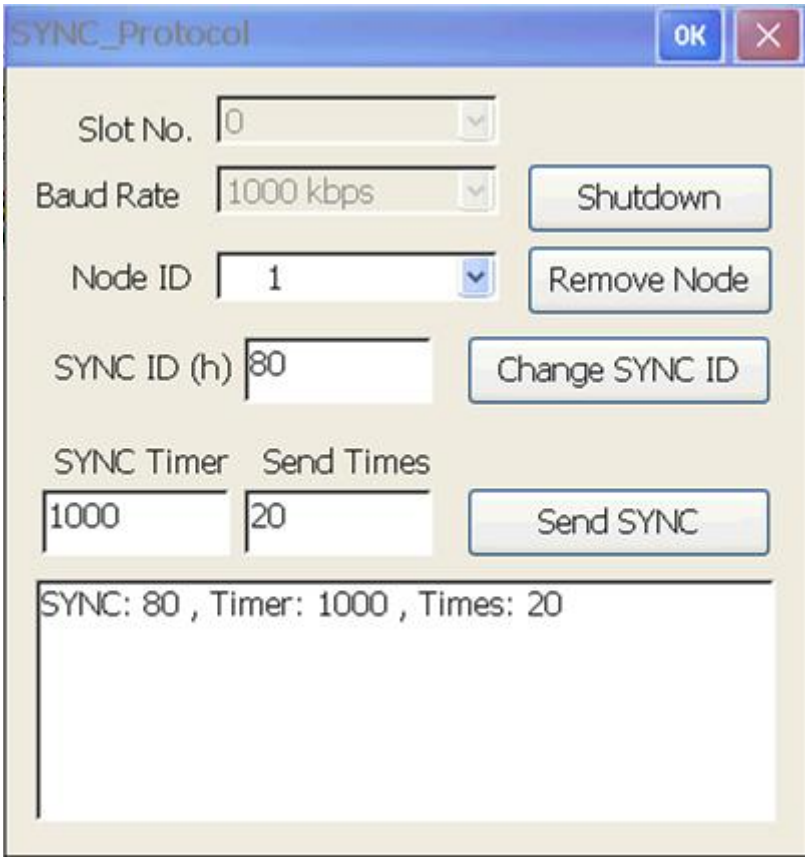
SDO protocol is a kind of the communication functions used to read/write CANopen object dictionary. You can write any data to the specific object of the object dictionary through the object address (index and sub-index) by SDO protocol. This demo is a good model to do that.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_SDOWriteData.

5.1.9. SYNC_Protocol

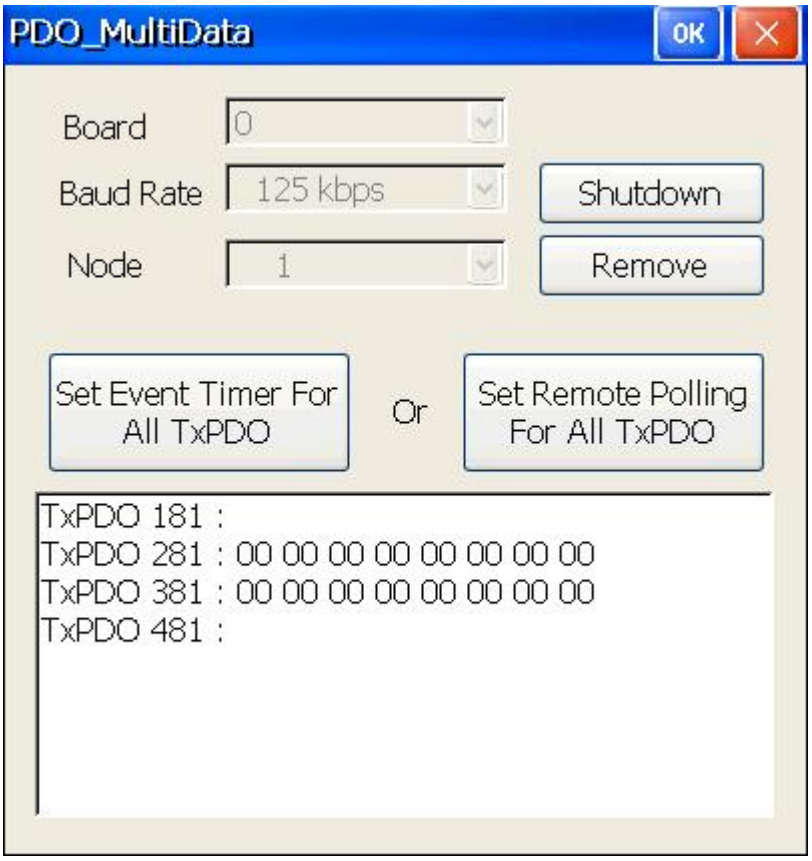
SYNC protocol is a synchronous function of the PDO communication. It is always used with the transmission type of the PDO communication. In this demo, users can know how to use the SYNC related functions.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_ChangeSYNCID, CPM100_GetSYNCID, CPM100_SendSYNCMsg, CPM100_GetCyclicSYNCInfo.

5.1.10. PDO_MultiData

Sometimes, users want to poll several PDO objects data at the same time for increasing the performance. But it is slower that 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 CPM100 or are replied from slave automatically, then use the CPM100_GetMultiPDOData function to obtain these PDO data from the buffer at the same time.



Applied function list:
CPM100_InitMaster, CPM100_Shutdown, CPM100_AddNode, CPM100_RemoveNode, CPM100_GetTxPDOID, CPM100_SetPDORemotePolling, CPM100_PDOEventTimer, CPM100_GetMultiPDOData

6. Update Firmware

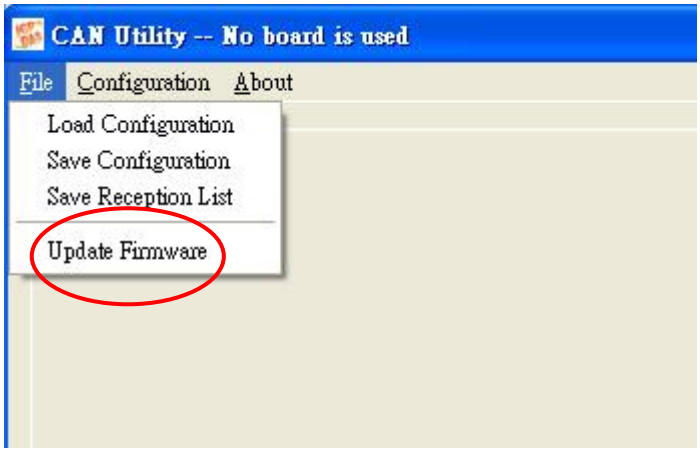
If users want to update the firmware of the CPM100, use the CANUtility tool to do it.

*** Where to find the CANUtility tool?**

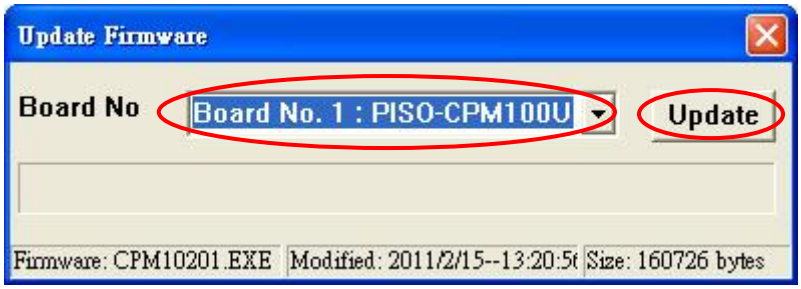
The path of field bus CD:
fieldbus_cd://can/pci/piso-cm100u/tools/2000_xp/

The address of the web site:
http://ftp.icpdas.com/pub/cd/fieldbus_cd/can/pci/piso-cm100u/tools/2000_xp/

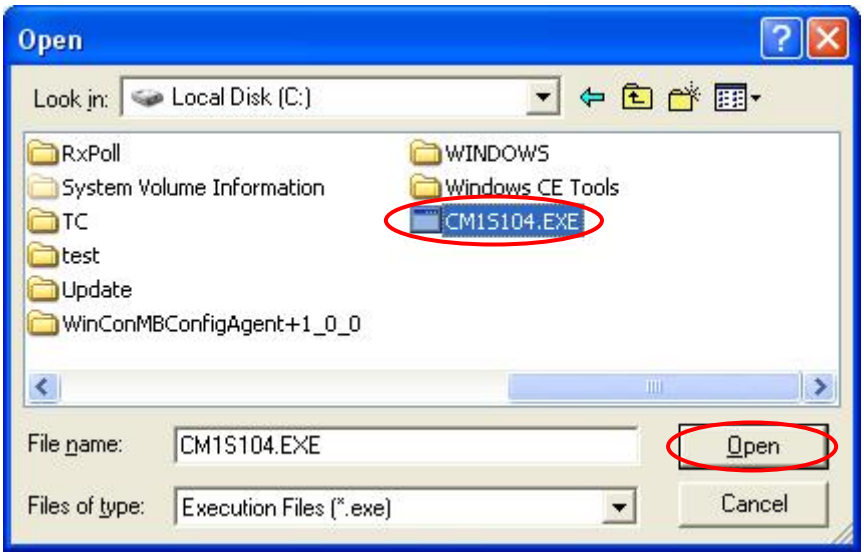
The following steps describe the method about how to update the CPM100 firmware by CANUtility tool. After executes the CANUtility, click the "File→Update Firmware" on the toolbar below.



When users apply the Update Firmware function, select the specified board firstly. Then, click Update button to select the proper firmware for the specified board.



Only .exe file can be downloaded into CPM100.



When finishing the download procedure, the Download OK dialog is popped up. Click OK button to continue.