I-87123 CANopen Master Module

User's Manual

Warranty

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

Warning

ICP DAS assume no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

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 serial communication protocol, 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 connections between simple industrial devices (sensors, actuators) and higher-level devices (controllers), as shown in Figure 1.1.

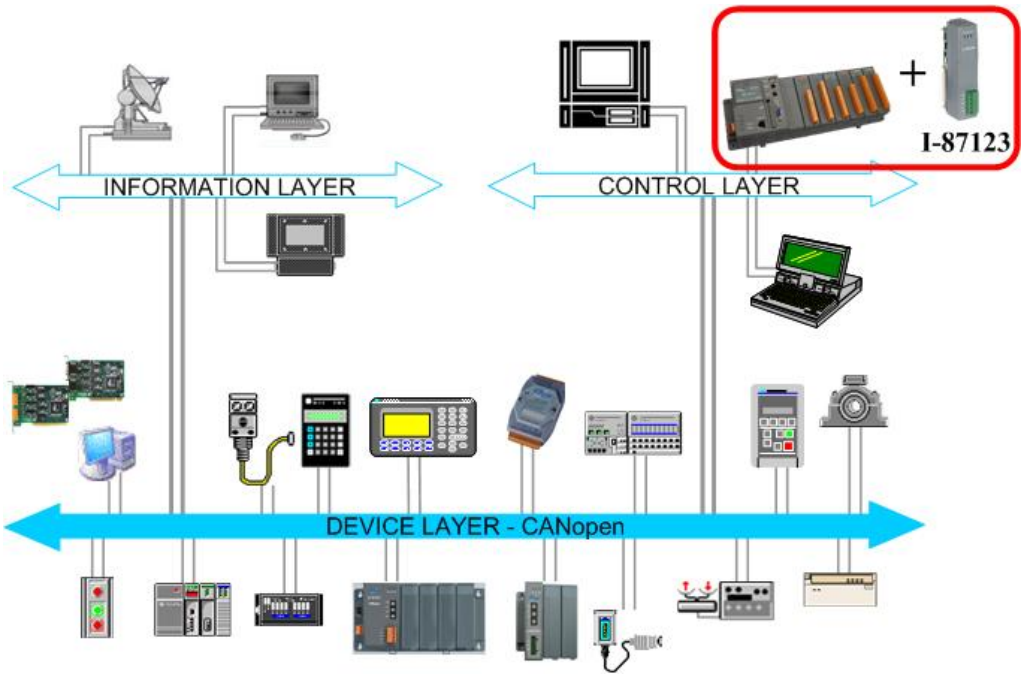


Figure 1.1 Example of the CANopen network

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

1.2. CANopen Applications

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

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

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

1.3. I-87123 Library Characteristics

In order to use the I-87123 module, we provide I-87123 library for I-8000, WinCon, WinPAC, and LinPAC main control unit, and users can use it establish CANopen communication network rapidly. Most of the CANopen communication protocols, such as PDO, SDO and NMT, will be handled by the library function 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 I-87123.

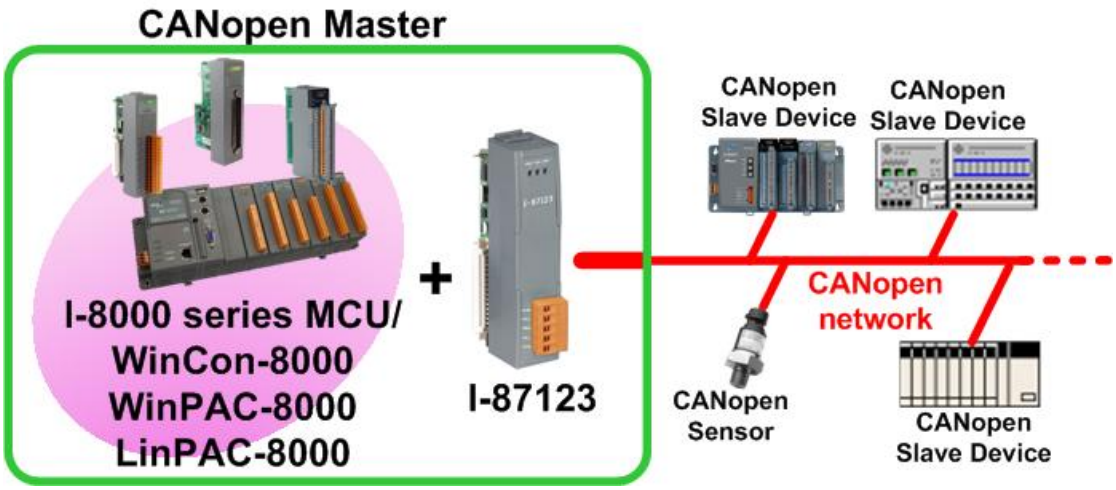


Figure 1.2 Application architecture

I-87123 follows the CiA CANopen specification DS-301 V4.01, and supports the several CANopen features. The CANopen communication general concept is shown as Figure 1.3.

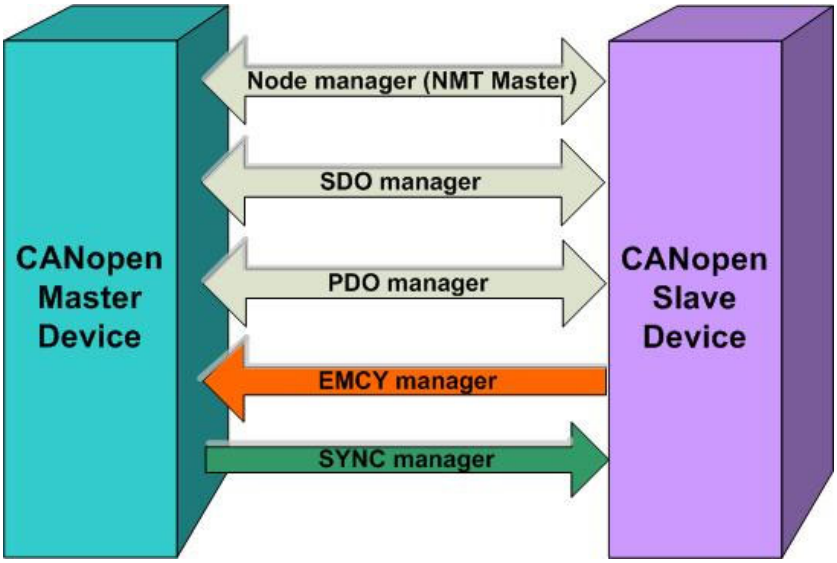


Figure 1.3 CANopen communication general concept

- **Node Manager (NMT Master)**
  - Functions for changing the slave device state
  - Node Guarding 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 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

- CPU:80186, 80MHz
- NXP SJA1000 CAN controller with 16MHz clock
- NXP 82C250 CAN transceiver
- Power LED, Tx/Rx LED, Error LED
- 120Ω terminal resister selected by jumper
- CAN bus interface: ISO 11898-2, 5-pin screw terminal with on-board optical isolator protection
- 2500 Vrms isolation on CAN side
- 1000 Vdc voltage protection
- Power Consumption: 2W
- Operating Temperature: -25℃ to +75℃
- Storage Temperature: -30℃ to +85℃
- Humidity: 5% ~ 95%

Features

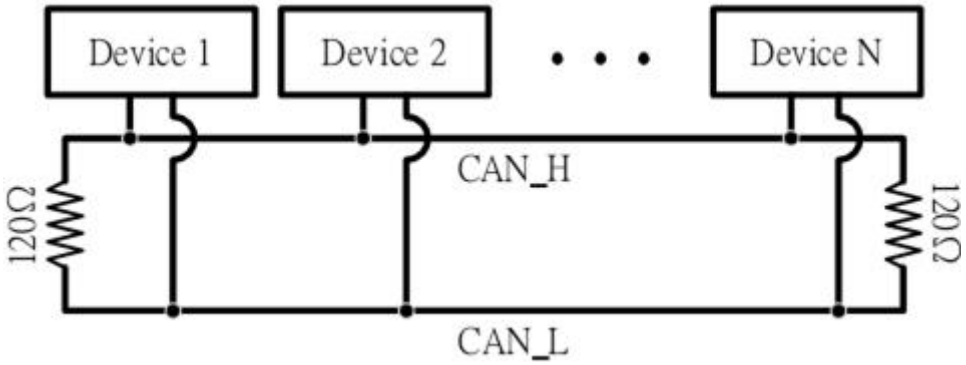
- One CAN port expansion for I-8000 series MCU (main control unit), WinCon-8000, WinPAC-8000, and LinPAC-8000.
- 256 records CANopen message receive buffer size
- Provide C/C++ function libraries to process CANopen message
- Three indication LEDs (Pwr, Tx/Rx and Err LEDs)
- Support 8 kinds baud: 10Kbps, 20Kbps, 50Kbps, 125Kbps, 250Kbps, 500Kbps, 800Kbps, and 1Mbps
- Each Port support maximum nodes up to 127
- Follow CiA DS-301 V4.01
- Support upload and download SDO Segment
- Support Node Guarding protocol
- Demos and utility are provided

2. Hardware Specification

2.1. Hardware Structure

2.2. Wire Connection

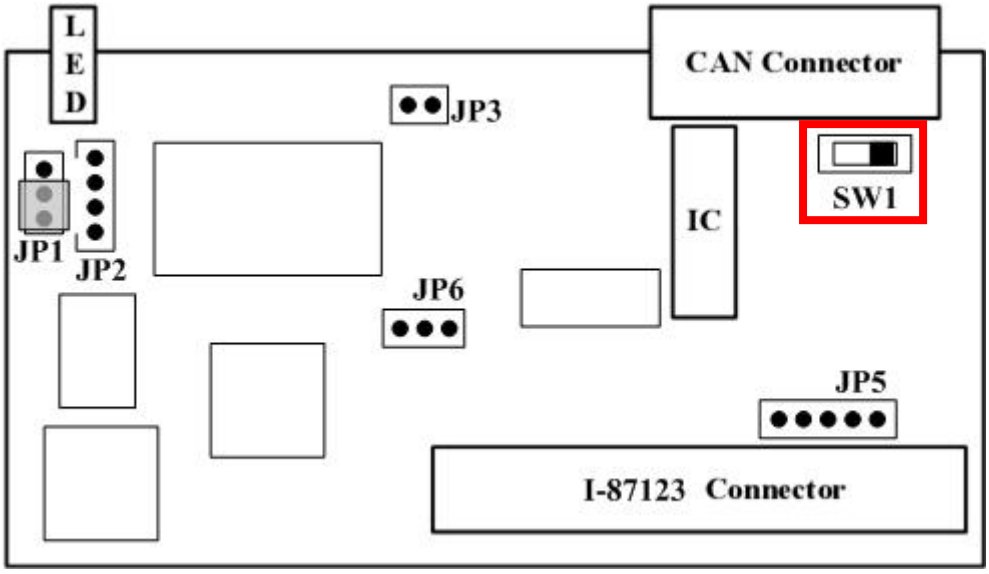
In order to minimize the reflection effects on the CAN bus line, the CAN bus line has to be terminated at both ends by two terminal resistances as in the following figure. According to the ISO 11898-2 spec, each terminal resistance is 120Ω(or between 108Ω~132Ω). The length related resistance should have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



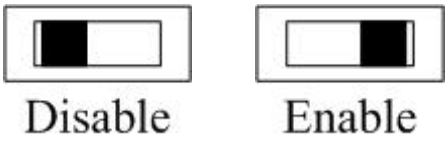
Moreover, to minimize the voltage drop over long distances, the terminal resistance should be higher than the value defined in the ISO 11898-2. The following table can be used as a good reference.

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

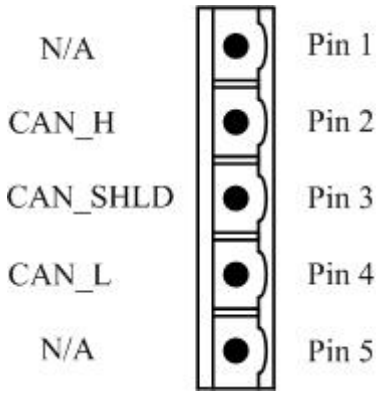
In the I-87123, the 120Ω terminal resistance is supplied. The SW1 of the I-87123 is for the terminal resistance. Its location is shown in the following figure.



The following connection statuses are presented for the condition if the terminal resister is enabled or disabled.



SW1 Switch selection.



Pin assignment of I-87123 connector

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

2.3. Power LED

I-87123 slot module needs 2W power consumption. If the electric power is supplied normally, the Power LED will be turn on always. If any other situation, please check the power supply or contact to your distributor.

2.4. Tx/Rx LED

Each I-87123 slot module provides Tx/Rx LED to check the CAN messages transmission and reception situation. If the I-87123 is transmitting or receiving a CAN message, the Tx/Rx LED will blink. If I-87123's loading is heavy, the Tx/Rx LED will always turn on.

2.5. ERR LED

The ERR LED indicates the error status of the CAN physical layer and indicates the errors due to missing CAN message.

3. I-87123 Function Library

3.1. Function List

In order to use the I-87123 more easily, we provide some useful and easy-to-use functions in I-87123 library. There are two function libraries for different compiler, such as BC/TC (I-8000 series main control unit), GCC(LinPAC-8000), and EVC (WinCon-8000, WinPAC-8000). Users can use these functions to control the I-87123 by using the functions. The following table shows the all functions provided by the I-87123 library.

Function Name	Description
I87123_GetVersion	Get the version of the I-87123 library
I87123_Configure	Activate the I-87123 module
I87123_ShutdownMaster	Remove all nodes and stop master
I87123_AddNode	Add a node into I-87123 master manager
I87123_RemoveNode	Remove a node from I-87123 master manager
I87123_ChangeState	Change the CANopen node state
I87123_GetState	Get CANopen node state
I87123_Guarding	Start to the node guarding function
I87123_SendSYNC	Send SYNC message
I87123_ChangeEMCYID	Change EMCY COB ID
I87123_ChangeSYNCID	Change SYNC COB ID
I87123_AbortSDO	Send SDO abort message
I87123_ReadSDO	Read data by upload SDO protocol
I87123_WriteSDO	Write data by download SDO protocol
I87123_SetPDOResponse	Set the PDO data response mechanisms
I87123_MsgResponse	Receive message that is responded automatically
I87123_InstallPDO	Install and enable a specific PDO
I87123_RemovePDO	Remove a specific PDO object
I87123_WritePDO	Use PDO to write data to CANopen node
I87123_RemotePDO	Use PDO to get data from CANopen node
I87123_PDOTxType	Set transmission type of TxPDO
I87123_WriteDO	Output 8 bits DO value
I87123_ReadDI	Read 8 bits DI value
I87123_WriteAO	Output one AO channel
I87123_ReadAI	Read one AI channel

Table 3.1 Description of functions

3.2. Function Return Code

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

Return Code	Error ID	Description
0	I87123_OK	OK
1	I87123_COMMAND_ERR	Command not support
2	I87123_CONFIG_ERR	I-87123 hasn't been configured successfully
3	I87123_DATALEN_ERR	Data length is erroneous
4	I87123_ADDNODE_ERR	Add a CANopen node failure
5	I87123_REMOVENODE_ERR	Remove a CANopen node failure
6	I87123_STATUS_ERR	CANopen slave NMT status error
7	I87123_SETGUARD_ERR	Set guard of CANopen slave node failure
8	I87123_GUARD_FAILED	CANopen slave node guarding failure
9	I87123_NODENUMBER_ERR	Set node number error
10	I87123_COBID_ERR	Set COB-ID error
11	I87123_SDO_WRITEDATA_ERR	Write data in the index-subindex object failure
12	I87123_SDO_SEND_ERR	Send SDO expedition message error
13	I87123_SDO_SEGMENT_ERR	Send SDO segment message error
15	I87123_PDO_SEND_ERR	Send PDO message error
16	I87123_PDO_TYPE_ERR	TxPDO or RxPDO type error
17	I87123_RESPONSE_ERR	Response message error
18	I87123_PDO_CHANNEL_ERR	Set subindex content of PDO error
19	I87123_PDO_INSTALL_ERR	Install or modify PDO content error
21	I87123_TIMEOUT	Message response timeout
25	I87123_PARATERS_ERR	Set function parameter failure
27	I87123_CHANNEL_ERR	This I/O channel isn't exist
30	I87123_SYNC_SEND_ERR	Send SYNC message error
40	I87123_SDO_DATA_LOSE	SDO buffer overflow
45	I87123_EMCY	Receive a EMCY message
49	I87123_EMPTY	Soft buffer is empty

Table 3.2 Description of return code

3.3. CANopen Master Library Application Flowchart

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

When users programs apply the CANopen Master Library functions, the function I87123_Configure must be call first. The function is used to initialize I-87123 and configure the CAN port.

After initializing the CAN interface successfully, users need to use the function I87123_AddNode to install at least one CANopen device into the node list.

If the function I87123_Configure and I87123_AddNode has been executed, the communication services (NMT, SYNC, EMCY, SDO, and PDO services) can be used at any time before calling the function I87123_ShutdownMaster, because the function I87123_ShutdownMaster will stop all process created by the function I87123_Configure.

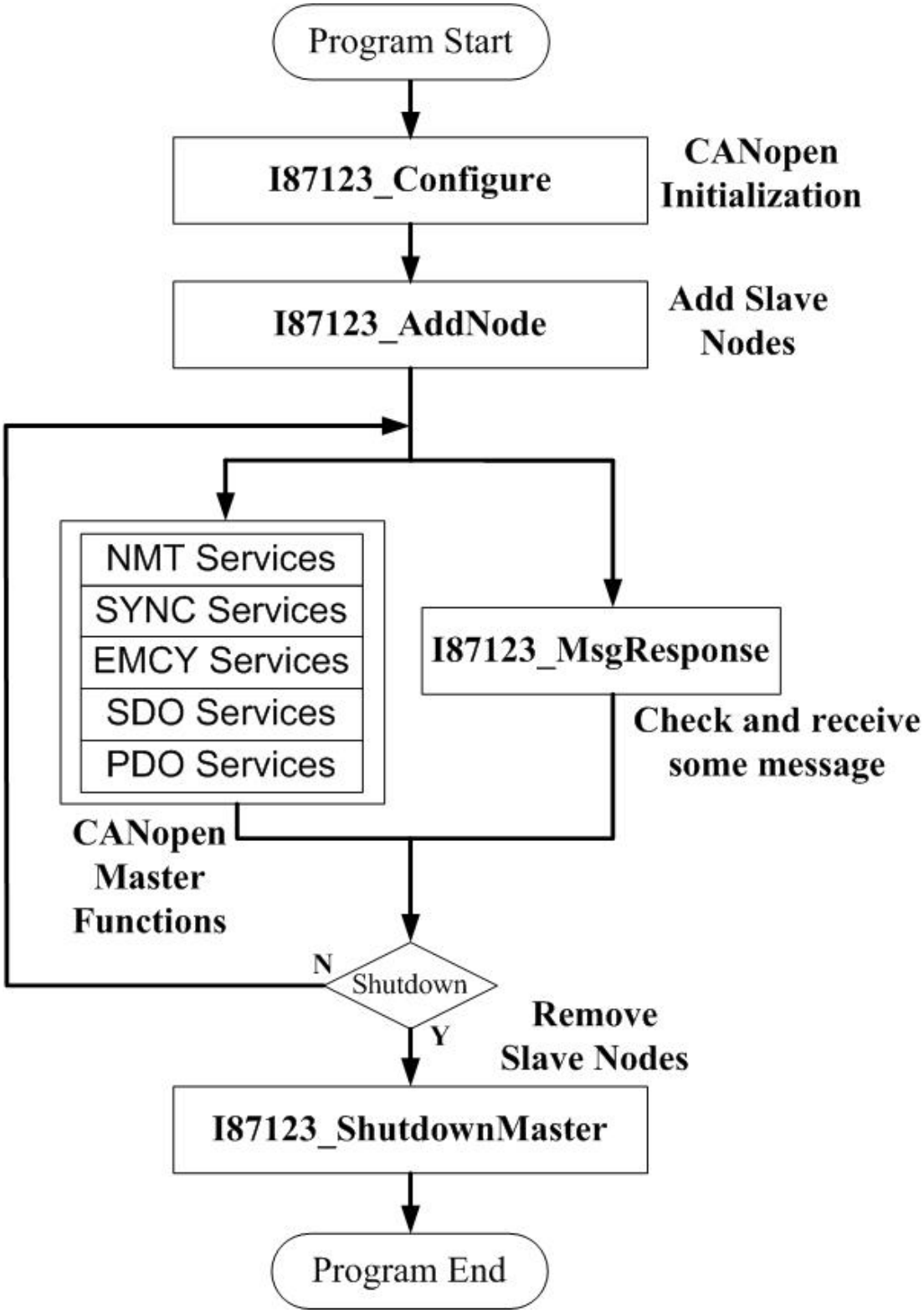


Figure 3.1 Main programming sequences

3.4. Communication Services Introduction

NMT Services

The CANopen Master Library provides several NMT services functions, such as the functions I87123_AddNode, I87123_RemoveNode, I87123_ChangeState, I87123_GetState, and I87123_Guarding. As the prerequisite for the master, the slave nodes have to be registered by I87123_AddNode, function with providing its Node-ID. The registered slave nodes can be individually removed from the node list by the function I87123_RemoveNode. Through NMT services, the NMT Master controls the state of the slave. Table 3.3 is the command value and corresponding NMT command for the input parameters of the function I87123_ChangeState. When using the function I87123_GetState, the slave status value and their descriptions are shown in the table 3.4. The Node Guarding protocol is implemented via the function I87123_Guarding. If the slave nodes are in the node list, users can change the node guarding parameters defined in the slave nodes by calling the function I87123_Guarding.

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

Table 3.3 NMT Command Specifier

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

Table 3.4 The State of The Slave

SDO Services

Initiate SDO download or Initiate SDO upload protocol is used when SDO data length $\leqq$ 4 bytes. If the SDO message data length $>$ 4 bytes, segment SDO download or upload protocol will be used. To call these two functions, I87123_ReadSDO and I87123_WriteSDO, the initiate protocol and segment protocol will be selected automatically according to data length.

I87123_AbortSDO function can abort a pending SDO transfer at any time. Applying the abort service will have no confirmation from the salve device.

PDO Services

The function I87123_InstallPDO is used for setting TPDOs or RPDOs mapping object. Each PDO object supports 0~8 application objects. These application objects defined in the CANopen specification DS401 are mapped to the DI/DO/AI/AO channels. After calling the function I87123_InstallPDO, the PDO communication object will be mapped and activated. If the PDO communication object is not needed no more, use the function I87123_RemovePDO to remove it.

The PDOs data are written to the PDO buffer by using the function I87123_WritePDO. This function can write all PDO 8-byte data or write some part of PDO 8-byte data. If users write some part of the PDO data, the other part of the PDO data will not be changed. Users can use the function I87123_SetPDOResponse to change the response type of TPDO. In I87123_SetPDOResponse function, there are three types, data event, timer event, and remote only to set, and, the data event is the DI default type, the remote only is the AI default type. When devices response PDO data with data event or timer event, users can use the function I87123_MsgResponse to read these data stored in the PDO buffer.

In CANopen specification, users can get the TxPDOs data by applying the remote transmit request CAN frame. The function I87123_RemotePDO is needed in this case.

SYNC Services

Calling the function I87123_SendSYNC starts the SYNC object transmission. This function supports single SYNC message and cyclic SYNC message. The timer parameter of the function I87123_SendSYNC can adjust the cyclic period of SYNC COB-ID sent by master if the cyclically parameter is 1. This timer parameter range is from 0 to 65535ms. If the timer parameter is set to 0, the SYNC object transmission will be stopped. When the cyclically parameter is 0, the function will send single SYNC message.

EMCY Services

Emergency objects are triggered by the occurrence of a device internal error situation. Users can call the function I87123_MsgResponse to receive EMCY message if there are CAN slaves to send EMCY.

3.5. Function Description

3.5.1. I87123_GetVersion

- **Description:**

  This function is used to obtain the version information of I87123.lib library.

- **Syntax:**

  float I87123_GetVersion(**void**)

- **Parameter:**

  None

- **Return:**

  LIB library version information.

3.5.2. I87123_Configure

- **Description:**

  The function must be applied when configuring the CAN controller and initialize the I-87123. It must be called once before using other functions of I87123.lib. **And note that before use this module**, users must use **ChangeSlotTo87K(unsigned char slot)** function in WinCE or use **ChangeToSlot(unsigned char slot)** function in I-8000 host to change slot to the slot of the I-87123 module.

- **Syntax:**

  int I87123_Configure(**unsigned char** baudrate)

  Parameter:

  baudrate: [input] The baudrate of the I-87123

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

- **Return:**

  I87123_OK
  I87123_CONFIG_ERR
  I87123_PARATERS_ERR
  I87123_TIMEOUT

3.5.3. I87123_ShutdownMaster

- **Description:**

  The function I87123_ShutdownMaster removes all the slaves that had added to master and stop all the functions of I-87123. The function must be call before exit the users' application programs.

- **Syntax:**

  int I87123_ ShutdownMaster (void)

- **Parameter:**

  None

- **Return:**

  I87123_OK
  I87123_REMOVE_NODE_ERR
  I87123_TIMEOUT

3.5.4. I87123_AddNode

- **Description:**

  The function I87123_AddNode can add a CANopen slave with specified Node ID into the master node list. After call this function to add a slave, the slave will into the operational state directly and the default TxPDO COB ID(0x180 + node ID, 0x280 + node ID, 0x380 + node ID, 0x480 + node ID)and RxPDO COB ID(0x200 + node ID, 0x300 + node ID, 0x400 + node ID, 0x500 + node ID) will also be installed if the slave supports these default PDO COB ID. The added node can be removed from the master node list by the function I87123_RemoveNode.

- **Syntax:**

  int I87123_AddNode(**unsigned char** node)

- **Parameter:**

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

- **Return:**

  I87123_OK
  I87123_ADD_NODE_ERR
  I87123_NODE_NUMBER_ERR
  I87123_TIMEOUT

3.5.5. I87123_RemoveNode

- **Description:**

  The function I87123_RemoveNode removes the slave with the specified Node-ID from node list of the node manager. It requires a valid Node-ID, which has installed by the function I87123_AddNode before.

- **Syntax:**

  int I87123_RemoveNode(**unsigned char** node)

- **Parameter:**

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

- **Return:**

  I87123_OK
  I87123_REMOVE_NODE_ERR
  I87123_NODE_NUMBER_ERR
  I87123_TIMEOUT

3.5.6. I87123_ChangeState

- **Description:**

  The function I87123_ChangeState 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 CAN network will be changed.

- **Syntax:**

  int I87123_ChangeState(**unsigned char** node,
  unsigned char state)

- **Parameter:**

  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

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_PARATERS_ERR
  I87123_TIMEOUT

3.5.7. I87123_GetState

- **Description:**

  The function I87123_GetState can get the NMT state from slaves.

- **Syntax:**

  int I87123_GetState(**unsigned char** node,
  unsigned char *state)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  ***state:** [output] The NMT state of the slave.

  4: STOPPED
  5: OPERATIONAL
  127: PRE-OPERATIONAL

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_TIMEOUT

3.5.8. I87123_Guarding

- **Description:**

  Use the function I87123_Guarding to set Guard Time and Life Time Factor of the specified slave with 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 form salve, I-87123 will send a error information to users. Users need to use I87123_MsgResponse to get this information. 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.

- **Syntax:**

  int I87123_Guarding(**unsigned char** node,
  unsigned short guardtime,
  unsigned char lifetime)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  guardtime: [input] Guard Time (1 ~ 65535).
  lifetime: [input] Life Time Factor (1 ~ 255).

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_SET_GUARD_ERR
  I87123_NODE_NUMBER_ERR
  I87123_TIMEOUT

3.5.9. I87123_SendSYNC

- **Description:**

  Use the function I87123_SendSYNC to send a SYNC message with specified COB-ID cyclically. If the timer is 0, the SYNC message will be stopped.

- **Syntax:**

  int I87123_SendSYNC(**unsigned short** cobid,
  unsigned char cyclically,
  unsigned short timer)

- **Parameter:**

  cobid: [input] COB-ID used by the SYNC object.
  cyclically: [input] If the parameter is 0, the SYNC message will be send once. If the parameter is 1, the SYNC message will be send cyclically with timer parameter.
  timer: [input] SYNC message transmission period. If the timer is 0, the SYNC message will be stopped. The parameter is useless if parameter cyclically is set to 0.

- **Return:**

  I87123_OK
  I87123_COBID_ERR
  I87123_SYNC_SEND_ERR
  I87123_TIMEOUT

3.5.10. I87123_ChangeEMCYID

- **Description:**

  Use the function I87123_ChangeEMCYID to change the EMCY COB-ID of a slave device.

- **Syntax:**

  int I87123_ ChangeEMCYID (**unsigned char** node,
  unsigned short cobid)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobid: [input] COB-ID used by the EMCY object.

- **Return:**

  I87123_OK
  I87123_NODE_NUMBER_ERR
  I87123_SDO_WRITEDATA_ERR
  I87123_SDO_SEND_ERR
  I87123_TIMEOUT

3.5.11. I87123_ChangeSYNCID

- **Description:**

  Use the function I87123_ChangeSYNCID to change the SYNC COB-ID of a slave device.

- **Syntax:**

  int I87123_ ChangeSYNCID (**unsigned char** node,
  unsigned short cobid)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobid: [input] COB-ID used by the SYNC object.

- **Return:**

  I87123_OK
  I87123_NODE_NUMBER_ERR
  I87123_SDO_WRITEDATA_ERR
  I87123_SDO_SEND_ERR
  I87123_TIMEOUT

3.5.12. I87123_AobrtSDO

- **Description:**

  Call function I87123_AbortSDO to cancel the SDO transmission. The parameter node is used to specify which SDO communication will be terminated between the master and the specified salve device.

- **Syntax:**

  int I87123_AbortSDO(**unsigned char** node,
  unsigned short index,
  unsigned char subindex)

- **Parameter:**

  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.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_SEND_ERR
  I87123_TIMEOUT

3.5.13. I87123_ReadSDO

- **Description:**

  The function I87123_ReadSDO is useful to the SDO upload from a specified slave. When users use this function, pass the slave device node ID, object and object subindex into this function. This function supports both expedition mode and segment mode.

- **Syntax:**

  int I87123_ReadSDO(**unsigned char** node,
  unsigned short index
  unsigned char subindex
  unsigned char *len
  unsigned char *rdata)

- **Parameter:**

  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.
  ***len:** [output] Total data length.
  ***rdata:** [output] SDO data respond from the specified slave device.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_SEND_ERR
  I87123_SDO_SEGMENT_ERR
  I87123_TIMEOUT

3.5.14. I87123_WriteSDO

- **Description:**

  The function I87123_WriteSDO can send out a SDO message to specified salve device. This procedure is also called download SDO protocol. The parameter node of the function I87123_WriteSDO 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 object dictionary of specified slave devices. This function support both expedition mode and segment mode.

- **Syntax:**

  int I87123_WriteSDO(**unsigned char** node,
  unsigned short index, **unsigned char** subindex,
  unsigned char len, **unsigned char** *tdata
  unsigned char *rlen, **unsigned char** *rdata)

- **Parameter:**

  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.
  len: [input] Total data size to be written.
  ***tdata:** [input] The SDO data written to slave device.
  ***rlen:** [input] Total data size of responded data.
  ***rdata:** [input] SDO data responded from the specified slave device.

- **Return:**

  I87123_OK
  I87123_DATA_LEN_ERR
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_WRITEDATA_ERR
  I87123_SDO_SEND_ERR
  I87123_SDO_SEGMENT_ERR
  I87123_TIMEOUT

3.5.15. I87123_SetPDOResponse

- **Description:**

  When users set the slave device to some PDO transmission types, the slave will return the TxPDO message automatically if some event is triggered or the event timer is time up. Call the function I87123_SetPDOResponse to set the PDO message modes. Afterwards, I-87123 will follow the selected PDO message mode to feedback the received PDO message to users.

- **Syntax:**

  int I87123_SetPDOResponse(**unsigned char** node,
  unsigned short cobid,
  unsigned char mode)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobid: [input] COB-ID used by the PDO object.
  mode: [input] PDO response mode.

Mode Value	Description
0	I-87123 will not feedback any received PDO messages send from slave expect remote-transmit-requist PDO
1	I-87123 will feedback all of received PDO messages to users

.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_COBID_ERR
  I87123_NODE_NUMBER_ERR
  I87123_RESPONSE_ERR
  I87123_TIMEOUT

3.5.16. I87123_MsgResponse

- **Description:**

  Call this function to receive error information produced from I-87123 or all active transmitted messages from slave devices. These messages include PDO (expect remote-transmit-requist PDO) and EMCY. If there is no message in software buffer, calling I87123_MsgResponse willl get the error code I87123_EMPTY. If this function receives PDO message, I87123_OK will be returned. Users can use parameters of this function to obtain the PDO messages. If other error codes are returned, users can also check the parameters of this function to know what is happen on specified node.

- **Syntax:**

  int I87123_MsgResponse(**unsigned char** *node,
  unsigned short *cobid,
  unsigned char *len,
  unsigned char *rdata)

- **Parameter:**

  ***node:** [output] Slave device Node-ID (1~127). This parameter is useless for receiving a PDO message.
  ***cobid:** [output] COB-ID used by the communication object.
  ***len:** [output] data size
  ***rdata:** [output] Data pointer to point the response data..

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_SETGUARD_ERR
  I87123_GUARD_FAILED
  I87123_EMCY
  I87123_RECEIVE_ERR
  I87123_EMPTY
  I87123_TIMEOUT

3.5.17. I87123_InstallPDO

- **Description:**

  After calling the I87123_InstallPDO function, a new PDO cobid will be installed or an old PDO cobid will be modified in the PDO object list of CANopen Master Library stack. If the slave device has defined the default PDO object, these default PDO will be installed automatically when the function I87123_AddNode is called.

- **Syntax:**

  int I87123_InstallPDO(**unsigned char** node,
  unsigned short cobid,
  unsigned char txrxtype,
  unsigned char channel,
  unsigned char *tdata)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobId: [input] COB-ID used by the PDO object.
  txrxtype: [input] PDO type (0: RxPDO, 1: TxPDO).
  channel: [input] PDO mapping object subindex value (1 ~ 8).
  ***tdata:** [input] 4-byte information of mapped application object. Users need to look up the user manual of CAN slave device to find the information of application object, and obey the following format to fill this parameter.

  tdata[0] : The numbers of bit of specified application object.
  tdata[1] : The subindex of specified application object.
  tdata[2] : The low byte of index of specified application object.
  tdata[3] : The high byte of index of specified application object.

  For example, there is an application object defined by some CAN slave device. This application object use index 0x6000 and subindex 0x06. It is used to store a 16-bit data. When users add this specified application object in the PDO object list of I-87123, tdata[0] is set to 0x10 (for indicating the stored data is 16-bit),

tdata[1] is 0x06 (for indicating the subindex is 0x06), tdata[2] is 0x00 (for indicating the low byte of the index 0x6000), and tdata[3] is 0x60 (for indicating the high byte of the index 0x6000).

- **Return:**

I87123_OK
I87123_STATUS_ERR
I87123_COBID_ERR
I87123_NODE_NUMBER_ERR
I87123_SDO_SEND_ERR
I87123_PDO_TYPE_ERR
I87123_PDO_INSTALL_ERR
I87123_PARATERS_ERR
I87123_TIMEOUT

3.5.18. I87123_RemovePDO

- **Description:**

  The function I87123_RemovePDO can remove a TxPDO or RxPDO had installed by the I87123_InstallPDO. This function also can remove single channel mapped in TxPDO or RxPDO.

- **Syntax:**

  int I87123_RemovePDO(**unsigned char** node,
  unsigned short cobid,
  unsigned char txrxtype,
  unsigned char channel)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobid: [input] COB-ID used by the PDO object.
  txrxtype: [input] PDO type (0: RxPDO, 1: TxPDO).
  channel: [input] [input] PDO mapping object subindex value (0 ~ 8). If channel parameter is 0, the specified PDO object will be removed. If others (1 ~ 8), the specified subindex content of the PDO will be removed.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_COBID_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_SEND_ERR
  I87123_PDO_TYPE_ERR
  I87123_PDO_CHANNEL_ERR
  I87123_PARATERS_ERR
  I87123_TIMEOUT

3.5.19. I87123_WritePDO

- **Description:**

Call the function I87123_WritePDO to send out a PDO message to the specified slave device. Before using this function, users need to use the function I87123_InstallPDO to install the PDO object if users want to use non-default PDO. Then, change the NMT state of target slave device to operational mode by using the function I87123_ChangeState if the slave is not in the operational mode. Use the parameter offset to set the start byte position of PDO data which need to be modified, and use the parameters *rdata and len to point the data and data length which users want to fill to the PDO data.

- **Syntax:**

int I87123_WritePDO(**unsigned short** cobid,
unsigned char offset,
unsigned char dlen,
unsigned char *rdata)

- **Parameter:**

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).
***rdata:** [output] The data pointer point to the PDO data.

- **Return:**

I87123_OK
I87123_DATA_LEN_ERR
I87123_STATUS_ERR
I87123_COBID_ERR
I87123_PDO_SEND_ERR
I87123_TIMEOUT

3.5.20. I87123_RemotePDO

- **Description:**

  Use the function I87123_RemotePDO to send a RTR (remote-transmit-request) PDO message to the slave device.

- **Syntax:**

  int I87123_RemotePDO(**unsigned short** cobid,
  unsigned char *len,
  unsigned char *rdata)

- **Parameter:**

  cobid: [input] COB-ID used by the PDO object.
  ***len:** [output] The data length of the RTR PDO message.
  ***rdata:** [output] The PDO message received from the slave device.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_COBID_ERR
  I87123_PDO_SEND_ERR
  I87123_TIMEOUT

3.5.21. I87123_PDOTxType

- **Description:**

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

- **Syntax:**

  int I87123_PDOTxType(**unsigned char** node,
  unsigned short cobid
  unsigned char txtype)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  cobid: [input] COB-ID used by the PDO object.
  txtype: [input] Transmission type of TxPDO (0 ~ 255).

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_COBID_ERR
  I87123_PDO_SEND_ERR
  I87123_TIMEOUT

3.5.22. I87123_WriteDO

- **Description:**

  Use this function to output one byte (8 channels) DO data.

- **Syntax:**

  int I87123_WriteDO(**unsigned char** node,
  unsigned char dochannel
  unsigned char value)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  dochannel: [input] The subindex of index 0x6200 of specified application object. Please refer to slave device user manual for more detail information.
  value: [input] The value for 8-channel digital output which is used 1 byte for presentation.

- **Return:**

  I87123_OK
  I87123_DATA_LEN_ERR
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_WRITEDATA_ERR
  I87123_SDO_SEND_ERR
  I87123_CHANNEL_ERR
  I87123_TIMEOUT

3.5.23. I87123_ReadDI

- **Description:**

  Use this function to read one byte (8 channels) DI data.

- **Syntax:**

  int I87123_ReadDI(**unsigned char** node,
  unsigned char dichannel
  unsigned char *value)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  dichannel: [input] The subindex of index 0x6000 of specified application object. Please refer to slave device user manual for more detail information.
  ***value:** [output] The value for 8-channel digital input which is used 1 byte for presentation.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_SEND_ERR
  I87123_CHANNEL_ERR
  I87123_TIMEOUT

3.5.24. I87123_WriteAO

- **Description:**

  Use this function to output one channel AO data.

- **Syntax:**

  int I87123_WriteAO(**unsigned char** node,
  unsigned char aochannel
  unsigned short value)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  aochannel: [input] The subindex of index 0x6411 of specified application object. Please refer to slave device user manual for more detail information.
  value: [input] One AO channel value which is used two bytes for presentation.

- **Return:**

  I87123_OK
  I87123_DATA_LEN_ERR
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_WRITEDATA_ERR
  I87123_SDO_SEND_ERR
  I87123_CHANNEL_ERR
  I87123_TIMEOUT

3.5.25. I87123_ReadAI

- **Description:**

  Use this function to read one channel AI data.

- **Syntax:**

  int I87123_ReadAI(**unsigned char** node,
  unsigned char aichannel
  unsigned short *value)

- **Parameter:**

  node: [input] Slave device Node-ID (1~127).
  aichannel: [input] The subindex of index 0x6401 of specified application object. Please refer to slave device user manual for more detail information.
  ***value:** [output] Read one AI channel value which is used two bytes for presentation.

- **Return:**

  I87123_OK
  I87123_STATUS_ERR
  I87123_NODE_NUMBER_ERR
  I87123_SDO_SEND_ERR
  I87123_CHANNEL_ERR
  I87123_TIMEOUT

4. Demo Programs

There are two kinds of CANopen master demos, I-8000 series demo, WinCon series demo, WinPAC series demo, and LinPAC series demo. Users can find these demos in the CAN CD or on the web site.

The path of CAN CD

can_cd://canopen/master/I-87123

The address of the web site

http://www.icpdas.com/products/Remote_IO/can_bus/I-87123.htm

4.1. Brief of the demo programs

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

NMT Services: I87123_ChangeState, I-87123_GetState, I87123_Guarding

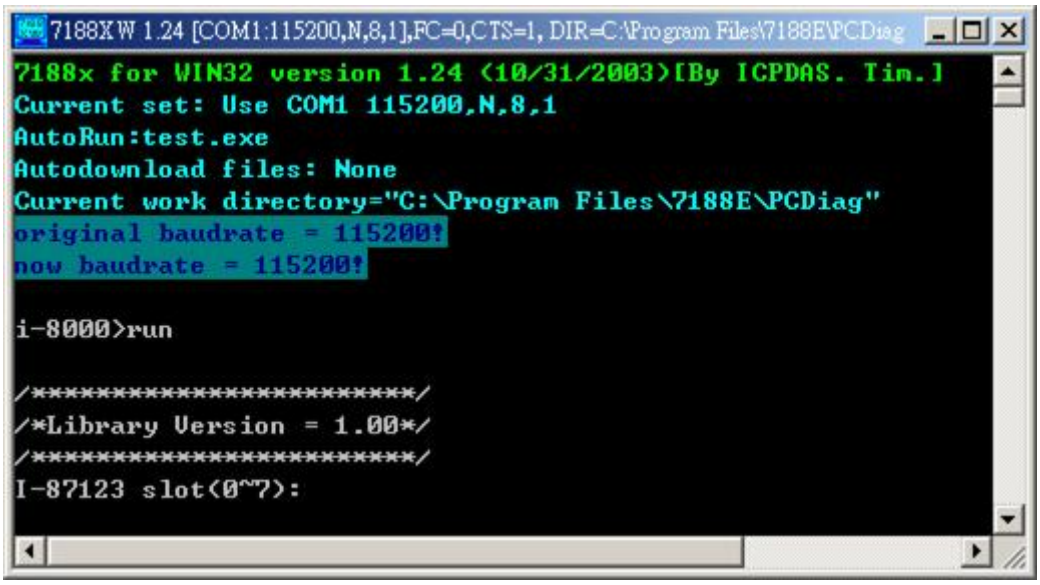
SDO Services: I87123_ReadSDO, I87123_WriteSDO, I87123_AbortSDO

PDO Services: I87123_InstallPDO, I87123_RemovePDO,
I87123_SetPDOResponse, I87123_PDOTxType,
I87123_WritePDO, I87123_RemotePDO

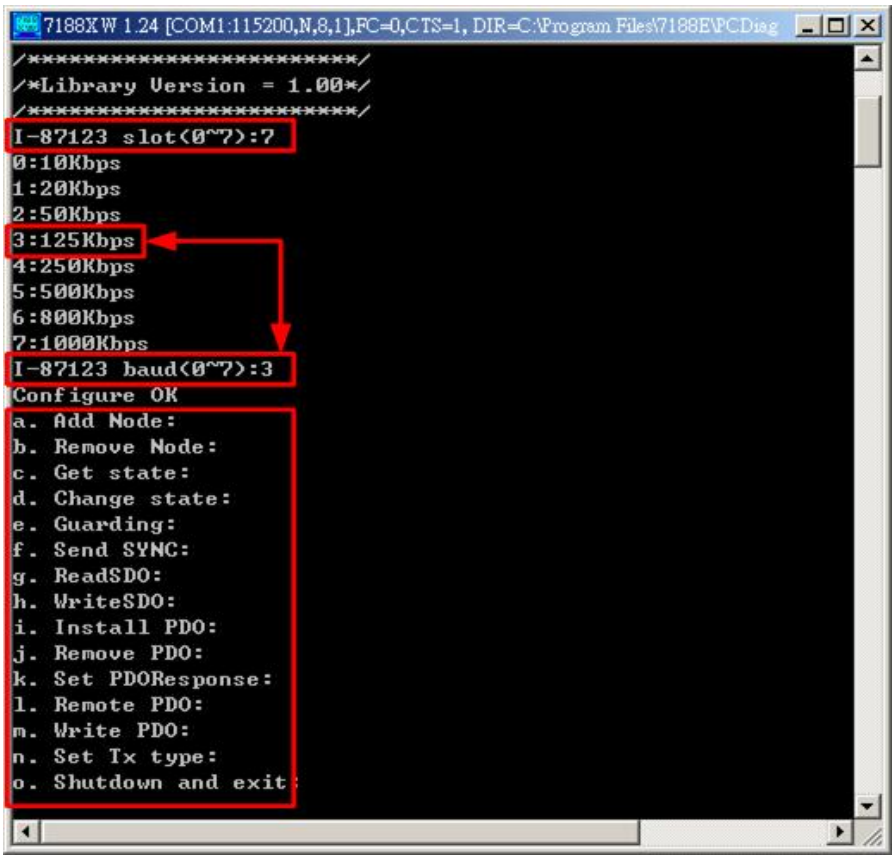
SYNC Services: I87123_ChangeSYNCID, I87123_SendSYNC

I-8000 series demo

When the demo runs, the user interface of the demo is shown below.

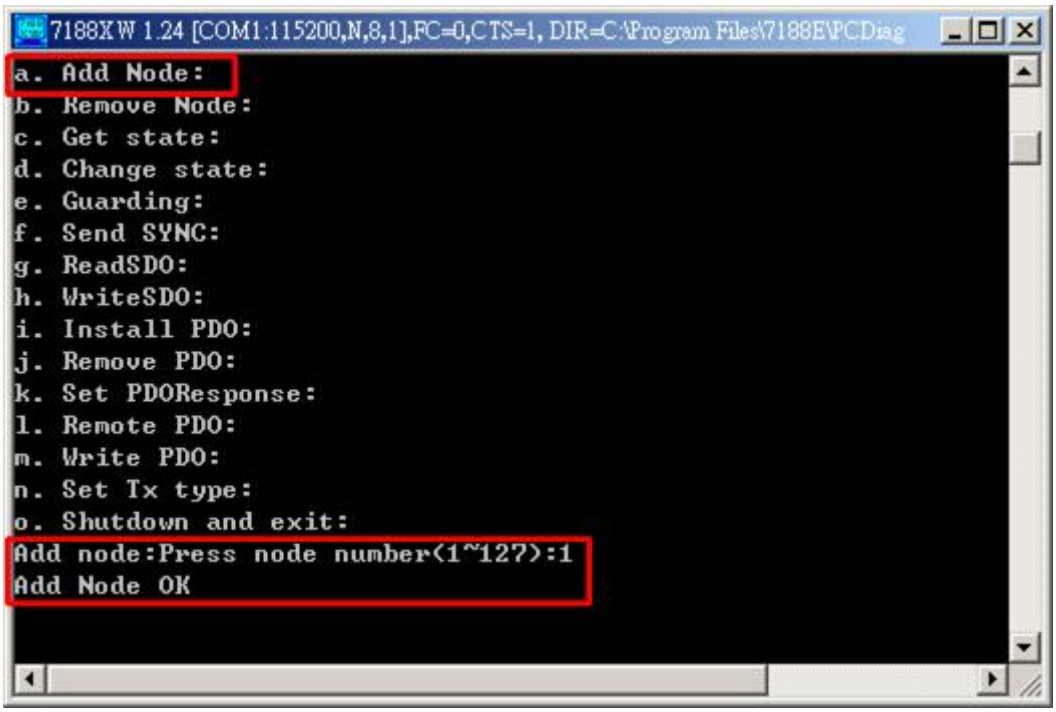


Users can see the version of the library, 1.00. Input the slot number and select the baud of I-87123 as below. Then, the I-87123 will be initialized and the total functions (a ~ o) of the demo will be represented.



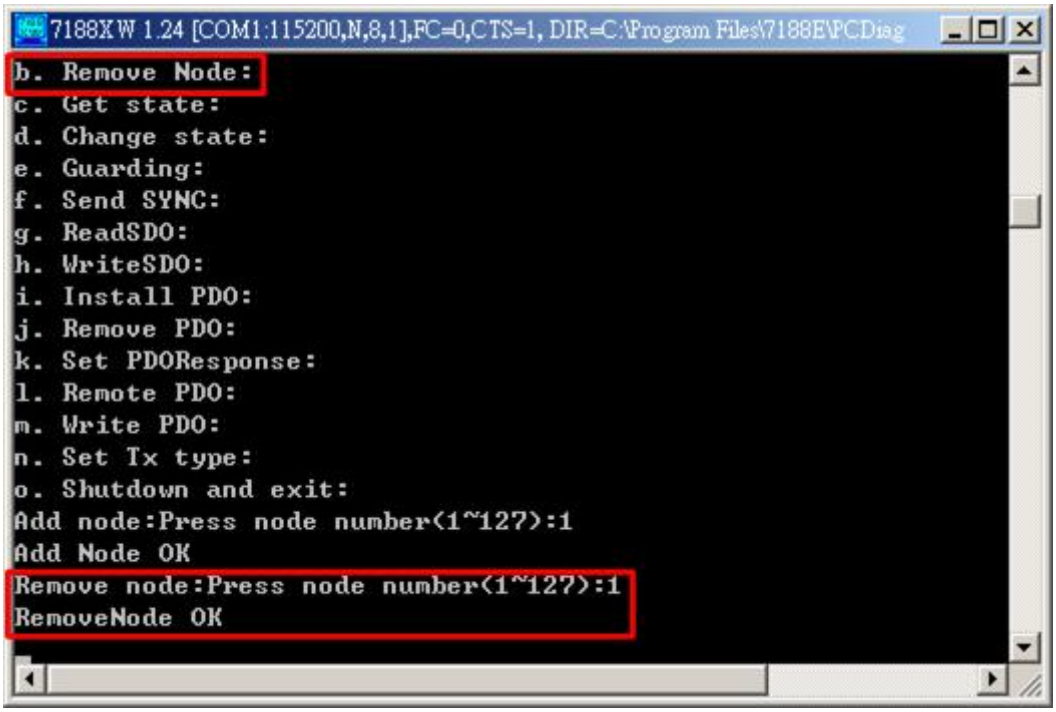
a. Add Node:

Before using other functions, the Add Node function must be call firstly. Here, add the CANopen slave node 1 to the I-87123 for example.



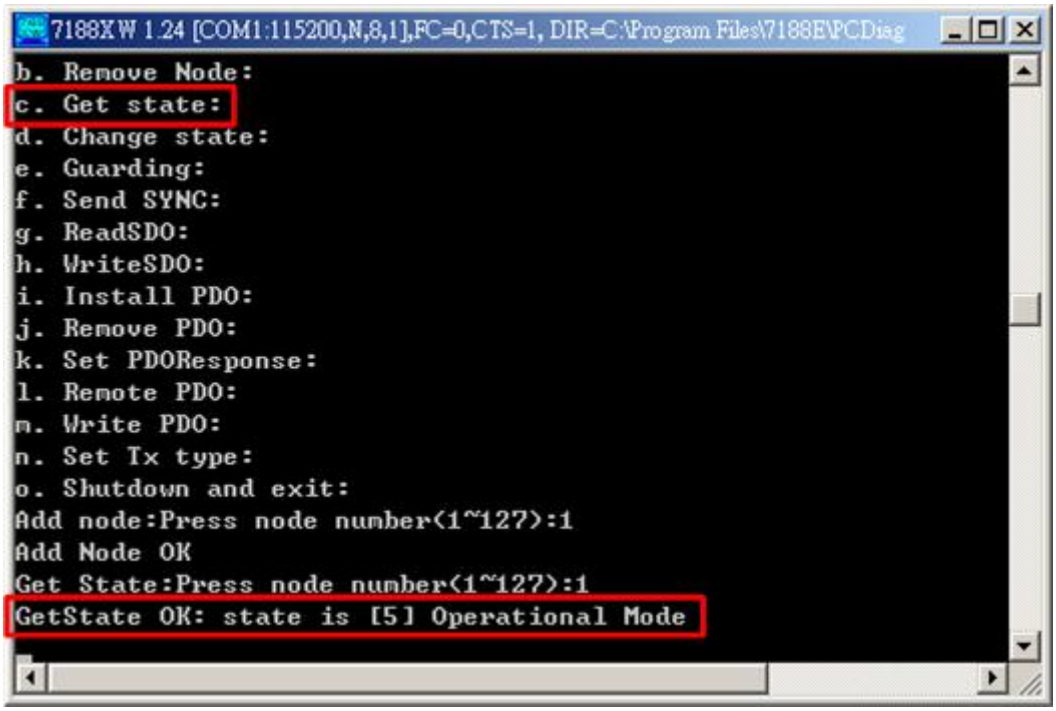
b. Remove Node:

When users want to delete the CANopen slave that had been add to I-87123, users can use Remove Node function to remove this slave.



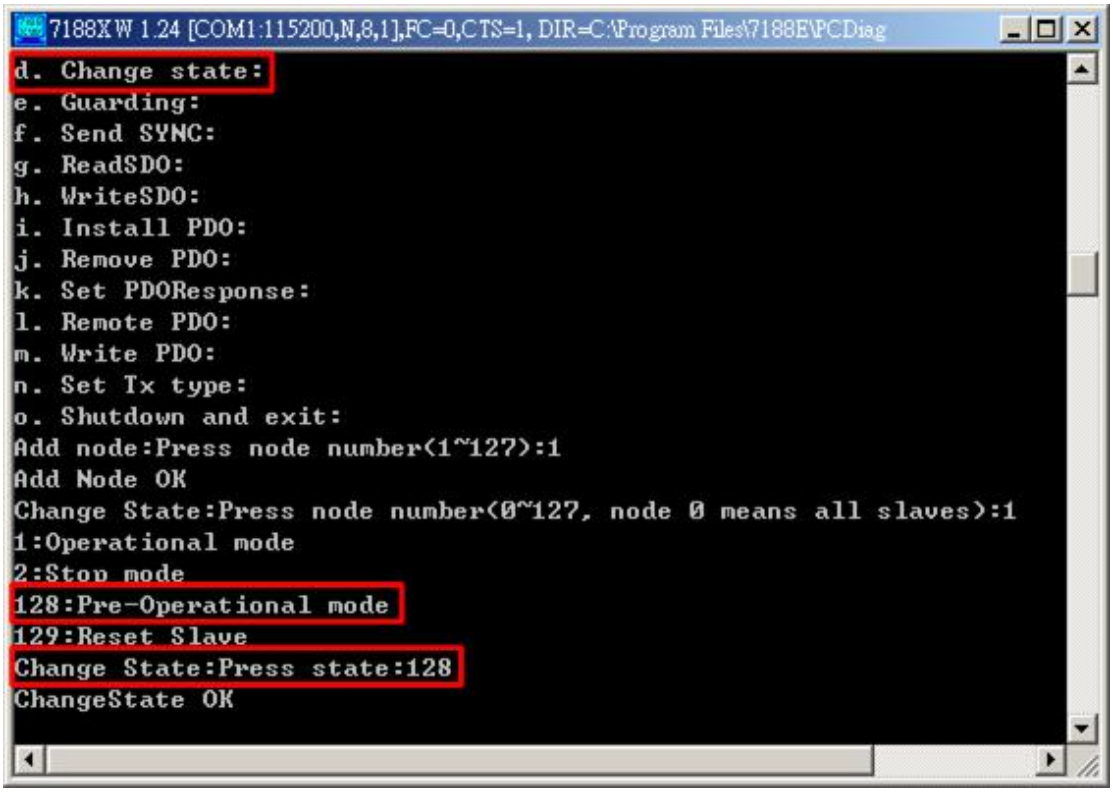
c. Get state:

If users want to get the state of the CANopen slave, the Get state function will be used. For example, press ‘c’ to get the state of node 1.



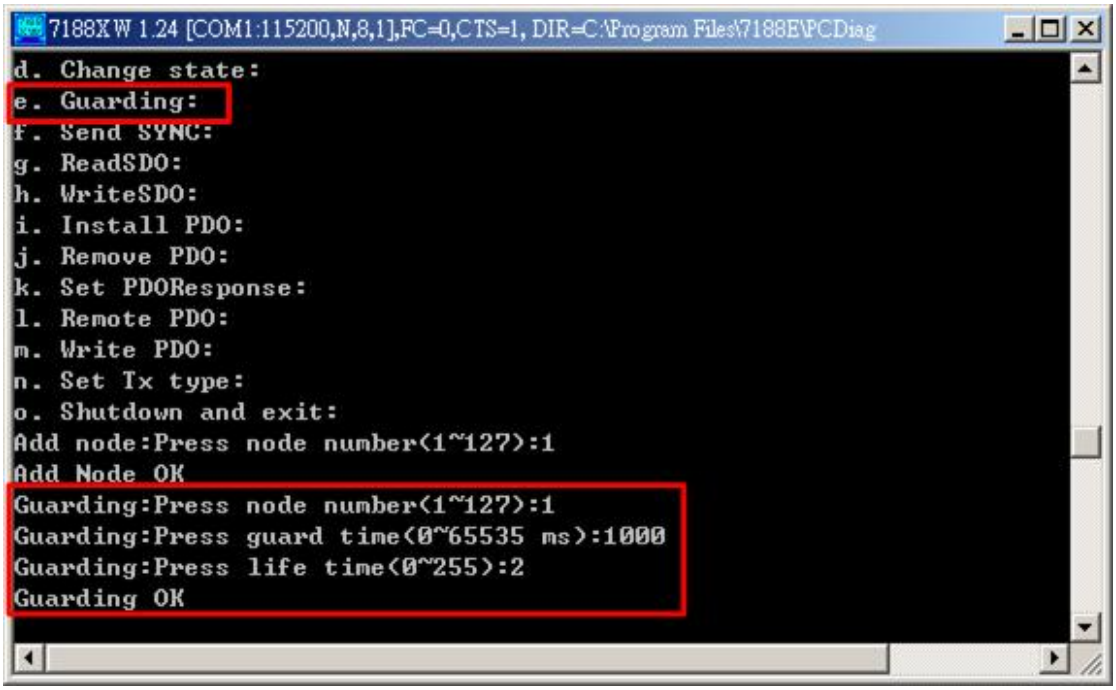
d. Change state:

Press ‘d’ to change the node state if users want to do that.

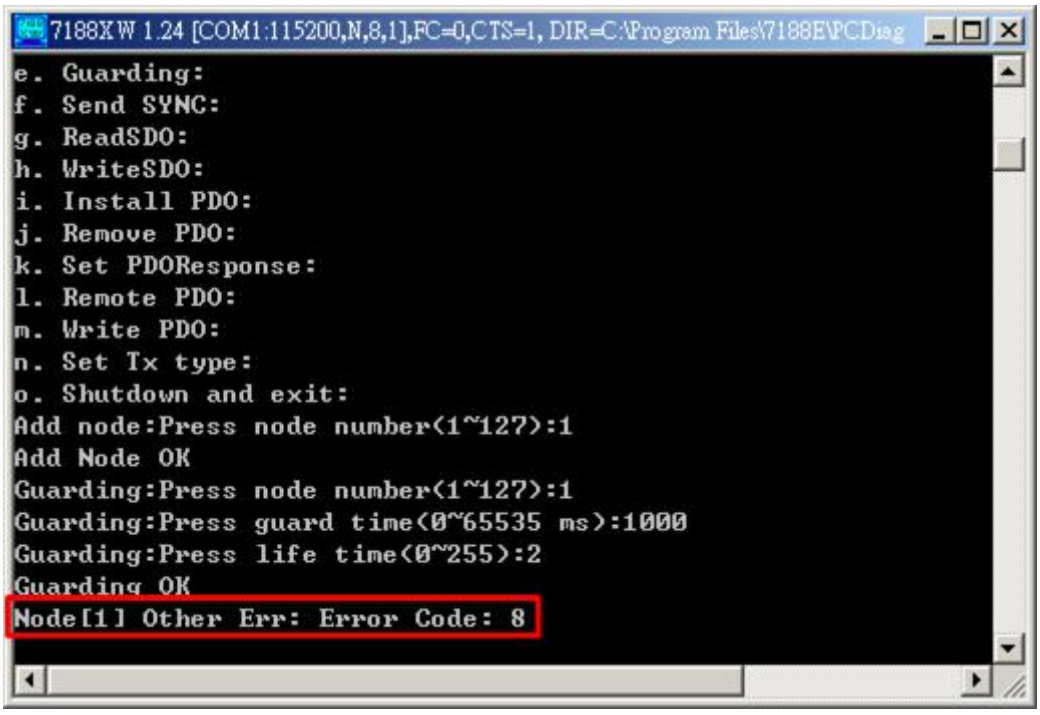


e. Guarding:

Users can use this function to guard the CANopen slave. For example, use this function to guard the node 1 and the guarding time and the life time are 1000ms and 2 times.

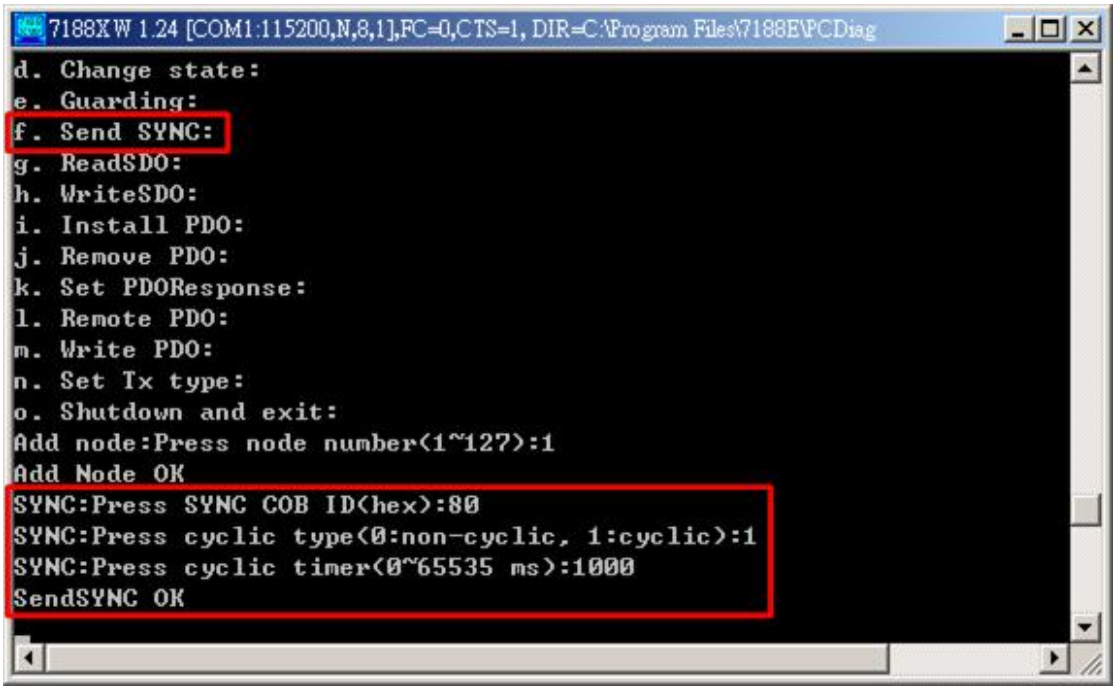


If the node 1 disconnects from I-87123, the guarding failed will be detected.



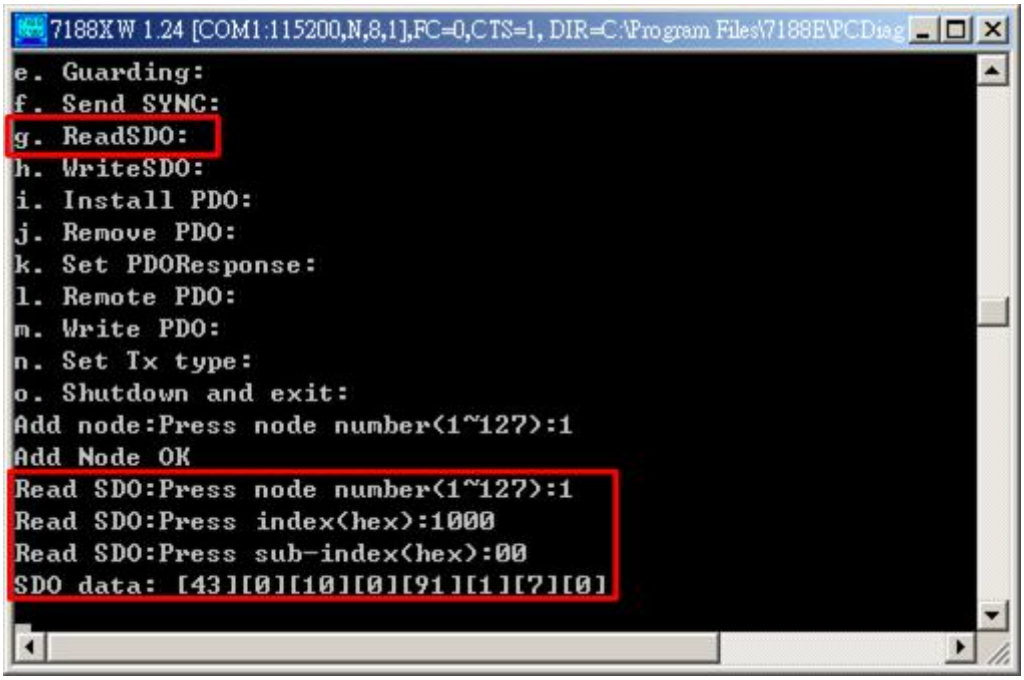
f. Send SYNC:

Users can send SYNC message through this function. If users want to send SYNC message once per second, users can set the cyclic parameter to 1 and timer parameter to 1000.



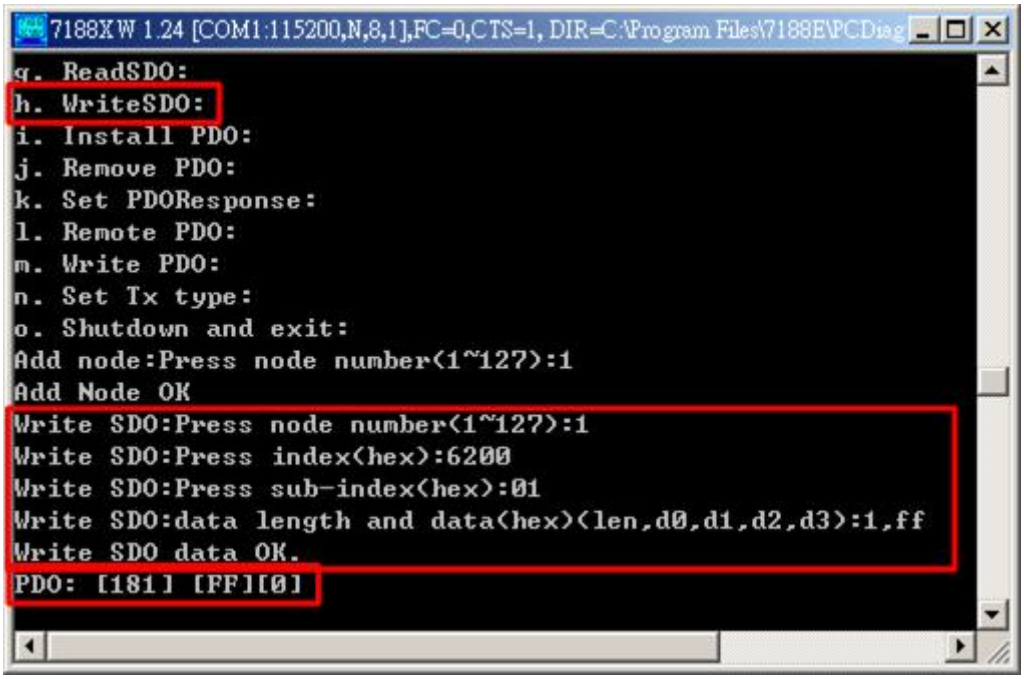
g. Read SDO:

Use SDO to read some data from the CANopen slave. Here, read the data of object with index 0x1000 and subindex 00.



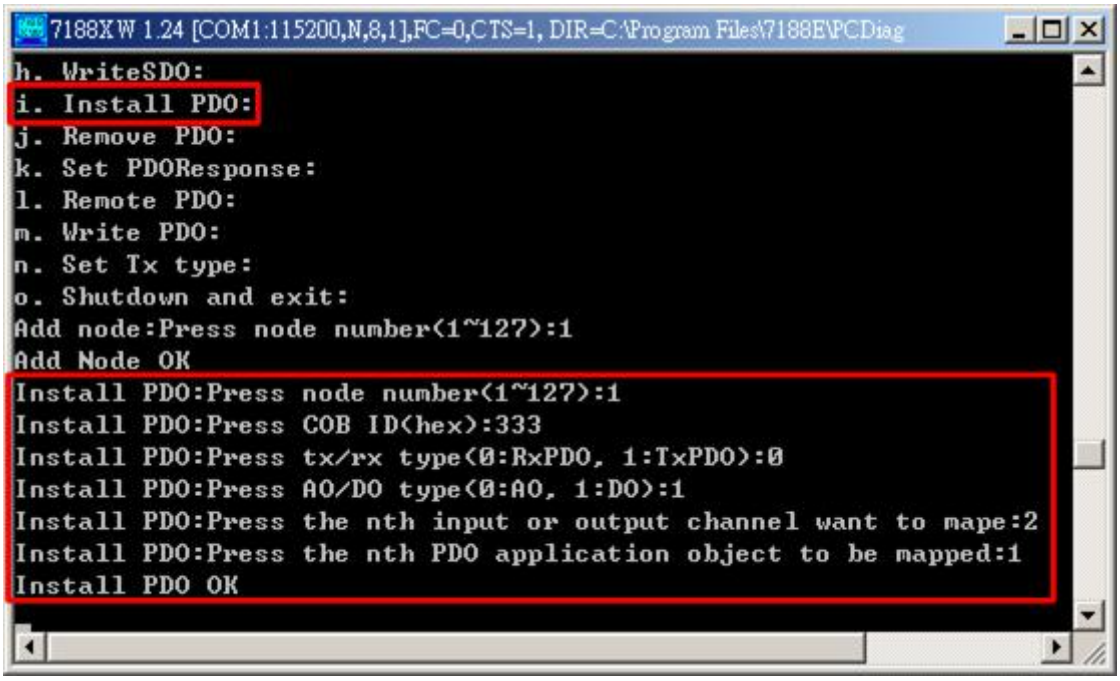
h. Write SDO:

Use SDO to write some data to the CANopen slave. For example, output the data 0xff to object which has index 0x6200 and sub-index 0x01 (For general I/O device, the range of index 0x6200 is usually indicated the data of DO channels of this device). The feedback DI data is 0xff if we connect the DO channels and DI channels of this general I/O device each other.



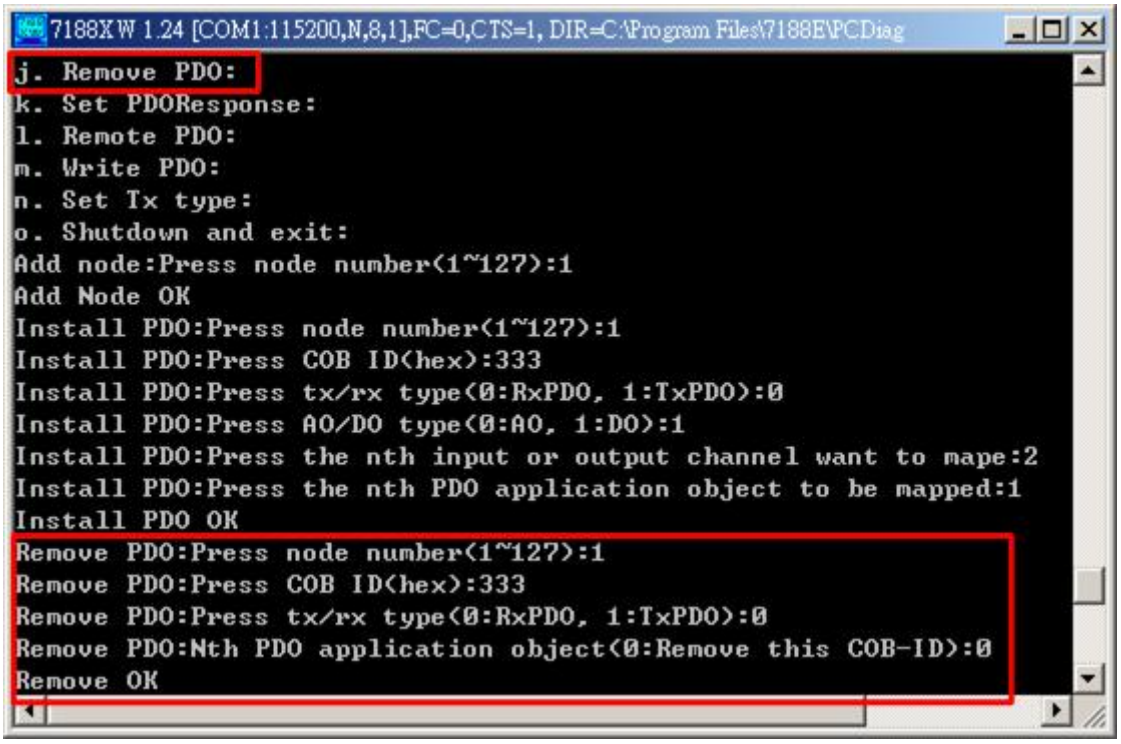
i. Install PDO:

If users want to change the PDO setting, add a new PDO COB-ID, or change PDO mapping data, install PDO function will be used.



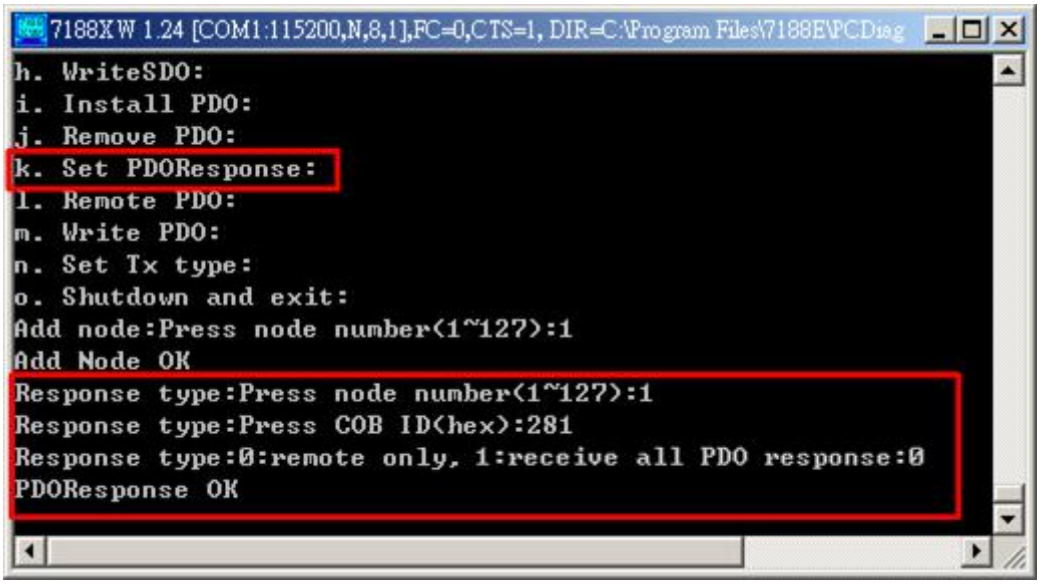
j. Remove PDO:

If users want to delete a PDO COB-ID or a PDO mapping data, the Remove PDO function is needed.



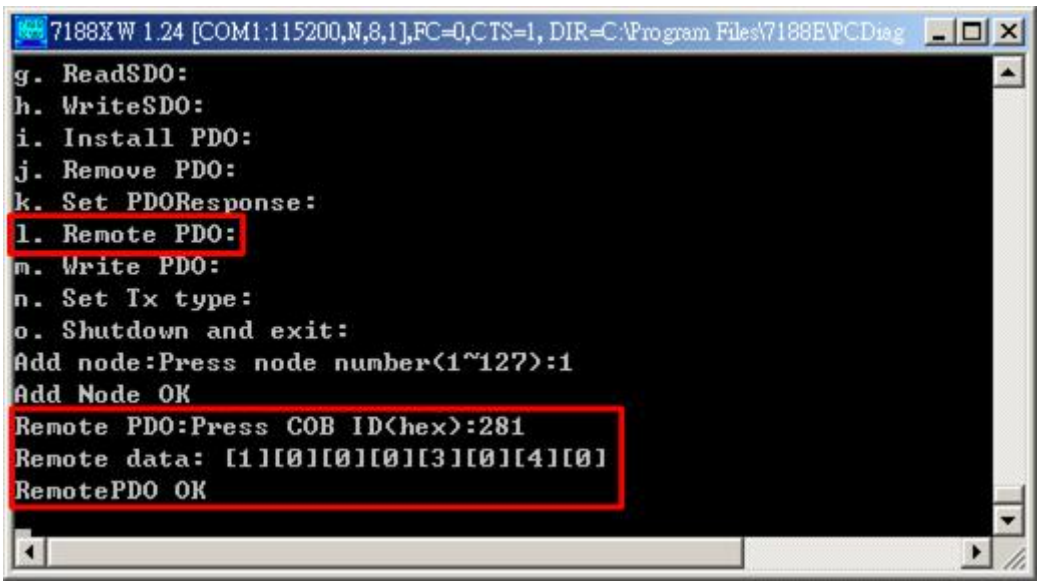
k. Set PDO Response:

There are two mode of PDO response type, remote only and receive all PDO response. Users can use this function to change the response type of the PDO.



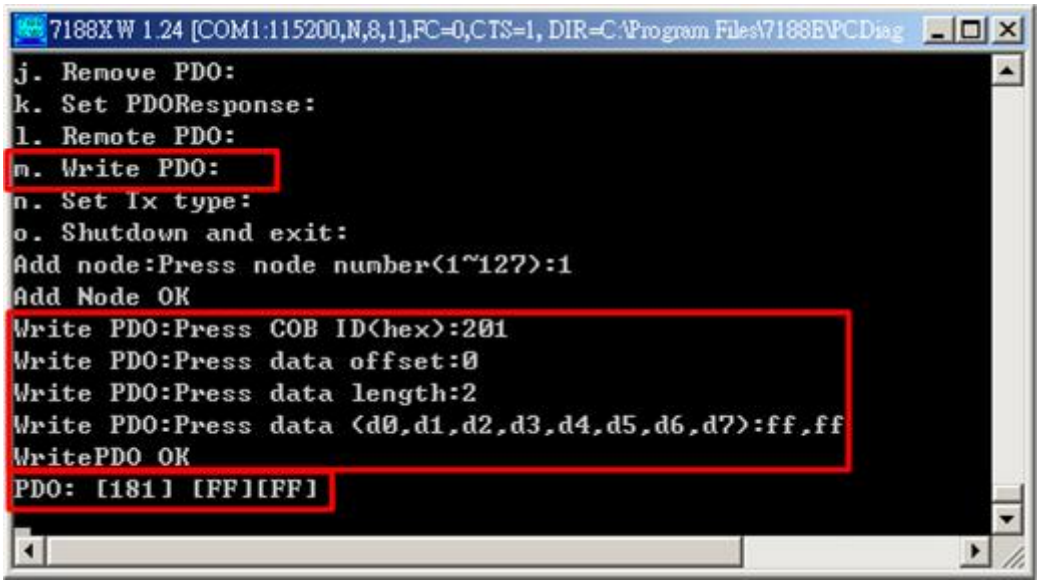
l. Remote PDO:

Use this function to get data through remote-transmit-request PDO.



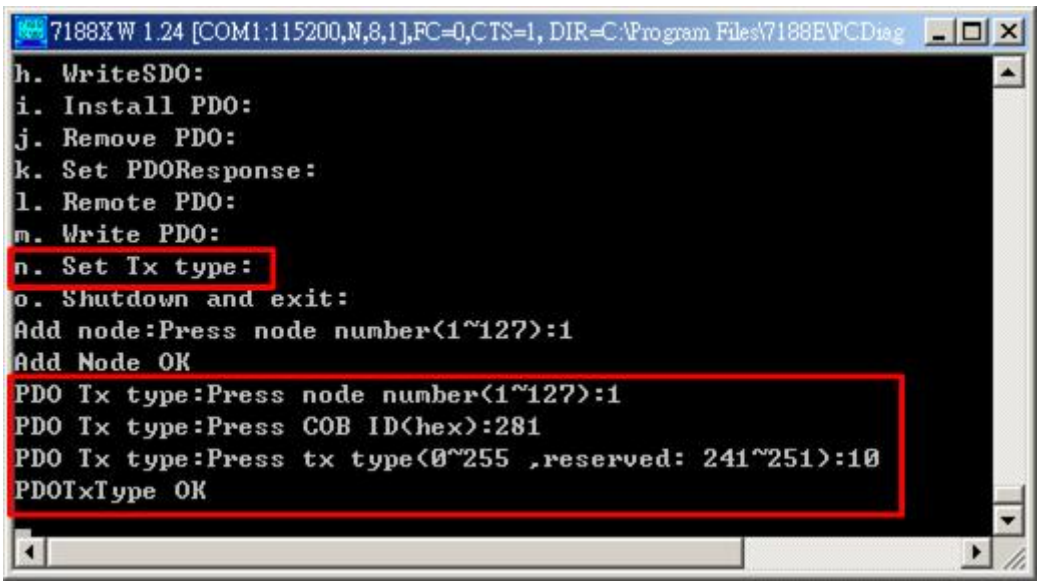
m. Write PDO:

Use this function to write data to slave through PDO. Users can use the function parameters to decide how many and what data will be output.



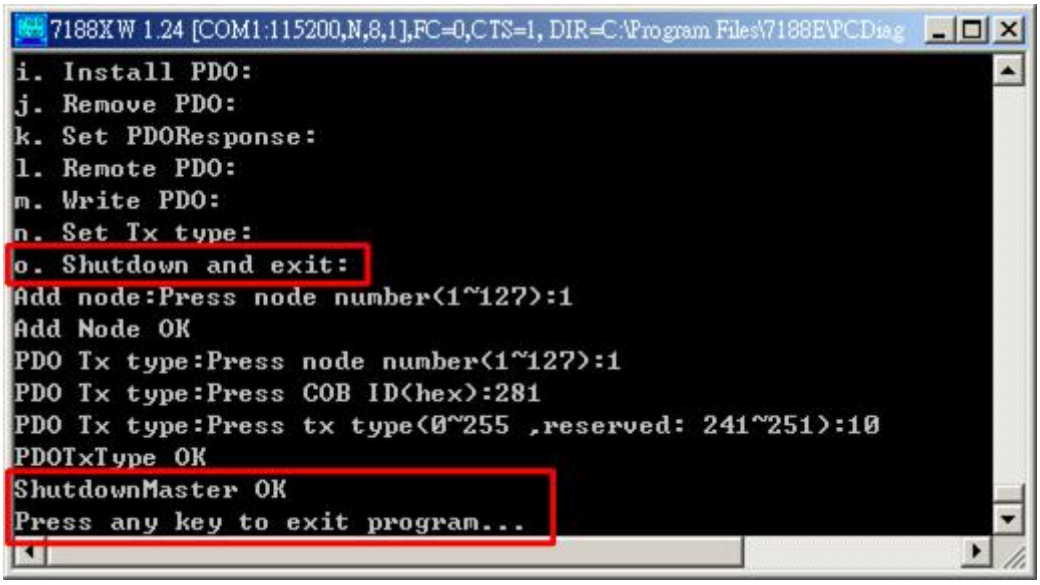
n. Set Tx type:

Users can use this function to set the transmission type of the PDO.



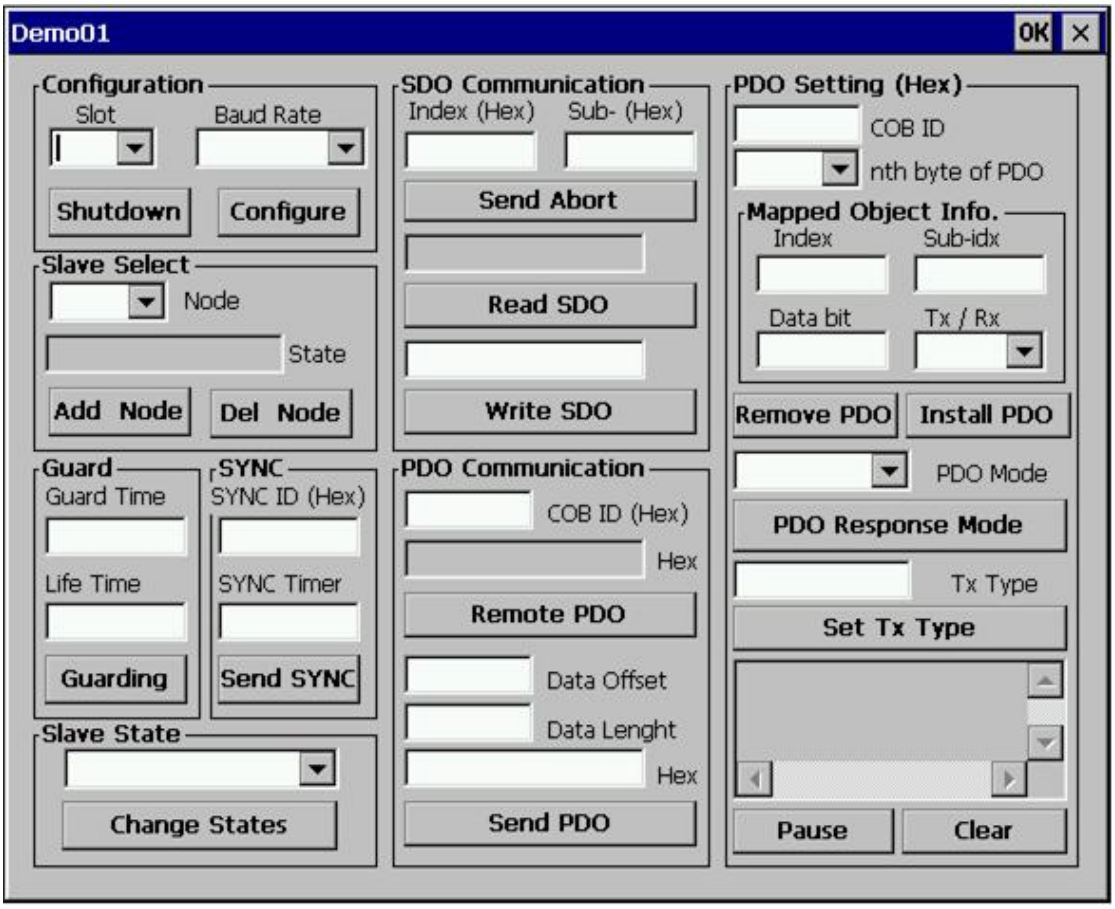
o. Shutdown and exit:

If users want to close this demo, don't forget to press 'o' to shutdown the master and exit this program.

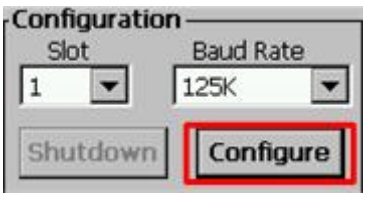


WinCon series demo and WinPAC series demo

When the demo is executed, the user interface of the demo is shown below. Users can see the version of the library, 1.00 on the top of the window.



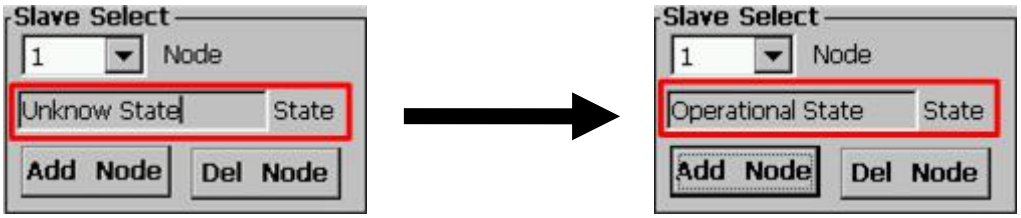
Before running the demo, users must to select which slot and what kind of baud rate the I-87123 would be configured, then click the Configure button to finish the configuration. Users also can click Shutdown button to stop the I-87123 before exiting this demo.



Configure I-87123

Shutdown I-87123

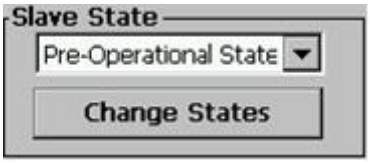
After finishing the configuration of I-87123, users can add CANopen nodes to I-87123 by selecting the node 1 at the Node combo box. If the State text shows “Unknow State”, it means that the node 1 has not been added to the I-87123 yet. If the node 1 has been added successfully, State text will show the NMT state of the node, such as “Operational State”.



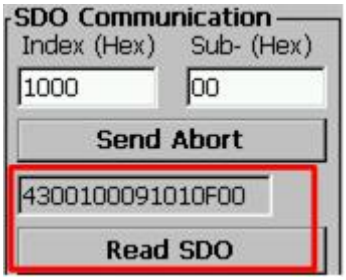
In the Guard frame, users can set the guarding time of a node. When users select the node 1 at the Node combo box, users can set the guard time and the life time to the node 1 after clicking the Guarding button.

In the SYNC frame, users can send the SYNC message by clicking Send SYNC button. If the SYNC Timer is 1000, then the I-87123 will send the SYNC ID once per second. If the SYNC Timer is 0, I-87123 will send the SYNC ID only once.

There are four kinds of NMT states in the Slave State frame. Select the node in the Slave Select frame, select the state in the Slave State frame to set the NMT state of slave, and click Change States button. The node will be changed to the new state.



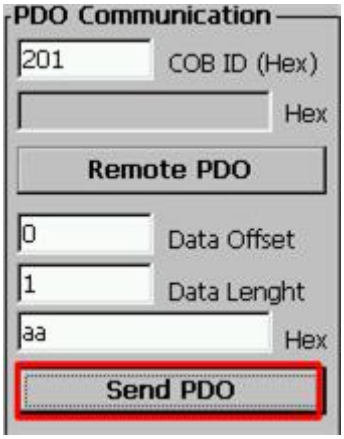
If users want to read SDO message, users must select the slave node firstly. Then, set index and sub-index of target object of slave device. Afterwards, click the Read SDO button, and the SDO message received from the slave will be shown on the edit box upper the Read SDO button.



If users want to write SDO message, users must select slave node, set index, sub-index, and input the data which will be written to CANopen slave at the edit box upper the Write SDO button. Then, the SDO response message will be shown on the edit box upper the Read SDO button.

In PDO Communication frame, users can input Tx-PDO COB ID at COB ID edit box and click Remote PDO button. The responded message of remote-transmit-request PDO will be shown on the edit box upper the Remote PDO button.

Except remote PDO, the PDO Communication frame can also write PDO data to CANopen slave. First, input Rx-PDO COB ID at COB ID edit box. Second, input the byte offset and byte length at Data Offset and Data Length edit box. Finally, input the PDO data which will be written at the edit box upper the Send PDO button, and click Send PDO button to send PDO message to slave device.



In PDO Setting frame, users can dynamically install a new PDO or remove a PDO, users can also set response mode of PDO message and transmission type in this frame. For PDO installation, users must select node id and set these six parameters. The first two parameters are PDO communication parameters, COB ID and nth byte of PDO. The last four parameters are PDO mapping object information, index, sub-index, data bit and Tx/Rx type. COB ID parameter is PDO COB ID which will be installation. For remove PDO, there are only four parameters, node number, COB ID, nth byte of PDO, and Tx/Rx type. If users want to remove all channels of a specified PDO COB ID, the parameter “nth object” must be set to ‘0’. After these parameters have been setting, click Install PDO button to install a new PDO channel or click Remove PDO button to remove a PDO channel which has been configured before.

In order to set response mode of a PDO message, the three parameters, node, COB ID and PDO Mode, must be used. The PDO parameter has two modes, 0 and 1. Mode 0 indicates that the I-87123 will response for remote-request-transmit PDO only, and mode 1 is for responding all PDO message from I-87123 to users.

Set three parameters, node, COB ID, and Tx Type, to modify PDO transmission type. This function is only for Tx-PDO.

Some responses such as PDO, EMCY, or error information, will be shown on the text frame as below. Clicking the Pause button will stop to receive these response messages, and the Pause button will change to Receive button. If users want to receive these response messages, click the Receive button to do that. For example, when the CANopen DI data is changed from 0x00 to 0xFF, the message edit box has received the CANopen PDO message “COB ID is 0x181 and data is 0xFF”. Afterwards, if the DI data is changed from 0xFF to 0xAA, the message edit box will show the PDO message “COB ID is 0x181 and data is 0xAA”. If users don’t want to receive any message, users can click the Pause button to pause it.