PISO-CAN200/400

Linux SocketCAN CANopen 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 may be registered trademarks of their respective companies.

Tables of Content

1. Linux Software Installation

The PISO-CAN200/400 SocketCAN driver can be used in linux kernel 2.6.25 or later kernel 2.6.X version. For Linux O.S, the recommended installation and uninstall steps are given in Sec 1.1 ~ 1.2

1.1 Linux SocketCAN Driver Installing Procedure

Step 1: Download the linux driver “ixcan-0.2.0.tar.gz” (or the later ixcan package version) from ICP DAS webpage http://www.icpdas.com/download/pci/piso-can/index.htm to the linux host.

Step 2: You must use the ‘**root**’ identity to compile and install linux SocketCAN driver.

Step 3: Decompress the tarball “ixcan.tar.gz”.

Step 4: Type ‘**cd**' to the directory containing the package's source code and type ‘**./configure**' to configure the package for your linux system.

Step 5: Type ‘**make**' to compile the package.

Step 6: Before user install PISO-CAN200/400 SocketCAN driver module user should check the linux kernel had supported the SocketCAN driver modules (please refer to Figure 1-1, 1-2, 1-3).

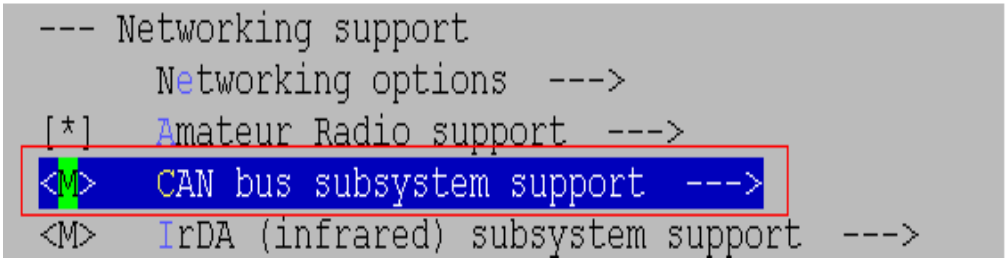

Figure 1-1

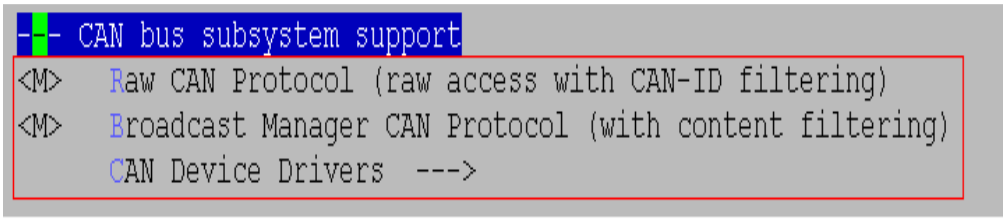

Figure 1-2

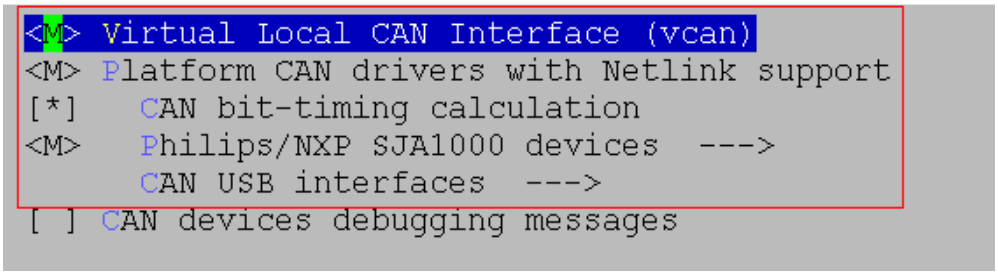

Figure 1-3

Step 7: You can type '**./ixcan.inst**' to install the PISO-CAN200/400 SocketCAN driver module and build the network device interface "canX". Please refer to the Figure 1-4(the figure show the PISO-CAN400 "canX" interface).

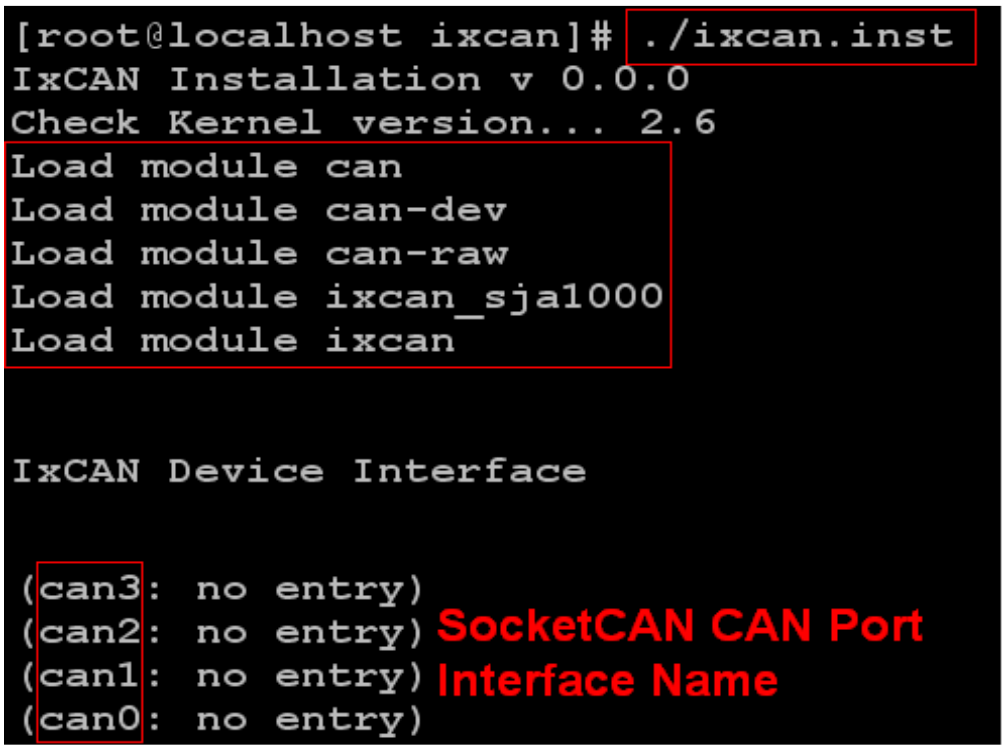

Figure 1-4

Step 8: You can type ‘dmesg' to check the number of CAN boards and channel. Please refer to the Figure 1-5 (the figure show the information of PISO-CAN400 boards).

```
CAN device driver interface
can: raw protocol (rev 20090105)
sja1000 CAN netdevice driver
icpdas-ixcan 0000:02:0a.0: PCI INT A -> GSI 22 (level, low) -> IRQ 22
icpdas-ixcan 0000:02:0a.0: Board #1 : Channel #1 at 0xe1000000, irq 22
icpdas-ixcan 0000:02:0a.0: Board #1 : Channel #2 at 0xe0db2000, irq 22
icpdas-ixcan 0000:02:0a.0: Board #1 : Channel #3 at 0xe0db6000, irq 22
icpdas-ixcan 0000:02:0a.0: Board #1 : Channel #4 at 0xe0dba000, irq 22
```

Figure 1-5

1.2 Startup and Stop SocketCAN Interface

Once the driver installed, the CAN interface has to be started and stopped like a standard net interface. Please follow the below steps to startup CAN interface:

Step 1: Use iproute2’s (version 2.6.31 or later version) command ‘ip’ to configure CAN baud rate and startup CAN interface. Please refer to below command and Figure 1-6(can2 baud rate is 1000k).

#ip link set can2 up type can bitrate 1000000

#ifconfig

```
[root@localhost ixcan]# ip link set can2 up type can bitrate 1000000
[root@localhost ixcan]# ip link set can3 up type can bitrate 1000000
[root@localhost ixcan]# ifconfig
can2      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-0
-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:22

can3      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-0
-00-00
          UP RUNNING NOARP  MTU:16  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:10
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)
          Interrupt:22
```

Figure 1-6

Step 2: Besides using ‘ip’ to startup can interface, user could use the ‘ip’ command to check can interface status. Please refer to below command and Figure 1-7.

 # ip -details link show can2

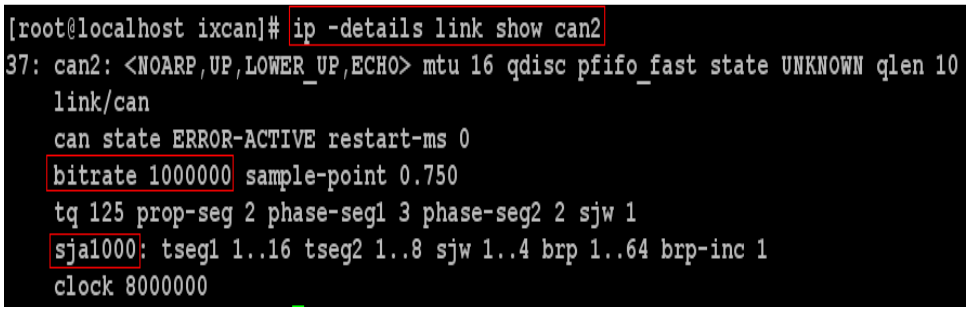

Figure 1-7

Step 3: If user want to stop can interface, user could use command ‘ip’ to stop can interface. Please refer to below command.

 # ip link set can2 down

1.3 Linux Driver Uninstalling Procedure

Step 1: Type ‘**cd**’ to the directory containing the package's source code.

Step 2: Type ‘**./ixcan.remove**' to remove the SocketCAN driver module.

2. SocketCAN CANopen Library Function Description

The SocketCAN CANopen library is the collection of function calls of the PISO-CAN200/400 cards for linux kernel 2.6.25(or later kernel version) system. The application structure is presented as following figure 2-1. The user application program developed by C(C++) language can call library “libsktcpm.a” in user mode. And then static library will call the SocketCAN modules to access the hardware system.

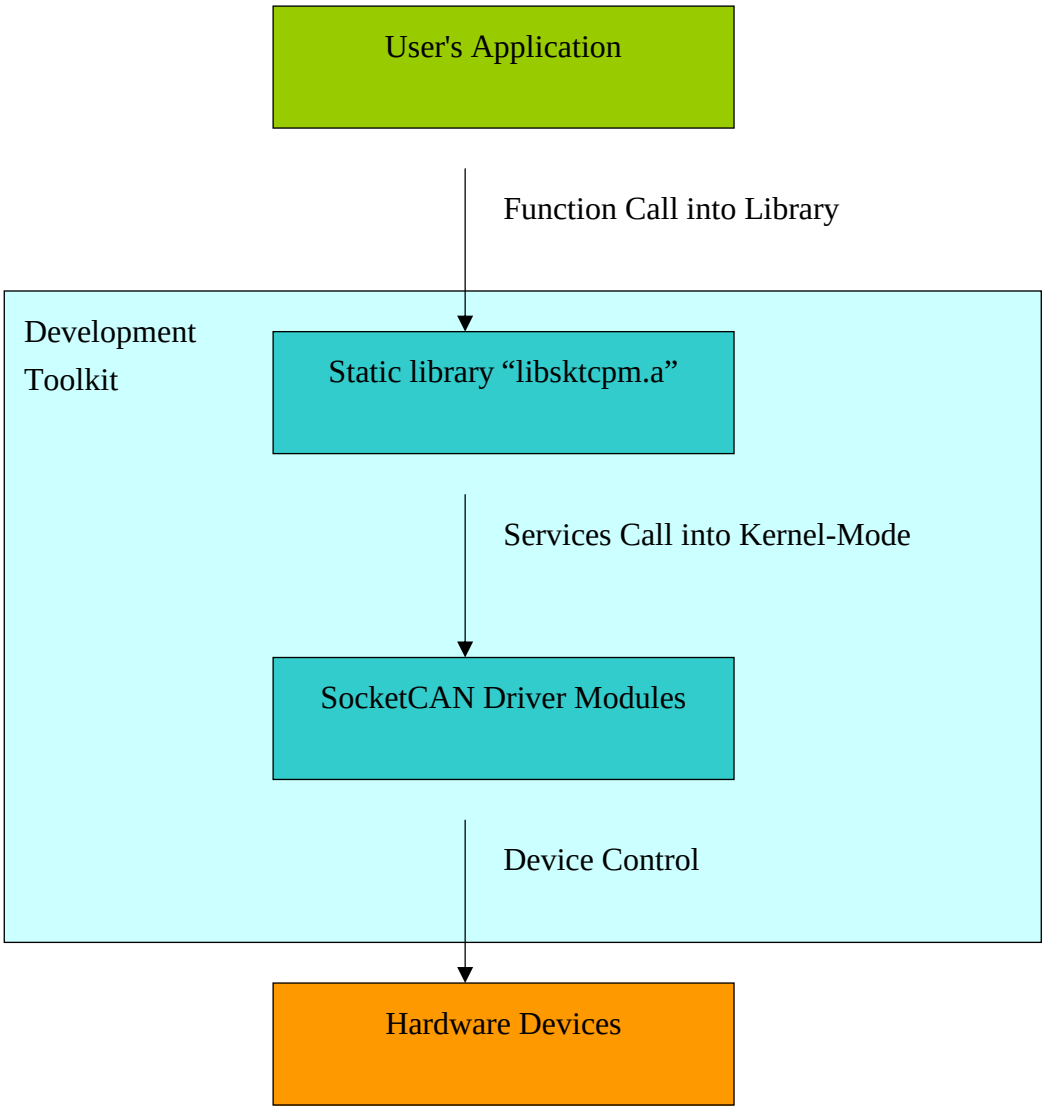

Figure 2-1

2.1 Table of Error Code and Error ID

Error Code	Error ID	Error String
0	CPM_NOERROR	OK (No error !)
1	CPM_OPEN_ERROR	Open PISO-CAN master device failure
2	CPM_CLOSE_ERROR	Close PISO-CAN master device failure
3	CPM_PORT_INIT_ERROR	Initial PISO-CAN interface failure
4	CPM_MASTER_INIT_ERROR	Initial PISO-CAN master device failure
5	CPM_NODEID_EXIST	Node ID has existed
6	CPM_NODEID_NOEXIST	Node ID isn't exist
7	CPM_NODEID_OVERRANGE	Node ID over range
8	CPM_SLAVESTATE_ERROR	The state of CANopen node error
9	CPM_COBID_EXIST	The cob id has existed
10	CPM_NODEADD_ERROR	Add CANopen node error
11	CPM_NODEREMOVE_ERROR	Remove CANopen node error
12	CPM_SENDMSG_ERROR	Send CAN message error
13	CPM_TIMEOUT	Response CAN message is timeout
14	CPM_ADDTHREAD_ERROR	Create receive or transmit data thread error
15	CPM_RESUMETHREAD_ERROR	Receive or transmit data thread resumed error
16	CPM_SUSPENDTHREAD_ERROR	Receive or transmit data thread suspended error
17	CPM_READ_SEGMENT	Upload SDO segment protocol need to be used

Error Code	Error ID	Error String
18	CPM_WRITE_SEGMENT	Download SDO segment protocol need to be used
19	CPM_READ_BLOCK	Upload SDO block protocol need to be used
20	CPM_WRITE_BLOCK	Download SDO block protocol need to be used
21	CPM_DATASIZE_RANGE_ERROR	Data size can't be 0
22	CPM_READDATA_ERROR	Upload Segment or block transmit error or slave didn't support block protocol
23	CPM_WRITEDATA_ERROR	Download Segment or block transmit error or slave didn't support block protocol
24	CPM_ENDBLOCK	Return block status
25	CPM_DATA_RESEND	Segment or block transmit failure, users need to resend the data
26	CPM_EMCYFIFO_EMPTY	No EMCY message in the buffer
27	CPM_PDOINSTALL_ERROR	Install new PDO error
28	CPM_COBID_NOTRXPDO	The COB-ID is not Rx PDO
29	CPM_COBID_NOTTXPDO	The COB-ID is not Tx PDO
30	CPM_ALLPDO_USING	All PDO objects are occupied
31	CPM_MAPPING_ENABLE_ERROR	Mapping PDO error
32	CPM_SETPDO_ERROR	PDO mapping parameters error
33	CPM_PDO_NOEXIST	PDO isn't exist
34	CPM_PDOREMOVE_ERROR	Remove PDO error
35	CPM_NORESPONSE	PDO message don't respond

Table 2.1

2.2 Function Descriptions

Function Definition
char * CPM_GetDriverVersion(void)
char * CPM_GetLibraryVersion(void)
int CPM_Open(BYTE BoardNo, BYTE Port)
WORD CPM_Close(BYTE BoardNo, BYTE Port)
WORD CPM_InitPort(BYTE BoardNo, BYTE Port, cpmchannel_t *Handle)
WORD CPM_Config(cpmchannel_t *Handle, struct can_filter *rfilter, WORD filter_size)
WORD CPM_InitMaster(cpmchannel_t *Handle, struct can_filter *rfilter, WORD filter_size, WORD SDOTimeOut)
WORD CPM_ChangeSDOTimeOut(cpmchannel_t *Handle,WORD SDOTimeOut)
WORD CPM_ShutdownMaster(cpmchannel_t *Handle)
WORD CPM_AddNode(cpmchannel_t *Handle, BYTE NodeID)
WORD CPM_RemoveNode(cpmchannel_t *Handle, BYTE NodeID)
WORD CPM_NMTGetState(cpmchannel_t *Handle, BYTE NodeID, BYTE *State)
WORD CPM_NMTChangeState(cpmchannel_t *Handle, BYTE NodeID, BYTE State)
WORD CPM_NMTGuarding(cpmchannel_t *Handle, BYTE NodeID, WORD GuardTime,BYTE LifeTimeFactor,DWORD GuardCycle)
WORD CPM_SDOReadData(cpmchannel_t *Handle, BYTE NodeID, WORD Index, BYTE SubIndex, canmsg_t *RxData, DWORD *RSize, BYTE Block)
WORD CPM_SDOReadSegment(cpmchannel_t *Handle, BYTE NodeID, canmsg_t *RxData)
WORD CPM_SDOWriteData(cpmchannel_t *Handle, BYTE NodeID, WORD Index, BYTE SubIndex, DWORD DataSize, canmsg_t *RxData, BYTE *Data, BYTE Block)

Function Definition
WORD CPM_SDOWriteSegment(cpmchannel_t *Handle, BYTE NodeID, canmsg_t *RxData, BYTE *Data)
WORD CPM_SDOReadBlock(cpmchannel_t *Handle, BYTE NodeID, BYTE *RxData)
WORD CPM_SDOWriteBlock(cpmchannel_t *Handle,BYTE NodeID,BYTE *Ackseq, BYTE *Data)
WORD CPM_SDOAbortTransmission(cpmchannel_t *Handle, BYTE NodeID)
WORD CPM_ChaneSYNCID(cpmchannel_t *Handle, BYTE NodeID, WORD CobID)
WORD CPM_SendSYNC(cpmchannel_t *Handle, DWORD Cobid, DWORD CycleTimer)
WORD CPM_ChangeEMCYID(cpmchannel_t *Handle, BYTE NodeID, WORD CobID)
WORD CPM_ReadEMCY(cpmchannel_t *Handle,canmsg_t *RData)
WORD CPM_InstallPDO(cpmchannel_t *Handle,BYTE NodeID,BYTE RxTxType,DWORD CobId,BYTE TransmitType,WORD Inhibitime,WORD EventTimer,BYTE EnableChannel)
WORD CPM_MappingPDO(cpmchannel_t *Handle,BYTE NodeID,BYTE RxTxType,DWORD CobId,BYTE Channel,BYTE *MappingData)
WORD CPM_RemovePDO(cpmchannel_t *Handle, BYTE NodeID, DWORD CobId)
WORD CPM_WritePDO(cpmchannel_t *Handle, DWORD CobId, BYTE *Data, BYTE Offset, BYTE DataLen);
WORD CPM_RemotePDO(cpmchannel_t *Handle,DWORD CobId,canmsg_t *RData)
WORD CPM_ResponsePDO(cpmchannel_t *Handle, canmsg_t *RData)
WORD CPM_ResPDOCount(cpmchannel_t *Handle)
WORD CPM_WriteDO(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, BYTE Value)
WORD CPM_WriteAO(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, WORD Value)
WORD CPM_ReadDI(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, BYTE *Value)

Function Definition
WORD CPM_ReadAI(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel,WORD *Value)
WORD CPM_ReadManufacturerName(cpmchannel_t *Handle, BYTE NodeID, BYTE Slave_dev[])

Table 2.2

2.3 SocketCAN CANopen Library FUNCTIONS

2.3.1 CPM_GetDriverVersion

- **Description:**
  To get the linux IxPCI driver version.
- **Syntax:**
  char * CPM_GetDriverVersion(void)
- **Parameter:**
  None
- **Return:**
  The ixpci linux driver version.

2.3.2 CPM_GetLibraryVersion

- **Description:**
  To get the linux CANopen master library version.
- **Syntax:**
  char * CPM_GetLibraryVersion(void)
- **Parameter:**
  None
- **Return:**
  The CANopen master library version.

2.3.3 CPM_Open

- **Description:**
  To open the device file of PISO-CAN CANopen master.

- **Syntax:**
  int CPM_Open(BYTE BoardNo, BYTE Port)
- **Parameter:**
  BoardNo: The board number (1~4) of PISO-CAN card
  Port: The port number (1~4) of PISO-CAN card
- **Return:**
  “CPM_NOERROR”
  “CPM_OPEN_ERROR”
  (Please refer to "Section 2.1 Error Code")

2.3.4 CPM_Close

- **Description :**
  To close PISO-CAN CANopen master device file.
- **Syntax :**
  WORD CPM_Close(BYTE BoardNo, BYTE Port)
- **Parameter :**
  BoardNo: The board number (1~4) of PISO-CAN card
  Port: The port number (1~4) of PISO-CAN card
- **Return:**
  “CPM_NOERROR”
  “CPM_CLOSE_ERROR”
  “CPM_OPEN_ERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.5 CPM_InitPort

- **Description :**
  To initial the configuration of PISO-CAN CANopen master port.
- **Syntax :**
  WORD CPM_Init(BYTE BoardNo, BYTE Port, cpmchannel_t *Handle)
- **Parameter :**
  BoardNo: The board number (1~4) of PISO-CAN card
  Port: The port number (1~4) of PISO-CAN card
  Handle: The pointer of structure “cpmchannel_t”. It is defined as following

  typedef struct cpmchannel

```
{
    //The board number of PISO-CAN card
    BYTE BoardNo;

    // The port number of PISO-CAN card
    BYTE Port;
} cpmchannel_t;
```

- **Return:**

  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_OPEN_ERROR”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.6 CPM_Config

- **Description :**

  To configure the PISO-CAN CANopen master port.

- **Syntax :**

  WORD CPM_Config(cpmchannel_t *Handle, struct can_filter *rfilter, WORD filter_size)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  rfilter: The point of structure for CANopen is defined as follows

```
typedef struct can_filter
{
    //relevant bits of CAN ID which are not masked out.
    DWORD can_id;

    //CAN mask
    DWORD can_mask;

} cpmconfig_t;
```

  filter_size: The rfilter size.

- **Return:**

  “CPM_OPEN_ERROR”

“CPM_PORT_INIT_ERROR”
“CPM_NOERROR”
(Please refer to "Section 2.1 Error Code").

2.3.7 CPM_InitMaster

- **Description :**
  To initial CANopen master device.
- **Syntax :**
  WORD CPM_InitMaster(cpmchannel_t *Handle, struct can_filter *rfilter, WORD filter_size, WORD SDOTimeOut)
- **Parameter :**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  rfilter: The point of structure for CANopen is defined as follows

```
typedef struct can_filter
{
   //relevant bits of CAN ID which are not masked out.
   DWORD can_id;

   //CAN mask
   DWORD can_mask;

} cpmconfig_t;
```

  filter_size: The rfilter size.
  SDOTimeOut: The SDO timeout value.
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_ADDTHREAD_ERROR”
  “CPM_RESUMETHREAD_ERROR”
  “CPM_INITMASTER_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_NODEADD_ERROR”
  “CPM_COBID_EXIST”

“CAN_NOERROR”
(Please refer to "Section 2.1 Error Code").

2.3.8 CPM_ChangeSDOTimeOut

- **Description :**
  To change SDO timeout value.
- **Syntax :**
  WORD CPM_ChangeSDOTimeOut(cpmchannel_t *Handle,WORD SDOTimeOut)
- **Parameter :**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  SDOTimeOut: The new SDO timeout value.
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.9 CPM_ShutdownMaster

- **Description :**
  To shutdown CANopen master.
- **Syntax :**
  WORD CPM_ShutdownMaster(cpmchannel_t *Handle)
- **Parameter :**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.10 CPM_AddNode

- **Description :**

To add CANopen slave with node id into CANopen master node list.

- **Syntax :**

  WORD CPM_AddNode(cpmchannel_t *Handle, BYTE NodeID)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node id (1~127).

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_NODEADD_ERROR”
  “CPM_TIMEOUT”
  “CPM_COBID_EXIST”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.11 CPM_RemoveNode

- **Description :**

  To remove the slave node id from CANopen master node list.

- **Syntax :**

  WORD CPM_RemoveNode(cpmchannel_t *Handle, BYTE NodeID)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node (1~127).

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”,
  “CPM_NODEID_EXIST”

“CPM_NODEREMOVE_ERROR”
“CPM_NODEID_NOEXIST”
“CPM_NOERROR”
(Please refer to "Section 2.1 Error Code").

2.3.12 CPM_NMTGetState

- **Description :**
  To get the NMT state from CANopen slave.
- **Syntax :**
  WORD CPM_NMTGetState(cpmchannel_t *Handle, BYTE NodeID, BYTE *State)
- **Parameter :**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: The CANopen slave node (1~127).
  State: The state of slave.
  State = 4 is “Stop Mode”
  State = 5 is “Operational Mode”
  State = 127 is “Pre-Operational Mode”
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.13 CPM_NMTChangeState

- **Description :**
  To be used to change the NMT state of a slave.
- **Syntax :**
  WORD CPM_NMTChangeState(cpmchannel_t *Handle, BYTE NodeID,

BYTE State)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node (1~127).

  State: The state of slave.

  State = 1 is “Operational Mode”

  State = 2 is “Stop Mode”

  State = 128 is “Pre-Operational Mode”

  State = 129 is “Reset Slave”

  State = 130 is “Rest Communication”

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_NODEID_OVERRANGE”

  “CPM_SLAVESTATE_ERROR”

  “CPM_NODEID_EXIST”

  “CPM_SENDMSG_ERROR”

  “CPM_NODEREMOVE_ERROR”

  “CPM_NODEID_NOEXIST”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code").

2.3.14 CPM_NMTGuarding

- **Description :**

  To set “Guard Time” and “Life Time Factor” of the specific slave with node ID.

- **Syntax :**

  WORD CPM_NMTGuarding(cpmchannel_t *Handle, BYTE NodeID, WORD GuardTime,BYTE LifeTimeFactor,DWORD GuardCycle)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node (1~127).

  GuardTime: Guard Time (1 ~ 65535).

  LifeTimeFactor: Life Time Factor (1 ~ 255).

GuardCycle: To send the Guard message from master.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_ADDTHREAD_ERROR”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.15 CPM_SDOReadData

- **Description :**

  The function CPM_SDOReadData is useful to the SDO upload from a specific slave.

- **Syntax :**

  WORD CPM_SDOReadData(cpmchannel_t *Handle, BYTE NodeID, WORD Index, BYTE SubIndex, canmsg_t *RxData, DWORD *RSize, BYTE Block)

- **Parameter :**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: The CANopen slave node (1~127).
  Index: SDO index in the object dictionary.
  SubIndex: SDO subindex in the object dictionary.
  RxData: SDO data respond from the specific slave device.
  RSize: Total data size.
  Block: Using SDO Block protocol.
  Block = 0 isn’t using SDO Block protocol.
  Block = 1 use SDO Block protocol.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”

“CPM_SLAVESTATE_ERROR”
“CPM_NODEID_EXIST”
“CPM_SENDMSG_ERROR”
“CPM_TIMEOUT”
“CPM_READ_SEGMENT”
“CPM_READ_BLOCK”
“CPM_NOERROR”
(Please refer to "Section 2.1 Error Code").

2.3.16 CPM_SDOReadSegment

- **Description:**

  If SDO data size is more than 7 bytes or get the code “CPM_READ_SEGMENT” from CPM_SDOReadData, user should use the function “CPM_SDOReadSegment” to read SDO data.

- **Syntax:**

  WORD CPM_SDOReadSegment(cpmchannel_t *Handle, BYTE NodeID, canmsg_t *RxData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: The CANopen slave node (1~127).
  RxData: Segment data uploaded from the slave device

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_READ_SEGMENT”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.17 CPM_SDOReadBlock

- **Description:**

If users send or receive the SDO message whose data size is more then 127 segments (127 * 7 bytes) or get code “CPM_READ_BLOCK”, user should use the function “CPM_SDOReadBlock” to read SDO data.

- **Syntax:**

  WORD CPM_SDOReadBlock(cpmchannel_t *Handle, BYTE NodeID, BYTE *RxData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node (1~127).

  RxData: At most 127 * 7 bytes of block data to be uploaded.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_READDATA_ERROR”
  “CPM_READ_BLOCK”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code")

2.3.18 CPM_SDOWriteData

- **Description:**

  The function can send out a SDO message to specific salve device. This procedure is also called download SDO message.

- **Syntax:**

  WORD CPM_SDOWriteData(cpmchannel_t *Handle, BYTE NodeID, WORD Index, BYTE SubIndex, DWORD DataSize, canmsg_t *RxData, BYTE *Data, BYTE Block)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: The CANopen slave node (1~127).

Index: SDO index in the object dictionary.
SubIndex: SDO subindex in the object dictionary.
DataSize: Total data size to be written.
RxData: SDO data respond from the specific slave device.
Data: The SDO Data which will be downloaded. (This parameter is useless when the parameter DataSize is over than 4 or the parameter Block is set to1).
Block: SDO download mode.
Block = 0: If the total data size of the object dictionary's object does not exceed 4 bytes. The function CPM_SDOWriteData will return the value CPM_Noerror. Otherwise, it will return the value CPM_WriteSegment.
Block = 1: The SDO Block download protocol is used and the function CPM_SDOWriteData returns the value CPM_WriteBlock.

- **Return:**

"CPM_OPEN_ERROR"
"CPM_PORT_INIT_ERROR"
"CPM_MASTER_INIT_ERROR"
"CPM_NODEID_OVERRANGE"
"CPM_SLAVESTATE_ERROR"
"CPM_NODEID_EXIST"
"CPM_SENDMSG_ERROR"
"CPM_TIMEOUT"
"CPM_DATASIZE_RANGE_ERROR"
"CPM_WRITE_SEGMENT"
"CPM_WRITE_BLOCK"
"CPM_NOERROR"
(Please refer to "Section 2.1 Error Code")

2.3.19 CPM_SDOWriteSegment

- **Description:**

When the function CPM_SDOWriteData returns the value "CPM_WRITE_SEGMENT", the function CPM_SDOWriteSegment must be called to continue the SDO segment download protocol. Afterwards, if the function CPM_SDOWriteSegment still returns the value "CPM_WRITE_SEGMENT", the function CPM_SDOWriteSegment must be used again.

- **Syntax:**

WORD CPM_SDOWriteSegment(cpmchannel_t *Handle, BYTE NodeID, canmsg_t *RxData, BYTE *Data)

- **Parameter:**

Handle: CAN channel handle pointer returned from the function CPM_InitPort.

NodeID: Slave device Node-ID (1~127).

RxData: SDO segment data responded from the slave device.

Data: The SDO Data which will be downloaded. The SDO data length can't exceed one segment.

- **Return:**

"CPM_OPEN_ERROR"
"CPM_PORT_INIT_ERROR"
"CPM_MASTER_INIT_ERROR"
"CPM_NODEID_OVERRANGE"
"CPM_SLAVESTATE_ERROR"
"CPM_NODEID_EXIST"
"CPM_SENDMSG_ERROR"
"CPM_TIMEOUT"
"CPM_WRITE_SEGMENT"
"CPM_NOERROR"
(Please refer to "Section 2.1 Error Code")

2.3.20 CPM_SDOWriteBlock

- **Description:**

When the parameter Block of the function CPM_SDOWriteData is set to 1, the function CPM_SDOWriteData will return the value "CPM_WRITE_BLOCK", and the function CPM_SDOWriteBlock must be used to continue the SDO block download protocol. If the data length of download SDO data is more then one block size defined by the function CMP_SDOWriteData, this function must be called again.

- **Syntax:**

WORD CPM_SDOWriteBlock(cpmchannel_t *Handle, BYTE NodeID, BYTE *Ackseq, BYTE *Data)

- **Parameter:**

Handle: CAN channel handle pointer returned from the function CPM_InitPort.

NodeID: Slave device Node-ID (1~127).

Ackseq: Last segment sequence number that was received successfully during the last block download.

Data: The SDO Data which will be downloaded. The SDO data length can't exceed one block.

- **Return:**

"CPM_OPEN_ERROR"
"CPM_PORT_INIT_ERROR"
"CPM_MASTER_INIT_ERROR"
"CPM_NODEID_OVERRANGE"
"CPM_SLAVESTATE_ERROR"
"CPM_NODEID_EXIST"
"CPM_SENDMSG_ERROR"
"CPM_TIMEOUT"
"CPM_DATA_RESEND "
"CPM_WRITE_BLOCK"
"CPM_WRITEDATA_ERROR"
"CPM_NOERROR"
(Please refer to "Section 2.1 Error Code")

2.3.21 CPM_SDOAbortTransmission

- **Description:**

To cancel the SDO transmission.

- **Syntax:**

WORD CPM_SDOAbortTransmission(cpmchannel_t *Handle, BYTE NodeID)

- **Parameter:**

Handle: CAN channel handle pointer returned from the function CPM_InitPort.

NodeID: Slave device Node-ID (1~127).

- **Return:**

"CPM_OPEN_ERROR"
"CPM_PORT_INIT_ERROR"
"CPM_MASTER_INIT_ERROR"
"CPM_NODEID_OVERRANGE"
"CPM_SLAVESTATE_ERROR"
"CPM_NODEID_EXIST"
"CPM_SENDMSG_ERROR"

“CPM_NOERROR”

(Please refer to "Section 2.1 Error Code")

2.3.22 CPM_InstallPDO

- **Description:**
  To create a new PDO object in the CANopen Master Library stack.
- **Syntax:**
  WORD CPM_InstallPDO(cpmchannel_t *Handle,BYTE NodeID,BYTE RxTxType,DWORD CobId,BYTE TransmitType,WORD Inhibitime,WORD EventTimer,BYTE EnableChannel)
- **Parameter:**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: Salve device Node-ID (1~127).
  TxRxType: PDO type (0: for receive PDO, 1: for transmit PDO).
  CobId: The COB-ID is used by the PDO object.
  TransmitType: PDO transmission type (0 ~ 255).
  Inhibitime: PDO inhibit time (0 ~ 65535 ms)(not used for RPDO).
  EventTimer: PDO event timer (0 ~ 65535 ms).
  EnableChannel: Number of mapped application objects in PDO (0 ~ 8)
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_PDOINSTALL_ERROR”
  “CPM_COBID_NOTRXPDO”
  “CPM_COBID_NOTTXPDO”
  “CPM_ALLPDO_USING”
  “CPM_MAPPING_ENABLE_ERROR”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code")

2.3.23 CPM_MappingPDO

- **Description:**

  This function can set the PDO mapping objects. If users use this function to define the PDO mapping object with the PDO COB-ID that is not installed by using the function CPM_InstallPDO, this PDO mapping object will be useless.

- **Syntax:**

  WORD CPM_MappingPDO(cpmchannel_t *Handle,BYTE NodeID,BYTE RxTxType,DWORD CobId,BYTE Channel,BYTE *MappingData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: Slave device Node-ID (1~127).

  TxRxType: Receive or Transmit PDO parameter (0: Rx, 1: Tx).

  CobId: COB-ID is used by the PDO object.

  Channel: PDO mapping for the nth application object to be mapped (at most is 8).

  MappingData: 4 bytes data of mapped.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_SETPDO_ERROR”
  “CPM_ALLPDO_USING”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code")

2.3.24 CPM_RemovePDO

- **Description:**

  To remove a TxPDO or RxPDO installed by the CPM_InstallPDO.

- **Syntax:**

WORD CPM_RemovePDO(cpmchannel_t *Handle, BYTE NodeID, DWORD CobId)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: Slave device Node-ID (1~127).

  CobId: COB-ID is used by the PDO object.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_NODEID_OVERRANGE”

  “CPM_SLAVESTATE_ERROR”

  “CPM_NODEID_EXIST”

  “CPM_SENDMSG_ERROR”

  “CPM_TIMEOUT”

  “CPM_PDO_NOEXIST”

  “CPM_PDOREMOVE_ERROR”

  “CPM_SETPDO_ERROR”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.25 CPM_WritePDO

- **Description:**

  To send a PDO message to the specific salve device.

- **Syntax:**

  WORD CPM_WritePDO(cpmchannel_t *Handle, DWORD CobId, BYTE *Data, BYTE Offset, BYTE DataLen)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  CobId: COB-ID is used by the PDO object.

  Data: Data pointer to point the PDO data.

  Offset: The first PDO data byte position (0 ~ 7).

  DataLen: data size of PDO Data (DataLen + Offset $\leqq$ 8).

- **Return:**

  “CPM_OPEN_ERROR”

“CPM_PORT_INIT_ERROR”
“CPM_MASTER_INIT_ERROR”
“CPM_NODEID_OVERRANGE”
“CPM_SLAVESTATE_ERROR”
“CPM_NODEID_EXIST”
“CPM_SENDMSG_ERROR”
“CPM_TIMEOUT”
“CPM_PDO_NOEXIST”
“CPM_NOERROR”
(Please refer to "Section 2.1 Error Code")

2.3.26 CPM_RemotePDO

- **Description:**

  To send a remote request PDO message to the slave device.

- **Syntax:**

  WORD CPM_RemotePDO(cpmchannel_t *Handle,DWORD CobId,canmsg_t *RData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  CobId: COB-ID is used by the PDO object.

  RData: The PDO message received from the remote salve device.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_PDO_NOEXIST”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code")

2.3.27 CPM_ResponsePDO

- **Description:**

To read the PDO message which is received from the slave device and stored in the PDO software buffer.

- **Syntax:**

  WORD CPM_ResponsePDO(cpmchannel_t *Handle, canmsg_t *RData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  RData: The PDO data responded from the slave device.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_NORESPONSE”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.28 CPM_ResPDOCount

- **Description:**

  To get the message count of PDO message from PDO buffer.

- **Syntax:**

  WORD CPM_ResPDOCount(cpmchannel_t *Handle)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.29 CPM_SendSYNC

- **Description:**

  To send a SYNC message with specific COB-ID cyclically

- **Syntax:**

  WORD CPM_SendSYNC(cpmchannel_t *Handle, DWORD Cobid, DWORD CycleTimer)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  CobId: COB-ID is used by the SYNC object.

  CycleTimer: SYNC message transmission period.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.30 CPM_ReadEMCY

- **Description:**

  The function CPM_ReadEMCY can return the emergency message stored in the buffer.

- **Syntax:**

  WORD CPM_ReadEMCY(cpmchannel_t *Handle,canmsg_t *RData)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  RData: Emergency message data stored in the buffer.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_EMCYFIFO_EMPTY”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.31 CPM_WriteDO

- **Description:**

  To output index 0x6200 with SDO protocol.

- **Syntax:**

  WORD CPM_WriteDO(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, BYTE Value)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: Slave device Node-ID (1~127).

  Channel: Output DO channel. The parameter is from 1.

  Value: Output data.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

  “CPM_NODEID_OVERRANGE”

  “CPM_SLAVESTATE_ERROR”

  “CPM_NODEID_EXIST”

  “CPM_SENDMSG_ERROR”

  “CPM_TIMEOUT”

  “CPM_DATASIZE_RANGE_ERROR”

  “CPM_WRITE_SEGMENT”

  “CPM_WRITE_BLOCK”

  “CPM_NOERROR”

  (Please refer to "Section 2.1 Error Code")

2.3.32 CPM_WriteAO

- **Description:**

  To output index 0x6411 with SDO protocol.

- **Syntax:**

  WORD CPM_WriteAO(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, WORD Value)

- **Parameter:**

  Handle: CAN channel handle pointer returned from the function CPM_InitPort.

  NodeID: Slave device Node-ID (1~127).

  Channel: Output AO channel. The parameter is from 1.

  Value: Output data.

- **Return:**

  “CPM_OPEN_ERROR”

  “CPM_PORT_INIT_ERROR”

  “CPM_MASTER_INIT_ERROR”

“CPM_NODEID_OVERRANGE”
“CPM_SLAVESTATE_ERROR”
“CPM_NODEID_EXIST”
“CPM_SENDMSG_ERROR”
“CPM_TIMEOUT”
“CPM_DATASIZE_RANGE_ERROR”
“CPM_WRITE_SEGMENT”
“CPM_WRITE_BLOCK”
“CPM_NOERROR”
(Please refer to "Section 2.1 Error Code")

2.3.33 CPM_ReadDI

- **Description:**
  To read index “0x6000” with SDO protocol.
- **Syntax:**
  WORD CPM_ReadDI(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel, BYTE *Value)
- **Parameter:**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: Slave device Node-ID (1~127).
  Channel: Read DI channel. The parameter is from 1.
  Value: Input data.
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_READ_SEGMENT”
  “CPM_READ_BLOCK”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.34 CPM_ReadAI

- **Description:**
  To read index “0x6000” with SDO protocol.
- **Syntax:**
  WORD CPM_ReadAI(cpmchannel_t *Handle, BYTE NodeID, BYTE Channel,WORD *Value)
- **Parameter:**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: Slave device Node-ID (1~127).
  Channel: Read DI channel. The parameter is from 1.
  Value: Input data.
- **Return:**
  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_READ_SEGMENT”
  “CPM_READ_BLOCK”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

2.3.35 CPM_ReadManufactureName

- **Description:**
  To read the name of CANopen slave device.
- **Syntax:**
  WORD CPM_ReadManufacturerName(cpmchannel_t *Handle, BYTE NodeID, BYTE slave_dev[])
- **Parameter:**
  Handle: CAN channel handle pointer returned from the function CPM_InitPort.
  NodeID: Slave device Node-ID (1~127).

Slave_dev: The slave device name received from Remote slave.

- **Return:**

  “CPM_OPEN_ERROR”
  “CPM_PORT_INIT_ERROR”
  “CPM_MASTER_INIT_ERROR”
  “CPM_NODEID_OVERRANGE”
  “CPM_SLAVESTATE_ERROR”
  “CPM_NODEID_EXIST”
  “CPM_SENDMSG_ERROR”
  “CPM_TIMEOUT”
  “CPM_READ_SEGMENT”
  “CPM_READ_BLOCK”
  “CPM_NOERROR”
  (Please refer to "Section 2.1 Error Code").

3. SocketCAN CANopen Demo For Linux

All of demo programs will not work normally if PISO-CAN200/400 SocketCAN driver would not be installed correctly. During the installation process, the install-scripts “ixcan.inst” will setup the correct SocketCAN driver. After driver (version 0.2.0 or the later driver version) compiled and installation, the related CANopen library, demo and header files for different development environments are presented as follows.

Table 3.1

Driver Name	Directory Path	File Name	Description
ixcan-0.2.0	include	sja1000.h	SocketCAN driver header
		pisocpm.h	The Linux SocketCAN CANopen library header
	lib	libsktcpm.a	The static library of CANopen.
	examples/ pisocan200_400	canopen_a.c	CANopen library Demo.

3.1 Demo code “canopen.c”

Using the CANopen master library function to provide under SDO, PDO, NMT CANopen communication application.

NMT Services: CPM_NMTChangeState, CPM_NMTGuarding

SDO Services: CPM_SDOReadData, CPM_SDOReadSegment,

CPM_SDOWriteData, CPM_SDOWriteSegment

PDO Services: CPM_InstallPDO, CPM_MappingPDO,

CPM_RemovePDO, CPM_WritePDO,

CPM_RemotePDO