PISO-CAN200/400-D/T

User 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 Contents

1. Linux Software Installation

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

1.1 Linux Driver Installing Procedure

Step 1: Copy the linux driver “ixpci.tar.gz”(version 0.7.5 or later version) to the linux host that you want to install driver.

Step 2: Decompress the tarball “ixpci.tar.gz”.

Step 3: Type `cd' to the directory containing the package's source code and type ./configure' to configure the package for your system.

Step 4: Type `make' to compile the package.

Step 5: Type `./ixpci.inst' to install the PCI driver module and build the device file “ixpci-canX” in the device directory “/dev” automatically. Please refer to the Figure 1.1

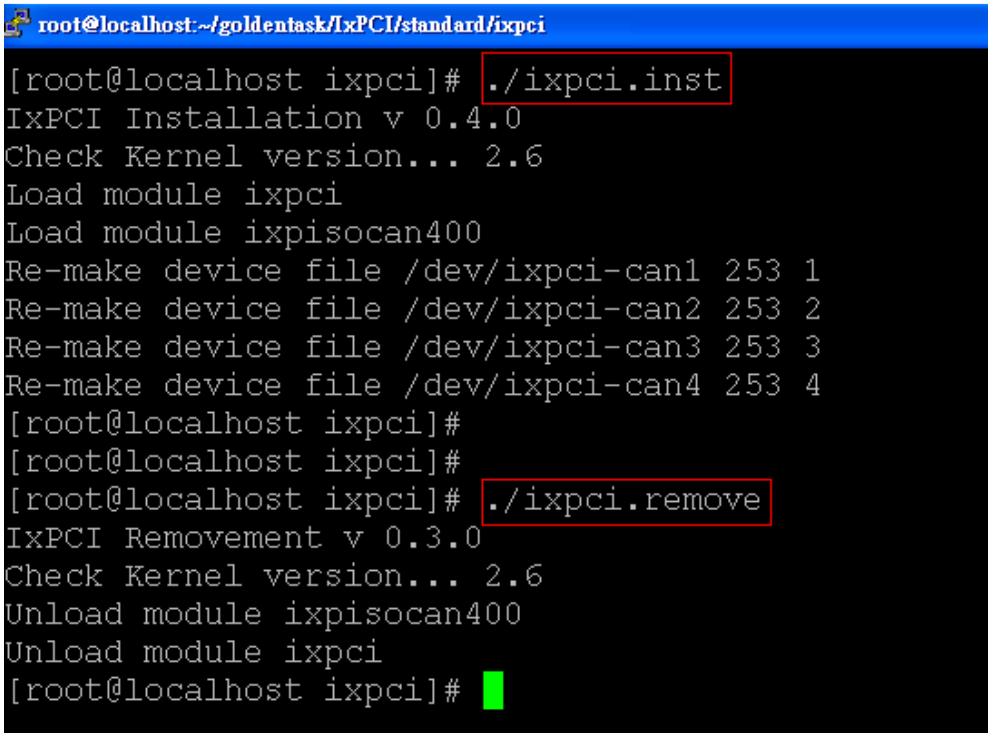

Figure 1-1

1.2 Linux Driver Uninstalling Procedure

Step 1: Type `cd' to the directory containing the package's source code.

Step 2: Type `./ixpci.remove' to remove the PCI driver module. Please refer to the Figure 1.1

2. Static Library Function Description

The static library is the collection of function calls of the PISO-CAN200/400 cards for linux kernel 2.6.x system. The application structure is presented as following figure. The user application program developed by C(C++) language can call library “libcan.a” in user mode. And then static library will call the module ixpisocan200/ ixpisocan400 to access the hardware system.

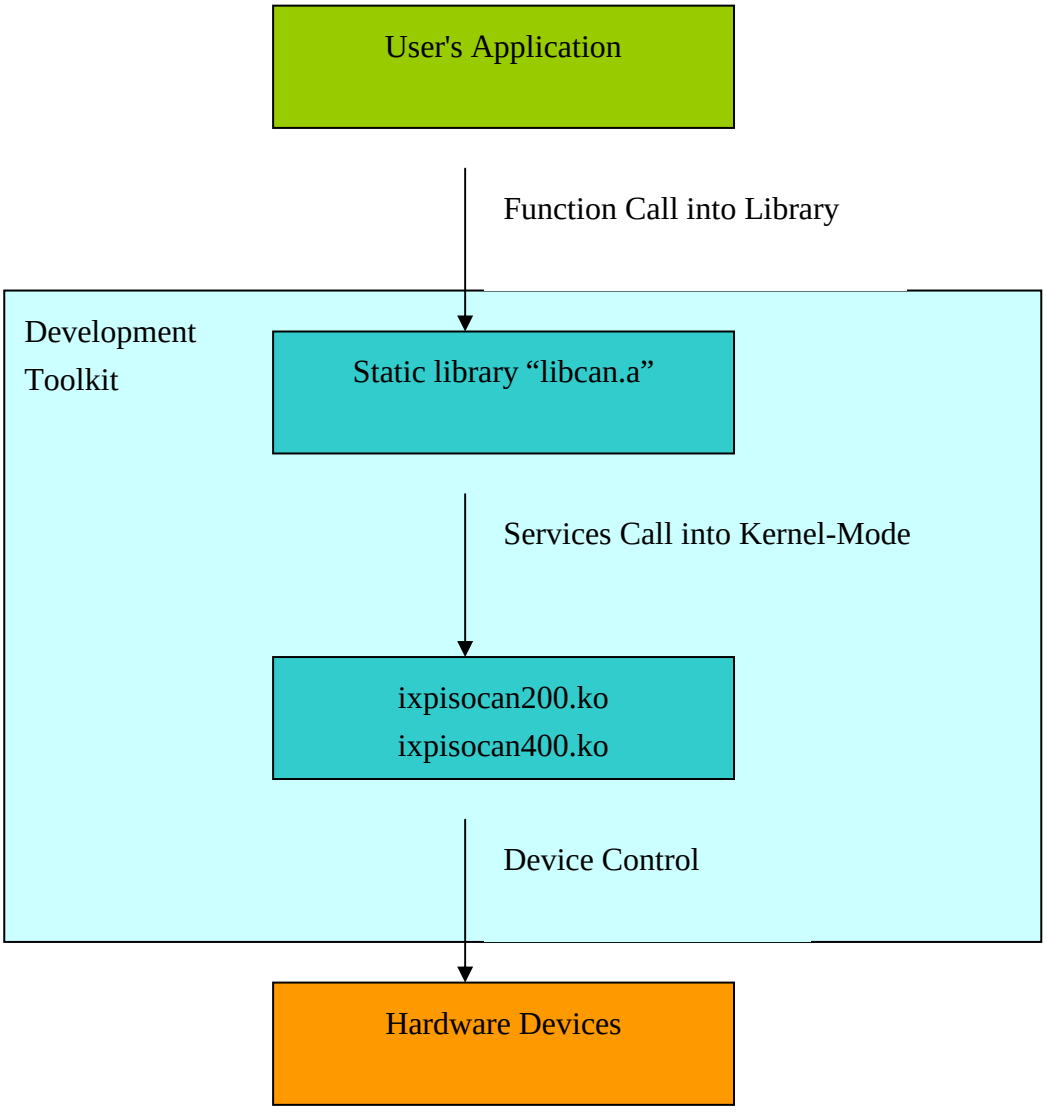

Figure 2.1

2.1 Table of Error Code and Error ID

Error Code	Error ID	Error String
0	CAN_NOERROR	OK (No error !)
1	CAN_OPEN_ERROR	Open PISO-CAN200/400 card device failure
2	CAN_CLOSE_ERROR	Close PISO-CAN200/400 card device failure
3	CAN_INIT_ERROR	Initial PISO-CAN200/400 card device failure
4	CAN_CONFIG_ERROR	Configure the can port of CAN200/400 failure
5	CAN_ENABLERXIRQ_ERROR	Enable CAN200/400 RX IRQ failure
6	CAN_DISABLERXIRQ_ERROR	Disable CAN200/400 RX IRQ failure
7	CAN_ENABLETXIRQ_ERROR	Enable CAN200/400 TX IRQ failure
8	CAN_DISABLETXIRQ_ERROR	Disable CAN200/400 TX IRQ failure
9	CAN_SENDMSG_WITHIRQ_ERROR	Send CAN message with TX interrupt failure
10	CAN_SENDMSG_WITHOUTIRQ_ERROR	Send CAN message with ioctl command failure
11	CAN_RECEIVEMSG_WITHIRQ_ERROR	Receive CAN message with RX interrupt failure
12	CAN_RECEIVEMSG_WITHOUTIRQ_ERROR	Receive CAN message with ioctl command failure
13	CAN_CLEARDATA_OVERRUN_ERROR	Clear CAN data overrun failure
14	CAN_IOCTL_COMMAND_ERROR	The Ioctl command error

Table 2.1

2.2 Function Descriptions

Function Definition
char * CAN_GetDriverVersion(void)
char * CAN_GetLibraryVersion(void)
int CAN_Open(char *)
WORD CAN_Close(int)
WORD CAN_Init(int)
WORD CAN_Config(int, DWORD, DWORD, DWORD)
WORD CAN_EnableRxIrq(int)
WORD CAN_DisableRxIrq(int)
WORD CAN_EnableTxIrq(int)
WORD CAN_DisableTxIrq(int)
WORD CAN_SendMsgWithIrq(int, canmsg_t [], DWORD)
WORD CAN_SendMsgWithoutIrq(int, ixpci_canreg_t *)
WORD CAN_ReceiveMsgWithIrq(int, canmsg_t [], DWORD)
WORD CAN_ReceiveMsgWithoutIrq(int, ixpci_canreg_t *)
WORD CAN_GetStatus(int, CanStatusPar_t *)
WORD CAN_RxIrqStatus(int)
WORD CAN_IrqStatus(int)
WORD CAN_ClearDataOverrun(int)
WORD CAN_RxMsgCount(int)

Table 2.2

2.3 PISO-CAN200/400 FUNCTIONS

2.3.1 CAN_GetDriverVersion

- **Description:**
  To get the linux version.
- **Syntax:**
  char * CAN_GetDriverVersion(Void)
- **Parameter:**
  None

- **Return:**
  The linux driver version.

2.3.2 CAN_GetLibraryVersion

- **Description:**
  To get the linux CAN library version.
- **Syntax:**
  WORD PIODIO_GetLibraryVersion(void)
- **Parameter:**
  None
- **Return:**
  The CAN library version.

2.3.3 CAN_Open

- **Description:**
  To open the device file of PISO-CAN200/400.
- **Syntax:**
  int CAN_Open(char *dev_file)
- **Parameter:**
  dev_file : The path of device file
- **Return:**
  The file descriptor of device file. If the file descriptor < 0, it means that open device file failure.

2.3.4 CAN_Close

- **Description :**
  To close PISO-CAN200/400 device file.
- **Syntax :**
  WORD CAN_Close(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open
- **Return:**
  The code “CAN_NOERROR” or “CAN_CLOSE_ERROR”(Please refer

to "Section 2.1 Error Code").

2.3.5 CAN_Init

- **Description :**
  To initial the configuration of PISO-CAN200/400 port.
- **Syntax :**
  WORD CAN_Init(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open
- **Return:**
  The code “CAN_INIT_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.6 CAN_Config

- **Description :**
  To configure the PISO-CAN200/400 port.
- **Syntax :**
  WORD CAN_Config(int fd, DWORD baud, DWORD acr, DWORD amr)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
  baud : 10(Kbps),
  20(Kbps),
  50(Kbps),
  100(Kbps),
  125(Kbps),
  250(Kbps, 250Kbps is default value),
  500(Kbps),
  800(Kbps),
  1000(Kbps)
  acr : The acceptance code for CAN controller.
  amr : The acceptance code for CAN controller.
- **Return:**
  The code “CAN_CONFIG_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.7 CAN_EnableRxIrq

- **Description :**
  To enable the RX interrupt of PISO-CAN200/400 port.
- **Syntax :**
  WORD CAN_EnableRxIrq(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  The code “CAN_ENABLERXIRQ_ERROR” or
  “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.8 CAN_DisableRxIrq

- **Description :**
  To disable the RX interrupt of PISO-CAN200/400 port.
- **Syntax :**
  WORD CAN_DisableRxIrq(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  The code “CAN_DISABLERXIRQ_ERROR” or
  “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.9 CAN_EnableTxIrq

- **Description :**
  To enable the TX interrupt of PISO-CAN200/400 port.
- **Syntax :**
  WORD CAN_EnableTxIrq(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  The code “CAN_ENABLETXIRQ_ERROR” or
  “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.10 CAN_DisableTxIrq

- **Description :**
  To disable the TX interrupt of PISO-CAN200/400 port.

- **Syntax :**
  WORD CAN_DisableTxIrq(int fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  The code “CAN_DISABLETXIRQ_ERROR” or
  “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.11 CAN_SendMsgWithIrq

- **Description :**
  To send CAN message with interrupt.
- **Syntax :**
  WORD CAN_SendMsgWithIrq(int fd, canmsg_t tx[], DWORD send_count)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
  tx[] : The point of structure for CAN message is defined as follows

```
typedef struct
{
    /* flags : defined data frame information */
    unsigned char flags;

    /* id : CAN message identifier
    unsigned long id;

    /* length : data length
    unsigned char  length;

    /* data : data byte */
    unsigned char data[CAN_MSG_LENGTH];
} canmsg_t;
```

send_count : the transmitting time of CAN message

- **Return:**
  The code “CAN_SENDMSG_WITHIRQ_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.12 CAN_SendMsgWithoutIrq

- **Description :**
  To send CAN message with ioctl command.
- **Syntax :**
  WORD CAN_SendMsgWithoutIrq(int fd, ixpci_canreg_t * canconf)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
  canconf : The point of structure for CAN register is defined as follows

```
typedef struct ixpci_canreg
{
    /* ioctl command id */
    unsigned int id;

    /* save CAN register value */
    unsigned int value;

    /* CAN status */
    CanStatusPar_t stat;

    /* CAN TX message */
    canmsg_t tx;

    /* CAN RX message */
    canmsg_t rx;

    /* the configure of  CAN controller */
    unsigned int baud;
    unsigned int acr;
    unsigned int amr;
} ixpci_canreg_t;
```

- **Return:**
  The code “CAN_SENDMSG_WITHOUTIRQ_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.13 CAN_ReceiveMsgWithIrq

- **Description :**
  To receive CAN message with interrupt.
- **Syntax :**
  WORD CAN_ReceiveMsgWithIrq(int fd, canmsg_t rx[], DWORD receive_count)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
  rx[] : The point of structure for CAN message is defined as follows

```
typedef struct
{
    /* flags : defined data frame information */
    unsigned char flags;

    /* id : CAN message identifier
    unsigned long id;

    /* length : data length
    unsigned char  length;

    /* data : data byte */
    unsigned char data[CAN_MSG_LENGTH];
} canmsg_t;
```

  receive_count : the receiving time of CAN message
- **Return:**
  The code “CAN_RECEIVEMSG_WITHIRQ_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.14 CAN_ReceiveMsgWithoutIrq

- **Description :**
  To receive CAN message with ioctl command.
- **Syntax :**
  WORD CAN_ReceiveMsgWithoutIrq(int fd, ixpci_canreg_t * canconf)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.

canconf : The point of structure for CAN register is defined as follows

```
typedef struct ixpci_canreg
{

    /* ioctl command id */
    unsigned int id;

    /* save CAN register value */
    unsigned int value;

    /* CAN status */
    CanStatusPar_t stat;

    /* CAN TX message */
    canmsg_t tx;

    /* CAN RX message */
    canmsg_t rx;

    /* the configure of save CAN controller */
    unsigned int baud;
    unsigned int acr;
    unsigned int amr;
} ixpci_canreg_t;
```

- **Return:**
  The code “CAN_RECEIVEMSG_WITHOUTIRQ_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.15 CAN_GetStatus

- **Description :**
  To get PISO-CAN200/400 port status.
- **Syntax :**
  WORD CAN_GetStatus(int fd, CanStatusPar_t * status)
- **Parameter :**
  fd : The file descriptor of device file that get from function CAN_Open.
  status : The point of structure for CAN status is defined as follows

```
typedef struct CanStatusPar
{
        /* actual bit rate */

        unsigned int baud;

         /* CAN controller status register */
        unsigned int status;

        /* the error warning limit */
        unsigned int error_warning_limit;

        /* content of RX error counter */
        unsigned int rx_errors;

        /* content of TX error counter */
        unsigned int tx_errors;

        /* content of error code register */
        unsigned int error_code;

        /* size of rx buffer  */
        unsigned int rx_buffer_size;

        /* number of messages */
        unsigned int rx_buffer_used;

        /* size of tx buffer  */
        unsigned int tx_buffer_size;

        /* number of messages */
        unsigned int tx_buffer_used;

        /* CAN controller type /
        unsigned int type;
} CanStatusPar_t;
```

- **Return:**

The code “CAN_IOCTL_COMMAND_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.16 CAN_RxIrqStatus

- **Description:**
  To get the status of RX IRQ.
- **Syntax:**
  WORD CAN_RxIrqStatus(int fd)
- **Parameter:**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  If the return value = 1, it means the IRQ of RX is enabled. If the return value = 0, it means the IRQ of RX is disabled.

2.3.17 CAN_IrqStatus

- **Description:**
  To get the status of IER register.
- **Syntax:**
  WORD CAN_IrqStatus(int fd)
- **Parameter:**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  To return the IER register value.

2.3.18 CAN_ClearDataOverrun

- **Description:**
  To clear the data overrun bit.
- **Syntax:**
  WORD CAN_ClearDataOverrun(int fd)
- **Parameter:**
  fd : The file descriptor of device file that get from function CAN_Open.
- **Return:**
  The code “CAN_CLEARDATA_OVERRUN_ERROR” or “CAN_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.19 CAN_RxMsgCount

- **Description:**

  To get the count of RX message.

- **Syntax:**

  WORD CAN_RxMsgCount(int fd)

- **Parameter:**

  fd : The file descriptor of device file that get from function CAN_Open.

- **Return:**

  To return RX message count.

3.PISO-CAN200/400 Demo Programs For Linux

All of demo programs will not work normally if PCI linux driver would not be installed correctly. During the installation process of PCI linux driver, the install-scripts “ixpci.inst” will setup the correct kernel driver. After driver(version 0.7.5 or the later driver version) compiled and installation, the related demo programs, declaration header files for different development environments are presented as follows.

Table 3.1

Driver Name	Directory Path	File Name	Description
Ixpci-0.7.5	examples/ pisocan200	getirqstatus_a	Using CAN library function to show IER register status
		getstatus_a	Using CAN library function to show CAN card status
		send_receive_irq_a	Using CAN library function to send CAN message from port 1 and receive CAN message from port 2 with TX/RX IRQ
		send_receive_irq_a2	To send 10000 CAN message from port 1 and receive from port2
		send_receive_noirq_a	Using CAN library function to send CAN message from port 1 and receive CAN message from port 2 with ioctl command
		getstatus	Show CAN card status.

		send_receive_irq	Using interrupt handler to send CAN message from port 1 and receive CAN message from port 2.
		send_receive_noirq	To send CAN message from port 1 and receive CAN message from port 2 without using interrupt handler.
	examples/ pisocan400	getirqstatus_a	Using CAN library function to show IER register status
		getstatus_a	Using CAN library function to show CAN card status
		send_receive_irq_a	Using CAN library function to send CAN message from port 1,3 and receive CAN message from port 2,4 with TX/RX IRQ
		send_receive_irq_a2	To send 10000 CAN message from port 1 and receive from port2
		send_receive_noirq_a	Using CAN library function to send CAN message from port 1,3 and receive CAN message from port 2,4 with ioctl command
		getstatus	Show CAN card status.
		send_receive_irq	Using interrupt handler to send CAN message from port 1, 3 and receive CAN message from port 2, 4.
		send_receive_noirq	To send CAN message from port 1, 3 and receive CAN message from port 2, 4 without using interrupt handler.

3.1 Demo code “getirqstatus_a”

Using the CAN library function “CAN_IrqStatus” to show IER register status.

3.2 Demo code “getstatus_a”

Using the CAN library function “CAN_GetStatus” to show CAN card status.

3.3 Demo code “send_receive_irq_a”

This demo program is used to teach how to use CAN library function “CAN_SendMsgWithIrq” and “CAN_ReceiveMsgWithIrq” to send CAN message from port 1 (and port 3) and receive CAN message from port2 (and port 4) with IRQ handler.

3.4 Demo code “send_receive_irq_a2”

This demo program send 10000 CAN message from port 1 and receive from port2.

3.5 Demo code “send_receive_noirq_a”

This demo program is used to teach how to use CAN library function “CAN_SendMsgWithoutIrq” and “CAN_ReceiveMsgWithoutIrq” to send one CAN message from port 1 (and port 3) and receive one CAN message from port2 (and port 4) without using IRQ handler.

3.6 Demo code “getstatus.c”

This demo program is used to output CAN card chip type、baud 、TX and RX status.

3.7 Demo code “send_receive_irq.c”

This demo program is used to send three CAN messages from port 1 (and port 3) and receive three CAN messages from port2 (and port 4) with IRQ handler.

3.8 Demo code “send_receive_noirq.c”

This demo program is used to send one CAN message from port 1 (and port 3) and receive one CAN message from port2 (and port 4) without using IRQ handler.