PCI-180X H/L

Linux Software 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 PCI-180X H/L can be used in linux kernel 2.4.X and 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-0.7.3.tar.gz”(or the later driver version) in the directory “NAPDOS\Linux” of the companion CD to the linux host that you want to install driver.

Step 2: Decompress the tarball “ixpci-0.7.3.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 “ixpciX” in the device directory “/dev” automatically.

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.

2. Static Libary Function Description

The static libary is the collection of function calls of the PCI cards for linux kernel 2.4.x and 2.6.x system. The application structure is presented as following figure. The user application program developed by C(C++) language can call library “libpci.a” in user mode. And then static libary will call the module ixpci to access the hardware system.

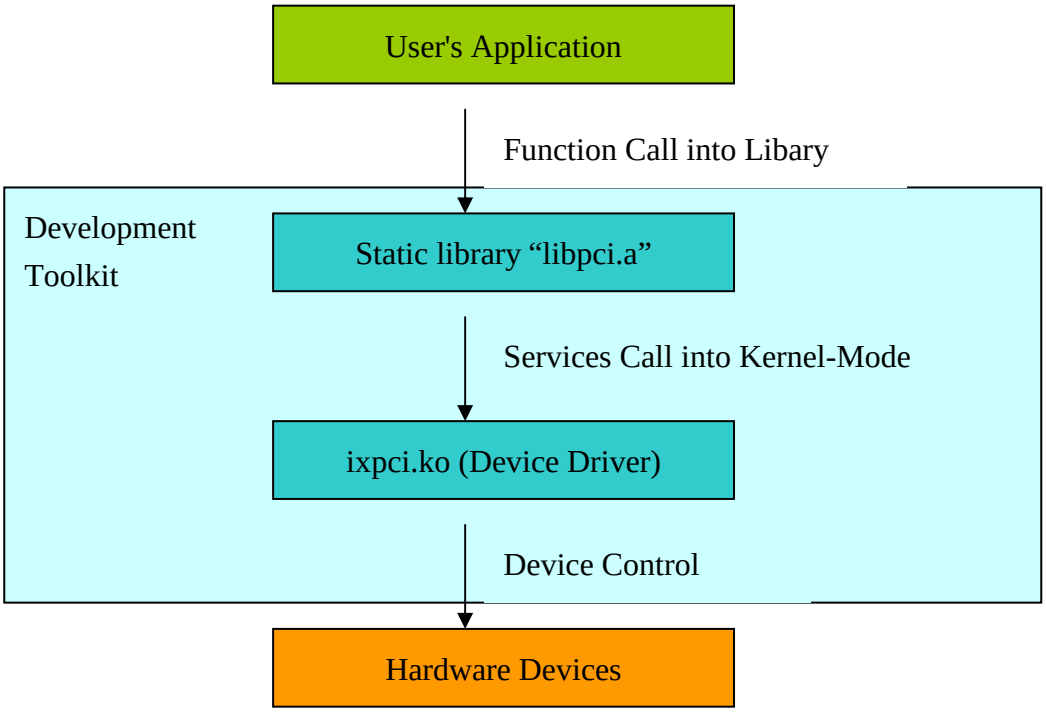

Figure 2.1

2.1 Table of ErrorCode and ErrorString

Table 2.1

Error Code	Error ID	Error String
0	PCIDA_NOERROR	OK (No error !)
1	PCIDA_MODULE_NAME_GET_ERROR	Module name can't get from file /proc/ixpci/ixpci
2	PCIDA_DIGITAL_OUTPUT_ERROR	Digital output error
3	PCIDA_DIGITAL_INPUT_ERROR	Digital input error
4	PCIDA_ANALOG_OUTPUT_ERROR	Analog output error
5	PCIDA_ANALOG_INPUT_ERROR	Analog input error
6	PCIDA_READ_SR_ERROR	Read status register error
7	PCIDA_WRITE_CR_ERROR	Write control register error
8	PCIDA_SOFTWARE_TRIGGER_ERROR	Software trigger error
9	PCIDA_CR_TIME_OUT	Control register time out
10	PCIDA_WRITE_8254CR_ERROR	Write 8254 control register error
11	PCIDA_WRITE_8254C0_ERROR	Write 8254 timer0 error
12	PCIDA_WRITE_8254C1_ERROR	Write 8254 timer1 error
13	PCIDA_WRITE_8254C2_ERROR	Write 8254 timer2 error
14	PCIDA_8254_DELAY_ERROR	8254 delay error
15	PCIDA_SET_CHANNEL_ERROR	Set AI channel error
16	PCIDA_CREATE_SCAN_THREAD_ERROR	Thread created error
17	PCIDA_MAGIC_SCAN_THREAD_ERROR	Magic scan thread error
18	PCIDA_CONFIG_CODE_ERROR	The Config code isn't right

2.2 Function Descriptions

Table 2.2

Common Function Definition
WORD PCIDA_GetDriverVersion(void);
WORD PCIDA_GetLibaryVersion(void);

int PCIDA_Open(char *);
WORD PCIDA_Close(WORD);
WORD PCIDA_DriverInit(WORD);
WORD PCIDA_SetChannelConfig(WORD, WORD, WORD);
WORD PCIDA_Digital_Output(WORD, DWORD);
WORD PCIDA_Digital_Input(WORD, DWORD *)
WORD PCIDA_Analog_DaF_Output(WORD, WORD, float);
WORD PCIDA_Analog_Da_Output(WORD, WORD, float);
PCI-1800H/L Function Definition
WORD PCI_180X_DelayUs(WORD, int);
WORD PCI_180X_AdsPolling(WORD, float [], WORD);
WORD PCI_180X_AdsPacer(WORD ,float [], WORD, WORD);
WORD PCI_180X_AddToScan(WORD, WORD, WORD, WORD, WORD, WORD, WORD);
WORD PCI_180X_SaveScan(WORD , WORD, WORD []);
WORD PCI_180X_ClearScan(WORD);
WORD PCI_180X_StartScan(WORD, WORD, WORD, int);
void PCI_180X_ReadScanStatus(WORD *, DWORD *, DWORD *);
WORD PCI_180X_StopMagicScan(WORD fd);
WORD PCI_180X_StartScanPostTrg(WORD, WORD, DWORD, int);
WORD PCI_180X_StartScanPreTrg(WORD, WORD, DWORD, int);
WORD PCI_180X_StartScanMiddleTrg(WORD, WORD, DWORD, DWORD, int);
WORD PCI_180X_StartScanPreTrgVerC(WORD, WORD, DWORD, int);
WORD PCI_180X_StartScanMiddleTrgVerC(WORD, WORD, DWORD, DWORD, int);
PCI-1800H/L M_Function Definition
WORD PCI_180X_M_FUN_1(WORD fd, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm);
WORD PCI_180X_M_FUN_2(WORD fd, WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD wAdClock, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);
WORD PCI_180X_M_FUN_3(WORD fd, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm);

WORD PCI_180X_M_FUN_4(WORD fd, WORD wType, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm);

2.3 Common Function Definition

2.3.1 PCIDA_GetDriverVersion

- **Description:**
  To show the version number of PCI linux driver.
- **Syntax:**
  WORD PCIDA_GetDriverVersion(Void)
- **Parameter:**
  None
- **Return:**
  The code “PCIDA_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.2 PCIDA_GetLibaryVersion

- **Description:**
  To show the version number of PCI linux static libary.
- **Syntax:**
  WORD PCIDA_GetLibaryVersion(void)
- **Parameter:**
  None
- **Return:**
  The code “PCIDA_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.3 PCIDA_Open

- **Description:**
  To open device file.

- **Syntax:**
  int PCIDA_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 PCIDA_Close

- **Description :**
  To close device file.
- **Syntax :**
  Word PCIDA_Close(WORD fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
- **Return:**
  The code “PCIDA_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.5 PCIDA_DriverInit

- **Description :**
  To allocates the computer resource for the device. This function must be called once before applying other PCIDA functions.
- **Syntax :**
  WORD PCIDA_DriverInit(WORD fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open
- **Return:**
  The code “PCIDA_MODULE_NAME_GET_ERROR”, or “PCIDA_NOERROR”(Please refer to "Section 2.1 Error Code").

2.3.6 PCIDA_SetChannelConfig

- **Description :**
  To set channel configuration
- **Syntax :**
  WORD PCIDA_SetChannelConfig(WORD fd, WORD ad_channel, WORD ad_config)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  ad_channel : The.analog input channel number.
  ad_config : The configuration code. Please refer to section 4.1 “The Configuration Code Table” of PCI-1202/1602/1800/1802 Hardware User’s Manual.
- **Return:**
  The code “PCIDA_READ_SR_ERROR”, “PCIDA_WRITE_CR_ERROR” or “PCIDA_NOERROR”.

2.3.7 PCIDA_Digital_Output

- **Description :**
  To output digital data from CON1.
- **Syntax :**
  WORD PCIDA_Digital_Output(WORD fd, DWORD data)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  data : 16 bits data.
- **Return:**
  The code “PCIDA_DIGITAL_OUTPUT_ERROR " or “PCIDA_NOERROR”.

2.3.8 PCIDA_Digital_Input

- **Description :**
  To read digital data from CON2..
- **Syntax :**
  WORD PCIDA_Digital_Input(WORD fd, DWORD *di_data)

- **Parameter :**

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

  di_data : A variable address used to storage the 16 bits input data.

- **Return:**

  The code “PCIDA_DIGITAL_INPUT_ERROR” or “PCIDA_NOERROR”.

2.3.9 PCIDA_Analog_DaF_Output

- **Description :**

  To output analog data from analog port 0,1.

- **Syntax :**

  WORD PCIDA_Analog_DaF_Output(WORD fd, WORD da_channel, float da_value)

- **Parameter :**

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

  da_channel : The AO port number.

  da_value : The voltage value.

- **Return:**

  The code “PCIDA_ANALOG_OUTPUT_ERROR” or “PCIDA_NOERROR”.

2.3.10 PCIDA_Analog_Da_Output

- **Description :**

  To output analog data from analog port 0,1.

- **Syntax :**

  WORD PCI_180X_Da_Output(WORD fd, WORD da_channel, float da_value)

- **Parameter :**

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

  da_channel : The AO port number.

  da_value : The digital value calculated by user.

- **Return:**

  The code “PCIDA_ANALOG_OUTPUT_ERROR” or

“PCIDA_NOERROR”.

2.4 PCI-180X H/L Function Definition

2.4.1 PCI_180X_DelayUs

- **Description :**
  The function is used to delay the setting time.
- **Syntax :**
  WORD PCI_180X_DelayUs(WORD fd, int us)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  us : The setting time that user want to delay.
- **Return:**
  The code “PCIDA_8254_DELAY_ERROR” or “PCIDA_NOERROR”.

2.4.2 PCI_180X_AdsPolling

- **Description :**
  To perform one AD conversion with polling.
- **Syntax :**
  WORD PCI_180X_AdsPolling(WORD fd, float fAdVal[], WORD index)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  fAdVal : A array is used to storage the conversion data.
  index : The frequency of AD polling.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that conversion data successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.3 PCI_180X_AdsPacer

- **Description :**
  To perform one AD conversion with pacer.
- **Syntax :**
  WORD PCI_180X_AdsPacer(WORD fd, float fAdVal[], WORD index,

WORD samplerate)

- **Parameter :**

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

  fAdVal : A array is used to storage the conversion data.

  Index : The frequency of AD polling.

  samplerate :The sample rate

- **Return:**

  If returned value = PCIDA_NOERROR, it means that conversion data successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.4 PCI_180X_AddToScan

- **Description :**

  To add the channels to the magicscan circular queue one by one.

- **Syntax :**

  WORD PCI_180X_AddToScan(WORD fd, WORD wAdChannel, WORD wAdConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType)

- **Parameter :**

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

  wAdChannel : The channel that added to magicscan circular queue.

  wAdConfig : The configuration code. Please refer to section 4.1 “The Configuration Code Table” of PCI-1202/1602/1800/1802 Hardware User’s Manual.

  wAverage : wAverage is a average factor of digital filter.

  wLowAlarm : A low alarm value.

  wHighAlarm : A high alarm value.

  wAlarmType : The alarm type. Please refer to section 4.8.4 “The High/Low Alarm of MagicScan” of PCI-1202/1602 /1800/1802 Hardware User’s Manual.

- **Return:**

  If returned value = PCIDA_NOERROR, it means that add channel to magic scan circular queue successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.5 PCI_180X_SaveScan

- **Description :**
  To save AD data.
- **Syntax :**
  WORD PCI_180X_SaveScan(WORD fd, WORD wQueueOrder, WORD wBuf[])
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  wQueueOrder : The channel order in the magic scan circular queue.
  wBuf : The buffer is used to save AD data.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that set data buffer successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.6 PCI_180X_ClearScan

- **Description :**
  To clear the data buffer.
- **Syntax :**
  WORD PCI_180X_ClearScan(WORD fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that clear data buffer successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.7 PCI_180X_StartScan

- **Description :**
  To start magicscan operation.
- **Syntax :**
  WORD PCI_180X_StartScan(WORD fd, WORD wSampleRateDiv, WORD wNum, int nPriority)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.

wSampleRateDiv : The sample rate.
wNum : The scan cycles.
nPriority : The pthread priority.

- **Return:**
  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.8 PCI_180X_ReadScanStatus

- **Description :**
  To read magicscan status.
- **Syntax :**
  void PCI_180X_ReadScanStatus(WORD *wStatus, DWORD *dwLowAlarm, DWORD *dwHighAlarm)
- **Parameter :**
  wStatus : A variable used to storage the magicscan status.
  dwLowAlarm : A variable used to storage the magicscan low alarm.
  dwHighAlarm: A variable used to storage the magicscan high alarm..
- **Return:**
  None.

2.4.9 PCI_180X_StopMagicScan

- **Description :**
  To stop magicscan controller.
- **Syntax :**
  WORD PCI_180X_StopMagicScan(WORD fd)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that stop magicscan controller successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.10 PCI_180X_StartScanPostTrg

- **Description :**
  To start the magicscan operation with post-trigger mode.

- **Syntax :**
  WORD PCI_180X_StartScanPostTrg(WORD fd, WORD wSampleRateDiv, DWORD dwNum, int nPriority)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  wSampleRateDiv : The sample rate.
  wNum : The scan cycles.
  nPriority : The pthread priority.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.11 PCI_180X_StartScanPreTrg

- **Description :**
  To start the magicscan operation with pre-trigger mode.
- **Syntax :**
  WORD PCI_180X_StartScanPreTrg(WORD fd, WORD wSampleRateDiv, DWORD dwNum, int nPriority)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  wSampleRateDiv : The sample rate.
  wNum : The scan cycles.
  nPriority : The pthread priority.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.12 PCI_180X_StartScanMiddleTrg

- **Description :**
  To start the magicscan operation with middle-trigger mode.
- **Syntax :**
  WORD PCI_180X_StartScanMiddleTrg(WORD fd, WORD wSampleRateDiv, DWORD dwN1, DWORD dwN2, int nPriority)
- **Parameter :**

fd : The file descriptor of device file that get from function PCIDA_Open.
wSampleRateDiv : The sample rate.
dwN1 : To acquire dwN1 AD data before trigger event.
dwN2 : To acquire dwN2 AD data after trigger event.
nPriority : The pthread priority.

- **Return:**
  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.13 PCI_180X_StartScanPreTrgVerC

- **Description :**
  To start the magicscan operation with pre-trigger mode for PCI-1800 version C.
- **Syntax :**
  WORD PCI_180X_StartScanPreTrgVerC(WORD fd, WORD wSampleRateDiv, DWORD dwN1, int nPriority)
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  wSampleRateDiv : The sample rate.
  dwN1 : To Acquire dwN1 AD data before trigger event.
  nPriority : The pthread priority.
- **Return:**
  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.4.14 PCI_180X_StartScanMiddleTrgVerC

- **Description :**
  To start the magicscan operation with middle-trigger mode for PCI-1800 version C.
- **Syntax :**
  WORD PCI_180X_StartScanMiddleTrgVerC(WORD fd, WORD wSampleRateDiv, DWORD dwN1, DWORD dwN2, int nPriority)
- **Parameter :**
  fd : The file descriptor of device file that get from function

PCIDA_Open.

wSampleRateDiv : The sample rate.

dwN1 : To acquire dwN1 AD data before trigger event.

dwN2 : To acquire dwN2 AD data after trigger event.

nPriority : The pthread priority.

- **Return:**

  If returned value = PCIDA_NOERROR, it means that start magicscan successfully. Otherwise, please refer to "Section 2.1 Error Code".

2.5 PCI-180X H/L M_Function Definition

2.5.1 PCI_180X_M_FUN_1

- **Description :**

  The PCI_180X_M_FUN_1 will automatic to compute the sine wave output image.

- **Syntax :**

  WORD P180X_M_FUN_1(WORD fd, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm)

- **Parameter :**

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

  wDaFrequency : The data frequency.

  wDaWave : The data wave.

  fDaAmplitude : The data amplitude.

  wAdClock : The sample rate.

  wAdNumber : The frequency of reading AD data.

  wAdConfig : The configuration code. Please refer to section 4.1 “The Configuration Code Table” of PCI-1202/1602/1800/1802 Hardware User’s Manual.

  wAdBuf : A array is used to storage the conversion data.

  fLowAlarm : The low alarm.

  fHighAlarm : The high alarm.

- **Return:**

  Please refer to "Section 2.1 Error Code".

2.5.2 PCI_180X_M_FUN_2

- **Description :**
  The PCI_180X_M_FUN_2 is designed for arbitrary waveform generation.
- **Syntax :**
  WORD PCI_180X_M_FUN_2(WORD fd, WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD wAdClock, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[])
- **Parameter :**
  fd : The file descriptor of device file that get from function PCIDA_Open.
  wDaNumber : The data frequency.
  wDaWave : The data wave.
  fDaBuf : The output data defined by user.
  wAdClock : The sample rate.
  wAdNumber : The frequency of reading AD data.
  wAdConfig : The configuration code. Please refer to section 4.1 “The Configuration Code Table” of PCI-1202/1602/1800/1802 Hardware User’s Manual.
  wAdBuf : A array is used to storage the conversion data.
- **Return:**
  Please refer to "Section 2.1 Error Code".

2.5.3 PCI_180X_M_FUN_3

- **Description :**
  The PCI_180X_M_FUN_3 is similar to PCI_180X_M_FUN_1 except the A/D input channels are programmable
- **Syntax :**
  WORD PCI_180X_M_FUN_3(WORD fd, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm)
- **Parameter :**
  Fd : The file descriptor of device file that get from function PCIDA_Open.
  wDaFrequency : The data frequency.

wDaWave : The data wave.
fDaAmplitude : The data amplitude.
wAdClock : The sample rate.
wAdNumber : The frequency of reading AD data.
wChannelStatus[] : To set each channel config or not, value 0:it means don't set channel config, value 1:it means set channel config.
wAdConfig[] : The configuration code. Please refer to section 4.1 "The Configuration Code Table" of PCI-1202/1602/1800/1802 Hardware User's Manual.
wAdBuf : A array is used to storage the conversion data.
fLowAlarm : The low alarm.
fHighAlarm : The high alarm.

- **Return:**

Please refer to "Section 2.1 Error Code".

2.5.4 PCI_180X_M_FUN_4

- **Description :**

The PCI_180X_M_FUN_4 will automatic to compute the sine, semi-square or square wave output image.

- **Syntax :**

WORD PCI_180X_M_FUN_4(WORD fd, WORD wType, WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm)

- **Parameter :**

fd : The file descriptor of device file that get from function PCIDA_Open.
wType : wType = 1, it means square wave, wType = 2, it means semi-square,default is sine wave.
wDaFrequency : The data frequency.
wDaWave : The data wave.
fDaAmplitude : The data amplitude.
wAdClock : The sample rate.
wAdNumber : The frequency of reading AD data.
wChannelStatus[] : To set each channel config or not, value 0:it means don't set channel config, value 1:it means set channel config.

wAdConfig[] : The configuration code. Please refer to section 4.1 “The Configuration Code Table” of PCI-1202/1602/1800/1802 Hardware User’s Manual.

wAdBuf : A array is used to storage the conversion data.

fLowAlarm : The low alarm.

fHighAlarm : The high alarm.

- **Return:**

  Please refer to "Section 2.1 Error Code".

3. PCI-180X H/L 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 0.7.3 or the later driver version) compiled and installation, the related demo programs , development library and declaration header files for different development environments are presented as follows.

Table 3.1

Driver Name	Directory Path	File Name	Description
ixpci-0.7.3	Include	pcidio.h	PCI library header
	lib	libpci.a	PCI static libary
	examples/pci1800	dio.c	DI/O demo
		reset.c	Reset control register
		time_span.c	Delay setting time
		dio_a.c	DI/O demo with libary
		dac_a.c	DA demo with libary
		adspolling_a.c	AD polling demo with libary
		adspacer_a.c	AD pacer demo with libary
		magicscan_a.c	MagicScan demo with libary

3.1 Demo code “dio.c”

This demo program is used to output data from CON1 and read data from CON2.

3.2 Demo code “reset.c”

This demo program is used to reset control register.

3.3 Demo code “time_span.c”

This demo program will use 8254 timer2 to delay 6000 us.

3.4 Demo code “dio_a.c”

This demo program coded by using the static library “libpci.a” to output digital data from CON1, and read digital data from CON2.

3.5 Demo code “dac_a.c”

This demo program output 5V from Analog channel 0 by using libary AO function.

3.6 Demo code “adspolling_a.c”

This demo program output 5V from AO channel 0 by using libary AO function and read data from AI channel 0 with library polling function.

3.7 Demo code “adspacer_a.c”

This demo program read data from AI channel 0 with library pacer function.

3.8 Demo code “magicscan_a.c”

This demo program add AI channel 4,3,5 to magicscan circular queue and start magicscan ability with library magicscan function.