PCI-1202/1602/1800/1802

Software User's Manual for DOS

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.

License

The user can use, modify and backup this software **on a single machine.** The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of Contents

1 . Introduction

The **PCI-1202/1602/1800/1802 DOS driver** is a collection of library for DOS application. These library can be called by TC 2.x, BC 3.x and MSC 5.x. These library can perform a variety of data acquisition operations as follows:

- Get software version
- Initialization
- Digital Input/Output
- A/D conversion
- D/A conversion
- Demo

The driver source are given in the companion floppy disk. There are about 19 demo program given in the companion CD-ROM or floppy disk. The user can print out theses files for reference.

The hardware I/O control register & its function are given in the "PCI-1202/1602/ 1800/1802 Hardware User's Manual". The user should refer to "PCI-1202/1602/1800/1802 Hardware User's Manual" for more information.

- **12-bit DAC Resolution for PCI-1202$_{(H/L)}$/1602/1602$_F$/1800$_{(H/L)}$/1802$_{(H/L)}$**

Data Input	Analog Ouput
MSB LSB	
1111 1111 1111	+Vref (2047/2048)
1000 0000 0001	+Vref (1/2048)
1000 0000 0000	0 Vlots
0111 1111 1111	-Vref (1/2048)
0000 0000 0000	-Vref (2048/2048)

Bipolar Output Code.

Vref is +5V or +10V selected by J1.

- **12-bit ADC Resolution for PCI-1202$_{(H/L)}$/1800$_{(H/L)}$/1802$_{(H/L)}$**

Analog Input	Digital Output Binary Code	Hex Code
	MSB LSB	
+9.99512V	0111 1111 1111	7FF
0V	0000 0000 0000	000
-4.88m V	1111 1111 1111	FFF
-10V	1000 0000 0000	800

Input Voltages and Output Codes.

- **16-bit ADC Resolution for PCI-1602/1602$_F$ Card**

Analog Input	Digital Output Binary Code	Hex Code
	MSB LSB	
+9.999695V	0111 1111 1111 1111	7FFF
0V	0000 0000 0000 0000	0000
-305uv	1111 1111 1111 1111	FFFF
-10V	1000 0000 0000 0000	8000

Input Voltages and Output Codes.

1.1 Software Installation

It is recommended to install the PCI-1202/1602/1800/1802 DOS driver to your hard disk and backup the companion CD ROM. The contents of PCI-1202/1602/1800/1802 DOS CD ROM are given as follows:

Rrefer to CD ROM readme.txt file.

1.1.1 Software content for PCI-1202(H/L)

…\1202\TC>.....For Turbo C 2.0 or later user
…\1202\BC>.....For Borland C++ 3.0 later user
…\1202\MSC>.....For Microsoft C 5.1 later user

…1202\TC\DRIVER>P1202DR.H.....Header file
…\1202\TC\DRIVER>P1202TCL.C.....Driver source
…\1202\TC\DRIVER>IOPORT.C.....32-bit I/O port driver source

…\1202\TC\LIB>P1202.H.....Header file
…\1202\TC\LIB>P1202TCL.LIB.....Large model library
…\1202\TC\LIB>P1202TCH.LIB.....Huge model library
…\1202\TC\LIB>IOPORTL.LIB.....Large model library
…\1202\TC\LIB>IOPORTH.LIB.....Huge model library

…\1202\TC\DEMO>P1202TCL.LIB.....Large model library
…\1202\TC\DEMO>IOPORTL.LIB.....Large model library
…\1202\TC\DEMO>P1202.H.....Header file
…\1202\TC\DEMO>DEMO1.C.....demo1 program source
…\1202\TC\DEMO>DEMO1.PRJ.....demo1 project file
…\1202\TC\DEMO>DEMO1.EXE.....demo1 execution
…\1202\TC\DEMO>DEMO2.C .....demo2 program source

:
:

```
                                    :
…\1202\TC\DEMO>DEMO19.C.....demo19 program source
…\1202\TC\DEMO>DEMO19.PRJ.....demo19 project file
…\1202\TC\DEMO>DEMO19.EXE.....demo19 execution file
```

1.1.2 Software content for PCI-1602/1602F

```
…\1602\TC>.....For Turbo C 2.0 or later user
…\1602\BC>.....For Borland C++ 3.0 later user
…\1602\MSC>.....For Microsoft C 5.1 later user

...\1602\TC\DRIVER>P1602DR.H.....Header file
…\1602\TC\DRIVER>P160XTCL.C.....Driver source
…\1602\TC\DRIVER>IOPORT.C.....32-bit I/O port driver source

…\1602\TC\LIB>P1602.H.....Header file
…\1602\TC\LIB>P1602TCL.LIB.....Large model library
…\1602\TC\LIB>P1602TCH.LIB.....Huge model library
…\1602\TC\LIB>IOPORTL.LIB.....Large model library
…\1602\TC\LIB>IOPORTH.LIB.....Huge model library

…\1602\TC\DEMO>P1602TCL.LIB.....Large model library
…\1602\TC\DEMO>IOPORTL.LIB.....Large model library
…\1602\TC\DEMO>P1602.H.....Header file
…\1602\TC\DEMO>DEMO1.C.....demo1 program source
…\1602\TC\DEMO>DEMO1.PRJ.....demo1 project file
…\1602\TC\DEMO>DEMO1.EXE.....demo1 execution file
                                    :
                                    :
                                    :
…\1602\TC\DEMO>DEMO19.C.....demo1 program source
…\1602\TC\DEMO>DEMO19.PRJ.....demo1 project file
…\1602\TC\DEMO>DEMO19.EXE.....demo1 execution file
```

1.1.3 Software content for PCI-1800(H/L)/1802(H/L)

…\180X\TC>.....For Turbo C 2.0 or later user
…\180X\BC>.....For Borland C++ 3.0 later user
…\180X\MSC>.....For Microsoft C 5.1 later user

…\180X\TC\DRIVER>P180XDR.H.....Header file
…\180X\TC\DRIVER>P180XTCL.C.....Driver source
…\180X\TC\DRIVER>IOPORT.C.....32-bit I/O port driver source

…\180X\TC\LIB>P180X.H.....Header file
…\180X\TC\LIB>P180XTCL.LIB.....Large model library
…\180X\TC\LIB>P180XTCH.LIB.....Huge model library
…\180X\TC\LIB>IOPORTL.LIB.....Large model library
…\180X\TC\LIB>IOPORTH.LIB.....Huge model library
…\180X\TC\DEMO>P180XTCL.LIB.....Large model library
…\180X\TC\DEMO>IOPORTL.LIB.....Large model library
…\180X\TC\DEMO>P180X.H.....Header file
…\180X\TC\DEMO>DEMO1.C.....demo1 program file
…\180X\TC\DEMO>DEMO1.PRJ.....demo1 project file
…\180X\TC\DEMO>DEMO1.EXE.....demo1 execution file
…\180X\TC\DEMO>DEMO2.C.....demo 2 program file
:
:
:
…\180X \TC\DEMO>DEMO19.C.....demo19 program source
…\180X\TC\DEMO>DEMO19.PRJ.....demo19 project file
…\180X\TC\DEMO>DEMO19.EXE.....demo19 execution file

1.2 Compiler & link using MSC

- The including file is **P1602.H**
- There are two PCI-1602/1602F library files: **P1602MCH.LIB & P1602MCL.LIB**
- There are two 32-bit I/O library files: **IOPORTH.LIB & IOPORTL.LIB**
- Support MSC 5.1 & later compiler
- **LARGE** model compiler & link command : CL /AL demo?.c P1602MCL.LIB IOPORTL.lib
- **HUGE** model compiler & link command : CL /AH demo?.c P1602MCH.LIB IOPORTH.LIB

- **Note: The steps are the same for PCI-1202(H/L)/1800(H/L)/1802(H/L) card.**

1.3 Compiler & link using TC

- The including file is **P1602.H**
- There are two PCI-1602/1602F library files: **P1602TCH.LIB & P1602TCL.LIB**
- There are two 32-bit I/O library files: **IOPORTH.LIB & IOPORTL.LIB**
- Support TC 2.x compiler
- Use the text editor to create a large model project file as follows:
  DEMO?.C
  P1602TCH.lib
  IOPORTH.lib
- Or use the text editor to create a huge model project file as follows:
  DEMO?.C
  P1602TCL.lib
  IOPORTL.lib
- Use TC integrated environment to **select the correct project file**
- Use TC integrated environment to **select the correct compiler model**
- **Press F9 to compiler&link**

- **Note: The steps are the same for PCI-1202(H/L)/1800(H/L)/1802(H/L) card.**

1.4 Compiler & link using BC

- The including file is **P1602.H**
- There are two PCI-1602/1602F library files: **P1602BCH.LIB & P1602BCL.LIB**
- There are two 32-bit I/O library files: **IOPORTH.LIB & IOPORTL.LIB**
- Support BC 3.x compiler
- Use BC integrated environment to create a large model project file as follows :
  DEMO?.C
  P1602BCL.LIB
  IOPORTL.LIB
- Or use BC integrated environment to create a huge model project file as follows :
  DEMO?.C
  P1602BCH.LIB
  IOPORTH.LIB
- Use BC integrated environment to **select the correct compiler model**
- **Press F9 to compiler & link**

- **Note: The steps are the same for PCI-1202(H/L)/1800(H/L)/1202(H/L) card.**

2 . C Language Library

The libraries are divided into several groups as follows:

- The test functions
- The M_Functions function
- The D/I/O functions
- The D/A function
- The A/D fixed-mode functions
- The A/D MagicScan mode functions
- The A/D continuous capture functions
- The Plug&Play functions
- The other functions

- **For PCI-1202(H/L) Card.**

The functions of fixed-channel mode are given as follows :

1. P1202_SetChannelConfig 2. P1202_AdPoling 3. P1202_AdsPolling 4. P1202_AdsPacer	Data in float format

The functions of MagicScan mode are given as follows:

1. P1202_ClearScan 2. P1202_StartScan 3. P1202_AddToScan 4. P1202_SaveScan 5. P1202_ReadMagicScanResult	**Data in 12 bits HEX format**

The functions of M_Functions are given as follows:

1. P1202_M_FUN_1
2. P1202_M_FUN_2
3. P1202_M_FUN_3

The functions of continuous capture are given as follows:

1. P1202_Card0_StartScan
2. P1202_Card0_ReadData
3. P1202_Card0_Stop

- **For PCI-1602/1602F Card.**

The functions of fixed-channel mode are given as follows :

5. P1602_SetChannelConfig 6. P1602_AdPoling 7. P1602_AdsPolling 8. P1602_AdsPacer	Data in float format

The functions of MagicScan mode are given as follows:

6. P1602_ClearScan 7. P1602_StartScan 8. P1602_AddToScan 9. P1602_SaveScan 10. P1602_ReadMagicScanResult	**Data in 16 bits HEX format**

The functions of M_Functions are given as follows:

4. P1602_M_FUN_1
5. P1602_M_FUN_2
6. P1602_M_FUN_3

The functions of continuous capture are given as follows:

4. P1602_Card0_StartScan
5. P1602_Card0_ReadData
6. P1602_Card0_Stop

- **For PCI-1800(H/L)/1802(H/L) Card.**

The functions of fixed-channel mode are given as follows :

9. P180X_SetChannelConfig 10. P180X_AdPoling 11. P180X_AdsPolling 12. P180X_AdsPacer	Data in float format

The functions of MagicScan mode are given as follows:

11. P180X_ClearScan 12. P180X_StartScan 13. P180X_AddToScan 14. P180X_SaveScan 15. P180X_ReadMagicScanResult	**Data in 12 bits HEX format**

The functions of M_Functions are given as follows:

7. P180X_M_FUN_1
8. P180X_M_FUN_2
9. P180X_M_FUN_3

The functions of continuous capture are given as follows:

7. P180X_Card0_StartScan
8. P180X_Card0_ReadData
9. P180X_Card0_Stop

2.1 The Configuration Code Table

2.1.1 PCI-1602/1602F Configuration Code Table

Bipolar/Unipolar	Input Signal Range	Gain	Settling Time	Configuration Code
Bipolar	+/- 10V	1	3 us	0
Bipolar	+/- 5.0V	2	3 us	1
Bipolar	+/- 2.5V	4	3 us	2
Bipolar	+/- 1.25V	8	3 us	3

2.1.2 PCI-1202L/1800L/1802L Configuration Code Table

Bipolar/Unipolar	Input Signal Range	Gain	Settling Time	Configuration Code
Bipolar	+/- 5V	1	3 us	0x00
Bipolar	+/- 2.5V	2	3 us	0x01
Bipolar	+/- 1.25V	4	3 us	0x02
Bipolar	+/- 0.625V	8	3 us	0x03
Bipolar	+/- 10V	0.5	3 us	0x04
Bipolar	+/- 5V	1	3 us	0x05
Bipolar	+/- 2.5V	2	3 us	0x06
Bipolar	+/- 1.25V	4	3 us	0x07
Unipolar	0V ~ 10V	1	3 us	0x08
Unipolar	0V ~ 5V	2	3 us	0x09
Unipolar	0V ~ 2.5V	4	3 us	0x0A
Unipolar	0V ~ 1.25V	8	3 us	0x0B

2.1.3 PCI-1202H/1800H/1802H Configuration Code Table

Bipolar/Unipolar	Input Signal Range	Gain	Settling Time	Configuration Code
Bipolar	+/- 5V	1	23 us	0x10
Bipolar	+/- 0.5V	10	28 us	0x11
Bipolar	+/- 0.05V	100	140 us	0x12
Bipolar	+/- 0.005V	1000	1300 us	0x13
Bipolar	+/- 10V	0.5	23 us	0x14
Bipolar	+/- 1V	5	28 us	0x15
Bipolar	+/- 0.1V	50	140 us	0x16
Bipolar	+/- 0.01V	500	1300 us	0x17
Unipolar	0V ~ 10V	1	23 us	0x18
Unipolar	0V ~ 1V	10	28 us	0x19
Unipolar	0V ~ 0.1V	100	140 us	0x1A
Unipolar	0V ~ 0.01V	1000	1300 us	0x1B

2.2 PCI-1202/1602/1800/1802 Header File

2.2.1 P1202.H for PCI-1202(H/L) Card

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <dos.h>
#include  <conio.h>
#include  <math.h>
#include  <graphics.h>

#define   WORD              unsigned int
#define   DWORD             unsigned long
#define   UCHAR             unsigned char

#define NoError                     0
#define DriverHandleError           1
#define DriverCallError             2
#define AdControllerError           3
#define M_FunExecError              4
#define ConfigCodeError             5
#define FrequencyComputeError       6
#define HighAlarm                   7
#define LowAlarm                    8
#define AdPollingTimeOut            9
#define AlarmTypeError             10
#define FindBoardError             11
#define AdChannelError             12
#define DaChannelError             13
#define InvalidateDelay            14
#define DelayTimeOut               15
#define InvalidateData             16
#define FifoOverflow               17
#define TimeOut                    18
#define ExceedBoardNumber          19
#define NotFoundBoard              20
```

```
#define OpenError                    21

#define  MAX_BOARD_NUMBER      16

float P1202_FloatSub2(float fA, float fB);
short P1202_ShortSub2(short nA, short nB);

WORD P1202_DriverInit(WORD *wBoards);
WORD P1202_DriverClose(void);
WORD P1202_GetDriverVersion(WORD *wDriverVersion);

WORD P1202_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,      WORD
      *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
WORD P1202_DelayUs(WORD wDelayUs);

WORD P1202_ActiveBoard( WORD wBoardNo );
WORD P1202_WhichBoardActive(void);
WORD P1202_Di(WORD *wDi);
WORD P1202_Do(WORD wDo);
WORD P1202_Da(WORD wDaChannel, WORD wDaVal);

WORD P1202_SetChannelConfig(WORD wAdChannel, WORD wConfig);
WORD P1202_AdPolling(float *fAdVal);
WORD P1202_AdsPolling(float fAdVal[], WORD wNum);
WORD P1202_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);

WORD P1202_M_FUN_1(WORD wDaFrequency, WORD wDaWave, float dfDaAmplitude, WORD
      wAdClock, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float
      fHighAlarm);

WORD P1202_M_FUN_2(WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD
      wAdClock, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);

WORD P1202_M_FUN_3(WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD
      wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float
      fAdBuf[], float fLowAlarm, float fHighAlarm);
```

```
WORD P1202_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[], WORD
      wChannelConfig[], WORD wCount);
WORD P1202_Card0_ReadData(void);
void   P1202_Card0_Stop(void);

WORD P1202_ClearScan(void);
WORD P1202_StartScan(WORD wSampleRate, WORD wNum);
WORD P1202_AddToScan(WORD wAdChannel, WORD wAdConfig, WORD wAverage, WORD
      wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
WORD P1202_SaveScan(WORD wAdChannel, WORD wBuf[]);
WORD P1202_ReadMagicScanResult(DWORD *dwHiAlarm, DWORD dwLoAlarm);

/***
wBuf0[] -> the scaned data is stored as
           1,0,1,0,1,0,1,0…   order
wBuf1[] -> the scaned data is stored as
           0,0,0,0,.....,0,1,1,1,1,.....,1 order
           ^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^
           |<-DATACOUNT->| |<-DATACOUNT->|
             ch:0 data         ch:1 data
***/

extern WORD    wBuf0[],wBuf1[];
```

2.2.2 P1602.H for PCI-1602/1602F Card

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <dos.h>
#include  <conio.h>
#include  <math.h>
#include  <graphics.h>


#define   WORD              unsigned int
#define   DWORD             unsigned long
#define   UCHAR             unsigned char

#define NoError                     0
#define DriverHandleError           1
#define DriverCallError             2
#define AdControllerError           3
#define M_FunExecError              4
#define ConfigCodeError             5
#define FrequencyComputeError       6
#define HighAlarm                   7
#define LowAlarm                    8
#define AdPollingTimeOut            9
#define AlarmTypeError             10
#define FindBoardError             11
#define AdChannelError             12
#define DaChannelError             13
#define InvalidateDelay            14
#define DelayTimeOut               15
#define InvalidateData             16
#define FifoOverflow               17
#define TimeOut                    18
#define ExceedBoardNumber          19
#define NotFoundBoard              20
#define OpenError                  21
```

```
#define  MAX_BOARD_NUMBER   16

float P1602_FloatSub2(float fA, float fB);
short P1602_ShortSub2(short nA, short nB);

WORD P1602_DriverInit(WORD *wBoards);
WORD P1602_DriverClose(void);
WORD P1602_GetDriverVersion(WORD *wDriverVersion);



WORD P1602_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,
        WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
WORD P1602_DelayUs(WORD wDelayUs);
WORD P1602_ActiveBoard( WORD wBoardNo );
WORD P1602_WhichBoardActive(void);
WORD P1602_Di(WORD *wDi);
WORD P1602_Do(WORD wDo);
WORD P1602_Da(WORD wDaChannel, WORD wDaVal);

WORD P1602_SetChannelConfig(WORD wAdChannel, WORD wConfig);
WORD P1602_AdPolling(float *fAdVal);
WORD P1602_AdsPolling(float fAdVal[], WORD wNum);
WORD P1602_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);

WORD P1602_M_FUN_1(WORD wDaFrequency, WORD wDaWave, float dfDaAmplitude,
        WORD wAdClock, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float
        fLowAlarm, float fHighAlarm);

WORD P1602_M_FUN_2(WORD wDaNumber, WORD wDaWave, WORD wDaBuf[],
        WORD wAdClock, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);

WORD P1602_M_FUN_3(WORD wDaFrequency, WORD wDaWave, float fDaAmplitude,
        WORD wAdClock, WORD wAdNumber, WORD wChannelStatus[],
        WORD wAdConfig[], float fAdBuf[], float fLowAlarm, float fHighAlarm);
```

```
WORD P1602_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[],
        WORD wChannelConfig[], WORD wCount);
WORD P1602_Card0_ReadData(void);
   void   P1602_Card0_Stop(void);

WORD P1602_ClearScan(void);
WORD P1602_StartScan(WORD wSampleRate, WORD wNum);
WORD P1602_AddToScan(WORD wAdChannel, WORD wAdConfig, WORD wAverage,
        WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
WORD P1602_SaveScan(WORD wAdChannel, WORD wBuf[]);
WORD P1602_ReadMagicScanResult(DWORD *dwHiAlarm, DWORD dwLoAlarm);
```

```
/***
wBuf0[] -> the scaned data is stored as
           1,0,1,0,1,0,1,0…   order
wBuf1[] -> the scaned data is stored as
           0,0,0,0,.....,0,1,1,1,1,.....,1 order
           ^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^
           |<-DATACOUNT->| |<-DATACOUNT->|
             ch:0 data         ch:1 data
***/

extern WORD    wBuf0[],wBuf1[];
```

2.2.3 P180X.H for PCI-1800(H/L)/1802(H/L) Card

```
#include  <stdio.h>
#include  <stdlib.h>
#include  <dos.h>
#include  <conio.h>
#include  <math.h>
#include  <graphics.h>

#define   WORD              unsigned int
#define   DWORD             unsigned long
#define   UCHAR             unsigned char

#define NoError                    0
#define DriverHandleError          1
#define DriverCallError            2
#define AdControllerError          3
#define M_FunExecError             4
#define ConfigCodeError            5
#define FrequencyComputeError      6
#define HighAlarm                  7
#define LowAlarm                   8
#define AdPollingTimeOut           9
#define AlarmTypeError            10
#define FindBoardError            11
#define AdChannelError            12
#define DaChannelError            13
#define InvalidateDelay           14
#define DelayTimeOut              15
#define InvalidateData            16
#define FifoOverflow              17
#define TimeOut                   18
#define ExceedBoardNumber         19
#define NotFoundBoard             20
#define OpenError                 21

#define  MAX_BOARD_NUMBER   16
```

```
float P180X_FloatSub2(float fA, float fB);
short P180X_ShortSub2(short nA, short nB);

WORD P180X_DriverInit(WORD *wBoards);
WORD P180X_DriverClose(void);
WORD P180X_GetDriverVersion(WORD *wDriverVersion);

WORD P180X_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,      WORD
      *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
WORD P180X_DelayUs(WORD wDelayUs);

WORD P180X_ActiveBoard( WORD wBoardNo );
WORD P180X_WhichBoardActive(void);
WORD P180X_Di(WORD *wDi);
WORD P180X_Do(WORD wDo);
WORD P180X_Da(WORD wDaChannel, WORD wDaVal);

WORD P180X_SetChannelConfig(WORD wAdChannel, WORD wConfig);
WORD P180X_AdPolling(float *fAdVal);
WORD P180X_AdsPolling(float fAdVal[], WORD wNum);
WORD P180X_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);

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

WORD P180X_M_FUN_2(WORD wDaNumber, WORD wDaWave, WORD wDaBuf[], WORD
      wAdClock, WORD wAdNumber, WORD wAdConfig, WORD wAdBuf[]);

WORD P180X_M_FUN_3(WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD
      wAdClock, WORD wAdNumber, WORD wChannelStatus[], WORD wAdConfig[], float
      fAdBuf[], float fLowAlarm, float fHighAlarm);
```

```
        WORD   wChannelConfig[], WORD wCount);
WORD P180X_Card0_ReadData(void);
  void   P1802_Card0_Stop(void);

WORD P180X_ClearScan(void);
WORD P180X_StartScan(WORD wSampleRate, WORD wNum);
WORD P180X_AddToScan(WORD wAdChannel, WORD wAdConfig, WORD wAverage,
        WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
WORD P180X_SaveScan(WORD wAdChannel, WORD wBuf[]);
WORD P180X_ReadMagicScanResult(DWORD *dwHiAlarm, DWORD dwLoAlarm);

/***
wBuf0[] -> the scaned data is stored as
           1,0,1,0,1,0,1,0…   order
wBuf1[] -> the scaned data is stored as
           0,0,0,0,.....,0,1,1,1,1,.....,1 order
           ^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^
           |<-DATACOUNT->| |<-DATACOUNT->|
             ch:0 data         ch:1 data
***/

extern WORD    wBuf0[],wBuf1[];
```

2.3 The Testing Functions

2.3.1 P1202_FloatSub2
P1602_FloatSub2
P180X_FloatSub2

- **Description:**

  Compute C=A-B in **float** format, **float=4 bytes floating point number.** This function is provided to test library linkage. **If this subroutine return the correct value, the other subroutine will work properly also.**
- **Syntax:**

  float P1202_FloatSub2(float fA, float fB); *for PCI-1202(H/L) Card.*

  float P1602_FloatSub2(float fA, float fB); *for PCI-1602/1602F Card.*

  float P180X_FloatSub2(float fA, float fB); *for PCI-1800(H/L)/1802(H/L) Card.*
- **Input Parameter :**

  fA : 4 bytes floating point value

  fB : 4 bytes floating point value
- **Return Value :** return=fA-fB
- **Demo Program : DEMO1.C**

2.3.2 P1202_ShortSub2
P1602_ShortSub2
P180X_ShortSub2

- **Description :**

  Compute C=A-B in SHORT format, **SHORT=16 bits signed number.** This function is provided to test library linkage. **If this subroutine return the correct value, the other subroutine will work properly also.**

- **Syntax :**

  short P1202_ShortSub2(Short nA, Short nB); *for PCI-1202(H/L) Card.*

  short P1602_ShortSub2(Short nA, Short nB); *for PCI-1602/1602F Card.*

  short P180X_ShortSub2(Short nA, Short nB); *for PCI-1800(H/L)/1802H(H/L) Card.*

- **Input Parameter :**

  nA : 16 bits value

  nB : 16 bits value

- **Return Value :** return=nA-Nb
- **Demo Program : DEMO1.C**

2.3.3 P1602_GetDriverVersion P180X_GetDriverVersion P1602_GetDriverVersion

- **Description :**

  This subroutine will read the software version number.

- **Syntax :**

  WORD P1202_GetDriverVersion(WORD *wDriverVersion);

  for PCI-1202(H/L) Card.

  WORD P1602_GetDriverVersion(WORK *wDriverVersion);

  for PCI-1602/1602F Card.

  WORD P180X_GetDriverVersion(WORD *wDriverVersion);

  for PCI-1800(H/L)/1802(H/L) Card.

- **Input Parameter :** *wDriverVersion : address of **wDriverVersion**

  wDriverVersion=0x200 → Version 2.0

- **Return Value :**

  NoError : OK.

- **Demo Program : DEMO1.C**

2.4 The M_Functions

2.4.1 P1602_M_FUN_1
P1602_M_FUN_1
P180X_M_FUN_1

- **Description :**

  The P1602_M_FUN_1 will compute the wave form image automatically. (Refer to "PCI-1202/1602/1800/1802 Hardware Manual" chapter-5 for details) (input=AD channel_0, output=DA channel_0). This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_M_FUN_1(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm);
  for PCI-1202(H/L) Card.
  WORD P1602_M_FUN_1(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm);
  for PCI-1602/1602F Card.

  WORD P180X_M_FUN_1(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm);
  for PCI-1800(H/L)/1802(H/L) Card.

● Input Parameter :

wDaFrequency : **DA output frequency = 1.8M/wDaFrequency (pentium 120)**
wDaWave : Number of DA wave form to be output
fDaAmplitude : Amplitude of DA output. NOTE : the hardware J1 must select +/-10V
wAdClock : **AD sampling clock = 8000000/wAdClock** samples/sec
wAdNumber: Number of AD data to be read.
wAdConfig : **A/D input range configuration code.**
fAdBuf[] : the starting address of **fAdBuf** which store the A/D data.
fLowAlarm : low alarm limit. if **fAdBuf[?]< fLowAlarm** → LowAlarm.
fHighAlarm : high alarm limit. if **fAdBuf[?]>fHighAlarm** → HighAlarm.

● Return Value :

0 : OK.
ExceedBoardNumber: invalidate board number.
FindBoardError: no PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
AdControllerError : embedded controller handshake error .
ConfigCodeError : **wAdConfig** configuration code error.
HighAlarm : **fAdBuf[?]>fHighAlarm.**
LowAlarm : **fAdBuf[?]< fLowAlarm.**

● Demo Program : DEMO5.C

2.4.2 P1202_M_FUN_2
P1602_M_FUN_2
P180X_M_FUN_2

- **Description :**

The P1602_M_FUN_2 will **not** compute the wave form image automatically. (Refer to "PCI-1202/1602/1800/1802 Hardware Manual" chapter-5 for details) (input=AD channel_0,output=DA channel_0) This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax :**

WORD P1202_M_FUN_2(WORD wDaFrequency, WORD wDaWave,
WORD wDaBuf[], WORD wAdClock, WORD wAdNumber
WORD wAdConfig, WORD wAdBuf[]);
for PCI-1202(H/L) Card.

WORD P1602_M_FUN_2(WORD wDaFrequency, WORD wDaWave,
WORD wDaBuf[], WORD wAdClock, WORD wAdNumber
WORD wAdConfig, WORD wAdBuf[]);
for PCI-1602/1602F Card.

WORD P180X_M_FUN_2(WORD wDaFrequency, WORD wDaWave,
WORD wDaBuf[], WORD wAdClock, WORD wAdNumber
WORD wAdConfig, WORD wAdBuf[]);
for PCI-1800(H/L)/1802(H/L) Card.

- **Input Parameter :**

wDaFrequency : **DA output frequency = 1.8M/wDaFrequency(pentium 120)**
wDaWave : Number of DA wave form to be output.
wDaBuf[] : The array store the D/A wave form image.
wAdClock : **AD sampling clock = 8000000/wAdClock** samples/sec.
wAdNumber: Number of AD data to be read.
wAdConfig : **A/D input range configuration code.**
wAdBuf[] : the starting address of **fAdBuf** which store the A/D data.

- **Return Value :**

  0 : OK.

  ExceedBoardNumber: invalidate board number.

  FindBoardError: no PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board

  AdControllererror: embedded controller handshake error .

- **Demo Program : DEMO7.C**

  The DA output wave form generator is a **machine dependent** function

  The DA output frequency = **1.8M/wDaFrequency** is machine dependent.

  The testing results are given as follows:

  DA output frequency = 1.8M/wDaFrequency for pentium 120

  DA output frequency = 2.0M/wDaFrequency for pentium 133

The user must test this value before using M_FUN_1 and M_FUN_2.

2.4.3 P1202_M_FUN_3
P1602_M_FUN_3
P180X_M_FUN_3

- **Description :**

  The P1602_M_FUN_3 will compute the wave form image automatically. (Refer to "PCI-1202/1602/1800/1802 Hardware Manual" chapter-5 for details) (input=programable channels,output=DA channel_0) This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax :**

  WORD P1602_M_FUN_3(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[],
  float fLowAlarm, float fHighAlarm)
  for PCI-1202(H/L) Card.

  WORD P1602_M_FUN_3(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[],
  float fLowAlarm, float fHighAlarm)
  for PCI-1602/1602F Card

  WORD P1602_M_FUN_3(WORD wDaFrequency, WORD wDaWave,
  float fDaAmplitude, WORD wAdClock, WORD wAdNumber,
  WORD wChannelStatus[], WORD wAdConfig[], float fAdBuf[],
  float fLowAlarm, float fHighAlarm)
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wDaFrequency : **DA output frequency = 1.8M/wDaFrequency (pentium 120)**
  wDaWave : Number of DA wave form to be output
  fDaAmplitude : Amplitude of DA output. NOTE : the hardware J1 must select +/-10V
  wAdClock : **AD sampling clock = 8000000/wAdClock** samples/sec
  wAdNumber: Number of AD data to be read
  wAdChannel[]: status (1=scan, 0=no scan) code of all 32 channels
  wAdConfig[]: configuration code of all 32 channels
  fAdBuf[] : the starting address of **fAdBuf** which store the A/D data
  fLowAlarm : low alarm limit. if **fAdBuf[?]< fLowAlarm** → LowAlarm
  fHighAlarm : high alarm limit. if **fAdBuf[?]>fHighAlarm** → HighAlarm

- **Return Value :**

  0 : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError: no PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdControllerError : embedded controller handshake error
  HighAlarm : **fAdBuf[?]>fHighAlarm**
  LowAlarm : **fAdBuf[?]< fLowAlarm**

- **Demo Program : DEMO9.C**

DA output frequency = 1.8M/wDaFrequency for pentium 120
DA output frequency = 2.0M/wDaFrequency for pentium 133

The user must test this value before using M_FUN_1 and M_FUN_2.

2.5 The DIO Functions

2.5.1 P1202_Di / P1602_Di / P180X_Di

- **Description :**
  This function will read the 16 bits D/I data from the current active board. Use P1602_ActiveBoard(…) to select the active board.
- **Syntax :**
  WORD P1202_Di(WORD *wDi); *for PCI-1202(H/L) Card*
  WORD P1602_Di(WORD *wDi); *for PCI-1602/1602F Card*
  WORD P180X_Di(WORD *wDi); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter :**
  *wDi : address of **wDi** which store the 16 bits D/I data
- **Return Value :**
  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
- **Demo Program : DEMO1.C**

2.5.2 P1202_Do / P1602_Do / P180X_Do

- **Description :** This function will send the 16 bits D/O data to the current active board. Use P1602_ActiveBoard(…) to select the active boards.
- **Syntax :** WORD P1602_Do(WORD wDo);
- **Input Parameter :**
  wDo : the 16 bits data sent to DO port
- **Return Value :**
  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
- **Demo Program : DEMO1.C**

2.6 The DA Functions

2.6.1 P1202_Da / P1602_Da / P180X_Da

- **Description :**

  This function will send the 12 bits D/A data to the current active board. Use P1602_ActiveBoard(…) to select the active boards.

- **Syntax :**

  WORD P1202_Da(WORD wChannel, WORD wDaVal);
  for PCI-1202(H/L) Card
  WORD P1602_Da(WORD wChannel, WORD wDaVal);
  for PCI-1602/1602F Card
  WORD P180X_Da(WORD wChannel, WORD wDaVal);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wChannel : 0 for channel_0 DA, 1 for channel_1 DA
  wDaVal : 12 bits data sent to DA port. 0=minimum and 4095=maximum. The DA output can be +/- 5V or +/- 10V setting by hardware JP1. The software can't detect the state of JP1. So 4095 maybe +5V or +10V (depend on JP1).

- **Return Value :**

  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  DaChannelError : channel number must be 0 or 1

- **Demo Program : DEMO1.C**

2.7 The AD Fixed-mode Functions

2.7.1 P1202_SetChannelConfig
P1602_SetChannelConfig
P180X_SetChannelConfig

- **Description :**

  This subroutine will set the AD channel & its configuration code. This subroutine will set the active AD channel for P1602_AdPolling, P1602_AdsPolling and P1602_AdsPacer. This function will refer to the current active PCI-1602/1602F board.
  Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_SetChannelConfig(WORD wChannel, WORD wConfig);
  for PCI-1202(H/L) Card, Maximun 16 Channels.

  WORD P1602_SetChannelConfig(WORD wChannel, WORD wConfig);
  for PCI-1602/1602F Card, Maximum 32 Channels.

  WORD P180X_SetChannelConfig(WORD wChannel, WORD wConfig);
  for PCI-1800(H/L)/1802(H/L) Card.
  Maximum 16 Channels for PCI-1800(H/L) Card.
  Maximum 32 Channels for PCI-1802(H/L) Card.

- **Input Parameter :**

  wChannel : AD channel number
  wConfig : Configuration code. Refer to "PCI-1202/1602/1800/1802 Hardware Manual" for details.

- **Return Value :**

  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdControllerError : MagicScan controller hardware handshake error

- **Demo Program : DEMO1.C**

2.7.2 P1202_AdPolling
P1602_AdPolling
P180X_AdPolling

- **Description :**

  This subroutine will perform one AD conversion by polling. The P1602_SetChannelConfig subroutine can be used to change channel or configuration code and the P1602_AdPolling will refer to that condition in later operation. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_AdPolling(float *fAdVal); *for PCI-1202(H/L) Card*
  WORD P1602_AdPolling(float *fAdVal); *for PCI-1602/1602F Card*
  WORD P180X_AdPolling(float *fAdVal); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter :**

  *fAdVal : address of **fAdVal** which store the AD data, this data is automatically computed based on the setting of **P1602_SetChannelConfig**.

- **Return Value :**

  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdPollingTimeOut : hardware timeout error

- **Demo Program : DEMO1.C**

2.7.3 P1202_AdsPolling
P1602_AdsPolling
P180X_AdsPolling

- **Description :**

  This subroutine will perform multiple AD conversions by polling.
  The **P1602_SetChannelConfig** subroutine can be used to change channel or configuration code and the **P1602_AdsPolling** will refer to that condition in later operation. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_AdsPolling(float fAdVal[], WORD wNum);
  for PCI-1202(H/L) Card

  WORD P1602_AdsPolling(float fAdVal[], WORD wNum);
  for PCI-1602/1602F Card

  WORD P180X_AdsPolling(float fAdVal[], WORD wNum);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  fAdVal[]: starting address of AD data buffer, these data will be automatically computed based on the setting of **P1602_SetChannelConfig**.
  wNum: number of AD conversions will be performed.

- **Return Value :**

  NoError: OK
  ExceedBoardNumber: invalidate board number
  FindBoardError: cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdPollingTimeOut: hardware timeout error

- **Demo Program : DEMO1.C**

2.7.4 P1202_AdsPacer
P1602_AdsPacer
P180X_AdsPacer

- **Description :**

  This subroutine will perform multiple AD conversions by pacer trigger. The **P1602_SetChannelConfig** subroutine can be used to change channel or configuration code and the **P1602_AdsPacer** will refer to that condition in later operation. The hardware pacer will generate trigger signal to AD converter periodically. So these AD data can be used to reconstruct the wave form of analog input. The **P1602_AdsPolling** is controlled by software polling , so the AD conversion operation will be interrupted by system OS. **It is recommended to use *P1602_AdsPacer* if the input wave form reconstruction is needed.** This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for PCI-1202(H/L) Card

  WORD P1602_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for PCI-1602F Card

  WORD P180X_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  fAdVal[] : starting address of AD data buffer, these data will be automatically computed based on the setting of **P1602_SetChannelConfig**.
  wNum : number of AD conversions will be performed.
  wSample : **AD sampling rate = 8M/wSample.**

For PCI-1202 Card

wSample=73 → Sampling rate=8M/73=110K → **Maximum for PCI-1202 Card**
wSample=80 → Sampling rate=8M/80=100K

For PCI-1602F/1602 Card

wSample=40 → Sampling rate=8M/40=200K → **Maximum for PCI-1602F Card**
wSample=80 → Sampling rate=8M/80=100K → **Maximum for PCI-1602 Card**

For PCI-1800/1802 Card

wSample=24 → Sampling rate=8M/24=333K → **Maximum**
wSample=80 → Sampling rate=8M/80=100K

- **Return Value :**

  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdPollingTimeOut : hardware timeout error

- **Demo Program : DEMO1.C**

P1602_SetChannelConfig
P1602_AdPolling
P1602_AdsPollng
P1602_AdsPacer

Fix channel AD conversion mode

2.8 The MagicScan Functions

2.8.1 P1202_ClearScan
P1602_ClearScan
P180X_ClearScan

- **Description :**

  This subroutine will initialize the MagicScan controller to the Initial state. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :** WORD P1202_ClearScan(); *for PCI-1202(H/L) Card*
  WORD P1602_ClearScan(); *for PCI-1602/1602F Card*
  WORD P180X_ClearScan(); *for PCI-1800(H/L)/1802(H/L) Card*
- **Input Parameter :** void
- **Return Value :**

  NoError : OK

  ExceedBoardNumber: invalidate board number

  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
- **Demo Program : DEMO11.C**

2.8.2 P1602_StartScan
P1602_StartScan
P1602_StartScan

- **Description :**

  This subroutine will start the MagicScan operation. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  WORD P1202_StartScan(WORD wSampleRate, WORD wNum);
  for PCI-1202(H/L) Card
  WORD P1602_StartScan(WORD wSampleRate, WORD wNum);
  for PCI-1602/1602F Card
  WORD P180X_StartScan(WORD wSampleRate, WORD wNum);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wSampleRate : **AD sampling rate = 8M/wSampleRate.**

For PCI-1202 Card

wSampleRate=73 → Sampling rate=8M/73=110K → **Maximum**
wSampleRate=80 → Sampling rate=8M/80=100K

For PCI-1602F/1602 Card

wSampleRate=40 → Sampling rate=8M/40=200K → **Maximum for PCI-1602FCard**
wSampleRate=80 → Sampling rate=8M/80=100K → **Maximum for PCI-1602 Card**

For PCI-1800/1802 Card

wSampleRate=24 → Sampling rate=8M/24=333K → **Maximum**
wSampleRate=80 → Sampling rate=8M/80=100K

wNum : Number of **MagicScan cycle** to perform

- **Return Value :**

  NoError : OK
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdControllerError : MagicScan controller hardware handshake error

- **Demo Program : DEMO11.C**

2.8.3 P1202_ReadMagicScanResult P1602_ReadMagicScanResult P180X_ReadMagicScanResult

- **Description :**

  This subroutine will read the alarm result of the MagicScan operation. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  void P1202_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD dwLowAlarm);

  for PCI-1202(H/L) Card

  void P1602_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD *dwLowAlarm);

  for PCI-1602/602F Card

  void P180X_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD *dwLowAlarm);

  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  *dwLowAlarm: address of **dwLowAlarm** which store the MagicScan alarm status (bit_0 → channel_0, bit_31 → channel_31, 0=no alarm, 1=low alarm)

  *dwHighAlarm: address of **dwHighAlarm** which store the MagicScan alarm tatus (bit_0 → channel_0, bit_31 → channel_31, 0=no alarm, 1=high alarm)

- **Return Value** : void

- **Demo Program : DEMO11.C**

dwLowAlarm	→ 32 bits corresponding to 32 channels
	→ 0 = no low alarm
	→ 1 = is low alarm
dwLowAlarm=0	→ all channels OK, no low alarm
dwLowAlarm=1	→ channel_0 is low alarm, others are OK
dwLowAlarm=3	→ channel_0 and channel_1 are low alarm, others are OK

dwHighAlarm	→ 32 bits corresponding to 32 channels
	→ 0 = no high alarm
	→ 1 = is high alarm
dwHighAlarm=0	→ all channels OK, no high alarm
dwHighAlarm=1	→ channel_0 is high alarm, others are OK
dwHighAlarm=3	→ channel_0 and channel_1 are high alarm, others are OK

2.8.4 P1202_AddToScan
P1602_AddToScan
P180X_AddToScan

- **Description :**

  This subroutine will add one channel to the **MagicScan circular queue**. This function will refer to the current active PCI-1602/1602F board.Use the P1602_ActiveBoard(…) to select the
  active board.

- **Syntax :**

  void P1202_AddToScan(WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for PCI-1202(H/L) Card

  void P1602_AddToScan(WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for PCI-1602/1602F Card

  void P180X_AddToScan(WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wAdChannel : AD channel number
  wConfig : the configuration code
  wAverage : the factor of digital average filter
  wLowAlarm : 16 bits low alarm data
  wHighAlarm : 16 bits high alarm data
  wAlarmType : 0=no alarm, 1=high alarm, 2=low alarm, 3=in-alarm, 4=out-alarm

- **Return Value :** void

  NoError : Ok
  ExceedBoardNumber: invalidate board number
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdChannelError : invalidate AD channel
  AlarmTypeError : only 0/1/2/3/4 are validate
  AdControllerError : MagicScan controller hardware handshake error

- **Demo Program : DEMO11.C**

2.8.5 P1202_SaveScan
P1602_SaveScan
P180X_SaveScan

- **Description :**

  This subroutine will specify the starting address of AD data buffer for MagicScan. This function will refer to the current active PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax :**

  void P1202_SaveScan(WORD wAdChannel, WORD wBuf[]);

  for PCI-1202(H/L) Card

  void P1602_SaveScan(WORD wAdChannel, WORD wBuf[]);

  for PCI-1602/1602F Card

  void P180X_SaveScan(WORD wAdChannel, WORD wBuf[]);

  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wAdChannel : Scan number in the scan queue. (Note: not the A/D channel number.)

  wBuf : starting address of AD data buffer for channel specified in **wAdChannel**

- **Return Value** :

  NoError : Ok

  ExceedBoardNumber: invalidate board number

  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board

  AdChannelError : invalidate AD channel

- **Demo Program :.DEMO11.C**

2.9 The Pulg&Play Functions

2.9.1 P1202_DriverInit
P1602_DriverInit
P180X_DriverInit

- **Description :**

  This function will detect all the PCI-1602/1602F boards installed in the system.
  This function must be called once before the other functions are called.

- **Syntax :**

  WORD P1202_DriverInit(WORD *wTotalBoard); *for PCI-1202(H/L) Card*
  WORD P1602_DriverInit(WORD *wTotalBoard); *for PCI-1602/1602F Card*
  WORD P180X_DriverInit(WORD *wTotalBoard); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter :**

  *wTotalBoard: address of **wTotalBoard**
  wTotalBoard=1 → one PCI-1602/1602F card in the system
  wTotalBoard=n → n*PCI-1602/1602F cards in the system

- **Return Value :**

  NoError : OK
  NoFoundBoard: detect no PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) Card

- **Demo Program : All DEMO program.**

2.9.2 P1202_DriverClose
P1602_DriverClose
P180X_DriverClose

- **Description :**

  Return all resources to system. This function must be called once before program is terminated.

- **Syntax :**

  WORD P1202_DriverClose(void); *for PCI-1202(H/L) Card*
  WORD P1602_DriverClose(void); *for PCI-1602/1602F Card*
  WORD P180X_DriverClose(void); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter :** void

- **Return Value :** NoError

- **Demo Program : All DEMO program.**

2.9.3 P1202_GetConfigAddressSpace P1602_GetConfigAddressSpace P180X_GetConfigAddressSpace

- **Description :**

  Get the I/O address of PCI-1602/1602F board n. This function is for debug. It is not necessary to call this function.

- **Syntax :**

  WORD P1202_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
  for PCI-1202(H/L) Card

  WORD P1602_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
  for PCI-1602/1602F Card

  WORD P180X_GetConfigAddressSpace(WORD wBoardNo, WORD *wAddrTimer,WORD *wAddrCtrl, WORD *wAddrDio, WORD *wAddrAdda);
  for PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter :**

  wBoardNo: PCI-1602/1602F board number
  wAddrTimer, wAddrCtrl, wAddrDio, wAddrAdda: refer to "PCI-1202/1602/1800/1802 Hardware manual" Chapter-3 for details.

- **Return Value :**

  NoError : OK
  FindBoardError: handshake check error
  ExceedBoardError: wBoardNo is invalidate

- **Demo Program : ALL DEMO program**

2.9.4 P1202_WhichBoardActive
P1602_WhichBoardActive
P180X_WhichBoardActive

- **Description:**
  Return the board number of the active board.

- **Syntax:**
  WORD P1202_WhichBoardActive(void); *for PCI-1202(H/L) Card*
  WORD P1602_WhichBoardActive(void); *for PCI-1602/1602F Card*
  WORD P180X_WhichBoardActive(void); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter:** void

- **Return Value:** board number of the active board.

- **Demo Program: DEMO1.C**

2.9.5 P1602_ActiveBoard
P1602_ActiveBoard
P180X_ActiveBoard

- **Description:**

  This function will active one of the PCI-1602/1602F boards installed in the system. This function must call once before the D/I/O, A/D, D/A functions are called.

- **Syntax:**

  WORD P1202_ActiveBoard(WORD wBoardNo); *for PCI-1202(H/L) Card*
  WORD P1602_ActiveBoard(WORD wBoardNo); *for PCI-1602/1602F Card*
  WORD P180X_ActiveBoard(WORD wBoardNo); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter:**

  wBoardNo: board number

- **Return Value :**

  NoError : OK
  ExceedBoardError: wBoardNo is invalidate

- **Demo Program : All DEMO program.**

The P1602_ActiveBoard(…) will take effect on all functions except the following:

1. P1602_FloatSub2
2. P1602_ShortSub2
3. P1602_GetDriverVersion
4. P1602_DriveInit
5. P1602_DriveClose
6. P1602_GetConfigAddressSpace
7. P1602_Card0_StartScan
8. P1602_Card0_ReadData
9. P1602_Card0_Stop

2.10 The Continuous Capture Functions

2.10.1 P1202_Card0_StartScan
P1602_Card0_StartScan
P180X_Card0_StartScan

- **Description:**

  This subroutine will start the continuous capture function. Refer to "PCI-1202/1602/1800/1802 Hardware User Manual chapter-6 for details"

- **Syntax :** WORD P1602_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[], WORD wChanelConfig[], WORD wCount);

- **Input Parameter :**

  wSampleRate : **AD sampling rate = 8M/wSampleRate:**

  For PCI-1202 Card:

  wSampleRate=73 → Sampling rate=8M/73=110K→**Maximum**
  wSampleRate=80 → Sampling rate=8M/80=100K

  For PCI-1602F/1602 Card:

  wSampleRate=40 →Sampling rate=8M/40=200K→**Maximum for PCI-1602F Card**
  wSampleRate=80 →Sampling rate=8M/80=100K→**Maximum for PCI-1602 Card**

  For PCI-1800/1802 Card:

  wSampleRate=24 →Sampling rate=8M/24=333K→**Maximum**
  wSampleRate=80 →Sampling rate=8M/80=100K

  wChannelStatus[]: (0=no scan, 1=scan) for each channel
  wChannelConfig[]: configuration code for each channel
  wCount: number of A/D data for each scan channel

- **Return Value :**

  NoError : OK
  FindBoardError : cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
  AdControllerError : MagicScan controller hardware handshake error

- **Demo Program : DEMO13.C**

2.10.2 P1202_Card0_ReadData
P1602_Card0_ReadData
P180X_Card0_ReadData

- **Description :**

  This subroutine will read the data of continuous capture function.
- **Syntax :**

  WORD P1202_Card0_ReadData(void); *for PCI-1202(H/L) Card*

  WORD P1602_Card0_ReadData(void); *for PCI-1602/1602F Card*

  WORD P180X_Card0_ReadData(void); *for PCI-1800(H/L)/1802(H/L) Card*
- **Input Parameter :**void
- **Return Value :**

  NoError: data is ready

  TimeOut: data not ready

  FifoOverflow: FIFO overflow
- **Demo Program : DEMO13.C**

2.10.3 P1202_Card0_Stop
P1602_Card0_Stop
P180X_Card0_Stop

- **Description :**

  This subroutine will stop the continuous capture function.
- **Syntax :**

  void P1202_Card0_Stop(void); *for PCI-1202(H/L) Card*

  void P1602_Card0_Stop(void); *for PCI-1602/1602F Card*

  void P180X_Card0_Stop(void); *for PCI-1800(H/L)/1802(H/L) Card*
- **Input Parameter :** void
- **Return Value :** void
- **Demo Program : DEMO13.C**

2.11 The Other Functions

2.11.1 P1202_DelayUs
P1602_DelayUs
P180X_DelayUs

- **Description :**

  This is a **machine independent timer.** This function can be used to delay the **settling time** or used as a **general purposed machine independent timer**.

  This function will refer to the current active PCI-1602/1602F board. Use the 1602_ActiveBoard(….) to select the active board.

- **Syntax :**

  void P1202_DelayUs(WORD wDelayUs); *for PCI-1202(H/L) Card*

  void P1602_DelayUs(WORD wDelayUs); *for PCI-1602/1602F Card*

  void P180X_DelayUs(WORD wDelayUs); *for PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter :**

  wDelayUs : number of us to delay, 8191 Max

  wDelayUs=1 → delay 1 us

  wDelayUs=1000 → delay 1000 us = 1 ms

  wDelayUs=8191 → delay 8191 us = 8.191 ms (maximum delay)

  wDelayUs=8192 → invalidate delay (will return error)

- **Return Value** :

  NoError : OK

  ExceedBoardNumber: invalidate board number

  FindBoardError :cannot find the PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board

  InvalidateDelay : **wDelayUs** > 8191

- **Demo Program : DEMO1.C**

- **Long Time Delay :**

```
WORD DelayMs(WORD dwDelayMs) // maximum delay=4294967.295 sec
   {
    WORD dwDelay,dwRetVal

   dwRetVal=0;
      for  ( dwDelay=0; dwDelay<dwDelayMs; dwDelay++ )
           dwRetVal += P1602_DelayUs( 1000 );
      return( dwRetVal );
   }
```

• Demo Programs

There are about 19 demo program given as follows:

- demo1: one board, D/I/O test, D/A test, A/D polling&pacer trigger test, general test
- demo2: two board, same as demo1
- demo3: one board, perform 32 channels of A/D conversion by software trigger(polling)
- demo4: two board, same as demo3
- demo5: one board, M_function_1 demo
- demo6: two board, same as demo5
- demo7: one board, M_function_2 demo
- demo8: two board, same as demo7
- demo9: one board, M_function_3 demo
- demo10: two board, same as demo9
- demo11: one board, MagicScan demo
- demo12: two board, same as demo11
- demo13: one board, continuous capture demo
- demo14: two board, continuous capture demo (Windows 95/98/NT only)
- demo15: all installed board, D/I/O test for board number identification
- demo16: one board, performance evaluation demo
- demo17: one board, MagicScan demo, scan sequence: 4→3→5
- demo18: one board, MagicScan demo, scan 32 channel, show channel 0/1/15/16/17
- demo19: one board, A/D calibration.

Refer to the company CD ROM for details.