OME-PCI-1202/1602/1800/1802 BOARDS
DOS Software User’s Manual
It is the policy of OMEGA to comply with all worldwide safety and EMC/EMI regulations that apply. OMEGA is constantly pursuing certification of its products to the European New Approach Directives. OMEGA will add the CE mark to every appropriate device upon certification.

The information contained in this document is believed to be correct, but OMEGA Engineering, Inc. accepts no liability for any errors it contains, and reserves the right to alter specifications without notice.

**WARNING:** These products are not designed for use in, and should not be used for, patient-connected applications.
# Table of Contents

1 INTRODUCTION ................................................................................................................... 3

1.1 SOFTWARE INSTALLATION ........................................................................................................ 5

1.2 COMPIL & LINK USING MSC ...................................................................................................... 8

1.3 COMPIL & LINK USING TC ........................................................................................................ 9

1.4 COMPIL & LINK USING BC ....................................................................................................... 10

2 C LANGUAGE LIBRARY ............................................................................................................. 11

2.1 THE CONFIGURATION CODE TABLE .......................................................................................... 14

2.2 OME-PCI-1202/1602/1800/1802 HEADER FILE ........................................................................ 16

2.3 THE TESTING FUNCTIONS ........................................................................................................... 25

2.4 THE M_FUNCTIONS ...................................................................................................................... 27

2.5 THE DIO FUNCTIONS ................................................................................................................... 33

2.6 THE DA FUNCTIONS .................................................................................................................... 34

2.7 THE AD FIXED-MODE FUNCTIONS ............................................................................................. 35

2.8 THE MAGICSCAN FUNCTIONS .................................................................................................... 40

2.9 THE PLUG & PLAY FUNCTIONS ................................................................................................ 45

2.10 THE CONTINUOUS CAPTURE FUNCTIONS ............................................................................... 50

2.11 THE OTHER FUNCTIONS .......................................................................................................... 52

3 DEMO PROGRAMS .................................................................................................................. 54
1. Introduction

The **OME-PCI-1202/1602/1800/1802 DOS driver** is a collection of library for DOS application. This library can be called by TC 2.x, BC 3.x and MSC 5.x. This 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 is given in the companion floppy disk. There are 19 demo programs given in the companion CD-ROM or floppy disk. The user can print out these files for reference.

The hardware I/O control register & its function are given in the “OME-PCI-1202/1602/1800/1802 Hardware User’s Manual”. The user should refer to “OME-PCI-1202/1602/1800/1802 Hardware User’s Manual” for more information.
• 12-bit DAC Resolution for OME-PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L)

<table>
<thead>
<tr>
<th>Data Input</th>
<th>Analog Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MSB</strong> <strong>LSB</strong></td>
<td></td>
</tr>
<tr>
<td>1111 1111 1111</td>
<td>+Vref (2047/2048)</td>
</tr>
<tr>
<td>1000 0000 0001</td>
<td>+Vref (1/2048)</td>
</tr>
<tr>
<td>1000 0000 0000</td>
<td>0 Vlots</td>
</tr>
<tr>
<td>0111 1111 1111</td>
<td>-Vref (1/2048)</td>
</tr>
<tr>
<td>0000 0000 0000</td>
<td>-Vref (2048/2048)</td>
</tr>
</tbody>
</table>

Bipolar Output Code.
Vref is +5V or +10V selected by J1.

• 12-bit ADC Resolution for OME-PCI-1202(H/L)/1800(H/L)/1802(H/L)

<table>
<thead>
<tr>
<th>Analog Input</th>
<th>Digital Output Binary Code</th>
<th>Hex Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>+9.99512V</td>
<td>0111 1111 1111</td>
<td>7FF</td>
</tr>
<tr>
<td>0V</td>
<td>0000 0000 0000</td>
<td>000</td>
</tr>
<tr>
<td>-4.88mV</td>
<td>1111 1111 1111</td>
<td>FFF</td>
</tr>
<tr>
<td>-10V</td>
<td>1000 0000 0000</td>
<td>800</td>
</tr>
</tbody>
</table>

Input Voltages and Output Codes.

• 16-bit ADC Resolution for OME-PCI-1602/1602F Card

<table>
<thead>
<tr>
<th>Analog Input</th>
<th>Digital Output Binary Code</th>
<th>Hex Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>+9.999695V</td>
<td>0111 1111 1111 1111</td>
<td>7FFFF</td>
</tr>
<tr>
<td>0V</td>
<td>0000 0000 0000 0000</td>
<td>0000</td>
</tr>
<tr>
<td>-305uv</td>
<td>1111 1111 1111 1111</td>
<td>FFFF</td>
</tr>
<tr>
<td>-10V</td>
<td>1000 0000 0000 0000</td>
<td>8000</td>
</tr>
</tbody>
</table>

Input Voltages and Output Codes.
1.1 Software Installation

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

Refer to CD ROM readme.txt file.

1.1.1 Software content for OME-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

:  
:  

Version 3.1 Date: 1999/08
1.1.2 Software content for OME-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.............demo19 program source
...\1602\TC\DEMO>DEMO19.PRJ...........demo19 project file
...\1602\TC\DEMO>DEMO19.EXE..........demo19 execution file
### 1.1.3 Software content for OME-PCI-1800(H/L)/1802(H/L)

<table>
<thead>
<tr>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>\180X\TC&gt;</code></td>
<td>For Turbo C 2.0 or later user</td>
</tr>
<tr>
<td><code>\180X\BC&gt;</code></td>
<td>For Borland C++ 3.0 later user</td>
</tr>
<tr>
<td><code>\180X\MSC&gt;</code></td>
<td>For Microsoft C 5.1 later user</td>
</tr>
<tr>
<td><code>\180X\TC\DRIVER&gt;</code></td>
<td><code>P180XDR.H</code></td>
</tr>
<tr>
<td><code>\180X\TC\DRIVER&gt;</code></td>
<td><code>P180XTCL.C</code></td>
</tr>
<tr>
<td><code>\180X\TC\DRIVER&gt;</code></td>
<td><code>IOPORT.C</code></td>
</tr>
<tr>
<td><code>\180X\TC\LIB&gt;</code></td>
<td><code>P180X.H</code></td>
</tr>
<tr>
<td><code>\180X\TC\LIB&gt;</code></td>
<td><code>P180XTCL.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\LIB&gt;</code></td>
<td><code>P180XTCH.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\LIB&gt;</code></td>
<td><code>IOPORTL.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\LIB&gt;</code></td>
<td><code>IOPORTH.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>P180XTCL.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>IOPORTL.LIB</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>P180X.H</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO1.C</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO1.PRJ</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO1.EXE</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO2.C</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO19.C</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO19.PRJ</code></td>
</tr>
<tr>
<td><code>\180X\TC\DEMO&gt;</code></td>
<td><code>DEMO19.EXE</code></td>
</tr>
</tbody>
</table>
1.2 Compile & link using MSC

- The including file is **P1602.H**
- There are two OME-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 OME-PCI-1202(H/L)/1800(H/L)/1802(H/L) card.
1.3 Compile & link using TC

- The including file is **P1602.H**
- There are two OME-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 compile & link

- Note: The steps are the same for OME-PCI-1202(H/L)/1800(H/L)/1802(H/L) card.
1.4 Compile & link using BC

- The including file is **P1602.H**
- There are two OME-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 compile & link

- **Note:** The steps are the same for OME-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 OME-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

The functions of MagicScan mode are given as follows:

1. P1202_ClearScan
2. P1202_StartScan
3. P1202_AddToScan
4. P1202_SaveScan
5. P1202_ReadMagicScanResult

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 OME-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

The functions of MagicScan mode are given as follows:

6. P1602_ClearScan
7. P1602_StartScan
8. P1602_AddToScan
9. P1602_SaveScan
10. P1602_ReadMagicScanResult

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

Data in float format

Data in 16 bits HEX format
• For OME-PCI-1800(H/L)/1802(H/L) Card

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

<table>
<thead>
<tr>
<th>9. P180X_SetChannelConfig</th>
<th>Data in float format</th>
</tr>
</thead>
<tbody>
<tr>
<td>10. P180X_AdPoling</td>
<td></td>
</tr>
<tr>
<td>11. P180X_AdsPolling</td>
<td></td>
</tr>
<tr>
<td>12. P180X_AdsPacer</td>
<td></td>
</tr>
</tbody>
</table>

The functions of MagicScan mode are given as follows:

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

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 OME-PCI-1602/1602F Configuration Code Table

<table>
<thead>
<tr>
<th>Bipolar/Unipolar</th>
<th>Input Signal Range</th>
<th>Gain</th>
<th>Settling Time</th>
<th>Configuration Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar</td>
<td>+/- 10V</td>
<td>1</td>
<td>3 us</td>
<td>0</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 5.0V</td>
<td>2</td>
<td>3 us</td>
<td>1</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 2.5V</td>
<td>4</td>
<td>3 us</td>
<td>2</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1.25V</td>
<td>8</td>
<td>3 us</td>
<td>3</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Bipolar/Unipolar</th>
<th>Input Signal Range</th>
<th>Gain</th>
<th>Settling Time</th>
<th>Configuration Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar</td>
<td>+/- 5V</td>
<td>1</td>
<td>3 us</td>
<td>0x00</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 2.5V</td>
<td>2</td>
<td>3 us</td>
<td>0x01</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1.25V</td>
<td>4</td>
<td>3 us</td>
<td>0x02</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.625V</td>
<td>8</td>
<td>3 us</td>
<td>0x03</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 10V</td>
<td>0.5</td>
<td>3 us</td>
<td>0x04</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 5V</td>
<td>1</td>
<td>3 us</td>
<td>0x05</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 2.5V</td>
<td>2</td>
<td>3 us</td>
<td>0x06</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1.25V</td>
<td>4</td>
<td>3 us</td>
<td>0x07</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 10V</td>
<td>1</td>
<td>3 us</td>
<td>0x08</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 5V</td>
<td>2</td>
<td>3 us</td>
<td>0x09</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 2.5V</td>
<td>4</td>
<td>3 us</td>
<td>0x0A</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 1.25V</td>
<td>8</td>
<td>3 us</td>
<td>0x0B</td>
</tr>
</tbody>
</table>
## 2.1.3 OME-PCI-1202H/1800H/1802H Configuration Code Table

<table>
<thead>
<tr>
<th>Bipolar/Unipolar</th>
<th>Input Signal Range</th>
<th>Gain</th>
<th>Settling Time</th>
<th>Configuration Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar</td>
<td>+/- 5V</td>
<td>1</td>
<td>23 us</td>
<td>0x10</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.5V</td>
<td>10</td>
<td>28 us</td>
<td>0x11</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.05V</td>
<td>100</td>
<td>140 us</td>
<td>0x12</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.005V</td>
<td>1000</td>
<td>1300 us</td>
<td>0x13</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 10V</td>
<td>0.5</td>
<td>23 us</td>
<td>0x14</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1V</td>
<td>5</td>
<td>28 us</td>
<td>0x15</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.1V</td>
<td>50</td>
<td>140 us</td>
<td>0x16</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.01V</td>
<td>500</td>
<td>1300 us</td>
<td>0x17</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 10V</td>
<td>1</td>
<td>23 us</td>
<td>0x18</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 1V</td>
<td>10</td>
<td>28 us</td>
<td>0x19</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 0.1V</td>
<td>100</td>
<td>140 us</td>
<td>0x1A</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 0.01V</td>
<td>1000</td>
<td>1300 us</td>
<td>0x1B</td>
</tr>
</tbody>
</table>
2.2 OME-PCI-1202/1602/1800/1802

Header File

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

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

extern WORD wBuf0[], wBuf1[];
2.2.2 P1602.H for OME-PCI-1602/1602F Card

```c
#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_AdsPolling(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 scanned data is stored as
    1,0,1,0,1,0,1,0... order
    wBuf1[] -> the scanned data is stored as
    0,0,0,0,...,0,1,1,1,...,1 order
    ^^^^^^^^^^^^^^^ ^^^^^^^^^^^^^^^
    |<-DATACOUNT->| |<-DATACOUNT->|
    ch:0 data       ch:1 data

    ***/

extern WORD wBuf0[], wBuf1[];
2.2.3 P180X.H for OME-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_FloatSub2(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 P180X_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[]),
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 scanned data is stored as
1,0,1,0,1,0,1,0...
1 order
wBuf1[] -> the scanned 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_FlotSub2
       P1602_FlotSub2
       P180X_FlotSub2

- **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_FlotSub2(float fA, float fB);   for OME-PCI-1202(H/L) Card.
  float P1602_FlotSub2(float fA, float fB);   for OME-PCI-1602/1602F Card.
  float P180X_FlotSub2(float fA, float fB);   for OME-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); \( \text{for OME-PCI-1202(H/L) Card.} \)
short P1602_ShortSub2(Short nA, Short nB); \( \text{for OME-PCI-1602/1602F Card.} \)
short P180X_ShortSub2(Short nA, Short nB); \( \text{for OME-PCI-1800(H/L)/1802H(H/L) Card.} \)

**Input Parameter:**

- nA : 16 bits value
- nB : 16 bits value

**Return Value:** return=nA-Nb

**Demo Program:** DEMO1.C

---

**2.3.3 P1602_GetDriverVersion**

**Description:**

This subroutine will read the software version number.

**Syntax:**

WORD P1202_GetDriverVersion(WORD *wDriverVersion);
\( \text{for OME-PCI-1202(H/L) Card.} \)

WORD P1602_GetDriverVersion(WORK *wDriverVersion);
\( \text{for OME-PCI-1602/1602F Card.} \)

WORD P180X_GetDriverVersion(WORK *wDriverVersion);
\( \text{for OME-PCI-1800(H/L)/1802H(H/L) Card.} \)

**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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax:**

  ```c
  WORD P1202_M_FUN_1(WORD wDaFrequency, WORD wDaWave, float fDaAmplitude, WORD wAdClock, WORD wAdNumber, WORD wAdConfig, float fAdBuf[], float fLowAlarm, float fHighAlarm);
  for OME-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 OME-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 OME-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.

**Return Value:**

- 0: OK.
- ExceedBoardNumber: invalidate board number.
- FindBoardError: no OME-PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board
- AdControllerError: embedded controller handshake error.
- ConfigCodeError: wAdConfig configuration code error.
- 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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax**:

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

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

  ```c
  WORD    P180X_M_FUN_2(WORD wDaFrequency, WORD wDaWave,  
                   WORD wDaBuf[], WORD wAdClock, WORD wAdNumber  
                   WORD wAdConfig, WORD wAdBuf[]);
  for OME-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 OME-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:

<table>
<thead>
<tr>
<th>DA output frequency = 1.8M/wDaFrequency for pentium 120</th>
</tr>
</thead>
<tbody>
<tr>
<td>DA output frequency = 2.0M/wDaFrequency for pentium 133</td>
</tr>
</tbody>
</table>

**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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard( ... ) to select the active board.

- **Syntax:**
  ```c
  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 OME-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 OME-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 OME-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 OME-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 OME-PCI-1202(H/L) Card`
  
  - `WORD P1602_Di(WORD *wDi);`  
  - `for OME-PCI-1602/1602F Card`
  
  - `WORD P180X_Di(WORD *wDi);`  
  - `for OME-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 OME-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 OME-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 OME-PCI-1202(H/L) Card
  
  WORD P1602_Da(WORD wChannel, WORD wDaVal);
  for OME-PCI-1602/1602F Card
  
  WORD P180X_Da(WORD wChannel, WORD wDaVal);
  for OME-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 OME-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 OME-PCI-1602/1602F board.
  Use the P1602_ActiveBoard(...) to select the active board.

- **Syntax:**
  WORD P1202_SetChannelConfig(WORD wChannel, WORD wConfig);
  for OME-PCI-1202(H/L) Card, Maximum 16 Channels.

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

  WORD P180X_SetChannelConfig(WORD wChannel, WORD wConfig);
  for OME-PCI-1800(H/L)/1802(H/L) Card.
  Maximum 16 Channels for OME-PCI-1800(H/L) Card.
  Maximum 32 Channels for OME-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 OME-PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L)
  board
  AdControllerError : MagicScan controller hardware handshake error
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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

**Syntax:**

```c
WORD P1202_AdPolling(float *fAdVal);    // for OME-PCI-1202(H/L) Card
WORD P1602_AdPolling(float *fAdVal);    // for OME-PCI-1602/1602F Card
WORD P180X_AdPolling(float *fAdVal);    // for OME-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 OME-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 OME-PCI-1602/1602F board. Use the
P1602_ActiveBoard(…) to select the active board.

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

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

WORD P180X_AdsPolling(float fAdVal[], WORD wNum);
for OME-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 OME-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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax:**
  ```c
  WORD P1202_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for OME-PCI-1202(H/L) Card
  
  WORD P1602_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for OME-PCI-1602F Card
  
  WORD P180X_AdsPacer(float fAdVal[], WORD wNum, WORD wSample);
  for OME-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 OME-PCI-1202 Card

<table>
<thead>
<tr>
<th>wSample</th>
<th>Sampling rate</th>
<th>Max Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>8M/73</td>
<td>110K</td>
</tr>
<tr>
<td>80</td>
<td>8M/80</td>
<td>100K</td>
</tr>
</tbody>
</table>

For OME-PCI-1602F/1602 Card

<table>
<thead>
<tr>
<th>wSample</th>
<th>Sampling rate</th>
<th>Max Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>8M/40</td>
<td>200K</td>
</tr>
<tr>
<td>80</td>
<td>8M/80</td>
<td>100K</td>
</tr>
</tbody>
</table>

For OME-PCI-1800/1802 Card

<table>
<thead>
<tr>
<th>wSample</th>
<th>Sampling rate</th>
<th>Max Sampling Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>8M/24</td>
<td>333K</td>
</tr>
<tr>
<td>80</td>
<td>8M/80</td>
<td>100K</td>
</tr>
</tbody>
</table>

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

- **Demo Program : DEMO1.C**

```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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(….) to select the active board.

- **Syntax:**
  - WORD P1202_ClearScan(); *for OME-PCI-1202(H/L) Card*
  - WORD P1602_ClearScan(); *for OME-PCI-1602/1602F Card*
  - WORD P180X_ClearScan(); *for OME-PCI-1800(H/L)/1802(H/L) Card*

- **Input Parameter:** void

- **Return Value:**
  - NoError : OK
  - ExceedBoardNumber: invalidate board number
  - FindBoardError : cannot find the OME-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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(...) to select the active board.

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

- **Input Parameter**:
  - `wSampleRate`: AD sampling rate = 8M/wSampleRate
  
  **For OME-PCI-1202 Card**
  
<table>
<thead>
<tr>
<th>wSampleRate</th>
<th>Sampling rate</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>73</td>
<td>8M/73=110K</td>
<td>Maximum</td>
</tr>
<tr>
<td>80</td>
<td>8M/80=100K</td>
<td></td>
</tr>
</tbody>
</table>

  **For OME-PCI-1602F/1602 Card**
  
<table>
<thead>
<tr>
<th>wSampleRate</th>
<th>Sampling rate</th>
<th>Maximum for OME-PCI-1602FCard</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>8M/40=200K</td>
<td>Maximum for OME-PCI-1602 Card</td>
</tr>
<tr>
<td>80</td>
<td>8M/80=100K</td>
<td>Maximum for OME-PCI-1602 Card</td>
</tr>
</tbody>
</table>

  **For OME-PCI-1800/1802 Card**
  
<table>
<thead>
<tr>
<th>wSampleRate</th>
<th>Sampling rate</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>8M/24=333K</td>
<td>Maximum</td>
</tr>
<tr>
<td>80</td>
<td>8M/80=100K</td>
<td></td>
</tr>
</tbody>
</table>

  - `wNum`: Number of MagicScan cycle to perform

- **Return Value**:
  - `NoError`: OK
  - `ExceedBoardNumber`: invalidate board number
  - `FindBoardError`: cannot find the OME-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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax:**
  ```
  void P1202_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD dwLowAlarm);
  for OME-PCI-1202(H/L) Card
  void P1602_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD *dwLowAlarm);
  for OME-PCI-1602/602F Card
  void P180X_ReadMagicScanResult(DWORD *dwHighAlarm, DWORD *dwLowAlarm);
  for OME-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 status
  (bit_0 ➔ channel_0, bit_31 ➔ channel_31, 0=no alarm, 1=high alarm)

- **Return Value**: void

- **Demo Program**: void 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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(…) to select the active board.

- **Syntax:**
  ```c
  void P1202_AddToScan( WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for OME-PCI-1202(H/L) Card
  
  void P1602_AddToScan( WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for OME-PCI-1602/1602F Card
  
  void P180X_AddToScan( WORD wAdChannel, WORD wConfig, WORD wAverage, WORD wLowAlarm, WORD wHighAlarm, WORD wAlarmType);
  for OME-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 OME-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 OME-PCI-1602/1602F board. Use the P1602_ActiveBoard(...) to select the active board.

- **Syntax:**
  
  ```c
  void P1202_SaveScan(WORD wAdChannel, WORD wBuf[]);
  for OME-PCI-1202(H/L) Card
  void P1602_SaveScan(WORD wAdChannel, WORD wBuf[]);
  for OME-PCI-1602/1602F Card
  void P180X_SaveScan(WORD wAdChannel, WORD wBuf[]);
  for OME-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 OME-PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board  
  - AdChannelError : invalidate AD channel

- **Demo Program:** DEMO11.C
2.9 The Plug & Play Functions

2.9.1 P1202_DriverInit
       P1602_DriverInit
       P180X_DriverInit

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

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

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

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

- **Demo Program:** All DEMO programs
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**:
  
  ```c
  WORD P1202_DriverClose( void );  for OME-PCI-1202(H/L) Card
  WORD P1602_DriverClose( void );  for OME-PCI-1602/1602F Card
  WORD P180X_DriverClose( void );  for OME-PCI-1800(H/L)/1802(H/L) Card
  ```

- **Input Parameter**: void

- **Return Value**: NoError

- **Demo Program**: All DEMO programs
2.9.3  P1202_GetConfigAddressSpace  
P1602_GetConfigAddressSpace  
P180X_GetConfigAddressSpace

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

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

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

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

- **Input Parameter:**  
  wBoardNo: OME-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 programs
2.9.4 P1202_WhichBoardActive
       P1602_WhichBoardActive
       P180X_WhichBoardActive

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

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

- **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 activate one of the OME-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 OME-PCI-1202(H/L) Card
  - WORD P1602_ActiveBoard(WORD wBoardNo);  
    
    for OME-PCI-1602/1602F Card
  - WORD P180X_ActiveBoard(WORD wBoardNo);  
    
    for OME-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:**
  ```c
  WORD P1602_Card0_StartScan(WORD wSampleRate, WORD wChannelStatus[], WORD wChanelConfig[], WORD wCount);
  ```

- **Input Parameter:**
  - `wSampleRate`: AD sampling rate = 8M/wSampleRate:
    - For OME-PCI-1202 Card:
      - `wSampleRate=73` → Sampling rate=8M/73=110K → Maximum
      - `wSampleRate=80` → Sampling rate=8M/80=100K
    - For OME-PCI-1602F/1602 Card:
      - `wSampleRate=40` → Sampling rate=8M/40=200K → Maximum for OME-PCI-1602F Card
      - `wSampleRate=80` → Sampling rate=8M/80=100K → Maximum for OME-PCI-1602 Card
    - For OME-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 OME-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**:
  ```c
  WORD P1202_Card0_ReadData(void); for OME-PCI-1202(H/L) Card
  WORD P1602_Card0_ReadData(void); for OME-PCI-1602/1602F Card
  WORD P180X_Card0_ReadData(void); for OME-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**:
  ```c
  void P1202_Card0_Stop(void); for OME-PCI-1202(H/L) Card
  void P1602_Card0_Stop(void); for OME-PCI-1602/1602F Card
  void P180X_Card0_Stop(void); for OME-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 OME-PCI-1602/1602F board. Use the 1602_ActiveBoard(….) to select the active board.

- **Syntax:**
  
  void P1202_DelayUs(WORD wDelayUs);  \*for OME-PCI-1202(H/L) Card
  void P1602_DelayUs(WORD wDelayUs);  \*for OME-PCI-1602/1602F Card
  void P180X_DelayUs(WORD wDelayUs);  \*for OME-PCI-1800(H/L)/1802(H/L) Card

- **Input Parameter:**
  
  wDelayUs : number of us to delay, 8191 Max  
  wDelayUs=1  \rightarrow delay 1 us  
  wDelayUs=1000  \rightarrow delay 1000 us = 1 ms  
  wDelayUs=8191  \rightarrow delay 8191 us = 8.191 ms (maximum delay)  
  wDelayUs=8192  \rightarrow invalidate delay (will return error)

- **Return Value:**
  
  NoError : OK  
  ExceedBoardNumber: invalidate board number  
  FindBoardError :cannot find the OME-PCI-1202(H/L)/1602/1602F/1800(H/L)/1802(H/L) board  
  InvalidateDelay : wDelayUs > 8191

- **Demo Program : DEMO1.C**
● Long Time Delay:

```c
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 );
}
```
3 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 CD ROM for details.
OMEGA ENGINEERING, INC. warrants this unit to be free of defects in materials and workmanship for a period of 13 months from date of purchase. OMEGA's WARRANTY adds an additional one (1) month grace period to the normal one (1) year product warranty to cover handling and shipping time. This ensures that OMEGA's customers receive maximum coverage on each product.

If the unit malfunctions, it must be returned to the factory for evaluation. OMEGA's Customer Service Department will issue an Authorized Return (AR) number immediately upon phone or written request. Upon examination by OMEGA, if the unit is found to be defective, it will be repaired or replaced at no charge. OMEGA's WARRANTY does not apply to defects resulting from any action of the purchaser, including but not limited to mishandling, improper interfacing, operation outside of design limits, improper repair, or unauthorized modification. This WARRANTY is VOID if the unit shows evidence of having been tampered with or shows evidence of having been damaged as a result of excessive corrosion; or current, heat, moisture or vibration; improper specification; misapplication; misuse or other operating conditions outside of OMEGA's control. Components which wear are not warranted, including but not limited to contact points, fuses, and triacs.

OMEGA is pleased to offer suggestions on the use of its various products. However, OMEGA neither assumes responsibility for any omissions or errors nor assumes liability for any damages that result from the use of its products in accordance with information provided by OMEGA, either verbal or written. OMEGA warrants only that the parts manufactured by it will be as specified and free of defects. OMEGA MAKES NO OTHER WARRANTIES OR REPRESENTATIONS OF ANY KIND WHATSOEVER, EXPRESS OR IMPLIED, EXCEPT THAT OF TITLE, AND ALL IMPLIED WARRANTIES INCLUDING ANY WARRANTY OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE HEREBY DISCLAIMED. LIMITATION OF LIABILITY: The remedies of purchaser set forth herein are exclusive, and the total liability of OMEGA with respect to this order, whether based on contract, warranty, negligence, indemnification, strict liability or otherwise, shall not exceed the purchase price of the component upon which liability is based. In no event shall OMEGA be liable for consequential, incidental or special damages.

CONDITIONS: Equipment sold by OMEGA is not intended to be used, nor shall it be used: (1) as a “Basic Component” under 10 CFR 21 (NRC), used in or with any nuclear installation or activity; or (2) in medical applications or used on humans. Should any Product(s) be used in or with any nuclear installation or activity, medical application, used on humans, or misused in any way, OMEGA assumes no responsibility as set forth in our basic WARRANTY/DISCLAIMER language, and, additionally, purchaser will indemnify OMEGA and hold OMEGA harmless from any liability or damage whatsoever arising out of the use of the Product(s) in such a manner.

RETURN REQUESTS/INQUIRIES

Direct all warranty and repair requests/inquiries to the OMEGA Customer Service Department. BEFORE RETURNING ANY PRODUCT(S) TO OMEGA, PURCHASER MUST OBTAIN AN AUTHORIZED RETURN (AR) NUMBER FROM OMEGA’S CUSTOMER SERVICE DEPARTMENT (IN ORDER TO AVOID PROCESSING DELAYS). The assigned AR number should then be marked on the outside of the return package and on any correspondence.

The purchaser is responsible for shipping charges, freight, insurance and proper packaging to prevent breakage in transit.

FOR WARRANTY RETURNS, please have the following information available BEFORE contacting OMEGA:
1. Purchase Order number under which the product was PURCHASED,
2. Model and serial number of the product under warranty, and
3. Repair instructions and/or specific problems relative to the product.

FOR NON-WARRANTY REPAIRS, consult OMEGA for current repair charges. Have the following information available BEFORE contacting OMEGA:
1. Purchase Order number to cover the COST of the repair,
2. Model and serial number of the product, and
3. Repair instructions and/or specific problems relative to the product.

OMEGA’s policy is to make running changes, not model changes, whenever an improvement is possible. This affords our customers the latest in technology and engineering.

OMEGA is a registered trademark of OMEGA ENGINEERING, INC.

© Copyright 2002 OMEGA ENGINEERING, INC. All rights reserved. This document may not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of OMEGA ENGINEERING, INC.
Where Do I Find Everything I Need for Process Measurement and Control? OMEGA...Of Course!

Shop online at www.omega.com

TEMPERATURE
- Thermocouple, RTD & Thermistor Probes, Connectors, Panels & Assemblies
- Wire: Thermocouple, RTD & Thermistor
- Calibrators & Ice Point References
- Recorders, Controllers & Process Monitors
- Infrared Pyrometers

PRESSURE, STRAIN AND FORCE
- Transducers & Strain Gages
- Load Cells & Pressure Gages
- Displacement Transducers
- Instrumentation & Accessories

FLOW/LEVEL
- Rotameters, Gas Mass Flowmeters & Flow Computers
- Air Velocity Indicators
- Turbine/Paddlewheel Systems
- Totalizers & Batch Controllers

pH/CONDUCTIVITY
- pH Electrodes, Testers & Accessories
- Benchtop/Laboratory Meters
- Controllers, Calibrators, Simulators & Pumps
- Industrial pH & Conductivity Equipment

DATA ACQUISITION
- Data Acquisition & Engineering Software
- Communications-Based Acquisition Systems
- Plug-in Cards for Apple, IBM & Compatibles
- Datalogging Systems
- Recorders, Printers & Plotters

HEATERS
- Heating Cable
- Cartridge & Strip Heaters
- Immersion & Band Heaters
- Flexible Heaters
- Laboratory Heaters

ENVIRONMENTAL MONITORING AND CONTROL
- Metering & Control Instrumentation
- Refractometers
- Pumps & Tubing
- Air, Soil & Water Monitors
- Industrial Water & Wastewater Treatment
- pH, Conductivity & Dissolved Oxygen Instruments