User’s Guide

Shop online at www.omega.com
e-mail: info@omega.com

OME-A822PG ISA-BUS
Multi-Functional Board
Windows 95/98/NT Software Manual
OMEGA Engineering, Inc. is committed to complying 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 ........................................................................................................................................... 4
   1.1 REFERENCES ................................................................................................................................................. 5
   1.2 RANGE CONFIGURATION ............................................................................................................................. 6

2. DECLARATIONS & DEMO ................................................................................................................................. 7
   2.1 USING VC++ & BC++BUILDER ................................................................................................................... 9
   2.1.1 THE VC++ DEMO RESULT ....................................................................................................................... 9
   2.1.2 BC++ BUILDER DEMO RESULT ............................................................................................................. 10
   2.1.3 A822.H (FOR WIN 95/98) .................................................................................................................... 11
   2.1.4 A822.H (FOR WIN NT) ........................................................................................................................ 17
   2.2 USING VISUAL BASIC ............................................................................................................................... 22
   2.2.1 THE VB DEMO RESULT ........................................................................................................................ 22
   2.2.2 A822.BAS (FOR WIN 95/98) .................................................................................................................. 23
   2.2.3 A822.BAS (FOR WIN NT) ..................................................................................................................... 29
   2.3 USING DELPHI ............................................................................................................................................. 35
   2.3.1 DELPHI DEMO RESULT ......................................................................................................................... 35
   2.3.2 A822.PAS (FOR WIN 95/98) ................................................................................................................... 36
   2.3.3 A822.PAS (FOR WIN NT) ...................................................................................................................... 45

3. FUNCTION DESCRIPTION ............................................................................................................................... 53
   3.1 ERROR CODES ............................................................................................................................................... 54
   3.2 FUNCTIONS NOT SUPPORTED .................................................................................................................... 57
   3.3 DRIVER FUNCTIONS .................................................................................................................................... 58
   3.3.1 A822_DriverInit ........................................................................................................................................ 58
   3.3.2 A822_DriverClose ................................................................................................................................. 58
   3.3.3 A822_DELAY ............................................................................................................................................ 59
   3.3.4 A822_Check_Address ............................................................................................................................ 59
   3.3.5 A822_SetTriggerMode .......................................................................................................................... 60
   3.4 TEST FUNCTION ........................................................................................................................................... 61
   3.4.1 A822_SHORT_SUB_2 ............................................................................................................................ 61
   3.4.2 A822_FLOAT_SUB_2 ............................................................................................................................ 61
   3.4.3 A822_Get_DLL_Version ......................................................................................................................... 62
   3.4.4 A822_GetDriverVersion ....................................................................................................................... 62
   3.5 COUNTER FUNCTION ............................................................................................................................... 63
   3.5.1 A822_SetCounter .................................................................................................................................. 63
   3.5.2 A822_ReadCounter ............................................................................................................................... 63
   3.6 DI/DO FUNCTION ....................................................................................................................................... 64
   3.6.1 A822_DI ..................................................................................................................................................... 64
   3.6.2 A822_DO ................................................................................................................................................ 64
   3.6.3 A822_OutputByte ................................................................................................................................. 65
   3.6.4 A822_OutputWord ............................................................................................................................... 65
   3.6.5 A822_InputByte .................................................................................................................................... 66
   3.6.6 A822_InputWord ................................................................................................................................... 66
   3.7 AD FUNCTIONS ......................................................................................................................................... 67
   3.7.1 A822_SetChGain ................................................................................................................................... 67
   3.7.2 A822_Hex2Float .................................................................................................................................... 67
   3.7.3 A822_Fast AD_Hex ............................................................................................................................... 68
   3.7.4 A822_Fast AD_Float ............................................................................................................................ 68
   3.7.5 A822_AD_Hex ...................................................................................................................................... 69
   3.7.6 A822_AD_Float .................................................................................................................................... 69
   3.7.7 A822_ADs_Hex .................................................................................................................................... 70
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7.8</td>
<td>A822_ADs_Float</td>
</tr>
<tr>
<td>3.8</td>
<td>DA FUNCTIONS</td>
</tr>
<tr>
<td>3.8.1</td>
<td>A822_DA</td>
</tr>
<tr>
<td>3.8.2</td>
<td>A822_Uni5_DA</td>
</tr>
<tr>
<td>3.8.3</td>
<td>A822_Uni10_DA</td>
</tr>
<tr>
<td>3.9</td>
<td>AD WITH INTERRUPT</td>
</tr>
<tr>
<td>3.9.1</td>
<td>A822_IntInstall</td>
</tr>
<tr>
<td>3.9.2</td>
<td>A822_IntGetCount</td>
</tr>
<tr>
<td>3.9.3</td>
<td>A822_IntStart</td>
</tr>
<tr>
<td>3.9.4</td>
<td>A822_IntGetHexBuf</td>
</tr>
<tr>
<td>3.9.5</td>
<td>A822_IntGetFloatBuf</td>
</tr>
<tr>
<td>3.9.6</td>
<td>A822_IntStop</td>
</tr>
<tr>
<td>3.9.7</td>
<td>A822_IntRemove</td>
</tr>
<tr>
<td>3.9.8</td>
<td>Architecture of Interrupt mode</td>
</tr>
<tr>
<td>3.10</td>
<td>AD DMA FUNCTION</td>
</tr>
<tr>
<td>3.10.1</td>
<td>A822_AD_DMA_InstallIrq</td>
</tr>
<tr>
<td>3.10.2</td>
<td>A822_AD_DMA_IsNotFinished</td>
</tr>
<tr>
<td>3.10.3</td>
<td>A822_AD_DMA_Start</td>
</tr>
<tr>
<td>3.10.4</td>
<td>A822_AD_DMA_GetBuffer</td>
</tr>
<tr>
<td>3.10.5</td>
<td>A822_AD_DMA_GetFloatBuffer</td>
</tr>
<tr>
<td>3.10.6</td>
<td>A822_AD_DMA_Stop</td>
</tr>
<tr>
<td>3.10.7</td>
<td>A822_AD_DMA_RemoveIrq</td>
</tr>
<tr>
<td>3.10.8</td>
<td>Architecture of DMA mode</td>
</tr>
<tr>
<td>3.11</td>
<td>AD WITH CHANNEL SCAN</td>
</tr>
<tr>
<td>3.11.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.11.2</td>
<td>A822_ChScan_Clear</td>
</tr>
<tr>
<td>3.11.3</td>
<td>A822_ChScan_Add</td>
</tr>
<tr>
<td>3.11.4</td>
<td>A822_ChScan_Set</td>
</tr>
<tr>
<td>3.11.5</td>
<td>A822_ChScan_PollingHex</td>
</tr>
<tr>
<td>3.11.6</td>
<td>A822_ChScan_PollingFloat</td>
</tr>
<tr>
<td>3.12</td>
<td>AD INTERRUPT, CHANNEL SCAN FUNCTION</td>
</tr>
<tr>
<td>3.12.1</td>
<td>Introduction</td>
</tr>
<tr>
<td>3.12.2</td>
<td>A822_ChScan_IntInstall</td>
</tr>
<tr>
<td>3.12.3</td>
<td>A822_ChScan_IntStart</td>
</tr>
<tr>
<td>3.12.4</td>
<td>A822_ChScan_IntStop</td>
</tr>
<tr>
<td>3.12.5</td>
<td>A822_ChScan_IntRemove</td>
</tr>
<tr>
<td>3.12.6</td>
<td>A822_ChScan_IntGetCount</td>
</tr>
<tr>
<td>3.12.7</td>
<td>A822_ChScan_IntGetHexBuf</td>
</tr>
<tr>
<td>3.12.8</td>
<td>A822_ChScan_IntGetFloatBuf</td>
</tr>
</tbody>
</table>

4. PROGRAM ARCHITECTURE ................................................................................. 94

5. REPORT PROBLEMS ......................................................................................... 95
1. INTRODUCTION

The OME-A-822PGH/L is a multifunction, 12-bit resolution, A/D, D/A and digital I/O card. The features of the OME-A-822PGH/L are given as below:

- 12-bit A/D, 16 single-ended channels or 8 differential channels
- OME-A-822PGL: low gain (1/2/4/8), the analog input signal range configuration code is given in Sec. 2.1
- OME-A-822PGH: high gain (1/10/100/1000), the analog input signal range configuration code is given in Sec. 2.1
- 12-bit D/A, 2 channels, 0-5V or 0-10V output by hardware JP1 setting
- 16 TTL-compatible digital input channels
- 16 TTL-compatible digital output channels

The A822.DLL and A822.Vxd (or A822.sys) are a collection of data acquisition subroutines for OME-A822PG for Windows 95/98 (or NT) Applications. These subroutines are written in C language and perform a variety of data acquisition operations.

The subroutines in A822.DLL are easy to understand because of the user friendly names. It provides powerful, easy-to-use subroutine for developing your data acquisition applications. Your program can call these DLL functions by VC++, VB, Delphi and Borland C++ Builder easily. To speed-up your development process, some demonstration source programs are provided.

The OME-A822 software consists of these DLLs and device drivers:

For Windows 95/98
- A822.dll → Libraries for A822 PGL/PGH card
- A822.Vxd → Device driver for Windows 95/98

For Windows NT
- A822.dll → Libraries for A822 PGL/PGH card
- A822.sys, Napwnt.sys → Device driver for Windows NT
1.1 REFERENCES

Please refer to the following user manuals:

- **Readme.txt:** Describes files that install into your system, and where you can find it

- **Whatnew.txt:** Describes the differences in the software versions

- **SoftInst.pdf:** How to install the software package under Windows 95/98/NT/2000

- **CallDll.pdf:** How to call the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3

- **ResCheck.pdf:** How to check the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/NT/2000
1.2 RANGE CONFIGURATION

The A/D converter of OME-A822PGH/L is 12 bits under all configuration codes. If the analog input range is configured to +/- 5V range, the resolution of one bit is equal to 2.44 mV. If the analog input range is configured to +/- 2.5V range, the resolution will be 1.22 mV. If the analog input signal is about 1 V, using configuration 0/1/2 (for OME-A822PGL) will get nearly the same result except resolution. So choose the correct configuration code can achieve the highest precision measurement.

OME-A-822PGL Input Signal Range Configuration Code Table

<table>
<thead>
<tr>
<th>Bipolar/Unipolar</th>
<th>Input Signal Range</th>
<th>Configuration Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar</td>
<td>+/- 5V</td>
<td>0</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 2.5V</td>
<td>1</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1.25V</td>
<td>2</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.0625V</td>
<td>3</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 10V</td>
<td>4</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 5V</td>
<td>5</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 2.5V</td>
<td>6</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0V ~ 1.25V</td>
<td>7</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 10V</td>
<td>8</td>
</tr>
</tbody>
</table>

OME-A-822PGH Input Signal Range Configuration Code Table

<table>
<thead>
<tr>
<th>Bipolar/Unipolar</th>
<th>Input Signal Range</th>
<th>Configuration Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bipolar</td>
<td>+/- 5V</td>
<td>0</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.5V</td>
<td>1</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.05V</td>
<td>2</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.005V</td>
<td>3</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0 ~ 10V</td>
<td>4</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0 ~ 1V</td>
<td>5</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0 ~ 0.1V</td>
<td>6</td>
</tr>
<tr>
<td>Unipolar</td>
<td>0 ~ 0.01V</td>
<td>7</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 10V</td>
<td>8</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 1V</td>
<td>9</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.1V</td>
<td>10</td>
</tr>
<tr>
<td>Bipolar</td>
<td>+/- 0.01V</td>
<td>11</td>
</tr>
</tbody>
</table>
2. DECLARATIONS & DEMO

Please refer to user manual “CallDLL.pdf”.

For Windows 95/98:

|--\Driver
  |--\A822.DLL ← Dynamic Linking Library
  |--\A822.Vxd ← Device driver for OME-A822PG

|--\BCB ← For Borland C++ Builder
  |--\A822.H ← Header file
  +--\A822.Lib ← Import Library for BCB only

|--\Delphi ← For Delphi
  +--\A822.pas ← Declaration file

|--\VB ← For Visual Basic
  +--\A822.bas ← Declaration file

+--\VC ← For Visual C++
  |--\A822.H ← Header file
  +--\A822.Lib ← Import Library for VC only
For Windows NT:

|--\Driver
  |--\A822.DLL ← Dynamic Linking Library
  |--\A822.sys ← device driver
  |--\Napwnt.sys ← device driver

|--\BCB ← For Borland C++ Builder
  |--\A822.H ← Header file
  |--\A822.Lib ← Import Library for BCB only

|--\Delphi ← For Delphi
  |--\A822.pas ← Declaration file

|--\VB ← For Visual Basic
  |--\A822.bas ← Declaration file

|--\VC ← For Visual C++
  |--\A822.H ← Header file
  |--\A822.Lib ← Import Library for VC only
2.1 USING VC++ & BC++BUILDER

2.1.1 THE VC++ DEMO RESULT

```
NOW --> Base=220, DMA=1, IRQ=9
4. [A822.VXD] Open OK
5. Vxd Version=101

NOW Setting Is --> Base=220, DMA=1, IRQ=9
1. DLL Version=210
2. Find a A822 in IOPORT_220
3. SHORT_SUB_2(1,2) = -1
4. FLOAT_SUB_2(1.0,2.0) = -1.000000
5. DO=0x55aa --> DI=55aa
6. DA_0=0x800 --> AD_0=0.197754
7. DA_1=0xffff --> AD_1=-1.181641
8. A822_ADs_Hex TEST : 1.886175
9. A822_ADs_Float TEST : 2.300615
6. Install Irq Ok
Hex --> b04 af9 884 5c5 43d
Float --> 2.393 1.850 0.322 -1.394 -2.35185
```
2.1.2 BC++ BUILDER DEMO RESULT
2.1.3 A822.H (FOR WIN 95/98)

#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

/******************** DEFINE A822 RELATIVE ADDRESS ******************/
#define A822_TIMER0       0x00
#define A822_TIMER1       0x01
#define A822_TIMER2       0x02
#define A822_TIMER_MODE   0x03
#define A822_AD_LO        0x04 // Analog to Digital, Low Byte
#define A822_AD_HI        0x05 // Analog to Digital, High Byte
#define A822_DA_CH0_LO    0x04 // Digit to Analog, CH 0
#define A822_DA_CH0_HI    0x05
#define A822_DA_CH1_LO    0x06 // Digit to Analog, CH 1
#define A822_DA_CH1_HI    0x07
#define A822_DI_LO        0x06 // Digit Input
#define A822_DO_LO        0x0D // Digit Output
#define A822_CLEAR_IRQ    0x08
#define A822_SET_GAIN     0x09
#define A822_SET_CH       0x0A
#define A822_SET_MODE     0x0B
#define A822_SOFT_TRIG    0x0C
#define A822_POLLING_MODE 1
#define A822_DMA_MODE     2
#define A822_INTERRUPT_MODE 6
/** define the gain mode **/
#define A822_BI_1    0
#define A822_BI_10   1
#define A822_BI_100  2
#define A822_BI_1000 3
#define A822_UNI_1   4
#define A822_UNI_10  5
#define A822_UNI_100 6
#define A822_UNI_1000 7
#define A822_BI_05   8
#define A822_BI_5    9
#define A822_BI_50   10
#define A822_BI_500  11

#define A822_BI_2    1
#define A822_BI_4    2
#define A822_BI_8    3
#define A822_UNI_2   5
#define A822_UNI_4   6
#define A822_UNI_8   7

#define A822PGL       0
#define A822PGH       1

#define A822_NoError                 0
#define A822_DriverOpenError         1
#define A822_DriverNoOpen            2
#define A822_GetDriverVersionError   3
#define A822_InstallIrqError          4
#define A822_ClearIntCountError       5
#define A822_GetIntCountError         6
#define A822_GetBufferError           7
#define A822_AdError1                 100
#define A822_AdError2                 -200.0
#define A822_InstallBufError          10
#define A822_AllocateMemoryError      11
#define A822_CARDTypeError            12
#define A822_TimeoutError             13
#define A822_OtherError               14
#define A822_ConfigCodeError          15
#define A822_IntStopError             16
#define A822_IntRemoveError           17
#define A822_IntInstallEventError     18
#define A822_BufferFull                19
#define A822_NoChannelToScan          20
#define A822_IntInstallChannelError   21
#define A822_IntInstallConfigError    22
#define A822_GetDmaStatusError        23

// Function of Driver
EXPORTS WORD   CALLBACK A822_DriverInit(void);
EXPORTS void   CALLBACK A822_DriverClose(void);
EXPORTS WORD   CALLBACK A822_DELAY
               (WORD wBase, WORD wDownCount);
EXPORTS WORD   CALLBACK A822_Check_Address(WORD wBase);
EXPORTS void   CALLBACK A822_SetTriggerMode(WORD wTriggerMode);

// Function of Test
EXPORTS short  CALLBACK A822_SHORT_SUB_2(short nA, short nB);
EXPORTS float  CALLBACK A822_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD   CALLBACK A822_Get_DLL_Version(void);
EXPORTS WORD   CALLBACK A822_GetDriverVersion
               (WORD *wDriverVersion);

// Function of Counter
EXPORTS void   CALLBACK A822_SetCounter
               (WORD wBase, WORD wCounterNo,
                WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD CALLBACK A822_ReadCounter
               (WORD wBase, WORD wCounterNo, WORD bCounterMode);

// Function of DI/DO
EXPORTS WORD   CALLBACK A822_DI(WORD wBase);
EXPORTS void   CALLBACK A822_DO(WORD wBase, WORD wHexValue);
EXPORTS void   CALLBACK A822_OutputByte
               (WORD wPortAddr, UCHAR bOutputVal);
EXPORTS void   CALLBACK A822_OutputWord
               (WORD wPortAddr, WORD wOutputVal);
EXPORTS WORD   CALLBACK A822_InputByte(WORD wPortAddr);
EXPORTS WORD   CALLBACK A822_InputWord(WORD wPortAddr);
// Function of AD
EXPORTS WORD  CALLBACK A822_SetChGain
    (WORD wBase, WORD wChannel, WORD wConfig, WORD wCardType);
EXPORTS WORD  CALLBACK A822_Fast_AD_Hex(WORD *wVal);
EXPORTS WORD  CALLBACK A822_Fast_AD_Float(float *fVal);
EXPORTS WORD  CALLBACK A822_AD_Hex
    (WORD wBase, WORD wChannel, 
     WORD wConfig, WORD wCardType, WORD *wVal);
EXPORTS WORD  CALLBACK A822_AD_Float
    (WORD wBase, WORD wChannel, 
     WORD wConfig, WORD wCardType, float *fVal);
EXPORTS WORD  CALLBACK A822_ADs_Hex
    (WORD wBase, WORD wChannel, WORD wConfig, 
     WORD wType, WORD wBuf[], WORD wCount);
EXPORTS WORD  CALLBACK A822_ADs_Float
    (WORD wBase, WORD wChannel, WORD wConfig, 
     WORD wType, float fBuf[], WORD wCount);
EXPORTS WORD  CALLBACK A822_Hex2Float(WORD wConfig, 
    WORD wCardType, WORD wHex, float *fVal);

// Please uses the A822_AD_Float() function
EXPORTS float CALLBACK A822_AD(WORD wBase, WORD wChannel, 
    WORD wConfig, WORD wType);

// Function of DA
EXPORTS void  CALLBACK A822_DA
    (WORD wBase, WORD wChannel, WORD wHexValue);
EXPORTS void  CALLBACK A822_Uni5_DA
    (WORD wBase, WORD wChannel, float fValue);
EXPORTS void  CALLBACK A822_Uni10_DA
    (WORD wBase, WORD wChannel, float fValue);

// Function of Interrupt
// Please uses the A822_Intxxxx series function set
EXPORTS WORD  CALLBACK A822_InstallIrq(WORD wBase, 
    WORD wIrq, HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD  CALLBACK A822_AD_INT_Start(WORD wCardType, 
    WORD Ch, WORD Gain, WORD c1, WORD c2);
EXPORTS WORD  CALLBACK A822_AD_INT_Stop(void);
EXPORTS WORD  CALLBACK A822_GetIntCount(DWORD *dwVal);
EXPORTS WORD  CALLBACK A822_GetBuffer
    (DWORD dwNum, WORD wBuffer[]);
EXPORTS WORD  CALLBACK A822_GetFloatBuffer
    (DWORD dwNum, float fBuffer[]);
// Function of Interrupt
EXPORTS WORD  CALLBACK A822_IntInstall(WORD wBase,
    WORD wIrq, HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD  CALLBACK A822_IntStart(WORD wCardType,
    WORD wChannel, WORD wGain, WORD c1, WORD c2);
EXPORTS WORD  CALLBACK A822_IntGetCount(DWORD *dwVal);
EXPORTS WORD  CALLBACK A822_IntGetHexBuf
    (DWORD dwNum, WORD wBuf[]);
EXPORTS WORD  CALLBACK A822_IntGetFloatBuf
    (DWORD dwNum, float fBuf[]);
EXPORTS WORD  CALLBACK A822_IntStop(void);
EXPORTS WORD  CALLBACK A822_IntRemove(void);

// Function of DMA
EXPORTS WORD  CALLBACK A822_AD_DMA_InstallIrq
    (WORD wBase, WORD wIrq, WORD wDmaChan);
EXPORTS WORD  CALLBACK A822_AD_DMA_RemoveIrq(void);
EXPORTS WORD  CALLBACK A822_AD_DMA_Start
    (WORD wCardType, WORD Ch, WORD Gain,
     WORD c1,WORD c2, DWORD cnt, WORD wPassOut[]);
EXPORTS WORD  CALLBACK A822_AD_DMA_Stop(void);
EXPORTS WORD  CALLBACK A822_AD_DMA_IsNotFinished(void);
EXPORTS WORD  CALLBACK A822_AD_DMA_GetBuffer(WORD *wBuf);
EXPORTS WORD  CALLBACK A822_AD_DMA_GetFloatBuffer(float *fBuf);

// Function of Channel-Scan with Polling
EXPORTS void CALLBACK A822_ChScan_Clear(void);
EXPORTS WORD CALLBACK A822_ChScan_Add
    (WORD wChannel, WORD wConfig);
EXPORTS WORD CALLBACK A822_ChScan_Set
    (WORD wChannel[], WORD wConfig[], WORD wChNum);
EXPORTS WORD CALLBACK A822_ChScan_PollingHex
    (WORD wBase, WORD wCardType,
     WORD wBuf[], WORD wNumPerCh);
EXPORTS WORD CALLBACK A822_ChScan_PollingFloat
    (WORD wBase, WORD wCardType, float fBuf[], WORD wNumPerCh);
// Function of Channel-Scan with Interrupt
EXPORTS WORD CALLBACK A822_ChScan_IntInstall
    (WORD wBase, WORD wIrq, HANDLE *hEvent, DWORD dwNumPerCh);
EXPORTS WORD CALLBACK A822_ChScan_IntStart
    (WORD c1, WORD c2, WORD wCardType);
EXPORTS WORD CALLBACK A822_ChScan_IntGetCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A822_ChScan_IntGetHexBuf(WORD wBuff[]);
EXPORTS WORD CALLBACK A822_ChScan_IntGetFloatBuf(float fBuff[]);
EXPORTS WORD CALLBACK A822_ChScan_IntStop(void);
EXPORTS WORD CALLBACK A822_ChScan_IntRemove(void);
2.1.4 A822.H (FOR WIN NT)

#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

/************************* DEFINE A822 RELATIVE ADDRESS **********************/
#define A822_TIMER0       0x00
#define A822_TIMER1       0x01
#define A822_TIMER2       0x02
#define A822_TIMER_MODE   0x03
#define A822_AD_LO        0x04 // Analog to Digital, Low Byte
#define A822_AD_HI        0x05 // Analog to Digital, High Byte
#define A822_DA_CH0_LO   0x04 // Digit to Analog, CH 0
#define A822_DA_CH0_HI   0x05
#define A822_DA_CH1_LO   0x06 // Digit to Analog, CH 1
#define A822_DA_CH1_HI   0x07
#define A822_DI_LO        0x06 // Digit Input
#define A822_DO_LO        0x0D // Digit Output
#define A822_CLEAR_IRQ   0x08
#define A822_SET_GAIN    0x09
#define A822_SET_CH       0x0A
#define A822_SET_MODE    0x0B
#define A822_SOFT_TRIG   0x0C
#define A822_POLLING_MODE     1
#define A822_DMA_MODE          2
#define A822_INTERRUPT_MODE   6
/*** define the gain mode ***/
define A822_BI_1 0
define A822_BI_10 1
define A822_BI_100 2
define A822_BI_1000 3
define A822_UNI_1 4
define A822_UNI_10 5
define A822_UNI_100 6
define A822_UNI_1000 7
define A822_BI_05 8
define A822_BI_5 9
define A822_BI_50 10
define A822_BI_500 11
define A822_BI_2 1
define A822_BI_4 2
define A822_BI_8 3
define A822_UNI_2 5
define A822_UNI_4 6
define A822_UNI_8 7
define A822PGL 0
define A822PGH 1

define A822_NoError 0
define A822_DriverOpenError 1
define A822_DriverNoOpen 2
define A822_GetDriverVersionError 3
define A822_InstallIrqError 4
define A822_ClearIntCountError 5
define A822_GetIntCountError 6
define A822_GetBufferError 7
define A822_AdError1 100
define A822_AdError2 -200.0
define A822_InstallBufError 10
define A822_AllocateMemoryError 11
define A822_CardTypeError 12
define A822_TimeoutError 13
define A822.OtherError 14
define A822_ConfigCodeError 15
#define A822_IntStopError 16
#define A822_IntRemoveError 17
#define A822_IntInstallEventError 18
#define A822_BufferFull 19
#define A822_NoChannelToScan 20
#define A822_IntInstallChannelError 21
#define A822_IntInstallConfigError 22

// Function of Driver
EXPORTS WORD CALLBACK A822_DriverInit(void);
EXPORTS void CALLBACK A822_DriverClose(void);
EXPORTS WORD CALLBACK A822_DELAY(WORD wBase, WORD wDownCount);
EXPORTS WORD CALLBACK A822_Check_Address(WORD wBase);
EXPORTS void CALLBACK A822_SetTriggerMode(WORD wTriggerMode);

// Function of Test
EXPORTS short CALLBACK A822_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK A822_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK A822_Get_DLL_Version(void);
EXPORTS WORD CALLBACK A822_GetDriverVersion(WORD *wDriverVersion);

// Function of Counter
EXPORTS void CALLBACK A822_SetCounter(WORD wBase, WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD CALLBACK A822_ReadCounter(WORD wBase, WORD wCounterNo, WORD bCounterMode);

// Function of DI/DO
EXPORTS WORD CALLBACK A822_DI(WORD wBase);
EXPORTS void CALLBACK A822_DO(WORD wBase, WORD wHexValue);
EXPORTS void CALLBACK A822_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
EXPORTS void CALLBACK A822_OutputWord(WORD wPortAddr, WORD wOutputVal);
EXPORTS WORD CALLBACK A822_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK A822_InputWord(WORD wPortAddr);
// Function of AD
EXPORTS WORD  CALLBACK A822_SetChGain(WORD wBase,
    WORD wChannel, WORD wConfig, WORD wCardType);
EXPORTS WORD  CALLBACK A822_Fast_AD_Hex(WORD *wVal);
EXPORTS WORD  CALLBACK A822_Fast_AD_Float(float *fVal);

EXPORTS WORD  CALLBACK A822_AD_Hex
    (WORD wBase, WORD wChannel,
     WORD wConfig, WORD wCardType, WORD *wVal);
EXPORTS WORD  CALLBACK A822_AD_Float
    (WORD wBase, WORD wChannel,
     WORD wConfig, WORD wCardType, float *fVal);
EXPORTS WORD  CALLBACK A822_ADs_Hex
    (WORD wBase, WORD wChannel, WORD wConfig,
     WORD wType, WORD wBuf[], WORD wCount);
EXPORTS WORD  CALLBACK A822_ADs_Float
    (WORD wBase, WORD wChannel, WORD wConfig,
     WORD wType, float fBuf[], WORD wCount);
EXPORTS WORD  CALLBACK A822_Hex2Float(WORD wConfig,
    WORD wCardType, WORD wHex, float *fVal);

// Please uses the A822_AD_Float() function
EXPORTS float CALLBACK A822_AD(WORD wBase, WORD wChannel,
    WORD wConfig, WORD wType);

// Function of DA
EXPORTS void  CALLBACK A822_DA
    (WORD wBase, WORD wChannel, WORD wHexValue);
EXPORTS void  CALLBACK A822_Uni5_DA
    (WORD wBase, WORD wChannel, float fValue);
EXPORTS void  CALLBACK A822_Uni10_DA
    (WORD wBase, WORD wChannel, float fValue);

// Function of Interrupt
// Please uses the A822_Intxxxx series function set
EXPORTS WORD   CALLBACK A822_InstallIrq(WORD wBase,
    WORD wlrq, HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD   CALLBACK A822_AD_INT_Start(WORD wCardType,
    WORD Ch, WORD Gain, WORD c1, WORD c2);
EXPORTS WORD   CALLBACK A822_AD_INT_Stop(void);
EXPORTS WORD   CALLBACK A822_GetIntCount(DWORD *dwVal);
EXPORTS WORD   CALLBACK A822_GetBuffer
    (DWORD dwNum, WORD wBuffer[]);
EXPORTS WORD   CALLBACK A822_GetFloatBuffer
    (DWORD dwNum, float fBuffer[]);
// Function of Interrupt
EXPORTS WORD   CALLBACK A822_IntInstall(WORD wBase,  
           WORD wIrq, HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD   CALLBACK A822_IntStart(WORD wCardType,  
           WORD wChannel, WORD wGain, WORD c1, WORD c2);
EXPORTS WORD   CALLBACK A822_IntGetCount(DWORD *dwVal);
EXPORTS WORD   CALLBACK A822_IntGetHexBuf  
           (DWORD dwNum, WORD wBuf[]);
EXPORTS WORD   CALLBACK A822_IntGetFloatBuf  
           (DWORD dwNum, float fBuff[]);
EXPORTS WORD   CALLBACK A822_IntStop(void);
EXPORTS WORD   CALLBACK A822_IntRemove(void);

// Function of Channel-Scan with Polling
EXPORTS void   CALLBACK A822_ChScan_Clear(void);
EXPORTS WORD   CALLBACK A822_ChScan_Add  
           (WORD wChannel, WORD wConfig);
EXPORTS WORD   CALLBACK A822_ChScan_Set  
           (WORD wChannel[], WORD wConfig[], WORD wChNum);
EXPORTS WORD   CALLBACK A822_ChScan_PollingHex (WORD wBase,  
           WORD wCardType, WORD wBuf[], WORD wNumPerCh);
EXPORTS WORD   CALLBACK A822_ChScan_PollingFloat  
           (WORD wBase, WORD wCardType, float fBuff[], WORD wNumPerCh);

// Function of Channel-Scan with Interrupt
EXPORTS WORD   CALLBACK A822_ChScan_IntInstall(WORD wBase,  
           WORD wIrq, HANDLE *hEvent, DWORD dwNumPerCh);
EXPORTS WORD   CALLBACK A822_ChScan_IntStart  
           (WORD c1, WORD c2, WORD wCardType);
EXPORTS WORD   CALLBACK A822_ChScan_IntGetCount(DWORD *dwVal);
EXPORTS WORD   CALLBACK A822_ChScan_IntGetHexBuf(WORD wBuf[]);
EXPORTS WORD   CALLBACK A822_ChScan_IntGetFloatBuf(float fBuff[]);
EXPORTS WORD   CALLBACK A822_ChScan_IntStop(void);
EXPORTS WORD   CALLBACK A822_ChScan_IntRemove(void);
2.2 USING VISUAL BASIC

2.2.1 THE VB DEMO RESULT
2.2.2 A822.BAS (FOR WIN 95/98)

Attribute VB_Name = "A822"

**************************************************************************
DEFINE A822 RELATIVE ADDRESS *****************************/
Global Const A822_TIMER0 = &H0
Global Const A822_Timer1 = &H1
Global Const A822_TIMER2 = &H2
Global Const A822_TIMER_MODE = &H3
Global Const A822_AD_LO = &H4  ' Analog to Digital, Low Byte
Global Const A822_AD_HI = &H5  ' Analog to Digital, High Byte
Global Const A822_DA_CH0_LO = &H4  ' Digit to Analog, CH 0
Global Const A822_DA_CH0_HI = &H5
Global Const A822_DA_CH1_LO = &H6  ' Digit to Analog, CH 1
Global Const A822_DA_CH1_HI = &H7
Global Const A822_DI_LO = &H6  ' Digit Input
Global Const A822_DO_LO = &HD  ' Digit Output
Global Const A822_CLEAR_IRQ = &H8
Global Const A822_SET_GAIN = &H9
Global Const A822_SET_CH = &HA
Global Const A822_SET_MODE = &HB
Global Const A822_SOFT_TRIG = &HC
Global Const A822_POLLING_MODE = 1
Global Const A822_DMA_MODE = 2
Global Const A822_INTERRUPT_MODE = 6

*** define the gain mode ***/
Global Const A822_BI_1 = 0
Global Const A822_BI_10 = 1
Global Const A822_BI_100 = 2
Global Const A822_BI_1000 = 3
Global Const A822_UNI_1 = 4
Global Const A822_UNI_10 = 5
Global Const A822_UNI_100 = 6
Global Const A822_UNI_1000 = 7
Global Const A822_BI_05 = 8
Global Const A822_BI_5 = 9
Global Const A822_BI_50 = 10
Global Const A822_BI_500 = 11
Global Const A822_BI_2   = 1
Global Const A822_BI_4   = 2
Global Const A822_BI_8   = 3
Global Const A822_UNI_2  = 5
Global Const A822_UNI_4  = 6
Global Const A822_UNI_8  = 7

Global Const A822PGL    = 0
Global Const A822PGH    = 1

Global Const A822_NoError    = 0
Global Const A822_DriverOpenError   = 1
Global Const A822_DriverNoOpen    = 2
Global Const A822_GetDriverVersionError = 3
Global Const A822_InstallIrrqError  = 4
Global Const A822_ClearIntCountError = 5
Global Const A822_GetIntCountError = 6
Global Const A822_GetBufferError = 7
Global Const A822_AdError1    = 100
Global Const A822_AdError2    = -200#
Global Const A822_InstallBufError = 10
Global Const A822_AllocateMemoryError = 11
Global Const A822_CardTypeError = 12
Global Const A822_TimeoutError = 13
Global Const A822_OtherError = 14
Global Const A822_ConfigCodeError = 15
Global Const A822_IntStopError    = 16
Global Const A822_IntRemoveError    = 17
Global Const A822_IntInstallEventError = 18
Global Const A822_BufferFull    = 19
Global Const A822_NoChannelToScan = 20
Global Const A822_IntInstallChannelError = 21
Global Const A822_IntInstallConfigError = 22
Global Const A822_GetDmaStatusError = 23

****** Driver Functions ******
Declare Function A822_DriverInit Lib "A822.DLL" () As Integer
Declare Sub A822_DriverClose Lib "A822.DLL" ()
Declare Function A822_DELAY Lib "A822.DLL" (ByVal wBase As Integer, ByVal wDownCount As Integer) As Integer
Declare Function A822_Check_Address Lib "A822.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A822_SetTriggerMode Lib "A822.DLL" (ByVal wTriggerMode As Integer)
'*'******** Test Functions '*********
Declare Function A822_SHORT_SUB_2 Lib "A822.DLL" _
   (ByVal nA As Integer, ByVal nB As Integer) As Integer
Declare Function A822_FLOAT_SUB_2 Lib "A822.DLL" _
   (ByVal fA As Single, ByVal fB As Single) As Single
Declare Function A822_Get_DLL_Version Lib "A822.DLL" () As Integer
Declare Function A822_GetDriverVersion Lib "A822.DLL" _
   (wDriverVersion As Integer) As Integer

'*'******** Counter Functions '*********
Declare Sub A822_SetCounter Lib "A822.DLL" _
   (ByVal wBase As Integer, ByVal wCounterNo As Integer, _
   ByVal bCounterMode As Integer, ByVal wCounterValue As Long)
Declare Function A822_ReadCounter Lib "A822.DLL" _
   (ByVal wBase As Integer, ByVal wCounterNo As Integer, _
   ByVal bCounterMode As Integer) As Long

'*'****** DI/DO Functions '******
Declare Function A822_DI Lib "A822.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A822_DO Lib "A822.DLL" _
   (ByVal wBase As Integer, ByVal wHexValue As Integer)
Declare Sub A822_OutputByte Lib "A822.DLL" _
   (ByVal wPortAddr As Integer, ByVal bOutputVal As Byte)
Declare Sub A822_OutputWord Lib "A822.DLL" _
   (ByVal wPortAddr As Integer, ByVal wOutputVal As Integer)
Declare Function A822_InputByte Lib "A822.DLL" _
   (ByVal wPortAddr As Integer) As Integer
Declare Function A822_InputWord Lib "A822.DLL" _
   (ByVal wPortAddr As Integer) As Integer

'*'****** AD Functions '******
Declare Function A822_SetChGain Lib "A822.DLL" _
   (ByVal wBase As Integer, ByVal wChannel As Integer, _
   ByVal wConfig As Integer, ByVal wCardType As Integer) As Integer
Declare Function A822_Fast_AD_Hex Lib "A822.DLL" _
   (wVal As Integer) As Integer
Declare Function A822_Fast_AD_Float Lib "A822.DLL" _
   (fVal As Single) As Integer
Declare Function A822_AD_Hex Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wCardType As Integer, wVal As Integer) As Integer

Declare Function A822_AD_Float Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wCardType As Integer, fVal As Single) As Integer

Declare Function A822_Hex2Float Lib "A822.DLL" (ByVal wConfig As Integer, ByVal wCardType As Integer, ByVal wHex As Integer, fVal As Single) As Integer

Declare Function A822_ADs_Hex Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer, wBuf As Integer, ByVal wCount As Integer) As Integer

Declare Function A822_ADs_Float Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer, fBuf As Single, ByVal wCount As Integer) As Integer

' Please uses the A822_AD_Float() function
Declare Function A822_AD Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer) As Single

'******* DA Functions **********
Declare Sub A822_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wHexValue As Integer)
Declare Sub A822_Uni5_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal fValue As Single)
Declare Sub A822_Uni10_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal fValue As Single)
****** Interrupt Functions **********
' Please uses the A822_Intxxxx series function set
Declare Function A822_InstallIrq Lib "A822.DLL" _
    ByVal wBase As Integer, ByVal wIrq As Integer, _
    ByVal hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A822_AD_INT_Start Lib "A822.DLL" _
    ByVal wCardType As Integer, ByVal Ch As Integer, _
    ByVal Gain As Integer, ByVal c1 As Integer, _
    ByVal c2 As Integer) As Integer
Declare Function A822_AD_INT_Stop Lib "A822.DLL" () As Integer
Declare Function A822_GetIntCount Lib "A822.DLL" (dwVal As Long) As Integer
Declare Function A822_GetBuffer Lib "A822.DLL" _
    ByVal dwNum As Long, ByVal wBuffer As Integer) As Integer
Declare Function A822_GetFloatBuffer Lib "A822.DLL" _
    ByVal dwNum As Integer, ByVal fbuffer As Single) As Integer

****** Interrupt Functions **********
Declare Function A822_IntInstall Lib "A822.DLL" _
    ByVal wBase As Integer, ByVal wIrq As Integer, _
    ByVal hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A822_IntStart Lib "A822.DLL" _
    ByVal wCardType As Integer, ByVal Ch As Integer, _
    ByVal Gain As Integer, ByVal c1 As Integer, _
    ByVal c2 As Integer) As Integer
Declare Function A822_IntGetCount Lib "A822.DLL" (dwVal As Long) As Integer
Declare Function A822_IntGetHexBuf Lib "A822.DLL" _
    ByVal dwNum As Long, ByVal wBuffer As Integer) As Integer
Declare Function A822_IntGetFloatBuf Lib "A822.DLL" _
    ByVal dwNum As Integer, ByVal fbuffer As Single) As Integer
Declare Function A822_IntStop Lib "A822.DLL" () As Integer
Declare Function A822_IntRemove Lib "A822.DLL" () As Integer
**DMA Functions**

 Declare Function A822_AD_DMA_InstallIrq Lib "A822.DLL" _
 (ByVal wBase As Integer, ByVal wIrq As Integer, _
 ByVal wDmaChan As Integer) As Integer

 Declare Function A822_AD_DMA_RemoveIrq Lib "A822.DLL" () As Integer

 Declare Function A822_AD_DMA_Start Lib "A822.DLL" _
 (ByVal wCardType As Integer, ByVal Ch As Integer, _
 ByVal Gain As Integer, ByVal c1 As Integer, ByVal c2 As Integer, _
 ByVal cnt As Integer, wPassOut As Integer) As Integer

 Declare Function A822_AD_DMA_Stop Lib "A822.DLL" () As Integer

 Declare Function A822_AD_DMA_IsNotFinished Lib "A822.DLL" () As Integer

 Declare Function A822_AD_DMA_GetBuffer Lib "A822.DLL" _
 (data As Integer) As Integer

 Declare Function A822_AD_DMA_GetFloatBuffer Lib "A822.DLL" _
 (fbuf As Single) As Integer

 ' Function of Channel-Scan with Polling
 Declare Sub A822_ChScan_Clear Lib "A822.DLL" ()

 Declare Function A822_ChScan_Add Lib "A822.DLL" _
 (ByVal wChannel As Integer, ByVal wConfig As Integer) As Integer

 Declare Function A822_ChScan_Set Lib "A822.DLL" _
 (wChannel As Integer, wConfig As Integer, _
 ByVal wChNum As Integer) As Integer

 Declare Function A822_ChScan_PollingHex Lib "A822.DLL" _
 (ByVal wBase As Integer, ByVal wCardType As Integer, _
 wBuf As Integer, ByVal wNumPerCh As Integer) As Integer

 Declare Function A822_ChScan_PollingFloat Lib "A822.DLL" _
 (ByVal wBase As Integer, ByVal wCardType As Integer, _
 fBuf As Single, ByVal wNumPerCh As Integer) As Integer

 ' Function of Channel-Scan with Interrupt
 Declare Function A822_ChScan_IntInstall Lib "A822.DLL" _
 (ByVal wBase As Integer, ByVal wIrq As Integer, _
 hEvent As Long, ByVal dwNumPerCh As Long) As Integer

 Declare Function A822_ChScan_IntStart Lib "A822.DLL" _
 (ByVal c1 As Integer, ByVal c2 As Integer, _
 ByVal wCardType As Integer) As Integer

 Declare Function A822_ChScan_IntGetCount Lib "A822.DLL" _
 (dwVal As Long) As Integer

 Declare Function A822_ChScan_IntGetHexBuf Lib "A822.DLL" _
 (wBuf As Integer) As Integer

 Declare Function A822_ChScan_IntGetFloatBuf Lib "A822.DLL" _
 (fBuf As Single) As Integer

 Declare Function A822_ChScan_IntStop Lib "A822.DLL" () As Integer

 Declare Function A822_ChScan_IntRemove Lib "A822.DLL" () As Integer
2.2.3 A822.BAS (FOR WIN NT)

Attribute VB_Name = "A822"

***************** DEFINE A822 RELATIVE ADDRESS *****************/
Global Const A822_TIMER0 = &H0
Global Const A822_Timer1 = &H1
Global Const A822_TIMER2 = &H2
Global Const A822_TIMER_MODE  = &H3
Global Const A822_AD_LO   = &H4  ' Analog to Digital, Low Byte
Global Const A822_AD_HI   = &H5  ' Analog to Digital, High Byte
Global Const A822_DA_CH0_LO  = &H4  ' Digit to Analog, CH 0
Global Const A822_DA_CH0_HI = &H5
Global Const A822_DA_CH1_LO  = &H6  ' Digit to Analog, CH 1
Global Const A822_DA_CH1_HI = &H7
Global Const A822_DI_LO   = &H6  ' Digit Input
Global Const A822_DO_LO   = &HD  ' Digit Output

Global Const A822_CLEAR_IRQ = &H8
Global Const A822_SET_GAIN = &H9
Global Const A822_SET_CH = &HA
Global Const A822_SET_MODE = &HB
Global Const A822_SOFT_TRIG = &HC

Global Const A822_POLLING_MODE = 1
Global Const A822_DMA_MODE = 2
Global Const A822_INTERRUPT_MODE = 6

*** define the gain mode ***/
Global Const A822_BI_1   = 0
Global Const A822_BI_10  = 1
Global Const A822_BI_100 = 2
Global Const A822_BI_1000 = 3
Global Const A822_UNI_1  = 4
Global Const A822_UNI_10 = 5
Global Const A822_UNI_100 = 6
Global Const A822_UNI_1000 = 7
Global Const A822_BI_05  = 8
Global Const A822_BI_5   = 9
Global Const A822_BI_50  = 10
Global Const A822_BI_500 = 11
Global Const A822\_BI\_2 = 1
Global Const A822\_BI\_4 = 2
Global Const A822\_BI\_8 = 3
Global Const A822\_UNI\_2 = 5
Global Const A822\_UNI\_4 = 6
Global Const A822\_UNI\_8 = 7

Global Const A822PGL = 0
Global Const A822PGH = 1

Global Const A822\_NoError = 0
Global Const A822\_DriverOpenError = 1
Global Const A822\_DriverNoOpen = 2
Global Const A822\_GetDriverVersionError = 3
Global Const A822\_InstallIrqError = 4
Global Const A822\_ClearIntCountError = 5
Global Const A822\_GetIntCountError = 6
Global Const A822\_GetBufferError = 7
Global Const A822\_AdError1 = 100
Global Const A822\_AdError2 = -200#
Global Const A822\_InstallBufError = 10
Global Const A822\_AllocateMemoryError = 11
Global Const A822\_CardTypeError = 12
Global Const A822\_TimeoutError = 13
Global Const A822\_OtherError = 14
Global Const A822\_ConfigCodeError = 15
Global Const A822\_IntStopError = 16
Global Const A822\_IntRemoveError = 17
Global Const A822\_IntInstallEventError = 18
Global Const A822\_BufferFull = 19
Global Const A822\_NoChannelToScan = 20
Global Const A822\_IntInstallChannelError = 21
Global Const A822\_IntInstallConfigError = 22

***** Driver Functions *********
Declare Function A822\_DriverInit Lib "A822.DLL" () As Integer
Declare Sub A822\_DriverClose Lib "A822.DLL" ()
Declare Function A822\_DELAY Lib "A822.DLL" (ByVal wBase As Integer, _
    ByVal wDownCount As Integer) As Integer
Declare Function A822\_Check\_Address Lib "A822.DLL" _
    (ByVal wBase As Integer) As Integer
Declare Sub A822\_SetTriggerMode Lib "A822.DLL"_
    (ByVal wTriggerMode As Integer)
******* Test Functions ***********
Declare Function A822_SHORT_SUB_2 Lib "A822.DLL" _
(ByVal nA As Integer, ByVal nB As Integer) As Integer
Declare Function A822_FLOAT_SUB_2 Lib "A822.DLL" _
(ByVal fA As Single, ByVal fB As Single) As Single
Declare Function A822_Get_DLL_Version Lib "A822.DLL" () As Integer
Declare Function A822_GetDriverVersion Lib "A822.DLL" _
(wDriverVersion As Integer) As Integer

******* Counter Functions ***********
Declare Sub A822_SetCounter Lib "A822.DLL" _
(ByVal wBase As Integer, ByVal wCounterNo As Integer, _
ByVal bCounterMode As Integer, ByVal wCounterValue As Long)
Declare Function A822_ReadCounter Lib "A822.DLL" _
(ByVal wBase As Integer, ByVal wCounterNo As Integer, _
ByVal bCounterMode As Integer) As Long

******* DI/DO Functions ***********
Declare Function A822_DI Lib "A822.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A822_DO Lib "A822.DLL" _
(ByVal wBase As Integer, ByVal wHexValue As Integer)
Declare Sub A822_OutputByte Lib "A822.DLL" _
(ByVal wPortAddr As Integer, ByVal bOutputVal As Byte)
Declare Sub A822_OutputWord Lib "A822.DLL" _
(ByVal wPortAddr As Integer, ByVal wOutputVal As Integer)
Declare Function A822_InputByte Lib "A822.DLL" _
(ByVal wPortAddr As Integer) As Integer
Declare Function A822_InputWord Lib "A822.DLL" _
(ByVal wPortAddr As Integer) As Integer

******* AD Functions ***********
Declare Function A822_SetChGain Lib "A822.DLL" _
(ByVal wBase As Integer, ByVal wChannel As Integer, _
ByVal wConfig As Integer, ByVal wCardType As Integer) As Integer
Declare Function A822_Fast_AD_Hex Lib "A822.DLL" _
(wVal As Integer) As Integer
Declare Function A822_Fast_AD_Float Lib "A822.DLL" _
(fVal As Single) As Integer
Declare Function A822_AD_Hex Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wCardType As Integer, wVal As Integer) As Integer
Declare Function A822_AD_Float Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wCardType As Integer, fVal As Single) As Integer
Declare Function A822_Hex2Float Lib "A822.DLL" (ByVal wConfig As Integer, ByVal wCardType As Integer, ByVal wHex As Integer, fVal As Single) As Integer
Declare Function A822_ADs_Hex Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer, wBuf As Integer, ByVal wCount As Integer) As Integer
Declare Function A822_ADs_Float Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer, fbuf As Single, ByVal wCount As Integer) As Integer

' Please uses the A822_AD_Float() function
Declare Function A822_AD Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wConfig As Integer, ByVal wType As Integer) As Single

****** DA Functions ******
Declare Sub A822_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal wHexValue As Integer)
Declare Sub A822_Uni5_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal fValue As Single)
Declare Sub A822_Uni10_DA Lib "A822.DLL" (ByVal wBase As Integer, ByVal wChannel As Integer, ByVal fValue As Single)
'****** Interrupt Functions **************
' Please use the A822_Intxxxx series function set
Declare Function A822_InstallIrq Lib "A822.DLL" _
    ByVal wBase As Integer, ByVal wIrq As Integer, _
    hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A822_AD_INT_Start Lib "A822.DLL" _
    ByVal wCardType As Integer, ByVal Ch As Integer, _
    ByVal Gain As Integer, ByVal c1 As Integer, _
    ByVal c2 As Integer) As Integer
Declare Function A822_AD_INT_Stop Lib "A822.DLL" () As Integer
Declare Function A822_GetIntCount Lib "A822.DLL" (dwVal As Long) As Integer
Declare Function A822_GetBuffer Lib "A822.DLL" _
    ByVal dwNum As Long, ByVal wBuffer As Integer) As Integer
Declare Function A822_GetFloatBuffer Lib "A822.DLL" _
    ByVal dwNum As Integer, ByVal fbuffer As Single) As Integer

'****** Interrupt Functions **************
Declare Function A822_IntInstall Lib "A822.DLL" _
    ByVal wBase As Integer, ByVal wIrq As Integer, _
    hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A822_IntStart Lib "A822.DLL" _
    ByVal wCardType As Integer, ByVal Ch As Integer, _
    ByVal Gain As Integer, ByVal c1 As Integer, _
    ByVal c2 As Integer) As Integer
Declare Function A822_IntGetCount Lib "A822.DLL" (dwVal As Long) As Integer
Declare Function A822_IntGetHexBuf Lib "A822.DLL" _
    ByVal dwNum As Long, ByVal wBuffer As Integer) As Integer
Declare Function A822_IntGetFloatBuf Lib "A822.DLL" _
    ByVal dwNum As Integer, ByVal fbuffer As Single) As Integer
Declare Function A822_IntStop Lib "A822.DLL" () As Integer
Declare Function A822_IntRemove Lib "A822.DLL" () As Integer
' Function of Channel-Scan with Polling
Declare Sub A822_ChScan_Clear Lib "A822.DLL" ()
Declare Function A822_ChScan_Add Lib "A822.DLL" _
    (ByVal wChannel As Integer, ByVal wConfig As Integer) As Integer
Declare Function A822_ChScan_Set Lib "A822.DLL" _
    (wChannel As Integer, wConfig As Integer, _
     ByVal wChNum As Integer) As Integer
Declare Function A822_ChScan_PollingHex Lib "A822.DLL" _
    (ByVal wBase As Integer, ByVal wCardType As Integer, _
     wBuf As Integer, ByVal wNumPerCh As Integer) As Integer
Declare Function A822_ChScan_PollingFloat Lib "A822.DLL" _
    (ByVal wBase As Integer, ByVal wCardType As Integer, _
     fBuf As Single, ByVal wNumPerCh As Integer) As Integer

' Function of Channel-Scan with Interrupt
Declare Function A822_ChScan_IntInstall Lib "A822.DLL" _
    (ByVal wBase As Integer, ByVal wIrq As Integer, _
     hEvent As Long, ByVal dwNumPerCh As Long) As Integer
Declare Function A822_ChScan_IntStart Lib "A822.DLL" _
    (ByVal c1 As Integer, ByVal c2 As Integer, _
     ByVal wCardType As Integer) As Integer
Declare Function A822_ChScan_IntGetCount Lib "A822.DLL" _
    (dwVal As Long) As Integer
Declare Function A822_ChScan_IntGetHexBuf Lib "A822.DLL" _
    (wBuf As Integer) As Integer
Declare Function A822_ChScan_IntGetFloatBuf Lib "A822.DLL" _
    (fBuf As Single) As Integer
Declare Function A822_ChScan_IntStop Lib "A822.DLL" () As Integer
Declare Function A822_ChScan_IntRemove Lib "A822.DLL" () As Integer
2.3 USING DELPHI

2.3.1 DELPHI DEMO RESULT
2.3.2 A822.PAS (FOR WIN 95/98)

unit A822;

interface

var

PSingle=^Single;
PWord=^Word;
PInteger=^Integer;

Const

//*************** DEFINE A822 RELATIVE ADDRESS ***************

A822_TIMER0 = $00; // Analog to Digital, Low Byte */
A822_TIMER1 = $01; // Analog to Digital, High Byte */
A822_TIMER2 = $02;
A822_TIMER_MODE = $03;
A822_TIMER_MODE = $03;
A822TIMER_MODE = 1;
A822_DMA_MODE = 2;
A822_INTERRUPT_MODE = 6;
///** define the gain mode **/
A822_A822_BI_1        =  0;
A822_A822_BI_10       =  1;
A822_A822_BI_100      =  2;
A822_A822_BI_1000     =  3;
A822_A822_UNI_1       =  4;
A822_A822_UNI_10      =  5;
A822_A822_UNI_100     =  6;
A822_A822_UNI_1000    =  7;
A822_A822_BI_05       =  8;
A822_A822_BI_5        =  9;
A822_A822_BI_50       = 10;
A822_A822_BI_500     = 11;
A822_A822_BI_2        =  1;
A822_A822_BI_4        =  2;
A822_A822_BI_8        =  3;
A822_A822_UNI_2       =  5;
A822_A822_UNI_4       =  6;
A822_A822_UNI_8       =  7;

A822PGL          =  0;
A822PGH          =  1;

A822_NoError                   = 0;
A822_DriverOpenError          = 1;
A822_DriverNoOpen             = 2;
A822_GetDriverVersionError    = 3;
A822_InstallIrqError          = 4;
A822_ClearIntCountError       = 5;
A822_GetIntCountError         = 6;
A822_GetBufferError           = 7;
A822_AdError1                  = 100;
A822_AdError2                  = -200;
A822_InstallBufError          = 10;
A822_AllocateMemoryError      = 11;
A822_CardTypeError            = 12;
A822_TimeoutError             = 13;
A822_OtherError                = 14;
A822_ConfigCodeError          = 15;
A822_IntStopError              = 16;
A822_IntRemoveError            = 17;
A822_IntInstallEventError      = 18;
A822_BufferFull                 = 19;
A822_NoChannelToScan           = 20;
A822_IntInstallChannelError    = 21;
A822_IntInstallConfigError     = 22;
A822_GetDmaStatusError         = 23;

// Function of Driver
Function  A822_DELAY(wBase,wDownCount:WORD):WORD; StdCall;
Function  A822_Check_Address(wBase:WORD):WORD; StdCall;
Function  A822_DriverInit:WORD; StdCall;
Procedure A822_DriverClose; StdCall;
Procedure A822_SetTriggerMode( wTriggerMode:WORD ); StdCall;

// Function of Test
Function  A822_SHORT_SUB_2(nA, nB : SmallInt):SmallInt; StdCall;
Function  A822_FLOAT_SUB_2(fA, fB : Single):Single; StdCall;
Function  A822_Get_DLL_Version:WORD; StdCall;
Function  A822_GetDriverVersion( var wDriverVersion:WORD ); StdCall;

// Function of Counter
Procedure A822_SetCounter( wBase:WORD; wCounterNo:WORD;
                          bCounterMode:WORD; wCounterValue:LongInt); StdCall;
Function  A822_ReadCounter( wBase:WORD; wCounterNo:WORD;
                          bCounterMode:WORD );LongInt; StdCall;

// Function of DI/DO
Procedure A822_DO(wBase, wHexValue:Word); StdCall;
Function  A822_DI(wBase:Word):Word; StdCall;

Procedure A822_OutputByte
    (wPortAddr:WORD; bOutputVal:Byte); StdCall;
Procedure A822_OutputWord
    (wPortAddr:WORD; wOutputVal:WORD); StdCall;
Function  A822_InputByte(wPortAddr:WORD):WORD; StdCall;
Function  A822_InputWord(wPortAddr:WORD):WORD; StdCall;
// Function of AD/DA
Function A822_SetChGain(wBase:WORD; wChannel:WORD;
   wConfig:WORD; wCardType:WORD):Word; StdCall;
Function A822_Fast_AD_Hex(var wVal:WORD):Word; StdCall;
Function A822_Fast_AD_Float(var fVal:Single):Word; StdCall;
Function A822_AD_Hex(wBase:WORD; wChannel:WORD; wConfig:WORD;
   wCardType:WORD; var wVal:WORD):Word; StdCall;
Function A822_AD_Float( wBase:WORD; wChannel:WORD; wConfig:WORD;
   wCardType:WORD; var fVal:Single):Word; StdCall;
Function A822_Hex2Float(wConfig:WORD; wCardType:WORD;
   wHex:WORD; var fVal:Single):Word; StdCall;
Function A822_ADs_Hex(  wBase,wChannel,wConfig,wType:WORD;
   wBuf:PInteger; wCount:WORD):WORD; StdCall;
Function A822_ADs_Float(wBase,wChannel,wConfig,wType:WORD;
   fBuf:PSingle; wCount:WORD):WORD; StdCall;

// Please uses the A822_AD_Float() function
Function A822_AD(wBase, wChannel, wConfig, wType:WORD)
   :Single; StdCall;

// Function of DA
Procedure A822_DA(wBase, wChannel, wHexValue:WORD); StdCall;
Procedure A822_Uni5_DA
   (wBase, wChannel:Word; fValue:Single); StdCall;
Procedure A822_Uni10_DA
   (wBase, wChannel:Word; fValue:Single); StdCall;

// Function of Interrupt
// Please uses the A822_Instxxxx series function set
Function A822_InstallIrq(wBase, wlrq:WORD; var hEvent:LongInt;
   dwCount:LongInt):WORD; StdCall;
Function A822_GetBuffer
   (dwNum:LongInt; wBuffer:PInteger):WORD; StdCall;
Function A822_GetFloatBuffer
   (dwNum:LongInt; fBuffer:PSingle):WORD; StdCall;
Function A822_GetIntCount(var dwVal:LongInt):WORD; StdCall;
Function A822_AD_INT_Start
   (wCardType,Ch,Gain,c1,c2:WORD):WORD; StdCall;
Function A822_AD_INT_Stop:WORD; StdCall;
// Function of Interrupt
Function A822_IntInstall(wBase, wIrq:WORD;
    var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function A822_IntStart
    (wCardType,Ch,Gain,c1,c2:WORD):WORD; StdCall;
Function A822_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A822_IntGetHexBuf
    (dwNum:LongInt; wBuffer:PInteger):WORD; StdCall;
Function A822_IntGetFloatBuf
    (dwNum:LongInt; fBuffer:PSingle):WORD; StdCall;
Function A822_IntStop:WORD; StdCall;
Function A822_IntRemove:WORD; StdCall;

// Function of DMA
Function A822_AD_DMA_InstallIrq
    (wBase,wIrq,wDmaChan:WORD):WORD; StdCall;
Function A822_AD_DMA_RemoveIrq:WORD; StdCall;
Function A822_AD_DMA_Start(wCardType,Ch,Gain:WORD; c1,c2:WORD;
    cnt:integer; wPassOut:PInteger):WORD; StdCall;
Function A822_AD_DMA_Stop:WORD; StdCall;
Function A822_AD_DMA_IsNotFinished:WORD; StdCall;
Function A822_AD_DMA_GetBuffer(wData:PWORD):WORD; StdCall;
Function A822_AD_DMA_GetFloatBuffer
    (fBuf:PSingle):WORD; StdCall;

// Function of Channel-Scan with Polling
procedure A822_ChScan_Clear; StdCall;
Function A822_ChScan_Add
    ( wChannel:WORD; wConfig:WORD):WORD; StdCall;
Function A822_ChScan_Set(wChannel:PWord; wConfig:PWord;
    wChNum:WORD):WORD; StdCall;
Function A822_ChScan_PollingHex(wBase:WORD; wCardType:WORD;
    wBuf:PWord; wNumPerCh:WORD):WORD; StdCall;
Function A822_ChScan_PollingFloat(wBase:WORD; wCardType:WORD;
    fBuf:PSingle; wNumPerCh:WORD):WORD; StdCall;

// Function of Channel-Scan with Interrupt
Function A822_ChScan_IntInstall(wBase:WORD; wIrq:WORD;
    var hEvent:LongInt; dwNumPerCh:LongInt):WORD; StdCall;
Function A822_ChScan_IntStart
    (c1:WORD; c2:WORD; wCardType:WORD):WORD; StdCall;
Function A822_ChScan_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A822_ChScan_IntGetHexBuf(wBuf:PWord):WORD; StdCall;
Function A822_ChScan_IntGetFloatBuf(fBuf:PSingle):WORD; StdCall;
Function A822_ChScan_IntStop:WORD; StdCall;
Function A822_ChScan_IntRemove:WORD; StdCall;

Ver: 3.2          Date: 11/27/00                                                   Page:  40
implementation

// Function of Driver
Function A822_DriverInit;                       external 'A822.DLL'
    name 'A822_DriverInit';
Procedure A822_DriverClose;                    external 'A822.DLL'
    name 'A822_DriverClose';
Function A822_DELAY;                           external 'A822.DLL'
    name 'A822_DELAY';
Function A822_Check_Address;                   external 'A822.DLL'
    name 'A822_Check_Address';
Procedure A822_SetTriggerMode;                 external 'A822.DLL'
    name 'A822_SetTriggerMode';

// Function of Test
Function A822_SHORT_SUB_2;                     external 'A822.DLL'
    name 'A822_SHORT_SUB_2';
Function A822_FLOAT_SUB_2;                     external 'A822.DLL'
    name 'A822_FLOAT_SUB_2';
Function A822_Get_DLL_Version;                 external 'A822.DLL'
    name 'A822_Get_DLL_Version';
Function A822_GetDriverVersion;                external 'A822.DLL'
    name 'A822_GetDriverVersion';

// Function of Counter
Procedure A822_SetCounter;                    external 'A822.DLL'
    name 'A822_SetCounter';
Function A822_ReadCounter;                     external 'A822.DLL'
    name 'A822_ReadCounter';

// Function of DI/O
Procedure A822_DO;                            external 'A822.DLL'
    name 'A822_DO';
Function A822_DI;                              external 'A822.DLL'
    name 'A822_DI';
Procedure A822_OutputByte;                     external 'A822.DLL'
    name 'A822_OutputByte';
Procedure A822_OutputWord;                     external 'A822.DLL'
    name 'A822_OutputWord';
Function A822_InputByte;                       external 'A822.DLL'
    name 'A822_InputByte';
Function A822_InputWord;                       external 'A822.DLL'
    name 'A822_InputWord';
// Function of AD
Function A822_SetChGain;                   external 'A822.DLL'
    name 'A822_SetChGain';
Function A822_Fast_AD_Hex;                 external 'A822.DLL'
    name 'A822_Fast_AD_Hex';
Function A822_Fast_AD_Float;               external 'A822.DLL'
    name 'A822_Fast_AD_Float';
Function A822_AD_Hex;                      external 'A822.DLL'
    name 'A822_AD_Hex';
Function A822_AD_Float;                    external 'A822.DLL'
    name 'A822_AD_Float';
Function A822_Hex2Float;                   external 'A822.DLL'
    name 'A822_Hex2Float';
Function A822_ADs_Hex;                     external 'A822.DLL'
    name 'A822_ADs_Hex';
Function A822_ADs_Float;                   external 'A822.DLL'
    name 'A822_ADs_Float';

// Please uses the A822_AD_Float() function
Function A822_AD;                          external 'A822.DLL'
    name 'A822_AD';

// Function of DA
Procedure A822_DA;                          external 'A822.DLL'
    name 'A822_DA';
Procedure A822_Uni5_DA;                     external 'A822.DLL'
    name 'A822_Uni5_DA';
Procedure A822_Uni10_DA;                    external 'A822.DLL'
    name 'A822_Uni10_DA';

// Function of Interrupt
// Please uses the A822_Intxxxx series function set
Function A822_InstallIrq;                   external 'A822.DLL'
    name 'A822_InstallIrq';
Function A822_AD_INT_Start;                 external 'A822.DLL'
    name 'A822_AD_INT_Start';
Function A822_AD_INT_Stop;                  external 'A822.DLL'
    name 'A822_AD_INT_Stop';
Function A822_GetIntCount;                  external 'A822.DLL'
    name 'A822_GetIntCount';
Function A822_GetBuffer;                    external 'A822.DLL'
    name 'A822_GetBuffer';
Function A822_GetFloatBuffer;               external 'A822.DLL'
    name 'A822_GetFloatBuffer';
// Function of Interrupt
Function A822_IntInstall; external 'A822.DLL'
    name 'A822_IntInstall';
Function A822_IntStart; external 'A822.DLL'
    name 'A822_IntStart';
Function A822_IntGetCount; external 'A822.DLL'
    name 'A822_IntGetCount';
Function A822_IntGetHexBuf; external 'A822.DLL'
    name 'A822_IntGetHexBuf';
Function A822_IntGetFloatBuf; external 'A822.DLL'
    name 'A822_IntGetFloatBuf';
Function A822_IntStop; external 'A822.DLL'
    name 'A822_IntStop';
Function A822_IntRemove; external 'A822.DLL'
    name 'A822_IntRemove';

// Function of DMA
Function A822_AD_DMA_InstallIrq; external 'A822.DLL'
    name 'A822_AD_DMA_InstallIrq';
Function A822_AD_DMA_RemoveIrq; external 'A822.DLL'
    name 'A822_AD_DMA_RemoveIrq';
Function A822_AD_DMA_Start; external 'A822.DLL'
    name 'A822_AD_DMA_Start';
Function A822_AD_DMA_Stop; external 'A822.DLL'
    name 'A822_AD_DMA_Stop';
Function A822_AD_DMA_IsNotFinished; external 'A822.DLL'
    name 'A822_AD_DMA_IsNotFinished';
Function A822_AD_DMA_GetBuffer; external 'A822.DLL'
    name 'A822_AD_DMA_GetBuffer';
Function A822_AD_DMA_GetFloatBuffer; external 'A822.DLL'
    name 'A822_AD_DMA_GetFloatBuffer';

// Function of Channel-Scan with Polling
procedure A822_ChScan_Clear; external 'A822.DLL'
    name 'A822_ChScan_Clear';
Function A822_ChScan_Add; external 'A822.DLL'
    name 'A822_ChScan_Add';
Function A822_ChScan_Set; external 'A822.DLL'
    name 'A822_ChScan_Set';
Function A822_ChScan_PollingHex; external 'A822.DLL'
    name 'A822_ChScan_PollingHex';
Function A822_ChScan_PollingFloat; external 'A822.DLL'
    name 'A822_ChScan_PollingFloat';
// Function of Channel-Scan with Interrupt
Function A822_ChScan_IntInstall; name 'A822_ChScan_IntInstall'; external 'A822.DLL'
Function A822_ChScan_IntStart; name 'A822_ChScan_IntStart';
Function A822_ChScan_IntGetCount; name 'A822_ChScan_IntGetCount';
Function A822_ChScan_IntGetHexBuf; name 'A822_ChScan_IntGetHexBuf';
Function A822_ChScan_IntGetFloatBuf; name 'A822_ChScan_IntGetFloatBuf';
Function A822_ChScan_IntStop; name 'A822_ChScan_IntStop';
Function A822_ChScan_IntRemove; name 'A822_ChScan_IntRemove';
end.
2.3.3 A822.PAS (FOR WIN NT)

unit A822;

interface

type PSingle=^Single;
  PWord=^Word;
  PInteger=^Integer;

Const
//***************** DEFINE A822 RELATIVE ADDRESS *****************/
A822_TIMER0     =  $00;
A822_TIMER1     =  $01;
A822_TIMER2     =  $02;
A822_TIMER_MODE =  $03;
A822_AD_LO      =  $04;  // Analog to Digital, Low Byte */
A822_AD_HI      =  $05;  // Analog to Digital, High Byte */
A822_DA_CH0_LO  =  $04;  // Digit to Analog, CH 0 */
A822_DA_CH0_HI  =  $05;
A822_DA_CH1_LO  =  $06;  // Digit to Analog, CH 1 */
A822_DA_CH1_HI  =  $07;
A822_DI_LO      =  $06;  // Digit Input */
A822_DO_LO      =  $0D;  // Digit Output */
A822_CLEAR_IRQ  =  $08;
A822_SET_GAIN   =  $09;
A822_SET_CH     =  $0A;
A822_SET_MODE   =  $0B;
A822_SOFT_TRIG  =  $0C;

A822_POLLING_MODE       =  1;
A822_DMA_MODE           =  2;
A822_INTERRUPT_MODE     =  6;
//*** define the gain mode ***/
A822_A822_BI_1 = 0;
A822_A822_BI_10 = 1;
A822_A822_BI_100 = 2;
A822_A822_BI_1000 = 3;
A822_A822_UNI_1 = 4;
A822_A822_UNI_10 = 5;
A822_A822_UNI_100 = 6;
A822_A822_UNI_1000 = 7;
A822_A822_BI_05 = 8;
A822_A822_BI_5 = 9;
A822_A822_BI_50 = 10;
A822_A822_BI_500 = 11;

A822_A822_BI_2 = 1;
A822_A822_BI_4 = 2;
A822_A822_BI_8 = 3;
A822_A822_UNI_2 = 5;
A822_A822_UNI_4 = 6;
A822_A822_UNI_8 = 7;

A822PGL = 0;
A822PGH = 1;

A822_NoError = 0;
A822_DriverOpenError = 1;
A822_DriverNoOpen = 2;
A822_GetDriverVersionError = 3;
A822_InstallIrqError = 4;
A822_ClearIntCountError = 5;
A822_GetIntCountError = 6;
A822_GetBufferError = 7;
A822_AdError1 = 100;
A822_AdError2 = -200;
A822_InstallBufError = 10;
A822_AllocateMemoryError = 11;
A822/CardTypeError = 12;
A822_TimeoutError = 13;
A822_OtherError = 14;
A822_ConfigCodeError = 15;
A822_IntStopError           = 16;
A822_IntRemoveError         = 17;
A822_IntInstallEventError   = 18;
A822_BufferFull             = 19;
A822_NoChannelToScan        = 20;
A822_IntInstallChannelError = 21;
A822_IntInstallConfigError  = 22;

// Function of Driver
Function    A822_DELAY(wBase,wDownCount:WORD):WORD; StdCall;
Function    A822_Check_Address(wBase:WORD):WORD; StdCall;
Function    A822_DriverInit:WORD; StdCall;
Procedure   A822_DriverClose; StdCall;
Procedure   A822_SetTriggerMode( wTriggerMode:WORD ); StdCall;

// Function of Test
Function    A822_SHORT_SUB_2(nA, nB : SmallInt):SmallInt; StdCall;
Function    A822_FLOAT_SUB_2(fA, fB : Single):Single; StdCall;
Function    A822_Get_DLL_Version:WORD; StdCall;
Function    A822_GetDriverVersion
               (var wDriverVersion:WORD):Word; StdCall;

// Function of Counter
Procedure   A822_SetCounter( wBase:WORD; wCounterNo:WORD;
                          bCounterMode:WORD; wCounterValue:LongInt); StdCall;
Function    A822_ReadCounter( wBase:WORD; wCounterNo:WORD;
                           bCounterMode:WORD ):LongInt; StdCall;

// Function of DI/DO
Procedure   A822_DO(wBase, wHexValue:Word); StdCall;
Function    A822_DI(wBase:Word):Word; StdCall;
Procedure   A822_OutputByte
               (wPortAddr:WORD; bOutputVal:Byte); StdCall;
Procedure   A822_OutputWord
               (wPortAddr:WORD; wOutputVal:WORD); StdCall;
Function    A822_InputByte(wPortAddr:WORD):WORD; StdCall;
Function    A822_InputWord(wPortAddr:WORD):WORD; StdCall;
// Function of AD/DA
Function A822_SetChGain(wBase:WORD; wChannel:WORD;
  wConfig:WORD; wCardType:WORD):Word; StdCall;
Function A822_Fast_AD_Hex(var wVal:WORD):Word; StdCall;
Function A822_Fast_AD_Float(var fVal:Single):Word; StdCall;

Function A822_AD_Hex(wBase:WORD; wChannel:WORD; wConfig:WORD;
  wCardType:WORD; var wVal:WORD):Word; StdCall;
Function A822_AD_Float(wBase:WORD; wChannel:WORD; wConfig:WORD;
  wCardType:WORD; var fVal:Single):Word; StdCall;
Function A822_Hex2Float(wConfig:WORD; wCardType:WORD;
  wHex:WORD; var fVal:Single):Word; StdCall;
Function A822_ADs_Hex( wBase,wChannel,wConfig,wType:WORD;
  wBuf:PInteger; wCount:WORD):WORD; StdCall;
Function A822_ADs_Float(wBase,wChannel,wConfig,wType:WORD;
  fBuf:PSingle; wCount:WORD):WORD; StdCall;

// Please uses the A822_AD_Float() function
Function A822_AD(wBase, wChannel, wConfig, wType:WORD)
  :Single; StdCall;

// Function of DA
Procedure A822_DA(wBase, wChannel, wHexValue:WORD); StdCall;
Procedure A822_Uni5_DA
  (wBase, wChannel:Word; fValue:Single); StdCall;
Procedure A822_Uni10_DA
  (wBase, wChannel:Word; fValue:Single); StdCall;

// Function of Interrupt
// Please uses the A822_Intxxxx series function set
Function A822_InstallIrq(wBase, wIrq:WORD; var hEvent:LongInt;
  dwCount:LongInt):WORD; StdCall;
Function A822_GetBuffer
  (dwNum:LongInt; wBuffer:PInteger):WORD; StdCall;
Function A822_GetFloatBuffer
  (dwNum:LongInt; fBuffer:PSingle):WORD; StdCall;
Function A822_GetIntCount(var dwVal:LongInt):WORD; StdCall;
Function A822_AD_INT_Start
  (wCardType,Ch,Gain,c1,c2:WORD):WORD; StdCall;
Function A822_AD_INT_Stop:WORD; StdCall;
// Function of Interrupt
Function A822_IntInstall(wBase, wIrq:WORD;
    var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function A822_IntStart
    (wCardType, Ch, Gain, c1, c2:WORD):WORD; StdCall;
Function A822_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A822_IntGetHexBuf
    (dwNum:LongInt; wBuffer:PInteger):WORD; StdCall;
Function A822_IntGetFloatBuf
    (dwNum:LongInt; fBuffer:PSingle):WORD; StdCall;
Function A822_IntStop:WORD; StdCall;
Function A822_IntRemove:WORD; StdCall;

// Function of Channel-Scan with Polling
procedure A822_ChScan_Clear; StdCall;
Function A822_ChScan_Add
    (wChannel:WORD; wConfig:WORD):WORD; StdCall;
Function A822_ChScan_Set(wChannel:PWord; wConfig:PWord;
    wChNum:WORD):WORD; StdCall;
Function A822_ChScan_PollingHex(wBase:WORD; wCardType:WORD;
    wBuf:PWord; wNumPerCh:WORD):WORD; StdCall;
Function A822_ChScan_PollingFloat(wBase:WORD; wCardType:WORD;
    fBuf:PSingle; wNumPerCh:WORD):WORD; StdCall;

// Function of Channel-Scan with Interrupt
Function A822_ChScan_IntInstall(wBase:WORD; wIrq:WORD;
    var hEvent:LongInt; dwNumPerCh:LongInt):WORD; StdCall;
Function A822_ChScan_IntStart
    (c1:WORD; c2:WORD; wCardType:WORD):WORD; StdCall;
Function A822_ChScan_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A822_ChScan_IntGetHexBuf(wBuf:PWord):WORD; StdCall;
Function A822_ChScan_IntGetFloatBuf(fBuf:PSingle):WORD; StdCall;
Function A822_ChScan_IntStop:WORD; StdCall;
Function A822_ChScan_IntRemove:WORD; StdCall;
implementation

// Function of Driver
Function A822_DriverInit;                  
  external 'A822.DLL'
  name 'A822_DriverInit';
Procedure A822_DriverClose;                
  external 'A822.DLL'
  name 'A822_DriverClose';
Function A822_DELAY;                       
  external 'A822.DLL'
  name 'A822_DELAY';
Function A822_Check_Address;              
  external 'A822.DLL'
  name 'A822_Check_Address';
Procedure A822_SetTriggerMode;             
  external 'A822.DLL'
  name 'A822_SetTriggerMode';

// Function of Test
Function A822_SHORT_SUB_2;                
  external 'A822.DLL'
  name 'A822_SHORT_SUB_2';
Function A822_FLOAT_SUB_2;                
  external 'A822.DLL'
  name 'A822_FLOAT_SUB_2';
Function A822_Get_DLL_Version;            
  external 'A822.DLL'
  name 'A822_Get_DLL_Version';
Function A822_GetDriverVersion;           
  external 'A822.DLL'
  name 'A822_GetDriverVersion';

// Function of Counter
Procedure A822_SetCounter;                  
  external 'A822.DLL'
  name 'A822_SetCounter';
Function A822_ReadCounter;                 
  external 'A822.DLL'
  name 'A822_ReadCounter';

// Function of D/I/O
Procedure A822_DO;                          
  external 'A822.DLL'
  name 'A822_DO';
Function A822_DI;                           
  external 'A822.DLL'
  name 'A822_DI';
Procedure A822_OutputByte;                  
  external 'A822.DLL'
  name 'A822_OutputByte';
Procedure A822_OutputWord;                  
  external 'A822.DLL'
  name 'A822_OutputWord';
Function A822_InputByte;                    
  external 'A822.DLL'
  name 'A822_InputByte';
Function A822_InputWord;                    
  external 'A822.DLL'
  name 'A822_InputWord';

Ver: 3.2          Date: 11/27/00                                                   Page:  50
// Function of AD
Function A822_SetChGain;                   external 'A822.DLL'
  name 'A822_SetChGain';
Function A822_Fast_AD_Hex;                 external 'A822.DLL'
  name 'A822_Fast_AD_Hex';
Function A822_Fast_AD_Float;               external 'A822.DLL'
  name 'A822_Fast_AD_Float';
Function A822_AD_Hex;                      external 'A822.DLL'
  name 'A822_AD_Hex';
Function A822_AD_Float;                    external 'A822.DLL'
  name 'A822_AD_Float';
Function A822_Hex2Float;                   external 'A822.DLL'
  name 'A822_Hex2Float';
Function A822_ADs_Hex;                     external 'A822.DLL'
  name 'A822_ADs_Hex';
Function A822_ADs_Float;                   external 'A822.DLL'
  name 'A822_ADs_Float';

// Please uses the A822_AD_Float() function
Function A822_AD;                          external 'A822.DLL'
  name 'A822_AD';

// Function of DA
Procedure A822_DA;                          external 'A822.DLL'
  name 'A822_DA';
Procedure A822_Uni5_DA;                     external 'A822.DLL'
  name 'A822_Uni5_DA';
Procedure A822_Uni10_DA;                    external 'A822.DLL'
  name 'A822_Uni10_DA';

// Function of Interrupt
// Please uses the A822_Intxxxx series function set
Function A822_InstallIrq;                   external 'A822.DLL'
  name 'A822_InstallIrq';
Function A822_AD_INT_Start;                external 'A822.DLL'
  name 'A822_AD_INT_Start';
Function A822_AD_INT_Stop;                 external 'A822.DLL'
  name 'A822_AD_INT_Stop';
Function A822_GetIntCount;                 external 'A822.DLL'
  name 'A822_GetIntCount';
Function A822_GetBuffer;                   external 'A822.DLL'
  name 'A822_GetBuffer';
Function A822_GetFloatBuffer;              external 'A822.DLL'
  name 'A822_GetFloatBuffer';
// Function of Interrupt
Function A822_IntInstall;
   name 'A822_IntInstall';
   external 'A822.DLL'
Function A822_IntStart;
   name 'A822_IntStart';
Function A822_IntGetCount;
   name 'A822_IntGetCount';
Function A822_IntGetHexBuf;
   name 'A822_IntGetHexBuf';
Function A822_IntGetFloatBuf;
   name 'A822_IntGetFloatBuf';
Function A822_IntStop;
   name 'A822_IntStop';
Function A822_IntRemove;
   name 'A822_IntRemove';

// Function of Channel-Scan with Polling
procedure A822_ChScan_Clear;
   name 'A822_ChScan_Clear';
Function A822_ChScan_Add;
   name 'A822_ChScan_Add';
Function A822_ChScan_Set;
   name 'A822_ChScan_Set';
Function A822_ChScan_PollingHex;
   name 'A822_ChScan_PollingHex';
Function A822_ChScan_PollingFloat;
   name 'A822_ChScan_PollingFloat';

// Function of Channel-Scan with Interrupt
Function A822_ChScan_IntInstall;
   name 'A822_ChScan_IntInstall';
Function A822_ChScan_IntStart;
   name 'A822_ChScan_IntStart';
Function A822_ChScan_IntGetCount;
   name 'A822_ChScan_IntGetCount';
Function A822_ChScan_IntGetHexBuf;
   name 'A822_ChScan_IntGetHexBuf';
Function A822_ChScan_IntGetFloatBuf;
   name 'A822_ChScan_IntGetFloatBuf';
Function A822_ChScan_IntStop;
   name 'A822_ChScan_IntStop';
Function A822_ChScan_IntRemove;
   name 'A822_ChScan_IntRemove';
end.
3. FUNCTION DESCRIPTION

The functions in the DLL are divided into several groups as follows:
1. Not supported functions
2. The Driver functions
3. The Test functions
4. The Counter functions
5. The DI/O functions
6. The DA functions
7. The AD Polling functions
8. The AD Interrupt functions
9. The AD DMA functions
10. The AD Channel-Scan Polling functions
11. The AD Channel-Scan Interrupt functions

(The DMA functions supported under Windows 95/98 only)

In this chapter, we use some keywords to indicate the attribute of Parameters.

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Parameter Set by user before calling this function ?</th>
<th>User Gets the data/value from this parameter after calling this function ?</th>
</tr>
</thead>
<tbody>
<tr>
<td>[In]</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>[Out]</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>[In, Out]</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note: All of the parameters need space allocation by the user.
# 3.1 ERROR CODES

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A822_NoError</td>
<td>OK!</td>
</tr>
<tr>
<td>A822_DriverOpenError</td>
<td>Failed to open the device driver. Please check if the card is installed properly in your computer and is the device driver installed. Or, please try to the card in the other slot after re-installing the device driver.</td>
</tr>
<tr>
<td>A822_DriverNoOpen</td>
<td>Users have to call the A822_DriverInit() function before calling other A822 functions.</td>
</tr>
<tr>
<td>A822_GetDriverVersionError</td>
<td>Failed to communication with device driver. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_InstallIrqError</td>
<td>Failed to install the ISR with the specified IRQ/DMA number. Please check if the driver is installed? Check does the IRQ/IO address and DMA resource conflicts with other devices? And check the system's resources and free some resources if necessary.</td>
</tr>
<tr>
<td>A822_ClearIntCountError</td>
<td>Failed to communication with device driver. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_GetIntCountError</td>
<td>Failed to communication with device driver. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_GetBufferError</td>
<td>Failed to communication with device driver. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_AllocateMemoryError</td>
<td>Fail to allocate memory for data buffer. Please check your system's resources and free some memory.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A822_CardTypeError</td>
<td>The CardType should be 0: A822PGL or 1: A822PGH</td>
</tr>
<tr>
<td>A822_TimeoutError</td>
<td>For A/D (Analog Input) functions, the DLL functions are waiting for A/D converter to complete the operation. The Max. waiting time is 500ms. It may always return the A822_TimeoutError, if the card is set for the Trigger-Mode to External-Trigger by jumper. (Thus, software-trigger will not function.) Please check your hardware settings of Base address. Try to install the card in other slot.</td>
</tr>
<tr>
<td>A822_ConfigCodeError</td>
<td>For valid configuration codes, please refer to Section &quot;1.2 Range Configuration&quot;</td>
</tr>
<tr>
<td>A822_IntStopError</td>
<td>Failed to communication with driver, or failed to stop the interrupt. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_IntRemoveError</td>
<td>Failed to communication with driver, or failed to remove the ISR/DMA. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_IntInstallEventError</td>
<td>Failed to install the event-object into device driver. Please check if the driver is installed? Check the system's resources and free some resources if necessary. Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_BufferFull</td>
<td>The buffer size of the Channel-Scan List is 100. Program can't add more than 100 channels.</td>
</tr>
<tr>
<td>A822_NoChannelToScan</td>
<td>Before calling the Channel-Scan Polling or Interrupt, users have to set the channels into the Channel-Scan List by calling related functions. Please refer to A822_ChScan_Clear(), A822_ChScan_Add() and A822_ChScan_Set() functions.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>A822_IntInstallChannelError</td>
<td>Failed to copy the channels of Channel-Scan List into device driver. Please check if the driver is installed? Check the system's resources and free some resources if possible. Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_IntInstallConfigError</td>
<td>Failed to copy the configuration-code of Channel-Scan List into device driver. Please check if the driver is installed? Check the system's resources and free some resources if possible. Or, try to re-install driver.</td>
</tr>
<tr>
<td>A822_GetDmaStatusError</td>
<td>Failed to communication with driver, or failed to get the DMA completion status. Please check if the driver is installed? Or, try to re-install driver.</td>
</tr>
</tbody>
</table>
### 3.2 FUNCTIONS NOT SUPPORTED

The following functions are not supported:

<table>
<thead>
<tr>
<th>Not Supported function</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>A822_AD()</td>
<td>Please refer to A822_AD_Float() function. This new function separates the error-code and A/D value.</td>
</tr>
<tr>
<td>A822_InstallIrq(), A822_AD_INT_Start(), A822_GetIntCount(), A822_GetBuffer(), A822_GetFloatBuffer(), A822_AD_INT_Stop()</td>
<td>Please refer to the following new functions… A822_IntInstall(), A822_IntStart(), A822_IntGetCount(), A822_IntGetHexBuf(), A822_IntGetFloatBuf(), A822_IntStop(), A822_IntRemove() These new function set has the better performance. User's program has to allocate the event-object and data buffer once.</td>
</tr>
</tbody>
</table>
3.3 DRIVER FUNCTIONS

3.3.1 A822_DriverInit

- **Description:**
  This subroutine will initialize the device driver and allocate the resources.

- **Syntax:**
  WORD A822_DriverInit(void);

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.3.2 A822_DriverClose

- **Description:**
  This subroutine will close the device driver and free the resources.

- **Syntax:**
  void A822_DriverClose(void);

- **Parameter:**
  None

- **Return:**
  None
### A822_DELAY

**Description:**
This subroutine will delay `wDownCount` (machine independent timer). This function uses the Counter0 to implement delay and will be used by the A/D related functions. The unit of `A822_DELAY()` is 0.5μSeconds. (2MHz → 2000K times/sec).

For Example: `A822_DELAY(2000);` → delays 1 mSeconds.

**Syntax:**
```c
WORD A822_DELAY(WORD wBase, WORD wDownCount);
```

**Parameter:**
- `wBase` : [In] I/O port base address, for example, 0x220
- `wDownCount` : [In] Number of count that will be delay, 2 count = 1 uSeconds.

**Return:**
Refer to "Section 3.1 Error Codes".

### A822_Check_Address

**Description:**
This subroutine will detect the OME-A-822PGH/L in I/O base address = `wBase`. This subroutine will perform one A/D conversion, if successful → found an OME-A-822PGH/L. This function will always return 0 if the user sets the trigger mode to external. Refer to the function "A822_SetTriggerMode".

**Syntax:**
```c
WORD A822_Check_Address(WORD wBase);
```

**Parameter:**
- `wBase` : [In] I/O port base address, for example, 0x220

**Return:**
Refer to "Section 3.1 Error Codes".
### 3.3.5 A822_SetTriggerMode

- **Description:**
  This subroutine will set the trigger mode to internal or external. The default value is set to internal trigger mode if the user does not use this function. The user has to call this function before calling any A/D function (include the function "A822_Check_Address") if the user uses external trigger mode.

  Please refer to the hardware manual to set the jumper JP4 (A/D Trigger Source Selection). The JP4 default setting is "INTTRG" (Internal-Trigger).

- **Syntax:**
  ```c
  void A822_SetTriggerMode(WORD wTriggerMode )
  ```

- **Parameter:**
  - `wTriggerMode` : [In] 0: Internal Trigger Mode
  - 1: External Trigger Mode

- **Return:**
  None
3.4 TEST FUNCTION

3.4.1 A822_SHORT_SUB_2

- **Description:**
  
  Compute C=A-B in **short** formats, **short = 16-bit signed integer**. This function is provided for testing purpose.

- **Syntax:**
  
  short A822_SHORT_SUB_2(short nA, short nB);

- **Parameter:**
  
  nA : [in] short integer  
  nB : [in] short integer

- **Return:**
  
  return=nA-nB \rightarrow \text{short integer}

3.4.2 A822_FLOAT_SUB_2

- **Description:**
  
  Compute A-B in **float** format, **float = 32-bit floating pointer number**. This function is provided for testing purpose.

- **Syntax:**
  
  float A822_FLOAT_SUB_2(float fA, float fB);

- **Parameter:**
  
  fA : [in] floating point value  
  fB : [in] floating point value

- **Return:**
  
  return=fA-fB \rightarrow \text{floating point value}
3.4.3 A822_Get_DLL_Version

- **Description:**
  Read the software version of the A822.DLL.

- **Syntax:**
  WORD A822_Get_DLL_Version(void);

- **Parameter:**
  None

- **Return:**
  Returns the DLL's version, for example 0x200 → Version 2.00
  *(WORD = 16-bit unsigned integer)*

3.4.4 A822_GetDriverVersion

- **Description:**
  This subroutine will get the version number for the device driver.

- **Syntax:**
  WORD A822_GetDriverVersion(WORLD *wDriverVersion);

- **Parameter:**
  For example: wDriverVerion=0x210 → version 2.10

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.5 COUNTER FUNCTION

3.5.1 A822_SetCounter

- **Description:**
  This subroutine will set the 8254 counter mode and value.

- **Syntax:**
  ```c
  void A822_SetCounter(WORD wBase, WORD wCounterNo,
   WORD bCounterMode, DWORD wCounterValue);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wCounterNo` : [In] Counter Number 0 to 2 for the 8254
  - `wCounterMode` : [In] Counter Mode 0 to 5 for the 8254
  - `wCounterValue` : [In] Counter Value 0 to 65535 for the 8254

- **Return:**
  None

3.5.2 A822_ReadCounter

- **Description:**
  This subroutine will read the 8254 counter value.

- **Syntax:**
  ```c
  DWORD A822_ReadCounter(WORD wBase, WORD wCounterNo,
   WORD bCounterMode);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wCounterNo` : [In] Counter Number 0 to 2 for the 8254
  - `wCounterMode` : [In] Counter Mode 0 to 5 for the 8254

- **Return:**
  Return the counter’s value and only the lower WORD is valid.
3.6  DI/DO FUNCTION

3.6.1  A822_DI

- **Description:**
  This subroutine will read the 16-bit data from the digital input port.

- **Syntax:**
  ```c
  WORD A822_DI(WORD wBase);
  ```

- **Parameter:**
  wBase : [In] I/O port base address, for example, 0x220

- **Return:**
  16-bit data read from the digital input port

3.6.2  A822_DO

- **Description:**
  This subroutine will send 16-bit data to digital output port.

- **Syntax:**
  ```c
  void A822_DO(WORD wBase, WORD wHexValue);
  ```

- **Parameter:**
  wBase : [In] I/O port base address, for example, 0x220
  wHexValue : [In] 16-bit data send to digital output port

- **Return:**
  None
3.6.3  A822_OutputByte

- **Description:**
  This subroutine will send the 8-bit data to the desired I/O port.

- **Syntax:**
  ```c
  void A822_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
  ```

- **Parameter:**
  - `wPortAddr`: [In] I/O port address, for example, 0x220
  - `bOutputVal`: [In] 8-bit data send to I/O port

- **Return:**
  None

---

3.6.4  A822_OutputWord

- **Description:**
  This subroutine will send the 16-bit data to the desired I/O port.

- **Syntax:**
  ```c
  void A822_OutputWord(WORD wPortAddr, WORD wOutputVal);
  ```

- **Parameter:**
  - `wPortAddr`: [In] I/O port address, for example, 0x220
  - `wOutputVal`: [In] 16-bit data send to I/O port

- **Return:**
  None
3.6.5 A822_InputByte

- **Description:**
  This subroutine will input the 8-bit data from the desired I/O port.

- **Syntax:**
  ```c
  WORD A822_InputByte(WORD wPortAddr);
  ```

- **Parameter:**
  ```
  wPortAddr : [In] I/O port address, for example, 0x220
  ```

- **Return:**
  16-bit data with the leading 8 bits all set to 0.

3.6.6 A822_InputWord

- **Description:**
  This subroutine will input the 16-bit data from the desired I/O port.

- **Syntax:**
  ```c
  WORD DIO_InputWord(WORD wPortAddr);
  ```

- **Parameter:**
  ```
  wPortAddr : [In] I/O port address, for example, 0x220
  ```

- **Return:**
  16-bit data.
3.7 AD FUNCTIONS

3.7.1 A822_SetChGain

- **Description:**
  This subroutine will set the multiplexer to the specified channel, configuration-code and delays for the settling time. Users has to call this function before calling A822_Fast_AD_Hex() and/or A822_Fast_AD_Float() functions.

- **Syntax:**
  
  ```c
  WORD A822_SetChGain( WORD wBase,  WORD wChannel,  
                       WORD wConfig,  WORD wCardType );
  ```

- **Parameter:**
  
  - `wBase`: [In] I/O port base address, for example, 0x220
  - `wChannel`: [In] A/D channel number,
  - `wConfig`: [In] Configuration code,
  - `wCardType`: [In] 0 ➔ OME-A-822PGL, 1 ➔ OME-A-822PGH

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.7.2 A822_Hex2Float

- **Description:**
  Compute the Hex(WORD) to floating value.

- **Syntax:**
  
  ```c
  WORD A822_Hex2Float(WORD wConfig, WORD wCardType,  
                       WORD wHex,      float *fVal);
  ```

- **Parameter:**
  
  - `wConfig`: [In] Refer to Section "1.2 Range Configuration".
  - `wCardType`: [In] 0 ➔ OME-A-822PGL, 1 ➔ OME-A-822PGH
  - `wHex`: [In] The Hex(WORD) value input.
  - `fVal`: [Out] Return the value in floating format.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.7.3 A822_Fast_AD_Hex

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for A822PGH/L.
  Users have to call the A822_SetChGain() function before calling this function. In fact,
  \[ A822_AD_Hex() = A822_SetChGain() + A822_Fast_AD_Hex(). \]

- **Syntax:**
  ```
  WORD A822_Fast_AD_Hex(WORD* wVal);
  ```

- **Parameter:**
  - `wVal` : [Out] Returns the A/D value in WORD format.

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.7.4 A822_Fast_AD_Float

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for A822PGH/L. This subroutine will compute the result according to the **configuration code**.
  Users have to call the A822_SetChGain() function before calling this function. In fact,
  \[ A822_AD_Float() = A822_SetChGain() + A822_Fast_AD_Float(). \]

- **Syntax:**
  ```
  WORD A822_Fast_AD_Float(float* fVal);
  ```

- **Parameter:**
  - `fVal` : [Out] Returns the A/D value in floating format.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.7.5 A822_AD_Hex

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for A822PGH/L.

- **Syntax:**
  
  ```
  WORD A822_AD_Hex(WORD wBase, WORD wChannel, WORD wConfig,
  WORD wCardType, WORD* wVal);
  ```

- **Parameter:**
  
  - `wBase` : `[In]` I/O port base address, for example, 0x220
  - `wChannel` : `[In]` A/D channel number,
  - `wConfig` : `[In]` Configuration code,
    
    Refer to Section 1.2 for detailed information
  - `wCardType` : `[In]` 0 ➔ OME-A-822PGL, 1 ➔ OME-A-822PGH
  - `wVal` : `[Out]` Returns the A/D value in WORD format.

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.7.6 A822_AD_Float

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for A822PGH/L. This subroutine will compute the result according to the configuration code.

- **Syntax:**
  
  ```
  WORD A822_AD_Float(WORD wBase, WORD wChannel, WORD wConfig,
  WORD wCardType, float* fVal);
  ```

- **Parameter:**
  
  - `wBase` : `[In]` I/O port base address, for example, 0x220
  - `wChannel` : `[In]` A/D channel number,
  - `wConfig` : `[In]` Configuration code,
    
    Refer to Section 1.2 for detailed information
  - `wCardType` : `[In]` 0 ➔ OME-A-822PGL, 1 ➔ OME-A-822PGH
  - `fVal` : `[Out]` Returns the A/D value in floating format.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.7.7 A822_ADs_Hex

- **Description:**
  This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A822_AD except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D conversion happens at the ISA bus’s max speed. The sampling rate is about 90Ksamples/second tested under Pentium-133 CPU. After A/D conversion, the A/D data are stored in a buffer in Hex format. The `wBuf` is the starting address of this data buffer.

- **Syntax:**
  ```c
  WORD A822_ADs_Hex(WORD wBase, WORD wChannel, WORD wConfig, 
                   WORD wType, WORD wBuf[], WORD wCount);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wChannel` : [In] A/D channel number
  - `wConfig` : [In] Configuration code,
    - Refer to Section 1.2 for detailed information
  - `wType` : [In] 0 ➔ OME-A-822PGL,   1 ➔ OME-A-822PGH
  - `wBuf` : [Out] Data buffer stores the AD value (In WORD format)
    - Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
  - `wCount` : [In] Number of A/D conversions that will be performed

- **Return:**
  Refer to "Section 3.1 Error Codes".
### 3.7.8 A822_ADs_Float

- **Description:**
  This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A822_AD except that this subroutine will perform `wCount` of conversions instead of just one conversion. The A/D conversion happens at the ISA bus’s max speed. The sampling rate is about 90K samples/second tested under Pentium-133 CPU. Then the A/D data are stored in a data buffer in Float format. The **`fBuf`** is the starting address of this data buffer.

- **Syntax:**
  ```c
  WORD A822_ADs_Float(WORD wBase, WORD wChannel, WORD wConfig, WORD wType, float fBuf[], WORD wCount);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wChannel` : [In] A/D channel number
  - `wConfig` : [In] Configuration codes, refer to 1.2 for detail information
  - `wType` : [In] 0 ➔ OME-A-822PGL, 1 ➔ OME-A-822PGH
  - `fBuf` : [Out] Data buffer stores the AD value (In float format)
    Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
  - `wCount` : [In] Number of A/D conversions that will be performed

- **Return:**
  Refer to "Section 3.1 Error Codes".
### 3.8 DA FUNCTIONS

#### 3.8.1 A822_DA

- **Description:**
  This subroutine will send the 12-bit data to D/A analog output. The output range of D/A maybe 0-5V or 0-10V **set by the hardware jumper, JP1.** The software cannot detect the output range of D/A converter. **For examples, if hardware is selected for -5V, the 0xfff will send out 5V. If hardware is selected for -10V, the 0xfff will send out 10V. The factory setting is selected for 0 - 5V D/A output range.**

- **Syntax:**
  ```c
  void A822_DA(WORD wBase, WORD wChannel, WORD wHexValue);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wChannel` : [In] D/A channels number, valid range is 0 to 1
  - `wHexValue` : [In] 12-bit data send to D/A converter

- **Return:**
  None
3.8.2 A822_Uni5_DA

- **Description:**
  This subroutine will send the 12-bit data to D/A analog output. The output range of D/A is **set by the hardware jumper, JP1 ( -5V or –10V) , JP10/JP11 (Bipolar or Unipolar).** The software **cannot detect** the output range of D/A converter. This subroutine can be used only when the jumper's settings are: **Unipolar, -5V.** The **output range is between 0.0V and 5.0V.** Please refer to the hardware manual for setting the jumpers.

- **Syntax:**
  ```
  void A822_Uni5_DA(WORD wBase, WORD wChannel, float fValue);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wChannel` : [In] D/A channels number, valid range is 0 to 1
  - `fValue` : [In] 12-bit data send to D/A converter

- **Return:**
  None

3.8.3 A822_Uni10_DA

- **Description:**
  This subroutine will send the 12-bit data to D/A analog output. The output range of D/A is **set by the hardware jumper, JP1 ( -5V or –10V) , JP10/JP11 (Bipolar or Unipolar).** The software **cannot detect** the output range of D/A converter. This subroutine can be used only when the jumper's settings are: **Unipolar, -10V.** The **output range is 0.0V to 10.0V.** Please refer to the hardware manual for setting the jumpers.

- **Syntax:**
  ```
  void A822_Uni10_DA(WORD wBase, WORD wChannel, float fValue);
  ```

- **Parameter:**
  - `wBase` : [In] I/O port base address, for example, 0x220
  - `wChannel` : [In] D/A channels number, valid range is 0 to 1
  - `fValue` : [In] floating value send to D/A converter

- **Return:**
  None
3.9 AD WITH INTERRUPT

3.9.1 A822_IntInstall

- **Description:**
  This subroutine will install interrupt handler for a specific IRQ level and allocate the data buffer in the device driver as required. For more detailed information of using interrupt please refer to *Section 3.9.8 Architecture of Interrupt Mode*.

- **Syntax:**
  ```c
  WORD A822_IntInstall(WORD wBase,  WORD wIrq, HANDLE *hEvent,DWORD dwCount );
  ```

- **Parameter:**
  - wBase : [In] the I/O port base address for A822 card.
  - wIrq : [In] the IRQ level.
  - hEvent : [In] a pointer point to a event-object that created by user.
  - dwCount : [In] the desired A/D entries count for interrupt transfer.

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.9.2 A822_IntGetCount

- **Description:**
  This subroutine will read the transferred count of interrupt.

- **Syntax:**
  ```c
  WORD A822_IntGetCount(DWORD *dwVal );
  ```

- **Parameter:**
  - dwVal : [Out] return the counter-value of the interrupt transferred.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.9.3  A822_IntStart

- **Description:**
  This subroutine will start the interrupt transfer for a specific A/D channel, program the gain code and sampling rate.

- **Syntax:**
  WORD A822_IntStart(WORD wCardType, WORD wChannel, WORD wGain, WORD c1, Word c2)

- **Parameter:**
  - wCardType : [In] 0: for A822PGL, 1: for A822PGH
  - wChannel : [In] the A/D channel. Valid range is 0 to 15.
  - wGain : [In] the Gain-Code. Please refer to Section 1.2.
  - c1,c2 : [In] the sampling rate is 2M/(c1*c2)
    - c1 ➔ Counter1, c2 ➔ Counter2
    - These values will be used only when the Trigger-Mode setting to Internal-Trigger. Please refer to the function "A822_SetTriggerMode".

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.9.4 A822_IntGetHexBuf

- **Description:**
  This subroutine will copy the transferred interrupted data into the user's buffer.

- **Syntax:**
  
  ```
  WORD A822_IntGetHexBuf(DWORD dwNum, WORD wBuffer[])
  ```

- **Parameter:**
  
  - `dwNum` : [In] data number to be copied.
  - `wBuffer` : [Out] the address of `wBuffer` (in WORD format).

  Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.9.5 A822_IntGetFloatBuf

- **Description:**
  This subroutine will copy the transferred interrupted data into the user's buffer.

- **Syntax:**
  
  ```
  WORD A822_IntGetFloatBuf(DWORD dwNum, float fBuffer[])
  ```

- **Parameter:**
  
  - `dwNum` : [In] data number to be copied
  - `fBuffer` : [Out] the address of `fBuffer` (in float format).

  Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.9.6 A822_IntStop

- **Description:**
  This subroutine will stop the interrupt transfer.

- **Syntax:**
  ```c
  WORD A822_IntStop(void )
  ```

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.9.7 A822_IntRemove

- **Description:**
  This subroutine will remove the installed interrupt handler and free
  the data buffer in the device driver. Thus, users have to get the data
  before the data buffer is freed.

- **Syntax:**
  ```c
  WORD A822_IntRemove(void )
  ```

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.9.8 Architecture of Interrupt mode

The 3.9.1 to 3.9.7 covers the functions to perform the A/D conversion with interrupt transfer. The flow chart to program these functions is given as follows:

- **A822_DriverInit()**: Initialize Device-Driver, Allocate resource for device
- **CreateEvent()**: Create event-object by Win32 API function
- **A822_IntInstall()**: Install the ISR and allocate the data buffer
- **A822_IntStart()**: Start the Interrupt Transfer
- **WaitForSingleObject()**: Calling the Win32 API to wait for the event-object be signaled by the device driver when the Interrupt transfer is completed.
- **A822_IntStop()**: Stop the Interrupt Transfer
- **ResetEvent()**: Reset the event-object to non-signaled status by Win32 API manually.
- **A822_IntGetHexBuf()** or **A822_IntGetFloatBuf()**: Copy data from driver into user's buffer.
- **A822_IntRemove()**: Remove the ISR and free the data buffer.
- **CloseHandle()**: Calling Win32 API to free the event-object's reference.
- **A822_DriverClose()**: Close the Device Driver and free the resource.
3.10 AD DMA FUNCTION

These DMA functions are supported on Windows 95/98 only.

3.10.1 A822_AD_DMA_InstallIrq

- **Description:**
  This subroutine will install interrupt handler for a specific IRQ Level n and program a DMA controller(8227) to handle DMA transfer for DMA Channel n. Usually, when a DMA transfer finished, a associated IRQ level n occur. For more detail information for using DMA, please refer to "Section 3.10.8 Architecture of DMA mode".

- **Syntax:**
  ```c
  WORD A822_AD_DMA_InstallIrq
    (WORD wBase,  WORD wIrq,  WORD wDmaChannel );
  ```

- **Parameter:**
  - `wBase`: [In] the I/O port base address for A822 card.
  - `wIrq`: [In] the IRQ level n.
  - `wDmaChannel`: [In] the DMA channel ( 1 or 3 ).

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.10.2 A822_AD_DMA_IsNotFinished

- **Description:**
  This subroutine is to detect if the DMA have finished.

- **Syntax:**
  ```c
  WORD A822_AD_DMA_IsNotFinished(void )
  ```

- **Parameter:**
  None

- **Return:**
  - 0: the DMA transfer is finish.
  - 1: the DMA transfer is proceeding.
3.10.3 A822_AD_DMA_Start

- **Description:**
  This subroutine will allocate a DMA buffer in the system area, program the gain code and sampling rate. Then it starts the DMA transfer for a specific A/D channel.

- **Syntax:**
  ```
  WORD A822_AD_DMA_Start(WORD wCardType, WORD Ch, WORD Gain, 
  WORD c1, Word c2, DWORD Count, WORD wPassOut[])
  ```

- **Parameter:**
  - `wCardType` : [In] 0: A822PGL 1: A822PGH
  - `Ch` : [In] the A/D channel. Valid range is 0 to 15.
  - `Gain` : [In] the Gain code. Please refer to Section 1.2.
  - `c1,c2` : [In] the DMA sampling rate is `2M/(c1*c2)`
    - `c1` ➔ Counter1, `c2` ➔ Counter2
  - `Count` : [In] the desired A/D entries count for DMA transfer.
  - `wPassOut[]` : [Out] Debug information, users have to allocate space for it.
    - `wPassOut[0]` : [Out] 0 : successful in starting DMA transfer.
    - Others: fail in starting DMA transfer.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.10.4  A822_AD_DMA_GetBuffer

- **Description:**
  This subroutine will copy the transferred DMA data into the user's buffer.

- **Syntax:**
  WORD A822_AD_DMA_GetBuffer( WORD wBuffer[] )

- **Parameter:**
  - wBuffer : [Out] the address of wBuffer(In WORD format).
  - Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.

- **Return:**
  Refer to Section "3.1 Error Codes".

3.10.5  A822_AD_DMA_GetFloatBuffer

- **Description:**
  This subroutine will copy the transferred DMA data into the user's buffer.

- **Syntax:**
  WORD A822_AD_DMA_GetFloatBuffer( float fBuffer[] )

- **Parameter:**
  - fBuffer : [Out] the address of fBuffer(In float format).
  - Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.10.6  A822_AD_DMA_Stop

- **Description:**
  This subroutine will free the allocated DMA buffer that is in system area.

- **Syntax:**
  WORD A822_AD_DMA_Stop(void )

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.10.7  A822_AD_DMA_RemoveIrq

- **Description:**
  This subroutine will remove the interrupt handler installed by A822_AD_DMA_InstallIrq(…).

- **Syntax:**
  WORD A822_AD_DMA_RemoveIrq(void )

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.10.8 Architecture of DMA mode

The 3.10.1 to 3.10.7 covers the functions to perform the A/D conversion with DMA transfer. The flow chart to program these functions is given as follows:

- **A822_AD_DMA_Init()**: Initialize Device-Driver, Allocate resource for device.
- **A822_AD_DMA_GetBuffer()** or **A822_AD_DMA_GetFloatBuffer()**: Copy data from driver into user's buffer.
- **A822_AD_DMA_Start()**: Start the DMA + Interrupt Transfer and allocate data buffer.
- **A822_AD_DMA_IsNotFinished()**: Check if the DMA + Interrupt transfer is completed?
- **A822_AD_DMA_RemoveReq()** or **A822_AD_DMA_RemoveFloatBuffer()**: Remove the ISR and free the data buffer.
- **A822_AD_DMA_RemoveIrq()**: Install the ISR
- **A822_DriverClose()**: Close the Device Driver and free the resource.
3.11 AD WITH CHANNEL SCAN

3.11.1 Introduction

The user can specify channels into a list buffer. Other functions will do the ADC to get the data. And then read the list buffer to change to next channel and set to specific configuration code.

The data will be saved into the following style:

```
List buffer for Channel Scan: CA, CB, CC
User's buffer: CAV, CAV, CAV, ...... CBV, CBV, CBV, ...... CCV, CCV, CCV....
```

Note:
CA = Channel A; CB = Channel B; CC = Channel C
CAV = Channel A’s value; CBV = Channel B’s value;
CCV = Channel C’s value

The user program’s architecture will be as following:

```
A822_ChScan_Clear: Clear the list buffer for Channel Scan
A822_ChScan_Add: Add channels to the list buffer for Channel Scan
A822_ChScan_PollingFloat: Do the Channel Scan to get data
```
### 3.11.2 A822_ChScan_Clear

- **Description:**
  
  This subroutine will clear the list buffer for the Channel Scan.

- **Syntax:**
  
  ```c
  void A822_ChScan_Clear(void);
  ```

- **Parameter:**
  
  None

- **Return:**
  
  None

### 3.11.3 A822_ChScan_Add

- **Description:**
  
  This function will add the specified channel number and configuration-code into the list buffer for the Channel Scan. The max number of channels in the list buffer for the Channel Scan is 100.

- **Syntax:**
  
  ```c
  WORD A822_ChScan_Add(WORD wChannel, WORD wConfig);
  ```

- **Parameter:**
  
  - `wChannel` : **[In]** The channel to be scanned.
  - `WConfig` : **[In]** Specify the configuration-code for this channel.
    
    Please refer to "Section 1.2 Range Configuration".

- **Return:**
  
  Refer to "Section 3.1 Error Codes".
3.11.4 A822_ChScan_Set

- **Description:**
  This function will clear the list buffer and then copy the specified list of channel(s) and configuration-code(s) into the list buffer for the Channel Scan. The max number of channels in the list buffer for the Channel Scan is 100.

- **Syntax:**
  ```c
  WORD A822_ChScan_Set
  (WORD wChannel[], WORD wConfig[], WORD wChNum);
  ```

- **Parameter:**
  - `wChannel` : [In] The list of channel(s) to be scanned.
  - `wConfig` : [In] The list of configuration-code(s) for channel(s).
    
      Please refer to Section 1.2.
  - `wChNum` : [In] Total channels to be scanned.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.11.5 A822_ChScan_PollingHex

- **Description:**
  This subroutine will perform a number of A/D conversions by polling. And after get the channel’s data, it will then read the list buffer for the Channel Scan to change to next channel and set to specified configuration code. The A/D conversion happens at the ISA bus’s max speed. After A/D conversion, the A/D data are stored in a buffer in Hex format.

  Before calling this function, the user has to call the A822_ChScan_Clear() and A822_ChScan_Add() or A822_ChScan_Set() functions to setup the list buffer for Channel Scan. Please refer to the “Section 3.11.1 Introduction” for more information.

- **Syntax:**
  ```c
  WORD A822_ChScan_PollingHex(WORD wBase, WORD wCardType, 
  WORD wBuf[], WORD wNumPerCh);
  ```

- **Parameter:**
  - `wBase` : [In] the I/O port base address for A822 card.
  - `wCardType` : [In] 0: A-822L 1: A-822H
  - `wBuf` : [Out] Starting address of the data buffer (WORD format)
    Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
    
    The buffer size
    \[
    \text{Total-Channels} \times \text{wNumPerCh} \times \text{sizeof(WORD)}
    \]
  - `wNumPerCh` : [In] Number of A/D conversions will be performed for every channel.

- **Return:**
  Refer to "Section 3.1 Error Codes".
### 3.11.6 A822_ChScan_PollingFloat

- **Description:**
  
  This subroutine will perform a number of A/D conversions by polling. After getting the channel’s data, it will then read the list buffer for the Channel Scan to change to next channel and set to specified configuration code. The A/D conversion happens at the ISA bus’s max speed. After A/D conversion, the A/D data is stored in a buffer in floating format.
  
  Before calling this function, the user has to call the A822_ChScan_Clear() and A822_ChScan_Add() or A822_ChScan_Set() functions to setup the list buffer for Channel Scan. Please refer to the “Section 3.11.1 Introduction” for more information.

- **Syntax:**

  ```c
  WORD A822_ChScan_PollingFloat(WORD wBase, WORD wCardType, float fBuf[], WORD wNumPerCh);
  ```

- **Parameter:**

  - **wBase** : [In] the I/O port base address for A822 card.
  - **wCardType** : [In] 0: A-822L 1: A-822H
  - **fBuf** : [Out] Starting address of the data buffer (floating format)
    
    Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
    
    The buffer size
    
    \[
    \text{buffer size} = \text{Total-Channels} \times \text{wNumPerCh} \times \text{sizeof(float)}
    \]

  - **wNumPerCh** : [In] Number of A/D conversions will be performed for every channel.

- **Return:**

  Refer to "Section 3.1 Error Codes".
3.12 AD INTERRUPT, CHANNEL SCAN FUNCTION

3.12.1 Introduction

The user can specify channels into a list buffer. The other functions will do the ADC to get the data. Then read the list buffer to change to next channel and set to specify configuration code.

List buffer for Channel Scan: CA, CB, CC

User's buffer: CAV, CAV, CAV, … CBV, CBV, CBV, … CCV, CCV, CCV, …

The data will be saved into the following style:

Note:
CA = Channel A; CB = Channel B; CC = Channel C
CAV = Channel A’s value; CBV = Channel B’s value;
CCV = Channel C’s value

After setting to the next channel and specified configuration code, it has to delay for the settling time before next ADC. The interrupt service routine doesn’t delay allowing for the settling time. Thus, to get the correct ADC data, the user has to slow-down the sampling-rate of interrupt.

The sampling-rate is for all channels.

For example:

The list buffer for the Channel Scan is set to channel-2 and channel-0. The sampling-rate is setting to 10 KHz. In fact, the channel-2 has the sampling-rate of 5 KHz and the channel-0 also has the sampling-rate 5KHz.
The user program's architecture as following:

- **A822_DriverInit()**: Initialize the Device-Driver and allocate resource.
- **A822_ChScan_Clear()**: Clear the list buffer for Channel Scan.
- **A822_ChScan_Add()**: Add channels to the list buffer for Channel Scan.
- **CreateEvent()**: Calling Win32 API to create the Event object.
- **A822_ChScan_IntInstall()**: Install the ISR, list buffer for Channel Scan and allocate data buffer for all channels.
- **A822_ChScan_IntStart()**: Starting the Interrupt Transfer.
- **WaitForSingleObject()**: Calling Win32 API to wait for the Event object be signaled by device driver.
- **A822_ChScan_IntStop()**: Stop the interrupt transfer.
- **ResetEvent()**: Calling Win32 API to reset the event-object to non-signaled status manually.
- **A822_ChScan_IntGetFloatBuf()**: Copy data into user's buffer.
- **A822_ChScan_IntRemove()**: Remove the interrupt handler and free allocated buffers.
- **CloseHandle()**: Free the Event-object's reference.
- **A822_DriverClose()**: Close the Device-Driver.
3.12.2  A822_ChScan_IntInstall

- **Description:**
  This subroutine will install interrupt handler, copy the list buffer for Channel Scan into kernel-mode driver and allocate buffers for every channels. Before installing the interrupt, the user has to call the "A822_ChScan_Clear()" and "A822_ChScan_Add()" or "A822_ChScan_Set() functions to setup the list buffer for Channel Scan. For more detail information of using interrupt please refer to "Section 3.12.1 Introduction".

- **Syntax:**
  ```
  WORD A822_ChScan_IntInstall(WORD wBase, WORD wIrq, HANDLE *hEvent, DWORD dwNumPerCh);
  ```

- **Parameter:**
  - wBase  : [In] the I/O port base address for A822 card.
  - wIrq   : [In] the IRQ level n.
  - hEvent : [In] The Event handle that was created by the user.
  - dwNumPerCh : [In] The desired A/D count for every channels to transfer.

- **Return:**
  Refer to "Section 3.1 Error Codes".

---

3.12.3  A822_ChScan_IntStart

- **Description:**
  This subroutine will clear the interrupt-counter and start the interrupt transfer for the specific A/D channels, program the gain code and sampling rate.

- **Syntax:**
  ```
  WORD A822_ChScan_IntStart(WORD c1, WORD c2, WORD wCardType);
  ```

- **Parameter:**
  - c1,c2  : [in] the sampling rate is 2M/(c1*c2); c1=Counter1, c2=Counter2
  - wCardType  : [In] 0: A-822L       1: A-822H

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.12.4  A822_ChScan_IntStop

- **Description:**
  This subroutine will stop the interrupt transfer.

- **Syntax:**
  WORD A822_ChScan_IntStop(void);

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.12.5  A822_ChScan_IntRemove

- **Description:**
  This subroutine will remove the interrupt handler and free the buffers.

- **Syntax:**
  WORD A822_ChScan_IntRemove(void);

- **Parameter:**
  None

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.12.6  A822_ChScan_IntGetCount

- **Description:**
  This subroutine will read the transferred count of interrupt.

- **Syntax:**
  WORD A822_Int_GetCount(DWORD *dwVal)

- **Parameter:**
  dwVal : [Out] Returns the interrupt transferred count.

- **Return:**
  Refer to "Section 3.1 Error Codes".
3.12.7  A822_ChScan_IntGetHexBuf

- **Description:**
  This subroutine will copy the transferred interrupted data into the user’s buffer.

- **Syntax:**
  WORD A822_ChScan_IntGetHexBuf(WORD wBuf[])

- **Parameter:**
  wBuf : [Out] The address of wBuf(WORD format).
  Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
  Buffer size = Total-Channels * dwNumPerCh * sizeof(WORD)

- **Return:**
  Refer to "Section 3.1 Error Codes".

3.12.8  A822_ChScan_IntGetFloatBuf

- **Description:**
  This subroutine will copy the transferred interrupted data into the user’s buffer.

- **Syntax:**
  WORD A822_ChScan_IntGetFloatBuf(float fBuf[])

- **Parameter:**
  fBuf : [Out] The address of fBuf(float format).
  Users have to allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. Users can analyze this data from the buffer after calling this function.
  Buffer size = Total-Channels * dwNumPerCh * sizeof(float)

- **Return:**
  Refer to "Section 3.1 Error Codes".
4. PROGRAM ARCHITECTURE

- Initialize the Device-Driver
- Access/Control the Device
  - A822_DriverInit()
  - A822_InputByte( … )
  - A822_OutputByte(…)
- Close the Device-Driver

User's Application
  - Function Call into DLLs
  - DLLs
  - Services Call into Kernel-Mode
  - .VXDs, .SYSs (Device Driver)
  - Device Control
- Hardware Devices

Development Toolkit

Ver: 3.2     Date: 11/27/00     Page: 94
5. REPORT PROBLEMS

Technical support is available at no charge as described below. The best way to report problems is send electronic mail to das@omega.com on the Internet.

When reporting problems, please include the following information:

1) Is the problem reproducible? If so, how?

2) What kind and version of Operation Systems that you running? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.

3) What kinds of our products that you using? Please see the product's manual.

4) If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.

5) If the problem involves other programs or hardware devices, what devices or version of the failing programs that you using?

6) Other comments relative to this problem or any Suggestions will be welcomed.

After we received your comments, we will take about two business days to testing the problems that you said. And then reply as soon as possible to you. Please check that we have received your comments? And please keeping contact with us.

E-mail: das@omega.com
Web-Site: http://www.omega.com
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