User's Guide

OME-A822PG ISA-BUS
Multi-Functional Board
Windows 2000 Software Manual

omega.com®
OMEGA®

OMEGAnet® Online Service www.omega.com	Internet e-mail info@omega.com

Servicing North America:

USA:
ISO 9001 Certified
One Omega Drive, P.O. Box 4047
Stamford CT 06907-0047
TEL: (203) 359-1660 FAX: (203) 359-7700
e-mail: info@omega.com

Canada:
976 Bergar
Laval (Quebec) H7L 5A1, Canada
TEL: (514) 856-6928 FAX: (514) 856-6886
e-mail: info@omega.ca

For immediate technical or application assistance:

USA and Canada: Sales Service: 1-800-826-6342 / 1-800-TC-OMEGA®
Customer Service: 1-800-622-2378 / 1-800-622-BEST®
Engineering Service: 1-800-872-9436 / 1-800-USA-WHEN®
TELEX: 996404 EASYLINK: 62968934 CABLE: OMEGA

Mexico: En Español: (001) 203-359-7803 e-mail: espanol@omega.com
FAX: (001) 203-359-7807 info@omega.com.mx

Servicing Europe:

Benelux:
Postbus 8034, 1180 LA Amstelveen, The Netherlands
TEL: +31 (0)20 3472121 FAX: +31 (0)20 6434643
Toll Free in Benelux: 0800 0993344
e-mail: sales@omegaeng.nl

Czech Republic:
Frystatska 184, 733 01 Karviná, Czech Republic
TEL: +420 (0)59 6311899 FAX: +420 (0)59 6311114
Toll Free: 0800-1-66342 e-mail: info@omegashop.cz

France:
11, rue Jacques Cartier, 78280 Guyancourt, France
TEL: +33 (0)1 61 37 29 00 FAX: +33 (0)1 30 57 54 27
Toll Free in France: 0800 466 342
e-mail: sales@omega.fr

Germany/Austria:
Daimlerstrasse 26, D-75392 Deckenpfronn, Germany
TEL: +49 (0)7056 9398-0 FAX: +49 (0)7056 9398-29
Toll Free in Germany: 0800 639 7678
e-mail: info@omega.de

United Kingdom:
ISO 9002 Certified
One Omega Drive, River Bend Technology Centre
Northbank, Irlam, Manchester
M44 5BD United Kingdom
TEL: +44 (0)161 777 6611 FAX: +44 (0)161 777 6622
Toll Free in United Kingdom: 0800-488-488
e-mail: sales@omega.co.uk

It is the policy of OMEGA to comply with all worldwide safety and EMC/EMI regulations that apply. OMEGA is constantly pursuing certification of its products to the European New Approach Directives. OMEGA will add the CE mark to every appropriate device upon certification.

OME-A-822PGL/H

Software Manual

[For Windows 2000]

Table of Contents

1. DECLARATION FILES

Please refer to user manual "CallDLL.pdf".

For Windows 2000:

```
|--\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
```

1.1 A822.H

```
#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_InstallBufError        10
#define   A822_AllocateMemoryError    11
#define   A822_CardTypeError          12
#define   A822_TimeoutError           13
#define   A822_OtherError             14
#define   A822_ExceedBoardNumber      15
#define   A822_CardNotFound           16
#define   A822_GetTotalBoardError     17
#define   A822_ChannelNoError         18
#define   A822_IntStopError           19
#define   A822_IntInstallEventError   20
#define   A822_GetConfigError         21
#define   A822_ActiveBoardError       22
#define   A822_ConfigCodeError        23
#define   A822_BufferFull             24
#define   A822_NoChannelToScan        25
#define   A822_IntInstallChannelError 26
#define   A822_IntInstallConfigError  27


// Functions 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);

// Functions of DI/DO
EXPORTS WORD  CALLBACK A822_DI(WORD *wInVal);
EXPORTS WORD  CALLBACK A822_DO(WORD wHexValue);
```

```
// Functions of AD/DA
EXPORTS WORD  CALLBACK A822_SetChGain
      (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 wChannel, WORD wConfig, WORD wCardType, WORD *wVal);
EXPORTS WORD  CALLBACK A822_AD_Float
      (WORD wChannel, WORD wConfig, WORD wCardType, float *fVal);
EXPORTS WORD  CALLBACK A822_ADs_Hex( WORD wBuf[], WORD
wCount );
EXPORTS WORD  CALLBACK A822_ADs_Float( float fBuf[], WORD wCount );
EXPORTS WORD  CALLBACK A822_Hex2Float
      (WORD wConfig, WORD wCardType, WORD wHex, float *fVal);

EXPORTS WORD  CALLBACK A822_DA_Hex(WORD wChannel, WORD
wHexValue);
EXPORTS WORD  CALLBACK A822_DA_Uni5(WORD wChannel, float fValue);
EXPORTS WORD  CALLBACK A822_DA_Uni10(WORD wChannel, float fValue);

// Functions of Driver
EXPORTS WORD  CALLBACK A822_DriverInit(WORD *wTotalBoards);
EXPORTS void  CALLBACK A822_DriverClose(void);
EXPORTS WORD  CALLBACK A822_DELAY(WORD wDownCount);
EXPORTS WORD  CALLBACK A822_Check_Address(void);
EXPORTS WORD  CALLBACK A822_GetConfigAddress
      (WORD *wAddrBase, WORD *wCurrentBoard);
EXPORTS WORD  CALLBACK A822_ActiveBoard( WORD wBoardNo );
EXPORTS void  CALLBACK A822_SetTriggerMode(WORD wTriggerMode );

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);
```

```
// Functions of Interrupt
EXPORTS WORD  CALLBACK A822_Int_Install
      (HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD  CALLBACK A822_Int_Start(WORD c1, WORD c2);
EXPORTS WORD  CALLBACK A822_Int_Stop(void);
EXPORTS WORD  CALLBACK A822_Int_Remove(void);
EXPORTS WORD  CALLBACK A822_Int_GetCount(DWORD *dwVal);
EXPORTS WORD  CALLBACK A822_Int_GetHexBuf
      (WORD wBuf[], DWORD dwNum );
EXPORTS WORD  CALLBACK A822_Int_GetFloatBuf
      (float fBuf[], DWORD dwNum );

// Functions of Channel Scan
EXPORTS void  CALLBACK A822_ChScan_Clear(void);
EXPORTS WORD  CALLBACK A822_ChScan_Add
      (WORD wChannel, WORD wConfig);
EXPORTS WORD  CALLBACK A822_ChScan_PollingHex
      (WORD wCardType, WORD wBuf[], WORD wNumPerCh);
EXPORTS WORD  CALLBACK A822_ChScan_PollingFloat
      (WORD wCardType, float fBuf[], WORD wNumPerCh);

// Functions of Channel Scan for Interrupt Only
EXPORTS WORD  CALLBACK A822_ChScan_IntInstall
      (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 fBuf[]);
EXPORTS WORD  CALLBACK A822_ChScan_IntStop(void);
EXPORTS WORD  CALLBACK A822_ChScan_IntRemove(void);
```

1.2 A822.BAS

```
Attribute VB_Name = "A822"
'********************************************************************************
'      The Declare of A822.DLL for A822 DAQ Card
'********************************************************************************

Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)

'***************** 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_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_ExceedBoardNumber      = 15
Global Const A822_CardNotFound           = 16
Global Const A822_GetTotalBoardError     = 17
Global Const A822_ChannelNoError         = 18
Global Const A822_IntStopError           = 19
Global Const A822_IntInstallEventError   = 20
Global Const A822_GetConfigError         = 21
Global Const A822_ActiveBoardError       = 22
Global Const A822_ConfigCodeError        = 23
Global Const A822_BufferFull             = 24
Global Const A822_NoChannelToScan        = 25
Global Const A822_IntInstallChannelError = 26
Global Const A822_IntInstallConfigError  = 27

'******* 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

'******* DI/DO Functions ***********
Declare Function A822_DI Lib "A822.DLL" _
      (wInVal As Integer) As Integer
Declare Function A822_DO Lib "A822.DLL" _
      (ByVal wHexValue As Integer) As Integer
```

```
'******* AD/DA Functions ***********
Declare Function A822_SetChGain Lib "A822.DLL" _
      (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 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 wChannel As Integer, ByVal wConfig As Integer, _
      ByVal wCardType As Integer, fVal As Single) As Integer
Declare Function A822_ADs_Hex Lib "A822.DLL" _
      (wBuf As Integer, ByVal wCount As Integer) As Integer
Declare Function A822_ADs_Float Lib "A822.DLL" _
      (fbuf As Single, ByVal wCount As Integer) As Integer
Declare Function A822_Hex2Float Lib "A822.DLL" _
      (ByVal wConfig As Integer, ByVal wCardType As Integer, _
      ByVal wVal As Integer, fVal As Single) As Integer

Declare Function A822_DA_Hex Lib "A822.DLL" _
      (ByVal wChannel As Integer, ByVal wHexValue As Integer) As Integer
Declare Function A822_DA_Uni5 Lib "A822.DLL" _
      (ByVal wChannel As Integer, ByVal fValue As Single) As Integer
Declare Function A822_DA_Uni10 Lib "A822.DLL" _
      (ByVal wChannel As Integer, ByVal fValue As Single) As Integer


'******* Driver Functions ***********
Declare Function A822_DriverInit Lib "A822.DLL" _
      (wTotalBoards As Integer) As Integer
Declare Sub A822_DriverClose Lib "A822.DLL" ()
Declare Function A822_DELAY Lib "A822.DLL" _
      (ByVal wDownCount As Integer) As Integer
Declare Function A822_Check_Address Lib "A822.DLL" () As Integer
Declare Function A822_GetConfigAddress Lib "A822.DLL" _
      (wAddrBase As Integer, wCurrentBoard As Integer) As Integer
Declare Function A822_ActiveBoard Lib "A822.DLL" _
      (ByVal wBoardNo As Integer) 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


'******* IRQ Functions ***********
Declare Sub A822_SetTriggerMode Lib "A822.DLL" _
      (ByVal wTriggerMode As Integer)
Declare Function A822_Int_Install Lib "A822.DLL" _
      (hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A822_Int_Start Lib "A822.DLL" _
      (ByVal c1 As Integer, ByVal c2 As Integer) As Integer
Declare Function A822_Int_Stop Lib "A822.DLL" () As Integer
Declare Function A822_Int_Remove Lib "A822.DLL" () As Integer
Declare Function A822_Int_GetCount Lib "A822.DLL" _
      (dwVal As Long) As Integer
Declare Function A822_Int_GetHexBuf Lib "A822.DLL" _
      (wBuffer As Integer, ByVal dwNum As Long) As Integer
Declare Function A822_Int_GetFloatBuf Lib "A822.DLL" _
      (fbuffer As Single, ByVal dwNum As Integer) As Integer

' Functions of Channel Scan
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_PollingHex Lib "A822.DLL" _
      (ByVal wCardType As Integer, wBuf as Integer, _
      ByVal wNumPerCh As Integer) As Integer
Declare Function A822_ChScan_PollingFloat Lib "A822.DLL" _
      (ByVal wCardType As Integer, fBuf as Single, _
      ByVal wNumPerCh As Integer) As Integer
```

```
' Functions of Channel Scan for Interrupt Only
Declare Function A822_ChScan_IntInstall Lib "A822.DLL" _
      (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
```

1.3 A822.PAS

```
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_BI_1          =  0;
 A822_BI_10         =  1;
 A822_BI_100        =  2;
 A822_BI_1000       =  3;
 A822_UNI_1         =  4;
 A822_UNI_10        =  5;
 A822_UNI_100       =  6;
 A822_UNI_1000      =  7;
 A822_BI_05         =  8;
 A822_BI_5          =  9;
 A822_BI_50         =  10;
 A822_BI_500        =  11;

 A822_BI_2          =  1;
 A822_BI_4          =  2;
 A822_BI_8          =  3;
 A822_UNI_2         =  5;
 A822_UNI_4         =  6;
 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_InstallBufError        = 10;
 A822_AllocateMemoryError    = 11;
 A822_CardTypeError          = 12;
 A822_TimeoutError           = 13;
 A822_OtherError             = 14;
 A822_ExceedBoardNumber      = 15;
 A822_CardNotFound           = 16;
 A822_GetTotalBoardError     = 17;
 A822_ChannelNoError         = 18;
 A822_IntStopError           = 19;
 A822_IntInstallEventError   = 20;
 A822_GetConfigError         = 21;
 A822_ActiveBoardError       = 22;
 A822_ConfigCodeError        = 23;
 A822_BufferFull             = 24;
 A822_NoChannelToScan        = 25;
 A822_IntInstallChannelError = 26;
 A822_IntInstallConfigError  = 27;

// 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 DI/DO
Function  A822_DO(wHexValue:Word):Word; StdCall;
Function  A822_DI(var wInVal:Word):Word; StdCall;
```

```
// Function of AD/DA
Function  A822_SetChGain(wChannel,wConfig,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
        (wChannel,wConfig,wCardType:WORD; var wVal:Word):Word; StdCall;
Function  A822_AD_Float
        (wChannel,wConfig,wCardType:WORD; var fVal:Single):Word; StdCall;
Function  A822_ADs_Hex( wBuf:PWord; wCount:WORD):WORD; StdCall;
Function  A822_ADs_Float(fBuf:PSingle; wCount:WORD):WORD; StdCall;
Function  A822_Hex2Float
        ( wConfig, wCardType:Word; wVal:Word; var fVal:Single ):WORD; StdCall;

Function  A822_DA_Hex(wChannel, wHexValue:WORD):WORD; StdCall;
Function  A822_DA_Uni5(wChannel:Word;fValue:Single):WORD; StdCall;
Function  A822_DA_Uni10(wChannel:Word;fValue:Single):WORD; StdCall;

// Function of Driver
Function  A822_DriverInit(var wTotalBoards:WORD):WORD; StdCall;
Procedure A822_DriverClose; StdCall;
Function  A822_DELAY(wDownCount:WORD):WORD; StdCall;
Function  A822_Check_Address:WORD; StdCall;
Function  A822_GetConfigAddress
        (var wAddrBase:WORD; var wCurrentBoard:WORD):WORD; StdCall;
Function  A822_ActiveBoard(wBoardNo: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 Interrupt
Procedure A822_SetTriggerMode( wTriggerMode:WORD ); StdCall;
Function  A822_Int_Install(var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function  A822_Int_Start(c1,c2:WORD):WORD; StdCall;
Function  A822_Int_Stop:WORD; StdCall;
Function  A822_Int_Remove:WORD; StdCall;
Function  A822_Int_GetCount(var dwVal:LongInt):WORD; StdCall;
Function  A822_Int_GetHexBuf(wBuf:PWORD; dwNum:LongInt):WORD; StdCall;
Function  A822_Int_GetFloatBuf(fBuf:PSingle; dwNum:LongInt):WORD; StdCall;
```

```
// Functions of Channel Scan
Procedure A822_ChScan_Clear; StdCall;
Function A822_ChScan_Add
      (wChannel:WORD; wConfig:WORD):WORD; StdCall;
Function A822_ChScan_PollingHex
      (wCardType:WORD; wBuf:PWORD; wNumPerCh:WORD):WORD; StdCall;
Function A822_ChScan_PollingFloat
      (wCardType:WORD; fBuf:PSingle; wNumPerCh:WORD):WORD; StdCall;

// Functions of Channel Scan for Interrupt Only
Function A822_ChScan_IntInstall
      (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  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  A822_DO;                       external 'A822.DLL' name
'A822_DO';
Function  A822_DI;                       external 'A822.DLL' name
'A822_DI';
```

```
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_ADs_Hex;                      external 'A822.DLL' name
'A822_ADs_Hex';
Function  A822_ADs_Float;                    external 'A822.DLL' name
'A822_ADs_Float';
Function  A822_Hex2Float;                    external 'A822.DLL' name
'A822_Hex2Float';

Function  A822_DA_Hex;                       external 'A822.DLL' name
'A822_DA_Hex';
Function  A822_DA_Uni5;                      external 'A822.DLL' name
'A822_DA_Uni5';
Function  A822_DA_Uni10;                     external 'A822.DLL' name
'A822_DA_Uni10';

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';
Function  A822_GetConfigAddress;             external 'A822.DLL' name
'A822_GetConfigAddress';
Function  A822_ActiveBoard;                  external 'A822.DLL' name
'A822_ActiveBoard';
```

```
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';

Procedure A822_SetTriggerMode;            external 'A822.DLL' name
'A822_SetTriggerMode';
Function  A822_Int_Install;               external 'A822.DLL' name
'A822_Int_Install';
Function  A822_Int_Start;                 external 'A822.DLL' name
'A822_Int_Start';
Function  A822_Int_Stop;                  external 'A822.DLL' name
'A822_Int_Stop';
Function  A822_Int_Remove;                external 'A822.DLL' name
'A822_Int_Remove';
Function  A822_Int_GetCount;              external 'A822.DLL' name
'A822_Int_GetCount';
Function  A822_Int_GetHexBuf;             external 'A822.DLL' name
'A822_Int_GetHexBuf';
Function  A822_Int_GetFloatBuf;           external 'A822.DLL' name
'A822_Int_GetFloatBuf';

// Functions of Channel Scan
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_PollingHex;         external 'A822.DLL' name
'A822_ChScan_PollingHex';
Function  A822_ChScan_PollingFloat;       external 'A822.DLL' name
'A822_ChScan_PollingFloat';
```

```
// Functions of Channel Scan for Interrupt Only
Function A822_ChScan_IntInstall;          external 'A822.DLL' name
'A822_ChScan_IntInstall';
Function A822_ChScan_IntStart;            external 'A822.DLL' name
'A822_ChScan_IntStart';
Function A822_ChScan_IntGetCount;         external 'A822.DLL' name
'A822_ChScan_IntGetCount';
Function A822_ChScan_IntGetHexBuf;        external 'A822.DLL' name
'A822_ChScan_IntGetHexBuf';
Function A822_ChScan_IntGetFloatBuf;      external 'A822.DLL' name
'A822_ChScan_IntGetFloatBuf';
Function A822_ChScan_IntStop;             external 'A822.DLL' name
'A822_ChScan_IntStop';
Function A822_ChScan_IntRemove;           external 'A822.DLL' name
'A822_ChScan_IntRemove';

end.
```

2. REFERENCE

2.1 RANGE CONFIGURATION CODE

The AD converter of OME-A822PGH/L is 12-bit 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, use configuration 0/1/2 (for OME-A822PGL) will get nearly the same result **except resolution. So choosing the correct configuration code can achieve the highest precision measurement.**

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

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 2.5V	1
Bipolar	+/- 1.25V	2
Bipolar	+/- 0.0625V	3
Unipolar	0V ~ 10V	4
Unipolar	0V ~ 5V	5
Unipolar	0V ~ 2.5V	6
Unipolar	0V ~ 1.25V	7
Bipolar	+/- 10V	8

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

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5V	0
Bipolar	+/- 0.5V	1
Bipolar	+/- 0.05V	2
Bipolar	+/- 0.005V	3
Unipolar	0 ~ 10V	4
Unipolar	0 ~ 1V	5
Unipolar	0 ~ 0.1V	6
Unipolar	0 ~ 0.01V	7
Bipolar	+/- 10V	8
Bipolar	+/- 1V	9
Bipolar	+/- 0.1V	10
Bipolar	+/- 0.01V	11

2.2 ERROR CODE

Error Code	Number	Description
A822_NoError	0	OK
A822_DriverOpenError	1	Please check if the driver installed correctly.
A822_DriverNoOpen	2	Please call the driver initial function to open the driver first.
A822_GetDriverVersionError	3	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_InstallIrqError	4	1. Please call the driver initial function to open the driver firstly. 2. Please check the resource that does not conflicted with other device.
A822_ClearIntCountError	5	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_GetIntCountError	6	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_GetBufferError	7	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_AllocateMemoryError	11	Can't allocate the memory for buffer. Please check your system's resource.
A822_CardTypeError	12	Valid range is : 0 to 1 (OME-A-822L or OME-A-822H)
A822_TimeoutError	13	ADC timeout error
A822_OtherError	14	Unknown error
A822_ExceedBoardNumber	15	The driver initial function will return number of total boards. The board number to be active must less then this number. For example: if the driver initial function returns 3, and the valid range is 0 to 2.

A822_CardNotFound	16	The card not found by the device driver. Please check your hardware and driver settings.
A822_ChannelNoError	17	The valid AD channel no range is 0 to 15. The valid DA channel no range is 0 to 1.
A822_IntStopError	18	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_IntInstallEventError	19	Can't call the device driver function. Please call the driver initial function to open the driver first.
A822_GetConfigError	20	1. Can't call the device driver function. Please call the driver initial function to open the driver firstly. 2. Call the "ActiveBoard" function first.
A822_ActiveBoardError	21	1. Please call the driver initial function to open the driver first. 2. The driver initial function will return number of total boards. The board number to be active must less then this number.
A822_ConfigCodeError	22	Please refer to "Section 2.1 Range Configuration Code" for the valid configuration code.
A822_BufferFull	23	The buffer for the Channel Scan had full. The max number of buffer is 100 channels to scan. Please use the A822_ChScan_Clear() function to clear the buffer.
A822_NoChannelToScan	24	The user has to setting the channels to scan. Please use the A822_ChScan_Add() function to add channel and configuration-code to buffer for the Channel Scan.
A822_IntInstallChannelError	25	Failed to install the list for the Channel Scan into the interrupt service routine. Please check your system's resource.
A822_IntInstallConfigError	26	Failed to install the list for the Channel Scan into the interrupt service routine. Please check your system's resource.

2.3 OTHER MANUALS

Please refer to the following user manuals:

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

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

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

- **PnPInstall.pdf:**
  Install the Plug and Play information file (*.inf) under Windows 95/98/2000.

3. FUNCTION DESCRIPTION

These functions in DLL are divided into several groups as following:

1. The test functions
2. The DI/O functions
3. The AD/DA fixed-mode functions
4. The Driver functions
5. The AD Interrupt Mode functions
6. The AD, Channel Scan functions
7. The AD Interrupt, Channel Scan functions

The functions of test listing as follows:

1. A822_SHORT_SUB_2
2. A822_FLOAT_SUB_2
3. A822_Get_DLL_Version
4. A822_GetDriverVersion

The functions of DI/O listing as follows:

1. A822_DI
2. A822_DO
3. A822_InputByte
4. A822_InputWord
5. A822_OutputByte
6. A822_OutputWord

The functions of AD/DA listing as follows:

1. A822_SetChGain
2. A822_Fast_AD_Hex
3. A822_Fast_AD_Float
4. A822_AD_Hex
5. A822_AD_Float
6. A822_ADs_Hex
7. A822_ADs_Float
8. A822_Hex2Float
9. A822_DA_Hex
10. A822 _DA_Uni5
11. A822 _DA_Uni10

The functions of Driver listing as follows:

1. A822_DriverInit
2. A822_DriverClose
3. A822_DELAY
4. A822_Check_Address
5. A822_SetTriggerMode
6. A822_GetConfigAddress
7. A822_ActiveBoard
8. A822_SetCounter
9. A822_ReadCounter

The functions of AD Interrupt listing as follows:

1. A822_Int_Install
2. A822_Int_Start
3. A822_Int_Stop
4. A822_Int_Remove
5. A822_Int_GetCount
6. A822_Int_GetHexBuf
7. A822_Int_GetFloatBuf

The functions of AD, Channel Scan listing as follows:

1. A822_ChScan_Clear
2. A822_ChScan_Add
3. A822_ChScan_PollingHex
4. A822_ChScan_PollingFlaot

The functions of AD Interrupt, Channel Scan listing as follows:

1. A822_ChScan_IntInstall
2. A822_ChScan_IntStart
3. A822_ChScan_IntStop
4. A822_ChScan_IntRemove
5. A822_ChScan_IntGetCount
6. A822_ChScan_IntGetHexBuf
7. A822_ChScan_IntGetFloatBuf

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

Keyword	Parameter Set by user before calling this function ?	User Gets the data/value from this parameter after calling this function ?
[In]	Yes	No
[Out]	No	Yes
[In, Out]	Yes	Yes

Note: All of the parameters require space allocation by the user.

3.1 TEST FUNCTION

3.1.1 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 → short integer

3.1.2 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 → floating point value

3.1.3 A822_Get_DLL_Version

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

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

- **Parameter:**
  void

- **Return:**
  return=0x200 → Version 2.00 **(WORD = 16-bit unsigned integer)**

3.1.4 A822_GetDriverVersion

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

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

- **Parameter:**
  wDriverVersion :[Out] the address of wDriverVersion.
  when wDriverVerion=0x210 → version 2.10

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.2 DI/DO FUNCTION

3.2.1 A822_DI

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

- **Syntax:**
  WORD A822_DI(WORD *wInVal);

- **Parameter:**
  wInVal : [In] 16 bits Digital-Input value.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.2.2 A822_DO

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

- **Syntax:**
  WORD A822_DO(WORD wHexValue);

- **Parameter:**
  wHexValue : [In] 16 bit data sent to digital output port

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.2.3 A822_OutputByte

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

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

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

- **Return:**
  void

3.2.4 A822_OutputWord

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

- **Syntax:**
  void A822_OutputByte(WORD wPortAddr, WORD wOutputVal);

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

- **Return:**
  void

3.2.5 A822_InputByte

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

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

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

- **Return:**
  16-bit data with the leading 8 bits are all 0

3.2.6 A822_InputWord

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

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

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

- **Return:**
  16-bit data.

3.3 A/D, D/A FUNCTION

3.3.1 A822_SetChGain

- **Description:**

  The subroutine sets the channel number and configuration code for the ADC. And then user delays for the settling time.

  The user have to call this function once before calling the "A822_Fast_AD_Hex()", "A822_Fast_AD_Float()", "A822_Int_Start()", "A822_ADs_Hex()" and "A822_ADs_Float()" functions.

- **Syntax:**

  WORD A822_SetChGain
  (WORD wChannel, WORD wConfig, WORD wCardType);

- **Parameter:**

  wChannel : [In] A/D channel number, 0 to 15.
  wConfig : [In] Configuration code, refer to "Section 2.1 Range Configuration Code" for detailed information.
  wCardType : [In] 0 → OME-A-822PGL, 1→ OME-A-822PGH

- **Return:**

  Please refer to "Section 2.2 Error Code" for the detailed information.

3.3.2 A822_Fast_AD_Hex

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12 bits for OME-A822PGH/L. The user has to call the "A822_SetChGain()" function before calling this function.

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

- **Parameter:**
  wVal : [Out] 12 bits hex value of Analog-Input.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.3.3 A822_Fast_AD_Float

- **Description:**
  This subroutine will perform an A/D conversion by polling. The A/D converter is 12 bits for OME-A822PGH/L. This subroutine will compute the result according to the **configuration code**. The user has to call the "A822_SetChGain()" function before calling this function.

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

- **Parameter:**
  fVal : [Out] Floating point value of Analog-Input.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.3.4 A822_AD_Hex

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

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

- **Parameter:**
  wChannel : [In] A/D channel number, 0 to 15.
  wConfig : [In] Configuration code, refer to "Section 2.1 Range Configuration Code" for detail information
  wCardType : [In] 0 → OME-A-822PGL, 1→ OME-A-822PGH
  wVal : [Out] 12 bits hex value of Analog-Input.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.3.5 A822_AD_Float

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

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

- **Parameter:**
  wChannel : [In] A/D channel number, 0 to 15.
  wConfig : [In] Configuration code, refer to "Section 2.1 Range Configuration Code" for detail information
  wCardType : [In] 0 → OME-A-822PGL, 1→ OME-A-822PGH
  fVal : [Out] Floating point value of Analog-Input.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.3.6 A822_ADs_Hex

- **Description:**
  This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A822_AD_Hex 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. 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. The user has to call the "A822_SetChGain()" function before calling this function.

- **Syntax:**
  WORD A822_ADs_Hex(WORD wBuf[], WORD wCount);

- **Parameter:**
  wBuf : [Out] Starting address of the data buffer
  (In WORD format)
  The user must allocate space for this buffer and send the address in the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.

  wCount : [In] Number of A/D conversions will be performed

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.3.7 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. Then the A/D data are stored in a data buffer in Float format. The **fBuf** is the starting address of this data buffer. The user has to call the "A822_SetChGain()" function before calling this function.

- **Syntax:**

  WORD A822_ADs_Float(float fBuf[], WORD wCount);

- **Parameter:**

  fBuf : [Out] Starting address of the data buffer
  (In float format)
  The user must allocate space for this buffer and send the address in the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.

  wCount : [In] Number of A/D conversions will be performed

- **Return:**

  Please refer to "Section 2.2 Error Code" for the detail information.

3.3.8 A822_DA_Hex

- **Description:**
  This subroutine will send 12 bits of 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 select -5V, the 0xfff will send out 5V. If hardware select -10V, the 0xfff will send out 10V. The factory setting select 0-5V D/A output range.**

- **Syntax:**
  WORD A822_DA_Hex(WORD wChannel, WORD wHexValue);

- **Parameter:**
  wChannel : [In] D/A channels number, validate for 0 or 1.
  wHexValue : [In] 12 bit data send to D/A converter

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.3.9 A822 _DA_Uni5

- **Description:**
  This subroutine will send the 12 bits of data to D/A analog output. The output range of D/A dependent on **hardware jumper settings, 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 hardware manual to setting jumpers.

- **Syntax:**
  void A822 _DA_Uni5 (WORD wChannel, float fValue);

- **Parameter:**
  wChannel : [In] D/A channel number, validated for 0 or 1
  fValue : [In] 12 bit data send to D/A converter

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.3.10 A822 _DA_Uni10

- **Description:**
  This subroutine will send the 12 bits of data to D/A analog output. The output range of D/A dependent on **hardware jumper settings, 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 between 0.0v and 10.0v.** Please refer to hardware manual to setting jumpers.

- **Syntax:**
  void A822 _DA_Uni10 (WORD wChannel, float fValue);

- **Parameter:**
  wChannel : [In] D/A channels number, validate for 0 or 1
  fValue : [In] 12 bit data send to D/A converter

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4 DRIVER FUNCTION

3.4.1 A822_DriverInit

- **Description:**
  This subroutine will open the device driver. After calling the A822_DriverInit() function, the user still have to call the A822_ActiveBoard() function before accessing the device.

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

- **Parameter:**
  WTotalBoards : [Out] Returns the number of boards that were found by the driver.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4.2 A822_DriverClose

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

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

- **Parameter:**
  None

- **Return:**
  None

3.4.3 A822_SetTriggerMode

- **Description:**

  This subroutine will set the trigger mode - internal or external. The default value/mode is the internal trigger mode if the user does not use this function. The user must call this function before calling any AD function (include the “A822_ActiveBoard” and "A822_Check_Address" functions) 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:**

  void A822_SetTriggerMode(WORD wTriggerMode)

- **Parameter:**

  wTriggerMode : [In] 0: Internal Trigger Mode
  1: External Trigger Mode

- **Return:**

  None

3.4.4 A822_DELAY

- **Description:**
  This subroutine will delay **wDownCount** mS (machine independent timer).
  This function uses the Counter0 to implement delay function.
  The unit of A822_DELAY() is 0.5uSeconds. (2MHz ➔ 2000K times/sec)

- **Syntax:**
  WORD A822_DELAY(WORD wDownCount);

- **Parameter:**
  wDownCount : [In] Number of 0.5uS it will delay

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4.5 A822_Check_Address

- **Description:**
  This subroutine will detect the OME-A-822PGH/L in I/O base address. This subroutine will perform one A/D conversion. If successful, it found an OME-A-822PGH/L. This function will always return 0 if the user set the trigger mode to external.

  Refer to the function "A822_SetTriggerMode".

- **Syntax:**
  WORD A822_Check_Address(void);

- **Parameter:**
  None

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4.6 A822_GetConfigAddress

- **Description:**
  This subroutine returns the Base-Address and board-number of the current board.

  If the current board is invalid, the Base-Address will be 0.

- **Syntax:**
  WORD A822_GetConfigAddress
  (WORD *wAddrBase, WORD *wCurrentBoard);

- **Parameter:**
  wAddrBase : [Out] Returns the Base-Address of the current board.
  wCurrentBoard : [Out] Returns the board-number of the current board.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4.7 A822_ActiveBoard

- **Description:**
  This subroutine activates the specified board. User can then call the A822_Check_Address() function to check this hardware automatically. If the function can not access this device, it also returns A822_CardNotFound error code. Please refer to the "A822_DriverInit()" function for the valid range of board numbers.

- **Syntax:**
  WORD A822_ActiveBoard(WORD wBoardNo);

- **Parameter:**
  wBoardNo : [In] The board number to be active.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.4.8 A822_SetCounter

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

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

- **Parameter:**
  wCounterNo : [Input] Counter Number 0 to 2 for the 8254
  wCounterMode : [Input] Counter Mode 0 to 5 for the 8254
  wCounterValue : [Input] Counter Value 0 to 65535 for the 8254

- **Return:**
  None

3.4.9 A822_ReadCounter

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

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

- **Parameter:**
  wCounterNo : [Input] Counter Number 0 to 2 for the 8254
  wCounterMode : [Input] Counter Mode 0 to 5 for the 8254

- **Return:**
  Return the counter's value and only the lower WORD is valid.

3.5 AD, INTERRUPT FUNCTION

3.5.1 A822_Int_Install

- **Description:**
  This subroutine will install interrupt handler and allocate buffer. For more detail information of using interrupt please refer to "Section 3.5.8 Architecture of Interrupt Mode".

- **Syntax:**
  WORD A822_Int_Install(HANDLE *hEvent, DWORD dwCount);

- **Parameter:**
  hEvent : [In] The Event handle that created by the user.
  dwCount : [In] The desired A/D entries count for interrupt transfer.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.2 A822_Int_Start

- **Description:**
  This subroutine will clear the interrupt-counter and start the interrupt transfer for a specific A/D channel and programming the gain code and sampling rate. The user has to call the "A822_SetChGain()" function once before this function.

- **Syntax:**
  WORD A822_Int_Start(WORD c1, Word c2);

- **Parameter:**
  c1,c2 : [In] the sampling rate is 2M/(c1*c2)
  c1➔ Counter1, c2➔ Counter2
  These values be used only when the Trigger-Mode is set to Internal-Trigger. Please refer to the function "A822_SetTriggerMode".

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.3 A822_Int_Stop

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

- **Syntax:**
  WORD A822_Int_Stop(void);

- **Parameter:**
  void.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.4 A822_Int_Remove

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

- **Syntax:**
  WORD A822_Int_Remove(void);

- **Parameter:**
  void.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.5 A822_Int_GetCount

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

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

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

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.6 A822_Int_GetHexBuf

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

- **Syntax:**
  WORD A822_Int_GetHexBuf(WORD wBuf[], DWORD dwNum)

- **Parameter:**
  wBuf : [Out] The address of wBuffer(In WORD format).
  The user must allocate spaces for this buffer and send the address in the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.
  dwNum : [In] The number to transfer.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.7 A822_Int_GetFloatBuf

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

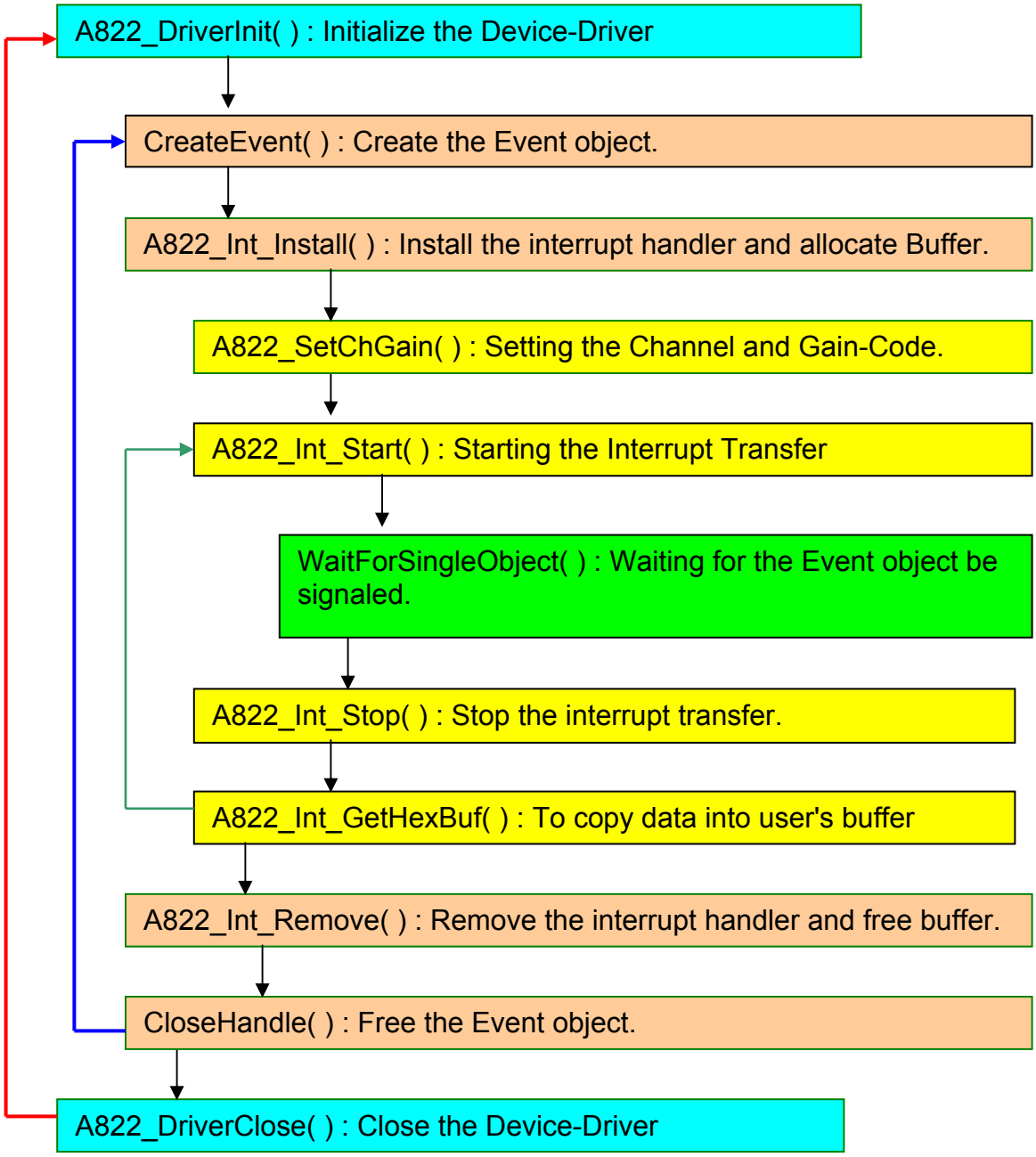
- **Syntax:**
  WORD A822_Int_GetFloatBuf(float fBuf[],DWORD dwNum)

- **Parameter:**
  fBuf : [Out] The address of fBuffer(In float format).
  The user must allocate spaces for this buffer and send the address in the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.
  dwNum : [In] The number to transfer.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.5.8 Architecture of Interrupt mode

The functions listed in sections 3.5.1 to 3.5.7 are the functions to perform the A/D conversion with interrupt transfer. The flow chart to program these functions is given as follows:

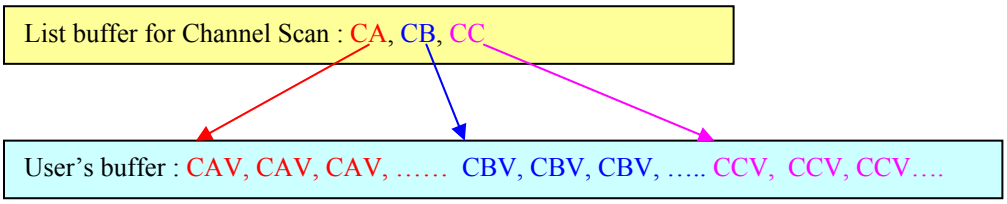


3.6 AD, CHANNEL SCAN FUNCTION

3.6.1 Introduction

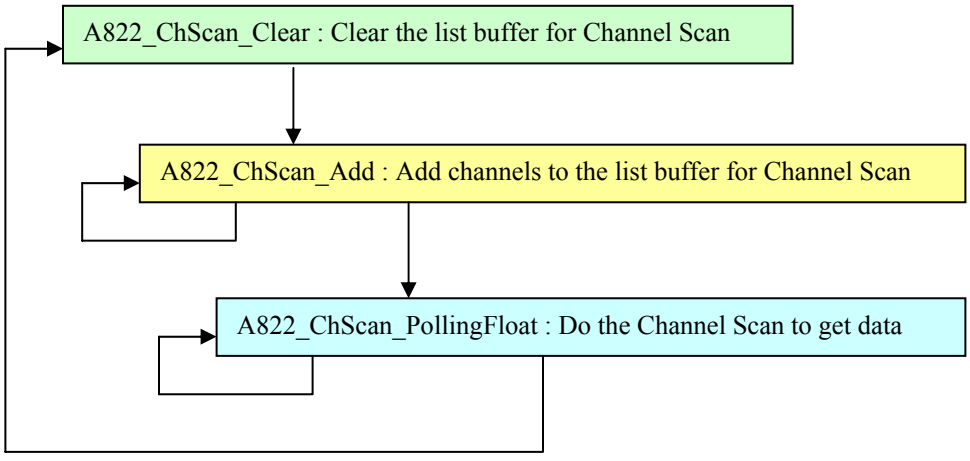
The user can specify channels into a list buffer. User then uses other functions to perform the ADC to get the data. Then user can read the list buffer to change to next channel and set to specify configuration code.

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

The user program's architecture should be as described below:



3.6.2 A822_ChScan_Clear

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

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

- **Parameter:**
  None

- **Return:**
  None

3.6.3 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 the list buffer (for the Channel Scan) is 100 channels.

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

- **Parameter:**
  wChannel : [In] Which channel to be scanned.
  WConfig : [In] Specify the configuration-code for this channel.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.6.4 A822_ChScan_PollingHex

- **Description:**

  This subroutine will perform a number of A/D conversions by polling. After getting the channel's data, it then reads 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 have to call the A822_ChScan_Clear() and A822_ChScan_Add() functions to setup the list buffer for Channel Scan. Please refer to the "Section 3.6.1 Introduction" for more information.

- **Syntax:**

  WORD A822_ChScan_PollingHex
  (WORD wCardType, WORD wBuf[], WORD wNumPerCh);

- **Parameter:**

  WCardType : [In] 0: OME-A-822L 1: OME-A-822H

  wBuf : [Out] Starting address of the data buffer (WORD format)
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user can analyzes these data from the buffer after calling this function.

  The buffer size
  = Total-Channels * wNumPerCh * sizeof(WORD)

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

- **Return:**

  Please refer to "Section 2.2 Error Code" for the detail information.

3.6.5 A822_ChScan_PollingFloat

- **Description:**

  This subroutine will perform a number of A/D conversions by polling. And 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 gets stored in a buffer in floating format.

  Before calling this function, the user have to call the A822_ChScan_Clear() and A822_ChScan_Add() functions to setup the list buffer for Channel Scan. Please refer to the "Section 3.6.1 Introduction" for more information.

- **Syntax:**

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

- **Parameter:**

  WCardType : [In] 0: OME-A-822L 1: OME-A-822H

  fBuf : [Out] Starting address of the data buffer (floating format)
  The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.

  The buffer size
  = Total-Channels * wNumPerCh * sizeof(float)

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

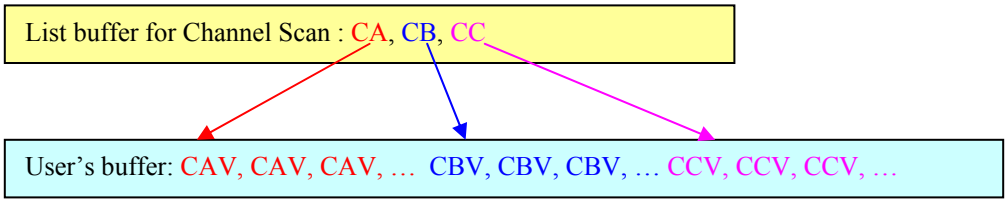
- **Return:**

  Please refer to "Section 2.2 Error Code" for the detail information.

3.7 AD INTERRUPT, CHANNEL SCAN FUNCTION

3.7.1 Introduction

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



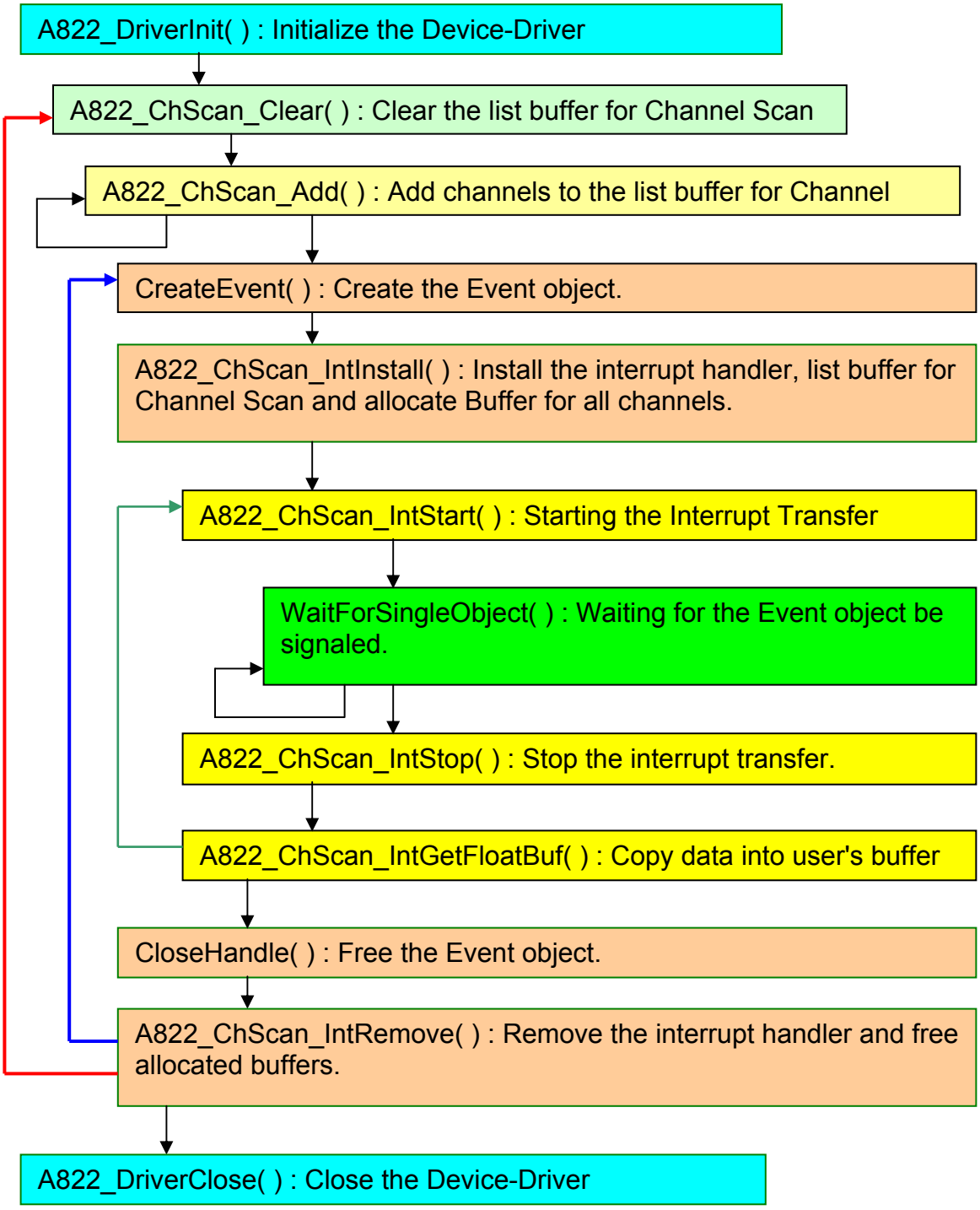
The data will be saved into the following style:

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 the next channel and specified configuration code, it will have to delay for period equal to the settling time before next ADC. The interrupt service routinue doesn't provide for the delay 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. Actually the channel 2 has the sampling rate of 5KHz and the channel 0 also has the sampling rate of 5KHz.

The user program architecture should be as described here:



3.7.2 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 install the interrupt, the user has to call the "A822_ChScan_Clear()" and "A822_ChScan_Add()" functions to setup the list buffer for Channel Scan firstly. For more detail information regarding using interrupts please refer to "Section 3.7.1 Introduction".

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

- **Parameter:**
  hEvent : [In] The Event handle created by the user.
  dwNumPerCh : [In] The desired A/D count for every channels to transfer.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.7.3 A822_ChScan_IntStart

- **Description:**
  This subroutine will clear the interrupt-counter and start the interrupt transfer for the specific A/D channels and 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
  These counter's value only used when the Trigger-Mode is set to Internal-Trigger. Please refer to the "A822_SetTriggerMode" function.
  wCardType : [In] 0: OME-A-822L 1: OME-A-822H

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.7.4 A822_ChScan_IntStop

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

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

- **Parameter:**
  void.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.7.5 A822_ChScan_IntRemove

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

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

- **Parameter:**
  void.

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.7.6 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:**
  Please refer to "Section 2.2 Error Code" for the detailed information.

3.7.7 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).
  The user must allocate spaces for this buffer and send the address in the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.
  Buffer size = Total-Channels * dwNumPerCh * sizeof(WORD)

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

3.7.8 A822_ChScan_IntGetFloatBuf

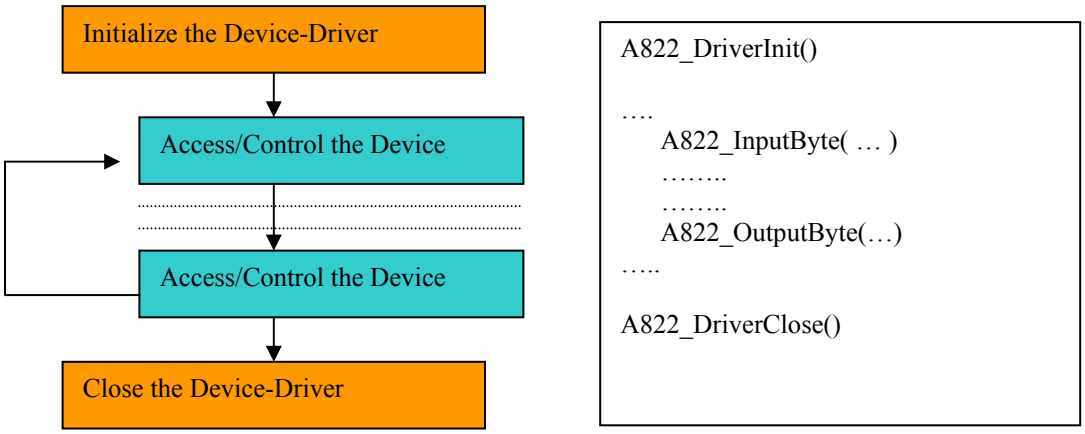
- **Description:**
  This subroutine will copy the transferred interrupted data into the user's buffer.

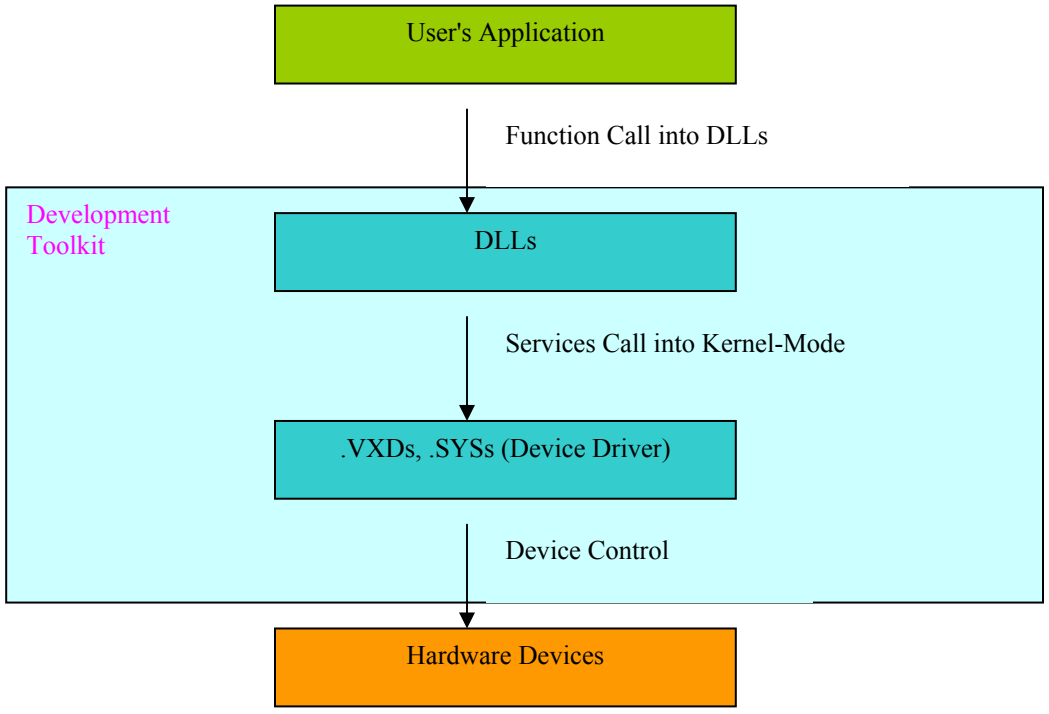
- **Syntax:**
  WORD A822_ChScan_IntGetFloatBuf(float fBuf[])

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

- **Return:**
  Please refer to "Section 2.2 Error Code" for the detail information.

4. PROGRAM ARCHITECTURE





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

WARRANTY/DISCLAIMER

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.

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

M4032/0104