User's Guide

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

ISO 9001 CERTIFIED CORPORATE QUALITY
STAMFORD, CT

ISO 9002 CERTIFIED CORPORATE QUALITY
MANCHESTER, UK

OME-A826PG
ISA-Bus Multi-Functional Board
Software Manual

omega.com®
ΩOMEGA®

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.

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.

OME-A-826PG

Software Manual

[For Windows 95/98/NT]

Table of Contents

1 Introduction 4
1.1 References 5
1.2 Range Configuration Code 6
2 Declaration Files 7
2.1 For C user 8
2.1.1 A826.H (for Win 95/98) 8
2.1.2 A826.H (for Win NT) 11
2.1.3 A826U.Cpp (for BCB) 13
2.1.4 The VC++ Demo Result: 14
2.1.5 Borland C++ Builder Demo Result 14
2.2 For The VB user 15
2.2.1 A826.BAS (for Win 95/98) 15
2.2.2 A826.BAS (for Win NT) 18
2.2.3 A826U.BAS 21
2.2.4 The VB Demo Result: 21
2.3 For The Delphi user 22
2.3.1 A826.PAS (for Win 95/98) 22
2.3.2 A826.PAS (for Win NT) 26
2.3.3 A826U.PAS 29
2.3.4 Delphi Demo Result : 30
3 Function Description 31
3.1 TEST Function 33
3.1.1 SHORT_SUB_2 33
3.1.2 FLOAT_SUB_2 33
3.1.3 A826_Get_DLL_Version 34
3.1.4 A826_GetDriverVersion 34
3.2 DI/DO Function 35
3.2.1 A826_Di 35
3.2.2 A826_Do 35
3.2.3 A826_OutputByte 36
3.2.4 A826_OutputWord 36
3.2.5 A826_InputByte 37
3.2.6 A826_InputWord 37
3.3 A/D , D/A Functions 38
3.3.1 A826_AD 38
3.3.2 A826_DA 38
3.3.3 A826_ADs_Hex 39
3.3.4 A826_ADs_Float 40
3.3.5 A826_Uni5_DA 41
3.3.6 A826_Uni10_DA 41
3.4 Driver Functions 42
3.4.1 A826_DriverInit 42
3.4.2 A826_DriverClose 42
3.4.3 A826_DELAY 43

3.4.4 A826_Check_Address 43
3.5 AD, Interrupt functions 44
3.5.1 A826_InstallIrq 44
3.5.2 A826_AD_INT_Start 44
3.5.3 A826_AD_INT_Stop 45
3.5.4 A826_GetIntCount 45
3.5.5 A826_GetBuffer 46
3.5.6 A826_GetFloatBuffer 46
3.5.7 Diagram of Interrupt Mode 47
3.6 AD , DMA functions 49
3.6.1 A826_AD_DMA_InstallIrq 49
3.6.2 A826_AD_DMA_RemoveIrq 49
3.6.3 A826_AD_DMA_Start 50
3.6.4 A826_AD_DMA_Stop 51
3.6.5 A826_AD_DMA_IsNotFinished 51
3.6.6 A826_AD_DMA_GetBuffer 52
3.6.7 A826_AD_DMA_GetFloatBuffer 52
3.6.8 Diagram of AD , DMA Mode 53
4 Program Architecture 55
5 Contact Us 56

1 Introduction

The OME-A-826PG is a multifunction, 16 bits resolution A/D, D/A and digital I/O card. The feature of the OME-A-826PG are given as below:

1. A/D=16 bits, 16 channels(single-ended) or 8 channels(differential)
2. A-826PG : low gain (1/2/4/8), the analog input signal range configuration code is given in Sec. 4.1
3. DA=12 bits, 2 channels, 0-5V or 0-10V output by **hardware JP1 setting**
4. 16 channels TTL compatible digital input
5. 16 channels TTL compatible digital output

The A826.DLL and A826.Vxd is a collection of data acquisition subroutines for the OME-A-826PG Windows 95/98 Applications. These subroutines are written with C language and perform a variety of data acquisition operations.

The subroutines in A826.DLL are easy understanding as its name standing for. It provides powerful, easy-to-use subroutine for developing your data acquisition application. Your program can call this DLL functions by VC++, VB , Delphi , Borland C++ Builder easily. To speed-up your developing process, some demonstration source program are provided.

The OME-A-826PG consists of these DLLs and device driver :

For Windows 95/98

- A826.dll, A826.lib →Libraries for A826 PG card
- A826.Vxd → A826 Device driver for Windows 95/98

For Windows NT

- A826.dll, A826.lib →Libraries for OME-A826 PGL/PGH card
- A826.sys, Napwnt.sys →A826 Device driver for Windows NT

These DLLs can perform a variety of data acquisition operations as follows:

1.1 References

Please refer to the following user manuals:

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

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

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

1.2 Range Configuration Code

The AD converter of the OME-A-826PG is 16 bits under all configuration code. 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-A-826PG), it will get nearly the same result **except resolution. So choose the correct configuration code can achieve the most high precision measurement.**

OME-A-826PG Input Signal Range Configuration Code Table

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	± 10V	0
Bipolar	± 5V	1
Bipolar	± 2.5V	2
Bipolar	± 1.25V	3

2 Declaration Files

For the Windows 95/98 user:

```
|--\Driver
|   |--\A826.DLL          <-- Dynamic Linking Library
|   |--\A826.Vxd          <-- Device driver for A826PG
|   |--\BCB3
|   |   |--\A826.h        <-- Header file
|   |   |--\A826.Lib      <-- Import Library for BCB
|   |   +--\A826u.cpp     <-- Some function for BCB
|   |--\Delphi3
|   |   |--\A826.pas      <-- Declaration file
|   |   +--\A826u.pas     <-- Some function for Delphi
|   |--\VB5
|   |   |--\A826.bas      <-- Declaration file
|   |   +--\A826u.bas     <-- Some function for VB
|   +--\VC5
|       |--\A826.h        <-- Header file
|       +--\A826.Lib      <-- Import Library for VC++
```

For the Windows NT user:

```
|--\Driver
|   |--\A826.DLL          <-- Dynamic Linking Library
|   |--\A826.sys          <-- device driver
|   |--\Napwnt.sys        <-- device driver
|   |--\BCB3
|   |   |--\A826.h        <-- Header file
|   |   |--\A826.Lib      <-- Import Library for BCB
|   |   +--\A826u.cpp     <-- Some function for BCB
|   |--\Delphi3
|   |   |--\A826.pas      <-- Declaration file
|   |   +--\A826u.pas     <-- Some function for Delphi
|   |--\VB5
|   |   |--\A826.bas      <-- Declaration file
|   |   +--\A826u.bas     <-- Some function for VB
|   +--\VC5
|       |--\A826.h        <-- Header file
|       +--\A826.Lib      <-- Import Library for VC++
```

2.1 For C user

2.1.1 A826.H (for Win 95/98)

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

/***************** DEFINE A826 RELATIVE ADDRESS *****************/
#define   TIMER0                0x00
#define   TIMER1                0x01
#define   TIMER2                0x02
#define   TIMER_MODE            0x03
#define   AD_LO                 0x04    /* Analog to Digital, Low Byte */
#define   AD_HI                 0x05    /* Analog to Digital, High Byte */
#define   DA_CH0_LO             0x04    /* Digit to Analog, CH 0 */
#define   DA_CH0_HI             0x05
#define   DA_CH1_LO             0x06    /* Digit to Analog, CH 1 */
#define   DA_CH1_HI             0x07
#define   DI_LO                 0x06    /* Digit Input */
#define   DO_LO                 0x0D    /* Digit Output */

#define   CLEAR_IRQ             0x08
#define   SET_GAIN              0x09
#define   SET_CH                0x0A
#define   SET_MODE              0x0B
#define   SOFT_TRIG             0x0C

#define   POLLING_MODE          1
#define   DMA_MODE              2
#define   INTERRUPT_MODE        6

/*** define the gain mode ***/
#define   A826_BI_1             0
#define   A826_BI_10            1
#define   A826_BI_100           2
#define   A826_BI_1000          3
#define   A826_UNI_1            4
#define   A826_UNI_10           5
#define   A826_UNI_100          6
#define   A826_UNI_1000         7
#define   A826_BI_05            8
#define   A826_BI_5             9
#define   A826_BI_50            10
#define   A826_BI_500           11

#define   A826_BI_2             1
#define   A826_BI_4             2
#define   A826_BI_8             3
#define   A826_UNI_2            5
#define   A826_UNI_4            6
#define   A826_UNI_8            7
```

```
#define NoError                 0
#define DriverOpenError         1
#define DriverNoOpen            2
#define GetDriverVersionError   3
#define InstallIrqError         4
#define ClearIntCountError      5
#define GetIntCountError        6
#define AdError1                100
#define AdError2                -200.0
#define InstallBufError         9
#define GetBufferError          10
#define INTStartError           11
#define INTStopError            12
#define InstallDmaIrqError      13
#define RemoveDmaIrqError       14
#define DmaStartError           15
#define DmaStopError            16
#define DmaGetDataError         17
#define TimeoutError            18
#define AllocateMemoryError     19
#define OtherError              20

//********* Test Functios ************
EXPORTS short CALLBACK SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK A826_Get_DLL_Version(void);
EXPORTS WORD CALLBACK A826_GetDriverVersion(WORD *wDriverVersion);

//********* DI/DO Functios ************
EXPORTS WORD CALLBACK A826_DI(WORD wBase);
EXPORTS void CALLBACK A826_DO(WORD wBase, WORD wHexValue);

//********* DA/AD Functios ************
EXPORTS float CALLBACK A826_AD(WORD wBase, WORD wChannel, WORD wConfig);
EXPORTS WORD CALLBACK A826_ADs_Hex(WORD wBase, WORD wChannel, WORD wConfig,
                                   short wBuf[], WORD wCount);
EXPORTS WORD CALLBACK A826_ADs_Float(WORD wBase, WORD wChannel, WORD wConfig,
                                     float fBuf[], WORD wCount);
EXPORTS void CALLBACK A826_DA(WORD wBase, WORD wChannel, WORD wHexValue);
EXPORTS void CALLBACK A826_Uni5_DA(WORD wBase, WORD wChannel, float fValue);
EXPORTS void CALLBACK A826_Uni10_DA(WORD wBase, WORD wChannel, float fValue);

//********* Driver Functios ************
EXPORTS WORD CALLBACK A826_DriverInit(void);
EXPORTS void CALLBACK A826_DriverClose(void);
EXPORTS WORD CALLBACK A826_DELAY(WORD wBase, WORD wDownCount);
EXPORTS WORD CALLBACK A826_Check_Address(WORD wBase);
EXPORTS void CALLBACK A826_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
EXPORTS void CALLBACK A826_OutputWord(WORD wPortAddr, WORD wOutputVal);
EXPORTS WORD CALLBACK A826_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK A826_InputWord(WORD wPortAddr);
```

```
//********* IRQ Functios ************
EXPORTS WORD CALLBACK A826_InstallIrq
            (WORD wBase, WORD wIrq, HANDLE *hEvent, DWORD dwCount);
EXPORTS WORD CALLBACK A826_AD_INT_Start(WORD Ch, WORD Gain, WORD c1, WORD c2);
EXPORTS WORD CALLBACK A826_AD_INT_Stop(void);
EXPORTS WORD CALLBACK A826_GetIntCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A826_GetBuffer(DWORD dwNum, short wBuffer[]);
EXPORTS WORD CALLBACK A826_GetFloatBuffer(DWORD dwNum, float fBuffer[]);

//********* DMA Functios ************
EXPORTS WORD CALLBACK A826_AD_DMA_InstallIrq(WORD wBase,WORD wIrq,WORD wDmaChan);
EXPORTS WORD CALLBACK A826_AD_DMA_RemoveIrq(void);
EXPORTS WORD CALLBACK A826_AD_DMA_Start(WORD Ch,WORD Gain,WORD c1,WORD c2,
                                        int cnt, WORD wPassOut[]);
EXPORTS WORD CALLBACK A826_AD_DMA_Stop(void);
EXPORTS WORD CALLBACK A826_AD_DMA_IsNotFinished(void);
EXPORTS WORD CALLBACK A826_AD_DMA_GetBuffer(short wBuf[]);
EXPORTS WORD CALLBACK A826_AD_DMA_GetFloatBuffer(float fBuf[]);
```

2.1.2 A826.H (for Win NT)

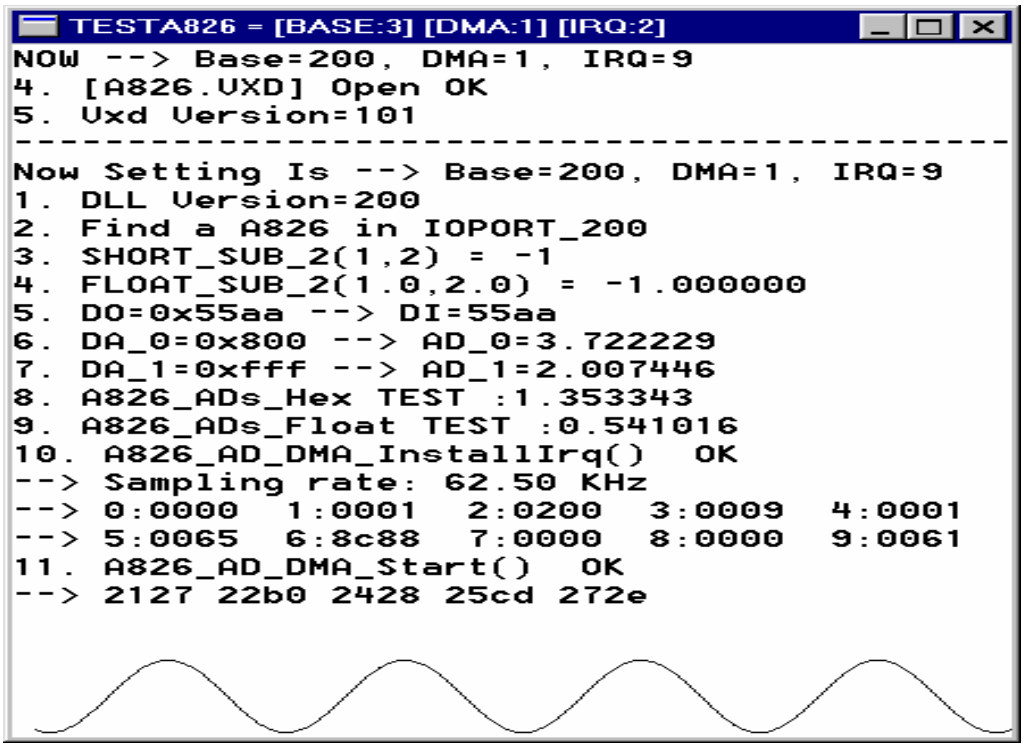
```
#ifdef __cplusplus
    #define EXPORTS extern "C" __declspec (dllimport)
#else
    #define EXPORTS
#endif

/***************** DEFINE A826 RELATIVE ADDRESS *****************/
#define   TIMER0                0x00
#define   TIMER1                0x01
#define   TIMER2                0x02
#define   TIMER_MODE            0x03
#define   AD_LO                 0x04    /* Analog to Digital, Low Byte */
#define   AD_HI                 0x05    /* Analog to Digital, High Byte */
#define   DA_CH0_LO             0x04    /* Digit to Analog, CH 0 */
#define   DA_CH0_HI             0x05
#define   DA_CH1_LO             0x06    /* Digit to Analog, CH 1 */
#define   DA_CH1_HI             0x07
#define   DI_LO                 0x06    /* Digit Input */
#define   DO_LO                 0x0D     /* Digit Output */

#define   CLEAR_IRQ             0x08
#define   SET_GAIN              0x09
#define   SET_CH                0x0A
#define   SET_MODE              0x0B
#define   SOFT_TRIG             0x0C

#define   POLLING_MODE          1
#define   DMA_MODE              2
#define   INTERRUPT_MODE        6

/*** define the gain mode ***/
#define   A826_BI_1             0
#define   A826_BI_10            1
#define   A826_BI_100           2
#define   A826_BI_1000          3
#define   A826_UNI_1            4
#define   A826_UNI_10           5
#define   A826_UNI_100          6
#define   A826_UNI_1000         7
#define   A826_BI_05            8
#define   A826_BI_5             9
#define   A826_BI_50            10
#define   A826_BI_500           11

#define   A826_BI_2             1
#define   A826_BI_4             2
#define   A826_BI_8             3
#define   A826_UNI_2            5
#define   A826_UNI_4            6
#define   A826_UNI_8            7
```

```
#define NoError                 0
#define DriverOpenError         1
#define DriverNoOpen            2
#define GetDriverVersionError   3
#define InstallIrqError         4
#define ClearIntCountError      5
#define GetIntCountError        6
#define AdError1                100
#define AdError2                -200
#define InstallBufError         9
#define AllocateMemoryError     10
#define CardTypeError           11
#define GetBufferError          12
#define TimeoutError            13
#define OtherError              14

//********* Test Functios ************
EXPORTS short CALLBACK SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK A826_Get_DLL_Version(void);
EXPORTS WORD CALLBACK A826_GetDriverVersion(WORD *wDriverVersion);

//********* DI/DO Functios ************
EXPORTS WORD CALLBACK A826_DI(WORD wBase);
EXPORTS void CALLBACK A826_DO(WORD wBase, WORD wHexValue);

//********* DA/AD Functios ************
EXPORTS float CALLBACK A826_AD(WORD wBase, WORD wChannel, WORD wConfig);
EXPORTS WORD CALLBACK A826_ADs_Hex(WORD wBase, WORD wChannel, WORD
wConfig,    short wBuf[], WORD wCount);
EXPORTS WORD CALLBACK A826_ADs_Float(WORD wBase, WORD wChannel, WORD
wConfig,    float fBuf[], WORD wCount);
EXPORTS void CALLBACK A826_DA(WORD wBase, WORD wChannel, WORD wHexValue);
EXPORTS void CALLBACK A826_Uni5_DA(WORD wBase, WORD wChannel, float fValue);
EXPORTS void CALLBACK A826_Uni10_DA(WORD wBase, WORD wChannel, float fValue);

//********* Driver Functios ************
EXPORTS WORD CALLBACK A826_DriverInit(void);
EXPORTS void CALLBACK A826_DriverClose(void);
EXPORTS WORD CALLBACK A826_DELAY(WORD wBase, WORD wDownCount);
EXPORTS WORD CALLBACK A826_Check_Address(WORD wBase);
EXPORTS void CALLBACK A826_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
EXPORTS void CALLBACK A826_OutputWord(WORD wPortAddr, WORD wOutputVal);
EXPORTS WORD CALLBACK A826_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK A826_InputWord(WORD wPortAddr);

//********* IRQ Functios ************
EXPORTS WORD CALLBACK A826_InstallIrq
        (WORD wBase, WORD wIrq, HANDLE *hEvent,DWORD dwCount);
EXPORTS WORD CALLBACK A826_AD_INT_Start
        (WORD Ch, WORD Gain, WORD c1, WORD c2);
EXPORTS WORD CALLBACK A826_AD_INT_Stop(void);
EXPORTS WORD CALLBACK A826_GetIntCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A826_GetBuffer(DWORD dwNum, short wBuffer[]);
EXPORTS WORD CALLBACK A826_GetFloatBuffer(DWORD dwNum, float fBuffer[]);
```

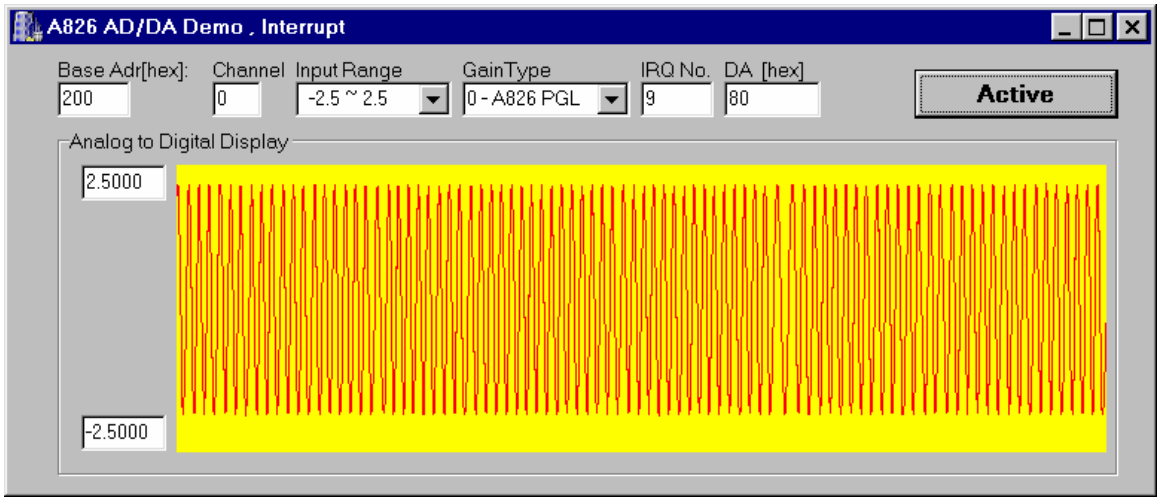
2.1.3 A826U.Cpp (for BCB)

```
#include <math.h>

//*-----------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                       *
//* HiLo : 1 --> High Gain , 0 --> Low Gain             *
//* Gain : 0-3                                                  *
//*-----------------------------------------------------*
float A826_AD2F(Word hex, int Gain )
{
   float ZeroBase, VoltageRange, FullRange ;
   short int   i ;

   ZeroBase  = 0.0 ;
   FullRange = 32767.0 ;
   VoltageRange = 10.0 ;
   i = hex ;
   Gain = Gain % 16;

   if ( (Gain < 0) || (Gain > 3) )
      return -100.0;

   return ((((i - ZeroBase) / FullRange) * VoltageRange) / pow(  2 , Gain));

}
```

2.1.4 The VC++ Demo Result:



2.1.5 Borland C++ Builder Demo Result



2.2 For The VB user

2.2.1 A826.BAS (for Win 95/98)

```
Attribute VB_Name = "A826"
'********************************************************************************
'      The Declare of A826.DLL for A826 DAQ Card
'********************************************************************************

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

'***************** DEFINE A826 RELATIVE ADDRESS *****************

Global Const TIMER0             = &H0
Global Const Timer1             = &H1
Global Const TIMER2             = &H2
Global Const TIMER_MODE         = &H3
Global Const AD_LO              = &H4           '* Analog to Digital, Low Byte *
Global Const AD_HI              = &H5           '* Analog to Digital, High Byte *
Global Const DA_CH0_LO          = &H4           '* Digit to Analog, CH 0 *
Global Const DA_CH0_HI          = &H5
Global Const DA_CH1_LO          = &H6           '* Digit to Analog, CH 1 *
Global Const DA_CH1_HI          = &H7
Global Const DI_LO              = &H6           '* Digit Input *
Global Const DO_LO              = &HD           '* Digit Output *

Global Const CLEAR_IRQ          = &H8
Global Const SET_GAIN           = &H9
Global Const SET_CH             = &HA
Global Const SET_MODE           = &HB
Global Const SOFT_TRIG          = &HC

Global Const POLLING_MODE  = 1
Global Const DMA_MODE      = 2
Global Const INTERRUPT_MODE = 6

'*** define the gain mode ***
Global Const A826_BI_1          = 0
Global Const A826_BI_10         = 1
Global Const A826_BI_100        = 2
Global Const A826_BI_1000       = 3
Global Const A826_UNI_1         = 4
Global Const A826_UNI_10        = 5
Global Const A826_UNI_100       = 6
Global Const A826_UNI_1000      = 7
Global Const A826_BI_05         = 8
Global Const A826_BI_5          = 9
Global Const A826_BI_50         = 10
Global Const A826_BI_500        = 11
```

```
Global Const A826_BI_2          = 1
Global Const A826_BI_4          = 2
Global Const A826_BI_8          = 3
Global Const A826_UNI_2         = 5
Global Const A826_UNI_4         = 6
Global Const A826_UNI_8         = 7

Global Const NoError                    = 0
Global Const DriverOpenError            = 1
Global Const DriverNoOpen               = 2
Global Const GetDriverVersionError      = 3
Global Const InstallIrqError            = 4
Global Const ClearIntCountError         = 5
Global Const GetIntCountError           = 6
Global Const AdError1                   = 100
Global Const AdError2                   = -200#
Global Const InstallBufError            = 9
Global Const GetBufferError             = 10
Global Const INTStartError              = 11
Global Const INTStopError               = 12
Global Const InstallDmaIrqError         = 13
Global Const RemoveDmaIrqError          = 14
Global Const DmaStartError              = 15
Global Const DmaStopError               = 16
Global Const DmaGetDataError            = 17
Global Const TimeoutError               = 18
Global Const AllocateMemoryError        = 19
Global Const OtherError                 = 20

'******* Test Functions ***********
Declare Function SHORT_SUB_2 Lib "A826.DLL" (ByVal nA As Integer, _
      ByVal nB As Integer) As Integer
Declare Function FLOAT_SUB_2 Lib "A826.DLL" (ByVal fA As Single, _
      ByVal fB As Single) As Single
Declare Function A826_Get_DLL_Version Lib "A826.DLL" () As Integer
Declare Function A826_GetDriverVersion Lib "A826.DLL" _
      (wDriverVersion As Integer) As Integer

'******* DI/DO Functions ***********
Declare Function A826_DI Lib "A826.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A826_DO Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wHexValue As Integer)

'******* AD/DA Functions ***********
Declare Function A826_AD Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer) As Single
Declare Function A826_ADs_Hex Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer, _
      wBuf As Integer, ByVal wCount As Integer) As Integer
Declare Function A826_ADs_Float Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer, _
      fBuf As Single, ByVal wCount As Integer) As Integer
Declare Sub A826_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wHexValue As Integer)
Declare Sub A826_Uni5_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal fValue As Single)
```

```
Declare Sub A826_Uni10_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal fValue As Single)

'******* Driver Functions ***********
Declare Function A826_DriverInit Lib "A826.DLL" () As Integer
Declare Sub A826_DriverClose Lib "A826.DLL" ()
Declare Function A826_DELAY Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wDownCount As Integer) As Integer
Declare Function A826_Check_Address Lib "A826.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A826_OutputByte Lib "A826.DLL" (ByVal wPortAddr As Integer, ByVal bOutputVal
As Byte)
Declare Sub A826_OutputWord Lib "A826.DLL" (ByVal wPortAddr As Integer, ByVal wOutputVal
As Integer)
Declare Function A826_InputByte Lib "A826.DLL" (ByVal wPortAddr As Integer) As Integer
Declare Function A826_InputWord Lib "A826.DLL" (ByVal wPortAddr As Integer) As Integer

'******* IRQ Functions ***********
Declare Function A826_InstallIrq Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wIrq As Integer, hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A826_GetIntCount Lib "A826.DLL" (dwVal As Long) As Integer
Declare Function A826_GetBuffer Lib "A826.DLL" (ByVal dwNum As Long, _
      wBuffer As Integer) As Integer
Declare Function A826_AD_INT_Start Lib "A826.DLL" (ByVal Ch As Integer, _
      ByVal Gain As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer
Declare Function A826_AD_INT_Stop Lib "A826.DLL" () As Integer
Declare Function A826_GetFloatBuffer Lib "A826.DLL" (ByVal dwNum As Long, _
      fBuffer As Single) As Integer

'******* DMA Functions ***********
Declare Function A826_AD_DMA_InstallIrq Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wIrq As Integer, ByVal wDmaChan As Integer) As Integer
Declare Function A826_AD_DMA_RemoveIrq Lib "A826.DLL" () As Integer
Declare Function A826_AD_DMA_Start Lib "A826.DLL" (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 A826_AD_DMA_Stop Lib "A826.DLL" () As Integer
Declare Function A826_AD_DMA_IsNotFinished Lib "A826.DLL" () As Integer
Declare Function A826_AD_DMA_GetBuffer Lib "A826.DLL" (wBuf As Integer) As Integer
Declare Function A826_AD_DMA_GetFloatBuffer Lib "A826.DLL" (fBuf As Single) As Integer
```

2.2.2 A826.BAS (for Win NT)

```
Attribute VB_Name = "A826"
'********************************************************************************
'      The Declare of A826.DLL for A826 DAQ Card
'********************************************************************************

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

'****************** DEFINE A826 RELATIVE ADDRESS *****************

Global Const TIMER0             = &H0
Global Const Timer1             = &H1
Global Const TIMER2             = &H2
Global Const TIMER_MODE         = &H3
Global Const AD_LO              = &H4           '* Analog to Digital, Low Byte *
Global Const AD_HI              = &H5           '* Analog to Digital, High Byte *
Global Const DA_CH0_LO          = &H4           '* Digit to Analog, CH 0 *
Global Const DA_CH0_HI          = &H5
Global Const DA_CH1_LO          = &H6           '* Digit to Analog, CH 1 *
Global Const DA_CH1_HI          = &H7
Global Const DI_LO              = &H6           '* Digit Input *
Global Const DO_LO              = &HD           '* Digit Output *

Global Const CLEAR_IRQ          = &H8
Global Const SET_GAIN           = &H9
Global Const SET_CH             = &HA
Global Const SET_MODE           = &HB
Global Const SOFT_TRIG          = &HC

Global Const POLLING_MODE           = 1
Global Const DMA_MODE               = 2
Global Const INTERRUPT_MODE         = 6

'*** define the gain mode ***
Global Const A826_BI_1              = 0
Global Const A826_BI_10             = 1
Global Const A826_BI_100            = 2
Global Const A826_BI_1000           = 3
Global Const A826_UNI_1             = 4
Global Const A826_UNI_10            = 5
Global Const A826_UNI_100           = 6
Global Const A826_UNI_1000          = 7
Global Const A826_BI_05             = 8
Global Const A826_BI_5              = 9
Global Const A826_BI_50             = 10
Global Const A826_BI_500            = 11

Global Const A826_BI_2              = 1
Global Const A826_BI_4              = 2
Global Const A826_BI_8              = 3
Global Const A826_UNI_2             = 5
Global Const A826_UNI_4             = 6
Global Const A826_UNI_8             = 7
```

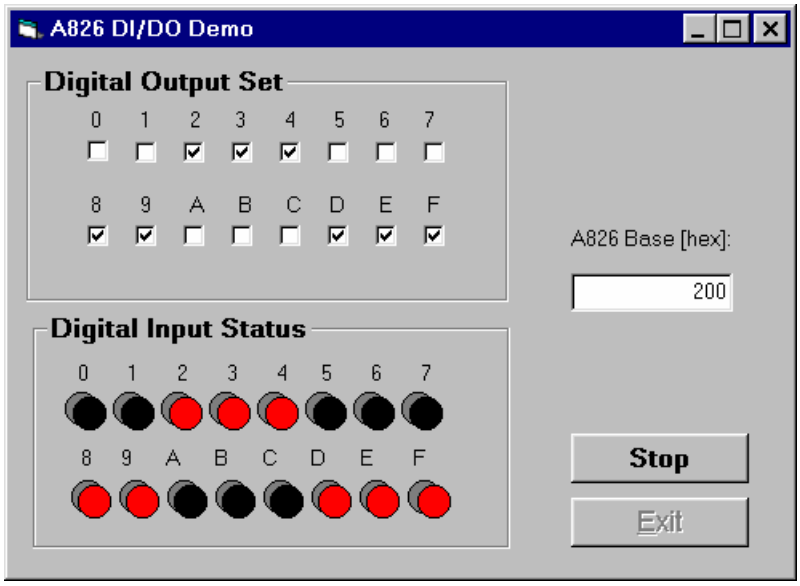
```
Global Const NoError                = 0
Global Const DriverOpenError        = 1
Global Const DriverNoOpen           = 2
Global Const GetDriverVersionError  = 3
Global Const InstallIrqError        = 4
Global Const ClearIntCountError     = 5
Global Const GetIntCountError       = 6
Global Const AdError1               = 100
Global Const AdError2               = -200#
Global Const InstallBufError        = 9
Global Const AllocateMemoryError    = 10
Global Const CardTypeError          = 11
Global Const GetBufferError         = 12
Global Const TimeoutError           = 13
Global Const OtherError             = 14

'******* Test Functions ***********
Declare Function SHORT_SUB_2 Lib "A826.DLL" (ByVal nA As Integer, _
      ByVal nB As Integer) As Integer
Declare Function FLOAT_SUB_2 Lib "A826.DLL" (ByVal fA As Single, _
      ByVal fB As Single) As Single
Declare Function A826_Get_DLL_Version Lib "A826.DLL" () As Integer
Declare Function A826_GetDriverVersion Lib "A826.DLL" _
      (wDriverVersion As Integer) As Integer

'******* DI/DO Functions ***********
Declare Function A826_DI Lib "A826.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A826_DO Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wHexValue As Integer)

'******* AD/DA Functions ***********
Declare Function A826_AD Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer) As Single
Declare Function A826_ADs_Hex Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer, _
      wBuf As Integer, ByVal wCount As Integer) As Integer
Declare Function A826_ADs_Float Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wConfig As Integer, _
      fBuf As Single, ByVal wCount As Integer) As Integer
Declare Sub A826_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal wHexValue As Integer)
Declare Sub A826_Uni5_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal fValue As Single)
Declare Sub A826_Uni10_DA Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wChannel As Integer, ByVal fValue As Single)
```

```
'******* Driver Functions ***********
Declare Function A826_DriverInit Lib "A826.DLL" () As Integer
Declare Sub A826_DriverClose Lib "A826.DLL" ()
Declare Function A826_DELAY Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wDownCount As Integer) As Integer
Declare Function A826_Check_Address Lib "A826.DLL" (ByVal wBase As Integer) As Integer
Declare Sub A826_OutputByte Lib "A826.DLL" (ByVal wPortAddr As Integer, ByVal bOutputVal
As Byte)
Declare Sub A826_OutputWord Lib "A826.DLL" (ByVal wPortAddr As Integer, ByVal wOutputVal
As Integer)
Declare Function A826_InputByte Lib "A826.DLL" (ByVal wPortAddr As Integer) As Integer
Declare Function A826_InputWord Lib "A826.DLL" (ByVal wPortAddr As Integer) As Integer

'******* IRQ Functions ***********
Declare Function A826_InstallIrq Lib "A826.DLL" (ByVal wBase As Integer, _
      ByVal wIrq As Integer, hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A826_GetIntCount Lib "A826.DLL" (dwVal As Long) As Integer
Declare Function A826_GetBuffer Lib "A826.DLL" (ByVal dwNum As Long, _
      wBuffer As Integer) As Integer
Declare Function A826_AD_INT_Start Lib "A826.DLL" (ByVal Ch As Integer, _
      ByVal Gain As Integer, ByVal c1 As Integer, ByVal c2 As Integer) As Integer
Declare Function A826_AD_INT_Stop Lib "A826.DLL" () As Integer
Declare Function A826_GetFloatBuffer Lib "A826.DLL" (ByVal dwNum As Long, _
      fBuffer As Single) As Integer
```

2.2.3 A826U.BAS

```
Attribute VB_Name = "A826u"

'*-----------------------------------------------------*
'* Return voltage value or -100.0 if any error occurs  *
'* or parameter is out of range.                       *
'* Gain : 0-3
'*-----------------------------------------------------*
Function A826_AD2F(ByVal hex As Integer, ByVal Gain As Integer) As Single
    Dim ZeroBase, BullRange, VoltageRange As Single

    ZeroBase = 0#
    FullRange = 32767#
    VoltageRange = 10#
    Gain = Gain Mod 16
    If Gain < 0 Or Gain > 3 Then
        A826_AD2F = -100#
        Exit Function
    End If

    A826_AD2F = ((((hex - ZeroBase) / FullRange) * VoltageRange) / (2 ^ Gain))
End Function
```

2.2.4 The VB Demo Result:



2.3 For The Delphi user

2.3.1 A826.PAS (for Win 95/98)

```
unit A826;

interface

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

const

//***************** DEFINE A826 RELATIVE ADDRESS *****************/
 TIMER0                  =  $00;
 TIMER1                  =  $01;
 TIMER2                  =  $02;
 TIMER_MODE              =  $03;
 AD_LO                   =  $04;    //* Analog to Digital, Low Byte */
 AD_HI                   =  $05;    //* Analog to Digital, High Byte */
 DA_CH0_LO               =  $04;    //* Digit to Analog, CH 0 */
 DA_CH0_HI               =  $05;
 DA_CH1_LO               =  $06;    //* Digit to Analog, CH 1 */
 DA_CH1_HI               =  $07;
 DI_LO                   =  $06;    //* Digit Input */
 DO_LO                   =  $0D;     //* Digit Output */

 CLEAR_IRQ               =  $08;
 SET_GAIN                =  $09;
 SET_CH                  =  $0A;
 SET_MODE                =  $0B;
 SOFT_TRIG               =  $0C;

 POLLING_MODE            =  1;
 DMA_MODE                =  2;
 INTERRUPT_MODE          =  6;

//*** define the gain mode ***/
 A826_BI_1               =  0;
 A826_BI_10              =  1;
 A826_BI_100             =  2;
 A826_BI_1000            =  3;
 A826_UNI_1              =  4;
 A826_UNI_10             =  5;
 A826_UNI_100            =  6;
 A826_UNI_1000           =  7;
 A826_BI_05              =  8;
 A826_BI_5               =  9;
```

```
 A826_BI_50             =  10;
 A826_BI_500            =  11;

 A826_BI_2              =  1;
 A826_BI_4              =  2;
 A826_BI_8              =  3;
 A826_UNI_2             =  5;
 A826_UNI_4             =  6;
 A826_UNI_8             =  7;

 NoError                =     0;
 DriverOpenError        =     1;
 DriverNoOpen           =     2;
 GetDriverVersionError  =     3;
 InstallIrqError        =     4;
 ClearIntCountError     =     5;
 GetIntCountError       =     6;
 AdError1               =   100;
 AdError2               = -200.0;
 InstallBufError        =     9;
 GetBufferError         =    10;
 INTStartError          =    11;
 INTStopError           =    12;
 InstallDmaIrqError     =    13;
 RemoveDmaIrqError      =    14;
 DmaStartError          =    15;
 DmaStopError           =    16;
 DmaGetDataError        =    17;
 TimeoutError           =    18;
 AllocateMemoryError    =    19;
 OtherError             =    20;

//********* Test Functios ************
Function  SHORT_SUB_2(nA, nB : SmallInt):SmallInt; StdCall;
Function  FLOAT_SUB_2(fA, fB : Single):Single; StdCall;
Function  A826_Get_DLL_Version:WORD; StdCall;
Function  A826_GetDriverVersion(var wDriverVersion:WORD):Word; StdCall;

//********* DI/DO Functios ************
Function  A826_DI(wBase:Word):Word; StdCall;
Procedure A826_DO(wBase, wHexValue:Word); StdCall;

//********* DA/AD Functios ************
Function  A826_AD(wBase,wChannel,wConfig:WORD):Single; StdCall;
Function  A826_ADs_Hex(wBase,wChannel,wConfig:WORD; wBuf:PSmallInt;
wCount:WORD):WORD; StdCall;
Function  A826_ADs_Float(wBase,wChannel,wConfig:WORD; fBuf:PSingle;
wCount:WORD):WORD; StdCall;
Procedure A826_DA(wBase, wChannel, wHexValue:WORD); StdCall;
Procedure A826_Uni5_DA(wBase,wChannel:Word;fValue:Single); StdCall;
Procedure A826_Uni10_DA(wBase,wChannel:Word;fValue:Single); StdCall;
```

```
//********* Driver Functios ************
Function  A826_DriverInit:WORD; StdCall;
Procedure A826_DriverClose; StdCall;
Function  A826_DELAY(wBase,wDownCount:WORD):WORD; StdCall;
Function  A826_Check_Address(wBase:WORD):WORD; StdCall;
Procedure A826_OutputByte(wPortAddr:WORD; bOutputVal:Byte); StdCall;
Procedure A826_OutputWord(wPortAddr:WORD; wOutputVal:WORD); StdCall;
Function  A826_InputByte(wPortAddr:WORD):WORD; StdCall;
Function  A826_InputWord(wPortAddr:WORD):WORD; StdCall;

//********* IRQ Functios ************
Function  A826_InstallIrq
          (wBase,wIrq:WORD; var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function  A826_AD_INT_Start(Ch,Gain,c1,c2:WORD):WORD; StdCall;
Function  A826_AD_INT_Stop:WORD; StdCall;
Function  A826_GetIntCount(var dwVal:LongInt):WORD; StdCall;
Function  A826_GetBuffer(dwNum:LongInt; wBuffer:PSmallInt):WORD; StdCall;
Function  A826_GetFloatBuffer(dwNum :LongInt;fBuffer:PSingle):Word; StdCall;

//********* DMA Functios ************
Function  A826_AD_DMA_InstallIrq(wBase,wIrq,wDmaChan:WORD):WORD; StdCall;
Function  A826_AD_DMA_RemoveIrq:WORD; StdCall;
Function  A826_AD_DMA_Start(Ch,Gain,c1,c2:WORD; cnt:integer; wPassOut:PInteger):WORD;
StdCall;
Function  A826_AD_DMA_Stop:WORD; StdCall;
Function  A826_AD_DMA_IsNotFinished:WORD; StdCall;
Function  A826_AD_DMA_GetBuffer(wBuf:PSmallInt):WORD; StdCall;
Function  A826_AD_DMA_GetFloatBuffer(fBuf:PSingle):Word; StdCall;

implementation

Function  SHORT_SUB_2;              external 'A826.DLL' name 'SHORT_SUB_2';
Function  FLOAT_SUB_2;              external 'A826.DLL' name 'FLOAT_SUB_2';
Function  A826_Get_DLL_Version;     external 'A826.DLL' name 'A826_Get_DLL_Version';
Procedure A826_DA;                  external 'A826.DLL' name 'A826_DA';
Procedure A826_Uni5_DA;             external 'A826.DLL' name 'A826_Uni5_DA';
Procedure A826_Uni10_DA;            external 'A826.DLL' name 'A826_Uni10_DA';
Procedure A826_DO;                  external 'A826.DLL' name 'A826_DO';
Function  A826_DI;                  external 'A826.DLL' name 'A826_DI';
Function  A826_AD;                  external 'A826.DLL' name 'A826_AD';
Function  A826_ADs_Hex;             external 'A826.DLL' name 'A826_ADs_Hex';
Function  A826_ADs_Float;           external 'A826.DLL' name 'A826_ADs_Float';
Function  A826_DELAY;               external 'A826.DLL' name 'A826_DELAY';
Function  A826_Check_Address;       external 'A826.DLL' name 'A826_Check_Address';

Function  A826_DriverInit;          external 'A826.DLL' name 'A826_DriverInit';
Procedure A826_DriverClose;         external 'A826.DLL' name 'A826_DriverClose';
Procedure A826_OutputByte;          external 'A826.DLL' name 'A826_OutputByte';
Procedure A826_OutputWord;          external 'A826.DLL' name 'A826_OutputWord';
Function  A826_InputByte;           external 'A826.DLL' name 'A826_InputByte';
Function  A826_InputWord;           external 'A826.DLL' name 'A826_InputWord';
```

```
Function  A826_GetDriverVersion;        external 'A826.DLL' name 'A826_GetDriverVersion';
Function  A826_InstallIrq;              external 'A826.DLL' name 'A826_InstallIrq';
Function  A826_GetBuffer;               external 'A826.DLL' name 'A826_GetBuffer';
Function  A826_GetFloatBuffer;          external 'A826.DLL' name 'A826_GetFloatBuffer';
Function  A826_GetIntCount;             external 'A826.DLL' name 'A826_GetIntCount';
Function  A826_AD_INT_Start;            external 'A826.DLL' name 'A826_AD_INT_Start';
Function  A826_AD_INT_Stop;             external 'A826.DLL' name 'A826_AD_INT_Stop';
Function  A826_AD_DMA_InstallIrq;       external 'A826.DLL' name 'A826_AD_DMA_InstallIrq';
Function  A826_AD_DMA_RemoveIrq;   external 'A826.DLL' name 'A826_AD_DMA_RemoveIrq';
Function  A826_AD_DMA_Start;            external 'A826.DLL' name 'A826_AD_DMA_Start';
Function  A826_AD_DMA_Stop;               external 'A826.DLL' name 'A826_AD_DMA_Stop';
Function  A826_AD_DMA_IsNotFinished;  external 'A826.DLL' name
'A826_AD_DMA_IsNotFinished';
Function  A826_AD_DMA_GetBuffer;         external 'A826.DLL' name 'A826_AD_DMA_GetBuffer';
Function  A826_AD_DMA_GetFloatBuffer; external 'A826.DLL' name
'A826_AD_DMA_GetFloatBuffer';


end.
```

2.3.2 A826.PAS (for Win NT)

```
unit A826;

interface

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

const

//***************** DEFINE A826 RELATIVE ADDRESS *****************/
 TIMER0                 =  $00;
 TIMER1                 =  $01;
 TIMER2                 =  $02;
 TIMER_MODE             =  $03;
 AD_LO                  =  $04;   //* Analog to Digital, Low Byte */
 AD_HI                  =  $05;   //* Analog to Digital, High Byte */
 DA_CH0_LO              =  $04;   //* Digit to Analog, CH 0 */
 DA_CH0_HI              =  $05;
 DA_CH1_LO              =  $06;   //* Digit to Analog, CH 1 */
 DA_CH1_HI              =  $07;
 DI_LO                  =  $06;   //* Digit Input */
 DO_LO                  =  $0D;    //* Digit Output */

 CLEAR_IRQ              =  $08;
 SET_GAIN               =  $09;
 SET_CH                 =  $0A;
 SET_MODE               =  $0B;
 SOFT_TRIG              =  $0C;

 POLLING_MODE       =  1;
 DMA_MODE           =  2;
 INTERRUPT_MODE     =  6;

//*** define the gain mode ***/
 A826_BI_1              =  0;
 A826_BI_10             =  1;
 A826_BI_100            =  2;
 A826_BI_1000           =  3;
 A826_UNI_1             =  4;
 A826_UNI_10            =  5;
 A826_UNI_100           =  6;
 A826_UNI_1000          =  7;
 A826_BI_05             =  8;
 A826_BI_5              =  9;
 A826_BI_50             =  10;
 A826_BI_500            =  11;

 A826_BI_2              =  1;
 A826_BI_4              =  2;
```

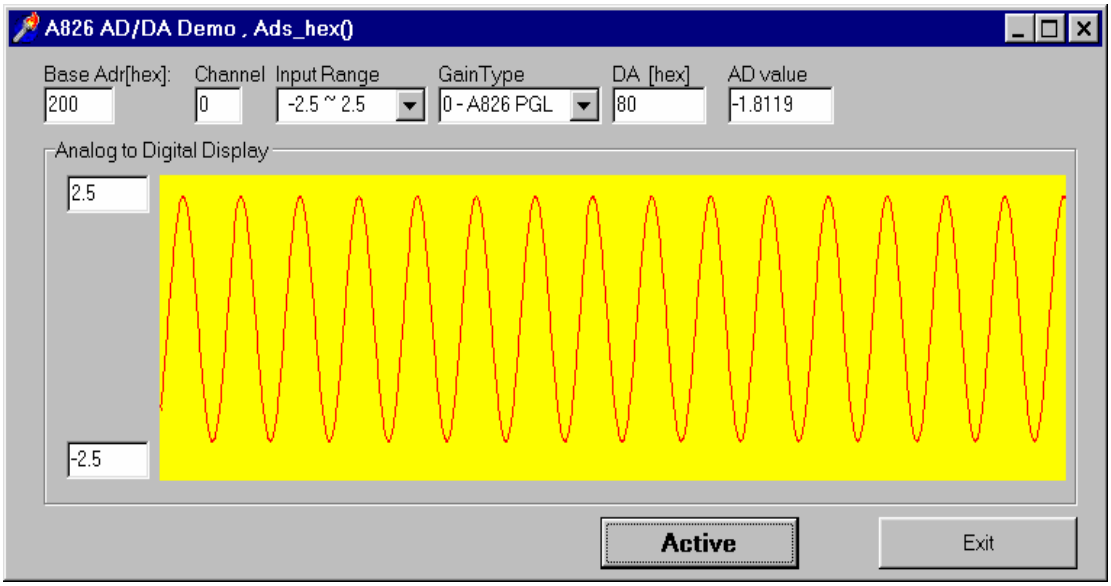
```
 A826_BI_8              =  3;
 A826_UNI_2             =  5;
 A826_UNI_4             =  6;
 A826_UNI_8             =  7;

 NoError                = 0;
 DriverOpenError        = 1;
 DriverNoOpen           = 2;
 GetDriverVersionError  = 3;
 InstallIrqError   = 4;
 ClearIntCountError     = 5;
 GetIntCountError       = 6;
 AdError1               = 100;
 AdError2               = -200.0;
 InstallBufError  = 9;
 AllocateMemoryError    = 10;
 CardTypeError          = 11;
 GetBufferError         = 12;
 TimeoutError           = 13;
 OtherError             = 14;

//********* Test Functios ************
Function  SHORT_SUB_2(nA, nB : SmallInt):SmallInt; StdCall;
Function  FLOAT_SUB_2(fA, fB : Single):Single; StdCall;
Function  A826_Get_DLL_Version:WORD; StdCall;
Function  A826_GetDriverVersion(var wDriverVersion:WORD):Word; StdCall;

//********* DI/DO Functios ************
Function  A826_DI(wBase:Word):Word; StdCall;
Procedure A826_DO(wBase, wHexValue:Word); StdCall;

//********* DA/AD Functios ************
Function  A826_AD(wBase,wChannel,wConfig:WORD):Single; StdCall;
Function  A826_ADs_Hex(wBase,wChannel,wConfig:WORD; wBuf:PSmallInt;
wCount:WORD):WORD; StdCall;
Function  A826_ADs_Float(wBase,wChannel,wConfig:WORD; fBuf:PSingle;
wCount:WORD):WORD; StdCall;
Procedure A826_DA(wBase, wChannel, wHexValue:WORD); StdCall;
Procedure A826_Uni5_DA(wBase,wChannel:Word;fValue:Single); StdCall;
Procedure A826_Uni10_DA(wBase,wChannel:Word;fValue:Single); StdCall;

//********* Driver Functios ************
Function  A826_DriverInit:WORD; StdCall;
Procedure A826_DriverClose; StdCall;
Function  A826_DELAY(wBase,wDownCount:WORD):WORD; StdCall;
Function  A826_Check_Address(wBase:WORD):WORD; StdCall;
Procedure A826_OutputByte(wPortAddr:WORD; bOutputVal:Byte); StdCall;
Procedure A826_OutputWord(wPortAddr:WORD; wOutputVal:WORD); StdCall;
Function  A826_InputByte(wPortAddr:WORD):WORD; StdCall;
Function  A826_InputWord(wPortAddr:WORD):WORD; StdCall;
```

```
//********* IRQ Functios ************
Function  A826_InstallIrq
          (wBase,wIrq:WORD; var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function  A826_AD_INT_Start(Ch,Gain,c1,c2:WORD):WORD; StdCall;
Function  A826_AD_INT_Stop:WORD; StdCall;
Function  A826_GetIntCount(var dwVal:LongInt):WORD; StdCall;
Function  A826_GetBuffer(dwNum:LongInt; wBuffer:PSmallInt):WORD; StdCall;
Function  A826_GetFloatBuffer(dwNum :LongInt;fBuffer:PSingle):Word; StdCall;


implementation

Function  SHORT_SUB_2;              external 'A826.DLL' name 'SHORT_SUB_2';
Function  FLOAT_SUB_2;              external 'A826.DLL' name 'FLOAT_SUB_2';
Function  A826_Get_DLL_Version;     external 'A826.DLL' name 'A826_Get_DLL_Version';
Procedure A826_DA;                  external 'A826.DLL' name 'A826_DA';
Procedure A826_Uni5_DA;             external 'A826.DLL' name 'A826_Uni5_DA';
Procedure A826_Uni10_DA;                    external 'A826.DLL' name 'A826_Uni10_DA';
Procedure A826_DO;                  external 'A826.DLL' name 'A826_DO';
Function  A826_DI;                  external 'A826.DLL' name 'A826_DI';
Function  A826_AD;                  external 'A826.DLL' name 'A826_AD';
Function  A826_ADs_Hex;             external 'A826.DLL' name 'A826_ADs_Hex';
Function  A826_ADs_Float;           external 'A826.DLL' name 'A826_ADs_Float';
Function  A826_DELAY;               external 'A826.DLL' name 'A826_DELAY';
Function  A826_Check_Address;       external 'A826.DLL' name 'A826_Check_Address';

Function  A826_DriverInit;          external 'A826.DLL' name 'A826_DriverInit';
Procedure A826_DriverClose;         external 'A826.DLL' name 'A826_DriverClose';
Procedure A826_OutputByte;          external 'A826.DLL' name 'A826_OutputByte';
Procedure A826_OutputWord;          external 'A826.DLL' name 'A826_OutputWord';
Function  A826_InputByte;           external 'A826.DLL' name 'A826_InputByte';
Function  A826_InputWord;           external 'A826.DLL' name 'A826_InputWord';

Function  A826_GetDriverVersion;    external 'A826.DLL' name 'A826_GetDriverVersion';
Function  A826_InstallIrq;          external 'A826.DLL' name 'A826_InstallIrq';
Function  A826_GetBuffer;           external 'A826.DLL' name 'A826_GetBuffer';
Function  A826_GetFloatBuffer;      external 'A826.DLL' name 'A826_GetFloatBuffer';
Function  A826_GetIntCount;         external 'A826.DLL' name 'A826_GetIntCount';
Function  A826_AD_INT_Start;        external 'A826.DLL' name 'A826_AD_INT_Start';
Function  A826_AD_INT_Stop;         external 'A826.DLL' name 'A826_AD_INT_Stop';


end.
```

2.3.3 A826U.PAS

```
unit A826U;

interface

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

Function  A826_AD2F(hex, Gain :Word): Single ; StdCall;

implementation

uses math;


//*-----------------------------------------------------*
//* Return voltage value or -100.0 if any error occurs  *
//* or parameter is out of range.                       *
//* Gain : 0-3                                                  *
//*-----------------------------------------------------*
Function  A826_AD2F(hex, Gain :Word): Single ;
Var
   ZeroBase, VoltageRange, FullRange : Single ;
   i : Integer ;

Begin

   ZeroBase  := 0;
   FullRange := 32767;
   VoltageRange := 10;

   i := hex;
   if i > $7FFF then
         i := ((Not hex) + 1) * -1 ;

   Gain := Gain mod 16 ;

   If (Gain < 0) Or (Gain > 3) Then
   begin
      result := -100;
      exit;
   end;

   Result := ((((i - ZeroBase) / FullRange) * VoltageRange) / Power(2, Gain));

End;

end.
```

2.3.4 Delphi Demo Result :



3 Function Description

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

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 DMA Mode functions

The functions of test listing as follows:

1. SHORT_SUB_2
2. FLOAT_SUB_2
3. A826_Get_DLL_Version
4. A826_GetDriverVersion

The functions of DI/O listing as follows:

1. A826_DI
2. A826_DO
3. A826_InputByte
4. A826_InputWord
5. A826_OutputByte
6. A826_OutputWord

The functions of AD/DA listing as follows:

1. A826_DA
2. A826_AD
3. A826_ADs_Hex
4. A826_ADs_Float

The functions of Driver listing as follows:

1. A826_DriverInit
2. A826_DriverClose
3. A826_DELAY
4. A826_Check_Address

The functions of AD Interrupt listing as follows:

1. A826_InstallIrq
2. A826_GetIntCount
3. A826_GetBuffer
4. A826_AD_INT_Start
5. A826_AD_INT_Stop
6. A826_GetFloatBuffer

The functions of AD DMA listing as follows:

1. A826_AD_DMA_InstallIrq
2. A826_AD_DMA_RemoveIrq
3. A826_AD_DMA_Start
4. A826_AD_DMA_Stop
5. A826_AD_DMA_IsNotFinished
6. A826_AD_DMA_GetBuffer
7. A826_AD_DMA_GetFloatBuffer

(The DMA function supports the Windows 95/98 only.)

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

Keyword	Setting parameter by user before calling this function ?	Get the data/value from this parameter after calling this function ?
[Input]	Yes	No
[Output]	No	Yes
[Input, Output]	Yes	Yes

3.1 TEST Function

3.1.1 SHORT_SUB_2

- **Description :**
  Compute C=nA-nB in **short** format, **Short=16 bits sign integer.** This function is provided for testing purpose.

- **Syntax :**
  short SHORT_SUB_2(short nA, short nB);

- **Parameter :**
  nA : [Input] Short
  nB : [Input] Short

- **Return Value :**
  Return = nA - nB ➔ Short

3.1.2 FLOAT_SUB_2

- **Description :**
  Compute C=fA-fB in **float** format, **float=32 bits floating pointer number.** This function is provided for testing purpose.

- **Syntax :**
  float FLOAT_SUB_2(float fA, float fB)

- **Parameter :**
  fA : [Input] Double value
  fB : [Input] Double value

- **Return Value :**
  return= fA - fB ➔ float value

3.1.3 A826_Get_DLL_Version

- **Description :**
  Read the software version

- **Syntax :**
  WORD A826_Get_DLL_Version(void);

- **Parameter :**
  Void

- **Return Value :**
  return=0x100 ➔ Version 1.0

3.1.4 A826_GetDriverVersion

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

- **Syntax :**
  WORD A826_GetDriverVersion (WORD *wDriverVersion);

- **Parameter :**
  *wDriverVersion : [Output] the address of wDriverVersion.
  When wDriverVersion=0x210 → version 2.10

- **Return Value :**
  NoError : successful in opening the device driver
  GetDriverVersionError : fail in opening the device driver.

3.2 DI/DO Function

3.2.1 A826_Di

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

- **Syntax :**
  WORD A826_DI(WORD wBase);

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

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

3.2.2 A826_Do

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

- **Syntax :**
  void A826_DO(WORD wBase, WORD wHexValue);

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

- **Return Value :**
  Void

3.2.3 A826_OutputByte

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

- **Syntax :**
  void A826_OutputByte(WORD wPortAddr, UCHAR bOutputVal);

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

- **Return Value :**
  void

3.2.4 A826_OutputWord

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

- **Syntax :**
  void A826_OutputByte(WORD wPortAddr, WORD wOutputVal);

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

- **Return Value :**
  void

3.2.5 A826_InputByte

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

- **Syntax :**
  WORD A826_InputByte(WORD wPortAddr);

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

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

3.2.6 A826_InputWord

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

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

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

- **Return Value :**
  16 bits data.

3.3 A/D , D/A Functions

3.3.1 A826_AD

- **Description :**
  This subroutine will perform a A/D conversion by polling. The A/D converter is 16 bits for A82PG. This subroutine will compute the result according to the **configuration code**.

- **Syntax :**
  float A826_AD(WORD wBase, WORD wChannel, WORD wConfig);
- **Parameter :**
  wBase : [Input] I/O port base address, for example, 0x220
  wChannel : [Input] A/D channel number,
  wConfig : [Input] Configuration code, refer to 1.2 for detail information
- **Return Value :**
  AdError2 : A/D converter error
  Other value : The **floating point value** of A/D conversion

3.3.2 A826_DA

- **Description :**
  This subroutine will send the 16 bits data to D/A analog output. The output range of D/A maybe 0 to 5V or 0 to10V **setting by hardware jumper, JP1.** The software **can not 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 A826_DA(WORD wBase, WORD wChannel, WORD wHexValue);
- **Parameter :**
  wBase : [Input] I/O port base address, for example, 0x220
  wChannel : [Input] D/A channel number, validate for 0
  wHexValue : [Input] 16 bits data send to D/A converter
- **Return Value :**
  NoError : no error
  DaChannelError : wChannel value error, validate only for 0

3.3.3 A826_ADs_Hex

- **Description :**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A826_AD except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D conversing at the ISA bus's max. speed. The sampling rate is about 90K samples/second which testing under Pentium-133 CPU. After A/D conversing, the A/D data are stored in a buffer in Hex format. The **wBuf** is the starting address of this data buffer.

- **Syntax :**

```
WORD A826_ADs_Hex(WORD wBase, WORD wChannel, WORD
                  wConfig,short wBuf[], WORD wCount);
```

- **Parameter :**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] A/D channel number
wConfig	: [Input] Configuration code, please refer to Section 1.2 for detail information
wBuf	: [Output] Starting address of the data buffer
wCount	: [Input] Number of A/D conversions will be performed

- **Return Value :**

AdError1	: A/D converter error
NoError	: Operation is OK

3.3.4 A826_ADs_Float

- **Description :**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A826_AD except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D conversing at the ISA bus's max. speed. The sampling rate is about 90K samples/second which testing 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 :**

WORD A826_ADs_Float(WORD wBase, WORD wChannel, WORD wConfig, float fBuf[], WORD wCount);

- **Parameter :**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] A/D channel number
wConfig	: [Input] Configuration code, refer to 1.2 for detail information
fBuf	: [Output] Starting address of the data buffer **(in float format)**
wCount	: [Input] Number of A/D conversions will be performed

- **Return Value :**

AdError1	: A/D converter error
NoError	: Operation is OK

3.3.5 A826_Uni5_DA

- **Description :**

This subroutine will send the 16 bits data to D/A analog output. The output range of D/A dependent on **setting by hardware jumper, JP1 (-5v or –10v), JP10/JP11 (Bipolar or Unipolar).** The software **can not detect** the output range of D/A converter. This subroutine can be used only when the jumpers settings are **: Unipolar , -5v .** The **output range is between 0.0v and 5.0v.** Please refer to hardware manual to setting jumpers.

- **Syntax :**

void A826_Uni5_DA(WORD wBase, WORD wChannel, float fValue);

- **Parameter :**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] D/A channel number, validate for 0
fValue	: [Input] 16 bits data send to D/A converter

- **Return Value :**

void

3.3.6 A826_Uni10_DA

- **Description :**

This subroutine will send the 16 bits data to D/A analog output. The output range of D/A dependent on **setting by hardware jumper, JP1 (-5v or –10v), JP10/JP11 (Bipolar or Unipolar).** The software **can not detect** the output range of D/A converter. This subroutine can be used only when the jumpers settings are**: Unipolar , -10v.** The **output range is between 0.0v and 10.0v.** Please refer to hardware manual to setting jumpers.

- **Syntax :**

void A826_Uni10_DA(WORD wBase, WORD wChannel, float fValue);

- **Parameter :**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] D/A channel number, validate for 0
fValue	: [Input] 16 bits data send to D/A converter

- **Return Value :**

void

3.4 Driver Functions

3.4.1 A826_DriverInit

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

- **Syntax :**
  WORD A826_DriverInit (void);

- **Parameter :**
  Void

- **Return Value :**
  NoError : successful in opening the device driver
  DriverOpenError : fail in opening the device driver.

3.4.2 A826_DriverClose

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

- **Syntax:**
  void A826_DriverClose(void);

- **Parameter:**
  void

- **Return Value:**
  void

3.4.3 A826_DELAY

- **Description:**
  This subroutine will delay **wDownCount** mS(machine independent timer).

- **Syntax:**
  WORD A826_DELAY(WORD wBase, WORD wDownCount);

- **Parameter:**
  wBase : [Input] I/O port base address, for example, 0x220
  wDownCount : [Input] Number of mS will be delay

- **Return Value:**
  NoError : Operation OK
  AdError1 : Operation failure

3.4.4 A826_Check_Address

- **Description:**
  This subroutine will detect the A-826PG in I/O base address = **wBase**. This subroutine will perform one A/D conversion, if success → find a A-826PG.

- **Syntax:**
  WORD A826_Check_Address(WORD wBase);

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

- **Return Value:**
  NoError : Find a A-826PG OK
  AdError1 : Operation failure

3.5 AD, Interrupt functions

3.5.1 A826_InstallIrq

- **Description:**
  This subroutine will install interrupt handler for a specific IRQ level n.

- **Syntax:**
  WORD A826_InstallIrq(WORD wBase, WORD wIrq,
  HANDLE *hEvent, DWORD dwCount);

- **Parameter:**
  wBase : [Input] The I/O port base address for A826 card.
  wIrq : [Input] The IRQ level .
  hEvent : [Input] The handle of event object that created by user.
  dwCount : [Input] The desired A/D entries count for interrupt transfer.

- **Return Value:**
  NoError : successful
  InstallIrqError: fail in install IRQ handler.

3.5.2 A826_AD_INT_Start

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

- **Syntax:**
  WORD A826_AD_INT_Start(WORD Ch, WORD Gain, WORD c1, Word c2)

- **Parameter:**
  Ch : [Input] the A/D channel.
  Gain : [Input] the Gain
  c1,c2 : [Input] the sampling rate is 2M/(c1*c2)

- **Return Value:**
  NoError : successful
  INTStartError : fail

3.5.3 A826_AD_INT_Stop

- **Description:**
  This subroutine will stop the interrupt transfer and remove the installed interrupt handler.

- **Syntax:**
  WORD A826_AD_INT_Stop(void)

- **Parameter:**
  void.

- **Return Value:**
  NoError : successful
  INTStopError : fail

3.5.4 A826_GetIntCount

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

- **Syntax:**
  WORD A826_GetIntCount(DWORD *dwVal)

- **Parameter:**
  *dwVal : [Output] the address of dwVal,
  the dwVal is the interrupt transferred count.

- **Return Value:**
  NoError : successful
  GetIntCountError : fail get interrupt count.

3.5.5 A826_GetBuffer

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

- **Syntax:**
  WORD A826_GetBuffer(DWORD dwNum, short wBuffer[])

- **Parameter:**

dwNum	: [Input]	the entry no to transfer.
*wBuffer	: [Output]	the address of wBuffer,

- **Return Value:**

NoError	: successful
GetBufferError	: fail

3.5.6 A826_GetFloatBuffer

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

- **Syntax:**
  WORD A826_GetFloatBuffer(DWORD dwNum, float fBuffer[])

- **Parameter:**

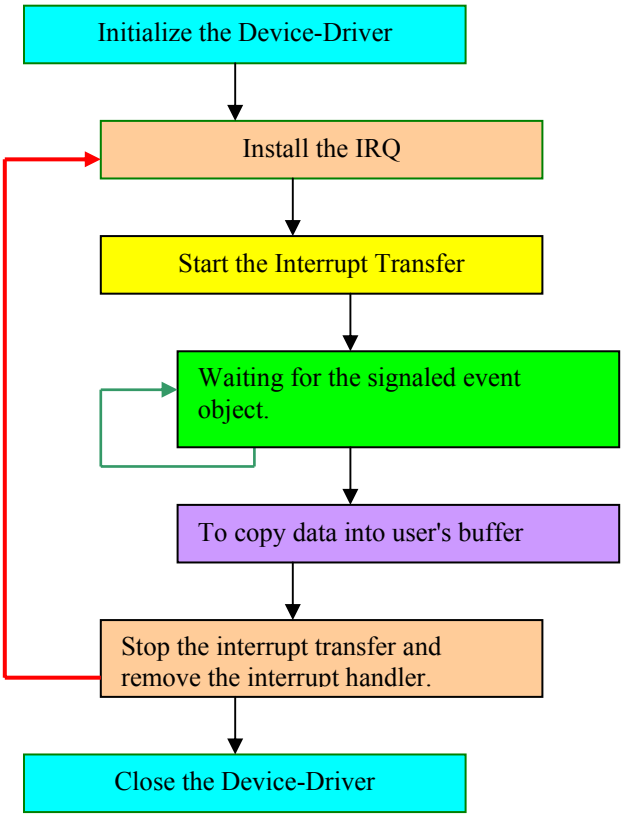
dwNum	: [Input]	the entry no to transfer.
*fBuffer	: [Output]	the address of fBuffer,

- **Return Value:**

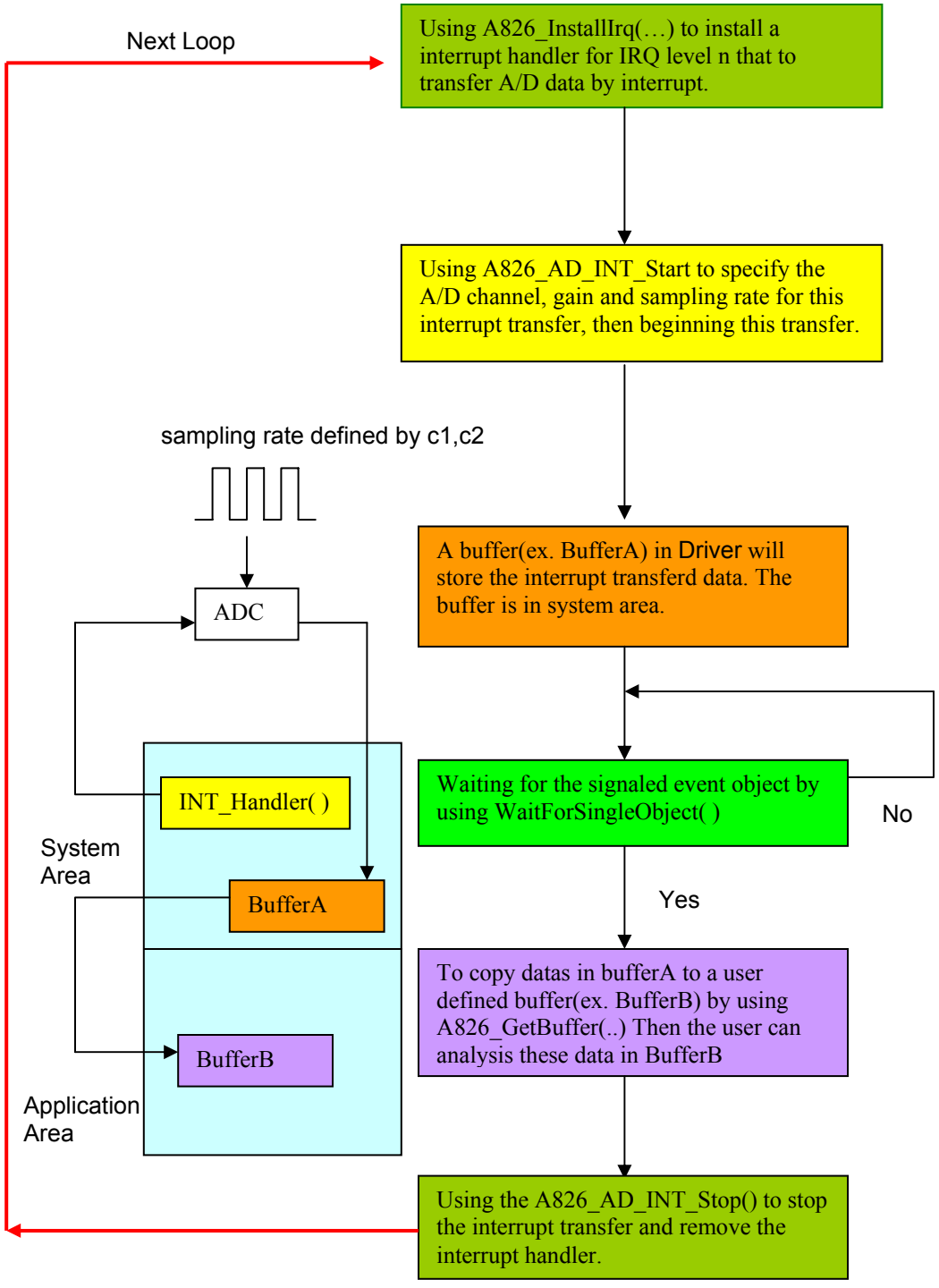
NoError	: successful
GetBufferError	: fail

3.5.7 Diagram of Interrupt Mode

The 3.5.1 to 3.5.6 are functions to perform the A/D conversion with interrupt transfer. The flow chart to program these function is giver as follows:




```
A826_DriverInit
    CreateEvent(…)
    A826_InstallIrq( … )
    ……
    A826_AD_INT_Start( … )
    ……
        WaitForSingleObject( …. )
        ……..
        A826_GetBuffer( …. )
    ……
    A826_AD_INT_Stop
    CloseHandle( hEvent )
A826_DriverClose
```


Next Loop
Using A826_InstallIrq(…) to install a interrupt handler for IRQ level n that to transfer A/D data by interrupt.
Using A826_AD_INT_Start to specify the A/D channel, gain and sampling rate for this interrupt transfer, then beginning this transfer.
sampling rate defined by c1,c2
A buffer(ex. BufferA) in Driver will store the interrupt transferd data. The buffer is in system area.
ADC
INT_Handler( )
System Area
BufferA
Waiting for the signaled event object by using WaitForSingleObject( )
No
Yes
To copy datas in bufferA to a user defined buffer(ex. BufferB) by using A826_GetBuffer(..) Then the user can analysis these data in BufferB
BufferB
Application Area
Using the A826_AD_INT_Stop() to stop the interrupt transfer and remove the interrupt handler.

3.6 AD , DMA functions

The DMA mechanism supports the Windows 95/98 only.

3.6.1 A826_AD_DMA_InstallIrq

- **Description:**
  This subroutine will install interrupt handler for a specific IRQ Level n and programming a DMA controller (8237) to handle DMA transfer for DMA Channel n. Usually, when a DMA transfer finished, a associated IRQ level n occur.

- **Syntax:**
  WORD A826_AD_DMA_InstallIrq(WORD wBase, WORD wIrq, WORD wDmaChannel);

- **Parameter:**
  wBase : [Input] the I/O port base address for A826 card.
  wIrq : [Input] the IRQ level n.
  wDmaChannel : [Input] the DMA channel.
  Usuallly, the wDmaChannel is 1 or 3.

- **Return Value:**
  NoError : successful in installing the handler
  InstallDMAIrqError : failure

3.6.2 A826_AD_DMA_RemoveIrq

- **Description:**
  This subroutine will remove the interrupt handler installed by. A826_AD_DMA_InstallIrq(…).

- **Syntax:**
  WORD A826_AD_DMA_RemoveIrq(void)

- **Parameter:**
  void

- **Return Value:**
  NoError : successful
  RemoveDmaIrqError : failure

3.6.3 A826_AD_DMA_Start

- **Description:**

This subroutine will allocate a DMA buffer in the system area, and programming the gain code and sampling rate. then starting the DMA transfer for a specific A/D channel.

- **Syntax:**

WORD A826_AD_DMA_Start(WORD Ch, WORD Gain, WORD c1, Word c2, int Count, WORD wPassOut[ ])

- **Parameter:**

Ch : [Input] the A/D channel.
Gain : [Input] the Gain code
c1,c2 : [Input] the DMA sampling rate is 2M/(c1*c2)
Count : [Input] the desired A/D entries count for DMA transfer.
wPassOut[] : [Output] the debug information.
wPassOut[0] : [Output] 0: successful in starting DMA transfer.
others: fail in starting DMA transfer.
wPassOut[1] : [Output] system DMA buffer ID.
wPassOut[2] : [Output] the I/O port base address.
wPassOut[3] : [Output] the IRQ level for DMA.
wPassOut[4] : [Output] the DMA channel no.
wPassOut[5] : [Output] reserved.
wPassOut[6] : [Output] reserved.
wPassOut[7] : [Output] reserved.
wPassOut[8] : [Output] the last 16 bits of physical address for DMA buffer in system area.
wPassOut[9] : [Output] the first 16 bits of physical address for DMA buffer in system area.

- **Return Value:**

NoError : successful
DmaStartError : failure

3.6.4 A826_AD_DMA_Stop

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

- **Syntax:**
  WORD A826_AD_DMA_Stop(void)

- **Parameter:**
  void.

- **Return Value:**
  NoError : successful
  DmaStopError : fail

3.6.5 A826_AD_DMA_IsNotFinished

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

- **Syntax:**
  WORD A826_AD_DMA_IsNotFinished(void)

- **Parameter:**
  void.

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

3.6.6 A826_AD_DMA_GetBuffer

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

- **Syntax:**
  WORD A826_AD_DMA_GetBuffer(short wBuffer[])

- **Parameter:**
  *wBuffer : [Output] the address of wBuffer,

- **Return Value:**
  The returned bytes no : when successful
  DmaGetDataError : failure

3.6.7 A826_AD_DMA_GetFloatBuffer

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

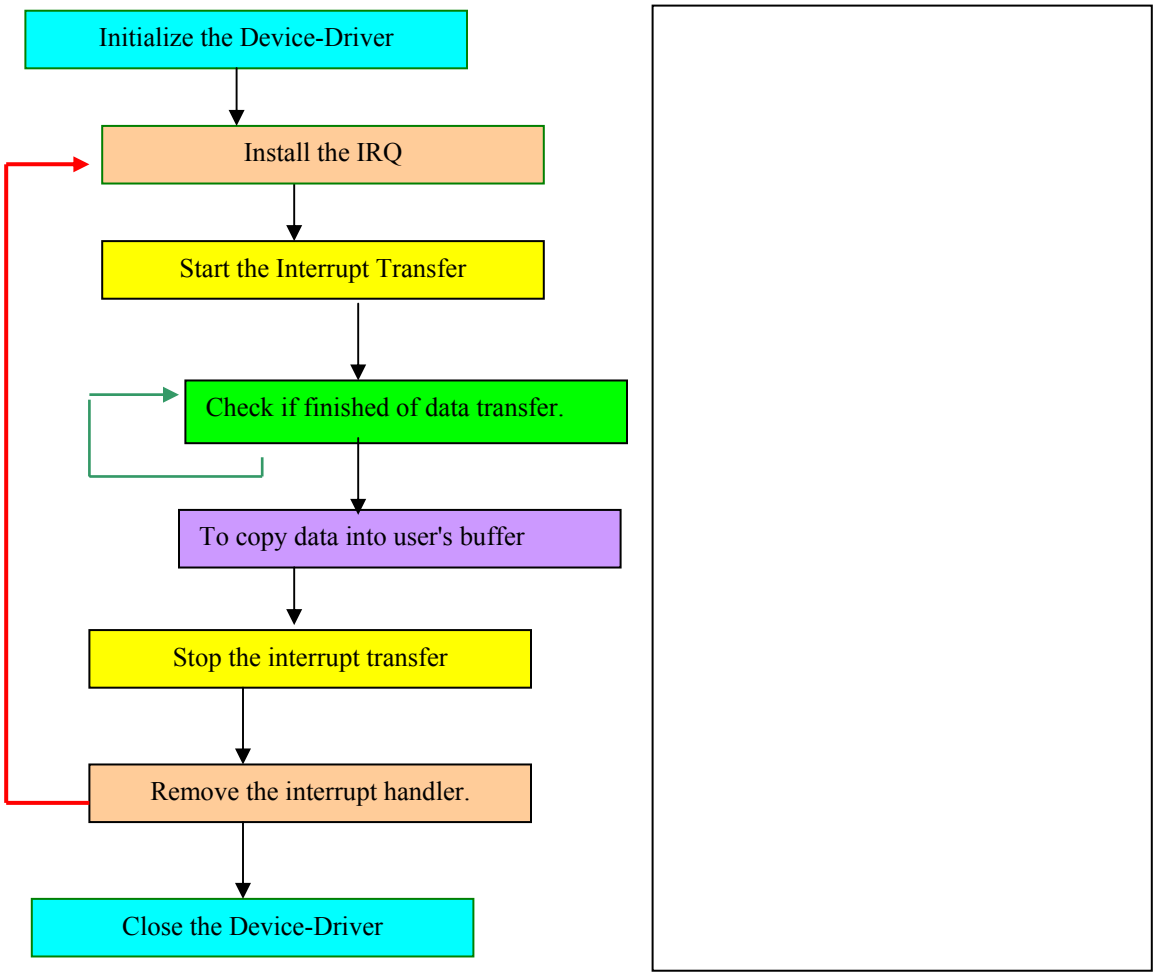
- **Syntax:**
  WORD A826_AD_DMA_GetfloatBuffer(float fBuf[])

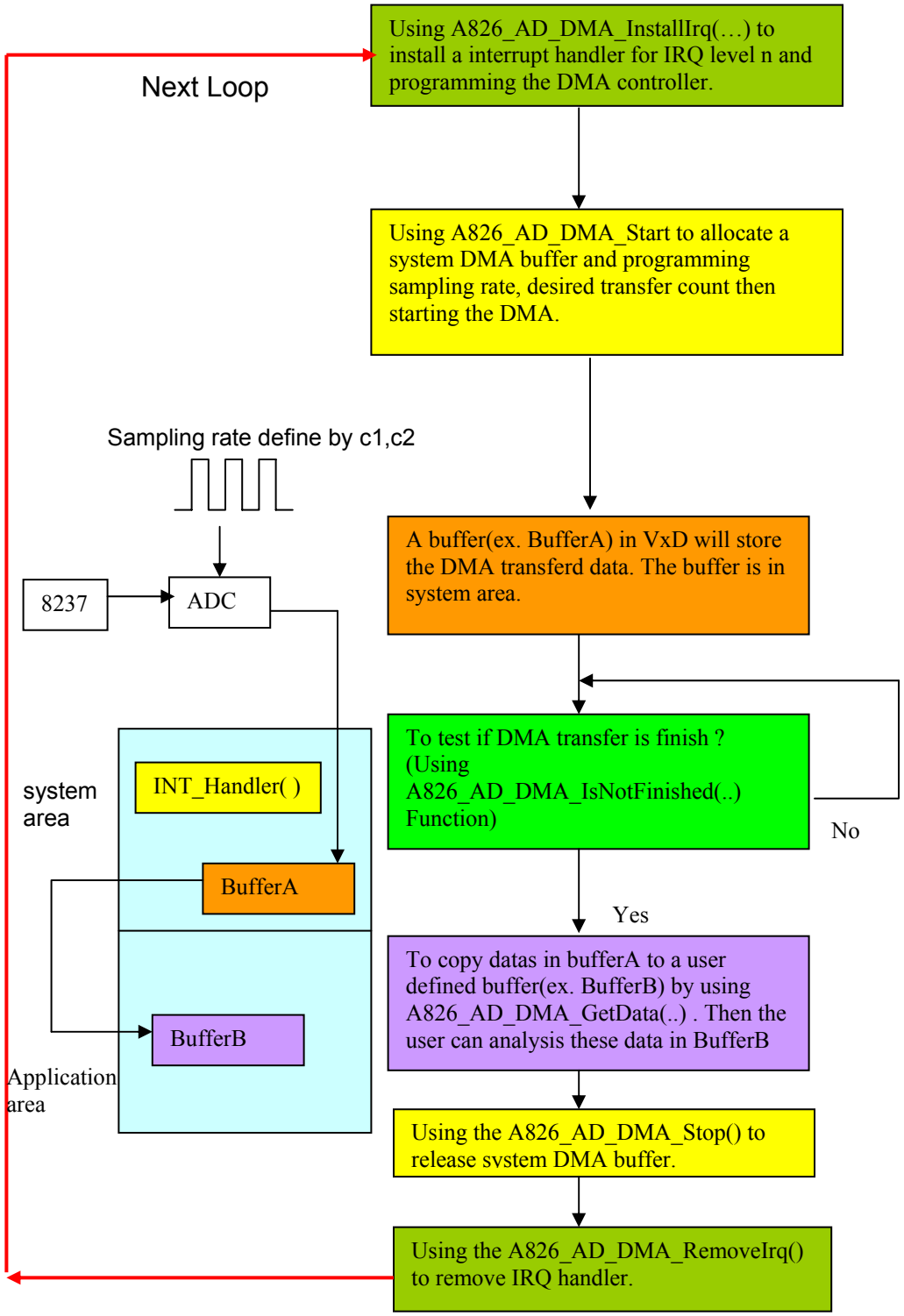
- **Parameter:**
  *fBuf : [Output] the address of fBuf

- **Return Value:**
  The returned bytes no : when successful
  DmaGetDataError : failure

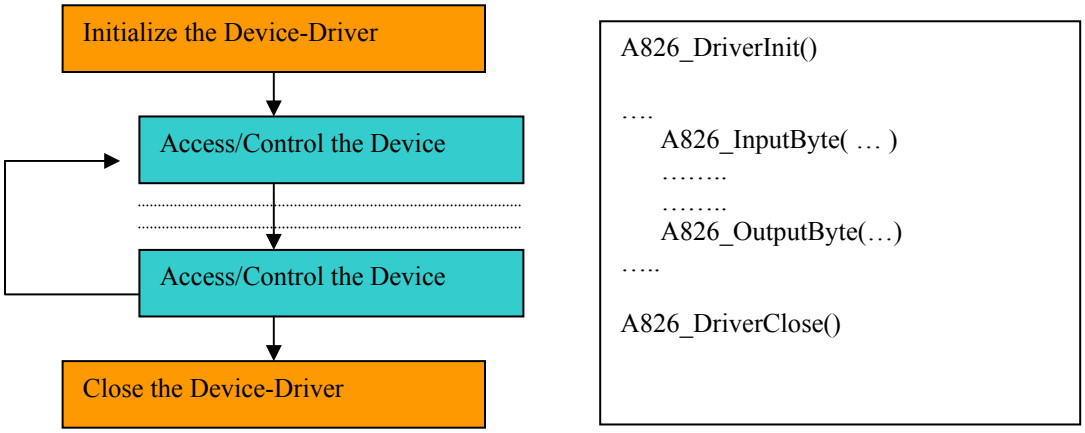
3.6.8 Diagram of AD , DMA Mode

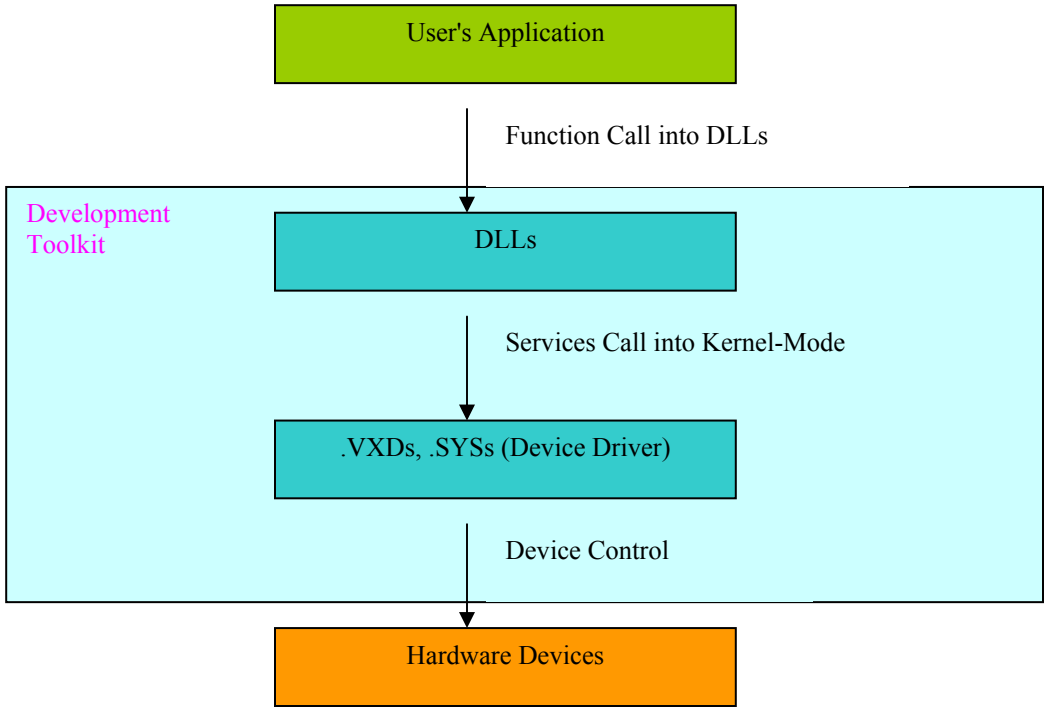
The 3.6.1 to 3.6.7 are functions to perform the A/D conversion with DMA transfer. The flow chart to program these function is giver as follows:




Using A826_AD_DMA_InstallIrq(…) to install a interrupt handler for IRQ level n and programming the DMA controller.
Next Loop
Using A826_AD_DMA_Start to allocate a system DMA buffer and programming sampling rate, desired transfer count then starting the DMA.
Sampling rate define by c1,c2
A buffer(ex. BufferA) in VxD will store the DMA transferd data. The buffer is in system area.
8237
ADC
To test if DMA transfer is finish ? (Using A826_AD_DMA_IsNotFinished(..) Function)
No
system area
INT_Handler( )
BufferA
Yes
To copy datas in bufferA to a user defined buffer(ex. BufferB) by using A826_AD_DMA_GetData(..) . Then the user can analysis these data in BufferB
BufferB
Application area
Using the A826_AD_DMA_Stop() to release system DMA buffer.
Using the A826_AD_DMA_RemoveIrq() to remove IRQ handler.

4 Program Architecture





5 Contact Us

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.

© 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

M4035/0104