A-812 PG

Software Manual

[ver. 1.9, JAN 2013]

Warranty

All products manufactured by ICP DAS are warranted against defective materials for a period of one year from the date of delivery to the original purchaser.

Warning

ICP DAS assumes no liability for damages consequent to the use of this product. ICP DAS reserves the right to change this manual at any time without notice. The information furnished by ICP DAS is believed to be accurate and reliable. However, no responsibility is assumed by ICP DAS for its use, nor for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Trademark

Names are used for identification only and may be registered trademarks of their respective companies.

Table of Contents

1. Declaration Files

Please refer to the "Calling_DLL_functions_in_VB_VB_Delphi_BCB.pdf" user manual.

For Windows 2000:

```
|--\ Driver
    |--\ A812.DLL          ← Dynamic Linking Library
    |--\ A812.sys          ← Device driver
    |--\ Napwnt.sys        ← Device driver
    |
    |--\ BCB               ← For Borland C++ Builder
    |    |--\ A812.H       ← Header File
    |    |--\ A812.Lib     ← Import Library for BCB only
    |
    |--\ Delphi            ← For Delphi
    |    |--\ A812.pas     ← Declaration File
    |
    |--\ VB                ← For Visual Basic
    |    |--\ A812.bas     ← Declaration File
    |
    |--\ VC                ← For Visual Basic
    |    |--\ A812.H       ← Header File
    |    |--\ A812.Lib     ← Import Library for BCB only
```

1.1 A812.H

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

/***************** DEFINE A812 RELATIVE ADDRESS ****************/

#define A812_TIMER0             0x00
#define A812_TIMER1             0x01
#define A812_TIMER2             0x02
#define A812_TIMER_MODE         0x03
#define A812_AD_LO              0x04 /* Analog to Digital, Low Byte */
#define A812_AD_HI              0x05 /* Analog to Digital, High Byte */
#define A812_DA_CH0_LO          0x04 /* Digital to Analog, CH0 */
#define A812_DA_CH0_HI          0x05
#define A812_DA_CH1_LO          0x06 /* Digital to Analog, CH1 */
#define A812_DA_CH1_HI          0x07
#define A812_DI_LO              0x06 /* Digital Input */
#define A812_DO_LO              0x0D /* Digital Output */

#define A812_CLEAR_IRQ          0x08
#define A812_SET_GAIN           0x09
#define A812_SET_CH             0x0A
#define A812_SET_MODE           0x0B
#define A812_SOFT_TRIG          0x0C

#define A812_POLLING_MODE       1
#define A812_DMA_MODE           2
#define A812_INTERRUPT_MODE     6

/*** Define the gain mode ***/

#define A812_BI_1               0
#define A812_BI_2               1
#define A812_BI_4               2
#define A812_BI_8               3
#define A812_BI_16              4

#define A812_NoError            0
#define A812_DriverOpenError    1
#define A812_DriverNoOpen       2
#define A812_GetDriverVersionError  3
```

```
#define A812_GetTotalBoardsError        4
#define A812_BoardNotFound              5
#define A812_ActiveBoardError           6
#define A812_ExceedBoardNo              7
#define A812_GetConfigError             8
#define A812_AllocateMemoryError        9
#define A812_TimeoutError               10

#define A812_InvalidGainCode            11
#define A812_InvalidJump10v             12
#define A812_InvalidChannelNo           13

#define A812_IntSetEventError           14
#define A812_IntInstallError            15
#define A812_IntClearCountError         16
#define A812_IntGetCountError           17
#define A812_IntGetBufferError          18
#define A812_IntRemoveError             19

#define A812_ChScanBufferFull           20
#define A812_ChScanNoChannelToScan      21
#define A812_ChScanIntSetChannelsError  22
#define A812_ChScanIntSetConfigsError   23

// Test Functions
EXPORTS short CALLBACK A812_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK A812_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK A812_Get_DLL_Version(void);
EXPORTS WORD CALLBACK A812_GetDriverVersion
(WORD *wDriverVersion);
```

```
// DI/DO Functions
EXPORTS WORD CALLBACK A812_DI(WORD *wInVal);
EXPORTS WORD CALLBACK A812_DO(WORD wHexValue);
EXPORTS void CALLBACK A812_OutputByte
(WORD wPortAddr, UCHAR bOutputVal);
EXPORTS void CALLBACK A812_OutputWord
(WORD wPortAddr, WORD wOutputVal);
EXPORTS WORD CALLBACK A812_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK A812_InputWord(WORD wPortAddr);

// AD Functions
EXPORTS WORD CALLBACK A812_Fast_SetChGain
(WORD wChannel, WORD wGainCode, WORD wJump10v);
EXPORTS WORD CALLBACK A812_Fast_AD_Hex(WORD *wVal);
EXPORTS WORD CALLBACK A812_Fast_AD_Float(float *fVal);
EXPORTS WORD CALLBACK A812_AD_Hex
(WORD wChannel, WORD wGainCode, WORD *wVal);
EXPORTS WORD CALLBACK A812_ADs_Hex
(WORD wChannel, WORD wGainCode, WORD wBuf[], WORD wCount);
EXPORTS WORD CALLBACK A812_AD_Float
(WORD wChannel, WORD wGainCode, WORD wJump10v, float *fVal);
EXPORTS WORD CALLBACK A812_ADs_Float
(WORD wChannel, WORD wGainCode, WORD wJump10v, float fBuf[],
WORD wCount);
EXPORTS WORD CALLBACK A812_Hex2Float
(WORD wHex, WORD wGainCode, WORD wJump10v, float *fVal);

// DA Functions
EXPORTS WORD CALLBACK A812_DA_Hex(WORD wChannel, WORD
wHexValue);
EXPORTS WORD CALLBACK A812_DA_Uni5(WORD wChannel, float
fValue);
EXPORTS WORD CALLBACK A812_DA_Uni10(WORD wChannel, float
fValue);
```

```
// Driver Functions
EXPORTS WORD CALLBACK A812_DriverInit(WORD *wTotalBoards);
EXPORTS void CALLBACK A812_DriverClose(void);
EXPORTS WORD CALLBACK A812_DELAY(WORD wDownCount);
EXPORTS WORD CALLBACK A812_Check_Address(void);
EXPORTS WORD CALLBACK A812_GetConfigAddress
(WORD *wAddrBase, WORD *wCurrentBoard);
EXPORTS WORD CALLBACK A812_ActiveBoard( WORD wBoardNo );

// Interrupt Functions
EXPORTS WORD CALLBACK A812_IntInstall
(HANDLE *hEvent, DWORD dwDataCount);
EXPORTS WORD CALLBACK A812_IntStart
(WORD wChannel, WORD wGainCode, WORD wJump10v, WORD
wCounter1, WORD wCounter2);
EXPORTS WORD CALLBACK A812_IntGetCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A812_IntGetHexBuf
(DWORD dwNum, WORD wBuf[]);
EXPORTS WORD CALLBACK A812_IntGetFloatBuf
(DWORD dwNum, float fBuf[]);
EXPORTS WORD CALLBACK A812_IntStop(void);
EXPORTS WORD CALLBACK A812_IntRemove(void);

// AD Channel-Scan Functions
EXPORTS void CALLBACK A812_ChScan_Clear(void);
EXPORTS WORD CALLBACK A812_ChScan_Add
(WORD wChannel, WORD wConfig);
EXPORTS WORD CALLBACK A812_ChScan_Set
(WORD wChannel[], WORD wConfig[], WORD wChNum);
EXPORTS WORD CALLBACK A812_ChScan_PollingHex
(WORD wBuf[], WORD wNumPerCh);
EXPORTS WORD CALLBACK A812_ChScan_PollingFloat
(WORD wJump10v, float fBuf[], WORD wNumPerCh);
```

```
// AD Channel-Scan with Interrupt Functions
EXPORTS WORD CALLBACK A812_ChScan_IntInstall
(HANDLE *hEvent, DWORD dwNumPerCh);
EXPORTS WORD CALLBACK A812_ChScan_IntStart
(WORD wCounter1, WORD wCounter2, WORD wJump10v);
EXPORTS WORD CALLBACK A812_ChScan_IntGetHexBuf(WORD wBuf[]);
EXPORTS WORD CALLBACK A812_ChScan_IntGetFloatBuf(float fBuf[]);
EXPORTS WORD CALLBACK A812_ChScan_IntGetCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A812_ChScan_IntStop(void);
EXPORTS WORD CALLBACK A812_ChScan_IntRemove(void);

// Timer/Counter Functions
EXPORTS void CALLBACK A812_SetCounter
(WORD wCounterNo, WORD bCounterMode, DWORD wCounterValue);
EXPORTS DWORD CALLBACK A812_ReadCounter
(WORD wCounterNo, WORD bCounterMode);
```

1.2 A812.BAS

```
'********************************************************************************
' Declarations of the A812.DLL for the A812 DAQ Card
'********************************************************************************

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

'***************** DEFINE A812 RELATIVE ADDRESS *****************
Global Const A812_TIMER0            = &H0
Global Const A812_Timer1            = &H1
Global Const A812_TIMER2            = &H2
Global Const A812_TIMER_MODE        = &H3
Global Const A812_AD_LO             = &H4 '* Analog to Digital, Low Byte *
Global Const A812_AD_HI             = &H5 '* Analog to Digital, High Byte*

Global Const A812_DA_CH0_LO         = &H4 '* Digit to Analog, CH0*
Global Const A812_DA_CH0_HI         = &H5
Global Const A812_DA_CH1_LO         = &H6 '* Digit to Analog, CH1*
Global Const A812_DA_CH1_HI         = &H7
Global Const A812_DI_LO             = &H6 '* Digital Input *
Global Const A812_DO_LO             = &HD '* Digital Output *

Global Const A812_CLEAR_IRQ         = &H8
Global Const A812_SET_GAIN          = &H9
Global Const A812_SET_CH            = &HA
Global Const A812_SET_MODE          = &HB
Global Const A812_SOFT_TRIG         = &HC

Global Const A812_POLLING_MODE       = 1
Global Const A812_DMA_MODE           = 2
Global Const A812_INTERRUPT_MODE     = 6
```

```
'*** define the gain mode ***/
Global Const A812_BI_1                    = 0
Global Const A812_BI_2                    = 1
Global Const A812_BI_4                    = 2
Global Const A812_BI_8                     = 3
Global Const A812_BI_16                   = 4

Global Const A812_NoError                 = 0
Global Const A812_DriverOpenError         = 1
Global Const A812_DriverNoOpen            = 2
Global Const A812_GetDriverVersionError   = 3
Global Const A812_GetTotalBoardsError     = 4

Global Const A812_BoardNotFound           = 5
Global Const A812_ActiveBoardError        = 6
Global Const A812_ExceedBoardNo           = 7
Global Const A812_GetConfigError          = 8
Global Const A812_AllocateMemoryError     = 9
Global Const A812_TimeoutError            = 10

Global Const A812_InvalidGainCode          = 11
Global Const A812_InvalidJump10v          = 12
Global Const A812_InvalidChannelNo        = 13

Global Const A812_IntSetEventError        = 14
Global Const A812_IntInstallError         = 15
Global Const A812_IntClearCountError      = 16
Global Const A812_IntGetCountError        = 17
Global Const A812_IntGetBufferError       = 18
Global Const A812_IntRemoveError          = 19

Global Const A812_ChScanBufferFull        = 20
Global Const A812_ChScanNoChannelToScan   = 21
Global Const A812_ChScanIntSetChannelsError = 22
Global Const A812_ChScanIntSetConfigsError  = 23
```

```
'******* Test Functions ***********
Declare Function A812_SHORT_SUB_2 Lib "A812.DLL" _
(ByVal nA As Integer, ByVal nB As Integer) As Integer
Declare Function A812_FLOAT_SUB_2 Lib "A812.DLL" _
(ByVal fA As Single, ByVal fB As Single) As Single
Declare Function A812_Get_DLL_Version Lib "A812.DLL" () As Integer
Declare Function A812_GetDriverVersion Lib "A812.DLL" _
(wDriverVersion As Integer) As Integer

'******* DI/DO Functions ***********
Declare Function A812_DI Lib "A812.DLL" (wInVal As Integer) As Integer
Declare Function A812_DO Lib "A812.DLL" _
(ByVal wHexValue As Integer) As Integer
Declare Sub A812_OutputByte Lib "A812.DLL" _
(ByVal wPortAddr As Integer, ByVal bOutputVal As Byte)
Declare Sub A812_OutputWord Lib "A812.DLL" _
(ByVal wPortAddr As Integer, ByVal wOutputVal As Integer)
Declare Function A812_InputByte Lib "A812.DLL" _
(ByVal wPortAddr As Integer) As Integer
Declare Function A812_InputWord Lib "A812.DLL" _
(ByVal wPortAddr As Integer) As Integer

'******* AD Functions ***********
Declare Function A812_Fast_SetChGain Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wGainCode As Integer, _
ByVal wJump10v As Integer) As Integer
Declare Function A812_Fast_AD_Hex Lib "A812.DLL" _
(wVal As Integer) As Integer
Declare Function A812_Fast_AD_Float Lib "A812.DLL" _
(fVal As Single) As Integer

Declare Function A812_AD_Hex Lib "A812.DLL" _
(ByVal wChannel As Integer, _
ByVal wGainCode As Integer, wVal As Integer) As Integer
Declare Function A812_ADs_Hex Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wGainCode As Integer, _
wBuf As Integer, ByVal wCount As Integer) As Integer
```

```
Declare Function A812_AD_Float Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wGainCode As Integer, _
ByVal wJump10v As Integer, fVal As Single) As Integer
Declare Function A812_ADs_Float Lib "A812.DLL" _
(ByVal wChannel As Integer, _
ByVal wGainCode As Integer, ByVal wJump10v As Integer, _
fBuf As Single, ByVal wCount As Integer) As Integer
Declare Function A812_Hex2Float Lib "A812.DLL" _
(ByVal wHex As Integer, ByVal wGainCode As Integer, _
ByVal wJump10v As Integer, fVal As Single) As Integer

'******* DA Functions ***********
Declare Function A812_DA_Hex Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wHexValue As Integer) As Integer
Declare Function A812_DA_Uni5 Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal fValue As Single) As Integer
Declare Function A812_DA_Uni10 Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal fValue As Single) As Integer

'******* Driver Functions ***********
Declare Function A812_DriverInit Lib "A812.DLL" _
(wTotalBoards As Integer) As Integer
Declare Sub A812_DriverClose Lib "A812.DLL" ()
Declare Function A812_DELAY Lib "A812.DLL" _
(ByVal wDownCount As Integer) As Integer
Declare Function A812_Check_Address Lib "A812.DLL" () As Integer
Declare Function A812_GetConfigAddress Lib "A812.DLL" _
(wAddrBase As Integer, wCurrentBoard As Integer) As Integer
Declare Function A812_ActiveBoard Lib "A812.DLL" _
(ByVal wBoardNo As Integer) As Integer

'******* Interrupt Functions ***********
Declare Function A812_IntInstall Lib "A812.DLL" _
(hEvent As Long, ByVal dwCount As Integer) As Integer
Declare Function A812_IntStart Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wGainCode As Integer, _
ByVal wJump10v As Integer, ByVal c1 As Integer, _
ByVal c2 As Integer) As Integer
```

```
Declare Function A812_IntStop Lib "A812.DLL" () As Integer
Declare Function A812_IntRemove Lib "A812.DLL" () As Integer
Declare Function A812_IntGetCount Lib "A812.DLL" (dwVal As Long) As
Integer
Declare Function A812_IntGetHexBuf Lib "A812.DLL" _
(ByVal dwNum As Long, wBuffer As Integer) As Integer
Declare Function A812_IntGetFloatBuf Lib "A812.DLL" _
(ByVal dwNum As Integer, fbuffer As Single) As Integer

'// Function of AD Channel-Scan
Declare Sub A812_ChScan_Clear Lib "A812.DLL" ()
Declare Function A812_ChScan_Add Lib "A812.DLL" _
(ByVal wChannel As Integer, ByVal wConfig As Integer) As Integer
Declare Function A812_ChScan_Set Lib "A812.DLL" _
(wChannel As Integer, wConfig As Integer, _
ByVal wChNum As Integer) As Integer
Declare Function A812_ChScan_PollingHex Lib "A812.DLL" _
(wBuf As Integer, ByVal wNumPerCh As Integer) As Integer
Declare Function A812_ChScan_PollingFloat Lib "A812.DLL" _
(ByVal wJump10v As Integer, _
fBuf As Single, ByVal wNumPerCh As Integer) As Integer

'// Function of AD Channel-Scan with Interrupt
Declare Function A812_ChScan_IntInstall Lib "A812.DLL" _
(hEvent As Long, ByVal dwNumPerCh As Long) As Integer
Declare Function A812_ChScan_IntStart Lib "A812.DLL" _
(ByVal wCounter1 As Integer, ByVal wCounter2 As Integer, _
ByVal wJump10v As Integer) As Integer
Declare Function A812_ChScan_IntGetHexBuf Lib "A812.DLL" _
(wBuf As Integer) As Integer
Declare Function A812_ChScan_IntGetFloatBuf Lib "A812.DLL" _
(fBuf As Single) As Integer
Declare Function A812_ChScan_IntGetCount Lib "A812.DLL" _
(dwVal As Long) As Integer
Declare Function A812_ChScan_IntStop Lib "A812.DLL" () As Integer
Declare Function A812_ChScan_IntRemove Lib "A812.DLL" () As Integer
```

```
'//Function of Timer/Counter
Declare Sub A812_SetCounter Lib "A812.DLL" (ByVal wCounterNo As
Integer, _ ByVal bCounterMode As Integer, ByVal wCounterValue As Long)
Declare Function A812_ReadCounter Lib "A812.DLL" _
(ByVal wCounterNo As Integer, ByVal bCounterMode As Integer) As Long
```

1.3 A812.PAS

```
unit A812;
interface
type PSingle=^Single;
PWord=^Word;
PInteger=^Integer;
Const

//***************** DEFINE A812 RELATIVE ADDRESS *****************/
A812_TIMER0                = $00;
A812_TIMER1                = $01;
A812_TIMER2                = $02;
A812_TIMER_MODE            = $03;
A812_AD_LO                 = $04; //* Analog to Digital, Low Byte */
A812_AD_HI                 = $05; //* Analog to Digital, High Byte */
A812_DA_CH0_LO             = $04; //* Digital to Analog, CH 0 */
A812_DA_CH0_HI             = $05;
A812_DA_CH1_LO             = $06; //* Digital to Analog, CH 1 */
A812_DA_CH1_HI             = $07;
A812_DI_LO                 = $06; //* Digital Input */
A812_DO_LO                 = $0D; //* Digital Output */

A812_CLEAR_IRQ             = $08;
A812_SET_GAIN              = $09;
A812_SET_CH                = $0A;
A812_SET_MODE              = $0B;
A812_SOFT_TRIG             = $0C;

A812_POLLING_MODE          = 1;
A812_DMA_MODE              = 2;
A812_INTERRUPT_MODE         = 6;
```

```
//*** Define the gain mode ***/
A812_BI_1               =0;
A812_BI_2               =1;
A812_BI_4               =2;
A812_BI_8               =3;
A812_BI_16              =4;

A812_NoError                    =0;
A812_DriverOpenError            =1;
A812_DriverNoOpen               =2;
A812_GetDriverVersionError      =3;

A812_GetTotalBoardsError        =4;
A812_BoardNotFound              =5;
A812_ActiveBoardError           =6;
A812_ExceedBoardNo              =7;
A812_GetConfigError             =8;
A812_AllocateMemoryError        =9;
A812_TimeoutError               =10;

A812_InvalidGainCode            =11;
A812_InvalidJump10v             =12;
A812_InvalidChannelNo           =13;

A812_IntSetEventError           =14;
A812_IntInstallError            =15;
A812_IntClearCountError         =16;
A812_IntGetCountError           =17;
A812_IntGetBufferError          =18;
A812_IntRemoveError             =19;

A812_ChScanBufferFull               =20;
A812_ChScanNoChannelToScan          =21;
A812_ChScanIntSetChannelsError      =22;
A812_ChScanIntSetConfigsError       =23;
```

```
// Test Functions
Function A812_SHORT_SUB_2(nA, nB : SmallInt):SmallInt; StdCall;
Function A812_FLOAT_SUB_2(fA, fB : Single):Single; StdCall;
Function A812_Get_DLL_Version:WORD; StdCall;
Function A812_GetDriverVersion(var wDriverVersion:WORD):Word; StdCall;

// DI/DO Functions
Function A812_DO(wHexValue:Word):Word; StdCall;
Function A812_DI(var wInVal:Word):Word; StdCall;
Function A812_InputByte(wPortAddr:WORD):WORD; StdCall;
Function A812_InputWord(wPortAddr:WORD):WORD; StdCall;
Procedure A812_OutputByte(wPortAddr:WORD; bOutputVal:Byte); StdCall;
Procedure A812_OutputWord(wPortAddr:WORD; wOutputVal:WORD);
StdCall;

// Function of AD
Function A812_Fast_SetChGain(wChannel:WORD;
wGainCode:WORD; wJump10v:WORD):WORD; StdCall;
Function A812_Fast_AD_Hex( var wVal:WORD):WORD; StdCall;
Function A812_Fast_AD_Float( var fVal:Single):WORD; StdCall;
Function A812_AD_Hex( wChannel:WORD; wGainCode:WORD; var
wVal:WORD):WORD; StdCall;
Function A812_ADs_Hex(wChannel:WORD; wGainCode:WORD;
wBuf:PWord; wCount:WORD):WORD; StdCall;
Function A812_AD_Float( wChannel:WORD; wGainCode:WORD;
wJump10v:WORD; var fVal:Single):WORD; StdCall;
Function A812_ADs_Float( wChannel:WORD; wGainCode:WORD;
wJump10v:WORD; fBuf:PSingle; wCount:WORD):WORD; StdCall;
Function A812_Hex2Float(wHex:WORD; wGainCode:WORD;
wJump10v:WORD; var fVal:Single):WORD; StdCall;

// DA Functions
Function A812_DA_Hex(wChannel, wHexValue:WORD):WORD; StdCall;
Function A812_DA_Uni5(wChannel:Word;fValue:Single):WORD; StdCall;
Function A812_DA_Uni10(wChannel:Word;fValue:Single):WORD; StdCall;
```

```
// Driver Functions
Function A812_DELAY(wDownCount:WORD):WORD; StdCall;
Function A812_Check_Address:WORD; StdCall;
Function A812_DriverInit(var wTotalBoards:WORD):WORD; StdCall;
Procedure A812_DriverClose; StdCall;
Function A812_GetConfigAddress(var wAddrBase:WORD; var
wCurrentBoard:WORD):WORD; StdCall;
Function A812_ActiveBoard( wBoardNo:WORD ):WORD; StdCall;

// Interrupt Functions
Function A812_IntInstall(
var hEvent:LongInt; dwCount:LongInt):WORD; StdCall;
Function A812_IntStart(wChannel:WORD; wGainCode:WORD;
wJump10v:WORD; wCounter1:WORD; wCounter2:WORD):WORD; StdCall;
Function A812_IntStop:WORD; StdCall;
Function A812_IntRemove:WORD; StdCall;
Function A812_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A812_IntGetHexBuf(dwNum:LongInt; wBuf:PWord):WORD; StdCall;
Function A812_IntGetFloatBuf(dwNum:LongInt; fBuf:PSingle):WORD; StdCall;

// AD Channel-Scan Functions
Procedure A812_ChScan_Clear; StdCall;
Function A812_ChScan_Add(wChannel:WORD; wConfig:WORD):WORD;
StdCall;
Function A812_ChScan_Set(wChannel:PWord; wConfig:PWord;
wChNum:WORD):WORD; StdCall;
Function A812_ChScan_PollingHex(
wBuf:PWORD; wNumPerCh:WORD):WORD; StdCall;
Function A812_ChScan_PollingFloat(
wJump10v:WORD; fBuf:PSingle; wNumPerCh:WORD):WORD; StdCall;

// AD Channel-Scan with Interrupt Functions
Function A812_ChScan_IntInstall(
var hEvent:LongInt; dwNumPerCh:LongInt):WORD; StdCall;
Function A812_ChScan_IntStart(
wCounter1:WORD; wCounter2:WORD; wJump10v:WORD):WORD; StdCall;
Function A812_ChScan_IntGetHexBuf(wBuf:PWord):WORD; StdCall;
Function A812_ChScan_IntGetFloatBuf(fBuf:PSingle):WORD; StdCall;
```

```
Function A812_ChScan_IntGetCount(var dwVal:LongInt):WORD; StdCall;
Function A812_ChScan_IntStop:WORD; StdCall;
Function A812_ChScan_IntRemove:WORD; StdCall;

// Timer/Counter Functions
Procedure A812_SetCounter(wCounterNo:WORD;
bCounterMode:WORD; wCounterValue:LongInt); StdCall;
Function A812_ReadCounter(
wCounterNo:WORD; bCounterMode:WORD):LongInt; StdCall;

implementation

Function A812_SHORT_SUB_2; external 'A812.DLL' name
'A812_SHORT_SUB_2';
Function A812_FLOAT_SUB_2; external 'A812.DLL' name
'A812_FLOAT_SUB_2';
Function A812_Get_DLL_Version; external 'A812.DLL' name
'A812_Get_DLL_Version';
Function A812_GetDriverVersion; external 'A812.DLL' name
'A812_GetDriverVersion';

Function A812_DO; external 'A812.DLL' name 'A812_DO';
Function A812_DI; external 'A812.DLL' name 'A812_DI';
Procedure A812_OutputByte; external 'A812.DLL' name 'A812_OutputByte';
Procedure A812_OutputWord; external 'A812.DLL' name 'A812_OutputWord';
Function A812_InputByte; external 'A812.DLL' name 'A812_InputByte';
Function A812_InputWord; external 'A812.DLL' name 'A812_InputWord';

Function A812_Fast_SetChGain; external 'A812.DLL' name
'A812_Fast_SetChGain';
Function A812_Fast_AD_Hex; external 'A812.DLL' name
'A812_Fast_AD_Hex';
Function A812_Fast_AD_Float; external 'A812.DLL' name
'A812_Fast_AD_Float';

Function A812_AD_Hex; external 'A812.DLL' name 'A812_AD_Hex';
Function A812_ADs_Hex; external 'A812.DLL' name 'A812_ADs_Hex';
Function A812_AD_Float; external 'A812.DLL' name 'A812_AD_Float';
```

```
Function A812_ADs_Float; external 'A812.DLL' name 'A812_ADs_Float';
Function A812_Hex2Float; external 'A812.DLL' name 'A812_Hex2Float';
Function A812_DA_Hex; external 'A812.DLL' name 'A812_DA_Hex';
Function A812_DA_Uni5; external 'A812.DLL' name 'A812_DA_Uni5';
Function A812_DA_Uni10; external 'A812.DLL' name 'A812_DA_Uni10';
Function A812_DriverInit; external 'A812.DLL' name 'A812_DriverInit';
Procedure A812_DriverClose; external 'A812.DLL' name 'A812_DriverClose';
Function A812_DELAY; external 'A812.DLL' name 'A812_DELAY';
Function A812_Check_Address; external 'A812.DLL' name
'A812_Check_Address';
Function A812_GetConfigAddress; external 'A812.DLL' name
'A812_GetConfigAddress';
Function A812_ActiveBoard; external 'A812.DLL' name 'A812_ActiveBoard';

Function A812_IntInstall; external 'A812.DLL' name 'A812_IntInstall';
Function A812_IntStart; external 'A812.DLL' name 'A812_IntStart';
Function A812_IntStop; external 'A812.DLL' name 'A812_IntStop';
Function A812_IntRemove; external 'A812.DLL' name 'A812_IntRemove';
Function A812_IntGetCount; external 'A812.DLL' name 'A812_IntGetCount';
Function A812_IntGetHexBuf; external 'A812.DLL' name 'A812_IntGetHexBuf';
Function A812_IntGetFloatBuf; external 'A812.DLL' name
'A812_IntGetFloatBuf';

// AD Channel-Scan Functions
Procedure A812_ChScan_Clear; external 'A812.DLL' name
'A812_ChScan_Clear';
Function A812_ChScan_Add; external 'A812.DLL' name 'A812_ChScan_Add';
Function A812_ChScan_Set; external 'A812.DLL' name 'A812_ChScan_Set';
Function A812_ChScan_PollingHex; external 'A812.DLL' name
'A812_ChScan_PollingHex';
Function A812_ChScan_PollingFloat; external 'A812.DLL' name
'A812_ChScan_PollingFloat';
```

```
// AD Channel-Scan (with Interrupt) Functions
Function A812_ChScan_IntInstall; external 'A812.DLL' name
'A812_ChScan_IntInstall';
Function A812_ChScan_IntStart; external 'A812.DLL' name
'A812_ChScan_IntStart';
Function A812_ChScan_IntGetHexBuf; external 'A812.DLL' name
'A812_ChScan_IntGetHexBuf';
Function A812_ChScan_IntGetFloatBuf; external 'A812.DLL' name
'A812_ChScan_IntGetFloatBuf';
Function A812_ChScan_IntGetCount; external 'A812.DLL' name
'A812_ChScan_IntGetCount';
Function A812_ChScan_IntStop; external 'A812.DLL' name
'A812_ChScan_IntStop';
Function A812_ChScan_IntRemove; external 'A812.DLL' name
'A812_ChScan_IntRemove';

// Timer/Counter Functions
Procedure A812_SetCounter; external 'A812.DLL' name 'A812_SetCounter';
Function A812_ReadCounter; external 'A812.DLL' name 'A812_ReadCounter';
end.
```

2. Reference

2.1 Range Configuration

The AD converter of the A812PG is 12 bits under all configuration codes. If the analog input range is configured to the +/-5V range, the resolution of one bit is equal to 2.44mV. If the analog input range is configured to the +/-10V range, the resolution will be 4.88mV. When the analog input signal is about 1V, use configuration code 0/1/2 if JP4 is adjusted to ±5V, or code 0/1/2/3 when the JP4 is adjusted to ±10V. This will achieve approximately the same result **except for the resolution. Choosing the correct configuration code will allow the highest precision measurement to be achieved.**

A-812PG Input Signal Range Configuration Code Table

JP4 = ± 5V

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 5 V	0
Bipolar	+/- 2.5 V	1
Bipolar	+/- 1.25 V	2
Bipolar	+/- 0.625 V	3
Bipolar	+/- 0.3125 v	4

JP4 = ± 10V

Bipolar/Unipolar	Input Signal Range	Configuration Code
Bipolar	+/- 10 V	0
Bipolar	+/- 5 V	1
Bipolar	+/- 2.5 V	2
Bipolar	+/- 1.25 V	3
Bipolar	+/- 0.625 v	4

2.2 Error Codes Table

For the most errors, it is recommended to check:

1. Does the device driver installs successful?
2. Does the card have plugged?
3. Does the card conflicts with other device?
4. Close other applications to free the system resources.
5. Try to use another slot to plug the card.
6. Restart your system to try again.

Error Code	Description
A812_NoError	OK (No error)
A812_DriverOpenError	Check whether the driver is installed correctly. If not, add the A-812 using the “Add new device” function in the Windows control panel.
A812_DriverNoOpen	Call the driver initialization function before opening the driver.
A812_GetDriverVersionError	Can’t call the device driver function. Call the driver initialization function before opening the driver.
A812_GetTotalBoardsError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_BoardNotFound	The board was not found. Add the A-812 using the “Add new device” function in the Windows control panel.
A812_ActiveBoardError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_ExceedBoardNo	The number of the board you select is out of range.
A812_GetConfigError	This error occurs when the DLL call to the A-812 driver was not successful.
A812_AllocateMemoryError	Windows does not have enough RAM resources.
A812_TimeoutError	The crystal or the 8254 chip has possibly failed.

A812_InvalidGainCode	An invalid parameter w3as detected while calling the A-812 function. (Valid range is 0~4)
A812_InvalidJump10v	An invalid parameter was detected while calling the A-812 function. (0 for 5V; 1 for 10V)
A812_InvalidChannelNo	An invalid parameter was detected while calling the A-812 function.
A812_IntSetEventError	An invalid parameter was detected while calling the A-812 function.
A812_IntInstallError	This error occurs when the system detects an IRQ conflict.
A812_IntClearCountError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_IntGetCountError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_IntGetBufferError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_IntRemoveError	This error occurs when a DLL call to the A-812 driver is not successful.
A812_ChScanBufferFull	The program buffer is full.
A812_ChScanNoChannelToScan	No channel was selected for the A/D operation.
A812_ChScanIntSetChannelsError	An invalid parameter was detected while calling the A-812 function.
A812_ChScanIntSetConfigsError	An invalid parameter was detected while calling the A-812 function.

2.3 Other Manuals

For more information, please refer to the following user manuals:

■ **PCI_ISA_PnP_Driver_Installation_in_Win9x_2K_XP.pdf:**
Installing the software package under Windows 95/98/NT/2000.

■ **Calling_DLL_functions_in_VB_VC_Delphi_BCB.pdf:**
Including the declaration files and calling the DLL functions with VC++5, VB5, Delphi3 and Borland C++ Builder 3.

■ **TroubleShooting_PCI_ISA_in_Win32_Resource_Conflict.pdf:**
Checking the resources I/O Port address, IRQ number and DMA number for add-on cards under Windows 95/98/NT/2000.

■ **PCI_ISA_PnP_Driver_Installation_in_Win9x_2K_XP.pdf:**
Installing the Plug and Play information file (*.inf) under Windows 95/98/2000.

3. Function Descriptions

In order to simplify and clarify the description, the attribute of the input and output parameters of the function is indicated as [In] and [Out], respectively, as shown in the following table.

Keyword	Parameter must be set by the user before calling the function	Data/value from this parameter is retrieved after calling the function
[In]	**Yes**	No
[Out]	No	**Yes**
[In, Out]	**Yes**	**Yes**

Note: All of the parameters need to be allocated spaces by the user.

Driver Functions	
Sec. 3.4.1	A812_DriverInit
Sec. 3.4.2	A812_DriverClose
Sec. 3.4.3	A812_DELAY
Sec. 3.4.4	A812_Check_Address
Sec. 3.4.5	A812_GetConfigAddress
Sec. 3.4.6	A812_ActiveBoard
AD Interrupt Functions	
Sec. 3.5.1	A812_IntInstall
Sec. 3.5.2	A812_IntStart
Sec. 3.5.3	A812_IntStop
Sec. 3.5.4	A812_IntRemove
Sec. 3.5.5	A812_IntGetCount
Sec. 3.5.6	A812_IntGetHexBuf
Sec. 3.5.7	A812_IntGetFloatBuf
AD, Channel Scan Functions	
Sec. 3.6.2	A812_ChScan_Clear
Sec. 3.6.3	A812_ChScan_Add
Sec. 3.6.4	A812_ChScan_Set
Sec. 3.6.5	A812_ChScan_PollingHex
Sec. 3.6.6	A812_ChScan_PollingFlaot
AD Interrupt, Channel Scan Functions	
Sec. 3.7.2	A812_ChScan_IntInstall
Sec. 3.7.3	A812_ChScan_IntStart
Sec. 3.7.4	A812_ChScan_IntStop
Sec. 3.7.5	A812_ChScan_IntRemove
Sec. 3.7.6	A812_ChScan_IntGetCount
Sec. 3.7.7	A812_ChScan_IntGetHexBuf
Sec. 3.7.8	A812_ChScan_IntGetFloatBuf
Timer, Count Functions	
Sec. 3.8.1	A812_SetCounter
Sec. 3.8.2	A812_ReadCounter

3.1 Test Functions

3.1.1 A812_SHORT_SUB_2

- **Description:**

  This function computes C=A-B in **short** format, **Short =16-bit signed integer.** This function is provided for testing purposes.

- **Syntax:**

  short **A812_SHORT_SUB_2**(short **nA**, short **nB**);

- **Parameters:**

nA	[In]	Short integer
nB	[In]	Short integer

- **Returns:**

  Return = nA – nB → short integer

3.1.2 A812_FLOAT_SUB_2

- **Description:**

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

- **Syntax:**

  float **A812_FLOAT_SUB_2**(float **fA**, float **fB**);

- **Parameters:**

fA	[In]	Floating point value
fB	[In]	Floating point value

- **Returns:**

  Return = fA – fB → floating point value

3.1.3 A812_Get_DLL_Version

- **Description:**
  This function reads the version number of the A812.DLL.

- **Syntax:**
  WORD **A812_Get_DLL_Version**(void);

- **Parameters:**
  void

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

3.1.4 A812_GetDriverVersion

- **Description:**
  This subroutine will identify the device driver of the version number.

- **Syntax:**
  WORD **A812_GetDriverVersion**(WORD ***wDriverVersion**);

- **Parameters:**

wDriverVersion	[Out]	The wDriverVersion address. When wDriverVersion = 0x210 → the version is 2.10

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.2 Digital I/O Functions

3.2.1 A812_DI

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

- **Syntax:**
  WORD **A812_DI**(WORD ***wInVal**);

- **Parameters:**

wInVal	[Out]	The 16-bit Digital Input value

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.2.2 A812_DO

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

- **Syntax:**
  WORD **A812_DO**(WORD **wHexValue**);

- **Parameters :**

wHexValue	[In]	The 16-bit data sent to the digital output port

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.2.3 A812_OutputByte

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

- **Syntax:**
  void **A812_OutputByte**(DWORD **wPortAddr**, UCHAR **bOutputVal**);

- **Parameters:**

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

- **Returns:**
  None

3.2.4 A812_OutputWord

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

- **Syntax:**
  void **A812_OutputWord**(DWORD **wPortAddr,** WORD **wOutputVal**);

- **Parameters:**

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

- **Returns:**
  valid

3.2.5 A812_InputByte

- **Description:**
  This subroutine will send 8-bit of data from the assigned I/O port.

- **Syntax:**
  WORD **A812_InputByte**(DWORD **wPortAddr**);

- **Parameters:**

wPortAddr	[In]	The I/O port address, for example, 0x220

- **Returns:**
  16-bit data where the leading 8 bits are all 0

3.2.6 A812_InputWord

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

- **Syntax:**
  WORD **A812_InputWord**(WORD **wPortAddr**);

- **Parameters:**

wPortAddr	[In]	The I/O port address, for example, 0x220

- **Returns:**
  16-bit data

3.3 A/D, D/A Functions

3.3.1 A812_Fast_SetChGain

■ **Description:**

This subroutine sets the channel number and configuration code for the ADC.

This function should be called once before calling either the "A812_Fast_AD_Hex()", "A812_Fast_AD_Float()", "A812_IntStart()", "A812_ADs_Hex()" or "A812_ADs_Float()" functions.

■ **Syntax:**

WORD **A812_Fast_SetChGain**(WORD **wChannel**, WORD **wGainCode**, WORD **wJump10v**);

■ **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to 15.
wGainCode	[In]	Configuration code. Refer to "Sec. 2.1 Range Configuration" for more detailed information.
wJump10v	[In]	Depends on the value chosen for JP4. (0: +/- 5V; 1: +/- 10V)

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.2 A812_Fast_AD_Hex

- **Description:**
  This subroutine will perform an A/D conversion using a polling approach. The A/D converter is 12 bits for A812PG. The "A812_Fast_SetChGain()" function must be called first.

- **Syntax:**
  WORD **A812_Fast_AD_Hex**(WORD ***wVal**);

- **Parameters:**

wVal	[Out]	The 12-bit Hex value for the analog input

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.3 A812_Fast_AD_Float

- **Description:**
  This subroutine will perform an A/D conversion using a polling approach. The A/D converter is 12 bits for the A-812PG. The value will be computed according to the **configuration code (see Sec. 2.1 for details)**. The "A812_Fast_SetChGain()" function must be called first.

- **Syntax:**
  WORD **A812_Fast_AD_Float**(float ***fval**);

- **Parameters:**

fVal	[Out]	The floating point value for the analog input

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.4 A812_AD_Hex

- **Description:**
  This subroutine will perform an A/D conversion using a polling approach. The A/D converter is 12 bits for the A-812PG.

- **Syntax:**
  WORD **A812_AD_Hex**(WORD **wChannel**, WORD **wGainCode,** WORD ***wVal**);

- **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to 15.
wGainCode	[In]	The configuration code, refer to “Sec. 2.1 Range Configuration Code” for more detailed information.
wVal	[Out]	The 12-bit Hex value for the analog input.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.5 A812_AD_Float

- **Description:**
  This subroutine will perform an A/D conversion using a polling. The A/D converter is 12 bits for the A-812PG. The value will be computed according to the configuration code (see Sec. 2.1 for details).

- **Syntax:**
  WORD **A812_AD_Float**(WORD **wChannel**, WORD **wGainCode**, WORD **wJump10v**, float ***fVal**);

- **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to15.
wGainCode	[In]	The configuration code, refer to “Sec. 2.1 Range Configuration” for more detailed information.
wJump10v	[In]	Depends on the value chosen for JP4. 0 → +/- 5V; 1→ 10V
fVal	[Out]	The floating point value for the analog input.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.6 A812_ADs_Hex

■ **Description:**

This subroutine will perform a number of A/D conversions using a polling approach, and is very similar to the A812_AD_Hex function except that it will perform a wCount number of conversions instead of just one conversion. The A/D conversing at the ISA bus's max. speed. After the A/D conversion is complete, the A/D data is stored in a buffer in Hex format. The wBuf value is the starting address of the data buffer. The "A812_Fast_SetChGain()" function must be called first.

■ **Syntax:**

WORD **A812_ADs_Hex**(WORD **wChannel**, WORD **wGainCode**, WORD **wBuf[]**, WORD **wCount**);

■ **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to 15.
wGainCode	[In]	The configuration code, refer to "Sec. 2.1 Range Configuration" for more detailed information.
wBuf	[Out]	The starting address of the data buffer (In WORD format) The user must first be allocated for this buffer and the address sent to the function. The function will input the data into this buffer. The data can then be analyzed after calling this function.
wCount	[In]	The number of A/D conversions to be performed

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.7 A812_ADs_Float

■ **Description:**

This subroutine will perform a number of A/D conversions using a polling approach, and is very similar to the A812_AD_Float except that this subroutine will perform a wCount numbers of conversions instead of just one conversion. After the A/D conversion is complete the A/D data is stored in a data buffer in Float format. The fBuf value is the starting address of the data buffer. The “A812_Fast_SetChGain()” function must be called first.

■ **Syntax:**

WORD **A812_ADs_Float**(WORD **wChannel**, WORD **wGainCode**, WORD **wJump10v**, float **fBuf[]**, WORD **wCount**);

■ **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to15.
wGainCode	[In]	The configuration code, refer to “Sec. 2.1 Range Configuration” for more detailed information.
wJump10v	[In]	Depends on the value chosen for JP4. 0 → +/- 5V; 1→ 10V
fBuf	[Out]	The starting address of the data buffer (In float format) The user must allocate for this buffer and the address sent to the function. The function will input the data into the buffer. The data can then be analyzed after calling this function.
wCount	[In]	The number of A/D conversions to be performed

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.8 A812_DA_Hex

■ **Description:**

This subroutine will send 12-bit data to the D/A analog output. The output range of the D/A may be either 0-5V or 0-10V and set using the hardware jumper, JP3. It is not possible for the software to detect the output range of the D/A converter. **For examples, if the jumper setting is selected as -5V, the maximum value of 5V will be sent. Conversely, if the jumper setting is selected as -10V, the maximum value of 10V will be sent. An output range of 0-5V is selected as the default factory setting.**

■ **Syntax:**

WORD **A812_DA_Hex**(WORD **wChannel,** WORD **wHexValue**);

■ **Parameters:**

wChannel	[In]	The D/A channel number. Can be either 0 or 1.
wHexValue	[In]	The 12-bit data sent to the D/A converter.

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.9 A812_DA_Uni5

- **Description:**
  This subroutine will send 12-bit data to the D/A analog output. The D/A output range is dependent on **setting of jumper JP3 (either -5V or -10V)**. It is not possible for the software to detect the output range of the D/A converter. The subroutine can only be used when the jumper setting is set as **-5V, which means that the output range is between 0.0V and 5.0V.** Please refer to hardware manual for more information regarding jumper settings.

- **Syntax:**
  Viod **A812_DA_Uni5**(WORD **wChannel**, float **fValue**);

- **Parameters:**

wChannel	[In]	The D/A channel number. 0 to 1.
fValue	[In]	The 12-bit data sent to the D/A converter.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.3.10 A812_DA_Uni10

- **Description:**
  This subroutine will send 12-bit data to the D/A analog output. The D/A output range is dependent on **setting of jumper JP3 (either -5V or -10V)**. It is not possible for the software to detect the output range of the D/A converter. The subroutine can only be used when the jumper setting is set as **-10V, which means that the output range is between 0.0V and 10.0V.** Please refer to hardware manual for more information regarding jumper settings.

- **Syntax:**
  Viod **A812_DA_Uni10**(WORD **wChannel**, float **fValue**);

- **Parameters:**

wChannel	[In]	The D/A channel number. 0 to 1.
fValue	[In]	The 12-bit data sent to the D/A converter.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.4 Driver Functions

3.4.1 A812_DriverInit

- **Description:**
  This subroutine will open the device driver. After calling the A812_DriverInit() function, the A812_ActiveBoard() function must be called first before access the device (See Sec. 3.4.6 for details).

- **Syntax:**
  WORD **A812_DriverInit**(WORD ***wTotalBoards**);

- **Parameters:**

wTotalBoards	[Out]	Returns the total numbers of boards that were found by the driver.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.4.2 A812_DriverClose

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

- **Syntax:**
  WORD **A812_DriverClose**(void);

- **Parameters:**
  void

- **Returns:**
  void

3.4.3 A812_DELAY

- **Description:**
  This subroutine will delay the **wDownCount** mS (machine independent timer), and uses the System Clock to implement the delay function. The unit used by the A812_DELAY() function are in 0.5μ periods.

- **Syntax:**
  WORD **A812_DELAY**(WORD **wDowncount**);

- **Parameters:**

wDownCount	[In]	Number of 0.5μS will be delayed

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.4.4 A812_Check_Address

- **Description:**
  This subroutine will detect the I/O base address of the A-812PG, and will perform a single A/D conversion, if the A-812PG is successfully detected. This function will always return 0 if the trigger mode is set as external.

- **Syntax:**
  WORD **A812_Check_Address**(void);

- **Parameters:**
  None

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.4.5 A812_GetConfigAddress

- **Description:**
  This subroutine returns the base address and the board-number of the current board.
  If the current board is invalid, the base address will be 0.

- **Syntax:**
  WORD **A812_GetCoufigAddress**(WORD ***wAddrBase**, WORD ***wCurrentBoard**);

- **Parameters:**

wAddrBase	[Out]	Returns the base address of the current board.
wCurrentBoard	[Out]	Returns the board number of the current board.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.4.6 A812_ActiveBoard

- **Description:**
  This subroutine activates the specified board and then calls the A812_Check_Address() function to automatically check the hardware. If the function cannot access the device, it returns the A812_CardNotFound error code. Please refer to the "A812_DriverInit()" function for the valid board number range (See Sec. 2.4.1 for details).

- **Syntax:**
  WORD **A812_ActiveBoard** (WORD **wBoardNo**);

- **Parameters:**

wBoardNo	[In]	The board number to be activated.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5 AD, Interrupt Functions

3.5.1 A812_IntInstall

- **Description:**

  This subroutine will install the interrupt handler and allocate the buffer. For more detailed information about using interrupts refer to the "Architecture of Interrupt Mode" in Sec. 3.5.8.

- **Syntax:**

  WORD **A812_IntInstall**(HANDLE ***hEvent**, DWORD **dwDataCount**);

- **Parameters:**

***hEvent**	[In]	The event handler created by the user.
dwDataCount	[In]	The numbers of A/D entries count for the interrupt transfer.

- **Returns:**

  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.2 A812_IntStart

■ **Description:**

This subroutine will clear the interrupt counter, start the interrupt transfer for a specific A/D channel, and program the gain code and sampling rate. **The "A812_Fast_SetChGain()" function should be called first.**

■ **Syntax:**

WORD **A812_IntStart**(WORD **wChannel**, WORD **wGainCode,** WORD **wJump10v,** WORD **wC1,** WORD **wC2**);

■ **Parameters:**

wChannel	[In]	The A/D channel number. The valid range is 0 to 15.
wGainCode	[In]	Configuration code. Refer to "Sec. 2.1 Range Configuration" for more detailed information.
wJump10v	[In]	Depends on the value chosen for JP4 0→ +/- 5V; 1→ +/-10V
wC1, wC2	[In]	The sampling rate is 2M/C1*C2 C1 → Counter1; C2 → Counter2

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.3 A812_IntStop

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

- **Syntax:**
  WORD **A812_IntStop**(void);

- **Parameters:**
  void

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.4 A812_IntRemove

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

- **Syntax:**
  WORD **A812_IntRemove**(void);

- **Parameters:**
  void

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.5 A812_IntGetCount

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

- **Syntax:**
  WORD **A812_IntGetCount**(DWORD ***dwVal**);

- **Parameters:**

dwVal	[Out]	Returns the interrupt transferred count.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.6 A812_IntGetHexBuf

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

- **Syntax:**
  WORD **A812_IntGetHexBuf**(DWORD **dwNum,** WORD **wBuf[]**);

- **Parameters:**

dwNum	[In]	The Total numbers of channels to be passed into function and scanned.
wBuf	[Out]	The address of the wBuffer (In WORD format). Space must be allocated for the buffer and the address sent into the function. The function will input the data into the buffer. The data can then be analyzed from the buffer after calling this function.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.7 A812_IntGetFloatBuf

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

- **Syntax:**
  WORD **A812_IntGetFloatBuf**(DWORD **dwNum,** float **fBuf[]**);

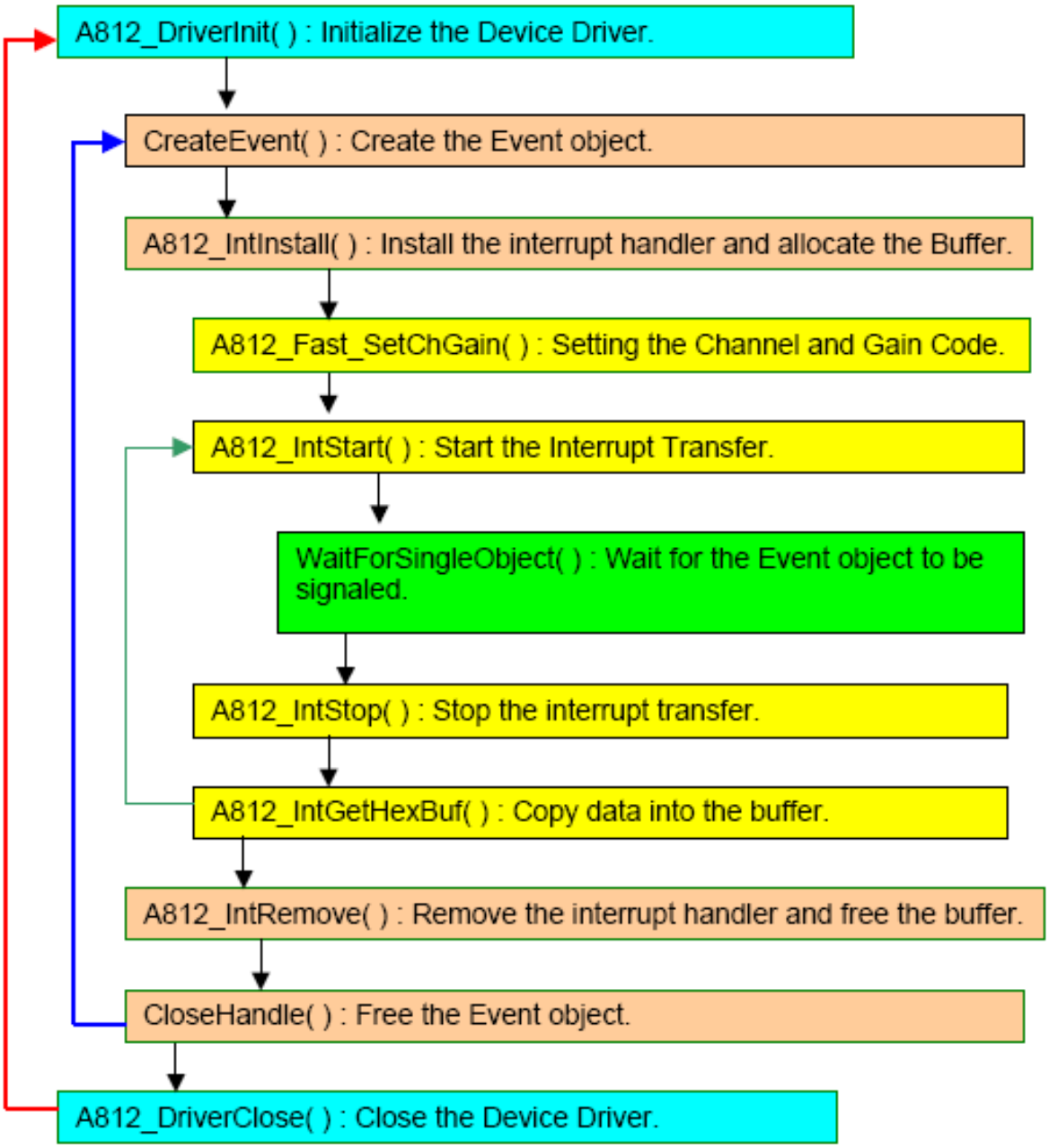
- **Parameters:**

dwNum	[In]	The Total numbers of channels to be passed into function and scanned.
fBuf	[Out]	The address of the fBuffer (In float format). Space must be allocated for the buffer and the address sent into the function. The function will input the data into the buffer. The data can then be analyzed from the buffer after calling this function.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.5.8 Interrupt Mode Architecture

The functions listed in Sec. 3.5.1 to 3.5.7 are used to perform A/D conversions with interrupt transfer. The flow chart for program these functions are as follows:

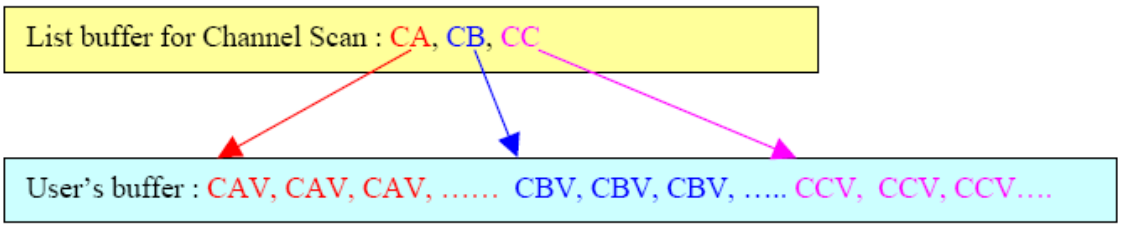


3.6 AD, Channel Scan Function

3.6.1 Introduction

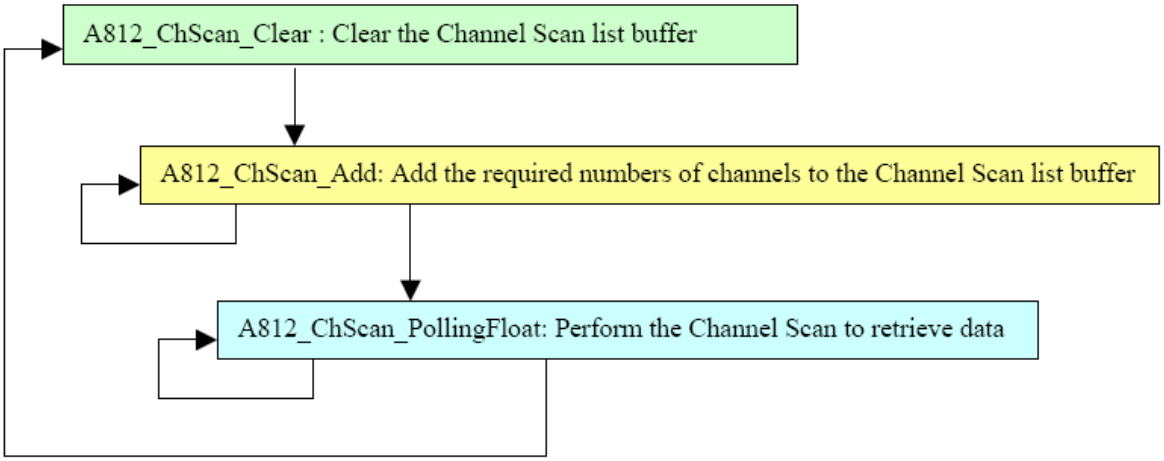
The user can specify the numbers of channels to be need the into the list buffer. Other functions will perform the A/D conversion to receive the data, and then read the list buffer to move to the next channel and set the specified configuration code.

The data will be stored into the following manner:



Note:
CA = Channel A; CB= Channel B; CC= Channel C
CAV = The value of Channel A;
CBV = The value of Channel B;
CCV = The value of Channel C

The program architecture is as follows:



3.6.2 A812_ChScan_Clear

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

■ **Syntax:**
void **A812_ChScan_Clear**(void);

■ **Parameters:**
None

■ **Returns:**
None

3.6.3 A812_ChScan_Add

■ **Description:**
This subroutine will add the specified numbers of channels and configuration-codes to the Channel Scan list buffer. The maximum number of Channel Scan buffers is 100 channels.

■ **Syntax:**
WORD **A812_ChScan_Add**(WORD **wChannel,** WORD **wConfig**);

■ **Parameters:**

wChannel	[In]	The which channel is to scanned
wConfig	[In]	Specifies the configuration code for the channel. Please refer to “Sec. 2.1 Configuration” for detailed information.

■ **Returns:**
Please refer to error codes in Sec. 2.2 for more detailed information.

3.6.4 A812_ChScan_Set

■ **Description:**

This function will clear the list buffer and then copy the specified list of channel(s) and configuration-codes(s) into the channel Scan list buffer. The maximum number of Channels Scan buffers is 100 channels.

■ **Syntax:**

WORD **A812_ChScan_Set**(WORD **wChannel[]**, WORD **wConfig[]**, WORD **wChNum**);

■ **Parameters:**

wChannel	[In]	The list of channel(s) to be scanned.
wConfig	[In]	The list of configuration code(s) for the channels(s). Please refer to “Sec. 2.1 Range Configuration”.
wChNum	[In]	The total numbers of channels to be passed into the function and scanned.

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.6.5 A812_ChScan_PollingHex

■ **Description:**

This subroutine will perform a number of A/D conversions using a polling approach. After retrieving the data from the channel, it then reads the Channel Scan list buffer to move to the next channel and sets the specified configuration code. Once the A/D conversion is complete, the A/D data is stored in a buffer in Hex format.

Before calling this function, must be called the A812_ChScan_Clear() and A812_ChScan_Add() functions to set up the Channel Scan list buffer. Please refer to Section 3.6.1 for more information.

■ **Syntax:**

WORD **A812_ChScan_PollingHex** (WORD **wBuf[]**, WORD **wNumPerCh**);

■ **Parameters:**

wBuf	[Out]	The starting address of the data buffer (WORD format) Space must be allocated for the buffer and the address sent into the function. The function will input the data into the buffer. The data can then be analyzed from the buffer after calling this function. Buffer size = Total Channels * wNumPerCh * sizeof(WORD)
wNumPerCh	[In]	Number of A/D conversions that will be performed for each channel.

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.6.6 A812_ChScan_PollingFloat

■ **Description:**

This subroutine will perform a number of A/D conversions using a polling approach. After retrieving the data from the channel, it then reads the Channel Scan list buffer to move to the next channel and sets the specified configuration code. Once the A/D conversion is complete, the A/D data is stored in a buffer in float format.

Before calling this function, must be called the A812_ChScan_Clear() and A812_ChScan_Add() functions to set up the Channel Scan list buffer. Please refer to Section 3.6.1 for more information.

■ **Syntax:**

WORD **A812_ChScan_PollingFloat** (WORD **wJump10v**, float **fBuf[]**, WORD **wNumPerCh**);

■ **Parameters:**

wJump10v	[In]	Depends on the value chosen for JP4. 0→ +/- 5V; 1 +/- 10V
fBuf	[Out]	The starting address of the data buffer (WORD format) Space must be allocated for the buffer and the address sent into the function. The function will input the data into the buffer. The data can then be analyzed from the buffer after calling this function. Buffer size = Total Channels * wNumPerCh * sizeof(WORD)
wNumPerCh	[In]	Number of A/D conversions that will be performed for each channel.

■ **Returns:**

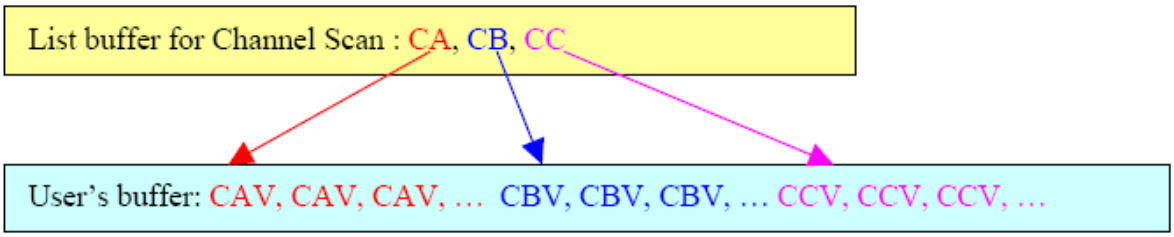
Please refer to error codes in Sec. 2.2 for more detailed information.

3.7 AD Interrupt and Channel Scan Functions

3.7.1 Introduction

The user can specify the numbers of channels to be read into the list buffer. The other function will perform the A/D conversion to retrieve the data, and then read the list buffer to move to the next channel and set the specified configuration code.

The data will be stored into the following manner:

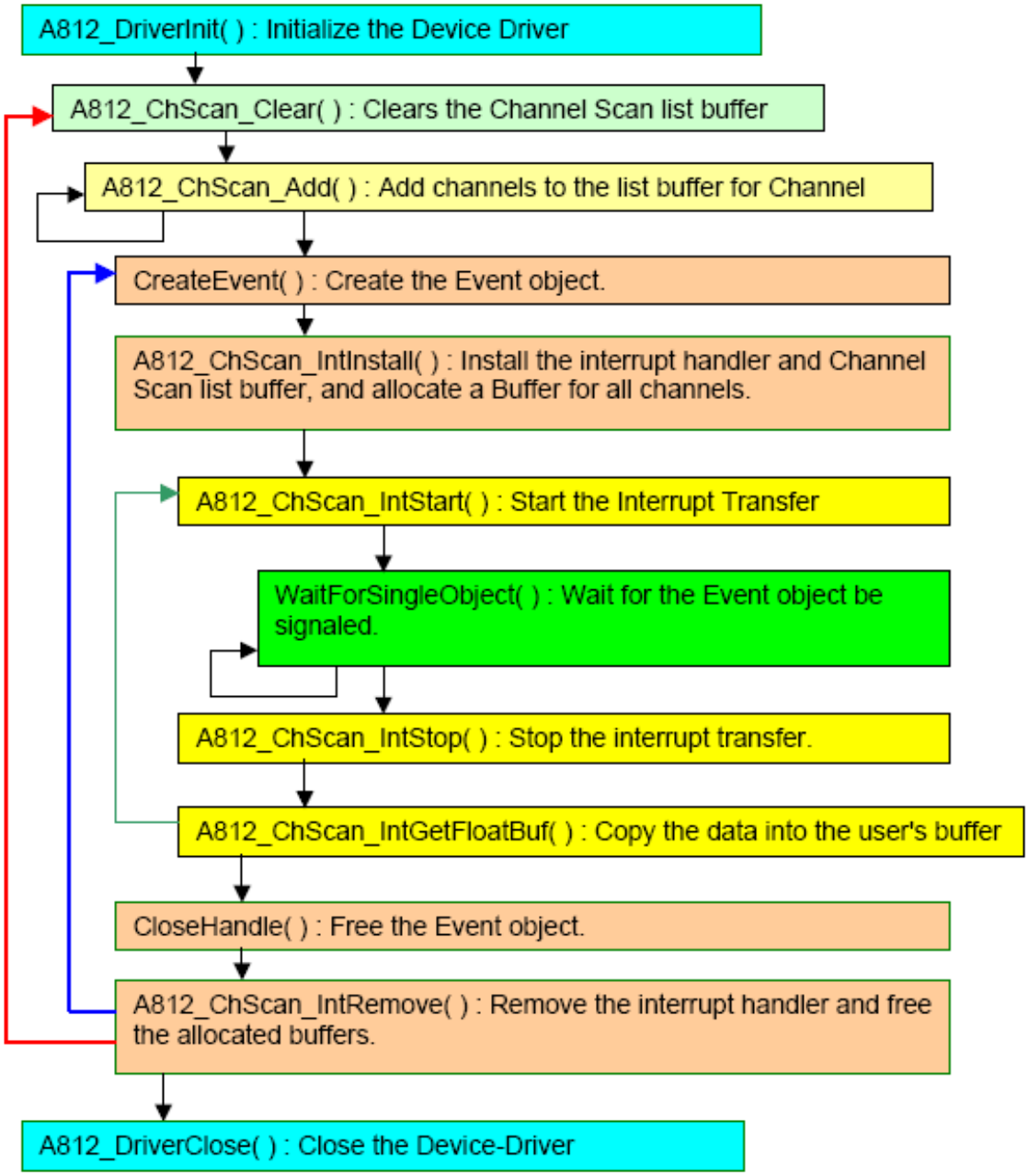


Note:
CA= Channel A; CB= Channel B; CC= Channel C
CAV= The value of Channel A; CBV= The value Channel B;
CCV= The value Channel C

After setting the next channel and the specified configuration code, there is slight to allow delay for settling time before the next A/D conversion. However, the interrupt service routinue doesn't account for the settling time delay. Thus, in order to retrieve the correct A/D conversion data, the interrupt of the sampling rate has to be slowed down.

The sampling rate is the same all channels, for example:
The list buffer for the Channel Scan is set to channel 2 and channel 0. The sampling rate is set to 10KHz. In reality, the both cannel 2 and channel 0 have a sampling-rate 5KHz.

The program architecture is as follows:



3.7.2 A812_ChScan_IntInstall

■ **Description:**

This subroutine will install the interrupt handler, copy the Channel Scan list buffer into the kernel mode driver and allocate a buffer for each channel. Before installing the interrupt, must be called "A812_ChScan_Clear()" and "A812_ChScan_Add()" functions to set up the Channel Scan list buffer. For more detailed information about using interrupts, please refer to the Introduction in Section 3.7.1.

■ **Syntax:**

WORD **A812_ChScan_IntInstall**(HANDLE ***hEvent**, DWORD **dwNumPerCh**);

■ **Parameters:**

***hEvent**	[In]	The Event handler that is created by the user.
dwNumPerCh	[In]	The desired A/D count for each channel to transfer.

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.3 A812_ChScan_IntStart

■ **Description:**

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

■ **Syntax:**

WORD **A812_ChScan_IntStart**(WORD **C1**, WORD **C2**, WORD **wJump10v**);

■ **Parameters:**

C1, C2	[In]	The sampling rate is 2M/(C1*C2) C1 → Counter1; C2 → Counter2
wJump10v	[In]	Depends on the value chosen for JP4. 0 → +/-5V; 1 → +/- 10V

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.4 A812_ChScan_IntStop

■ **Description:**

This subroutine will stop the interrupt transfer.

■ **Syntax:**

WORD **A812_ChScan_IntStop**(void);

■ **Parameters:**

void

■ **Returns:**

Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.5 A812_ChScan_IntRemove

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

- **Syntax:**
  WORD **A812_ChScan_IntRemove**(void);

- **Parameters:**
  Void

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.6 A812_ChScan_IntGetCount

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

- **Syntax:**
  WORD **A812_IntGetCount**(DWORD ***dwVal**);

- **Parameters:**

dwVal	[Out]	Returns the interrupt transfer count.

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.7 A812_ChScan_IntGetHexBuf

- **Description:**
  This subroutine will copy the transferred interrupt data into the user buffer.

- **Syntax:**
  WORD **A812_ChScan_IntGetHexBuf**(WORD ***wBuf[]**);

- **Parameters:**

wBuf	[Out]	The starting address of the data buffer (in WORD format) Space must be allocated for the buffer and the address sent into the function. The function will input the data into the buffer. The data can then be analyzed from the buffer after calling this function. Buffer size = Total Channels * wNumPerCh * sizeof(WORD)

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.7.8 A812_ChScan_IntGetFloatBuf

- **Description:**
  This subroutine will copy the transferred interrupt data into the user buffer.

- **Syntax:**
  WORD **A812_ChScan_IntGetFloatBuf**(WORD **fBuf[]**);

- **Parameters:**

fBuf	[Out]	The address of the fBuf (in float 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 cans analyze these data from the buffer after calling this function. Buffer size = Total Channels * dwNumPerCh * sizeof(float)

- **Returns:**
  Please refer to error codes in Sec. 2.2 for more detailed information.

3.8 Timer, Counter Function

3.8.1 A812_SetCounter

■ **Description:**

This subroutine is used to set the channel number, operation and count value of the 8254 timer chip. For detailed programming information regarding the 8254 chip, please refer to Intel's "Microsystem Components Handbook."

■ **Syntax:**

void **A812_SetCounter** (WORD **wCounterNo**, WORD **bCounterMode**, DWORD **wCounterValue**);

■ **Parameters:**

wCounterNo	[In]	The channel unmber of the 8254. Valid parameters: 0/1/2.
wCounterMode	[In]	The 8254 Counter-Mode: 0 to 5. Mode 0: Interrupt on Terminal Count Mode 1: Hardware Retriggerable One-Shot Mode 2: Rate Generator Mode 3: Square Wave Mode Mode 4: Software Triggered Mode Mode 5: Hardware Retriggerable Strobe (Retriggerable)
wCounterValue	[In]	The count value

■ **Returns:**

None

3.8.2 A812_ReadCounter

■ **Description:**

This subroutine is used to set the channel number, operation and count value of the 8254 timer chip. For detailed programming information regarding the 8254 chip, please refer to Intel's "Microsystem Components Handbook."

■ **Syntax:**

DWORD **A812_ReadCounter**(WORD **wCounterNo,** WORD **bCounterMode**);

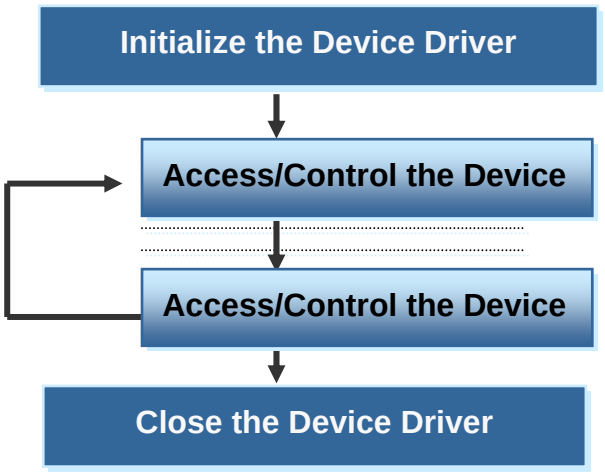
■ **Parameters:**

wCounterNo	[In]	The channel unmber of the 8254. Valid parameters: 0/1/2.
wCounterMode	[In]	The 8254 Counter-Mode: 0 to 5. Mode 0: Interrupt on Terminal Count Mode 1: Hardware Retriggerable One-Shot Mode 2: Rate Generator Mode 3: Square Wave Mode Mode 4: Software Triggered Mode Mode 5: Hardware Retriggerable Strobe (Retriggerable)

■ **Returns:**

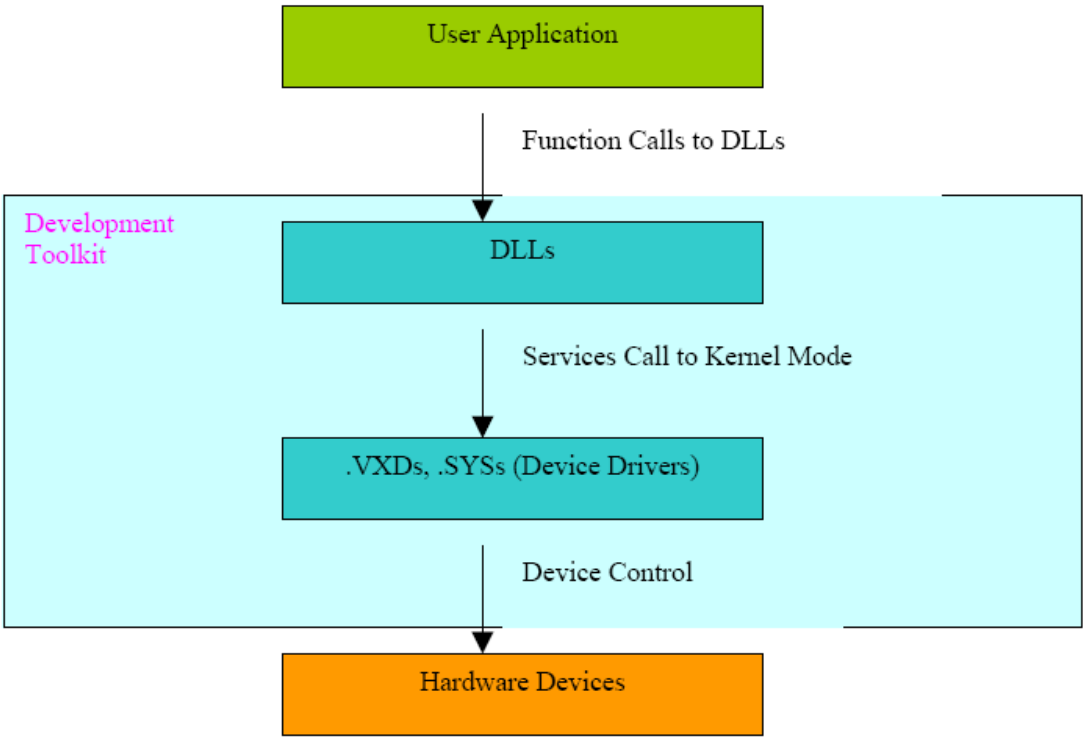
None

4. Program Architecture




```
A812_DriverInit()
….
    A812_InputByte( … )
    ……..
    ……..
    A812_OutputByte(…)
…..

A812_DriverClose()
```



5. Reporting Problems

Technical support is available at no charge as described below. The best way to report problems is to send electronic mail to Service@icpdas.com or Service.icpdas@ gmail.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 **platform** that you using? For example, Windows 98, Windows 2000 or 32-bit Windows XP/2003/Vista/2008/7.
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 test 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 had received your comments, we will take about two business days to test the problems that you said. And then reply as soon as possible to you. Please check that if we had received you comments? And please keeps contact with us.

E-mail: Service@icpads.com
Service.icpdas@ gmail.com

Web Site: http://www.icpdas.com
http://www.icpdas.com.tw