

User's Guide



Shop online at

www.omega.com

e-mail: info@omega.com



**OME-A-8111 ISA-Bus
Multi-Functional Board
Software Manual**



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-8111

Software Manual

[Windows 95/98/NT/2000/XP]

Table of Contents

1.	Hardware Guide	4
1.1	OME-A-8111	4
1.2	Range Configuration Code	4
2.	Introduction.....	5
2.1	Software Installation	6
2.2	References.....	9
3.	CALL DLLs & Demos	10
3.1	Using Visual C++.....	11
3.2	Using MFC.....	12
3.2.1	VC++ Demo Result	13
3.2.2	A8111.h for VC++	14
3.3	Using Visual Basic	18
3.3.1	VB Demo Result:.....	18
3.3.2	A8111.BAS	19
3.4	Using Delphi	23
3.4.1	Delphi Demo Result :	23
3.4.2	A8111.PAS	24
3.5	Using Borland C++ Builder	30
3.5.1	Borland C++ Builder Demo Result :	30
3.5.2	A8111.H	31
4.	Software Function Guide	35
4.1	Test Functions	37
4.1.1	A8111_SHORT_SUB_2	37
4.1.2	A8111_FLOAT_SUB_2.....	38
4.1.3	A8111_Get_DLL_Version	39
4.1.4	A8111_GetDriverVersion.....	39
4.2	Driver Functions	40
4.2.1	A8111_DriverInit	40
4.2.2	A8111_DriverClose.....	40
4.2.3	A8111_DELAY.....	41
4.2.4	A8111_Check_Address.....	41
4.3	DI/DO Functions.....	42
4.3.1	A8111_DI.....	42
4.3.2	A8111_DO	42
4.3.3	A8111_OutputByte	43
4.3.4	A8111_OutputWord	43

4.3.5	A8111_InputByte	44
4.3.6	A8111_InputWord	44
4.4	A/D and D/A Functions	45
4.4.1	A8111_AD_Hex	45
4.4.2	A8111_AD_Float	46
4.4.3	A8111_ADs_Hex	47
4.4.4	A8111_ADs_Float	48
4.4.5	A8111_ADsPacer_Float	49
4.4.6	A8111_AD2F	50
4.4.7	A8111_DA	51
4.4.8	A8111_Uni5_DA	52
4.4.9	A8111_Uni10_DA	53
4.5	Interrupt Functions	54
4.5.1	A8111_Int_Install	54
4.5.2	A8111_Int_Start	55
4.5.3	A8111_Int_Remove	56
4.5.4	A8111_Int_GetCount	56
4.5.5	A8111_Int_GetBuffer	57
4.5.6	A8111_Int_GetFloatBuffer	58
4.5.7	IRQ-mode conceptual diagram	59
5.	Program Architecture	61
6.	Demo List	62
6.1	Visual C++ Demo	62
6.2	Visual Basic Demo	62
6.3	Delphi Demo	63
6.4	Borland C++ Builder Demo	63
7.	Report Problems	64

1. Hardware Guide

1.1 OME-A-8111

The OME-A-8111 is a multifunction 12-bit resolution A/D, D/A, and digital I/O card. The features of the OME-A-8111 are given as below:

- A/D=12 bits, 8 channels(differential)
- OME-A-8111 : gain 1/2/4/8/16
- D/A=12 bits, 1 channel, 0-5V or 0-10V output by **hardware JP1 setting**
- 16 channels TTL compatible digital input
- 16 channels TTL compatible digital output

1.2 Range Configuration Code

The AD converter of A8111 is 12 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, using configuration 0/1/2/3/4 (for A8111) will get nearly the same result **except resolution.** **So choose the correct configuration code can achieve the highest precision measurement.**

OME-A-8111 Input Signal Range Configuration Code Table

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

2. Introduction

The board is provided with a collection of DLLs for Windows 95/98/NT application. These 32-bit DLLs can be called by Visual C/C++, Borland C++, Visual BASIC, Delphi or LabVIEW.

The installation provides these DLLs and device driver:

For Windows 95/98

- A8111.dll → Libraries
- A8111.Vxd → Device driver for Windows 95/98

For Windows NT

- A8111.dll → Libraries
- A8111.sys → Device driver for Windows NT
- Napwnt.sys → Device driver for Windows NT

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

- Software version
- Initialization
- Digital Input/Output
- A/D and D/A conversion

2.1 Software Installation

A). The CD-ROM Disk Contents:

```
--Win95
|  |--\README.TXT
|  |--\SETUP
|      |--\Setup.exe    ← the Setup program for WINDOWS 95/98
|
|--WinNT
    |--\README.TXT
    |--\SETUP
        |--\Setup.exe    ← the Setup program for Windows NT
```

B). Installation Steps

Step 1). Insert CD into CD-ROM.

Step 2). Clicking Start/Run in the task Bar

Step 3). Enter the path as:

E:\Win95\Setup\Setup.exe (for Win 95/98, CD-ROM drive is E:)

E:\WinNT\Setup\Setup.exe (for Win NT, CD-ROM drive is E:)

Step 4). Follow the instructions in installation process.

After installation is complete,

Windows 95/98:

The A8111.DLL, A8111.Vxd will be copied into C:\Windows\SYSTEM.

Windows NT:

The A8111.DLL will be copied into C:\WinNT\system32,

The A8111.sys and Napwnt.sys will be copied into

C:\WinNT\system32\drivers

C) After the installation, the sub-directory tree should be as follows:

For Windows 95/98:

[Base Directory] ← the directory you selected to setup

```
--\Demo
|   |--\VB5           ← demo code of Visual Basic 5.0
|   |--\VC5           ← demo code of Visual C++ 5.0
|   |--\Delphi3       ← demo code of Delphi 3.0
|   |--\BCB3          ← demo code of Borland C++ Builder 3.0
|
|--\Diag
|   |--\Diag.exe      ← Diagnostic Program
|
|--\Driver             ← Device driver & DLL for Win95/98
|   |--\A8111.dll     ← Dynamic linking library for A8111
|   |--\A8111.Vxd    ← Virtual Device Driver for Win95/98
|
|--\Manual             ← User's manual
|   |--\A8111sof.pdf  ← the software manual
|   |--\CallDLL.pdf   ← Call the DLL functions with VB5, Delphi3
|   |                 BCB3 and VC 5.
|   |--\ResCheck.pdf ← Check the resource (I/O Port, IRQ, DMA)
|   |                 under Windows 95/98 and NT
|   |--\SoftInst.pdf  ← Install the software
|
|--\readme.txt
```

For Windows NT:

[Base Directory] ← the directory you selected to setup

```

|--\Demo
|   |--\VB5           ← demo code of Visual Basic 5.0
|   |--\VC5           ← demo code of Visual C++
|   |--\Delphi3       ← demo code of Delphi 3.0
|   |--\BCB3          ← demo code of Borland C++ Builder 3.0
|
|--\Diag
|   |--\Diag.exe      ← Diagnostic Program
|
|--\Driver             ← Device driver & DLL for Win NT
|   |--\A8111.dll     ← dynamic linking library for A8111
|   |--\A8111.sys     ← Virtual Device Driver for Win NT
|   |--\NapWnt.sys   ← Virtual Device Driver for Win NT
|
|--\Manual             ← User's manual
|   |--\A8111sof.pdf  ← the software manual
|   |--\CallDLL.pdf   ← Call the DLL functions with VB5, Delphi3
|   |                 BCB3 and VC 5.
|   |--\ResCheck.pdf  ← Check the resource (I/O Port, IRQ, DMA)
|   |                 under Windows 95/98 and NT
|   |--\SoftInst.pdf  ← Install the software
|
|--\readme.txt

```

Note:

These VC++ demos are developed with VC++ 5.0. Please setup the environment exactly. Then using the NMAKE.EXE to compiling/linking the demo code. For examples: `\DEMO\VC5\nmake /f demo1.mak`

2.2 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.

3. CALL DLLs & Demos

Please refer to user manual "[CallDLL.pdf](#)".

For Windows 95/98:

```
|--\Driver
| |--\A8111.DLL      <-- Dynamic Linking Library
| |--\A8111.sys     <-- device driver
| |--\Napwnt.sys    <-- device driver
| |
| |--\BCB           <-- for Borland C++ Builder
| | |--\A8111.h     <-- Header file
| | |--\A8111.Lib   <-- Import Library for BCB only
| |
| |--\Delphi        <-- for Delphi
| | |--\A8111.Pas   <-- Declaration file
| |
| |--\VB            <-- for Visual Basic
| | |--\A8111.BAS   <-- Declaration file
| |
| |--\VC            <-- for Visual C++
| | |--\A8111.h     <-- Header file
| | |--\A8111.Lib   <-- Import Library for VC only
```

For Windows NT:

```
|--\Driver
| |--\A8111.DLL      <-- Dynamic Linking Library
| |--\A8111.sys     <-- device driver
| |--\Napwnt.sys    <-- device driver
| |
| |--\BCB           <-- for Borland C++ Builder
| | |--\A8111.h     <-- Header file
| | |--\A8111.Lib   <-- Import Library for BCB only
```

```
| |
| |--\Delphi      <-- for Delphi
| | |--A8111.Pas  <-- Declaration file
| |
| |--VB          <-- for Visual Basic
| | |--A8111.BAS <-- Declaration file
| |
| |--VC          <-- for Visual C++
|   |--A8111.h   <-- Header file
|   |--A8111.Lib <-- Import Library for VC only
```

3.1 Using Visual C++

The demo program given in the companion floppy disk is designed with C language. It is tested under Windows 95/98/NT and Visual C++ 5.0 compiler. The key points to remember are given below:

1. Make sure the PATH include the Visual C++ compiler
2. Execute the \MSDEV\BIN\VCVARS32.BAT to setup the environment.
The VCVARS32.BAT is provided by Visual C++.
3. The source program must include "A8111.H"
4. Copy the A8111.LIB, import library, to the directory where same as source program.
5. Copy the A8111.DLL, to the directory where same as source program
6. Edit the source program (refer to \Demo\VC5\DEMO1.C)
7. (refer to \Demo\VC5\DEMO1.C)
8. Edit the NMAKE file (refer to \Demo\VC5\DEMO1.MAK)
(refer to \Demo\VC5\DEMO1.MAK)
9. NMAKE /f DEMO1.MAK
10. Execute DEMO1.EXE

NOTE: The A8111.lib is used in linking time and the A8111.DLL is used in run time.

3.2 Using MFC

The usage of A8111 for MFC user is very similar to that for C user. It tests OK under Windows 95/98/NT and Visual C++ 5.0. The key points are given as below:

1. Use MFC wizard to create source code
2. The source program must include "A8111.H"
3. Copy the A8111.LIB, import library, to the directory same as source program
4. Copy the A8111.DLL, to the directory same as source program
5. Select **Build/Settings/Link** and key "A8111.lib" in the **object/library modules** field

NOTE: The A8111.lib is used in linking time and the A8111.DLL is used in run time.

3.2.1 VC++ Demo Result

```
■ TESTA8111 = [BASE:220] [IRQ:4]
NOW --> Base=220, IRQ=4
Driver Version=101
-----
Now Setting Is --> Base=220, DMA=1, IRQ=4
1. DLL Version=110
2. Find a A8111 in IOPORT_220
3. SHORT_SUB_2(1,2) = -1
4. FLOAT_SUB_2(1.0,2.0) = -1.000000
5. DO=0x55aa --> DI=55aa
It take 0.043 Second to run 10,000 time A8111_DO().
The performance of A8111_DO() is 232.56 KHz.
6. DA_0=0x800 --> AD_0=0.092819
8. A8111_ADs_Hex TEST :0.217391
9. A8111_ADs_Float TEST :0.392037
6. Install Irq Ok
7. The Interrupt sampling rate: 10.00kHz
7a6 7b6 7ca 7e2 7fa
```



Fig. 1.

3.2.2 A8111.h for VC++

```

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

/***** DEFINE A8111 RELATIVE ADDRESS *****/
#define A8111_TIMER0          0x00
#define A8111_TIMER1          0x01
#define A8111_TIMER2          0x02
#define A8111_TIMER_MODE     0x03
#define A8111_AD_LO           0x04 /* Analog to Digital, Low Byte */
#define A8111_AD_HI           0x05 /* Analog to Digital, High Byte */
#define A8111_DA_CH0_LO       0x04 /* Digit to Analog, CH 0 */
#define A8111_DA_CH0_HI       0x05
#define A8111_DA_CH1_LO       0x06 /* Digit to Analog, CH 1 */
#define A8111_DA_CH1_HI       0x07
#define A8111_DI_LO           0x06 /* Digit Input */
#define A8111_DO_LO           0x0D /* Digit Output */

#define A8111_CLEAR_IRQ       0x08
#define A8111_SET_GAIN        0x09
#define A8111_SET_CH          0x0A
#define A8111_SET_MODE        0x0B
#define A8111_SOFT_TRIG       0x0C

#define A8111_POLLING_MODE    1
#define A8111_DMA_MODE        2
#define A8111_INTERRUPT_MODE  6

/**** define the gain mode ****/
#define A8111_BI_1            0
#define A8111_BI_2            1
#define A8111_BI_4            2

```


#define A8111_BI_8 3
#define A8111_BI_16 4

```

#define A8111_NoError          0
#define A8111_DriverOpenError  1
#define A8111_DriverNoOpen    2
#define A8111_GetDriverVersionError 3
#define A8111_InstallIrqError  4
#define A8111_ClearIntCountError 5
#define A8111_GetIntCountError  6
#define A8111_GetBufferError    7
#define A8111_AdError1          100
#define A8111_AdError2          -200.0
#define A8111_InstallBufError   10
#define A8111_AllocateMemoryError 11
#define A8111_TimeoutError      0xffff
#define A8111_OtherError        13

```

// Function of Test

```

EXPORTS short CALLBACK A8111_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK A8111_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD  CALLBACK A8111_Get_DLL_Version(void);
EXPORTS WORD  CALLBACK A8111_GetDriverVersion(
    WORD *wDriverVersion);

```

// Function of DI/DO

```

EXPORTS WORD  CALLBACK A8111_DI(WORD wBase);
EXPORTS void  CALLBACK A8111_DO(WORD wBase, WORD wHexValue);
EXPORTS void  CALLBACK A8111_OutputByte(WORD wPortAddr,
    UCHAR bOutputVal);
EXPORTS void  CALLBACK A8111_OutputWord(WORD wPortAddr,
    WORD wOutputVal);
EXPORTS WORD  CALLBACK A8111_InputByte(WORD wPortAddr);
EXPORTS WORD  CALLBACK A8111_InputWord(WORD wPortAddr);

```

// Function of AD/DA

```

EXPORTS WORD  CALLBACK A8111_AD_Hex(WORD wBase,
    WORD wChannel, WORD wGainCode);
EXPORTS WORD  CALLBACK A8111_ADs_Hex(WORD wBase,
    WORD wChannel, WORD wGainCode , WORD wBuf[], WORD wCount);

```

```

EXPORTS float CALLBACK A8111_AD_Float(WORD wBase,
    WORD wChannel, WORD wGainCode);
EXPORTS WORD CALLBACK A8111_ADs_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, float fBuf[], WORD wCount);
EXPORTS WORD CALLBACK A8111_ADsPacer_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, float fBuf[], WORD wCount,
    WORD counter1, WORD counter2);
EXPORTS float CALLBACK A8111_AD2F(WORD wHex,
    WORD wGainCode );
EXPORTS void CALLBACK A8111_DA(WORD wBase, WORD wHexValue);
EXPORTS void CALLBACK A8111_Uni5_DA(WORD wBase, float fValue);
EXPORTS void CALLBACK A8111_Uni10_DA(WORD wBase, float fValue);

// Function of Driver
EXPORTS WORD CALLBACK A8111_DriverInit(void);
EXPORTS void CALLBACK A8111_DriverClose(void);
EXPORTS WORD CALLBACK A8111_DELAY(WORD wBase,
    WORD wDownCount);
EXPORTS WORD CALLBACK A8111_Check_Address(WORD wBase);

// Function of Interrupt
EXPORTS WORD CALLBACK A8111_Int_Install(WORD wBase,
    WORD wlrq, DWORD dwCount);
EXPORTS WORD CALLBACK A8111_Int_Start(WORD wChannel,
    WORD wGainCode, WORD wCounter1, WORD wCounter2);
EXPORTS WORD CALLBACK A8111_Int_Remove(void);
EXPORTS WORD CALLBACK A8111_Int_GetCount(DWORD *dwVal);
EXPORTS WORD CALLBACK A8111_Int_GetBuffer(DWORD dwNum,
    WORD wBuffer[]);
EXPORTS WORD CALLBACK A8111_Int_GetFloatBuffer(DWORD dwNum,
    float fBuffer[]);

```

3.3 Using Visual Basic

3.3.1 VB Demo Result:

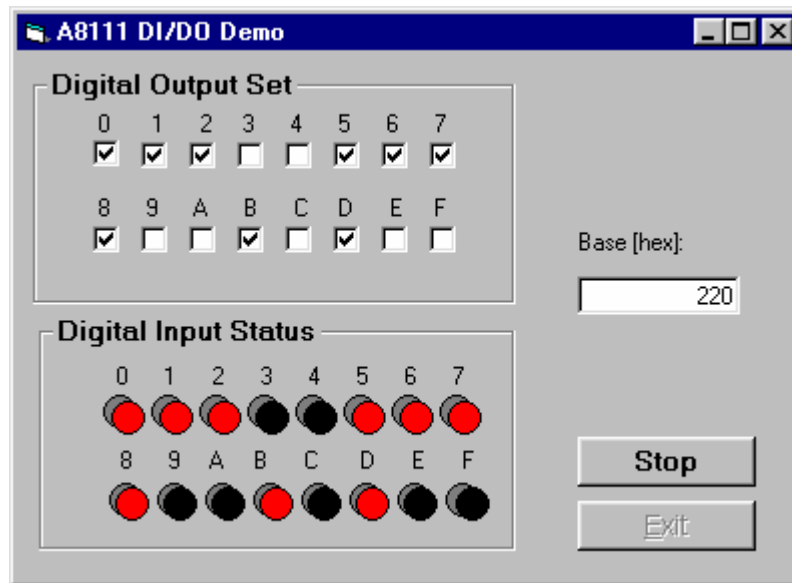


Fig. 2.

3.3.2 A8111.BAS

' The Declare of A8111.DLL for A8111 DAQ Card

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

***** DEFINE A8111 RELATIVE ADDRESS *****

```

Global Const A8111_TIMER0      = &H0
Global Const A8111_Timer1     = &H1
Global Const A8111_TIMER2     = &H2
Global Const A8111_TIMER_MODE = &H3
Global Const A8111_AD_LO      = &H4 ' Analog to Digital, Low Byte */
Global Const A8111_AD_HI      = &H5 ' Analog to Digital, High Byte */
Global Const A8111_DA_CH0_LO  = &H4 ' Digit to Analog, CH 0 */
Global Const A8111_DA_CH0_HI  = &H5
Global Const A8111_DA_CH1_LO  = &H6 ' Digit to Analog, CH 1 */
Global Const A8111_DA_CH1_HI  = &H7
Global Const A8111_DI_LO      = &H6 ' Digit Input */
Global Const A8111_DO_LO      = &HD ' Digit Output */

Global Const A8111_CLEAR_IRQ  = &H8
Global Const A8111_SET_GAIN   = &H9
Global Const A8111_SET_CH     = &HA
Global Const A8111_SET_MODE   = &HB
Global Const A8111_SOFT_TRIG  = &HC

Global Const A8111_POLLING_MODE = 1
Global Const A8111_DMA_MODE     = 2
Global Const A8111_INTERRUPT_MODE = 6

*** define the gain mode ***
Global Const A8111_BI_1         = 0
Global Const A8111_BI_2         = 1
Global Const A8111_BI_4         = 2
    
```

Global Const A8111_BI_8 = 3
 Global Const A8111_BI_16 = 4

Global Const A8111_NoError = 0
 Global Const A8111_DriverOpenError = 1
 Global Const A8111_DriverNoOpen = 2
 Global Const A8111_GetDriverVersionError = 3
 Global Const A8111_InstallIrqError = 4
 Global Const A8111_ClearIntCountError = 5
 Global Const A8111_GetIntCountError = 6
 Global Const A8111_GetBufferError = 7
 Global Const A8111_AdError1 = 100
 Global Const A8111_AdError2 = -200#
 Global Const A8111_InstallBufError = 10
 Global Const A8111_AllocateMemoryError = 11
 Global Const A8111_TimeoutError = &Hffff
 Global Const A8111_OtherError = 13

' Function of Test

Declare Function A8111_SHORT_SUB_2 Lib "A8111.DLL" (
 ByVal nA As Integer, ByVal nB As Integer) As Integer
 Declare Function A8111_FLOAT_SUB_2 Lib "A8111.DLL" (
 ByVal fA As Single, ByVal fB As Single) As Single
 Declare Function A8111_Get_DLL_Version Lib "A8111.DLL" () As Integer
 Declare Function A8111_GetDriverVersion Lib "A8111.DLL" (
 wDriverVersion As Integer) As Integer

' Function of DI/DO

Declare Function A8111_DI Lib "A8111.DLL" (ByVal wBase As Integer)
 As Integer
 Declare Sub A8111_DO Lib "A8111.DLL" (ByVal wBase As Integer,
 ByVal wHexValue As Integer)
 Declare Sub A8111_OutputByte Lib "A8111.DLL" (
 ByVal wPortAddr As Integer, ByVal bOutputVal As Byte)
 Declare Sub A8111_OutputWord Lib "A8111.DLL" (
 ByVal wPortAddr As Integer, ByVal wOutputVal As Integer)
 Declare Function A8111_InputByte Lib "A8111.DLL" (
 ByVal wPortAddr As Integer) As Integer

```
Declare Function A8111_InputWord Lib "A8111.DLL" (  
    ByVal wPortAddr As Integer) As Integer
```

```
' Function of AD/DA
```

```
Declare Function A8111_AD_Hex Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal wChannel As Integer, ByVal wGainCode As Integer) As Integer
```

```
Declare Function A8111_ADs_Hex Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal wChannel As Integer, ByVal wGainCode As Integer,  
    wBuf As Integer, ByVal wCount As Integer) As Integer
```

```
Declare Function A8111_AD_Float Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal wChannel As Integer, ByVal wGainCode As Integer) As Single
```

```
Declare Function A8111_ADs_Float Lib "A8111.DLL" (  
    ByVal wBase As Integer, ByVal wChannel As Integer,  
    ByVal wGainCode As Integer, fBuf As Single,  
    ByVal wCount As Integer) As Integer
```

```
Declare Function A8111_ADsPacer_Float Lib "A8111.DLL" (  
    ByVal wBase As Integer, ByVal wChannel As Integer,  
    ByVal wGainCode As Integer, fBuf As Single, ByVal wCount As Integer,  
    ByVal counter1 As Integer, ByVal counter2 As Integer) As Integer
```

```
Declare Function A8111_AD2F Lib "A8111.DLL" (ByVal wHex As Integer,  
    ByVal wGainCode As Integer) As Single
```

```
Declare Sub A8111_DA Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal wHexValue As Integer)
```

```
Declare Sub A8111_Uni5_DA Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal fValue As Single)
```

```
Declare Sub A8111_Uni10_DA Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal fValue As Single)
```

```
' Function of Driver
```

```
Declare Function A8111_DriverInit Lib "A8111.DLL" () As Integer
```

```
Declare Sub A8111_DriverClose Lib "A8111.DLL" ()
```

```
Declare Function A8111_DELAY Lib "A8111.DLL" (ByVal wBase As Integer,  
    ByVal wDownCount As Integer) As Integer
```

```
Declare Function A8111_Check_Address Lib "A8111.DLL" (  
    ByVal wBase As Integer) As Integer
```

```
' Function of Interrupt
```

```
Declare Function A8111_Int_Install Lib "A8111.DLL" (ByVal wBase As Integer,
```

```
    ByVal wlrq As Integer, ByVal dwCount As Long) As Integer
Declare Function A8111_Int_Start Lib "A8111.DLL" (ByVal Ch As Integer,
    ByVal Gain As Integer, ByVal c1 As Integer, ByVal c2 As Integer)
    As Integer
Declare Function A8111_Int_Remove Lib "A8111.DLL" () As Integer
Declare Function A8111_Int_GetCount Lib "A8111.DLL" (dwVal As Long)
    As Integer
Declare Function A8111_Int_GetBuffer Lib "A8111.DLL" (
    ByVal dwNum As Integer, wBuffer As Integer) As Integer
Declare Function A8111_Int_GetFloatBuffer Lib "A8111.DLL" (
    ByVal dwNum As Integer, fBuffer As Single) As Integer
```


3.4 Using Delphi

3.4.1 Delphi Demo Result :

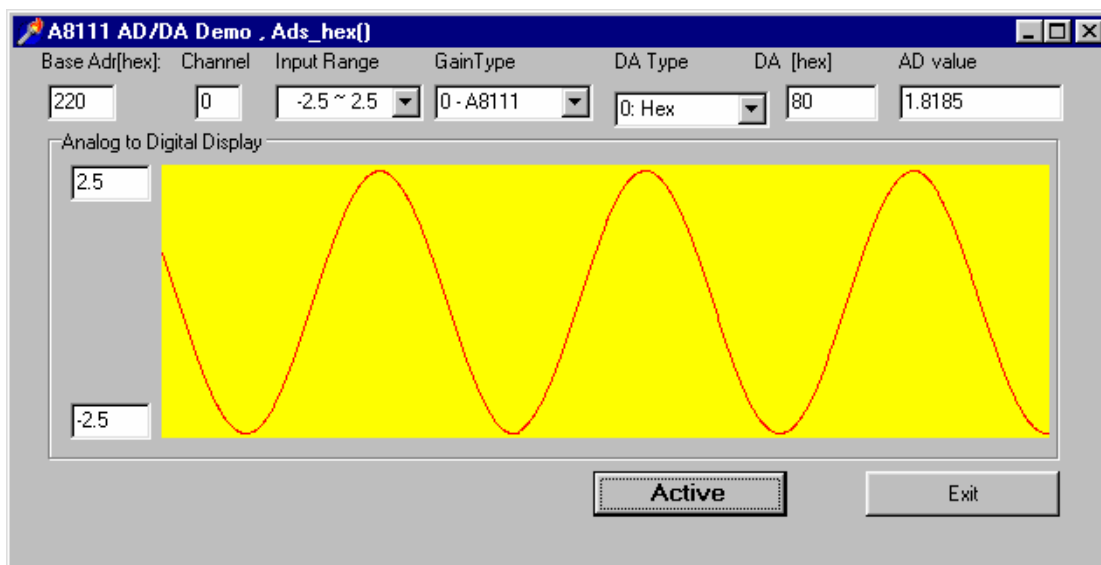


Fig. 4.

3.4.2 A8111.PAS

```
unit A8111;
```

```
interface
```

```
type PSingle=^Single;
```

```
type PWord=^Word;
```

```
const
```

```
A8111_TIMER0           = $00;
A8111_TIMER1           = $01;
A8111_TIMER2           = $02;
A8111_TIMER_MODE       = $03;
A8111_AD_LO            = $04; // Analog to Digital, Low Byte
A8111_AD_HI            = $05; // Analog to Digital, High Byte
A8111_DA_CH0_LO        = $04; // Digit to Analog, CH 0
A8111_DA_CH0_HI        = $05;
A8111_DA_CH1_LO        = $06; // Digit to Analog, CH 1
A8111_DA_CH1_HI        = $07;
A8111_DI_LO            = $06; // Digit Input
A8111_DO_LO            = $0D; // Digit Output
A8111_CLEAR_IRQ        = $08;
A8111_SET_GAIN          = $09;
A8111_SET_CH            = $0A;
A8111_SET_MODE          = $0B;
A8111_SOFT_TRIG        = $0C;
A8111_POLLING_MODE     = 1;
A8111_DMA_MODE          = 2;
A8111_INTERRUPT_MODE   = 6;
```

```
/** define the gain mode */
A8111_BI_1      = 0;
A8111_BI_2      = 1;
A8111_BI_4      = 2;
A8111_BI_8      = 3;
A8111_BI_16     = 4;
A8111_NoError   = 0;
A8111_DriverOpenError = 1;
A8111_DriverNoOpen = 2;
A8111_GetDriverVersionError = 3;
A8111_InstallIrqError = 4;
A8111_ClearIntCountError = 5;
A8111_GetIntCountError = 6;
A8111_GetBufferError = 7;
A8111_AdError1  = 100;
A8111_AdError2  = -200.0;
A8111_InstallBufError = 10;
A8111_AllocateMemoryError = 11;
A8111_TimeoutError = $ffff;
A8111_OtherError = 13;

// Function of Test
function A8111_SHORT_SUB_2(nA,nB:SmallInt):SmallInt; StdCall;
function A8111_FLOAT_SUB_2(fA,fB:Single):Single; StdCall;
function A8111_Get_DLL_Version:Word; StdCall;
function A8111_GetDriverVersion(var wDriverVersion:Word):Word; StdCall;

// Function of DI/DO
function A8111_DI(wBase:Word):Word; StdCall;
procedure A8111_DO(wBase,wHexValue:word); StdCall;
procedure A8111_OutputByte(wPortAddr:WORD;bOutputVal:Byte); StdCall;
procedure A8111_OutputWord( wPortAddr:WORD;
    wOutputVal:WORD ); StdCall;
function A8111_InputByte(wPortAddr:WORD):WORD; StdCall;
function A8111_InputWord(wPortAddr:WORD):WORD; StdCall;
```

// Function of AD/DA

```
function A8111_AD_Hex(wBase, wChannel, wGainCode:Word ):
    WORD; StdCall;
function A8111_ADs_Hex(wBase, wChannel, wGainCode:Word; Buf:PWord;
    wCount:Word):Word; StdCall;
function A8111_AD_Float(wBase, wChannel, wGainCode:Word):
    Single; StdCall;
function A8111_ADs_Float( wBase, wChannel, wGainCode:Word;
    fBuf:PSingle; wCount:Word):Word; StdCall;
function A8111_ADsPacer_Float(wBase, wChannel, wGainCode:Word;
    fBuf:PSingle; wCount:Word; counter1:WORD; counter2:WORD):
    Word; StdCall;
function A8111_AD2F(wHex,wGainCode:Word):Single; StdCall;
procedure A8111_DA(wBase,wHexValue:Word); StdCall;
procedure A8111_Uni5_DA(wBase:Word; fValue:Single); StdCall;
procedure A8111_Uni10_DA(wBase:Word;fValue:Single); StdCall;
```

// Function of Driver

```
function A8111_DriverInit:Word; StdCall;
procedure A8111_DriverClose; StdCall;
function A8111_DELAY(wBase,wDownCount:Word):Word; StdCall;
function A8111_Check_Address(wBase:Word):Word; StdCall;
```

// Function of Interrupt

```
function A8111_Int_Install(wBase,wlrq:Word;dwCount:LongInt):
    Word; StdCall;
function A8111_Int_Start(wChannel, wGainCode, wCounter1,
    wCounter2:Word): Word; StdCall;
function A8111_Int_Remove:Word; StdCall;
function A8111_Int_GetCount(var dwVal:LongInt):Word; StdCall;
function A8111_Int_GetBuffer(dwNum:LongInt; wBuffer:PWord):
    Word; StdCall;
function A8111_Int_GetFloatBuffer(dwNum:LongInt; fBuffer:PSingle):
    Word; StdCall;
```

implementation

```
function A8111_SHORT_SUB_2; external 'A8111.DLL' name
```

```
'A8111_SHORT_SUB_2';  
function A8111_FLOAT_SUB_2; external 'A8111.DLL' name  
'A8111_FLOAT_SUB_2';  
function A8111_Get_DLL_Version; external 'A8111.DLL' name  
'A8111_Get_DLL_Version';  
function A8111_GetDriverVersion; external 'A8111.DLL' name  
'A8111_GetDriverVersion';
```

```

procedure A8111_DO;           external 'A8111.DLL' name 'A8111_DO';
function A8111_DI;           external 'A8111.DLL' name 'A8111_DI';
procedure A8111_OutputByte;  external 'A8111.DLL' name
    'A8111_OutputByte';
procedure A8111_OutputWord;  external 'A8111.DLL' name
    'A8111_OutputWord';
function A8111_InputByte;    external 'A8111.DLL' name
    'A8111_InputByte';
function A8111_InputWord;    external 'A8111.DLL' name
    'A8111_InputWord';

function A8111_AD_Hex;       external 'A8111.DLL' name
    'A8111_AD_Hex';
function A8111_ADs_Hex;     external 'A8111.DLL' name
    'A8111_ADs_Hex';
function A8111_AD_Float;    external 'A8111.DLL' name
    'A8111_AD_Float';
function A8111_ADs_Float;   external 'A8111.DLL' name
    'A8111_ADs_Float';
function A8111_ADsPacer_Float; external 'A8111.DLL' name
    'A8111_ADsPacer_Float';
function A8111_AD2F;        external 'A8111.DLL' name
    'A8111_AD2F';
procedure A8111_DA;          external 'A8111.DLL' name 'A8111_DA';
procedure A8111_Uni5_DA;    external 'A8111.DLL' name
    'A8111_Uni5_DA';
procedure A8111_Uni10_DA;   external 'A8111.DLL' name
    'A8111_Uni10_DA';
function A8111_DriverInit;  external 'A8111.DLL' name
    'A8111_DriverInit';
procedure A8111_DriverClose; external 'A8111.DLL' name
    'A8111_DriverClose';
function A8111_DELAY;       external 'A8111.DLL' name
    'A8111_DELAY';
function A8111_Check_Address; external 'A8111.DLL' name
    'A8111_Check_Address';

```

```
function A8111_Int_Install;          external 'A8111.DLL' name
    'A8111_Int_Install';
function A8111_Int_Start;          external 'A8111.DLL' name
    'A8111_Int_Start';
function A8111_Int_Remove;        external 'A8111.DLL' name
    'A8111_Int_Remove';
function A8111_Int_GetCount;      external 'A8111.DLL' name
    'A8111_Int_GetCount';
function A8111_Int_GetBuffer;     external 'A8111.DLL' name
    'A8111_Int_GetBuffer';
function A8111_Int_GetFloatBuffer; external 'A8111.DLL' name
    'A8111_Int_GetFloatBuffer';
```

end.

3.5 Using Borland C++ Builder

3.5.1 Borland C++ Builder Demo Result :

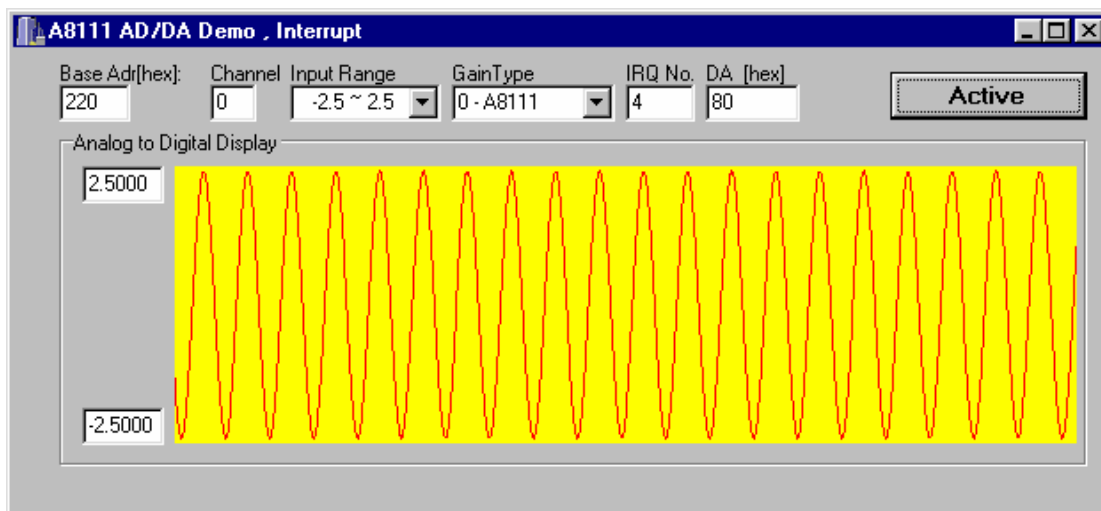


Fig. 6.

3.5.2 A8111.H

```

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

/***** DEFINE A8111 RELATIVE ADDRESS *****/
#define A8111_TIMER0          0x00
#define A8111_TIMER1          0x01
#define A8111_TIMER2          0x02
#define A8111_TIMER_MODE      0x03
#define A8111_AD_LO           0x04 /* Analog to Digital, Low Byte */
#define A8111_AD_HI           0x05 /* Analog to Digital, High Byte */
#define A8111_DA_CH0_LO       0x04 /* Digit to Analog, CH 0 */
#define A8111_DA_CH0_HI       0x05
#define A8111_DA_CH1_LO       0x06 /* Digit to Analog, CH 1 */
#define A8111_DA_CH1_HI       0x07
#define A8111_DI_LO           0x06 /* Digit Input */
#define A8111_DO_LO           0x0D /* Digit Output */

#define A8111_CLEAR_IRQ       0x08
#define A8111_SET_GAIN        0x09
#define A8111_SET_CH          0x0A
#define A8111_SET_MODE        0x0B
#define A8111_SOFT_TRIG       0x0C

#define A8111_POLLING_MODE    1
#define A8111_DMA_MODE        2
#define A8111_INTERRUPT_MODE  6

/**** define the gain mode ****/
#define A8111_BI_1            0
#define A8111_BI_2            1
#define A8111_BI_4            2

```

```

#define A8111_BI_8          3
#define A8111_BI_16       4

#define A8111_NoError      0
#define A8111_DriverOpenError  1
#define A8111_DriverNoOpen  2
#define A8111_GetDriverVersionError  3
#define A8111_InstallIrqError  4
#define A8111_ClearIntCountError  5
#define A8111_GetIntCountError  6
#define A8111_GetBufferError  7
#define A8111_AdError1      100
#define A8111_AdError2     -200.0
#define A8111_InstallBufError  10
#define A8111_AllocateMemoryError  11
#define A8111_TimeoutError   0xffff
#define A8111_OtherError    13

```

// Function of Test

```

EXPORTS short CALLBACK A8111_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK A8111_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD  CALLBACK A8111_Get_DLL_Version(void);
EXPORTS WORD  CALLBACK A8111_GetDriverVersion(
    WORD *wDriverVersion);

```

// Function of DI/DO

```

EXPORTS WORD  CALLBACK A8111_DI(WORD wBase);
EXPORTS void  CALLBACK A8111_DO(WORD wBase, WORD wHexValue);
EXPORTS void  CALLBACK A8111_OutputByte(WORD wPortAddr,
    UCHAR bOutputVal);
EXPORTS void  CALLBACK A8111_OutputWord(WORD wPortAddr,
    WORD wOutputVal);
EXPORTS WORD  CALLBACK A8111_InputByte(WORD wPortAddr);
EXPORTS WORD  CALLBACK A8111_InputWord(WORD wPortAddr);

```

// Function of AD/DA

```

EXPORTS WORD  CALLBACK A8111_AD_Hex(WORD wBase,
    WORD wChannel, WORD wGainCode);

```

```
EXPORTS WORD CALLBACK A8111_ADs_Hex(WORD wBase,
    WORD wChannel, WORD wGainCode , WORD wBuf[], WORD wCount);
EXPORTS float CALLBACK A8111_AD_Float(WORD wBase,
    WORD wChannel, WORD wGainCode);
EXPORTS WORD CALLBACK A8111_ADs_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, float fBuf[], WORD wCount);
EXPORTS WORD CALLBACK A8111_ADsPacer_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, float fBuf[], WORD wCount,
    WORD counter1, WORD counter2);
EXPORTS float CALLBACK A8111_AD2F(WORD wHex,WORD wGainCode );
EXPORTS void CALLBACK A8111_DA(WORD wBase, WORD wHexValue);
EXPORTS void CALLBACK A8111_Uni5_DA(WORD wBase, float fValue);
EXPORTS void CALLBACK A8111_Uni10_DA(WORD wBase, float fValue);

// Function of Driver
EXPORTS WORD CALLBACK A8111_DriverInit(void);
EXPORTS void CALLBACK A8111_DriverClose(void);
EXPORTS WORD CALLBACK A8111_DELAY(WORD wBase,
    WORD wDownCount);
EXPORTS WORD CALLBACK A8111_Check_Address(WORD wBase);
```

// Function of Interrupt

```
EXPORTS WORD CALLBACK A8111_Int_Install(WORD wBase,  
    WORD wIrq, DWORD dwCount);
```

```
EXPORTS WORD CALLBACK A8111_Int_Start(WORD wChannel,  
    WORD wGainCode, WORD wCounter1, WORD wCounter2);
```

```
EXPORTS WORD CALLBACK A8111_Int_Remove(void);
```

```
EXPORTS WORD CALLBACK A8111_Int_GetCount(DWORD *dwVal);
```

```
EXPORTS WORD CALLBACK A8111_Int_GetBuffer(DWORD dwNum,  
    WORD wBuffer[]);
```

```
EXPORTS WORD CALLBACK A8111_Int_GetFloatBuffer(DWORD dwNum,  
    float fBuffer[]);
```

4. Software Function Guide

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

- The test functions
- The driver functions
- The DI/DO functions
- The A/D and D/A conversion functions
- The IRQ-mode A/D and D/A conversion functions

The functions of test listing as follows:

1. A8111_SHORT_SUB_2
2. A8111_FLOAT_SUB_2
3. A8111_Get_DLL_Version
4. A8111_GetDriverVersion

The functions of driver listing as follows:

1. A8111_DriverInit
2. A8111_DriverClose
3. A8111_Check_Address
4. A8111_DELAY

The functions of DI/DO listing as follows:

1. A8111_DI
2. A8111_DO
3. A8111_InputByte
4. A8111_InputWord
5. A8111_OutputByte
6. A8111_OutputWord

The functions of A/D , D/A Conversion listing as follows:

1. A8111_AD_Hex
2. A8111_AD_Float
3. A8111_ADs_Hex
4. A8111_ADs_Float
5. A8111_AD2F
6. A8111_DA
7. A8111_Uni5_DA
8. A8111_Uni10_DA

The functions of IRQ-Mode , A/D , D/A Conversion listing as follows:

1. A8111_Int_Install
2. A8111_Int_Start
3. A8111_Int_Remove
4. A8111_Int_GetCount
5. A8111_Int_GetBuffer
6. A8111_Int_GetFloatBuffer

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

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

Note: All of the parameters need space allocation performed by the user.

4.1 Test Functions

4.1.1 A8111_SHORT_SUB_2

- **Description:**

Compute $C = nA - nB$ in **short** formats, **short = 16-bit signed integer.**
This function is provided for testing purpose.
- **Syntax:**

`short A8111_SHORT_SUB_2(short nA, short nB);`
- **Parameter:**

nA : [Input] short integer
nB : [Input] short integer
- **Return:**

return = $nA - nB \rightarrow$ short integer

4.1.2 A8111_FLOAT_SUB_2

- **Description:**

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

- **Syntax:**

float A8111_FLOAT_SUB_2(float fA, float fB);

- **Parameter:**

fA : [Input] floating point value

fB : [Input] floating point value

- **Return:**

return = fA - fB → floating point value

4.1.3 A8111_Get_DLL_Version

- **Description:**
Read the software version of the A8111.DLL.
- **Syntax:**
WORD A8111_Get_DLL_Version(void) ;
- **Parameter:**
void
- **Return:**
return=0x200 → Version 2.00 **(WORD = 16-bit unsigned integer)**

4.1.4 A8111_GetDriverVersion

- **Description:**
This subroutine will get the version number for the device driver.
- **Syntax:**
WORD A8111_GetDriverVersion(WORD *wDriverVersion) ;
- **Parameter:**
wDriverVersion : [Output] the address of wDriverVersion.
When wDriverVerion=0x210 → version 2.10
- **Return:**
A8111_NoError : Successful in opening the device driver
A8111_GetDriverVersionError : Failed opening the device driver.

4.2 Driver Functions

4.2.1 A8111_DriverInit

- **Description:**

This subroutine will open the device driver and allocate the resources for the device driver. This function must be called once before calling other functions.

- **Syntax:**

WORD A8111_DriverInit (void);

- **Parameter:**

void

- **Return:**

A8111_NoError : successful in opening the device driver
A8111_OpenError : failed in opening the device driver.

4.2.2 A8111_DriverClose

- **Description:**

This subroutine will close the device driver and release the resources.

- **Syntax:**

void A8111_DriverClose (void);

- **Parameter:**

void

- **Return:**

void

4.2.3 A8111_DELAY

This function has been reserved. The user cannot call this function.

4.2.4 A8111_Check_Address

- **Description:**

This subroutine will detect the OME-A-8111 at I/O base address = **wBase**. This subroutine will perform one A/D conversion, if successful → found an OME-A-8111.

- **Syntax:**

WORD A8111_Check_Address(WORD wBase);

- **Parameter:**

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

- **Return:**

A8111_NoError : Found an OME-A-8111 OK
A8111_TimeoutError : Operation failed

4.3 DI/DO Functions

4.3.1 A8111_DI

- **Description:**
This subroutine will read the 16 bits of data from the digital input port.
- **Syntax:**
WORD A8111_DI(WORD wBase);
- **Parameter:**
wBase : [Input] I/O port base address, for example, 0x220
- **Return:**
16-bit data read from the digital input port

4.3.2 A8111_DO

- **Description:**
This subroutine will send the 16 bits of data to the digital output port.
- **Syntax:**
void A8111_DO(WORD wBase, WORD wHexValue);
- **Parameter:**
wBase : [Input] I/O port base address, for example, 0x220
wHexValue : [Input] 16-bit data send to digital output port
- **Return:**
void

4.3.3 A8111_OutputByte

- **Description:**
This subroutine will send the 8 bits of data to the desired I/O port.
- **Syntax:**
void A8111_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
bOutputVal : [Input] 8-bit data send to I/O port
- **Return:**
void

4.3.4 A8111_OutputWord

- **Description:**
This subroutine will send the 16 bits of data to the desired I/O port.
- **Syntax:**
void A8111_OutputByte(WORD wPortAddr, WORD wOutputVal);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
wOutputVal : [Input] 16-bit data send to I/O port
- **Return:**
void

4.3.5 A8111_InputByte

- **Description:**
This subroutine will input the 8 bits of data from the desired I/O port.
- **Syntax:**
WORD A8111_InputByte(WORD wPortAddr);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
- **Return:**
16-bit data where the leading 8 bits are all 0

4.3.6 A8111_InputWord

- **Description:**
This subroutine will input the 16 bits of data from the desired I/O port.
- **Syntax:**
WORD A8111_InputWord(WORD wPortAddr);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
- **Return:**
16-bit data.

4.4 A/D and D/A Functions

4.4.1 A8111_AD_Hex

- **Description:**

This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for OME-A-8111. This subroutine will compute the result according to the **configuration code** (Section 1.2).

- **Syntax:**

```
WORD A8111_AD_Hex(WORD wBase, WORD wChannel,  
                  WORD wGainCode);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wChannel : [Input] A/D channel number,
wGainCode : [Input] Configuration codes, refer to Section 1.2.

- **Return:**

A8111_TimeoutError : A/D converter error (return 0xffff)
Other value : The **Hex value** of A/D conversion. (0x0 ~ 0x0fff)

4.4.2 A8111_AD_Float

- **Description:**

This subroutine will perform an A/D conversion by polling. The A/D converter is 12-bit for OME-A-8111. This subroutine will compute the result according to the **configuration code** (Section 1.2).

- **Syntax:**

```
float A8111_AD_Float(WORD wBase, WORD wChannel,  
                    WORD wGainCode);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wChannel : [Input] A/D channel number,
wGainCode : [Input] Configuration codes, refer to Section 1.2

- **Return:**

A8111_AdError2 : A/D converter error (return -200.0)
Other value : The **floating-point value** of A/D conversion

4.4.3 A8111_ADs_Hex

- **Description:**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A8111_AD_Hex except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D conversion can proceed at the ISA bus's max speed. After A/D conversion, the A/D data are stored in a buffer in Hex format. The **wBuf** is the starting address of this data buffer.

- **Syntax:**

```
WORD A8111_ADs_Hex(WORD wBase, WORD wChannel,
                  WORD wGainCode, WORD wBuf[], WORD wCount);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wChannel : [Input] A/D channel number
wGainCode : [Input] Configuration codes, refer to Section 1.2
wBuf : [Output] Starting address of the data buffer

The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.

wCount : [Input] Number of A/D conversions to be performed

- **Return:**

A8111_TimeoutError : A/D converter error (return 0xffff)
A8111_NoError : Operation was successful

4.4.4 A8111_ADs_Float

- **Description:**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to A8111_AD except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D conversion can proceed at the ISA bus's max speed. Then the A/D data is stored in a data buffer in Float format. The **fBuf** is the starting address of this data buffer.

- **Syntax:**

```
WORD A8111_ADs_Float(WORD wBase, WORD wChannel,
                    WORD wGainCode, float fBuf[], WORD wCount);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wChannel : [Input] A/D channel number
wGainCode : [Input] Configuration codes, refer to Section 1.2
fBuf : [Output] Starting address of the data buffer
(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 can analyze this data from the buffer after calling this function.

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

- **Return:**

A8111_TimeoutError : A/D converter error (return 0xffff)
A8111_NoError : Operation was successful

4.4.5 A8111_ADsPacer_Float

- **Description:**

This subroutine will perform a number of A/D conversions by Pacer Trigger + Software Transfer. This subroutine is very similar to A8111_Ads_Float() except that this subroutine will trigger the AD converter by the Pacer (Uses the counter1 and counter2) .The A/D data is stored in a data buffer in Float format. The **fBuf** is the starting address of this data buffer.

- **Syntax:**

```
WORD A8111_ADsPacer_Float(WORD wBase, WORD wChannel,
                          WORD wGainCode, float fBuf[], WORD wCount,
                          WORD counter1, WORD counter2);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wChannel : [Input] A/D channel number
wGainCode : [Input] Configuration codes, refer to Section 1.2
fBuf : [Output] Starting address of the data buffer
(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 can analyze this data from the buffer after calling this function.

wCount : [Input] Number of A/D conversions will be performed
counter1 : [Input] The value of Counter 1
counter2 : [Input] The value of Counter 2

(The sampling rate is : 2Mhz / (counter1 * counter 2))

- **Return:**

A8111_TimeoutError : A/D converter error (return 0xffff)
A8111_NoError : Operation was successful

4.4.6 A8111_AD2F

- **Description:**

This subroutine will convert the Hex value to Floating value depending on GainCode, Bipolar/Unipolar and 10v/20v settings.

- **Syntax:**

float ISO813_AD2F(WORD wHexValue, WORD wGainCode);

- **Parameter:**

wHexValue : [Input] Hex Value 0 ~ 0x0FFF
wGainCode : [Input] Refer to Sec. 1.2 for detailed information

- **Return:**

A8111_AdError2 : A/D converter error (-200.0)
Other value : The **floating-point value** of A/D conversion

4.4.7 A8111_DA

- **Description:**

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

- **Syntax:**

```
void A8111_DA(WORD wBase, WORD wHexValue);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
wHexValue : [Input] 12-bit data sent to D/A converter
(0x0 ~ 0x0fff)

- **Return:**

None

4.4.8 A8111_Uni5_DA

- **Description:**

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

- **Syntax:**

```
void A8111_Uni5_DA(WORD wBase, float fValue);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
fValue : [Input] Floating-point sent to D/A converter
(0.0v ~ 5.0v)

- **Return:**

None

4.4.9 A8111_Uni10_DA

- **Description:**

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

- **Syntax:**

```
void A8111_Uni10_DA(WORD wBase, float fValue);
```

- **Parameter:**

wBase : [Input] I/O port base address, for example, 0x220
fValue : [Input] Floating-point send to D/A converter
(0.0v ~ 10.v)

- **Return:**

None

4.5 Interrupt Functions

4.5.1 A8111_Int_Install

- **Description:**

This subroutine will install interrupt handler for a specific IRQ level n. For more detail information of using interrupt please refer to [Section 4.5.7 "IRQ-mode conceptual diagram"](#)

- **Syntax:**

```
WORD A8111_Int_Install(WORD wBase, WORD wlrq,  
                      DWORD dwCount );
```

- **Parameter:**

wBase : [Input] the I/O port base address for OME-A-8111 card.
wlrq : [Input] the IRQ level . (Valid numbers are: 2, 3, 4, 5, 6 or 7)
dwCount : [Input] the desired A/D entries count for interrupt transfer.

- **Return:**

A8111_NoError : successful
A8111_InstallIrqError : failed to install IRQ handler.

4.5.2 A8111_Int_Start

- **Description:**

This subroutine will start the interrupt transfer for a specific A/D channel at programmed gain code and sampling rate.

- **Syntax:**

WORD A8111_Int_Start(WORD wCh, WORD wGain, WORD c1,
WORD c2)

- **Parameter:**

wCh : [Input] the A/D channel.
wGain : [Input] the Gain Code, refer to sec 1.2.
c1,c2 : [Input] the sampling rate is $2M/(c1*c2)$

- **Return:**

A8111_NoError : successful
A8111_INTStartError : failed

4.5.3 A8111_Int_Remove

- **Description:**

This subroutine will stop the interrupt transfer and remove the installed interrupt handler.

- **Syntax:**

WORD A8111_Int_Remove(void)

- **Parameter:**

void.

- **Return:**

A8111_NoError : successful

A8111_INTStopError : failed

4.5.4 A8111_Int_GetCount

- **Description:**

This subroutine will read the transferred count of interrupt.

- **Syntax:**

WORD A8111_Int_GetCount(DWORD *dwVal)

- **Parameter:**

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

- **Return:**

A8111_NoError : successful

A8111_GetIntCountError : failed to get interrupt count.

4.5.5 A8111_Int_GetBuffer

- **Description:**

This subroutine will copy the transferred interrupted data into the user's buffer.

- **Syntax:**

WORD A8111_Int_GetBuffer(DWORD dwNum, WORD wBuffer[])

- **Parameter:**

dwNum : [Input] the entry no to transfer.

wBuffer : [Output] the address of wBuffer(WORD format)

The user must allocate spaces for this buffer and send the address into the function. This function will fill the data into this buffer. The user can analyze this data from the buffer after calling this function.

- **Return:**

A8111_NoError : successful

A8111_GetBufferError : failed

4.5.6 A8111_Int_GetFloatBuffer

- **Description:**

This subroutine will copy the transferred interrupted data into the user's buffer.

- **Syntax:**

WORD A8111_Int_GetFloatBuffer(DWORD dwNum, float fBuffer[])

- **Parameter:**

dwNum : [Input] the entry no to transfer.

fBuffer : [Output] the address of fBuffer(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 can analyze this data from the buffer after calling this function.

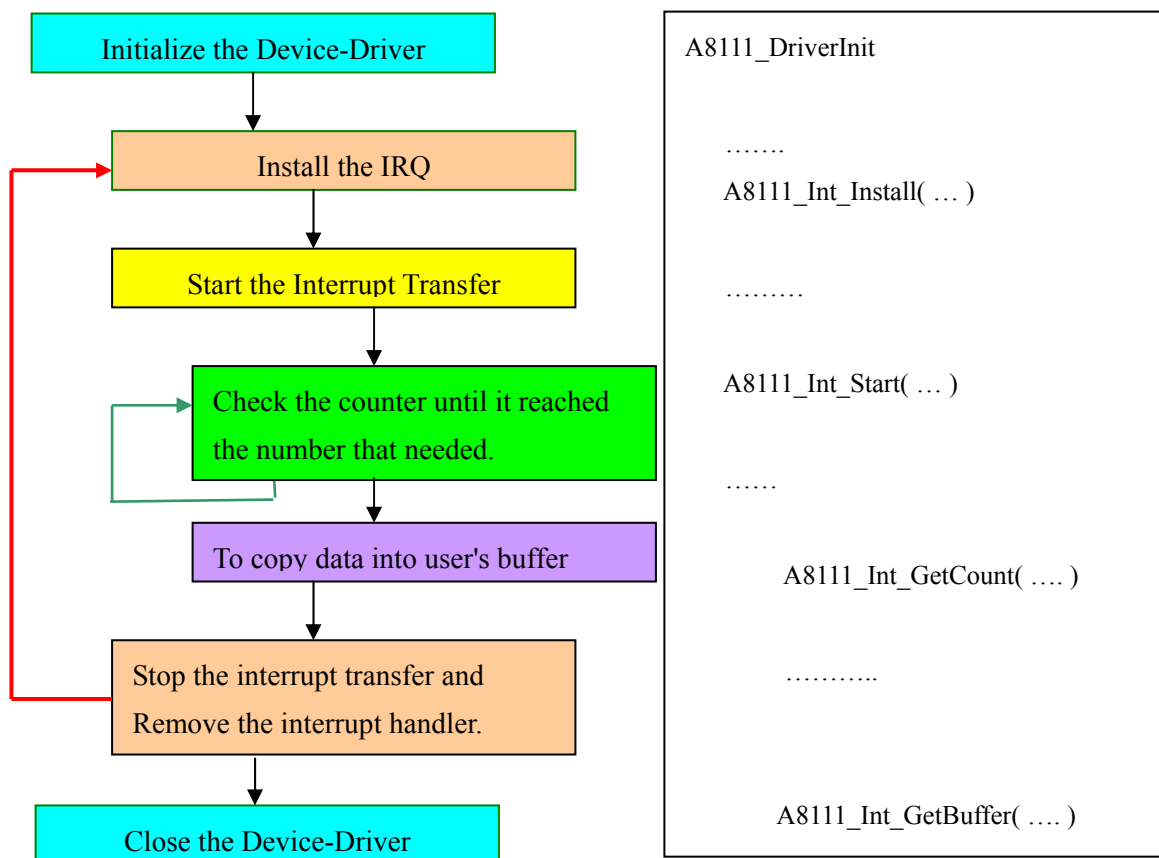
- **Return:**

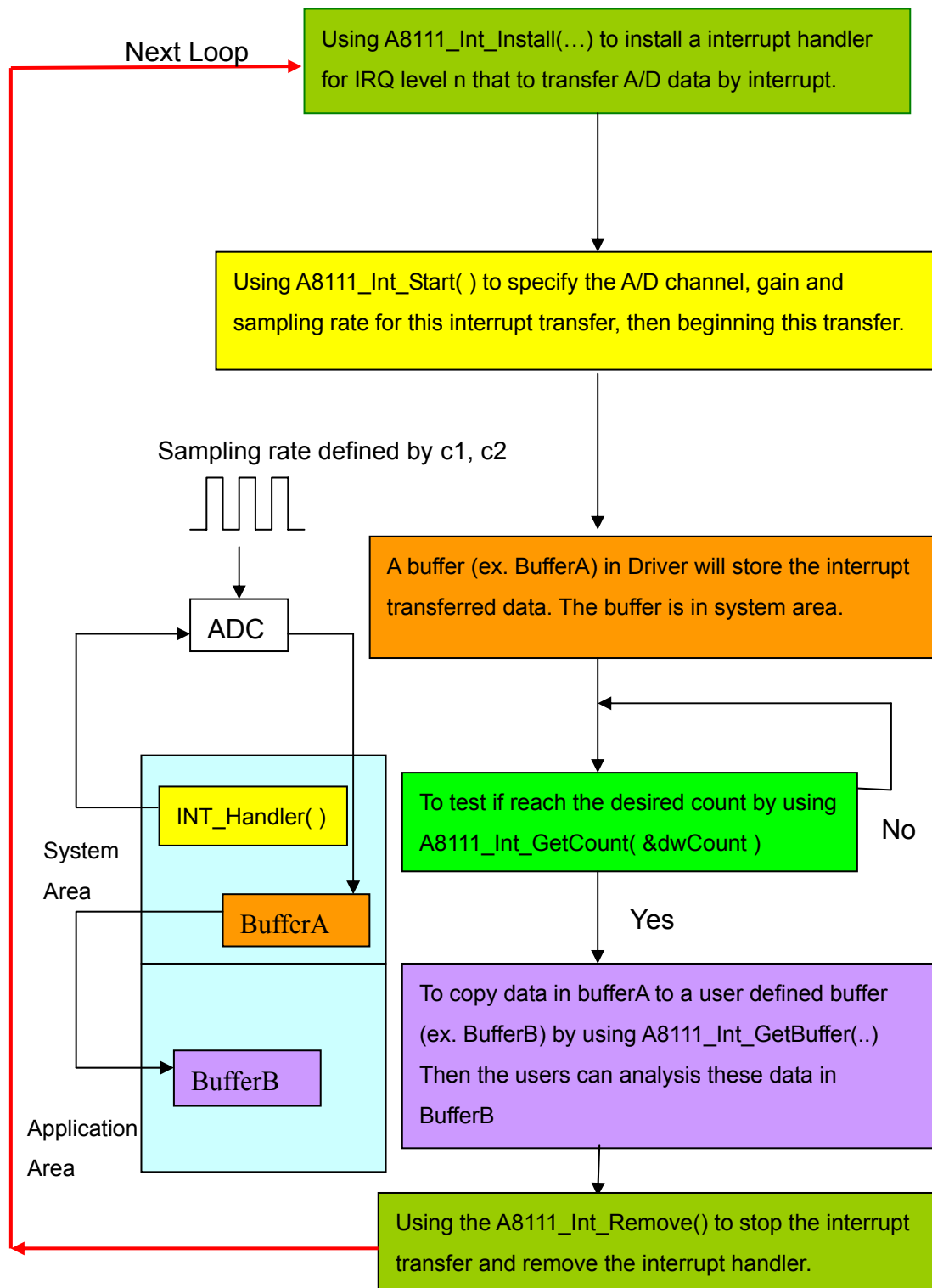
A8111_NoError : successful

A8111_GetBufferError : failed

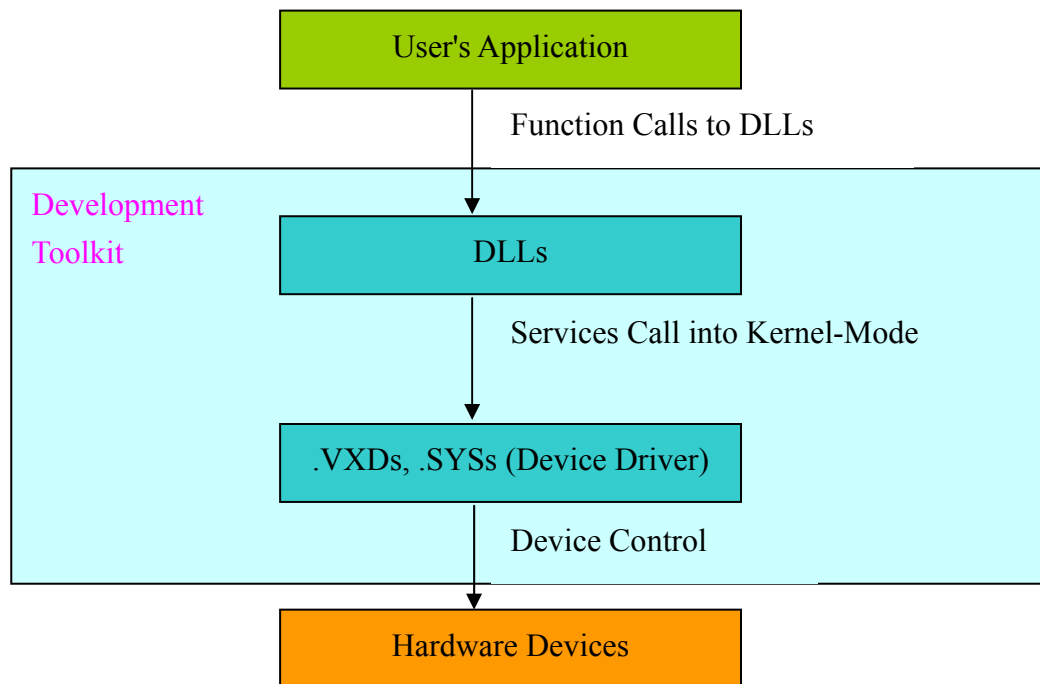
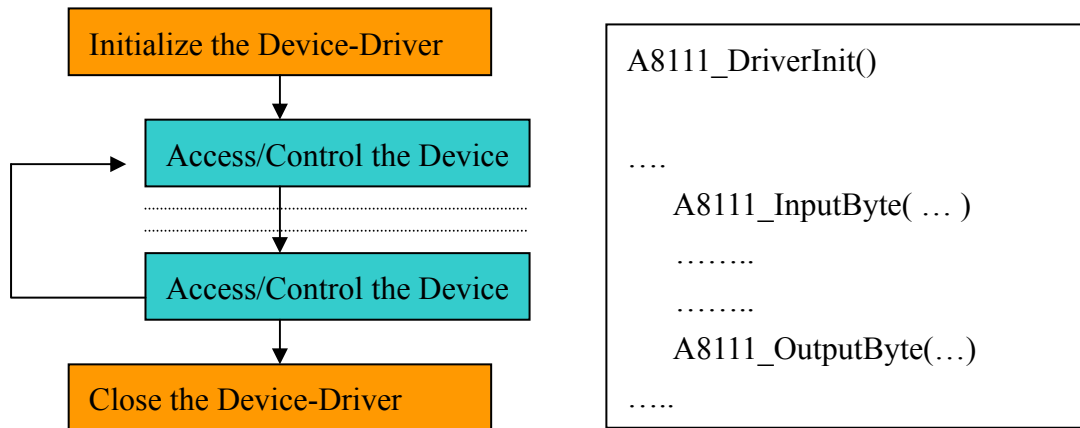
4.5.7 IRQ-mode conceptual diagram

The 4.5.1 to 4.5.6 are the functions to perform the A/D conversion with interrupt transfer. The flow chart to program these functions is given as follows:





5. Program Architecture



6. Demo List

6.1 Visual C++ Demo

There are some demo programs provided as follows:

```
|--\Demo
  |--\VC5
    |--\01
      |--Demo1.c → DI/DO,A/D,D/A,Interrupt Demo
      |--A8111.h → include header file for VC++
      |--A8111.lib → linkage library file for VC++
```

Refer to the software disk for details.

6.2 Visual Basic Demo

There are some demo programs provided as follows:

```
|--\Demo
  |--\VB5
    |--\01 → DLL function and configuration test
    |--\02 → DI/DO function test
    |--\03 → A/D, D/A function Demo by Ads_Hex
    |--\04 → A/D, D/A function Demo by Ads_Float
    |--\05 → A/D, D/A function Demo by Interrupt Mode
    |--A8111.BAS → Include Module file for VB
```

Refer to the software disk for details.

6.3 Delphi Demo

There are some demo programs provided as follows:

```
|--\Demo
  |--\Delphi3
    |--\01      → DLL function and configuration test
    |--\02      → DI/DO function test
    |--\03      → A/D, D/A function Demo by Ads_Hex
    |--\04      → A/D, D/A function Demo by Ads_Float
    |--\05      → A/D, D/A function Demo by Interrupt Mode
    |--\A8111.Pas → include Unit file for Delphi
```

Refer to the software disk for details.

6.4 Borland C++ Builder Demo

There are some demo programs provided as follows:

```
|--\Demo
  |--\BCB3
    |--\01      → DLL function and configuration test
    |--\02      → DI/DO function test
    |--\03      → A/D, D/A function Demo by Ads_Hex
    |--\04      → A/D, D/A function Demo by Ads_Float
    |--\05      → A/D, D/A function Demo by Interrupt Mode
    |--\A8111.H  → include header file for Borland C++ Builder
    |--\A8111.lib → linkage library file for Borland C++ Builder
```

Refer to the software disk for details.

7. Report Problems

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

When reporting problems, please include the following information:

- 1) Is the problem reproducible? If so, how?
- 2) What kind and version of Operation Systems that you running? For example, Windows 3.1, Windows for Workgroups, Windows NT 4.0, etc.
- 3) What kinds of our products that you using? Please see the product's manual.
- 4) If a dialog box with an error message was displayed, please include the full text of the dialog box, including the text in the title bar.
- 5) If the problem involves other programs or hardware devices, what devices or version of the failing programs that you using?
- 6) Other comments relative to this problem or any Suggestions will be welcomed.

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

E-mail: das@omega.com

Web-Site: <http://www.omega.com>

WARRANTY/DISCLAIMER

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

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

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

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

RETURN REQUESTS/INQUIRIES

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

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

FOR **WARRANTY** RETURNS, please have the following information available BEFORE contacting OMEGA:

1. Purchase Order number under which the product was PURCHASED,
2. Model and serial number of the product under warranty, and
3. Repair instructions and/or specific problems relative to the product.

FOR **NON-WARRANTY** REPAIRS, consult OMEGA for current repair charges. Have the following information available BEFORE contacting OMEGA:

1. Purchase Order number to cover the COST of the repair,
2. Model and serial number of the product, and
3. Repair instructions and/or specific problems relative to the product.

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

OMEGA is a registered trademark of OMEGA ENGINEERING, INC.

© 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