

ISO-813

Software Manual [For DOS/ Windows95/98/NT]

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 damage 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, ICP DAS assumes no responsibility for its use, or for any infringements of patents or other rights of third parties resulting from its use.

Copyright

Copyright 1999 by ICP DAS. All rights are reserved.

Trademark

The names used for identification only maybe registered trademarks of their respective companies.

License

The user can use, modify and backup this software **on a single machine**. The user may not reproduce, transfer or distribute this software, or any copy, in whole or in part.

Table of Contents

1. INTRODUCTION.....	4
1.1 General Description.....	4
1.2 Installation	5
1.3 A/D Gain Code table	9
1.4 References.....	10
2. CALL LIBS (FOR DOS).....	11
2.1 Using MSC.....	11
2.2 Using TC	12
2.3 Using BC	13
2.4 Demo source.....	16
2.4.1. ISO813.H	16
2.4.2. ISO813U.C.....	18
2.4.3. Demo1.C	21
3. CALL DLLS & DEMOS.....	22
3.1 Using Visual C++.....	22
3.2 Using MFC	23
3.2.1. The VC++ Demo Result:	24
3.2.2. ISO813.H	25
3.3 Using VB	27
3.3.1. The VB Demo Result:.....	27
3.3.2. ISO813.BAS.....	28
3.4 Using Delphi.....	30
3.4.1. Delphi Demo Result :	30
3.4.2. ISO813.PAS.....	31
3.5 Using Borland C++ Builder.....	34
3.5.1. Borland C++ Builder Demo Result	34
3.5.2. ISO813.H	35
4. FUNCTION DESCRIPTION	37
4.1. TEST Function	39
4.1.1. ISO813_SHORT_SUB_2	39
4.1.2. ISO813_FLOAT_SUB_2	39
4.1.3. ISO813_Get_DLL_Version.....	40
4.1.4. ISO813_GetDriverVersion.....	40
4.2. DI/DO Function	41
4.2.1. ISO813_OutputByte	41
4.2.2. ISO813_OutputWord.....	41
4.2.3. ISO813_InputByte	42
4.2.4. ISO813_InputWord.....	42
4.3. A/D Functions.....	43
4.3.1. ISO813_AD_Hex.....	43
4.3.2. ISO813_ADs_Hex.....	44

4.3.3. ISO813_AD_Float	45
4.3.4. ISO813_ADs_Float	46
4.3.5. ISO813_AD2F	47
4.3.6. ISO813_AD_SetReadyTicks	47
4.4. Driver functions	48
4.4.1. ISO813_DriverInit	48
4.4.2. ISO813_DriverClose	48
4.4.3. ISO813_Check_Address	49
4.5. Timer functions	50
4.5.1. ISO813_TimerRead	50
4.5.2. ISO813_TimerDelay	50
5. PROGRAM ARCHITECTURE	51
6. DEMO PROGRAM	52
7. PROBLEMS REPORT	53

1. Introduction

1.1. General Description

The ISO-813 is a bus type isolated 12-bit 32-channel analog input board for the PC/AT compatible computer. The isolation range is increased to 3000V and extends the application field to real industry application. It is backward compatible to ACL-813 add X16 programmable gain control range.

The ISO-813 LIB/DLL/Vxd is a collection of data acquisition subroutines for ISO-813 for DOS/ Windows95/98/NT Applications. These subroutines are written with C language and perform a variety of data acquisition operations.

The subroutines in ISO-813 are easy understanding as its name standing for. It provides powerful, easy-to-use subroutine for developing your data acquisition application. Your program can call these functions by TC, BC, MSC, VC++, VB, Delphi and Borland C++ Builder easily. To speed-up your developing process, some demonstration source program are provided.

The ISO-813 consists of these libraries and device driver:

For DOS

- Driver\ISO813S.lib → The small mode library.
- Driver\ISO813M.lib → The medium mode library.
- Driver\ISO813C.lib → The compact mode library.
- Driver\ISO813L.lib → The large mode library.
- Driver\ISO813H.lib → The huge mode library.

(These libraries are tested with MSC6.0 / BC3.1 / TC2.0)

For Windows 95/98

- Driver\ISO813.dll → Libraries for ISO-813 card
- Driver\Napdio.Vxd → Device driver for Windows 95/98

For Windows NT

- Driver\ISO813.dll → Libraries for ISO-813 card
- Driver\Napwnt.sys → Device driver for Windows NT

1.2. Installation

It is recommended to install the ISO-813 software to your hard disk to get the best performance. Before beginning, to make a backup copy of the ISO-813 software. Store the original diskette in a safe place.

A). The CD Disc Contents:

```
|--\DOS
|   |--\README.TXT
|   |--\ISO813.exe    ← The Self-Decompressed program for DOS
|
|--\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 'ISO-813 Setup CD' into CD-ROM.

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

Step 3). Enter the path as:

C:\DOS\ISO813\ISO813.exe (for DOS; Copy the program
to the folder which will contain the decompressed files)

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). Following those instructions in installation process to complete it.

After installed,

Windows NT:

The ISO813.DLL will be copied into C:\WINNT\SYSTEM32
and the Napwnt.sys into C:\WINNT\SYSTEM32\DRIVERS.

Windows 95:

The ISO813.DLL, Napdio.Vxd will be copied into C:\Windows\SYSTEM.

C). After installed, the sub-directory tree as follows:

- for DOS :

```
|--\[BaseDirectory]
|   |--\Demo
|       |--\BC           → Demo program for Borland C++ 3.1
|       |--\MSC          → Demo program for Microsoft C 6.0
|       |--\TC           → Demo program for Turbo C 2.0
|
|   |--\Driver
|       |--\ISO813S.LIB  → The small mode library.
|       |--\ISO813M.LIB  → The medium mode library.
|       |--\ISO813C.LIB  → The compact mode library.
|       |--\ISO813L.LIB  → The large mode library.
|       |--\ISO813H.LIB  → The huge mode library.
|
|   |--\Manual
|       |--\I813Hard.pdf → ISO-813 Hardware Manual
|       |--\I813Soft.pdf → ISO-813 Software Manual
|
|   |--\813Dig
|       |--\Setup.exe    → ISO-813 Diagnostic Setup program
|                           for Windows 31/95/98/NT
|
|   |--\Readme.txt
```

- for WINDOWS 95 :

```
--\[BaseDirectory]
|--\Demo
|   |--\BCB3           → Demo for Borland C++ Builder 3.0
|   |--\Delphi3        → Demo program for Delphi 3.0
|   |--\VB5            → Demo program for Visual Basic 5.0
|   |--\VC5            → Demo program for Visual C++
|
|--\Driver
|   |--\ISO813.DLL      → DLL Library file
|   |--\Napdio.Vxd      → Device driver for Windows 95/98
|
|--\Manual
|   |--\I813Hard.pdf    → ISO-813 Hardware Manual
|   |--\I813Soft.pdf    → ISO-813 Software Manual
|
|--\Diag
|   |--\Diag.exe        → ISO-813 Diagnostic program
|
|--\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\VC\nmake /f demo1.mak

- for WINDOWS NT :

```
|--\[BaseDirectory]
|   |--\Demo
|       |--\BCB           → Demo for Borland C++ Builder 3.0
|       |--\Delphi        → Demo program for Delphi 3.0
|       |--\VB            → Demo program for Visual Basic 5.0
|       |--\VC            → Demo program for Visual C++
|
|   |--\Driver
|       |--\ISO813.DLL     → DLL Library file
|       |--\Napwnt.sys    → Device driver for Windows NT
|
|   |--\Manual
|       |--\I813Hard.pdf   → ISO-813 Hardware Manual
|       |--\I813Soft.pdf  → ISO-813 Software Manual
|
|   |--\Diag
|       |--\Diag.exe       → ISO-813 Diagnostic program
|
|   |--\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\VC\nmake /f demo1.mak

1.3. A/D Gain Code table

ISO-813 Bipolar mode GAIN CONTROL CODE TABLE

- JP2 : Bipolar

	JP1 : 10V	JP1 : 20V				(Gain Code)
GAIN	Input Range	Input Range	GAIN2	GAIN1	GAIN0	Hex
1	$\pm 5V$	$\pm 10V$	0	0	0	0x0
2	$\pm 2.5V$	$\pm 5V$	0	0	1	0x1
4	$\pm 1.25V$	$\pm 2.5V$	0	1	0	0x2
8	$\pm 0.625V$	$\pm 1.25V$	0	1	1	0x3
16	$\pm 0.3125V$	$\pm 0.625V$	1	0	0	0x4

ISO-813 Unipolar mode GAIN CONTROL CODE TABLE

- JP2 : Unipolar

	JP1 : 10V	JP1 : 20V				(Gain Code)
GAIN	Input Range	Input Range	GAIN2	GAIN1	GAIN0	Hex
1	0~10V	Not use	0	0	0	0x0
2	0~5V	Not use	0	0	1	0x1
4	0~2.5V	Not use	0	1	0	0x2
8	0~1.25V	Not use	0	1	1	0x3
16	0~0.625V	Not use	1	0	0	0x4

1.4. 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.

2. Call LIBs (for DOS)

Driver\ISO813.H	→ Header file for including
Driver\ISO813u.C	→ Some functions for Library (floating-point computation)
Driver\ISO813S.Lib	→ The small mode library
Driver\ISO813M.Lib	→ The medium mode library
Driver\ISO813C.Lib	→ The compact mode library
Driver\ISO813L.Lib	→ The large mode library
Driver\ISO813H.Lib	→ The huge mode library

These files support MSC 6.0 / Borland C++ 3.1 / Turbo C 2.0.

2.1. Using MSC

Support MSC 6.x compiler

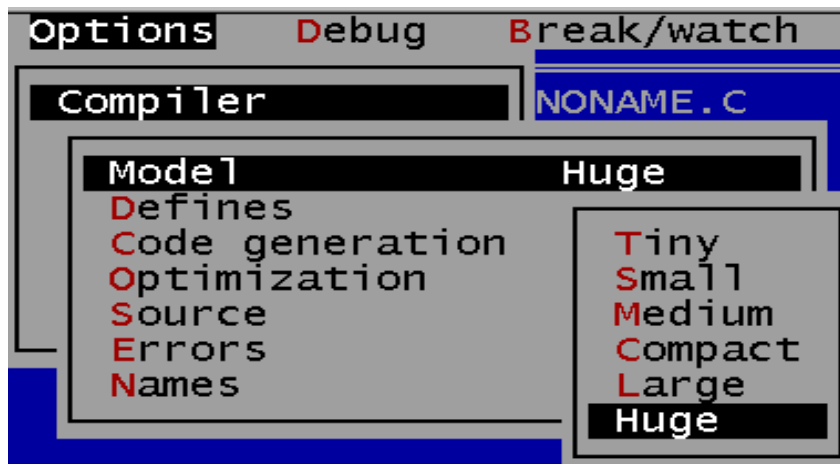
- The including file is **ISO813.H**
- **SMALL** mode compiler & link command :
CL /AS program.c iso813u.c ISO813S.LIB
- **COMPACT** mode compiler & link command :
CL /AC program.c iso813u.c ISO813C.LIB
- **MEDIUM** mode compiler & link command :
CL /AM program.c iso813u.c ISO813M.LIB
- **LARGE** mode compiler & link command :
CL /AL program.c iso813u.c ISO813L.LIB
- **HUGE** mode compiler & link command :
CL /AH program.c iso813u.c ISO813H.LIB

A:\ISO813\demo\msc*.bat give some examples for compiler&link batch file.

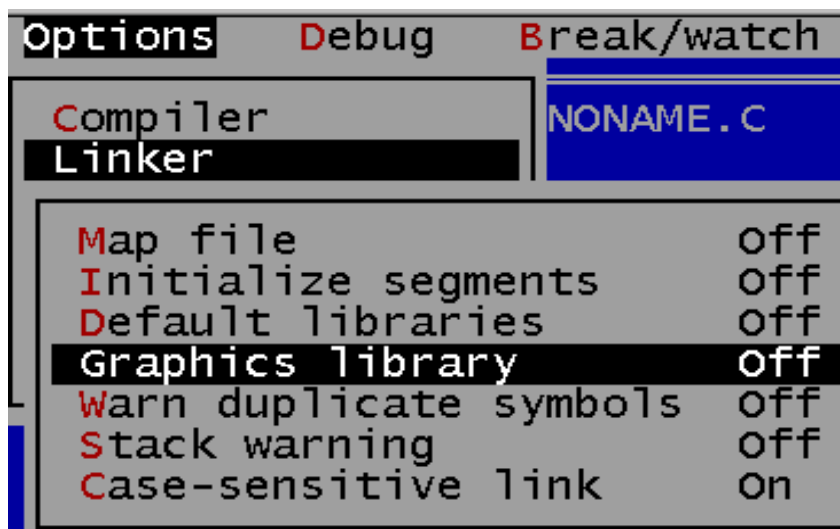
2.2. Using TC

Support TC 2.x compiler

- The including file is **ISO813.H**
- Use text editor to create a project file include :
program.c iso813u.c ISO813?.lib
- Use TC integrated environment to **select the correct compiler model**



Select the compiler model that is you used



Include or Exclude the Graphics library

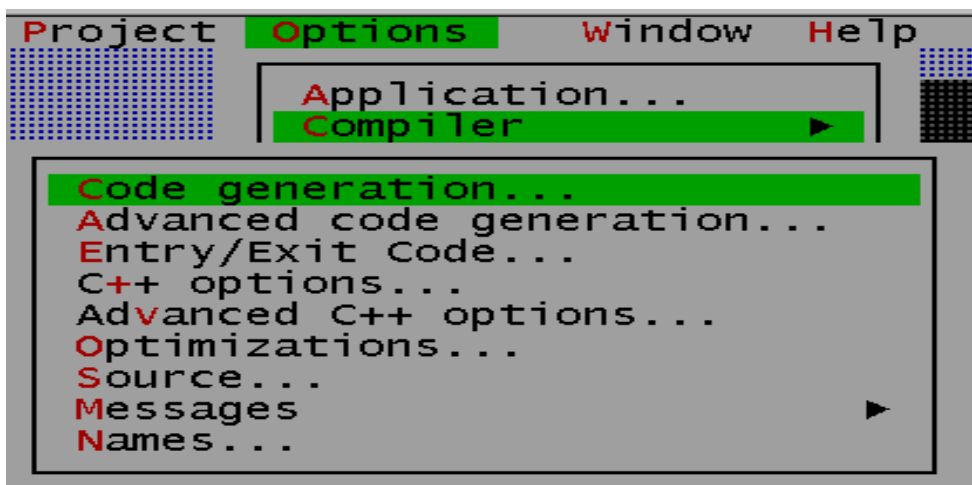
A:\ISO813\demo\TC*.prj give some examples for compiler&link project file

Note : Demo2 need to include the graphics library.

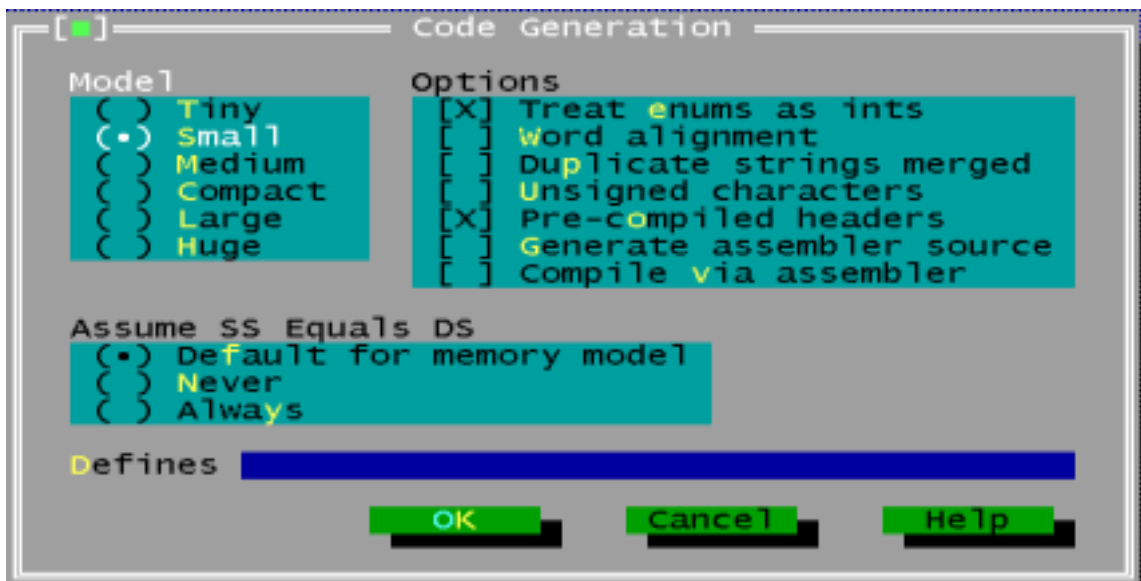
2.3. Using BC

Support BC 3.x compiler

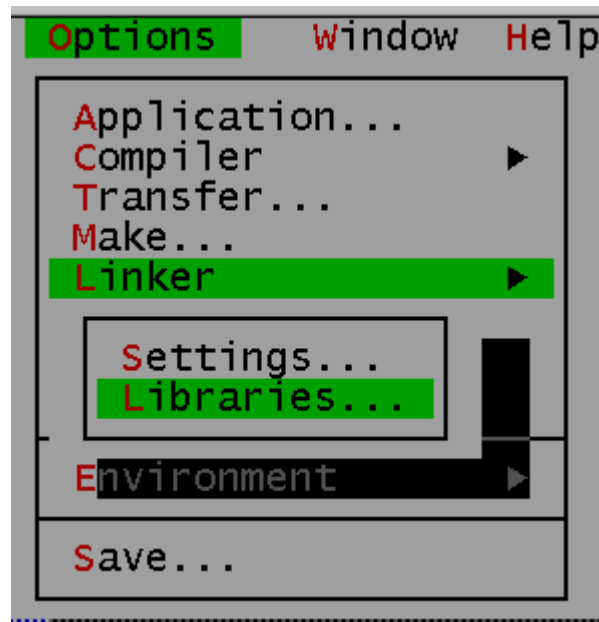
- The including file is **ISO813.H**
- Use BC integrated environment to create a project file include :
program.c iso813u.c ISO813?.lib
- Use BC integrated environment to **select the correct compiler model**



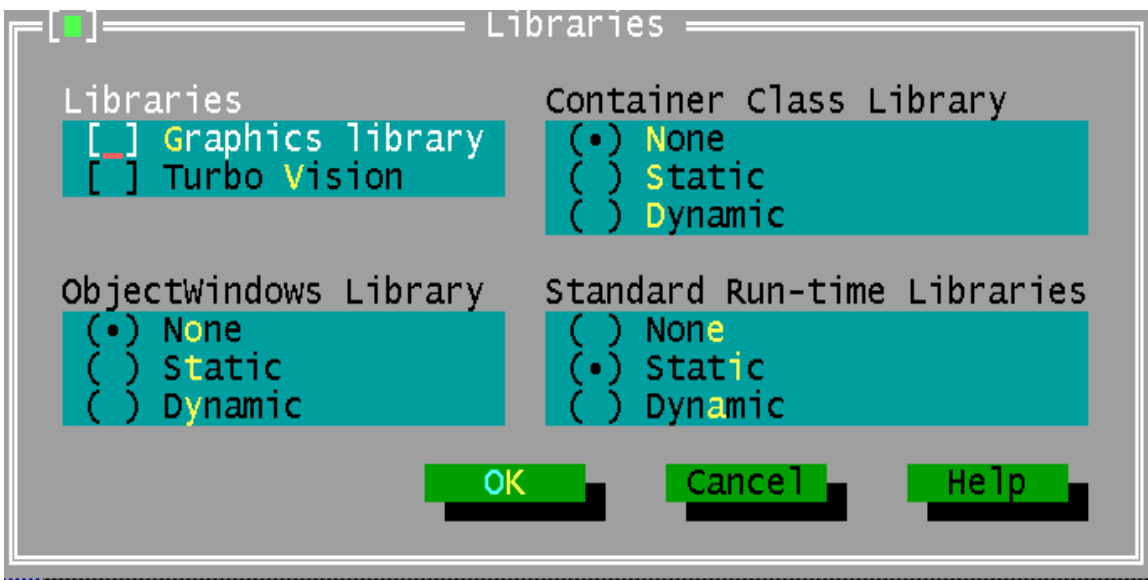
Use [Code generation...] to popup the Code Generation window



Use "Model" Radio-Button to select the compiler model



Use [Libraries...] to popup the Libraries window

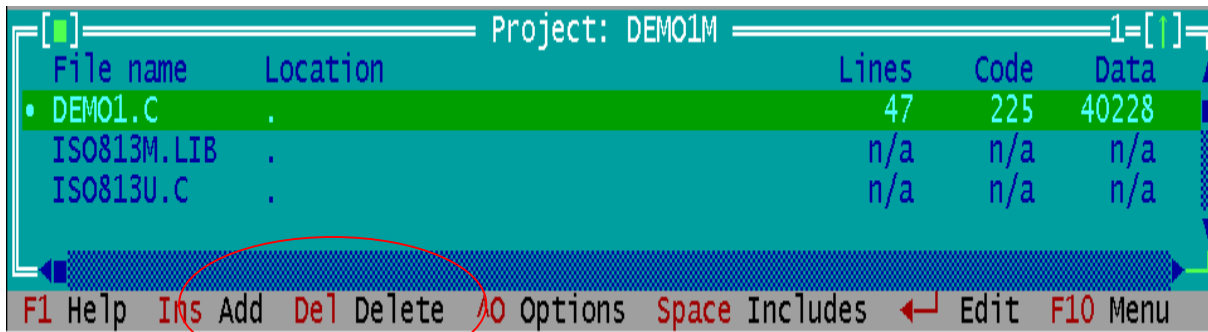


Use "Libraries" checkbox to include/exclude the graphics library

Note : Demo2 need include the Graphics library



Use [Project] to popup the Project window



In the Project window, you can use [Ins] and [Del] key to insert a file to the project or delete from the project.

A:\ISO813\demo\BC*.prj give some examples for compiler&link project file

2.4. Demo source

2.4.1. ISO813.H

```
/****** define the gain mode *****/
#define ISO813_BI_1 0
#define ISO813_BI_2 1
#define ISO813_BI_4 2
#define ISO813_BI_8 3
#define ISO813_BI_16 4
#define ISO813_UNI_1 0
#define ISO813_UNI_2 1
#define ISO813_UNI_4 2
#define ISO813_UNI_8 3
#define ISO813_UNI_16 4
/****** define the error number *****/
#define ISO813_NoError 0
#define ISO813_CheckBoardError 1
#define ISO813_DriverOpenError 2
#define ISO813_DriverNoOpen 3
#define ISO813_AdError 4
#define ISO813_AdError2 -100.0
#define ISO813_OtherError 5
#define ISO813_GetDriverVersionError 6
#define ISO813_TimeOutError 0xffff

#define WORD unsigned int
#define DWORD unsigned long
#define UCHAR unsigned char

/****** Function of Test *****/
short ISO813_SHORT_SUB_2(short nA, short nB);
WORD ISO813_GetDriverVersion(WORD *wDriverVersion);

/****** Function of Driver *****/
unsigned int ISO813_DriverInit(void);
void ISO813_DriverClose(void);
WORD ISO813_Check_Address(WORD wBase);
```



```

/***** Function of AD *****/
WORD ISO813_AD_Hex(WORD wBase, WORD wChannel,
    WORD wGainCode );
WORD ISO813_ADs_Hex( WORD wBase,  WORD wChannel,
    WORD wGainCode, WORD *wBuffer, DWORD dwDataNo );
void ISO813_AD_SetReadyTicks(WORD wTicks);

/***** Declare 8253 Timer Interface *****/
WORD ISO813_TimerRead(WORD *wTicks);
void ISO813_TimerDelay(DWORD dwTicks);

/***** Function of DIO *****/
void ISO813_OutputByte(WORD wPortAddr, UCHAR bOutputVal);
void ISO813_OutputWord(WORD wPortAddr, WORD wOutputVal);
WORD ISO813_InputByte(WORD wPortAddr);
WORD ISO813_InputWord(WORD wPortAddr);

```

2.4.2. ISO813U.C

```
#include<stdlib.h>
#include "iso813.h"

float ISO813_AD_Float( WORD wBase, WORD wChannel, WORD wGainCode,
WORD wBipolar,WORD wJump10v );
WORD ISO813_ADs_Float(WORD wBase, WORD wChannel,
WORD wGainCode, WORD wBipolar, WORD wJump10v,float *fBuffer,
DWORD dwDataNo, WORD *wTmpBuf );
float ISO813_AD2F(WORD wHex, WORD wGainCode, WORD wBipolar,
WORD wJump10v);

/*-----*/
/* Bipolar, 1:Bipolar 0:Unipolar */
/* Jump10v, 1:10v 0:20v */
/*-----*/
float ISO813_AD_Float( WORD wBase, WORD wChannel, WORD wGainCode,
WORD wBipolar,WORD wJump10v )
{
    WORD wVal ;

    wVal = ISO813_AD_Hex(wBase, wChannel , wGainCode);
    if ( wVal != ISO813_TimeOutError )
        return( ISO813_AD2F(wVal, wBipolar, wGainCode, wJump10v) );
    /* return fVal or AdError2 */
    else
        return ( ISO813_AdError2 );
}
```

```
/*-----*/
/* Bipolar, 1:Bipolar 0:Unipolar */
/* Jump10v, 1:10v 0:20v */
/* 'fBuffer' point to a float array with 'dwDataNo' size. */
/*-----*/
WORD ISO813_ADs_Float(WORD wBase, WORD wChannel,
                     WORD wGainCode, WORD wBipolar, WORD wJump10v, float *fBuffer,
                     DWORD dwDataNo, WORD *wTmpBuf )
{
    DWORD i;
    float ZeroBase, FullRange, VoltageRange;

    if ( wGainCode > 4 ) return ISO813_AdError;
    if ( wGainCode == 0 && ( wBipolar==0 ) && ( wJump10v==0 ) )
        return ISO813_AdError;

    if (wBipolar==1) {
        ZeroBase = 2048.0;
        FullRange = 2048.0;
        if (wJump10v==1)
            VoltageRange = 5.0;
        else
            VoltageRange = 10.0;
    }
    else {
        ZeroBase = 0.0;
        FullRange = 4095.0;

        if (wJump10v==1)
            VoltageRange = 10.0;
        else
            VoltageRange = 20.0;
    }

    if ( ISO813_ADs_Hex(wBase, wChannel, wGainCode, wTmpBuf , dwDataNo)
        == ISO813_NoError )
    {
        for (i=0;i<dwDataNo;i++)
            fBuffer[i] = (((wTmpBuf[i]-ZeroBase)/FullRange)*VoltageRange)
                        / (float) (1<<wGainCode) ;
    }
    else
        return ISO813_AdError;
    return ISO813_NoError;
}
```

```
/*-----*/
/* Return voltage value or -100.0 if any error occurs */
/* or parameter is out of range. */
/* Bipolar, 1:Bipolar 0:Unipolar */
/* Jump10v, 1:10v 0:20v */
/*-----*/
float ISO813_AD2F(WORD wHex, WORD wGainCode, WORD wBipolar,
WORD wJump10v)
{
    float ZeroBase, FullRange, VoltageRange;

    if ( wGainCode > 4 ) return(ISO813_AdError2);
    if ( wGainCode == 0 && ( wBipolar==0 ) && (wJump10v==0) )
return(ISO813_AdError2);

    if (wBipolar==1) {
        ZeroBase = 2048.0;
        FullRange = 2048.0;
        if (wJump10v==1)
            VoltageRange = 5.0;
        else
            VoltageRange = 10.0;
    }
    else {
        ZeroBase = 0.0;
        FullRange = 4095.0;

        if (wJump10v==1)
            VoltageRange = 10.0;
        else
            VoltageRange = 20.0;
    }

    return( (((wHex-ZeroBase)/FullRange)*VoltageRange)/(float)(1<<wGainCode)
);
}
```

2.4.3. Demo1.C

```
#include<stdio.h>
#include"iso813.h"

int Base = 0x220;
WORD Buf[20000];
/* DWORD ArrayTimes , ErrorTimes; */

void main(void)
{
    int rtErr;
    DWORD t0 , t1 , i, dt;
    WORD wGainCode, wBipolar, wJump10v, wChannel ;

    /* Initialize Timer and ISO813 Board and then check it. */
    rtErr = ISO813_DriverInit();
    if (rtErr != ISO813_NoError ) {
        printf("\nISO813 Driver open error!");
        return;
    }

    rtErr = ISO813_Check_Address( Base );
    if (rtErr != ISO813_NoError ){
        printf("\nISO813 Board Address 0x220 Error");
        return ;
    }

    printf("\nCard 0x220 [polling test]");
    printf("\nUsing [ISO813_ADs_Hex(BaseAddress, Channel, GainCode, wBuf,
DataNo) ]");
    wChannel = 0; /* Set to Channel 0 */
    wGainCode = 0; /* Gain 0 */
    rtErr = ISO813_ADs_Hex( Base, wChannel, wGainCode, Buf, 1000 );
    if ( rtErr == ISO813_NoError )
        for( i=0;i<1000;i+=1)
            printf("\n[%3lu]:[%4x]:[%6d]",i,Buf[i],Buf[i]);
    else
        printf("\nISO813_ADs_Hex Error!!\nErrorCode : %d",rtErr);

    ISO813_DriverClose();
}
```

3. Call DLLs & Demos

Please refer to the user manual "**CallDLL.pdf**".

```
|--\Driver
|   |--\BC
|       |--\ISO813.h
|       |--\ISO813.Lib
|   |--\Delphi
|       |--\ISO813.Pas
|   |--\VB
|       |--\ISO813.Bas
|   |--\VC
|       |--\ISO813.h
|       |--\ISO813.Lib
```

3.1. Using Visual C++

The entire demo programs given in the companion floppy disk are designed with C language. It is testing under Windows 95/98/NT and Visual C++ 5.0 compiler. The key points are given as 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 "ISO813.H"
4. Copy the ISO813.LIB , import library, to the directory where same as source program.
5. Copy the ISO813.DLL, to the directory where same as source program
6. Edit the source program (refer to \ISO813\Demo\VC\DEMO1.C)
7. Edit the NMAKE file (refer to \ISO813\Demo\VC\DEMO1.MAK)
8. NMAKE /f DEMO1.MAK
9. Execute DEMO1.EXE

NOTE : The **ISO813.Lib** is used in linking time and the **ISO813.DLL** is used in run time.

3.2. Using MFC

The usage of ISO-813 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 "ISO813.H"
3. Copy the ISO813.LIB, import library, to the directory same as source program
4. Copy the ISO813.DLL, to the directory same as source program
5. Select **Build/Settings/Link** and key "ISO813.lib" in the **object/library modules** field

NOTE : The **ISO813.lib** is used in linking time and the **ISO813.DLL** is used in run time.

3.2.1. The VC++ Demo Result:

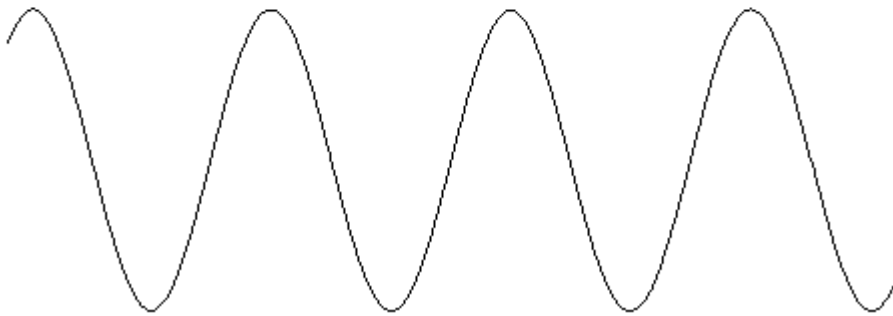
 Demo ISO813 = [BASE:220]

NOW --> Base=220,

Uxd Version=101

Now Setting Is --> Base=220

1. DLL Version=110
 2. Find a ISO813 in IOPORT_220
 3. ISO813_SHORT_SUB_2(1,2) = -1
 4. ISO813_FLOAT_SUB_2(1.0,2.0) = -1.000000
 5. ISO813_AD_Float TEST Error
 6. ISO813_ADs_Hex TEST :2.739375
 7. ISO813_ADs_Float TEST :4.268555
- f0d f0e f0a eff ef1



3.2.2. ISO813.H

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

/***** define the gain mode *****/
#define ISO813_BI_1          0x00
#define ISO813_BI_2          0x01
#define ISO813_BI_4          0x02
#define ISO813_BI_8          0x03
#define ISO813_BI_16         0x04
#define ISO813_UNI_1         0x00
#define ISO813_UNI_2         0x01
#define ISO813_UNI_4         0x02
#define ISO813_UNI_8         0x03
#define ISO813_UNI_16        0x04

/***** define the error number *****/
#define ISO813_NoError        0
#define ISO813_CheckBoardError 1
#define ISO813_DriverOpenError 2
#define ISO813_DriverNoOpen   3
#define ISO813_AdError        4
#define ISO813_AdError2       -100.0
#define ISO813_OtherError     5
#define ISO813_GetDriverVersionError 6
#define ISO813_TimeOutError   0xffff

/***** Function of Test *****/
EXPORTS short CALLBACK ISO813_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK ISO813_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK ISO813_Get_DLL_Version(void);
EXPORTS WORD CALLBACK ISO813_GetDriverVersion(
    WORD *wDriverVersion);

/***** Function of Driver *****/
EXPORTS WORD CALLBACK ISO813_DriverInit(void);
EXPORTS void CALLBACK ISO813_DriverClose(void);
EXPORTS WORD CALLBACK ISO813_Check_Address(WORD wBase);
```

```

/***** Function of AD *****/
EXPORTS WORD CALLBACK ISO813_AD_Hex( WORD wBase,
    WORD wChannel, WORD wGainCode );
EXPORTS WORD CALLBACK ISO813_ADs_Hex( WORD wBase,
    WORD wChannel, WORD wGainCode, WORD *wBuffer,
    DWORD dwDataNo );
EXPORTS float CALLBACK ISO813_AD_Float( WORD wBase,
    WORD wChannel, WORD wGainCode, WORD wBipolar,
    WORD wJump10v );
EXPORTS WORD CALLBACK ISO813_ADs_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, WORD wBipolar,
    WORD wJump10v, float *fBuffer, DWORD dwDataNo );
EXPORTS float CALLBACK ISO813_AD2F(WORD whex, WORD wGainCode,
    WORD wBipolar, WORD wJump10v);
EXPORTS void CALLBACK ISO813_AD_SetReadyTicks(WORD wTicks);

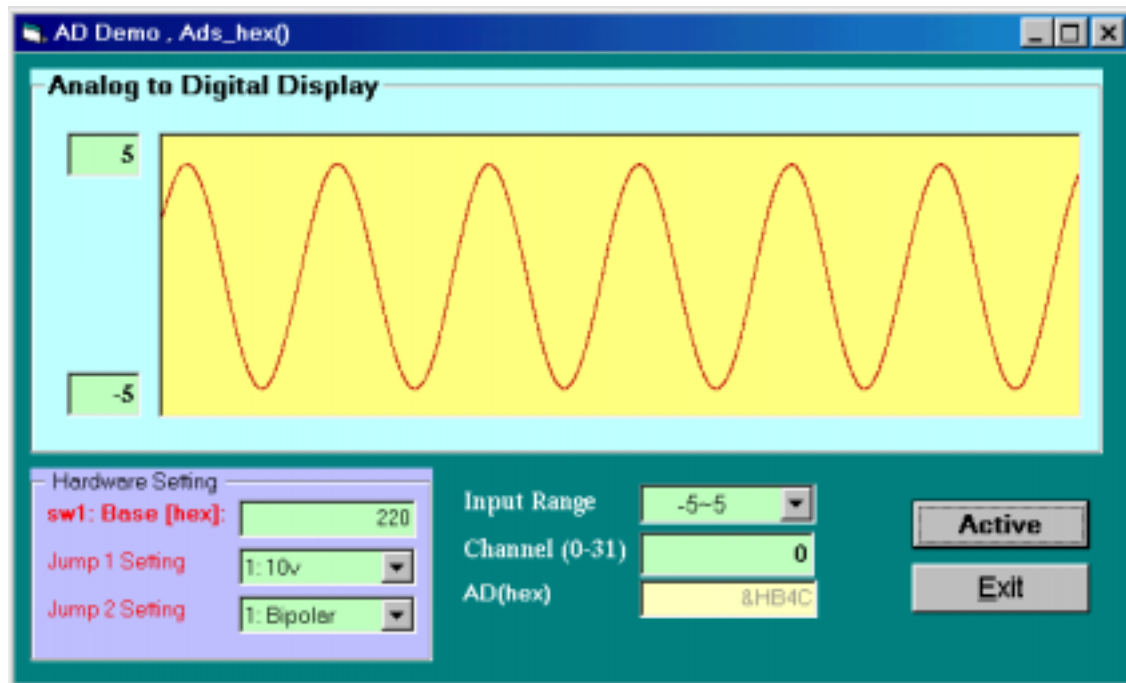
/***** Declare 8253 Timer Interface *****/
EXPORTS WORD CALLBACK ISO813_TimerRead(WORD *wTicks);
EXPORTS void CALLBACK ISO813_TimerDelay(DWORD dwTicks);

/***** Function of DIO *****/
EXPORTS void CALLBACK ISO813_OutputByte(WORD wPortAddr,
    UCHAR bOutputVal);
EXPORTS void CALLBACK ISO813_OutputWord(WORD wPortAddr,
    WORD wOutputVal);
EXPORTS WORD CALLBACK ISO813_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK ISO813_InputWord(WORD wPortAddr);

```

3.3. Using VB

3.3.1. The VB Demo Result:



3.3.2. ISO813.BAS

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

'***** define the gain mode *****/

Global Const ISO813_BI_1	= 0
Global Const ISO813_BI_2	= 1
Global Const ISO813_BI_4	= 2
Global Const ISO813_BI_8	= 3
Global Const ISO813_BI_16	= 4
Global Const ISO813_UNI_1	= 0
Global Const ISO813_UNI_2	= 1
Global Const ISO813_UNI_4	= 2
Global Const ISO813_UNI_8	= 3
Global Const ISO813_UNI_16	= 4

'***** define the error number *****/

Global Const ISO813_NoError	= 0
Global Const ISO813_CheckBoardError	= 1
Global Const ISO813_DriverOpenError	= 2
Global Const ISO813_DriverNoOpen	= 3
Global Const ISO813_AdError	= 4
Global Const ISO813_AdError2	= -100.0
Global Const ISO813_OtherError	= 5
Global Const ISO813_GetDriverVersionError	= 6
Global Const ISO813_TimeOutError	= &HFFFF

' Function of Test

```
Declare Function ISO813_SHORT_SUB_2 Lib "ISO813.DLL" (  
    ByVal nA As Integer, ByVal nB As Integer) As Integer  
Declare Function ISO813_FLOAT_SUB_2 Lib "ISO813.DLL" (  
    ByVal fA As Single, ByVal fB As Single) As Single  
Declare Function ISO813_Get_DLL_Version Lib "ISO813.DLL" () As Integer  
Declare Function ISO813_GetDriverVersion Lib "ISO813.DLL" (  
    wDriverVersion As Integer) As Integer
```

' Function of Driver

```
Declare Function ISO813_DriverInit Lib "ISO813.DLL" () As Integer  
Declare Sub ISO813_DriverClose Lib "ISO813.DLL" ()  
Declare Function ISO813_Check_Address Lib "ISO813.DLL" (  
    ByVal wBase As Integer) As Integer
```

' Function of AD

```
Declare Function ISO813_AD_Hex Lib "ISO813.DLL" (ByVal wBase As Integer,
    ByVal wChannel As Integer, ByVal wGainCode As Integer) As Integer
Declare Function ISO813_ADs_Hex Lib "ISO813.DLL" (ByVal wBase As Integer,
    ByVal wChannel As Integer, ByVal wGainCode As Integer,
    wBuf As Integer, ByVal dwDataNo As Long) As Integer
Declare Function ISO813_AD_Float Lib "ISO813.DLL" (ByVal wBase As Integer,
    ByVal wChannel As Integer, ByVal wGainCode As Integer,
    ByVal wBipolar As Integer, ByVal wJump10v As Integer) As Single
Declare Function ISO813_ADs_Float Lib "ISO813.DLL" (
    ByVal wBase As Integer, ByVal wChannel As Integer,
    ByVal wGainCode As Integer, ByVal wBipolar As Integer,
    ByVal wJump10v As Integer, fBuf As Single, ByVal dwDataNo As Long)
    As Integer
Declare Function ISO813_AD2F Lib "ISO813.DLL" (ByVal wHex As Integer,
    ByVal wGainCode As Integer, ByVal wBipolar As Integer,
    ByVal wJump10v As Integer) As Single
Declare Sub ISO813_AD_SetReadyTicks Lib "ISO813.DLL" (
    ByVal wTicks As Integer)
```

***** Declare 8253 Timer Interface *****

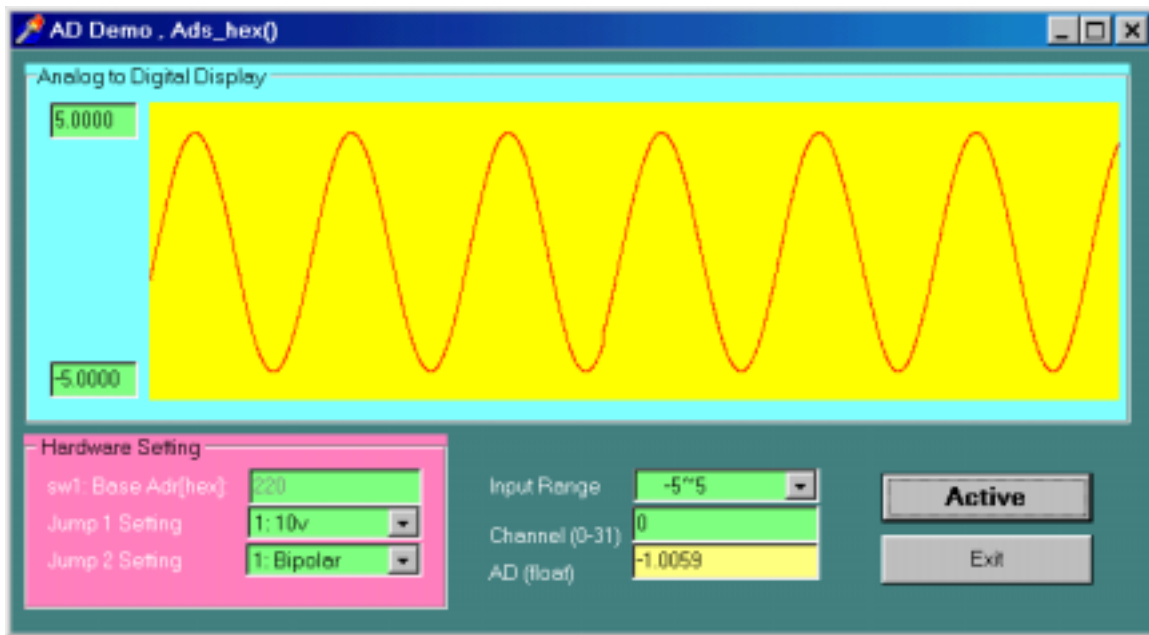
```
Declare Function ISO813_TimerRead Lib "ISO813.DLL" (wTicks As Integer)
    As Integer
Declare Sub ISO813_TimerDelay Lib "ISO813.DLL" (ByVal wTicks As Long)
```

' The DIO functions

```
Declare Sub ISO813_OutputByte Lib "dio.dll" (ByVal address As Integer,
    ByVal dataout As Byte)
Declare Sub ISO813_OutputWord Lib "dio.dll" (ByVal address As Integer,
    ByVal dataout As Integer)
Declare Function ISO813_InputByte Lib "dio.dll" (ByVal address As Integer)
    As Integer
Declare Function ISO813_InputWord Lib "dio.dll" (ByVal address As Integer)
    As Integer
```

3.4. Using Delphi

3.4.1. Delphi Demo Result :



3.4.2. ISO813.PAS

```
unit ISO813;

interface

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

const

    ISO813_BI_1           =0;
    ISO813_BI_2           =1;
    ISO813_BI_4           =2;
    ISO813_BI_8           =3;
    ISO813_BI_16          =4;

    ISO813_UNI_1           =0;
    ISO813_UNI_2           =1;
    ISO813_UNI_4           =2;
    ISO813_UNI_8           =3;
    ISO813_UNI_16          =4;

    //***** define the error number *****
    ISO813_NoError         =0;
    ISO813_CheckBoardError =1;
    ISO813_DriverOpenError =2;
    ISO813_DriverNoOpen    =3;
    ISO813_AdError          =4;
    ISO813_AdError          = -100.0 ;
    ISO813_OtherError       =5;
    ISO813_GetDriverVersionError =6;
    ISO813_TimeOutError     =$ffff;

    // Function of Test
function ISO813_SHORT_SUB_2(nA,nB:SmallInt):SmallInt; StdCall;
function ISO813_FLOAT_SUB_2(fA,fB:Single):Single; StdCall;
function ISO813_Get_DLL_Version:Word; StdCall;
function ISO813_GetDriverVersion(var wDriverVersion:Word):Word; StdCall;
```

// Function of Driver

```
function ISO813_DriverInit:Word; StdCall;  
procedure ISO813_DriverClose; StdCall;  
function ISO813_Check_Address(wBase:Word):Word; StdCall;
```

// Function of AD

```
function ISO813_AD_Hex(wBase,wChannel,wGainCode:Word):Single; StdCall;  
function ISO813_ADs_Hex(wBase, wChannel, wConfig:Word; wBuf:PWord;  
    dwDataNo:LongInt) : Word; StdCall;  
function ISO813_AD_Float(wBase, wChannel, wConfig, wBipolar,  
    wJmp10v:Word ):Single; StdCall;  
function ISO813_ADs_Float(wBase, wChannel, wConfig, wBipolar,  
    wJmp10v:Word; fBuf:PSingle; dwDataNo:LongInt) : Word; StdCall;  
function ISO813_AD2F(wHex, wGainCode, wBipolar, wJump10v :Word  
    ): Single ; StdCall;  
procedure ISO813_AD_SetReadyTicks(wTicks:WORD);StdCall;
```

//***** Declare 8253 Timer Interface *****/

```
function ISO813_TimerRead(var wTicks:Word) :Word; StdCall;  
procedure ISO813_TimerDelay(dwTicks:DWord);StdCall;
```

// The DIO functions

```
procedure ISO813_OutputByte(wPortAddr : word; bOutputVal :byte); StdCall;  
procedure ISO813_OutputWord(wPortAddr : word; wOutputVal :word);  
    StdCall;  
function ISO813_InputByte(wPortAddr :word) : word; StdCall;  
function ISO813_InputWord(wPortAddr :word) : word; StdCall;
```

implementation

uses math;

```
function ISO813_SHORT_SUB_2; external 'ISO813.DLL' name  
    'ISO813_SHORT_SUB_2';  
function ISO813_FLOAT_SUB_2; external 'ISO813.DLL' name  
    'ISO813_FLOAT_SUB_2';  
function ISO813_Get_DLL_Version; external 'ISO813.DLL' name  
    'ISO813_Get_DLL_Version';  
function ISO813_GetDriverVersion; external 'ISO813.DLL' name  
    'ISO813_GetDriverVersion';  
  
function ISO813_DriverInit; external 'ISO813.DLL' name  
    'ISO813_DriverInit';  
procedure ISO813_DriverClose; external 'ISO813.DLL' name  
    'ISO813_DriverClose';
```



```
function ISO813_Check_Address;      external 'ISO813.DLL' name
    'ISO813_Check_Address';

function ISO813_AD_Hex;             external 'ISO813.DLL' name
    'ISO813_AD_Hex';
function ISO813_ADs_Hex;            external 'ISO813.DLL' name
    'ISO813_ADs_Hex';
function ISO813_AD_Float;           external 'ISO813.DLL' name
    'ISO813_AD_Float';
function ISO813_ADs_Float;          external 'ISO813.DLL' name
    'ISO813_ADs_Float';
function ISO813_AD2F;               external 'ISO813.DLL' name
    'ISO813_AD2F';
procedure ISO813_AD_SetReadyTicks; external 'ISO813.DLL' name
    'ISO813_AD_SetReadyTicks';

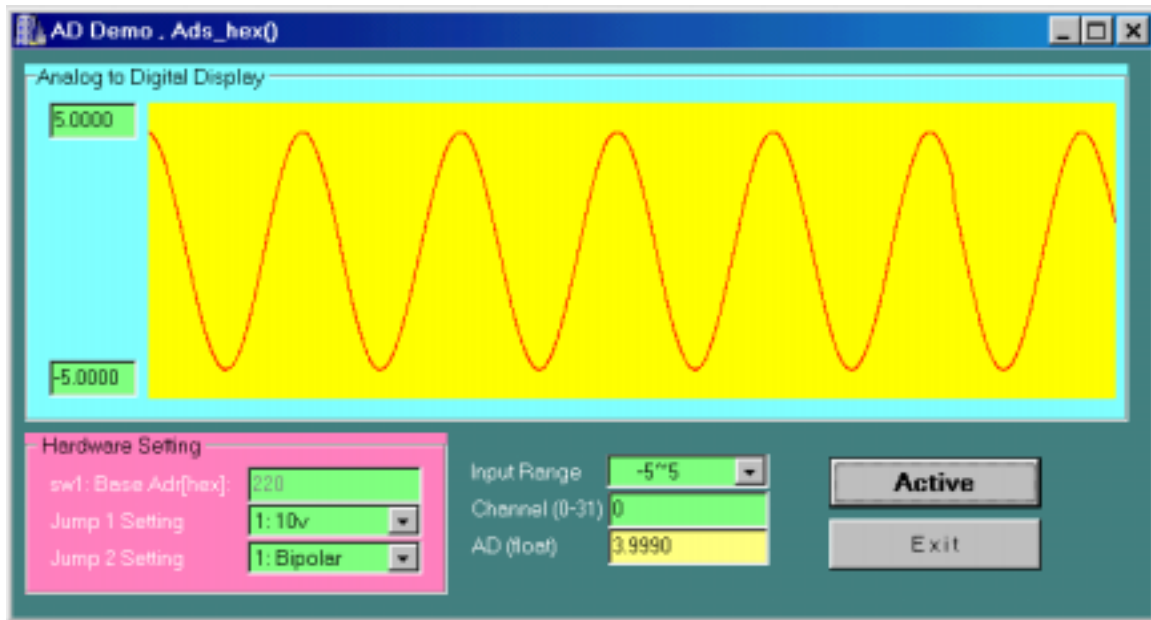
function ISO813_TimerRead;          external 'ISO813.DLL' name
    'ISO813_TimerRead';
procedure ISO813_TimerDelay;        external 'ISO813.DLL' name
    'ISO813_TimerDelay';

// The DIO functions
procedure ISO813_OutputByte;        external 'ISO813.DLL' name
    'ISO813_OutputByte';
procedure ISO813_OutputWord;        external 'ISO813.DLL' name
    'ISO813_OutputWord';
function ISO813_InputByte;          external 'ISO813.DLL' name
    'ISO813_InputByte';
function ISO813_InputWord;          external 'ISO813.DLL' name
    'ISO813_InputWord';

end.
```

3.5. Using Borland C++ Builder

3.5.1. Borland C++ Builder Demo Result



3.5.2. ISO813.H

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

/***** define the gain mode *****/
#define ISO813_BI_1          0x00
#define ISO813_BI_2          0x01
#define ISO813_BI_4          0x02
#define ISO813_BI_8          0x03
#define ISO813_BI_16         0x04
#define ISO813_UNI_1         0x00
#define ISO813_UNI_2         0x01
#define ISO813_UNI_4         0x02
#define ISO813_UNI_8         0x03
#define ISO813_UNI_16        0x04

/***** define the error number *****/
#define ISO813_NoError        0
#define ISO813_CheckBoardError 1
#define ISO813_DriverOpenError 2
#define ISO813_DriverNoOpen   3
#define ISO813_AdError        4
#define ISO813_AdError2       -100.0
#define ISO813_OtherError     5
#define ISO813_GetDriverVersionError 6
#define ISO813_TimeOutError   0xffff

/***** Function of Test *****/
EXPORTS short CALLBACK ISO813_SHORT_SUB_2(short nA, short nB);
EXPORTS float CALLBACK ISO813_FLOAT_SUB_2(float fA, float fB);
EXPORTS WORD CALLBACK ISO813_Get_DLL_Version(void);
EXPORTS WORD CALLBACK ISO813_GetDriverVersion(
    WORD *wDriverVersion);

/***** Function of Driver *****/
EXPORTS WORD CALLBACK ISO813_DriverInit(void);
EXPORTS void CALLBACK ISO813_DriverClose(void);
EXPORTS WORD CALLBACK ISO813_Check_Address(WORD wBase);
```

```

/***** Function of AD *****/
EXPORTS WORD CALLBACK ISO813_AD_Hex( WORD wBase,
    WORD wChannel, WORD wGainCode );
EXPORTS WORD CALLBACK ISO813_ADs_Hex( WORD wBase,
    WORD wChannel, WORD wGainCode, WORD *wBuffer,
    DWORD dwDataNo );
EXPORTS float CALLBACK ISO813_AD_Float( WORD wBase,
    WORD wChannel, WORD wGainCode, WORD wBipolar,
    WORD wJump10v );
EXPORTS WORD CALLBACK ISO813_ADs_Float(WORD wBase,
    WORD wChannel, WORD wGainCode, WORD wBipolar,
    WORD wJump10v, float *fBuffer, DWORD dwDataNo );
EXPORTS float CALLBACK ISO813_AD2F(WORD whex,
    WORD wGainCode, WORD wBipolar, WORD wJump10v);
EXPORTS void CALLBACK ISO813_AD_SetReadyTicks(WORD wTicks);

/***** Declare 8253 Timer Interface *****/
EXPORTS WORD CALLBACK ISO813_TimerRead(WORD *wTicks);
EXPORTS void CALLBACK ISO813_TimerDelay(DWORD dwTicks);

/***** Function of DIO*****/
EXPORTS void CALLBACK ISO813_OutputByte(WORD wPortAddr,
    UCHAR bOutputVal);
EXPORTS void CALLBACK ISO813_OutputWord(WORD wPortAddr,
    WORD wOutputVal);
EXPORTS WORD CALLBACK ISO813_InputByte(WORD wPortAddr);
EXPORTS WORD CALLBACK ISO813_InputWord(WORD wPortAddr);

```

4. Function description

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

1. The test functions
2. The DI/O functions
3. The AD functions
4. The Driver functions
5. The Timer functions

- **The test functions**

1. ISO813_SHORT_SUB_2
2. ISO813_FLOAT_SUB_2
3. ISO813_Get_DLL_Version
4. ISO813_GetDriverVersion

- **The DI/O functions**

1. ISO813_OutputByte
2. ISO813_OutputWord
3. ISO813_InputByte
4. ISO813_InputWord

- **The AD functions**

1. ISO813_AD_Hex
2. ISO813_ADs_Hex
3. ISO813_AD_Float
4. ISO813_ADs_Float
5. ISO813_AD2F
6. ISO813_AD_SetReadyTicks

- **The Driver functions**

1. ISO813_DriverInit
2. ISO813_DriverClose
3. ISO813_Check_Address

- **The Timer functions**

1. ISO813_TimerRead
2. ISO813_TimerDelay

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

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

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

4.1. TEST Function

4.1.1. ISO813_SHORT_SUB_2

- **Description:**
Compute $nA - nB$ in **short** format, short=16 bits sign integer. This function is provided for testing purpose.
- **Syntax:**
`short ISO813_SHORT_SUB_2(short nA, short nB);`
- **Parameter:**

nA	: [Input] short integer
nB	: [Input] short integer
- **Return:**
`return = nA - nB → short integer`

4.1.2. ISO813_FLOAT_SUB_2

- **Description:**
Compute $fA - fB$ in **float** format, float=32 bits floating pointer number. This function is provided for testing purpose.
Note: This function doesn't support the DOS applications.
- **Syntax :**
`float ISO813_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. ISO813_Get_DLL_Version

- **Description:**
Read the software version of the ISO813.DLL.
Note: This function doesn't support the DOS applications.
- **Syntax:**
WORD ISO813_Get_DLL_Version(void) ;
- **Parameter:**
void
- **Return:**
return=0x200 → Version 2.00 **(WORD=16 bits unsigned integer)**

4.1.4. ISO813_GetDriverVersion

- **Description:**
This subroutine will get the version number about the device driver.
- **Syntax:**
WORD ISO813_GetDriverVersion(WORD *wDriverVersion) ;
- **Parameter:**
wDriverVersion : **[Output]** the address of wDriverVersion.
When wDriverVerion=0x210 → version 2.10
- **Return:**
NoError : successful in opening the device driver
GetDriverVersionError : fail in opening the device driver.

4.2. DI/DO Function

4.2.1. ISO813_OutputByte

- **Description:**
This subroutine will send the 8 bits data to the desired I/O port.
- **Syntax:**
`void ISO813_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

4.2.2. ISO813_OutputWord

- **Description:**
This subroutine will send the 16 bits data to the desired I/O port.
- **Syntax:**
`void ISO813_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

4.2.3. ISO813_InputByte

- **Description:**
This subroutine will input the 8 bit data from the desired I/O port.
- **Syntax:**
WORD ISO813_InputByte(WORD wPortAddr);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
- **Return:**
16 bits data with the leading 8 bits are all 0.

4.2.4. ISO813_InputWord

- **Description:**
This subroutine will input the 16 bit data from the desired I/O port.
- **Syntax:**
WORD ISO813_InputWord(WORD wPortAddr);
- **Parameter:**
wPortAddr : [Input] I/O port address, for example, 0x220
- **Return:**
16 bit data.

4.3. A/D Functions

4.3.1. ISO813_AD_Hex

- **Description:**

This subroutine will perform an A/D conversion by polling. The A/D converter is 12 bits for ISO-813. This subroutine will compute the result according to the **configuration code** (Section 1.3).

- **Syntax:**

```
WORD ISO813_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] Refer to Sec. 1.3.

- **Return:**

ISO813_TimeOutError	: A/D converter error (return 0xffff)
Other value	: The Hex value of A/D conversion

4.3.2. ISO813_ADs_Hex

- **Description:**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to ISO813_AD_Hex except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D converting at the ISA bus's max. speed. The sampling rate is about 10K samples/second which testing under Pentium-133 CPU. After A/D converting, the A/D data are stored in a buffer in Hex format. The **wBuf** is the starting address of this data buffer.

- **Syntax:**

WORD ISO813_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] Refer to Section 1.3.
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 these data from the buffer after calling this function.

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

- **Return:**

ISO813_TimeOutError	: A/D converter error
ISO813_NoError	: Operation is OK

4.3.3. ISO813_AD_Float

- **Description:**

This subroutine will perform an A/D conversion by polling. The A/D converter is 12 bits for ISO-813. This subroutine will compute the result according to the **configuration code** (Section 1.3).

- **Syntax:**

```
float ISO813_AD_Float(WORD wBase, WORD wChannel, WORD  
                      wGainCode, WORD wBipolar, WORD wJump10v);
```

- **Parameter:**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] A/D channel number,
wGainCode	: [Input] Refer to Section. 1.3.
wBipolar	: [Input] 1:Bipolar 0:Unipolar
wJump10v	: [Input] 1:10v 0:20v

- **Return:**

ISO813_AdError2	: A/D converter error (return -100.0)
Other value	: The floating-point value of A/D conversion

4.3.4. ISO813_ADs_Float

- **Description:**

This subroutine will perform a number of A/D conversions by polling. This subroutine is very similar to ISO813_AD_Float except that this subroutine will perform wCount of conversions instead of just one conversion. The A/D converting at the ISA bus's max. speed. The sampling rate is about 10K samples/second which testing under Pentium-133 CPU. Then the A/D data are stored in a data buffer in Float format. The **fBuf** is the starting address of this data buffer.

- **Syntax:**

```
WORD ISO813_ADs_Float(WORD wBase, WORD wChannel,  
                      WORD wGainCode, WORD wBipolar,  
                      WORD wJump10v, float fBuf[], WORD wCount);
```

- **Parameter:**

wBase	: [Input] I/O port base address, for example, 0x220
wChannel	: [Input] A/D channel number
wGainCode	: [Input] Refer to Section 1.3
wBipolar	: [Input] 1:Bipolar 0:Unipolar
wJump10v	: [Input] 1:10v 0:20v
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 these data from the buffer after calling this function.

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

- **Return:**

ISO813_TimeOutError	: A/D converter error
ISO813_NoError	: Operation is OK

4.3.5. ISO813_AD2F

- **Description:**

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

- **Syntax:**

float ISO813_AD2F(WORD wHexValue, WORD wGainCode,
WORD wBipolar, WORD wJump10v);

- **Parameter:**

wHexValue	: [Input] Hex Value 0 ~ 0x0FFF
wGainCode	: [Input] Refer to Sec. 1.3 for detail information
wBipolar	: [Input] 1:Bipolar 0:Unipolar
wJump10v	: [Input] 1:10v 0:20v

- **Return:**

ISO813_AdError2	: A/D converter error (return -100.0)
Other value	: The floating-point value of A/D conversion

4.3.6. ISO813_AD_SetReadyTicks

- **Description:**

After issues the request of AD conversion, it must take some delay to wait the AD convert completed. The delay time is appropriate 85 Ticks on 1.19318 MHz. **The default value of ReadyTicks is 85.** The user can increase the value of ReadyTicks to slow down the sampling rate, or decrement to improve the sampling rate.

The ReadyTicks is recorded in these library files, the user can set the ReadyTicks on the user's application program when its start up, or leave the default value to 85 ticks. This subroutine can let the user to adjust the ReadyTicks easily.

- **Syntax:**

void ISO813_AD_SetReadyTicks(WORD wTicks);

- **Parameter:**

wTicks	: [Input] The ticks for wait the AD convert completed.
--------	--

- **Return:**

None

4.4. Driver functions

4.4.1. ISO813_DriverInit

- **Description:**

This subroutine will open the device driver and allocate the resources for the device driver. The user must call this function once before calling any other functions.

- **Syntax:**

WORD ISO813_DriverInit(void);

- **Parameter:**

None

- **Return:**

ISO813_NoError : successful in opening the device driver
ISO813_DriverOpenError : fail in opening the device driver.

4.4.2. ISO813_DriverClose

- **Description:**

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

- **Syntax:**

void ISO813_DriverClose(void);

- **Parameter:**

None

- **Return:**

None

4.4.3. ISO813_Check_Address

- **Description:**

This subroutine will detect the ISO-813 in I/O base address = **wBase**. This subroutine will perform one A/D conversion, if success → found the ISO-813.

- **Syntax:**

WORD ISO813_Check_Address(WORD wBase);

- **Parameter:**

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

- **Return:**

ISO813_NoError : Found the ISO-813.
ISO813_CheckBoardError : Operation failure, Not Found

4.5. Timer functions

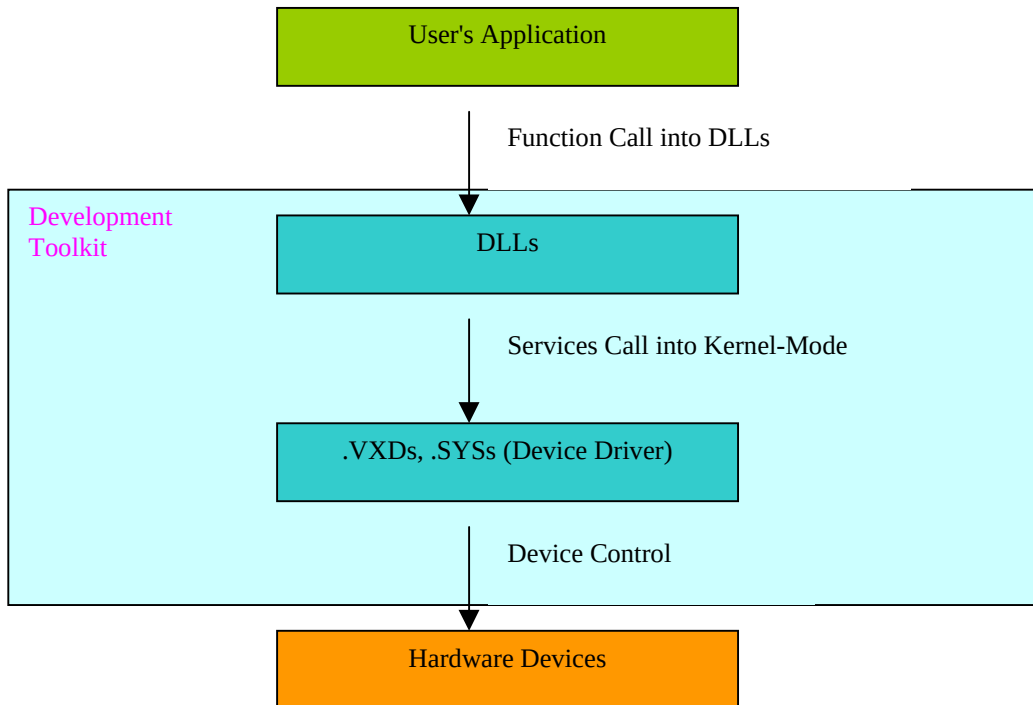
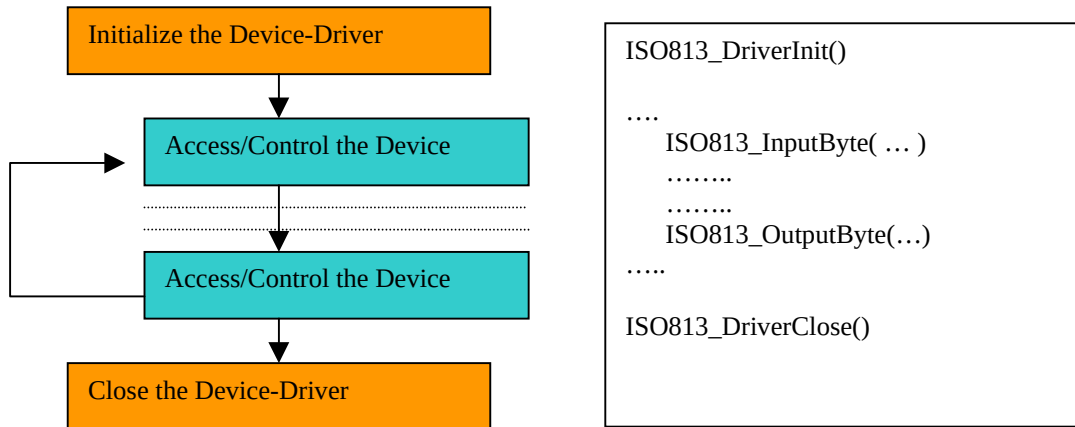
4.5.1. ISO813_TimerRead

- **Description:**
This subroutine will read the timer ticks (counter0) of 8253 chip from the user's system (main board).
- **Syntax:**
WORD ISO813_TimerRead(WORD *wTicks);
- **Parameter:**
wTicks : [Output] Return the ticks (0 ~ 65535)
which is read from 8253
- **Return:**
ISO813_DriverNotOpen: Use the ISO813_DriverInit() to initialize the driver.
ISO813_NoError : successful read the ticks.

4.5.2. ISO813_TimerDelay

- **Description:**
This subroutine will delay dwTicks ticks .
- **Syntax:**
void ISO813_TimerDelay(DWORD dwTicks);
- **Parameter:**
dwTicks : [Input] How many ticks that the user wants to delay.
- **Return:**
None

5. Program Architecture



6. Demo Program

- **For DOS**

--\Demo	
----\BC	➔ Borland C++ 3.1 Demo program
--\Demo1	➔ Polling, ADs_Hex Demo
--\Demo2	➔ Polling, ADs_Float Demo
----\TC	➔ Turbo C 2.0 Demo program
--\Demo1	➔ Polling, ADs_Hex Demo
--\Demo2	➔ Polling, ADs_Float Demo
----\MSC	➔ Microsoft C 6.0 Demo program
--\Demo1	➔ Polling, ADs_Hex Demo

- **For Windows 95/98/NT**

--\Demo	
----\Delphi3	➔ Delphi 3.0 Demo program
--\01	➔ Driver testing Demo program
--\03	➔ Polling, ADs_Hex Demo
--\04	➔ Polling, ADs_Float Demo
----\BCB3	➔ Borland C++ Builder 3.0 Demo program
--\01	➔ Driver testing Demo program
--\03	➔ Polling, ADs_Hex Demo
--\04	➔ Polling, ADs_Float Demo
----\VB5	➔ Visual Basic 5.0 Demo program
--\01	➔ Driver testing Demo program
--\03	➔ Polling, ADs_Hex Demo
--\04	➔ Polling, ADs_Float Demo
----\VC5	➔ Visual C++ 5.0 Demo program
--\01	➔ Polling, ADs_Float Demo

7. Problems Report

Technical support is available at no charge as described below. The best way to report problems is send electronic mail to

Service@icpdas.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 Operating Systems that you using? 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: Service@icpdas.com
Web-Site: <http://www.icpdas.com>