Decision Library (DecLib)
V 3.0

A Universal
Application Programming Interface
To
PCI Direct I/O Card

Users Manual

Design & Implementation by
Decision Computer International Company

Contents

1. Introduction

This document provides the Decision Library (DecLib) Specifications, including all function calls, installation requirements, and operating procedures.

Disclaimer:

2. Features

The Decision Library (DecLib) was created to provide a standard way to access the functionality provided by all PCI Direct I/O cards products. Specifically, the DecLib provides the following features:

- PCI Device Support
  DecLib support all DECISION PCI Direct I/O cards. If you would like to use ISA cards, please use Dynamic Industrial Interface (DII).

- Platform-independent
  The library is compatible under Windows 98, windowsME, Windows 2000 and windows XP. The compatibility under these operation systems guarantees that programs written for either operating system will work unchanged on the other, even without recompilation.

- Abstracts Card Functionality from Card Design
  The interface concentrates on a card's functionality and hides the user from having to know specifics about the card design, for example, which port needs to be accessed in order to access specific functionality.

- Multiple Card Support
  Old DecLib don't support multiple card access, but it does now. You could access device by its name or by its information (device type, index).

- Device Naming
  Each device can be given it's own name, specified by the function "Device Naming" on DII Control Panel Applet. This allows easier access of devices and cards in situations, where multiple cards of the same type are installed.

- Programming Language Independent
  The library provides a language independent way to access the industrial I/O cards, by using a Dynamic-Link-Library architecture.

3. Distribution Contents

The Decision Library (DecLib) Distribution consists of the following components:

1. A Control Panel Applet to view the device information and for device naming/renaming.

2. WDM Device Driver for Windows 98, ME, 2000, XP.

3. A Win32 DLL (file DecLib.dll) for accessing the driver. In order to access the DLL, a C-Header file and a visual Basic file containing the function declarations are included.

4. The Visual C++ sample applications
   The Sample applications demonstrate the use of the driver API by graphically.The applications runs unchanged under Windows 98/ME/2k/XP. They were developed using Visual C++ 6.0. The full source codes are included.

   After installation, you can find a folder “DecLib Test Programs” under your installation directory, and all sample applications are in it (includes VC++ and VB samples).

5. The Visual Basic sample applications
   The Visual Basic samples are similar to the Visual C++ sample, except it’s written using Visual Basic Version 6.0.

   After installation, you can find a folder “DecLib Test Programs” under your installation directory, and all sample applications are in it (includes VC++ and VB samples).

4.Install and Uninstall

This chapter specify installation and uninstalltion for DDI software and Decision device. DDI (Decision Device Interface) software contains two kinds of library, DII and DecLib, and users could choose each of them during installing DDI software (Notice that DII and DecLib can not been mix-use).

4.1 DDI (Decision Device Interface) Software Installation:

Step 1: Execute the DDI install program (DDI_130.exe)

Step 2: Following the specifications of DDI install program. During installtion, program would query which library you use. To choose your favorite library and continue to finish installing.

After installation, you could find folder “Decision Device Interface 3.0” on [Start] -> [Programs].

4.2 DDI Software Uninstallation:

You could uninstall by following two methods:

Method 1: click [Start] -> [Programs] -> [Decision Device Interface 3.0] -> Remove Decision Device Interface 3.0

Method 2: On the “Add/Remove Programs” on Control Panel, You could find “Decision Device Interface 3.0”, and remove it.

4.3 Device Installation:

Step1: Plug your card on empty PCI or ISA slot, and boot.

Step2: When entering operation system, hardware wizard would find your card automatically

Step3: following the hardware wizard to continue installation. During installing, you might specify your driver location automatically.

Step4: After installing driver, computer may be reboot. And you would see youir device on “Device Manager”.

4.4 Device Unnstallation:

Step1: On [Control Panel] -> [System] -> [Hardware] -> [Device Manager], choose the device you would like to uninstall and click right mouse button --> uninstall.

Step2: Under the folder “C:\Windows\inf” (2k/XP is on “C:\WinNt\inf”), you could find some oem#.inf files (# represent digits). To delete these oem#.inf files that contains your card information (open these .inf as test program).

Step3: Under the folder “C:\Windows\system32\drivers” (2k/XP is on “C:\WINNT\system32\drivers”), you could find driver for your card, delete them. (file name: XXXX.sys, XXXX represent your device name).

5. Device Naming

The Decision Library (DecLib) has a powerful feature: Device Naming.

After the installation of devices, each device is assign a default unique name. And you could view or fix device's name on the DII Control Panel Applet. When creating your application program, you can then access that device using the name, or you may prompt the user for the name, or even let the user graphically browse for the correct device for your application.

This technique has one important feature:

- More clarity when using multiple devices
  Using device names, you can now access more easily and clearly multiple devices of the same type in your machine. For example, if you have 3 pieces of 8 Relay 8 Photo Isolator cards in your machine, you could access these cards not only by their type and index, which often lead to confusion. You can now just assign a name to each card on DII Control Panel Applet, especially a meaningful name.

There is something to care when using DII Control Panel Applet for naming/renaming:

- Please reboot your computer after you change device's name or the device can't work.
- You might name/rename your device only using characters, digits, space and underscore. Others might cause problem.
- Don't name/rename the same name to different devices or it would cause problem.
- The maximum length of device name is 100 bytes (100 characters).

6. Backward compatibility

This new version of Decision Library (DecLib 3.0) would good backward compatibility feature. You won't change any piece of your old program codes (but something would be concerned, list below), because the interface of the new-version DecLib is the same to that of the old-version one. You would only replace new file DecLib.DLL and DecLib.h with old ones, and then your programs could work well as before.

Some differences to old version of DecLib (using WDM 1.20v device driver):

- Internal implementation of some functions is changed, but the interfaces of all functions are the same to old-version DecLib.

- Because old-version DecLib don't support multiple-card access, the function *CloseDecPort* only closes the device that used in program in past. But now, this function closes all devices that were opened before this function been called. If you would like to close only one device at one time, you could use the new function CloseDevice.

- Some new functions add to this version of library. Like *CloseDevice*, *GetDevicePortAddress*.

- The meaning of device index is not the same to the past. The older one represents its PCI bus position, but now is another. You don't need to know what it means. The only thing you needs to know is that device index of any installed device is unique and you could check it on Device Information of DII Control Panel Applet.

7. DecLib function calls and Device Type definition

Since the Dll was developed in the C++ language, some data types used may not be present in the programming language you want to use.

Please find the following data type conversion table for your convenience:

BOOL	A 32-bit integer, either 0 (FALSE) or 1 (TRUE)
LPTSTR	A 32-bit flat pointer to a zero terminated string
UCHAR	Unsigned char
WORD	double bytes

Also note that the DLL employs the Standard Call (Pascal) calling mechanism, which is used for all system Dlls as well and is compatible with Visual Basic.

Below is Device Type definition:

DECISION_PCI_IND_CARD	0x0001
DECISION_PCI_8255_CARD	0x0002
DECISION_PCI_4P4R_CARD	0x0003
DECISION_PCI_8P8R_CARD	0x0004
DECISION_PCI_16P16R_CARD	0x0005
DECISION_PCI_M8255_CARD	0x0007
DECISION_PCI_12ADDA_CARD	0x0008
DECISION_PCI_14ADDA_CARD	0x0009

7.1 Functions to open and close Devices

OpenDecPort

This function opens devices port for further access.

Declaration

```
BOOL OpenDecPort ( void );
```

Parameters

no any parameter.

Return value

TRUE if successful, FALSE otherwise.

Example

```
OpenDecPort ();
```

Remarks

This function is provided for backward compatibility. Now open-device facility is implemented by functions that get device base address (See 7.2 “Function to get device base address” for detail). New users could directly use functions of “GetDevicePortAddress*” series.

CloseDecPort

This function closes all devices.

Declaration

```
        void CloseDecPort ( void );
```

Parameters

no any parameter.

Return value

no any return value.

Example

```
CloseDecPort ();
```

CloseDevice

This function closes a specified device;

Declaration

```
BOOL CloseDevice ( unsigned short      usAddress);
```

Parameters

usAddress	Device base address for that device you would like to close.

Return value

TRUE if successful, FALSE otherwise.

If FALSE, the device base address you specified is not correct.

Example

```
CloseDevice (usBaseAddress);
```

7.2 Functions to get device base address

GetDevicePortAddress

The function return device base address. User provides device type, and this function choose any one installed device that fits to this type to open.

Declaration

```
BOOL GetDevicePortAddress ( unsigned int      DeviceType,
                            unsigned short  *pusAddress
                           );
```

Parameters

DeviceType	Specfied device type to open. For more information, please see the chapter “DecLib function calls and Device Type definition”.
pusAddress	A pointer to a unsigned short variable receiving the base address of opened device.

Return value

TRUE if successful, FALSE otherwise.

Example

```
 unsinged short usBaseAddress;

if (!GetDevicePortAddress (DECISION_PCI_IND_CARD, &usBaseAddress))
{
        printf (“ Find no any PCI IND Card”);
        return FALSE;
}
```

Remarks

Function GetDevicePortAddress not only gets device base address, but implements open-device facility now. It means that user don't use function OpenDecPort any more. But older Program that uses function OpenDecPort won't be changed because we also reserves OpenDecPort for compatibility.

GetDevicePortAddressEx

The function return device base address. User provides nothing, but this function choose any one installed device to open.

Declaration

```
    BOOL GetDevicePortAddressEx ( unsigned short   *pusAddress );
```

Parameters

pusAddress	A pointer to a unsigned short variable receiving the base address of opened device.

Return value

TRUE if successful, FALSE otherwise.

Example

```
    unsigned short usBaseAddress;

    if (!GetDevicePortAddressEx (&usBaseAddress))
    {
            printf (" Find no any Decision Card");
            return FALSE;
    }
```

Remarks

Function GetDevicePortAddressEx not only gets device base address, but implements open-device facility now. It means that user don't use function OpenDecPort any more. But older Program that uses function OpenDecPort won't be changed because we also reserves OpenDecPort for compatibility.

GetDevicePortAddressEx2

The function return device base address. User provides device type and device index, and then this function opens this specified device.

Declaration

```
BOOL GetDevicePortAddressEx2 ( unsigned int      DeviceType,
                               unsigned int      CardID,
                               unsigned short   *pusAddress
                             );
```

Parameters

DeviceType	Device type for specified device.
CardID	Device index for specified device. Please see the chapter “Backward Compatibility” for more information.
pusAddress	A pointer to a unsigned short variable receiving the base address of opened device.

Return value

TRUE if successful, FALSE otherwise.

Example

```
unsigned short usBaseAddress;

if (!GetDevicePortAddressEx2 (DECISION_PCI_IND_CARD, 1, &usBaseAddress))
{
        printf (“Find no specified PCI IND Card”);
        return FALSE;
}
```

Remarks

Function GetDevicePortAddressEx2 not only gets device base address, but implements open-device facility now. It means that user don't use function OpenDecPort any more. But older Program that uses function OpenDecPort won't be changed because we also reserves OpenDecPort for compatibility.

GetDevicePortAddressEx3

The function return device base address. User provides device name, and then this function opens this specified device.

Declaration

```
BOOL GetDevicePortAddressEx3 ( LPTSTR          lpszDeviceName,
                               unsigned short  *pusAddress
                             );
```

Parameters

lpszDeviceName	The name of the device to open.
pusAddress	A pointer to a unsigned short variable receiving the base address of opened device.

Return value

TRUE if successful, FALSE otherwise.

Example

```
unsigned short usBaseAddress;

if (!GetDevicePortAddressEx3 ("PCI_IND0001", &usBaseAddress))
{
        printf (" Find no PCI device: PCI_IND0001");
        return FALSE;
}
```

Remarks

Function GetDevicePortAddressEx3 not only gets device base address, but implements open-device facility now. It means that user don't use function OpenDecPort any more. But older Program that uses function OpenDecPort won't be changed because we also reserves OpenDecPort for compatibility.

7.3 Functions for I/O Access

outportb

This function writes byte-data to device.

Declaration

```
BOOL outportb ( unsigned short      usAddress,
                unsigned char       ucData
              );
```

Parameters

usAddress	The starting address to writeA variable receiving the address of opened device
ucData	Byte-Data to write

Return value

TRUE if successful, FALSE otherwise.

Example

```
unsigned short usBaseAddress;     // device base address
                    :
                    :
                    :
unsigned short usOffset = 0;     // offset; port number
BOOL bRetCode;

bRetCode = outportb (usBaseAddress + usOffset, 1);
if (!bRetCode)
{
    // error handle code
}
```

outport

This function writes WORD-data (double bytes) to device.

Declaration

```
        BOOL outportb ( unsigned short          usAddress,
                         unsigned short          usData
                       );
```

Parameters

usAddress	A variable receiving the address of opened device
ucData	WORD-Data to write

Return value

TRUE if successful, FALSE otherwise.

Example

```
        unsigned short usBaseAddress;      // device base address
                                :
                                :
                                :
        unsigned short usOffset = 0;       // offset; port number
        BOOL bRetCode;

        bRetCode = outportb (usBaseAddress + usOffset, 1);
        if (!bRetCode)
        {
           // error handle code
        }
```

inportb

This function reads byte-data from device.

Declaration

```
        unsigned char inportb ( unsigned short   usAddress );
```

Parameters

usAddress	A variable receiving the address of opened device

Return value

byte-Data that read from device

Example

```
        unsigned short usBaseAddress;   // device base address
                                :
                                :
                                :
        unsigned char  ucData;                // returned data
        unsigned short usOffset = 0;         // offset; port number

        ucData = inportb (usBaseAddress + usOffset);
```

inport

This function reads WORD-data (double bytes) from device.

Declaration

```
unsigned short outportb ( unsigned short     usAddress );
```

Parameters

usAddress	A variable receiving the address of opened device

Return value

word-Data that read from device

Example

```
unsigned short usBaseAddress;       // device base address
                          :
                          :
                          :
unsigned short  usData;
unsigned short usOffset = 0;        // offset; port number

usData = inport (usBaseAddress + usOffset);
```

8. Using the DecLib with different programming languages

This chapter provides an overview about how to best utilize the DecLib in various programming languages.

If you experience difficulties calling the DecLib functions from your programming language, or are using a programming language not covered in this documentation, please feel free to visit our web-site, to which we will post updated information regarding DecLib programming issues. You may also contact our technical support through our web-site.

8.1. C++

Since the DecLib DLL was developed using C++, you may easily use it to access PCI direct I/O devices. For this purpose, a C++ header file ("DecLib.h") as well as an import library ("DecLib.lib") are being shipped with the interface library. Make sure that you have installed the development release, not the retail release, which does not include support programming files.

In your C/C++ source code files, just include the "DecLib.h" include file, then you can use any of the functions provided by the DecLib DLL. Be sure to include the import library "DecLib.lib" during the linking step of your application, so your applications successfully references the actual interface DLL.

8.2. Visual Basic

If you are using Visual Basic to access any I/O Devices supported by the Decision Library (DecLib), you can call the DecLib DLL directly. But before that, you should import them.

You may also consult the Visual Basic sample application for more information about using Visual Basic to access the Decision Library (DecLib).

9. Technical Support And Feedback

We believe that customer input is the most valuable source for creating successful products.

We continuously update and extend the Decision Library (DecLib) with new functionality, for specific devices, for specific applications, to meet your specific needs, and provide supportive products around the DecLib.