

**Decision Computers Library
DecLib V 2.2**

**Application Programmers Interface
Users Manual**

1. Functions to Select and Load Devices

LoadDeviceType

This function selects a device of a particular type, i.e. a 4 bit Relay Card. It is useful if there is only one card of a type in the PC and this card will always be the one to be used.

Declaration

```
BOOL LoadDeviceType (    DWORD    dwDeviceType,  
                        LPDWORD   pdwDeviceIndex );
```

Parameters

dwDeviceType The type of device to open, which can be any of the following relating to the card you wish to load.

```
DECISION_PCI_IND_CARD  
DECISION_PCI_8255_CARD  
DECISION_PCI_4P4R_CARD  
DECISION_PCI_8P8R_CARD  
DECISION_PCI_16P16R_CARD  
DECISION_PCI_WATCHDOG_CARD  
DECISION_PCI_M8255_CARD  
DECISION_PCI_12ADDA_CARD  
DECISION_PCI_14ADDA_CARD
```

dwDeviceIndex A pointer to which the function will return the index value for the card. This index value should be retained ready for subsequent OpenDevice() calls.

Return Value

TRUE (Non Zero) on success, or FALSE (zero) on failure.

Select Device

This function lets the operator select a device from a list of installed devices. It is useful if there are more than one card of a particular type in the PC.

Declaration

```
BOOL SelectDevice ( LPDWORD   pdwDeviceIndex );
```

Parameters

dwDeviceIndex A pointer to which the function will return the index value for the card. This index value should be retained ready for subsequent OpenDevice() calls.

Return Value

TRUE (Non Zero) on success, or FALSE (zero) on failure.

2. Functions to open, close and identify Devices

OpenDevice

This function opens a device for further access.

Declaration

```
HANDLE OpenDevice ( DWORD dwDeviceIndex );
```

Parameters

dwDeviceIndex The device index returned from the call to LoadDeviceType() or SelectDevice()

Return value

A valid handle representing the device, or INVALID_HANDLE_VALUE (-1) if an error occurred.

CloseDevice

This function closes an open device.

Declaration

```
BOOL CloseDevice ( HANDLE hDevice );
```

Parameters

hDevice A valid device handle, previously obtained from OpenDevice()

Return value

TRUE if successful, FALSE otherwise.

GetDeviceType

This function determines the type of open device.

Declaration

```
int GetDeviceType ( HANDLE hDevice );
```

Parameters

hDevice A valid device handle, previously obtained from OpenDevice()

Return value

The device type value which will be one of the following types

```
DECISION_PCI_IND_CARD  
DECISION_PCI_8255_CARD  
DECISION_PCI_4P4R_CARD  
DECISION_PCI_8P8R_CARD  
DECISION_PCI_16P16R_CARD  
DECISION_PCI_WATCHDOG_CARD  
DECISION_PCI_M8255_CARD  
DECISION_PCI_12ADDA_CARD  
DECISION_PCI_14ADDA_CARD
```

3. Functions for digital input/output

SetDigitalBit

This function sets or clears a single bit on a digital output line.

Declaration

```
BOOL SetDigitalBit ( HANDLE    hDevHandle,  
                    DWORD     dwLine,  
                    BOOL      bState );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice
dwLine The index of the bit on the card to manipulate. The first bit has index 0.
bState The new state of the bit, either set (1/TRUE) or cleared (0/FALSE)

Return value

TRUE if successful, FALSE otherwise.

SetDigitalByte

This function outputs a complete byte to a digital output port of a device.

Declaration

```
BOOL SetDigitalByte ( HANDLE    hDevHandle,  
                     WORD      wPort,  
                     BYTE      byState );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice
wPort The port on the device to output the value to.
byState The data to output to the port.

Return value

TRUE if successful, FALSE otherwise.

SetDigitalWord

This function outputs a word to a digital output port of a device. The word is treated as two bytes or 'short' and is written to two successive output ports.

Declaration

```
BOOL SetDigitalWord ( HANDLE    hDevHandle,  
                     WORD      wPort,  
                     WORD      usState );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice
wPort The index of the bit on the card to manipulate. The first bit has index 0.
usState The data to output to the port

Return value

TRUE if successful, FALSE otherwise.

GetDigitalBit

This function returns the state of a single bit on an input port of a device

Declaration

```
BOOL GetDigitalBit ( HANDLE hDevHandle,  
                    DWORD dwLine );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

dwLine The index of the bit on the card to manipulate. The first bit has index 0.

Return value

The state of the bit, either set (1/TRUE) or cleared (0/FALSE)

Example

```
DWORD dev;  
SelectDevice ( &dev )  
HANDLE hDevice = OpenDevice ( dev );  
BOOL bState = GetDigitalBit ( hDevice, 0 ); // reads the state of the first bit  
CloseDevice ( hDevice);
```

GetDigitalByte

This function reads a byte (8bits) from a digital input port of a device.

Declaration

```
BYTE GetDigitalByte( HANDLE hDevHandle,  
                    WORD wPort );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

wPort The index of the port on the card to read. The first port has index 0.

Return value

The current state of the port

Example

```
DWORD dev;  
SelectDevice ( &dev )  
HANDLE hDevice = OpenDevice ( dev );  
BYTE byState = GetDigitalByte( hDevice, 0 );  
CloseDevice (hDevice);
```

GetDigitalWord

This function reads a word (16bits) from a digital input port of a device. If the port is only 8 bits wide this function will read data from two consecutive devices to obtain the word value.

Declaration

```
WORD GetDigitalWord( HANDLE hDevHandle,  
                    WORD wPort );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

wPort The index of the port on the card to read. The first port has index 0.

Return value

The current state of the port.

GetOutputBit

This function returns the state of a single bit on a digital output of a device

Declaration

```
BOOL GetOutputBit ( HANDLE hDevHandle,  
                    DWORD dwLine );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

dwLine The index of the bit on the card to manipulate. The first bit has index 0.

Return value

The state of the bit, either set (1/TRUE) or cleared (0/FALSE)

Example

```
DWORD dev;  
SelectDevice ( &dev )  
HANDLE hDevice = OpenDevice ( dev );  
BOOL bState = GetOutputBit ( hDevice, 0 ); // reads the state of the first bit  
CloseDevice ( hDevice);
```

GetOutputByte

This function reads the current state of a digital output port of a device. This function reads a whole port byte i.e. 8 bits.

Declaration

```
BYTE GetOutputByte( HANDLE hDevHandle,  
                    WORD wPort );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

wPort The index of the output port on the card to read. The first port has index 0.

Return value

The current state of the output port

Example

```
DWORD dev;  
SelectDevice ( &dev )  
HANDLE hDevice = OpenDevice ( dev );  
BYTE byState = GetOutputByte( hDevice, 0 );  
CloseDevice (hDevice);
```

GetOutputWord

This function reads the current state of two consecutive digital output ports of a device i.e. this reads a 16 bit word.

Declaration

```
WORD GetOutputWord( HANDLE hDevHandle,  
                    WORD wPort );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

wPort The index of the output port on the card to read. The first port has index 0.

Return value

The current 16 bit state of the two output ports starting at *wPort*

Example

```
DWORD dev;  
SelectDevice ( &dev )  
HANDLE hDevice = OpenDevice ( dev );  
WORD wdState = GetOutputWord( hDevice, 0 );  
CloseDevice (hDevice);
```

4. Functions to access the pulse counters

EnablePulseCounting

This function enables pulse counting for the card specified.

Declaration

```
BOOL EnablePulseCounting ( HANDLE hDevHandle,  
                           DWORD dwNumberOfPorts,  
                           BOOL bCountUp = TRUE,  
                           UINT uiPollInterval = 100 );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice
dwNumberOfPorts The number of ports to support or scan. To monitor all ports, specify 0.

bCountUp Direction of counters. TRUE for count up (default), FALSE for count down

uiPollInterval Cards without interrupts are polled at the interval set here. Timings are in mS the default is 100mS or ten times a second. The faster this is set the more resources are used and the likelihood of catching switch bounce is increased.

Return value

TRUE if successful, FALSE otherwise.

Remarks

At initialisation the counters are reset to zero.

DisablePulseCounting

This function disables pulse counting for the card specified.

Declaration

```
BOOL DisablePulseCounting ( HANDLE hDevHandle);
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice

Return value

TRUE if successful, FALSE otherwise.

SetPulseCounter

This function is called to reconfigure a counter into up-or down pulse counter

Declaration

```
BOOL SetPulseCounter (      HANDLE hDevHandle,  
                           DWORD dwCounter,  
                           DWORD dwValue,  
                           BOOL bCountUp = TRUE );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice.
dwCounter The counter to reconfigure. The counter corresponds to a single digital I/O line. The first I/O line is specified with 0.

dwValue The value to preset the counter.

bCountUp If TRUE the counter is configured to count up (default) else the counter is configured to be a downcounter,

Return value

TRUE if successful, FALSE otherwise.

GetPulseCounterValue

This function is called to reconfigure a counter into up-or down pulse counter

Declaration

```
BOOL GetPulseCounter( HANDLE hDevHandle,  
                       DWORD dwCounter,  
                       BOOL bReset = FALSE );
```

Parameters

hDevHandle A valid device handle, previously obtained from OpenDevice.

dwCounter The counter to reconfigure. The counter corresponds to a single digital I/O line. The first I/O line is specified with 0.

bReset Specifies whether the counter shall be reset while reading. If reset, an up-counter is set to zero, and a down-counter is set to the initial value again. If this parameter is FALSE, the counter is not reset.

Return value

TRUE if successful, FALSE otherwise.