# Introduction to the DLL for the USB Interface Board K8061

The K8061 interface board has 8 digital input channels and 8 digital output channels. In addition, there are 8 analogue inputs, 8 analogue outputs and one PWM output. The number of inputs/outputs can be further expanded by connecting more (up to a maximum of eight) cards to the PC's USB connectors. Each card is given its own identification number by means of three jumpers, A1, A2 and A3 (see table 1 below for card numbering).

All communication routines are contained in a Dynamic Link Library (DLL) K8061.DLL.

This document describes all functions and procedures of the DLL that are available for your application programme. Calling the functions and procedures exported by the DLL, you may write custom Windows (98SE, 2000, Me, XP, Vista, Windows 7) applications in Delphi, Visual Basic, C++ Builder or any other 32-bit Windows application development tool that supports calls to a DLL.

A complete overview of the procedures and functions that are exported by the K8061.DLL follows. At the end of this document there are listings of example programmes in order to gain an insight as to how to construct your own application programmes. The examples are written in Delphi, Visual Basic and C++ Builder. In the listings there are full declarations for the DLL function and procedures.

| A1 | A2 | A3 | CARD ADDRESS |
|---|---|---|---|
| ON | ON | ON | 0 |
| OFF | ON | ON | 1 |
| ON | OFF | ON | 2 |
| OFF | OFF | ON | 3 |
| ON | ON | OFF | 4 |
| OFF | ON | OFF | 5 |
| ON | OFF | OFF | 6 |
| OFF | OFF | OFF | 7 |

**TABLE 1: Jumper A1, A2, A3 Settings**

**IMPORTANT NOTES!**

1. The library MPUSBAPI.DLL must be copied to the same folder with the K8061.DLL. Best place to copy these files is the Windows' folder System32 or SysWOW64. Also these files can be copied to your application folder.
2. The attached driver (in subfolder USB_driver, v1.0.0.6) must be installed to use this version (v3.0.0.1) of the K8061.DLL.
3. The address settings must be done before the USB cable is connected to the K8061 card or before turning the PC on.
4. If the USB cable is disconnected all the digital and analog outputs are reset to zero.
5. All the examples in the function and procedure description section are written for Delphi.
6. In Delphi **longint** is a 32-bit integer.

# Overview of the Procedures and Functions of the K8061.DLL

**General procedures**

| | |
|---|---|
| `OpenDevice` | *Opens the communication link to the K8061 device* |
| `CloseDevices` | *Closes the link to all open K8061 devices* |
| `CloseDevice` | *Closes the link to one open K8061 device* |
| `PowerGood` | *Checks that IC6 is working properly* |
| `Connected` | *Checks that USB connection to the card is valid* |
| `ReadVersion(CardAddress, Buffer)` | *Reads the software version of IC3 and IC6* |

**Analogue to Digital converter procedures**

| | |
|---|---|
| `ReadAnalogChannel(CardAddress, Channelno)` | *Reads the status of one analogue input-channel* |
| `ReadAllAnalog(CardAddress, Buffer)` | *Reads the status of all analogue input-channels* |

**Digital to Analogue conversion procedures**

| | |
|---|---|
| `OutputAnalogChannel(CardAddress,Channel,Data)` | *Sets the analogue output channel according to the data* |
| `OutputAllAnalog(CardAddress, Buffer)` | *Sets all analogue output channels according to the data* |
| `ClearAnalogChannel(CardAddress, Channel)` | *Sets the analogue output channel to minimum* |
| `ClearAllAnalog(CardAddress)` | *Sets all analogue output channels to minimum* |
| `SetAnalogChannel(CardAddress, Channel)` | *Sets the analogue output channel to maximum* |
| `SetAllAnalog(CardAddress)` | *Sets all analogue output channels to maximum* |

**PWM Output procedure**

| | |
|---|---|
| `OutputPWM(CardAddress, Data)` | *Sets the PWM output according to the data* |

**Digital Output procedures**

| | |
|---|---|
| `OutputAllDigital(CardAddress, Data)` | *Sets the digital outputs according to the data* |
| `ClearDigitalChannel(CardAddress, Channel)` | *Clears the output channel* |
| `ClearAllDigital(CardAddress)` | *Clears all output channels* |
| `SetDigitalChannel(CardAddress, Channel)` | *Sets the output channel* |
| `SetAllDigital(CardAddress)` | *Sets all output channels* |

**Digital Input procedures and functions**

| | |
|---|---|
| `ReadDigitalChannel(CardAddress, Channel)` | *Reads the status of the input channel* |
| `ReadAllDigital(CardAddress, Buffer)` | *Reads the status of all the input channels* |

**Readback procedures and functions**

`ReadBackDigitalOut(CardAddress)`
*Reads back the digital output value of the card*

`ReadBackAnalogOut(CardAddress, Buffer)`
*Reads back the current D/A output values of the card*

`ReadBackPWMOut(CardAddress)`
*Reads back the status of the PWM output*

# Procedures and Functions of the K8061.DLL

## OpenDevice

*Syntax*
```
FUNCTION OpenDevice: Longint;
```

*Result*
`Longint:` If succeeded the return value will be the card address read from the K8061 hardware. The card address is value between 0 and 7 which corresponds to the jumper (A1, A2, A3) setting on the K8061 card. Use this value in the next function calls to access this card.
If more K8061 cards are connected to the PC, repeat this function call until all the cards are opened.
Return value -1 indicates that all K8061 cards are opened.
Return value -2 indicates that no card was found.

*Description*
Opens a communication link to the K8061 card. Loads the drivers needed to communicate via the USB port. This procedure must be performed before any attempts to communicate with the K8061 card.

*Example*
```
var CardAddress: longint;
BEGIN
   CardAddress:=OpenDevice;
   // Opens the link to card and returns the "CardAddress"
END;
```

## CloseDevices

*Syntax*
```
PROCEDURE CloseDevices;
```

*Description*
Unloads the communication routines for K8061 cards and unloads the driver needed to communicate via the USB port. This is the last action of the application program before termination.

*Example*
```
BEGIN
```

```
CloseDevices; // Communication to all the K8061 cards is closed
  END;
```

## CloseDevice

*Syntax*
```
PROCEDURE CloseDevice(CardAddress: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.

*Description*
Closes the communication routines for one K8061 card.

*Example*
```
  BEGIN
     CloseDevice(0); // Communication to the K8061 card address 0 is closed
  END;
```

## ReadAnalogChannel

*Syntax*
```
FUNCTION ReadAnalogChannel(CardAddress: Longint; Channel: Longint): Longint;
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the AD channel whose status is to be read.

*Result*
`Longint:` The corresponding Analogue to Digital Converter data is read.

*Description*
The input voltage of the selected 8-bit Analogue to Digital converter channel is converted to a value which lies between 0 and 1023.

*Example*
```
  var data: longint;
  BEGIN
      data := ReadAnalogChannel(0, 1);
      // AD channel 1 of card address 0 is read to variable 'data'
  END;
```

## ReadIAllAnalog

*Syntax*
```
PROCEDURE ReadAllAnalog(CardAddress: Longint; Buffer: Pointer);
```

*Parameter*
`CardAddress:` The address of previously opened card.
`Buffer:` Pointer to array of long integers where the data will be read.

*Description*
The status of all eight Analogue to Digital Converters are read to an array of long integers.

*Example*
```
var Buffer: Array[0..7] of Longint;
begin
```

```
ReadAllAnalog(0, @Buffer[0]);
  // Reads all the analog data from the K8061 card address 0 to array
'Buffer'
end;
```

## OutputAnalogChannel

*Syntax*

```
PROCEDURE OutputAnalogChannel(CardAddress: Longint; Channel: Longint; Data:
Longint);
```

*Parameters*

CardAddress: The address of previously opened card.
Channel: Value between 1 and 8 which corresponds to the channel number whose data is to be set.
Data: Value between 0 and 255 which is to be sent to the 8-bit Digital to Analogue Converter .

*Description*

The indicated 8-bit Digital to Analogue Converter channel is altered according to the new data. This means that the data corresponds to a specific voltage. The value 0 corresponds to a minimum output voltage (0 Volt) and the value 255 corresponds to a maximum output voltage (+5V or 10V according to the jumper setting).

*Example*

```
  BEGIN
    OutputAnalogChannel (0, 1, 127);
    // DA channel 1 of card address 0 is set to 2.5V
    // or 5V according to the DA jumper settings
  END;
```

## OutputAllAnalog

*Syntax*

```
PROCEDURE OutputAllAnalog(CardAddress: Longint; Buffer: Pointer);
```

*Parameters*

CardAddress: The address of previously opened card.
Buffer: Pointer to an array of long integers which hold the data to be output to the Digital to Analogue Converters.

*Description*

All eight Digital to Analogue Converter channels are altered according to the new data. This means that the data corresponds to a specific voltage. The value 0 corresponds to a minimum output voltage (0 Volt) and the value 255 corresponds to a maximum output voltage (+5V or +10V according to the jumper setting).

*Example*

*Example*

```
var Buffer: Array[0..7] of Longint;
i: integer;
begin
  for i:=0 to 7 do Buffer[i]:=10*i;
  OutputAllAnalog(0, @Buffer[0]);
  // Outputs the analog data from array 'Buffer' to the K8061 card
end;
```

## ClearAnalogChannel

*Syntax*
```
PROCEDURE ClearAnalogChannel(CardAddress: Longint; Channel: Longint);
```

*Parametes*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the 8-bit DA channel number in which the data is to be erased.

*Description*
The selected DA-channel is set to minimum output voltage (0 Volt).

*Example*
```
BEGIN
  ClearAnalogChannel (0, 1);
  // DA channel 1 of card address 0 is set to 0V
END;
```

---

## ClearAllAnalog

*Syntax*
```
PROCEDURE ClearAllAnalog(CardAddress: Longint);
```

*Paramete*
`CardAddress:` The address of previously opened card.

*Description*
All DA-channels are set to minimum output voltage (0 Volt) .

*Example*
```
BEGIN
  ClearAllAnalog(0);
  // All DA channels of card address 0 are set to 0V
END;
```

---

## SetAnalogChannel

*Syntax*
```
PROCEDURE SetAnalogChannel(CardAddress: Longint; Channel: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the 8-bit DA channel number in which the data is to be set to maximum.

*Description*
The selected 8-bit Digital to Analogue Converter channel is set to maximum output voltage.

*Example*
```
BEGIN
  SetAnalogChannel(0, 1);
 // DA channel 1 of card address 0 is set to +5V
 // or +10V according to the DA jumper settings
END;
```

## SetAllAnalog

*Syntax*
```
PROCEDURE SetAllAnalog(CardAddress: Longint);
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Description*
All channels of the 8-bit Digital to Analogue Converters are set to maximum output voltage.

*Example*
```
  BEGIN
    SetAllAnalog(0);
    // DA channels 1 to 8 of card address 0 are set to +5V
    // or +10V according to the DA jumper settings
  END;
```

## OutputAllDigital

*Syntax*
```
PROCEDURE OutputAllDigital(CardAddress: Longint; Data: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Data:` Value between 0 and 255 that is sent to the digital output port (8 channels).

*Description*
The channels of the digital output port are updated with the status of the corresponding bits in the data parameter. A high (1) level means that the microcontroller IC1 output is set, and a low (0) level means that the output is cleared.

*Example*
```
  BEGIN
    OutputAllDigital(0, 7);
    // Output channels 1...3 of card address 0 are on and channels 4...8 are
off
  END;
```

## ClearDigitalChannel

*Syntax*
```
PROCEDURE ClearDigitalChannel(CardAddress: Longint; Channel: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the output channel that is to be cleared.

*Description*
The selected channel is cleared.

*Example*
```
  BEGIN
    ClearIOchannel(0, 4);
    // Digital output channel 4 of card address 0 is OFF
  END;
```

## ClearAllDigital

*Syntax*
```
PROCEDURE ClearAllDigital(CardAddress: Longint);
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Result*
All digital outputs are cleared.

*Example*
```
BEGIN
  ClearAllDigital(0);
  // All Output channels 1 to 8 of card address 0 are OFF
END;
```

## SetDigitalChannel

*Syntax*
```
PROCEDURE SetDigitalChannel(CardAddress: Longint; Channel: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the output channel that is to be set.

*Description*
The selected digital output channel is set ON.

*Example*
```
BEGIN
  SetDigitalChannel(0, 1);
  // Digital output channel 3 of card address 0 is ON
END;
```

## SetAllDigital

*Syntax*
```
PROCEDURE SetAllDigital(CardAddress: Longint);
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Description*
All the digital output channels are set ON.

*Example*
```
BEGIN
  SetAllDigital(0);
  // All Output channels of card address 0 are ON
END;
```

## ReadDigitalChannel

*Syntax*
```
FUNCTION ReadDigitalChannel(CardAddress: Longint; Channel: Longint):
Boolean;
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Channel:` Value between 1 and 8 which corresponds to the input channel whose status is to be read.

*Result*
`Boolean`: TRUE means that the input voltage of the corresponding digital input of the card is HIGH.
`Boolean`: FALSE means that the input is LOW.

*Description*
The status of the selected Input channel is read.

*Example*
```
  var status: boolean;
  BEGIN
    status := ReadIOchannel(0, 2);
    // Read Input channel 2 of card address 0
  END;
```

## ReadAllDigital

*Syntax*
```
FUNCTION ReadAllDigital(CardAddress: Longint): Longint;
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Result*
`Longint:` The 8 LSB correspond to the status of the digital input channels. '1' means that the corresponding input of the card is HIGH, '0' means that the input is LOW.

*Description*
The function returns the status of the digital inputs of the card.

*Example*
```
  var status: longint;
  BEGIN
    status := ReadAllDigital(0);
    // Read the Input channels of card address 0
  END;
```

## OutputPWM

*Syntax*
```
PROCEDURE OutputPWM(CardAddress: Longint; Data: Longint);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Data:` Value between 0 and 1023 which is to be sent to the PWM output of the card .

*Description*
The 10-bit PWM output is set according to the new data.

*Example*

```
BEGIN
  OutputPWM(0, 255);
  // The duty cycle of the PWM output of card address 0 is set to 25%
END;
```

## PowerGood

*Syntax*

```
FUNCTION PowerGood(CardAddress: Longint): Boolean;
```

*Parameter*

`CardAddress:` The address of previously opened card.

*Result*

`Boolean`: TRUE indicates that IC6 (PIC16F871) is powered and running and the data link between IC3 and IC6 is working properly.
`Boolean`: FALSE indicates that either power supply voltage is missing from the IC6 or there are problems with the data link between IC3 and IC6.

*Description*

Checks that IC6 and the data link between IC3 and IC6 is working properly.

*Example*

```
if PowerGood(CardAddress) then
begin
  Label19.Caption:='CPU OK';
  Label19.Font.Color:=ClGreen;
end
else
begin
  Label19.Caption:='CPU FAIL';
  Label19.Font.Color:=ClRed;
end;
```

## Connected

*Syntax*

```
FUNCTION Connected(CardAddress: Longint): Boolean;
```

*Parameter*

`CardAddress:` The address of previously opened card.

*Result*

`Boolean`: TRUE indicates that USB connection to the card is opened and working.
`Boolean`: FALSE indicates that USB connection to the card is not opened or the USB cable is disconnected.

*Description*

Checks that USB connection to the card is valid.

*Example*

```
if Connected(CardAddress) then
begin
  Label20.Caption:='USB Connected';
  Label20.Font.Color:=ClGreen;
```

```
end
  else
  begin
    Label20.Caption:='USB Disconnected';
    Label20.Font.Color:=ClRed;
    label12.caption:='Card disconnected';
  end;
```

## ReadVersion

*Syntax*
```
PROCEDURE ReadVersion(CardAddress: Longint; Buffer: Pointer);
```

*Parameters*
`CardAddress:` The address of previously opened card.
`Buffer:` Pointer to an array of long integers where the software (firmware) version info data will be read. The size of the buffer should be minimum 50 long integers (200 bytes).

*Description*
The software (firmware) version info of IC3 and IC6 is read from the internal EEPROM of the IC.

*Example*
```
var Buffer: array[0..50] of longint;
    i:integer;
begin
  ReadVersion(cardnumber,@Buffer[0]);
  label6.caption:='';
  for i:=0 to 49 do label6.caption:=label6.caption+chr(Buffer[i]);
end;
```

## ReadBackDigitalOut

*Syntax*
```
FUNCTION ReadBackDigitalOut(CardAddress: Longint): Longint;
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Result*
`Longint:` Value between 0 and 255 that is sent to the digital output port.

*Description*
The byte sent to the digital output port is read back.

*Example*
```
  var data: longint;
  BEGIN
      data := ReadBackDigitalOut(0);
      // digital output byte of card address 0 is read to variable 'data'
  END;
```

## ReadIBackAnalogOut

*Syntax*
```
PROCEDURE ReadIBackAnalogOut(CardAddress: Longint; Buffer: Pointer);
```

*Parameter*
`CardAddress:` The address of previously opened card.
`Buffer:` Pointer to array of long integers where the data will be read.

*Description*
The values of all eight Digital to Analogue converters are read back to an array of long integers.

*Example*
```
var Buffer: Array[0..7] of Longint;
begin
  ReadBackAnalogOut(0, @Buffer[0]);
  // Reads all the analog output data from the K8061 card address 0 to
array 'Buffer'
end;
```

## ReadBackPWMOut

*Syntax*
```
FUNCTION ReadBackPWMOut(CardAddress: Longint): Longint;
```

*Parameter*
`CardAddress:` The address of previously opened card.

*Result*
`Longint:` Value between 0 and 1023 that is sent to the PWM output of the card.

*Description*
The 10-bit PWM output data is read back.

*Example*
```
  var data: longint;
  BEGIN
      data := ReadBackPWMOut(0);
      // PWM output data of card address 0 is read to variable 'data'
  END;
```

## Using the K8061.DLL in Delphi

In this application example there are the declarations of the K8061.DLL procedures and functions and an example how to use the two most important DLL function calls: **`OpenDevice`** and **`CloseDevices.`**

```
unit K8061dmo;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, ComCtrls, Math, Buttons;

type
  TForm1 = class(TForm)
    GroupBox1: TGroupBox;
    A2: TCheckBox;
    A1: TCheckBox;
    A3: TCheckBox;
    Timer1: TTimer;
    Button3: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button3Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
function OpenDevice: Longint; stdcall; external 'K8061.dll';
procedure CloseDevices; stdcall; external 'K8061.dll';
function ReadAnalogChannel(CardAddress: Longint; Channel: Longint):Longint; stdcall; external
'K8061.dll';
procedure ReadAllAnalog(CardAddress: Longint; Buffer: Pointer); stdcall; external 'K8061.dll';
procedure OutputAnalogChannel(CardAddress: Longint; Channel: Longint; Data: Longint); stdcall;
external 'K8061.dll';
procedure OutputAllAnalog(CardAddress: Longint; Buffer: Pointer); stdcall; external
'K8061.dll';
procedure ClearAnalogChannel(CardAddress: Longint; Channel: Longint); stdcall; external
'K8061.dll';
procedure ClearAllAnalog(CardAddress: Longint) ; stdcall; external 'K8061.dll';
procedure SetAnalogChannel(CardAddress: Longint; Channel: Longint); stdcall; external
'K8061.dll';
procedure SetAllAnalog(CardAddress: Longint); stdcall; external 'K8061.dll';
procedure OutputAllDigital(CardAddress: Longint; Data: Longint);stdcall;  external
'K8061.dll';
procedure ClearDigitalChannel(CardAddress: Longint; Channel: Longint); stdcall; external
'K8061.dll';
procedure ClearAllDigital(CardAddress: Longint); stdcall; external 'K8061.dll';
procedure SetDigitalChannel(CardAddress: Longint; Channel: Longint); stdcall; external
'K8061.dll';
procedure SetAllDigital(CardAddress: Longint); stdcall; external 'K8061.dll';
function ReadDigitalChannel(CardAddress: Longint; Channel: Longint): Boolean; stdcall;
external 'K8061.dll';
function ReadAllDigital(CardAddress: Longint): Longint; stdcall; external 'K8061.dll';
procedure OutputPWM(CardAddress: Longint; Data: Longint); stdcall; external 'K8061.dll';
function PowerGood(CardAddress: Longint): Boolean; stdcall; external 'K8061.dll';
function Connected(CardAddress: Longint): Boolean; stdcall; external 'K8061.dll';
procedure ReadVersion(CardAddress: Longint; Buffer: Pointer); stdcall; external 'K8061.dll';
function ReadBackDigitalOut(CardAddress: Longint): Longint; stdcall; external 'K8061.dll';
procedure ReadBackAnalogOut(CardAddress: Longint; Buffer: Pointer ); stdcall; external
'K8061.dll';
```

```
function ReadBackPWMOut(CardAddress: Longint):Longint; stdcall; external 'K8061.dll';

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  CloseDevices;
end;

procedure TForm1.Button3Click(Sender: TObject);
var h:longint;
begin
  h:= OpenDevice;
  case h of
    0..7: begin
            label12.caption:='Card '+ inttostr(h)+' connected';
            RadioGroup3.items[h]:=inttostr(h);
            RadioGroup3.enabled:=true;
            RadioGroup3.itemindex:=h;
            timer1.enabled:=true;
            CardAddress:=h;
          end;
    -1: label12.caption:='All cards connected.';
    -2: label12.caption:='Card not found.';
  end;
end;
```

## Using the K8061.DLL in Visual Basic

In the listing of an application example there are the declarations of the K8061.DLL procedures and functions and an example how to use the two most important DLL function calls: **`OpenDevice`** and **`CloseDevices.`**

**Note:** Make sure that the file K8061.DLL is copied to the Windows' SYSTEM32 folder:

```
Option Explicit
Dim DoNothing As Boolean
Dim n As Integer
Dim CardAddress As Long

Private Declare Function OpenDevice Lib "k8061.dll" () As Long
Private Declare Sub CloseDevices Lib "k8061.dll" ()
Private Declare Function ReadAnalogChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long) As Long
Private Declare Function PowerGood Lib "k8061.dll" (ByVal CardAddress As Long) As Boolean
Private Declare Function Connected Lib "k8061.dll" (ByVal CardAddress As Long) As Boolean
Private Declare Sub ReadVersion Lib "k8061.dll" (ByVal CardAddress As Long, Buffer As Long)
Private Declare Sub ReadAllAnalog Lib "k8061.dll" (ByVal CardAddress As Long, Buffer As Long)
Private Declare Sub OutputAnalogChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long, ByVal Data As Long)
Private Declare Sub OutputAllAnalog Lib "k8061.dll" (ByVal CardAddress As Long, Buffer As
Long)
Private Declare Sub ClearAnalogChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long)
Private Declare Sub SetAllAnalog Lib "k8061.dll" (ByVal CardAddress As Long)
Private Declare Sub ClearAllAnalog Lib "k8061.dll" (ByVal CardAddress As Long)
Private Declare Sub SetAnalogChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal Channel
As Long)
Private Declare Sub OutputAllDigital Lib "k8061.dll" (ByVal CardAddress As Long, ByVal Data As
Long)
Private Declare Sub ClearDigitalChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long)
Private Declare Sub ClearAllDigital Lib "k8061.dll" (ByVal CardAddress As Long)
Private Declare Sub SetDigitalChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long)
Private Declare Sub SetAllDigital Lib "k8061.dll" (ByVal CardAddress As Long)
Private Declare Function ReadDigitalChannel Lib "k8061.dll" (ByVal CardAddress As Long, ByVal
Channel As Long) As Boolean
Private Declare Function ReadAllDigital Lib "k8061.dll" (ByVal CardAddress As Long) As Long
Private Declare Sub OutputPWM Lib "k8061.dll" (ByVal CardAddress As Long, ByVal Data As Long)
Private Declare Function ReadBackDigitalOut Lib "k8061.dll" (ByVal CardAddress As Long) As
Long
Private Declare Sub ReadBackAnalogOut Lib "k8061.dll" (ByVal CardAddress As Long, Buffer As
```

```
Long)
Private Declare Function ReadBackPWMOut Lib "k8061.dll" (ByVal CardAddress As Long) As Long


Private Sub Connect_Click()
    Dim h As Long
    h = OpenDevice
    Select Case h
        Case 0, 1, 2, 3, 4, 5, 6, 7
            Label1.Caption = "Card " + Str(h) + " connected"
            Option1(h).Enabled = True
            Option1(h).Value = True
            Timer1.Enabled = True
        Case -2
            Label1.Caption = "Card not found"
        Case -1
            Label1.Caption = "All cards opened"
    End Select
End Sub

Private Sub Form_Terminate()
    CloseDevices
End Sub
```

## Using the K8061.DLL in Borland C++Builder

Below there is a listing of the K8061.h including the declarations of the K8061.DLL procedures and functions. A listing of an application example shows how to use the two most important DLL function calls: **`OpenDevice`** and **`CloseDevices.`**

**Note:** Include K8061.LIB to the project. (Select the *Project* | *Add To Project* menu option and add the K8061.LIB file to the project.)

```
//Listing K8061.h
#ifdef __cplusplus
extern "C" {
#endif

#define FUNCTION __declspec(dllimport)

FUNCTION long __stdcall OpenDevice();
FUNCTION __stdcall CloseDevices();
FUNCTION long __stdcall ReadAnalogChannel(long CardAddress,long Channel);
FUNCTION __stdcall ReadAllAnalog(long CardAddress, long *Buffer);
FUNCTION __stdcall OutputAnalogChannel(long CardAddress, long Channel, long Data);
FUNCTION __stdcall OutputAllAnalog(long CardAddress, long *Buffer);
FUNCTION __stdcall ClearAnalogChannel(long CardAddress, long Channel);
FUNCTION __stdcall ClearAllAnalog(long CardAddress);
FUNCTION __stdcall SetAnalogChannel(long CardAddress,long Channel);
FUNCTION __stdcall SetAllAnalog(long CardAddress);
FUNCTION __stdcall OutputAllDigital(long CardAddress, long Data);
FUNCTION __stdcall ClearDigitalChannel(long CardAddress, long Channel);
FUNCTION __stdcall ClearAllDigital(long CardAddress);
FUNCTION __stdcall SetDigitalChannel(long CardAddress, long Channel);
FUNCTION __stdcall SetAllDigital(long CardAddress);
FUNCTION bool __stdcall ReadDigitalChannel(long CardAddress, long Channel);
FUNCTION long __stdcall ReadAllDigital(long CardAddress);
FUNCTION __stdcall OutputPWM(long CardAddress, long Data);
FUNCTION bool __stdcall PowerGood(long CardAddress);
FUNCTION bool __stdcall Connected(long CardAddress);
FUNCTION __stdcall ReadVersion(long CardAddress, long *Buffer);
FUNCTION long __stdcall ReadBackDigitalOut(long CardAddress);
FUNCTION __stdcall ReadBackAnalogOut(long CardAddress, long *Buffer);
FUNCTION long __stdcall ReadBackPWMOut(long CardAddress);

#ifdef __cplusplus
}
#endif
```

```
//Listing Unit1.cpp
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "K8061.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
  bool DisableOtherFunctionCall = false;
  long CardAddr;
  int n=8;
//---------------------------------------------------------------------------

__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{

}
//---------------------------------------------------------------------------

void __fastcall TForm1::Connect1Click(TObject *Sender)
{
  int h = OpenDevice();
  switch (h) {
    case  0:
    case  1:
    case  2:
    case  3:
    case  4:
    case  5:
    case  6:
    case  7:
      Label1->Caption = "Card " + IntToStr(h) + " connected";
      RadioGroup1->Items->Strings[h] = IntToStr(h);
      RadioGroup1->Enabled = true;
      RadioGroup1->ItemIndex = h;
      Timer1->Enabled = true;
      break;
    case  -1 :
      Label1->Caption = "All cards connected.";
      break;
    case -2 :
      Label1->Caption = "Card not found";
  }
}
//---------------------------------------------------------------------------

void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
{
  CloseDevices();
}
//---------------------------------------------------------------------------
```