Introduction to the DLL for the USB Experiment Interface Board K8055

The K8055 interface board has 5 digital input channels and 8 digital output channels. In addition, there are two analogue inputs, two analogue voltage outputs and two PWM (Pulse Width Modulation) outputs with 8 bit resolution. The number of inputs/outputs can be further expanded by connecting more (up to a maximum of four) cards to the PC's USB connectors. Each card is given its own identification number by means of two jumpers, SK5 and SK6 (see table 1 below for card numbering).

All communication routines are contained in a Dynamic Link Library (DLL) K8055D.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) 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 K8055D.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.

Note that all the examples in the function and procedure description section are written for Delphi.

<table>
<thead>
<tr>
<th>SK5</th>
<th>SK6</th>
<th>CARD ADDRESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ON</td>
<td>ON</td>
<td>0</td>
</tr>
<tr>
<td>OFF</td>
<td>ON</td>
<td>1</td>
</tr>
<tr>
<td>ON</td>
<td>OFF</td>
<td>2</td>
</tr>
<tr>
<td>OFF</td>
<td>OFF</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE 1: Jumper SK5, SK6 Settings**

**Note:** These settings must be done before the USB cable is connected to the K8055 card or before turning the PC on.
Overview of the Procedures and Functions of the K8055D.DLL

General procedures
- OpenDevice(CardAddress)
  Opens the communication link to the K8055 device
- CloseDevice
  Closes the link to the K8055 device

Analogue to Digital converter procedures
- ReadAnalogChannel(Channelno)
  Reads the status of one analogue input-channel
- ReadAllAnalog(Data1, Data2)
  Reads the status of both analogue input-channels

Digital to Analogue conversion procedures
- OutputAnalogChannel(Channel, Data)
  Sets the analogue output channel according to the data
- OutputAllAnalog(Data1, Data2)
  Sets both analogue output channels according to the data
- ClearAnalogChannel(Channel)
  Sets the analogue output channel to minimum
- ClearAllAnalog
  Sets all analogue output channels to minimum
- SetAnalogChannel(Channel)
  Sets the analogue output channel to maximum
- SetAllAnalog
  Sets all analogue output channels to maximum

Digital Output procedures
- WriteAllDigital(Data)
  Sets the digital outputs according to the data
- ClearDigitalChannel(Channel)
  Clears the output channel
- ClearAllDigital
  Clears all output channels
- SetDigitalChannel(Channel)
  Sets the output channel
- SetAllDigital
  Sets all output channels

Digital Input procedures and functions
- ReadDigitalChannel(Channel)
  Reads the status of the input channel
- ReadAllDigital(Buffer)
  Reads the status of all the input channels

Counter procedures and functions
- ResetCounter(CounterNr)
  Resets the 16 bit pulse counter number 1 or counter number 2
- ReadCounter(CounterNr)
  Reads the content of the pulse counter number 1 or counter number 2
- SetCounterDebounceTime(CounterNr, DebounceTime)
  Sets the debounce time to the pulse counter
Procedures And Functions of the K8055D.DLL

OpenDevice

Syntax

\[
\text{FUNCTION} \quad \text{OpenDevice(CardAddress: Longint): Longint;}
\]

Parameter

CardAddress: Value between 0 and 3 which corresponds to the jumper (SK5, SK6) setting on the K8055 board. See table 1.

Result

Longint: If succeeded the return value will be the card address read from the K8055 hardware. Return value -1 indicates that K8055 card was not found.

Description

Opens the communication link to the K8055 card. Loads the drivers needed to communicate via the USB port. This procedure must be performed before any attempts to communicate with the K8055 card.

This function can also be used to selects the active K8055 card to read and write the data. All the communication routines after this function call are addressed to this card until the other card is selected by this function call.

Example

\[
\text{var h: longint;}
\text{BEGIN}
\text{\quad h:=OpenDevice(0); // Opens the link to card number 0}
\text{\quad END;}
\]

CloseDevice

Syntax

\[
\text{PROCEDURE} \quad \text{CloseDevice;}
\]

Description

Unloads the communication routines for K8055 card and unloads the driver needed to communicate via the USB port. This is the last action of the application program before termination.

Example

\[
\text{BEGIN}
\text{\quad CloseDevice; // The communication to the K8055 device is closed}
\text{\quad END;}
\]

ReadAnalogChannel

Syntax

\[
\text{FUNCTION} \quad \text{ReadAnalogChannel (Channel: Longint): Longint;}
\]
Parameter
Channel: Value between 1 and 2 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 255.

Example

```pascal
var data: longint;
BEGIN
  data := ReadAnalogChannel(1); // AD channel 1 is read to variable 'data'
END;
```

ReadAllAnalog

Syntax
PROCEDURE ReadAllAnalog(var Data1, Data2: Longint);

Parameter
Data1, Data2: Pointers to the long integers where the data will be read.

Description
The status of both Analogue to Digital Converters are read to an array of long integers.

Example

```pascal
procedure TForm1.Button1Click(Sender: TObject);
var Data1, Data2: Longint;
begin
  ReadAllAnalog(Data1, Data2); // Read the data from the K8055
  Label1.caption:=inttostr(Data1); // Display CH1 data
  Label2.caption:=inttostr(Data2); // Display CH2 data
end;
```

OutputAnalogChannel

Syntax
PROCEDURE OutputAnalogChannel(Channel: Longint; Data: Longint);

Parameters

- Channel: Value between 1 and 2 which corresponds to the 8-bit DA 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). A value of 'Data' lying in between these extremes can be translated by the following formula: \( \frac{\text{Data}}{255} \times 5V. \)

**Example**

```
BEGIN
  OutputAnalogChannel (1,127);
  // DA channel 1 is set to 2.5V
END;
```

---

**OutputAllAnalog**

**Syntax**

PROCEDURE OutputAllAnalog(Data1: Longint; Data2: Longint);

**Parameters**

Data1, Data2: Value between 0 and 255 which is to be sent to the 8-bit Digital to Analogue Converter.

**Description**

Both 8-bit 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). A value of 'Data1' or 'Data2' lying in between these extremes can be translated by the following formula: \( \frac{\text{Data}}{255} \times 5V. \)

**Example**

```
BEGIN
  OutputAllAnalog(127, 255);
  // DA channel 1 is set to 2.5V and channel 2 is set to 5V
END;
```

---

**ClearAnalogChannel**

**Syntax**

PROCEDURE ClearAnalogChannel(Channel: Longint);

**Parameter**

Channel: Value between 1 and 2 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 (1); // DA channel 1 is set to 0V
END;
```

© 2003 ... Velleman
ClearAllAnalog

Syntax
PROCEDURE ClearAllAnalog;

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

Example
BEGIN
    ClearAllAnalog; // All DA channels 1 and 2 are set to 0V
END;

SetAnalogChannel

Syntax
PROCEDURE SetAnalogChannel(Channel: Longint);

Parameter
Channel: Value between 1 and 2 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(1); // DA channel 1 is set to +5V
END;

SetAllAnalog

Syntax
PROCEDURE SetAllAnalog;

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

Example
BEGIN
    SetAllAnalog; // DA channels 1 and 2 are set to +5V
END;

WriteAllDigital
Syntax
PROCEDURE WriteAllDigital(Data: Longint);

Parameter
Data: Value between 0 and 255 that is sent to the 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
    WriteAllDigital(7); // Output channels 1...3 are on, output channels 4...8 are off
END;

ClearDigitalChannel

Syntax
PROCEDURE ClearDigitalChannel(Channel: Longint);

Parameter
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(4); // Digital output channel 4 is OFF
END;

ClearAllDigital

Syntax
PROCEDURE ClearAllDigital;

Result
All digital outputs are cleared.

Example
BEGIN
    ClearAllDigital; // All Output channels 1 to 8 are OFF
END;

SetDigitalChannel
**SetDigitalChannel**

*Syntax*

PROCEDURE SetDigitalChannel(Channel: Longint);

*Parameter*

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.

*Example*

BEGIN
    SetDigitalChannel(1); // Digital output channel 3 is ON
END;

---

**SetAllDigital**

*Syntax*

PROCEDURE SetAllDigital;

*Description*

All the digital output channels are set.

*Example*

BEGIN
    SetAllDigital; // All Output channels are ON
END;

---

**ReadDigitalChannel**

*Syntax*

FUNCTION ReadDigitalChannel(Channel: Longint): Boolean;

*Parameter*

Channel: Value between 1 and 5 which corresponds to the input channel whose status is to be read.

*Result*

Boolean: TRUE means that the channel has been set and FALSE means that it has been cleared.

*Description*

The status of the selected Input channel is read.

*Example*

var status: boolean;
BEGIN
    status := ReadIOchannel(2); // Read Input channel 2
END;
ReadAllDigital

Syntax
FUNCTION ReadAllDigital: Longint;

Result
Longint: The 5 LSB correspond to the status of the input channels. A high (1) means that the channel is HIGH, a low (0) means that the channel is LOW.

Description
The function returns the status of the digital inputs.

Example
var status: longint;
BEGIN
  status := ReadAllDigital; // Read the Input channels
END;

ResetCounter

Syntax
PROCEDURE ResetCounter(CounterNumber: Longint);

Parameter
CounterNumber: Value 1 or 2, which corresponds to the counter to be reset.

Description
The selected pulse counter is reset.

Example
BEGIN
  ResetCounter(2); // Reset the counter number 2
END;

ReadCounter

Syntax
FUNCTION ReadCounter(CounterNumber: Longint): Longint;

Parameter
CounterNumber: Value 1 or 2, which corresponds to the counter to be read.

Result
Longint: The content of the 16 bit pulse counter.

Description
The function returns the status of the selected 16 bit pulse counter. The counter number 1 counts the pulses fed to the input I1 and the counter number 2 counts the pulses fed to the input I2.
Example
var pulses: longint;
BEGIN
  pulses := ReadCounter(2); // Read the counter number 2
END;

SetCounterDebounceTime

Syntax
PROCEDURE SetCounterDebounceTime(CounterNr, DebounceTime: Longint);

Parameter
CounterNumber: Value 1 or 2, which corresponds to the counter to be set.
DebounceTime: Debounce time for the pulse counter.
The DebounceTime value corresponds to the debounce time in milliseconds (ms) to be set for the pulse counter. Debounce time value may vary between 0 and 5000.

Description
The counter inputs are debounced in the software to prevent false triggering when mechanical switches or relay inputs are used. The debounce time is equal for both falling and rising edges. The default debounce time is 2ms. This means the counter input must be stable for at least 2ms before it is recognised, giving the maximum count rate of about 200 counts per second.

If the debounce time is set to 0, then the maximum counting rate is about 2000 counts per second.

Example
BEGIN
  SetCounterDebounceTime(1,100);
  // The debounce time for counter number 1 is set to 100ms
END;
Using the K8055D.DLL in Delphi

In this application example there are the declarations of the K8055D.DLL procedures and functions and an example how to use the two most important DLL function calls: OpenDevice and CloseDevice.

unit K8055;
interface
uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, ExtCtrls, ComCtrls;
type TForm1 = class(TForm)
  GroupBox1: TGroupBox;
  SK6: TCheckBox;
  SK5: TCheckBox;
  Button1: TButton;
  Label1: TLabel;
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Button1Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
var Form1: TForm1;
timed:boolean;
implementation
{$R *.DFM}
function OpenDevice(CardAddress: Longint): Longint; stdcall external 'K8055d.dll';
procedure CloseDevice; stdcall external 'K8055d.dll';
function ReadAnalogChannel(Channel: Longint):Longint; stdcall; external 'K8055d.dll';
procedure ReadAllAnalog(var Data1, Data2: Longint); stdcall; external 'K8055d.dll';
procedure OutputAnalogChannel(Channel: Longint; Data: Longint); stdcall; external 'K8055d.dll';
procedure OutputAllAnalog(Data1: Longint; Data2: Longint); stdcall; external 'K8055d.dll';
procedure ClearAnalogChannel(Channel: Longint); stdcall; external 'K8055d.dll';
procedure SetAnalogChannel(Channel: Longint); stdcall; external 'K8055d.dll';
procedure ClearAllAnalog; stdcall; external 'K8055d.dll';
procedure ReadDigitalChannel(Channel: Longint); stdcall; external 'K8055d.dll';
function ReadCounter(CounterNr: Longint): Longint; stdcall; external 'K8055d.dll';
procedure ResetCounter(CounterNr: Longint); stdcall; external 'K8055d.dll';
procedure SetCounterDebounceTime(CounterNr, DebounceTime:Longint); stdcall; external 'K8055d.dll';
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  CloseDevice;
end;
procedure TForm1.Button1Click(Sender: TObject);
var h,CardAddr:longint;
begin
  CardAddr:= 3-(integer(SK5.Checked) + integer(SK6.Checked) * 2);
  h:= OpenDevice(CardAddr);
  case h of
0..3: label12.caption:='Card ' + inttostr(h) + ' connected';
-1: label12.caption:='Card ' + inttostr(CardAddr) + ' not found';
end;
end;
end.
Using the K8055D.DLL in Visual Basic

In the listing of an application example there are the declarations of the K8055D.DLL procedures and functions and an example how to use the two most important DLL function calls: OpenDevice and CloseDevice.

Note: Make sure that the file K8055D.DLL is copied to the Windows' SYSTEM32 folder:

```vba
Option Explicit
Private Declare Function OpenDevice Lib "k8055d.dll" (ByVal CardAddress As Long) As Long
Private Declare Sub CloseDevice Lib "k8055d.dll" ()
Private Declare Function ReadAnalogChannel Lib "k8055d.dll" (ByVal Channel As Long) As Long
Private Declare Sub OutputAnalogChannel Lib "k8055d.dll" (ByVal Channel As Long, ByVal Data As Long)
Private Declare Sub OutputAllAnalog Lib "k8055d.dll" (ByVal Data1 As Long, ByVal Data2 As Long)
Private Declare Sub ClearAnalogChannel Lib "k8055d.dll" (ByVal Channel As Long)
Private Declare Sub SetAllAnalog Lib "k8055d.dll" ()
Private Declare Sub ClearAllAnalog Lib "k8055d.dll" ()
Private Declare Sub SetAnalogChannel Lib "k8055d.dll" (ByVal Channel As Long)
Private Declare Sub WriteAllDigital Lib "k8055d.dll" (ByVal Data As Long)
Private Declare Sub ClearDigitalChannel Lib "k8055d.dll" (ByVal Channel As Long)
Private Declare Sub ClearAllDigital Lib "k8055d.dll" ()
Private Declare Sub SetDigitalChannel Lib "k8055d.dll" (ByVal Channel As Long)
Private Declare Sub SetAllDigital Lib "k8055d.dll" ()
Private Declare Function ReadDigitalChannel Lib "k8055d.dll" (ByVal Channel As Long) As Boolean
Private Declare Function ReadAllDigital Lib "k8055d.dll" () As Long
Private Declare Function ReadCounter Lib "k8055d.dll" (ByVal CounterNr As Long) As Long
Private Declare Sub ResetCounter Lib "k8055d.dll" (ByVal CounterNr As Long)
Private Declare Sub SetCounterDebounceTime Lib "k8055d.dll" (ByVal CounterNr As Long, ByVal DebounceTime As Long)

Private Sub Connect_Click()
    Dim CardAddress As Long
    Dim h As Long
    CardAddress = 0
    CardAddress = 3 - (Check1(0).Value + Check1(1).Value * 2)
    h = OpenDevice(CardAddress)
    Select Case h
        Case 0, 1, 2, 3
            Label1.Caption = "Card " & Str(h) & " connected"
        Case -1
            Label1.Caption = "Card " & Str(CardAddress) & " not found"
    End Select
End Sub
```

```vba
Private Sub Form_Terminate()
    CloseDevice
End Sub
```
Using the K8055D.DLL in Borland C++ Builder

Below there is a listing of the K8055D.h including the declarations of the K8055D.DLL procedures and functions. A listing of an application example shows how to use the two most important DLL function calls: `OpenDevice` and `CloseDevice`.

```
//Listing K8055D.h
#ifndef __cplusplus
extern "C" {
#endif
#define FUNCTION __declspec(dllimport)
FUNCTION __stdcall long __stdcall OpenDevice(long CardAddress);
FUNCTION __stdcall CloseDevice();
FUNCTION __stdcall long __stdcall ReadAnalogChannel(long Channel);
FUNCTION __stdcall long __stdcall ReadAllAnalog(long *Data1, long *Data2);
FUNCTION __stdcall long __stdcall ReadAllDigital();
FUNCTION __stdcall long __stdcall ReadCounter(long CounterNr);
FUNCTION __stdcall long __stdcall ResetCounter(long CounterNr);
FUNCTION __stdcall bool __stdcall ReadDigitalChannel(long Channel);
FUNCTION __stdcall long __stdcall ReadAllDigital();
FUNCTION __stdcall long __stdcall ReadAllAnalog();
FUNCTION __stdcall long __stdcall WriteAllDigital(long Data);
FUNCTION __stdcall SetAllAnalog();
FUNCTION __stdcall ClearAllDigital();
FUNCTION __stdcall ClearAllAnalog();
FUNCTION __stdcall ClearAllAnalogChannel(long Channel);
FUNCTION __stdcall ClearAllDigitalChannel(long Channel);
FUNCTION __stdcall ClearAnalogChannel(long Channel);
FUNCTION __stdcall ClearDigitalChannel(long Channel);
FUNCTION __stdcall OutputAnalogChannel(long Channel, long Data);
FUNCTION __stdcall OutputAllAnalog(long Data, long Data2);
FUNCTION __stdcall SetAnalogChannel(long Channel);
FUNCTION __stdcall SetAllAnalog();
FUNCTION __stdcall WriteAllDigital(long Data);
FUNCTION __stdcall SetAllDigital();
FUNCTION __stdcall ReadDigitalChannel(long Channel);
FUNCTION __stdcall ReadAllDigital();
FUNCTION __stdcall ReadAllAnalog();
FUNCTION __stdcall ReadCounter(long CounterNr);
FUNCTION __stdcall ResetCounter(long CounterNr);
FUNCTION __stdcall ReadAllAnalog();
FUNCTION __stdcall WriteAllDigital(long Data);
FUNCTION __stdcall ClearAllDigital();
FUNCTION __stdcall ClearAllAnalog();
FUNCTION __declspec(dllexport) long __stdcall GetFunction();

#endif __cplusplus
}

//Listing Unit1.cpp
.INVISIBLE
//-------------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "K8055D.h"

#pragma package(smart_init)
#pragma resource "*.dfm"
 TForm1 *Form1;

fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}

fastcall TForm1::Connect1Click(TObject *Sender)
{
    int CardAddr = 3 - (int)(CheckBox1->Checked) + int(CheckBox2->Checked) * 2;
    int h = OpenDevice(CardAddr);
    switch (h) {
        case 0:
        case 1:
        case 2:
        case 3:
            Label1->Caption = "Card " + IntToStr(h) + " connected";
            break;
```
case -1 :
    Label1->Caption = "Card " + IntToStr(CardAddr) + " not found";
}
//------------------------------------------------------------------

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