Introduktion til DLL'en for K8055 USB Interface-kort

K8055 USB interface-kort har 5 digitale indgangskanaler og 8 digitale udgangskanaler. samtidigt er der to analoge indgange, to analoge spændingsudgange og to PWM (Pulsbredde modulerede) udgange med 8-bit opløsning.
Antallet af indgange og udgange kan udvides yderligere, ved at tilslutte flere (op til max. 4) kort til PC'ens USB-stik. Hvert kort tildeles sin egen adresse, ved hjælp af jumpers SK5 og SK6 (se tabel 1 herunder for kort-nummerering).

Alle kommunikations-rutiner er indeholdt i en "Dynamic Link Library fil" (DLL), som hedder: K8055D.DLL.

Dette dokument beskriver alle funktioner og procedurer i denne DLL, som kan anvendes i din egen programmering. Gennem kald af funktioner og procedurer indeholdt i DLL'en, kan du skrive dine egne programmer til Windows 98SE, 2000, Me samt XP i Delphi, Visual Basic, C++ Builder, eller enhvert andet 32-bit Windows udviklingsværktøj, som understøtter kald til en DLL.

En komplet oversigt over procedurer og funktioner i K8055D.DLL følger. I slutningen af dokumentet finder du kode til eksempel-programmer - så du har chancen for at sætte dig ind i, hvordan du laver dine egne programmer.
Programmerne er skrevet i Delphi, Visual Basic og C++ Builder. I source-kode findes fuld dokumentation for proce-durer og funktioner i DLL'en.

Bemærk venligst, at samtlige eksempler i beskrivelsen af funktioner og procedurer er skrevet i Delphi.

SK5	SK6	KORT ADDRESSE
ON	ON	0
OFF	ON	1
ON	OFF	2
OFF	OFF	3

TABEL 1: Jumper SK5, SK6 indstillinger

Note: Disse indstillinger skal udføres før USB-kablet tilsluttes til K8055, eller før computeren tændes.

Oversigt over Procedurer og Funktioner i K8055D.DLL

Generelle procedurer

OpenDevice(CardAddress)	*Åbner kommunikationen til K8055*
CloseDevice	*Lukker kommunikationen til K8055*

Analog til digital konverter procedurer

ReadAnalogChannel(Channelno)	*Læser status af én analog indgangskanal*
ReadAllAnalog(Data1, Data2)	*Læser status af begge, analoge indgangskanaler*

Digital til analog konverter procedurer

OutputAnalogChannel(Channel,Data)	*Indstiller den analoge udgangskanal i henhold til "data"*
OutputAllAnalog(Data1,Data2)	*Indstiller begge analoge udgangskanaler i henhold til "data"*
ClearAnalogChannel(Channel)	*Indstiller den analoge udgangskanal til minimum*
ClearAllAnalog	*Indstiller alle analoge udgangskanaler til minimum*
SetAnalogChannel(Channel)	*Indstiller den analoge udgangskanal til maximum*
SetAllAnalog	*Indstiller alle analoge udgangskanaler til maximum*

Digital udgang procedurer

WriteAllDigital(Data)	*Indstiller alle digitale udgangskanaler i henhold til "data"*
ClearDigitalChannel(Channel)	*Nulstiller udgangskanalen (0)*
ClearAllDigital	*Nulstiller alle udgangskanaler (0)*
SetDigitalChannel(Channel)	*Sætter udgangskanalen (1)*
SetAllDigital	*Sætter alle udgangskanaler (1)*

Digital indgang procedurer og funktioner

ReadDigitalChannel(Channel)	*Læser status af indgangskanalen*
ReadAllDigital(Buffer)	*Læser status af alle indgangskanaler*

Tæller procedurer og funktioner

ResetCounter(CounterNr)	*Nulstiller 16-bit pulstæller nummer 1 eller tæller nummer 2*
ReadCounter(CounterNr)	*Læser 16-bit pulstæller nummer 1 eller tæller nummer 2*
SetCounterDebounceTime(CounterNr, DebounceTime)	*Indstiller tilbagefalds forsinkelsen til pulstælleren*

Procedures And Functions of the K8055D.DLL

OpenDevice

Syntax

```
FUNCTION 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

```
  var h: longint;
  BEGIN
     h:=OpenDevice(0); // Opens the link to card number 0
  END;
```

CloseDevice

Syntax

```
PROCEDURE 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

```
  BEGIN
     CloseDevice; // The communication to the K8055 device is closed
  END;
```

ReadAnalogChannel

Syntax

```
FUNCTION 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

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

ReadIAllAnalog

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

```
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 : Data / 255 x 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 : Data / 255 x 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;
```

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 15
```
  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

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 ClearAllAnalog; stdcall; external 'K8055d.dll';
procedure SetAnalogChannel(Channel: Longint); stdcall; external 'K8055d.dll';
procedure SetAllAnalog; stdcall; external 'K8055d.dll';
procedure WriteAllDigital(Data: Longint);stdcall;  external 'K8055d.dll';
procedure ClearDigitalChannel(Channel: Longint); stdcall; external 'K8055d.dll';
procedure ClearAllDigital; stdcall; external 'K8055d.dll';
procedure SetDigitalChannel(Channel: Longint); stdcall; external 'K8055d.dll';
procedure SetAllDigital; stdcall; external 'K8055d.dll';
function ReadDigitalChannel(Channel: Longint): Boolean; stdcall; external 'K8055d.dll';
function ReadAllDigital: 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:

```
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 ReadAllAnalog Lib "k8055d.dll" (Data1 As Long, Data2 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


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
#ifdef __cplusplus
extern "C" {
#endif

#define FUNCTION __declspec(dllimport)

FUNCTION long __stdcall OpenDevice(long CardAddress);
FUNCTION __stdcall CloseDevice();
FUNCTION long __stdcall ReadAnalogChannel(long Channel);
FUNCTION __stdcall ReadAllAnalog(long *Data1, long *Data2);
FUNCTION __stdcall OutputAnalogChannel(long Channel, long Data);
FUNCTION __stdcall OutputAllAnalog(long Data1, long Data2);
FUNCTION __stdcall ClearAnalogChannel(long Channel);
FUNCTION __stdcall ClearAllAnalog();
FUNCTION __stdcall SetAnalogChannel(long Channel);
FUNCTION __stdcall SetAllAnalog();
FUNCTION __stdcall WriteAllDigital(long Data);
FUNCTION __stdcall ClearDigitalChannel(long Channel);
FUNCTION __stdcall ClearAllDigital();
FUNCTION __stdcall SetDigitalChannel(long Channel);
FUNCTION __stdcall SetAllDigital();
FUNCTION bool __stdcall ReadDigitalChannel(long Channel);
FUNCTION long __stdcall ReadAllDigital();
FUNCTION long __stdcall ReadCounter(long CounterNr);
FUNCTION __stdcall ResetCounter(long CounterNr);
FUNCTION __stdcall SetCounterDebounceTime(long CounterNr, long DebounceTime);

#ifdef __cplusplus
}
#endif


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

#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)
{

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

void __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;
}
//---------------------------------------------------------------------------
```