Introduction à la DLL pour la carte 'USB Experiment Interface Board K8055'

La carte interface K8055 dispose de 5 canaux d'entrée digitaux et de 8 canaux de sortie digitaux. Puis il y a encore e entrées analogues, 2 sorties de tension analogues et 2 sorties PWM (Pulse Width Modulation) à résolution 8 bits. Le nombre d'entrées et de sorties peut être agrandi en connectant plus de cartes (max. 4) sur les connecteurs USB de l'ordinateur. Chaque carte a un propre numéro d'identification qui est déterminé avec 2 cavaliers, SK5 et SK6 (voir tableau 1 ci-dessous pour la numérotation des cartes).

Toutes les routines de communication sont groupés dans une DLL (Dynamic Link Library - bibliothèque de liens dynamiques) K8055D.DLL.

Ce document décrit toutes les fonctions et procédures de la DLL qui sont accessibles par votre programme d'application. En appelant les fonctions et les procédures de la DLL, vous pouvez écrire vos propres applications en Windows (98SE, 2000, Me, XP), Delphi, Visual Basic, C++ Builder ou n'importe quel autre outil de développement d'application à 32 bits qui supporte des appels à une DLL.

Ci-dessous, vous trouverez un aperçu complet des procédures et fonctions exportées par la K8055D.DLL. A la fin de ce document vous trouverez des listings de programmes exemplaires pour vous donner une idée comment vous pouvez faire vos propres programmes d'application. Les exemples sont écrits en Delphi, Visual Basic et C++ Builder.
Les listings contiennent des explications complètes pour les fonctions et procédures DLL.

Notez que tous les exemples dans la partie des descriptions des fonctions et des procédures sont écrits pour Delphi.

SK5	SK6	ADRESSE DE LA CARTE
ON	ON	0
OFF	ON	1
ON	OFF	2
OFF	OFF	3

TABLEAU 1: Réglages des cavaliers SK5, SK6

Remarque: le réglage des cavaliers doit être fait avant que le câble USB soit connecté avec la carte K8055 ou avant que l'ordinateur soit allumé.

Aperçu des procédures et fonctions de la K8055D.DLL

Procédures générales

`OpenDevice(CardAddress)`	*Ouvre le lien de communication avec le K8055*
`CloseDevice`	*Ferme le lien avec le K8055*

Procédures de conversion analogue-numérique

`ReadAnalogChannel(Channelno)`	*Lit l'état d'un canal d'entrée analogue*
`ReadAllAnalog(Data1, Data2)`	*Lit l'état des deux canaux d'entrée analogue*

Procédures de conversion numérique analogue

`OutputAnalogChannel(Channel,Data)`	*Règle le canal de sortie analogue selon les données*
`OutputAllAnalog(Data1,Data2)`	*Règle les deux canaux de sortie analogues selon les données*
`ClearAnalogChannel(Channel)`	*Met le canal de sortie analogue sur le minimum*
`ClearAllAnalog`	*Met tous les canaux de sortie analogues sur le minimum*
`SetAnalogChannel(Channel)`	*Met le canal de sortie analogue sur le maximum*
`SetAllAnalog`	*Met tous les canaux de sortie analogues sur le maximum*

Procédures des sorties numériques

`WriteAllDigital(Data)`	*Règle les sorties numériques selon les données*
`ClearDigitalChannel(Channel)`	efface *le canal de sortie numérique*
`ClearAllDigital`	*efface tous les canaux de sortie numériques*
`SetDigitalChannel(Channel)`	*Règle le canal de sortie*
`SetAllDigital`	*Règle tous les canaux de sortie*

Fonctions et procédures des entrées numériques

`ReadDigitalChannel(Channel)`	*Lit l'état du canal d'entrée*
`ReadAllDigital(Buffer)`	*Lit l'état de tous les canaux d'entrée*

Procédures et fonctions des compteurs

`ResetCounter(CounterNr)`	*Réinitialise le compteur d'impulsions 16 bit numéro 1 ou 2*
`ReadCounter(CounterNr)`	*Lit le contenu du compteur d'impulsions numéro 1 ou 2*
`SetCounterDebounceTime(CounterNr, DebounceTime)`	*Règle le temps d'élimination du rebondissement selon le compteur d'impulsions*

Procédures et fonctions de la K8055D.DLL

OpenDevice

Syntaxe
```
FUNCTION OpenDevice(CardAddress: Longint): Longint;
```

Paramètre
CardAddress: Valeur entre 0 et 3 qui correspond au réglage du cavalier (SK5, SK6) sur le K8055. Voi tableau 1..

Résultat
Longint: Si réussi, la valeur de retour sera l'adresse de carte du matériel K8055. ne valeur de retour de -1 indique que la carte K8055 n'a pas été trouvée.

Description
Ouvre le lien de communication avec la carte K8055. Charge les programmes de gestion nécessaires pour pouvoir communiquer par la porte USB. Cette procédure doit être exécutée avant que vous puissiez essayer de communiquer avec la carte K8055.
Cette fonction peut également être utilisée pour écrire sur la carte K8055 active et la lire. Toutes les routines de communication qui viennent après cette fonction sont dirigées à cette carte jusqu'à ce qu'une autre carte soit sélectionnée avec cette fonction.

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

CloseDevice

Syntaxe
```
PROCEDURE CloseDevice;
```

Description
Charge les routines de communication pour la carte K8055 et le programme de gestion pour la communication par la porte USB. C'est la dernière action du programme d'application avant qu'il se ferme.

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

ReadAnalogChannel

Syntaxe
```
FUNCTION ReadAnalogChannel (Channel: Longint): Longint;
```

Paramètre
`Channel:` Valeur entre 1 et 2, correspond au canal AD dont l'état doit être lu.

Résultat
`Longint:` Les données correspondantes du convertisseur numérique-analogue sont lues.

Description
La tension d'entrée du canal de convertisseur analogue-numérique 8 bit sélectionné est convertie en une valeur entre 0 et 255.

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

ReadIAllAnalog

Syntaxe
```
PROCEDURE ReadAllAnalog(var Data1, Data2: Longint);
```

Paramètre
`Data1, Data2:` Réfère aux entiers longs où sont lus les données.

Description
L'état des deux convertisseurs analogue-numérique est lu dans une série d'entiers longs.

Exemple
```
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

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

Paramètres
`Channel:` Valeur entre 1 et 2 qui correspond au numéro de canal DA 8 bits dont les données doivent être déterminées.
`Data:` Valeur entre 0 en 255 qui doit être envoyé au convertisseur numérique-analogue 8 bit.

Description
Le canal de convertisseur numérique-analogue 8 bit indiqué est modifié selon les nouvelles données. Ceci signifie que les données correspondent à une tension spécifique. La valeur 0 correspond à la tension de sortie minimum (0 Volt) et la valeur 255 correspond à la tension de sortie maximum (+5V). Une valeur de donnée entre ces extrémités peut être traduit avec la formule: Donnée / 255 x 5V.

Exemple

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

OutputAllAnalog

Syntaxe

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

Paramètres

`Data1, Data2:` entre 0 et 255 qui doit être envoyé au convertisseur numérique-analogue 8 bit.

Description

Les deux canaux de convertisseur numérique-analogue 8 bit sont modifiés selon les nouvelles données. Ceci signifie que les données correspondent à une tension spécifique. La valeur 0 correspond à la tension de sortie minimum (0 Volt) et la valeur 255 correspond à la tension de sortie maximum (+5V). Une valeur de donnée entre ces extrémités peut être traduit avec la formule: Donnée / 255 x 5V.

Exemple

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

ClearAnalogChannel

Syntaxe

```
PROCEDURE ClearAnalogChannel(Channel: Longint);
```

Paramètre

`Channel:` Valeur entre 1 et 2 qui correspond au numéro du canal NA 8 bit dont les données doivent être effacées.

Description

Le canal NA sélectionné est mis sur la tension de sortie minimum (0 Volt).

Exemple

```
  BEGIN
    ClearAnalogChannel (1); // DA channel 1 is set to 0V
  END;
```

ClearAllAnalog

Syntaxe

```
PROCEDURE ClearAllAnalog;
```

Description
Les deux canaux NA sont mis à la tension de sortie minimum (0 Volt) .

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

SetAnalogChannel

Syntaxe
```
PROCEDURE SetAnalogChannel(Channel: Longint);
```

Paramètre
`Channel:` Valeur entre 1 et 2 qui correspond au numéro du canal NA 8 bit dont les données doivent être mises au maximum.

Description
Le canal de convertisseur NA 8 bit sélectionné est mis sur la tension de sortie maximum.

Exemple 15
```
  BEGIN
    SetAnalogChannel(1); // DA channel 1 is set to +5V
  END;
```

SetAllAnalog

Syntaxe
```
PROCEDURE SetAllAnalog;
```

Description
Tous les canaux de convertisseurs NA 8 bit sont mis sur la tension de sortie maximum.

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

WriteAllDigital

Syntaxe
```
PROCEDURE WriteAllDigital(Data: Longint);
```

Paramètre
`Data:` Valeur entre 0 et 255 qui est envoyé à la porte de sortie (8 canaux).

Description
Les canaux de la porte de sortie numérique sont mis à jour avec l'état des bits correspondants dans le paramètre des données. Un niveau haut (1) signifie que la sortie du microcontrôleur IC1 est réglée, et un niveau bas (0) signifie que la sortie est effacée.

Exemple
```
  BEGIN
    WriteAllDigital(7);
    // Output channels 1...3 are on, output channels 4...8 are off
  END;
```

ClearDigitalChannel

Syntaxe
```
PROCEDURE ClearDigitalChannel(Channel: Longint);
```

Paramètre
`Channel:` Valeur entre 1 et 8 qui correspond au canal de sortie à effacer.

Description
Le canal sélectionné est effacé.

Exemple
```
  BEGIN
    ClearIOchannel(4); // Digital output channel 4 is OFF
  END;
```

ClearAllDigital

Syntaxe
```
PROCEDURE ClearAllDigital;
```

Résultat
Toutes les sorties numériques sont effacées.

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

SetDigitalChannel

Syntaxe
```
PROCEDURE SetDigitalChannel(Channel: Longint);
```

Paramètre
`Channel:` Valeur entre 1 et 8 qui correspond au canal de sortie à régler

Description
Le canal de sortie numérique sélectionné est réglé.

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

SetAllDigital

Syntaxe
```
PROCEDURE SetAllDigital;
```

Description
Tous les canaux de sortie numériques sont réglés.

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

ReadDigitalChannel

Syntaxe
```
FUNCTION ReadDigitalChannel(Channel: Longint): Boolean;
```

Paramètre
`Channel:` Valeur entre 1 et 5 qui correspond au canal d'entrée à être lu.

Résultat
`Boolean`: TRUE signifie que le canal est réglé et FALSE signifie que le canal est effacé.

Description
L'état du canal d'entrée sélectionné est lu.

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

ReadAllDigital

Syntaxe
```
FUNCTION ReadAllDigital: Longint;
```

Résultat

`Longint:` Les 5 LSB correspondent à l'état des canaux d'entrée. Un signal haut (1) signifie que le canal est haut (HIGH), un signal bas (0) signifie que le canal est bas (LOW).

Description
La fonction retourne les états des entrées numériques.

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

ResetCounter

Syntaxe
```
PROCEDURE ResetCounter(CounterNumber: Longint);
```

Paramètre
`CounterNumber:` Valeur 1 ou 2, qui correspond au compteur à réinitialiser.

Description
Le compteur d'impulsions sélectionné est réinitialisé.

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

ReadCounter

Syntaxe
```
FUNCTION ReadCounter(CounterNumber: Longint): Longint;
```

Paramètre
`CounterNumber:` Valeur 1 ou 2, qui correspond au compteur à être lu

Résultat
`Longint:` Le contenu de compteur d'impulsions 16 bit.

Description
La fonction retourne l'état du compteur d'impulsions 16 bit sélectionné.
Compteur numéro 1 compte les impulsions entrantes par l'entrée I1 et compteur numéro 2 compte les impulsions entrantes par l'entrée I2.

Exemple
```
var pulses: longint;
BEGIN
  pulses := ReadCounter(2); // Read the counter number 2
END;
```

SetCounterDebounceTime

Syntaxe
```
PROCEDURE SetCounterDebounceTime(CounterNr, DebounceTime: Longint);
```

Paramètre
`CounterNumber:` Valeur 1 ou 2, qui correspond au compteur à régler.
`DebounceTime:` temps d'élimination du rebondissement -pour le compteur d'impulsions.
La valeur `DebounceTime` correspond au temps d'élimination du rebondissement en millisecondes (ms) à être réglé pour le compteur d'impulsions. Le temps d'élimination du rebondissement peut varier de 0 à 5000.

Description
Les signaux d'entrée du compteur sont soumis à une élimination du rebondissement dans le logiciel afin d'éviter des fausses activations quand les entrées sont mécaniques ou à relais. Le temps d'élimination du rebondissement est le même pour des faces descendantes que pour des faces montantes. Le temps d'élimination du rebondissement standard est de 2ms. Ceci signifie que l'entrée du compteur doit rester stabile pendant au moins 2ms pour être reconnu, ce qui limite le nombre d'impulsions par seconde à environ 200.
Si le temps d'élimination du rebondissement est mis à 0, un maximum de 2000 impulsions par seconde est possible.

Exemple
```
  BEGIN
    SetCounterDebounceTime(1,100);
    // The debounce time for counter number 1 is set to 100ms
  END;
```

Utiliser le K8055D.DLL en Delphi

Dans cet exemple d'application se trouvent les explications des procédures et fonctions K8055D.DLL, et un exemple démontrant l'utilisation des deux fonctions les plus importantes, **`OpenDevice`** et **`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.
```

Utiliser le K8055D.DLL en Visual Basic

Dans cet exemple d'application se trouvent les explications des procédures et fonctions K8055D.DLL, et un exemple démontrant l'utilisation des deux fonctions les plus importantes, **`OpenDevice`** et **`CloseDevice`**.

Remarque: Assurez-vous du fait que le fichier K8055D.DLL soit copié au fichier 'SYSTEM32' dans Windows:

```
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
```

Utiliser le K8055D.DLL en Borland C++ Builder

Ci-dessous se trouve un listing du K8055D.h qui contient les explications des procédures et fonctions K8055D.DLL. Un listing d'un exemple d'application démontre l'utilisation des deux fonctions les plus importantes, **`OpenDevice`** et **`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;
}
//---------------------------------------------------------------------------
```