K8055D.DLL v5.0.0.0

Technical Guide

Introduction

General

The K8055N interface board has 5 digital input channels and 8 digital output channels. In addition, there are two Analog inputs, two Analog 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 the Dynamic Link Library K8055D.DLL.

In this manual we will describe each of these functions provided by the DLL in detail. Calling the functions exported by the DLL, you can write custom Windows applications in Delphi, Visual Basic 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.
Note that all the examples in the function description section are written in C++.
K8055 examples folder includes examples written in Visual Basic 2008 Express, Visual C# 2008 Express, Visual C++ 2008 Express, VB6.0, MS Excel VBA, Delphi 5, Borland C++Builder 6 and Dev-C++.

Readers should have an understanding of the basic data types as well as basic knowledge of the Microsoft Windows operating system.

Microsoft Visual Studio users please note: The K8055D.DLL is a standard Windows DLL, you cannot reference it.

Calling convention

A calling convention is a scheme for how functions receive parameters from their caller and how they return a result. Different programming languages use different calling conventions, so it is important to know which calling convention is used by your programming language and which calling convention is used by the K8055D.DLL.

The most common calling convention is the *stdcall* calling convention, and this is also the one we have used for our DLL.

If you are using .NET (VB.NET or C#) you do not need to worry about this since the calling convention in .NET is also *stdcall*. However if you are using C to import the functions provided by the DLL, you will need to pay special attention to this.

Card Address Setting

SK5	SK6	CARD ADDRESS
ON	ON	0
OFF	ON	1
ON	OFF	2
OFF	OFF	3

TABLE 1: Jumper SK5, SK6 Settings

The card address settings must be done before the USB cable is connected to the K8055N card or before turning the PC on.

Overview of the Functions

General functions

`int OpenDevice(int CardAddress);`	*Opens the communication link to the K8055N device*
`void CloseDevice();`	*Closes the link to the K8055N device*
`int SearchDevices();`	*Gives information about the number of connected devices on the computer*
`int SetCurrentDevice(int lngCardAddress);`	*Set the current controlled device*
`int Version();`	*Gives information about the DLL software version number*

Analog to Digital converter functions

`int ReadAnalogChannel(int Channel);`	*Reads the status of one analog input-channel*
`void ReadAllAnalog(int *Data1, int *Data2);`	*Reads the status of both analog input-channels*

Digital to Analog conversion functions

`void OutputAnalogChannel(int Channel, int Data);`	*Sets the analog output channel according to the data*
`void OutputAllAnalog(int Data1, int Data2);`	*Sets both analog output channels according to the data*
`void ClearAnalogChannel(int Channel);`	*Sets the analog output channel to minimum*
`void ClearAllAnalog();`	*Sets all analog output channels to minimum*
`void SetAnalogChannel(int Channel);`	*Sets the analog output channel to maximum*
`void SetAllAnalog();`	*Sets all analog output channels to maximum*

Digital Output functions

`void WriteAllDigital(int Data);`	*Sets the digital outputs according to the data*
`void ClearDigitalChannel(int Channel);`	*Clears the digital output channel*
`void ClearAllDigital();`	*Clears all digital output channels*
`void SetDigitalChannel(int Channel);`	*Sets the digital output channel*
`void SetAllDigital();`	*Sets all digital output channels*

Digital Input functions

`bool ReadDigitalChannel(int Channel);`	*Reads the status of the input channel*
`int ReadAllDigital();`	*Reads the status of all the input channels*

Counter functions

`void ResetCounter(int CounterNr);`	*Resets the 16 bit pulse counter number 1 or counter number 2*
`int ReadCounter(int CounterNr);`	*Reads the content of the pulse counter number 1 or counter number 2*
`void SetCounterDebounceTime(int CounterNr, int DebounceTime);`	*Sets the debounce time to the pulse counter*

Readback procedures and functions

`int ReadBackDigitalOut();`	*Reads back the digital output value of the card*
`void ReadBackAnalogOut(int *Buffer);`	*Reads back the current D/A output values of the card*

Function List

OpenDevice

Syntax

```
int OpenDevice(int CardAddress);
```

Parameter

`CardAddress:` Value between 0 and 3 which corresponds to the jumper (SK5, SK6) setting on the K8055N board. See table 1.

Result

`Int:` If succeeded the return value will be the card address read from the K8055N hardware.
Return value -1 indicates that K8055N card was not found.

Description

Opens the communication link to the K8055N card. Loads the drivers needed to communicate via the USB port. This procedure must be performed before any attempts to communicate with the K8055N card.
This function can also be used to selects the active K8055N 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

```
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->Text = "Card " + h.ToString() + " connected";
        Timer1->Enabled = true;
    break;
    case  -1 :
        Label1->Text = "Card " + CardAddr.ToString() + " not found";
    break;
}
```

SearchDevices

Syntax

```
int SearchDevices();
```

Description

Using this function all the K8055N cards can be opened. No need to use OpenDevice.
This function returns all the connected K8055N devices on the computer. The returned value is a bit field.
Returned value

- Bin 0000, Dec 0 : No devices was found
- Bin 0001, Dec 1 : Card address 0 was found.
- Bin 0010, Dec 2 : Card address 1 was found.
- Bin 0100, Dec 4 : Card address 2 was found.
- Bin 1000, Dec 8 : Card address 3 was found.

Example : return value 9 = devices with address 0 and 3 are connected.

Example

```
k = SearchDevices();
if (k)
   Timer1->Enabled = true;
if (k & 1)
```

```
        {
           RadioButton9->Enabled = true;
        }
        if (k & 2)
        {
           RadioButton10->Enabled = true;
        }
        if (k & 4)
        {
           RadioButton11->Enabled = true;
        }
        if (k & 8)
        {
           RadioButton12->Enabled = true;
        }
```

SetCurrentDevice

Syntax

```
int SetCurrentDevice(int lngCardAddress);
```

Description

The function set the current controlled device. The returned value is the device address, if this value is –1 no device with the address parameter was found.

Parameter

Address: Value 0 to 3, which corresponds to the device address.

Example

```
SetCurrentDevice(0);
```

CloseDevice

Syntax

```
void CloseDevice();
```

Description

Unloads the communication routines for K8055N cards and unloads the driver needed to communicate via the USB port. This is the last action of the application program before termination.

Example

```
private: System::Void Form1_FormClosed(System::Object^  sender,
System::Windows::Forms::FormClosedEventArgs^  e)
       {
          CloseDevice();
       }
```

ReadAnalogChannel

Syntax

```
int ReadAnalogChannel(int Channel);
```

Parameter

`Channel:` Value between 1 and 2 which corresponds to the AD channel whose status is to be read.

Result

`Int:` The corresponding Analog to Digital Converter data is read.

Description

The input voltage of the selected 8-bit Analog to Digital converter channel is converted to a value which lies between 0 and 255.

Example
```
Label1->Text = ReadAnalogChannel(1).ToString();
```

ReadIAllAnalog

Syntax
```
void ReadAllAnalog(int *Data1, int *Data2);
```

Parameter
`Data1, Data2:` Pointers to the integers (32-bit) where the data will be read.

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

Example
```
            int Data1;
            int Data2;
            ReadAllAnalog(&Data1, &Data2);
```

OutputAnalogChannel

Syntax
```
void OutputAnalogChannel(int Channel, int Data);
```

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 Analog Converter .

Description
The indicated 8-bit Digital to Analog 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
```
OutputAnalogChannel(1, 255 - VScrollBar1->Value);
```

OutputAllAnalog

Syntax
```
void OutputAllAnalog(int Data1, int Data2);
```

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

Description
Both 8-bit Digital to Analog 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 * 5V.

Example
```
OutputAllAnalog(50, 255);
```

ClearAnalogChannel

Syntax
```
void ClearAnalogChannel(int Channel);
```

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
```
ClearAnalogChannel(1);
```

ClearAllAnalog

Syntax
```
void ClearAllAnalog();
```

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

Example
```
ClearAllAnalog();
```

SetAnalogChannel

Syntax
```
void SetAnalogChannel(int Channel);
```

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 Analog Converter channel is set to maximum output voltage.

Example
```
SetAnalogChannel(1);
```

SetAllAnalog

Syntax
```
void SetAllAnalog();
```

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

Example
```
SetAllAnalog();
```

WriteAllDigital

Syntax
```
void WriteAllDigital(int Data);
```

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 IC3 output is set, and a low (0) level means that the output is cleared.

Example
```
WriteAllDigital(0x55);
```

ClearDigitalChannel

Syntax
```
void ClearDigitalChannel(int Channel);
```

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
```
CheckBox9->Checked ? SetDigitalChannel(1): ClearDigitalChannel(1);
```

ClearAllDigital

Syntax
```
void ClearAllDigital();
```

Result
All digital outputs are cleared.

Example
```
ClearAllDigital();
```

SetDigitalChannel

Syntax
```
void SetDigitalChannel(int Channel);
```

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
```
CheckBox9->Checked ? SetDigitalChannel(1): ClearDigitalChannel(1);
```

SetAllDigital

Syntax
```
void SetAllDigital();
```

Description
All the digital output channels are set.

Example
```
SetAllDigital();
```

ReadDigitalChannel

Syntax

```
bool ReadDigitalChannel(int Channel);
```

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

Result
`bool`: 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
```
CheckBox4->Checked = ReadDigitalChannel(1);
```

ReadAllDigital

Syntax
```
int ReadAllDigital();
```

Result
`int:` The 5 LSB correspond to the status of the digital 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
```
i = ReadAllDigital();
CheckBox4->Checked = (i & 1)>0;
CheckBox5->Checked = (i & 2)>0;
CheckBox6->Checked = (i & 4)>0;
CheckBox7->Checked = (i & 8)>0;
CheckBox8->Checked = (i & 16)>0;
```

ResetCounter

Syntax
```
void ResetCounter(int CounterNr);
```

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

Description
The selected pulse counter is reset.

Example
```
ResetCounter(1);
```

ReadCounter

Syntax
```
int ReadCounter(int CounterNr);
```

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

Result
`int:` 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

```
TextBox1->Text = ReadCounter(1).ToString();
```

SetCounterDebounceTime

Syntax

```
void SetCounterDebounceTime(int CounterNr, int DebounceTime);
```

Parameter

`CounterNr:` 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

```
SetCounterDebounceTime(1, 10);
```

Version

Syntax

```
int Version();
```

Result

`int:` A 32 bit integer where the DLL version (4 digits) is represented. Each byte is one digit.

Description

The DLL version info is read.

Example

```
int ver = Version();
Label9->Text = (ver >> 24).ToString()+"."+((ver >> 16) & 0xFF).ToString()+"."
   +((ver >> 8) & 0xFF).ToString()+"."+(ver & 0xFF).ToString();
```

ReadBackDigitalOut

Syntax

```
int ReadBackDigitalOut();
```

Result

`int:` 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

```
int DigitalOut;
DigitalOut = ReadBackDigitalOut();
ReadBackAnalogOut(AnalogOut);

CheckBox9->Checked = (DigitalOut & 1)>0;
CheckBox10->Checked = (DigitalOut & 2)>0;
CheckBox11->Checked = (DigitalOut & 4)>0;
CheckBox12->Checked = (DigitalOut & 8)>0;
```

```
        CheckBox13->Checked = (DigitalOut & 16)>0;
        CheckBox14->Checked = (DigitalOut & 32)>0;
        CheckBox15->Checked = (DigitalOut & 64)>0;
        CheckBox16->Checked = (DigitalOut & 128)>0;
```

ReadIBackAnalogOut

Syntax

```
void ReadBackAnalogOut(int *Buffer);
```

Parameter

`Buffer:` Pointer to an array of 32-bit integers where the data will be read.

Description

The values of all two Digital-to-Analogue converters are read back to an array of 32-bit integers.

Example

```
        ReadBackAnalogOut(AnalogOut);

        OutputAnalogChannel(1, 255 - VScrollBar1->Value);
        OutputAnalogChannel(2, 255 - VScrollBar2->Value);
        Label4->Text = (255 - VScrollBar1->Value).ToString();
        Label5->Text = (255 - VScrollBar2->Value).ToString();
```

Function declarations in other programming languages

Visual Basic 6.0

```
    Private Declare Sub ReadAll Lib "k8055d.dll" (ByVal data As Long)
    Private Declare Function Version Lib "k8055d.dll" () As Long
    Private Declare Function SearchDevices Lib "k8055d.dll" () As Long
    Private Declare Function SetCurrentDevice Lib "k8055d.dll" (ByVal CardAddress As Long) As Long
    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" (ByVal Data1 As Long, ByVal 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 Declare Function ReadBackDigitalOut Lib "k8055d.dll" () As Long
    Private Declare Sub ReadBackAnalogOut Lib "k8055d.dll" (ByRef Buffer As Long)
```

Visual Basic 2008 Express

```
    Private Declare Function OpenDevice Lib "k8055d.dll" (ByVal CardAddress As Integer) As Integer
    Private Declare Sub CloseDevice Lib "k8055d.dll"
    Private Declare Function Version Lib "k8055d.dll" () As Integer
    Private Declare Function SearchDevices Lib "k8055d.dll" () As Integer
    Private Declare Function SetCurrentDevice Lib "k8055d.dll" (ByVal CardAddress As Integer) As Integer
    Private Declare Function ReadAnalogChannel Lib "k8055d.dll" (ByVal Channel As Integer) As Integer
```

```
    Private Declare Sub ReadAllAnalog Lib "k8055d.dll" (ByRef Data1 As Integer, ByRef Data2 As Integer)
    Private Declare Sub OutputAnalogChannel Lib "k8055d.dll" (ByVal Channel As Integer, ByVal Data As
Integer)
    Private Declare Sub OutputAllAnalog Lib "k8055d.dll" (ByVal Data1 As Integer, ByVal Data2 As Integer)
    Private Declare Sub ClearAnalogChannel Lib "k8055d.dll" (ByVal Channel As Integer)
    Private Declare Sub SetAllAnalog Lib "k8055d.dll" ()
    Private Declare Sub ClearAllAnalog Lib "k8055d.dll" ()
    Private Declare Sub SetAnalogChannel Lib "k8055d.dll" (ByVal Channel As Integer)
    Private Declare Sub WriteAllDigital Lib "k8055d.dll" (ByVal Data As Integer)
    Private Declare Sub ClearDigitalChannel Lib "k8055d.dll" (ByVal Channel As Integer)
    Private Declare Sub ClearAllDigital Lib "k8055d.dll" ()
    Private Declare Sub SetDigitalChannel Lib "k8055d.dll" (ByVal Channel As Integer)
    Private Declare Sub SetAllDigital Lib "k8055d.dll" ()
    Private Declare Function ReadDigitalChannel Lib "k8055d.dll" (ByVal Channel As Integer) As Boolean
    Private Declare Function ReadAllDigital Lib "k8055d.dll" () As Integer
    Private Declare Function ReadCounter Lib "k8055d.dll" (ByVal CounterNr As Integer) As Integer
    Private Declare Sub ResetCounter Lib "k8055d.dll" (ByVal CounterNr As Integer)
    Private Declare Sub SetCounterDebounceTime Lib "k8055d.dll" (ByVal CounterNr As Integer, ByVal
DebounceTime As Integer)
    Private Declare Function ReadBackDigitalOut Lib "k8055d.dll" () As Integer
    Private Declare Sub ReadBackAnalogOut Lib "k8055d.dll" (ByRef Buffer As Integer)
```

Visual C# 2008 Express

```
        [DllImport("k8055d.dll")]
        public static extern int OpenDevice(int CardAddress);

        [DllImport("k8055d.dll")]
        public static extern void CloseDevice();

        [DllImport("k8055d.dll")]
        public static extern int ReadAnalogChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern void ReadAllAnalog(ref int Data1, ref int Data2);

        [DllImport("k8055d.dll")]
        public static extern void OutputAnalogChannel(int Channel, int Data);

        [DllImport("k8055d.dll")]
        public static extern void OutputAllAnalog(int Data1, int Data2);

        [DllImport("k8055d.dll")]
        public static extern void ClearAnalogChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern void SetAllAnalog();

        [DllImport("k8055d.dll")]
        public static extern void ClearAllAnalog();

        [DllImport("k8055d.dll")]
        public static extern void SetAnalogChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern void WriteAllDigital(int Data);

        [DllImport("k8055d.dll")]
        public static extern void ClearDigitalChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern void ClearAllDigital();

        [DllImport("k8055d.dll")]
        public static extern void SetDigitalChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern void SetAllDigital();

        [DllImport("k8055d.dll")]
        public static extern bool ReadDigitalChannel(int Channel);

        [DllImport("k8055d.dll")]
        public static extern int ReadAllDigital();

        [DllImport("k8055d.dll")]
        public static extern int ReadCounter(int CounterNr);

        [DllImport("k8055d.dll")]
        public static extern void ResetCounter(int CounterNr);
```

```
[DllImport("k8055d.dll")]
public static extern void SetCounterDebounceTime(int CounterNr, int DebounceTime);

[DllImport("k8055d.dll")]
public static extern int Version();

[DllImport("k8055d.dll")]
public static extern int SearchDevices();

[DllImport("k8055d.dll")]
public static extern int SetCurrentDevice(int lngCardAddress);

[DllImport("k8055d.dll")]
public static extern int ReadBackDigitalOut();

[DllImport("k8055d.dll")]
public static extern void ReadBackAnalogOut(int[] Buffer);
```

Delphi

```
function SetCurrentDevice(CardAddress: integer): integer; stdcall; external 'K8055d.dll';
function OpenDevice(CardAddress: integer): integer; stdcall; external 'K8055d.dll';
function SearchDevices: integer; stdcall; external 'K8055d.dll';
function Version: integer; stdcall; external 'K8055d.dll';
procedure CloseDevice; stdcall; external 'K8055d.dll';
function ReadAnalogChannel(Channel: integer):integer; stdcall; external 'K8055d.dll';
procedure ReadAllAnalog(var Data1, Data2: integer); stdcall; external 'K8055d.dll';
procedure OutputAnalogChannel(Channel: integer; Data: integer); stdcall; external 'K8055d.dll';
procedure OutputAllAnalog(Data1: integer; Data2: integer); stdcall; external 'K8055d.dll';
procedure ClearAnalogChannel(Channel: integer); stdcall; external 'K8055d.dll';
procedure ClearAllAnalog; stdcall; external 'K8055d.dll';
procedure SetAnalogChannel(Channel: integer); stdcall; external 'K8055d.dll';
procedure SetAllAnalog; stdcall; external 'K8055d.dll';
procedure WriteAllDigital(Data: integer);stdcall;  external 'K8055d.dll';
procedure ClearDigitalChannel(Channel: integer); stdcall; external 'K8055d.dll';
procedure ClearAllDigital; stdcall; external 'K8055d.dll';
procedure SetDigitalChannel(Channel: integer); stdcall; external 'K8055d.dll';
procedure SetAllDigital; stdcall; external 'K8055d.dll';
function ReadDigitalChannel(Channel: integer): Boolean; stdcall; external 'K8055d.dll';
function ReadAllDigital: integer; stdcall; external 'K8055d.dll';
function ReadCounter(CounterNr: integer): integer; stdcall; external 'K8055d.dll';
procedure ResetCounter(CounterNr: integer); stdcall; external 'K8055d.dll';
procedure SetCounterDebounceTime(CounterNr, DebounceTime:integer); stdcall; external 'K8055d.dll';
procedure SetPWM(Channel: integer; Data: integer; Frequency: integer); stdcall; external 'K8055d.dll';
function ReadBackDigitalOut:Longint; stdcall; external 'K8055d.dll';
procedure ReadBackAnalogOut(Buffer: Pointer); stdcall; external 'K8055d.dll';
```

Borland C++Builder

```
#ifdef __cplusplus
extern "C" {
#endif

#define FUNCTION __declspec(dllimport)

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

```
FUNCTION int __stdcall SetCurrentDevice(int CardAddress);
FUNCTION int __stdcall  ReadBackDigitalOut();
FUNCTION void __stdcall  ReadBackAnalogOut(int *Buffer);

#ifdef __cplusplus
}
#endif
```