VM167.DLL

Technical Guide

VM167.DLL

Technical guide

Table of Contents

(I) Introduction 3

(II) Overview of the Functions 5

(III) Function List 7

(1) OpenDevices ... 8
(2) CloseDevices ... 9
(3) ReadAnalogChannel ... 10
(4) ReadIAllAnalog ... 11
(5) InOutMode ... 12
(6) OutputAllDigital ... 13
(7) ClearDigitalChannel ... 14
(8) ClearAllDigital ... 15
(9) SetDigitalChannel ... 16
(10) SetAllDigital ... 17
(11) ReadDigitalChannel ... 18
(12) ReadAllDigital ... 19
(13) SetPWM ... 20
(14) OutputAllPWM ... 21
(15) ResetCounter ... 22
(16) ReadCounter ... 23
(17) Connected ... 24
(18) VersionFirmware ... 25
(19) VersionDLL ... 26
(20) ReadBackPWMOut ... 27
(21) ReadBackInOutMode ... 28

(IV) Function declarations in other programming languages 30

(1) Visual Basic 6.0 ... 31
(2) Visual Basic 2008 Express ... 32
(3) Visual C# 2008 Express ... 33
(4) Delphi ... 34
(5) Borland C++Builder ... 35

Introduction

General

The VM167 interface board has 8 digital input/output channels, five analog input channels and two PWM outputs. The number of inputs/outputs can be expanded by connecting two VM167 cards to the PC. Both of the cards has its own identification number (card address) by means of the jumper setting.

All communication routines are contained in a Dynamic Link Library VM167.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 VM167.DLL follows. Note that all the examples in the function description section are written in C++.
VM167 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 VM167.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 VM167 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

J5	Card Address
ON	0
OFF	1

TABLE 1: Jumper J5 Settings

The card address settings must be done before the USB cable is connected to the VM167 card or before turning the PC on.
If the USB cable is disconnected and reconnected then all the digital terminals are set as inputs and the PWM outputs are reset to zero.

Overview of the Functions

General functions

```
int OpenDevices()
```
Opens the communication link to the VM167 devices

```
void CloseDevices()
```
Closes the link to the VM167 devices

```
int Connected()
```
Checks that the USB connection to the cards is valid

```
int VersionFirmware(int CardAddress)
```
Reads the firmware version number

```
int VersionDLL()
```
Reads the DLL version number

```
void InOutMode(int CardAddress, int HighNibble, int LowNibble)
```
Set the digital terminals either inputs or outputs

Analog to Digital converter functions

```
int ReadAnalogChannel(int CardAddress, int Channel)
```
Reads the status of one analog input-channel

```
void ReadAllAnalog(int CardAddress, int *Buffer)
```
Reads the status of all analog input-channels

PWM Output functions

```
void SetPWM(int CardAddress, int Channel, int Data, int Freq)
```
Sets the status of one PWM output

```
void OutputAllPWM(int CardAddress, int Data1, int Data2)
```
Sets both of the PWM outputs

Digital Output functions

```
void OutputAllDigital(int CardAddress, int Data)
```
Sets the digital outputs according to the data

```
void ClearDigitalChannel(int CardAddress, int Channel)
```
Clears the output channel

```
void ClearAllDigital(int CardAddress)
```
Clears all output channels

```
void SetDigitalChannel(int CardAddress, int Channel)
```
Sets the output channel

```
void SetAllDigital(int CardAddress)
```
Sets all output channels

Digital Input functions

```
bool ReadDigitalChannel(int CardAddress, int Channel)
```
Reads the status of the input channel

```
int ReadAllDigital(int CardAddress)
```
Reads the status of all the input channels

Counter functions

```
void ResetCounter(int CardAddress)
```
Resets the 32 bit pulse counter

```
unsigned int ReadCounter(int CardAddress)
```
Reads the content of the 32 bit pulse counter

Readback procedures and functions

```
void ReadBackPWMOut(int CardAddress, int *Buffer)
```
Reads back the status of the PWM outputs

```
int ReadBackInOutMode(int CardAddress)
```
Reads back the current in/out mode of the digital terminals

Function List

- OpenDevices
- CloseDevices
- ReadAnalogChannel
- ReadIAllAnalog
- InOutMode
- OutputAllDigital
- ClearDigitalChannel
- ClearAllDigital
- SetDigitalChannel
- SetAllDigital
- ReadDigitalChannel
- ReadAllDigital
- SetPWM
- OutputAllPWM
- ResetCounter
- ReadCounter
- Connected
- VersionFirmware
- VersionDLL
- ReadBackPWMOut
- ReadBackInOutMode

3.1

Open Devices

Syntax
int OpenDevices();

Result
int: If succeeded the return value will indicate the number of VM167 cards found.
1: Card in the Card Address 0 found.
2: Card in the Card Address 1 found.
3: Card in the Card Address 0 and 1 found.

Return value -1 indicates that no VM167 cards found.
Return value 0 indicates that there was problems to open the driver. Disconnect and reconnect the USB cable.

Description
Opens the communication link to the VM167 card. Loads the drivers needed to communicate via the USB port. This procedure must be performed before any attempts to communicate with the VM167 cards.

Example

```
                int Cards;
                ...
                Cards = OpenDevices();
                switch (Cards)
                {
        case 0:
           Label1->Text = "Card open error.";
           break;
        case 1:
           Label1->Text = "Card 0 connected.";
           RadioButton1->Enabled = true;
           RadioButton1->Checked = true;
           CardAddress = 0;
           Timer1->Enabled = true;
           break;
        case 2:
           Label1->Text = "Card 1 connected.";
           RadioButton2->Enabled = true;
           RadioButton2->Checked = true;
           CardAddress = 1;
           Timer1->Enabled = true;
           break;
        case 3:
           Label1->Text = "Cards 0 and 1 connected.";
           RadioButton1->Enabled = true;
           RadioButton1->Checked = true;
           RadioButton2->Enabled = true;
           RadioButton2->Checked = false;
           CardAddress = 0;
           Timer1->Enabled = true;
           break;
        case -1:
           Label1->Text = "Card not found.";
           break;
                }
```

3.2

CloseDevices

Syntax
void CloseDevices();

Description
Unloads the communication routines for VM167 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)
                {
                        CloseDevices();
                }
```

3.3

ReadAnalogChannel

Syntax
int ReadAnalogChannel(int CardAddress, int Channel);

Parameters
CardAddress: The address of previously opened card.
Channel: Value between 1 and 5 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 10-bit Analog to Digital converter channel is converted to a value which lies between 0 and 1023.

Example

```
int a5;
a5 = ReadAnalogChannel(CardAddress, 5);
label19->Text = a5.ToString();
```

3.4

ReadIAllAnalog

Syntax
void ReadAllAnalog(int CardAddress, int *Buffer);

Parameter
CardAddress: The address of previously opened card.
Buffer: Pointer to an array of five 32 bit integers where the data will be read.

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

Example

```
                int Buffer[8];
    ReadAllAnalog(CardAddress, Buffer);
```

3.5

InOutMode

Syntax
void InOutMode(int CardAddress, int HighNibble, int LowNibble);

Parameters
CardAddress: The address of previously opened card.
LowNibble: 0: the digital I/O terminals 1 to 4 are outputs
LowNibble: 1: the digital I/O terminals 1 to 4 are inputs.
HighNibble: 0: the digital I/O terminals 5 to 8 are outputs
HighNibble: 1: the digital I/O terminals 5 to 8 are inputs.

Description
Set the digital terminals either inputs or outputs.

Example
```
InOutMode(CardAddress, 0, 0);
// All the digital I/O pins of card are set to outputs.
```

3.6

OutputAllDigital

Syntax
```
void OutputAllDigital(int CardAddress, int Data);
```

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.

Example
```
InOutMode(CardAddress, 0, 0);
OutputAllDigital(CardAddress, 0x55);
```

3.7

ClearDigitalChannel

Syntax
void ClearDigitalChannel(int CardAddress, int Channel);

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

```
InOutMode(CardAddress, 0, 0);
ClearDigitalChannel(CardAddress, 3);
```

3.8

ClearAllDigital

Syntax
```
void ClearAllDigital(int CardAddress);
```

Parameter
CardAddress: The address of previously opened card.

Result
All digital outputs are cleared.

Example
```
                InOutMode(CardAddress, 0, 0);
    ClearAllDigital(CardAddress);
```

3.9

SetDigitalChannel

Syntax
void SetDigitalChannel(int CardAddress, int Channel);

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.

Example

```
if (CheckBox3->Checked)
      SetDigitalChannel(CardAddress,  3);
else
   ClearDigitalChannel(CardAddress,  3);
```

3.10

SetAllDigital

Syntax
void SetAllDigital(int CardAddress);

Parameter
CardAddress: The address of previously opened card.

Description
All the digital output channels are set.

Example
```
InOutMode(CardAddress, 0, 0);
SetAllDigital(CardAddress);
```

3.11

ReadDigitalChannel

Syntax
```
bool ReadDigitalChannel(int CardAddress, int Channel);
```

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
bool: TRUE means that the corresponding digital input of the card is HIGH.
FALSE means that the input is LOW.

Description
The status of the selected Input channel is read.

Example
```
if(ReadDigitalChannel(CardAddress, 3))
        label19->Text = "3: On";
else
        label19->Text = "3: Off";
```

3.12

ReadAllDigital

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

Parameter
CardAddress: The address of previously opened card.

Result
int: 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
```
i = ReadAllDigital(CardAddress);
CheckBox1->Checked = (i & 1)>0;
        CheckBox2->Checked = (i & 2)>0;
        CheckBox3->Checked = (i & 4)>0;
        CheckBox4->Checked = (i & 8)>0;
        CheckBox5->Checked = (i & 16)>0;
        CheckBox6->Checked = (i & 32)>0;
        CheckBox7->Checked = (i & 64)>0;
        CheckBox8->Checked = (i & 128)>0;
```

3.13

SetPWM

Syntax
```
void SetPWM(int CardAddress, int Channel, int Data, int Freq);
```

Parameters
CardAddress: The address of previously opened card.

Channel: The PWM output channel 1 or 2.

Data: Value between 0 and 255 which is to be sent to the PWM output of the card. The duty cycle of the PWM output corresponds to the data value: 0 = 0%, 255 = 100% duty cycle.

Freq: The PWM frequency:
1: 2929.68 Hz
2: 11718.75 Hz
3: 46875 Hz

Example
```
SetPWM(CardAddress, 1, 128, 3);
// The duty cycle of the PWM output #1 is set to 50%, frequency 46875 Hz
```

3.14

OutputAllPWM

Syntax
```
void OutputAllPWM(int CardAddress, int Data1, int Data2);
```

Parameters
CardAddress: The address of previously opened card.
Data1: Value between 0 and 255 which is to be sent to the PWM 1 output of the card .
Data2: Value between 0 and 255 which is to be sent to the PWM 2 output of the card

Example
```
OutputAllPWM (0, 128, 128);
// The duty cycle of the PWM outputs of card address 0 are set to 50%
```

3.15

ResetCounter

Syntax
void ResetCounter(int CardAddress);

Parameter
CardAddress: The address of previously opened card.

Description
The 32 bit pulse counter is reset.

Example
```
ResetCounter(CardAddress);
```

3.16

ReadCounter

Syntax

```
unsigned int ReadCounter(int CardAddress);
```

Parameter

CardAddress: The address of previously opened card.

Result

unsigned int: The content of the 32 bit pulse counter.

Description

The function returns the status of the 32 bit pulse counter.
The counter counts the pulses fed to the digital terminal number 1. Both input and output pulses are counted. The counter increments on the rising edge.

Example

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

3.17

Connected

Syntax
int Connected();

Result
int: The return value indicate the number of VM167 cards connected.
0: No cards connected
1: Card at the Card Address 0 found.
2: Card at the Card Address 1 found.
3: Cards at the Card Address 0 and 1 found.

Description
Checks that USB connection to the cards is valid.

Example

```
int Cards;
...
Button1_Click(..)
{
        Cards = OpenDevices();          // open
        ...
}
...
if (Cards != Connected())               // check if card still connected
        Button1_Click(sender, e);  // if not, try to reconnect
```

3.18

VersionFirmware

Syntax
int VersionFirmware(int CardAddress);

Parameters
CardAddress: The address of previously opened card.

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

Description
The firmware version info of the card is read.

Example

```
int ver;
ver = VersionFirmware(CardAddress);
label17->Text  = "Firmware  v"+(ver >> 24).ToString()+ "."       +((ver >> 16) &
0xFF).ToString()
          +"."+((ver  >> 8)  & 0xFF).ToString()+  "."+(ver  & 0xFF).ToString();
```

3.19

VersionDLL

Syntax
int VersionDLL();

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;
ver = VersionDLL();
label18->Text  = "DLL v"+(ver >> 24).ToString()+ "."+((ver >> 16) &
0xFF).ToString()+ "."
                +((ver >> 8) & 0xFF).ToString()+  "."+(ver & 0xFF).ToString();
```

3.20

ReadBackPWMOut

Syntax
void ReadBackPWMOut(int CardAddress, int *Buffer);

Parameter
CardAddress: The address of previously opened card.
Buffer: Pointer to array of two 32 bit integers where the data will be read.

Description
The values of the PWM outputs are read back to an array of two 32 bit integers.

Example

```
int PWM[2];
ReadBackPWMOut(CardAddress,  PWM);
TrackBar1->Value = PWM[0];
TrackBar2->Value = PWM[1];
Label15->Text = TrackBar1->Value.ToString();
Label16->Text = TrackBar2->Value.ToString();
```

3.21

ReadBackInOutMode

Syntax
int ReadBackInOutMode(int CardAddress);

Parameter
CardAddress: The address of previously opened card.

Result
int: Value between 0 and 3 representing the input/output mode of the digital terminals.
0: All the digital I/O terminals are **outputs**
1: The digital I/O terminals 1 to 4 are **inputs** and terminals 5 to 8 are **outputs**.
2: The digital I/O terminals 1 to 4 are **outputs** and terminals 5 to 8 are **inputs**.
3: All the digital I/O terminals are **inputs**

Description
The input / output mode of the digital terminals is read back.

Example

```
int IOmode;
IOmode = ReadBackInOutMode(CardAddress);
if ((IOmode & 1)>0)
        RadioButton3->Checked = true;
else
        RadioButton4->Checked = true;
if ((IOmode & 2)>0)
        RadioButton6->Checked = true;
else
        RadioButton5->Checked = true;
```


IV

Function declarations in other programming languages

- Visual Basic 6.0
- Visual Basic 2008 Express
- Visual C# 2008 Express
- Delphi
- Borland C++Builder

4.1

Visual Basic 6.0

```
Private Declare Function OpenDevices Lib "vm167.dll" () As Long
Private Declare Sub CloseDevices Lib "vm167.dll" ()
Private Declare Sub InOutMode Lib "vm167.dll" (ByVal CardAddress As Long, ByVal HighNibble As Long, ByVal LowNibble As Long)
Private Declare Function ReadAnalogChannel Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Channel As Long) As Long
Private Declare Sub ReadAllAnalog Lib "vm167.dll" (ByVal CardAddress As Long, ByRef Buffer As Long)
Private Declare Sub OutputAllDigital Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Data As Long)
Private Declare Sub ClearDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Channel As Long)
Private Declare Sub ClearAllDigital Lib "vm167.dll" (ByVal CardAddress As Long)
Private Declare Sub SetDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Channel As Long)
Private Declare Sub SetAllDigital Lib "vm167.dll" (ByVal CardAddress As Long)
Private Declare Function ReadDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Channel As Long) As Boolean
Private Declare Function ReadAllDigital Lib "vm167.dll" (ByVal CardAddress As Long) As Long
Private Declare Function ReadCounter Lib "vm167.dll" (ByVal CardAddress As Long) As Long
Private Declare Sub ResetCounter Lib "vm167.dll" (ByVal CardAddress As Long)
Private Declare Sub SetPWM Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Channel As Long, ByVal Data As Long, ByVal Freq As Long)
Private Declare Sub OutputAllPWM Lib "vm167.dll" (ByVal CardAddress As Long, ByVal Data1 As Long, ByVal Data2 As Long)
Private Declare Function VersionDLL Lib "vm167.dll" () As Long
Private Declare Function VersionFirmware Lib "vm167.dll" (ByVal CardAddress As Long) As Long
Private Declare Function Connected Lib "vm167.dll" () As Long
Private Declare Sub ReadBackPWMOut Lib "vm167.dll" (ByVal CardAddress As Long, ByRef Buffer As Long)
Private Declare Function ReadBackInOutMode Lib "vm167.dll" (ByVal CardAddress As Long) As Long
```

4.2

Visual Basic 2008 Express

```
Private Declare Function OpenDevices Lib "vm167.dll" () As Integer
Private Declare Sub CloseDevices Lib "vm167.dll" ()
Private Declare Sub InOutMode Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal HighNibble As Integer, ByVal LowNibble As Integer)
Private Declare Function ReadAnalogChannel Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Channel As Integer) As Integer
Private Declare Sub ReadAllAnalog Lib "vm167.dll" (ByVal CardAddress As Integer, ByRef Buffer As Integer)
Private Declare Sub OutputAllDigital Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Data As Integer)
Private Declare Sub ClearDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Channel As Integer)
Private Declare Sub ClearAllDigital Lib "vm167.dll" (ByVal CardAddress As Integer)
Private Declare Sub SetDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Channel As Integer)
Private Declare Sub SetAllDigital Lib "vm167.dll" (ByVal CardAddress As Integer)
Private Declare Function ReadDigitalChannel Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Channel As Integer) As Boolean
Private Declare Function ReadAllDigital Lib "vm167.dll" (ByVal CardAddress As Integer) As Integer
Private Declare Function ReadCounter Lib "vm167.dll" (ByVal CardAddress As Integer) As UInteger
Private Declare Sub ResetCounter Lib "vm167.dll" (ByVal CardAddress As Integer)
Private Declare Sub SetPWM Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Channel As Integer, ByVal Data As Integer, ByVal Freq As Integer)
Private Declare Sub OutputAllPWM Lib "vm167.dll" (ByVal CardAddress As Integer, ByVal Data1 As Integer, ByVal Data2 As Integer)
Private Declare Function VersionDLL Lib "vm167.dll" () As Integer
Private Declare Function VersionFirmware Lib "vm167.dll" (ByVal CardAddress As Integer) As Integer
Private Declare Function Connected Lib "vm167.dll" () As Integer
Private Declare Sub ReadBackPWMOut Lib "vm167.dll" (ByVal CardAddress As Integer, ByRef Buffer As Integer)
Private Declare Function ReadBackInOutMode Lib "vm167.dll" (ByVal CardAddress As Integer) As Integer
```

4.3

Visual C# 2008 Express

```
[DllImport("vm167.dll")]
public static extern int OpenDevices();

[DllImport("vm167.dll")]
public static extern void CloseDevices();

[DllImport("vm167.dll")]
public static extern int ReadAnalogChannel( int CardAddress, int Channel);

[DllImport("vm167.dll")]
public static extern void ReadAllAnalog(int CardAddress, int[] Buffer);

[DllImport("vm167.dll")]
public static extern void SetPWM(int CardAddress, int Channel, int Data, int Freq);

[DllImport("vm167.dll")]
public static extern void OutputAllPWM(int CardAddress, int Data1, int Data2);

[DllImport("vm167.dll")]
public static extern void OutputAllDigital( int CardAddress, int Data);

[DllImport("vm167.dll")]
public static extern void ClearDigitalChannel( int CardAddress, int Channel);

[DllImport("vm167.dll")]
public static extern void ClearAllDigital(int CardAddress);

[DllImport("vm167.dll")]
public static extern void SetDigitalChannel( int CardAddress, int Channel);

[DllImport("vm167.dll")]
public static extern void SetAllDigital(int CardAddress);

[DllImport("vm167.dll")]
public static extern bool ReadDigitalChannel( int CardAddress, int Channel);

[DllImport("vm167.dll")]
public static extern int ReadAllDigital( int CardAddress);

[DllImport("vm167.dll")]
public static extern void InOutMode(int CardAddress, int HighNibble, int LowNibble);

[DllImport("vm167.dll")]
public static extern uint ReadCounter(int CardAddress);

[DllImport("vm167.dll")]
public static extern void ResetCounter(int CardAddress);

   [DllImport("vm167.dll")]
public static extern int Connected();

[DllImport("vm167.dll")]
public static extern int VersionFirmware( int CardAddress);

[DllImport("vm167.dll")]
public static extern int VersionDLL();

[DllImport("vm167.dll")]
public static extern void ReadBackPWMOut(int CardAddress, int[] Buffer);

                              [DllImport("vm167.dll")]
public static extern int ReadBackInOutMode( int CardAddress);
```

4.4

Delphi

```
function OpenDevices: integer; stdcall; external 'VM167.dll';
procedure CloseDevices; stdcall; external 'VM167.dll';
function ReadAnalogChannel(CardAddress: integer; Channel: integer):integer; stdcall; external 'VM167.dll';
procedure ReadAllAnalog(CardAddress: integer; Buffer: Pointer); stdcall; external 'VM167.dll';
function ReadAllDigital(CardAddress: integer): integer; stdcall; external 'VM167.dll';
procedure SetPWM(CardAddress: integer; Channel: integer; Data: integer; Frequency: integer); stdcall; external
'VM167.dll';
procedure OutputAllPWM(CardAddress: integer; Data1: integer; Data2: integer); stdcall; external 'VM167.dll';
procedure InOutMode(CardAddress: integer; HighNibble: integer; LowNibble: integer);stdcall; external 'VM167.dll';
procedure OutputAllDigital(CardAddress: integer; Data: integer);stdcall; external 'VM167.dll';
procedure ClearAllDigital(CardAddress: integer); stdcall; external 'VM167.dll';
procedure ClearDigitalChannel(CardAddress: integer; Channel: integer); stdcall; external 'VM167.dll';
procedure SetDigitalChannel(CardAddress: integer; Channel: integer); stdcall; external 'VM167.dll';
procedure SetAllDigital(CardAddress: integer); stdcall; external 'VM167.dll';
function ReadCounter(CardAddress: integer):cardinal; stdcall; external 'VM167.dll';
procedure ResetCounter(CardAddress: integer); stdcall; external 'VM167.dll';
function VersionFirmware(CardAddress: integer): integer; stdcall; external 'VM167.dll';
function VersionDLL: integer; stdcall; external 'VM167.dll';
function Connected: integer; stdcall; external 'VM167.dll';
procedure ReadBackPWMOut(CardAddress: integer; Buffer: Pointer); stdcall; external 'VM167.dll';
function ReadBackInOutMode(CardAddress: integer):integer; stdcall; external 'VM167.dll';
```

4.5

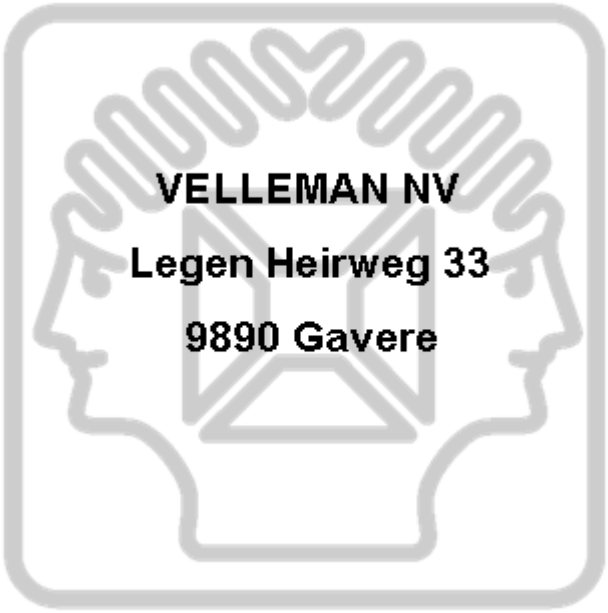
Borland C++Builder

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

#define FUNCTION __declspec (dllimport )

FUNCTION int __stdcall OpenDevices();
FUNCTION void __stdcall CloseDevices();
FUNCTION int __stdcall ReadAnalogChannel( int CardAddress, int Channel);
FUNCTION void __stdcall ReadAllAnalog( int CardAddress, int *Buffer);
FUNCTION void __stdcall SetPWM( int CardAddress, int Channel, int Data, int Freq);
FUNCTION void __stdcall OutputAllPWM( int CardAddress, int Data1, int Data2);
FUNCTION void __stdcall OutputAllDigital( int CardAddress, int Data);
FUNCTION void __stdcall ClearDigitalChannel( int CardAddress, int Channel);
FUNCTION void __stdcall ClearAllDigital( int CardAddress);
FUNCTION void __stdcall SetDigitalChannel( int CardAddress, int Channel);
FUNCTION void __stdcall SetAllDigital( int CardAddress);
FUNCTION bool __stdcall ReadDigitalChannel( int CardAddress, int Channel);
FUNCTION int __stdcall ReadAllDigital( int CardAddress);
FUNCTION void __stdcall InOutMode( int CardAddress, int HighNibble, int LowNibble);
FUNCTION unsigned int __stdcall ReadCounter( int CardAddress);
FUNCTION void __stdcall ResetCounter( int CardAddress);
FUNCTION int __stdcall Connected();
FUNCTION int __stdcall VersionFirmware( int CardAddress);
FUNCTION int __stdcall VersionDLL();
FUNCTION void __stdcall ReadBackPWMOut( int CardAddress, int *Buffer);
FUNCTION int __stdcall ReadBackInOutMode( int CardAddress);

#ifdef __cplusplus
}
#endif
```



VELLEMAN NV

Legen Heirweg 33

9890 Gavere