

**USBAD8DAC2
USBAD8DAC2/14
USBTTL24 and
respective SER &
ETH Versions**

Copyright © QUANCOM Informationssysteme GmbH

Alle Angaben in diesem Handbuch sind nach sorgfältiger Prüfung zusammengestellt worden, gelten jedoch nicht als Zusicherung von Produkteigenschaften. QUANCOM haftet ausschließlich in dem Umfang, der in den Verkaufs- und Lieferbedingungen festgelegt ist. Weitergabe und Vervielfältigung dieses Handbuches und die Verwertung seines Inhaltes sowie der zum Produkt gehörenden Software sind nur mit schriftlicher Erlaubnis von QUANCOM gestattet. Änderungen, die dem technischen Fortschritt dienen, bleiben vorbehalten.

Wesseling, Februar 2010 Version 4.4.0

Inhaltsverzeichnis

Kapitel I	Overview	1
1	Our experience is your profit.....	1
2	Customer Communication.....	1
3	Changes in this manual and software updates.....	2
4	Extend of Delivery.....	2
Kapitel II	Installation procedures	3
1	System requirements.....	3
2	Safety precautions.....	3
3	Installation of the Module.....	4
4	Addressing the module.....	5
Kapitel III	Technical Hardware description	6
1	Description.....	6
2	Card overview USBAD8DAC2, USBAD8DAC2/14, USBTTL24.....	7
3	Card overview Ethernet and Serial Versions.....	9
4	Pin assisments by differential - ended (4 channels).....	10
5	Pin assisments by single - ended (8 channels).....	10
6	Pin configuration 37 pol. D-Sub connector.....	11
7	Change power supply.....	12
8	A/D Differential-ended (d.e).....	12
9	A/D Single-ended (s.e.).....	12
10	D/A channels.....	12
11	TTL I/O.....	13
12	Counter.....	14
13	Miscellaneous.....	14
Kapitel IV	Programming of the Module	15
1	Wich software do I need?.....	15
2	QLIB: High Level programming (Windows XP / 2000 / ME / 98).....	15
	QLIB (QUANCOM Driver Library).....	16
Kapitel V	High-speed installation of the QLIB for USB	17
Kapitel VI	QLIB commands	18
1	Administrative functions.....	18
	AD/DA (simple).....	20
2	QLIB Instructions (extended).....	25
	Administrative functions (extended).....	25
	AD/DA advanced functions.....	27
	Other functions.....	31

3	Programming with the QLIB (QUANCOM Driver Library).....	32
	Read in the A/D channels under VB.....	32
	Approach the D/A channels under VB.....	33
	Read and write the TTL outputs under VB.....	34
	Read Counter 0 and Counter 1 under VB.....	36

Kapitel VII Annex

39

1	Frequently asked questions (FAQ).....	39
	General Information.....	39
	Problems with boards running under Windows 98/95 and Windows 2000/NT.....	40
2	Customer Communication and Help.....	42
3	Technical support form.....	45
4	Documentation Comment Form.....	45
5	Hardware and Software configuration form.....	47
6	Trademark.....	47

1. Overview

We congratulate you on buying the QUANCOM high quality measurement and automation board. You have chosen a product which attributes and functions show the latest updates of technology.

The following special attributes are included:

- Einfacher Anschluß über USB
- Einfach programmierbar
- Diverse Beispielprogramme in verschiedenen Programmiersprachen
- Treiberunterstützung unter Windows XP, 2000 und ME/98 mit der QLIB (**QUANCOM Driver Library**)

1.1 Our experience is your profit

QUANCOM is specialised in development of hard- and software. QUANCOM has become one of the leading suppliers of measuring and automation technology in industry. At its design centres QUANCOM has developed an impressive range of products.

1.2 Customer Communication

QUANCOM wants to receive your comments on our products and manuals. We are interested in the applications you develop with our products, and we want to help you if you have problems with them. For easy contacting, this manual contains comment and configuration forms for you to complete, which are in chapter "**Customer Communication and Help**" at the end of this manual.

1.3 Changes in this manual and software updates

QUANCOM - products are marked out by their constant further development. You can watch all the actual information of the changes in the README-file on the installation disk or CD. You can always get more information and free software updates from our internet website. www.quancom.de

1.4 Extend of Delivery

- USB-Messtechnik-Modul
- 1.8 Meter USB-Cable
- QUANCOM CD with User's manual (PDF)

If a component is missing please contact your dealer. QUANCOM reserves the right to change the extent of delivery without a preliminary announcement

2. Installation procedures

2.1 System requirements

- Personal computer: The QUANCOM boards are assigned to operate in IBM-AT compatible computers with 80X86 or compatible. (i.e. Pentium)
- Your computer must have one free port on the Universal Serial Bus (USB).



You can find more information in chapter "**Technical Hardware Description**".

2.2 Safety precautions

For the sake of your security and for a safe function of your new QUANCOM board obey the following advice:

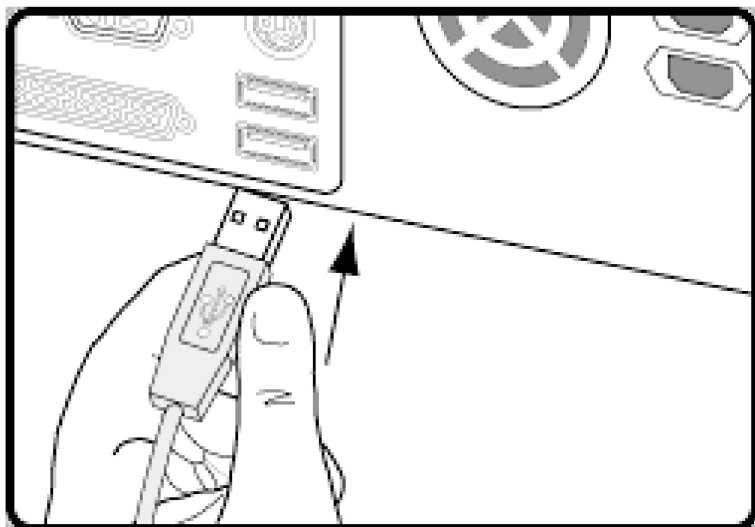
- Please unplug the computer before opening.
- Computer motherboards and components contain very delicate integrated circuit (IC) chips. You have to obey some precautions whenever you work on your computer to protect them against damage from static electricity. Use a grounded wrist strap before handling computer components. If you don't have one, touch both of your hands to a safely grounded object or to a metal object, such as the power supply case.
- Hold components by the edges and try not to touch the integrated circuit chips, leads or circuitry.
- Place components on a grounded anti-static pad or on the bag that was sent with the component whenever the components are separated from the system.



Modifications, made at the device without express permission of QUANCOM, lead to the loss of warranty and certification

2.3 Installation of the Module

Switch on the PC, start Windows and connect the USB connecting-cable with the USB port of the PC.



2.4 Adressing the module

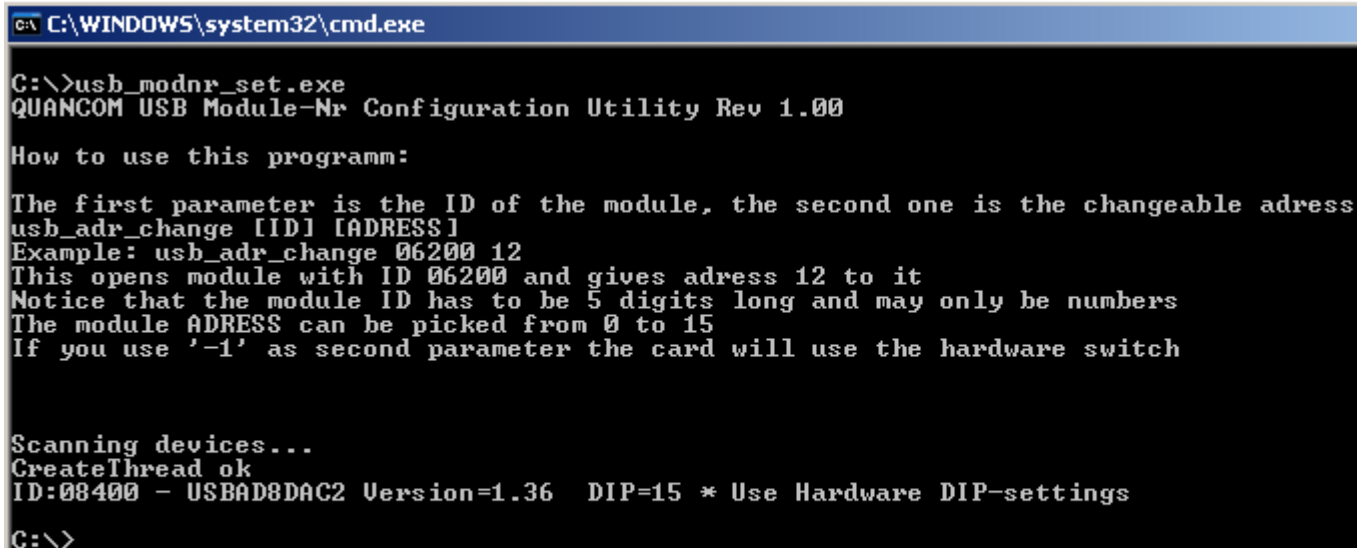
If more than one module of the same type is installed on a system Windows automatically sorts the modules in the order they where recognized.

When using this kind of addressing it may happen that the addresses change after a reboot because Windows recognizes them in another order.

To avoid that QUANCOM USBAD8DAC2, USBAD8DAC2/14 and USBTTL24 modules support the possibility to change their addresses through a small console based software tool.

You can download this program from our homepage, by following this hyperlink --> [QUANCOM Address Changer](#)

In the picture below the output is shown when no parameter has been given to the program. Further informations and a short usage description is written in the README file that comes with the program.



```
C:\WINDOWS\system32\cmd.exe
C:\>usb_modnr_set.exe
QUANCOM USB Module-Nr Configuration Utility Rev 1.00
How to use this programm:
The first parameter is the ID of the module, the second one is the changeable address
usb_adr_change [ID] [ADDRESS]
Example: usb_adr_change 06200 12
This opens module with ID 06200 and gives adress 12 to it
Notice that the module ID has to be 5 digits long and may only be numbers
The module ADDRESS can be picked from 0 to 15
If you use '-1' as second parameter the card will use the hardware switch

Scanning devices...
CreateThread ok
ID:08400 - USBAD8DAC2 Version=1.36 DIP=15 * Use Hardware DIP-settings
C:\>
```

3. Technical Hardware description

3.1 Description

The USB modules can be plugged very simply to your PC with the attached USB cable. Cause of the installed before QLIB the equipment is recognized by Windows immediately. The power supply takes place through the USB Bus. Optionally you have the possibility to operate the USB modules over an external 12V power supply. Therefore you must switch the jumper of the current supply from USB to "Extern" and plug in a separate 12V power supply unit. For the connection of the signals which should be measured there are comfortable, plug-in to you at the disposal.

AD/DA

With these professional USB modules you have the selection between 2 different entrance modes. The differential ended mode with 4 A/D channels and a resolution 12 (14) bits or the single ended mode with 8 /AD channels and a 11 (13) bits resolution. The selection which mode you want to use, you can choose easily by software. The single ended mode uses a voltage range of +10V with the same transformation time.

As outputs 2 D/A channels with a resolution of 10 bits are to you at the disposal. The D/A output uses a voltage range of + 0..5V with a current input of max. 10mA.

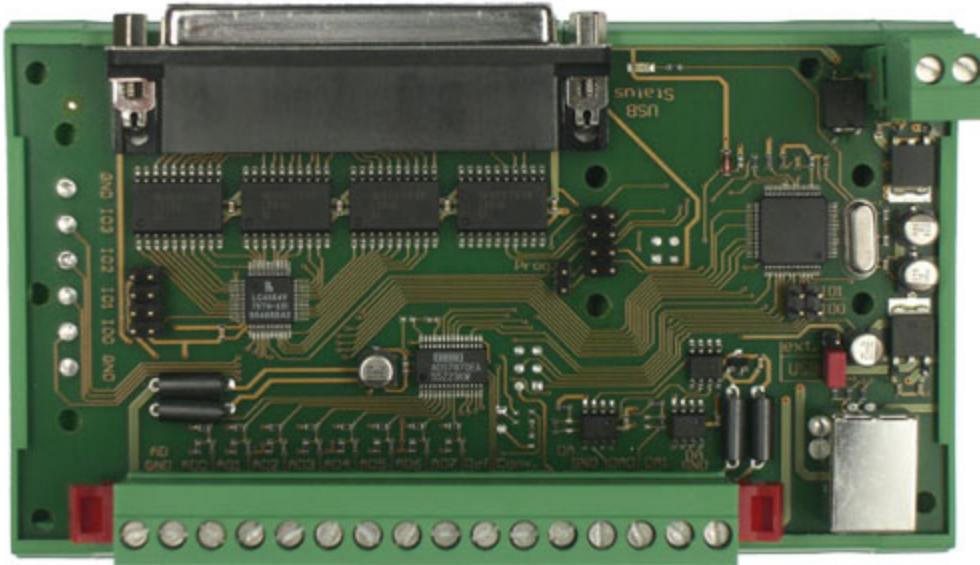
TTL

Additionally these modules have 2*32Bit Counter and 24 TTL channels which can be used as in- or outputs. The input and output of the 24 TTL/IO channels and the two 32 bits Counter works through the 37-pin D-SUB-socket. You get to know the pin allocation of the D-SUB socket in chapter 3.6.

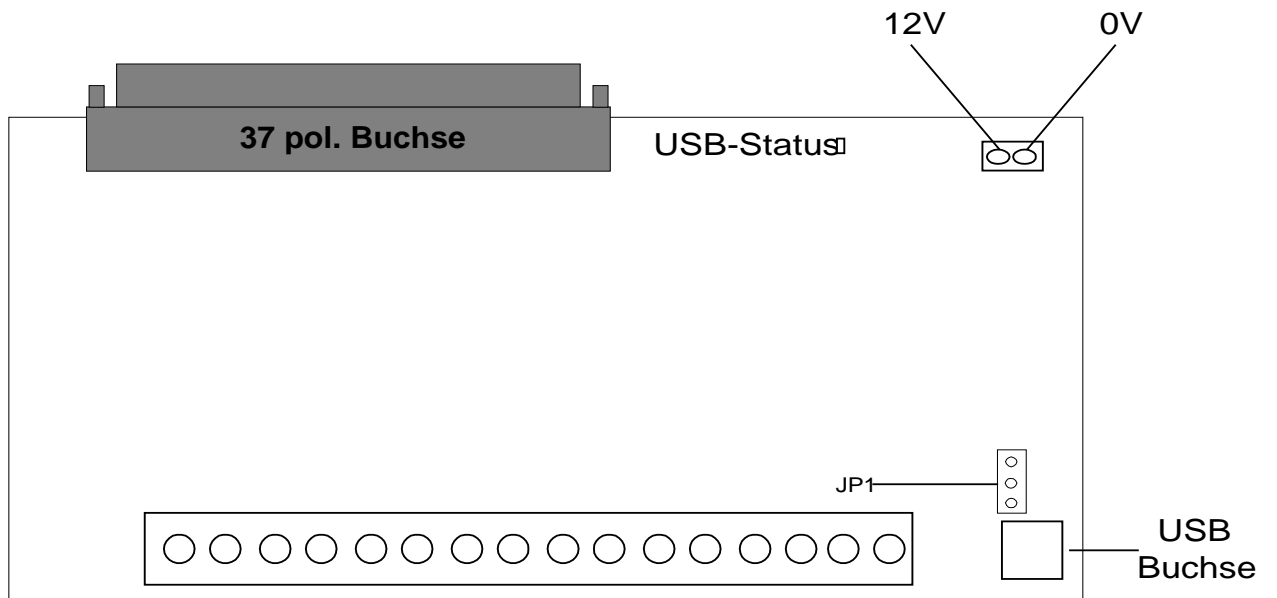
3.2 Card overview USBAD8DAC2, USBAD8DAC2/14, USBTTL24

Shown below is the QUANCOM USBAD8DAC2 / USBAD8DAC2/14. The USBTTL24 is almost the same layout except the missing 37 pole D-SUB connector.

Original View



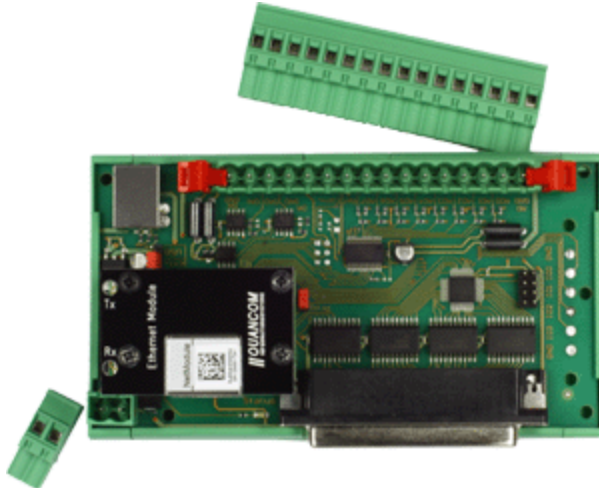
Schematic View



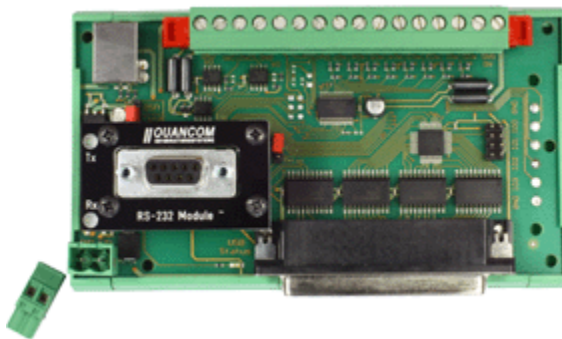
3.3 Card overview Ethernet and Serial Versions

Shown below is the QUANCOM ETHAD8DAC2 / ETHAD8DAC2/14 and SERAD8DAC2 / SERAD8DAC2/14. The SERTTL24 and ETHHTTL24 is almost the same layout except the missing 37 pole D-SUB connector.

Original View Ethernet Version

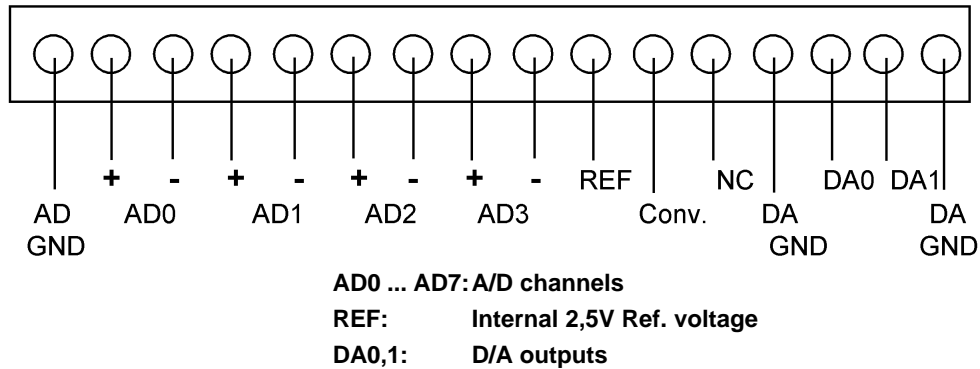


Original View Serial Version



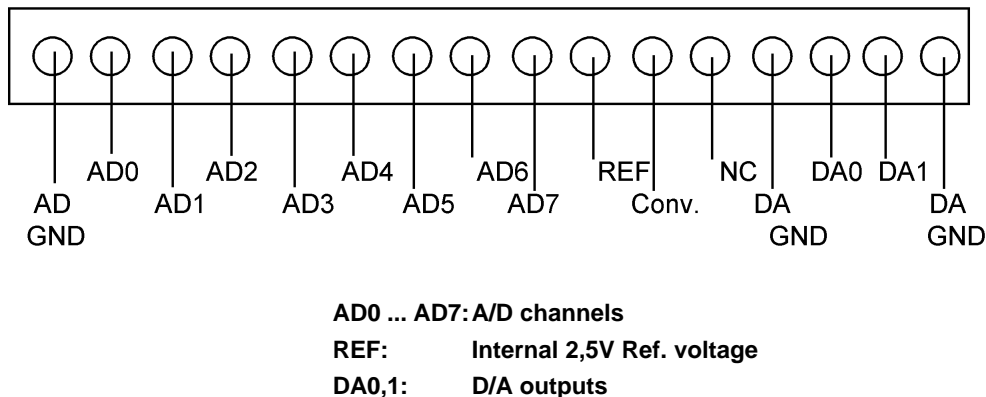
3.4 Pin assignments by differential - ended (4 channels)

Below you see the pin assignments of the plug able 16 pole pin in differential ended mode. The REF is the +5V reference voltage. Conv. and NC are not connected.



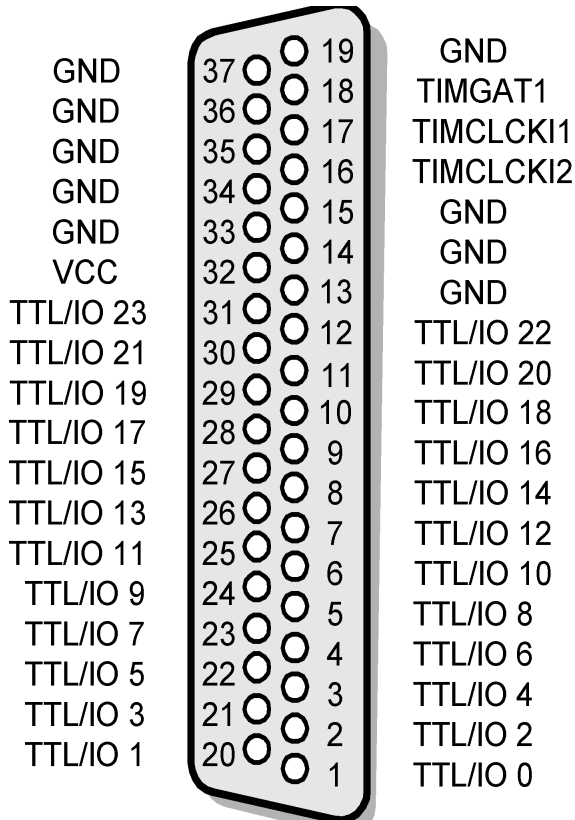
3.5 Pin assignments by single - ended (8 channels)

Below you see the pin assignments of the plug able 16 pole pin in single ended mode. The REF is the +2,5V reference voltage. Conv. and NC are not connected.



3.6 Pin configuration 37 pol. D-Sub connector

Below you see the pin assignments of 37 pole D-SUB connector.



TTL/IO 0 - 23	24 TTL/IO channels
TIMGAT1	Gate for Timer 1 (Timer is enabled by Gate =1)
TIMCLKI 1-2	Counter Inputs 1 - 2
VCC	+5 V
GND	Ground

3.7 Change power supply

With these modules you can choose between power supply through USB and an external 5V - 6V supply. By changing the jumper setting of JP1 you may change the power supply to fit your needs.

Note that the ETH and SER versions do need an external power supply in order to work properly.



extern



USB*

(* default setting)

3.8 A/D Differential-ended (d.e)

A/D channels	4
A/D Resolution	12 Bit (optional 14 Bit)
A/D Conversion time	20 us
A/D Span	$\pm 20\text{ V}$, $\pm 10\text{ V}$, $\pm 5\text{ V}$, $\pm 4\text{ V}$, $\pm 2,5\text{ V}$, $\pm 2\text{ V}$, $\pm 1,25\text{ V}$, $\pm 1\text{ V}$

3.9 A/D Single-ended (s.e.)

A/D Channels	8
A/D Resolution	11 Bit (optional 13 Bit)
A/D Conversion time	20 us
A/D Span	$\pm 10\text{ V}$

3.10 D/A channels

D/A channels	2
D/A Resolution	10 Bit
D/A Voltage range	0..5V
D/A max. electricity	10mA
Connector	16 pol. Terminal block

3.11 TTL I/O

Channels	24 (in 8er blocks)
Connector	37-pol. D-Sub

3.12 Counter

Number of	2
Resolution	32 Bit
Frequency	Max. 1MHz
Connector	37-pol. D-Sub

3.13 Miscellaneous

Dimension	132*72 mm
Temperature area	0...50°

4. Programming of the Module

4.1 Wich software do I need?

The needed software depends on the used operating system and / or programming IDE.

To obtain access to the card using a program written by your own you have the following possibilities:

Method 1

High Level programming using the QLIB driver library and its given DLLs on all common Windows systems. The QLIB does support almost all high level languages and compilers such as, MS Visual C, MS Visual Basic, Borland C++ Builder and Borland Delphi, for example. Samples are supported after the installation of our "**QLIB Samples Package**" which is accessible on our homepage through the following link

[QLIB Samples Package](#)

Method 2

Using the QLIB functions through a graphical programming system such as LabView or Agilent VEE. Samples are supported after the installation of our "**QLIB Samples Package**" which is accessible on our homepage through the following link

[QLIB Samples Package](#)

4.2 QLIB: High Level programming (Windows XP / 2000 / ME / 98)

4.2.1 QLIB (QUANCOM Driver Library)

The **QLIB**, which stands for **QUANCOM Driver LIBRARY**, was developed with the target to allow the simple programming of all our data acquisition products under various operating systems. So it is easy to write an application that runs under the operating systems Windows Me/98/95 and Windows XP/2000/NT4.0. This driver interface is not limited to PC boards or other I/O adapters but is also targeted towards supporting the next product generations currently being developed. The used functions and parameters are the same for all operating systems.

Supported operating systems:

- <%OPERATINGSYS_ALL%>

Compiler:

C / C++

- Borland C++ 3.1, 4.x, 5.x
- Microsoft® Visual C++ 1.x, 2.x, 4.x, 5.x, 6.x

Pascal

- Borland Turbo Pascal

Delphi

- Borland Delphi

Basic

- Microsoft® Visual Basic 3.x, 4.x, 5.x; 6.x

Graphical Programming Language

- Agilent VEE from Agilent Technologies
- LabView® from National Instruments

5. High-speed installation of the QLIB for USB



For the installation of the drivers and run time environment administrator rights are necessary. Without the appropriate rights the driver and the run time environment cannot be installed correctly.

If you have an old version of the QLIB installed on your system please uninstall this version before installing the new one.

Windows XP/2000	Windows ME/98
Connect USB module	Connect USB module
Start system	Start system
A new device will be recognized by Windows. Please advise the path to the drivers, which are located on the installation CD in the directory WINXP or WIN2000 .	Optional: A new device will be recognized by Windows. Please advise the path to the drivers, which are located on the installation CD in the directory WINME or WIN98 .
Optional: Insert the installation CD and wait the auto start program to be started. follow the installation procedures on your screen to install drivers, IDE DLLs and programming samples.	Optional: Insert the installation CD and wait the auto start program to be started. follow the installation procedures on your screen to install drivers, IDE DLLs and programming samples.
Reboot your system	Reboot your system

6. QLIB commands

The following chapter is about all Qlib commands for this Card. They differ by extended (QAOIEXT...) and simple

Die folgende Auflistung enthält alle von dieser Karte verwendeten QLIB-Befehle. Diese unterscheiden sich in erweiterte (QAPIExt...) und einfache (QAPI...) Funktionen. Bei den einfachen Funktionen werden keine Kartenhandles (Kartenparameter) übergeben. Dadurch wird nur die erste vom System erkannte Karte eines Typs angesprochen und verwendet. Sollen mehrere Karten verwendet werden, so müssen die erweiterten Funktionen angewendet werden.

6.1 Administrative functions

QAPINumOfCards

```
ULONG QAPINumOfCards (void);
```

It is possible to ask , which used cards are supported by the QLIB with the function QAPINumOfCards.

QAPIGetLastError

The function **QAPIGetLastError** give out the last failure code of the called Threads.

Its possible to save more then only one failure code.

ULONG QAPIGetLastError (void);

QAPIGetLastErrorCode

The function **QAPIGetLastErrorCode** give out the last extended failure code from a QAPIGetLastError commands.

ULONG QAPIGetLastErrorCode (void);

6.1.1 AD/DA (simple)

QAPIGetAD

Use the function **QAPIGetAD** to read a digital value from the Input channel of the A/D-Card.

ULONG **QAPIGetAD** (ULONG **cardid**,ULONG **channel**,);

Function prototype for Delphi and VB

Parameter

cardid

Give out the ID of the Card to get the Information from the Card.

channel

Indicates the channel, by which the digital value is to be read in

Mode		Value (hex)	Value (dez)
MODE_DIFFERENTIAL_ENDED or MODE_BI_20V	0x10000+10	0x1000A	65546
MODE_DIFFERENTIAL_ENDED or MODE_BI_10V	0x10000+1	0x10001	65537
MODE_DIFFERENTIAL_ENDED or MODE_BI_5V	0x10000+0	0x10000	65536
MODE_DIFFERENTIAL_ENDED or MODE_BI_4V	0x10000+20	0x10014	65556
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V5	0x10000+6	0x10006	65542
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V	0x10000+12	0x1000C	65548
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V25	0x10000+7	0x10007	65543
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V	0x10000+14	0x1000E	65550
MODE_SINGLE_ENDED or MODE_BI_10V	0x20000+1	0x20001	131073

QAPIPutDA

Use the function **QAPIPutDA** to give out a digital value to a channel of the A/D-Card.

ULONG QAPIPutDA (ULONG **cardid,ULONG **channel**,ULONG **value**);**

Function prototype for Delphi and VB

Parameter

cardid

Give out the ID of the Card to give out Information to the Card.

channel

Give out the channel to give out a digital.

value

Give out the digital value.

QAPIConvertDWTToVoltage

With the function **QAPIConvertDWTToVoltage** a digital value get convert to a analoge value.

This function is for all Cards with a A/D Converter.

```
float QAPIConvertDWTToVoltage (ULONG cardid,ULONG value,ULONG mode);
```

Bis QLIB Version 1.96

```
float QAPIConvertDWTToVoltage (ULONG cardid,ULONG value,);
```

Function prototype for Delphi and VB

Parameter

cardid

Give out the ID of the Card from wich the digital value will get convert.

value

Give out the digital value who will be convert.

mode

Give out the Card dependent parameters.

Mode		Value (hex)	Value (dez)
MODE_DIFFERENTIAL_ENDED or MODE_BI_20V	0x10000+10	0x1000A	65546
MODE_DIFFERENTIAL_ENDED or MODE_BI_10V	0x10000+1	0x10001	65537
MODE_DIFFERENTIAL_ENDED or MODE_BI_5V	0x10000+0	0x10000	65536
MODE_DIFFERENTIAL_ENDED or MODE_BI_4V	0x10000+20	0x10014	65556
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V5	0x10000+6	0x10006	65542
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V	0x10000+12	0x1000C	65548
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V25	0x10000+7	0x10007	65543
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V	0x10000+14	0x1000E	65550
MODE_SINGLE_ENDED or MODE_BI_10V	0x20000+1	0x20001	131073

QAPICovertVoltageToDW

With the function **QAPICovertVoltageToDW** a analoge value get convert to a digital value.

This function is for all Cards with a D/A Converter.

ULONG QAPICovertVoltageToDW (**ULONG cardid**,float **value**,**ULONG mode**);

Function prototype for Delphi and VB

Parameter

cardid

Give out the ID of the Card from wich the digital value will get convert.

value

Give out the digital value who will be convert.

mode

Give out the Card dependent parameters.

Mode		Value (hex)	Value (dez)
MODE_UNI_5V	0x+4	0x4	4

6.2 QLIB Instructions (extended)

6.2.1 Administrative functions (extended)

QAPIExtOpenCard

```
ULONG QAPIExtOpenCard ( ULONG cardid, ULONG devnum );
```

Use the function QAPIExtOpenCard to open a board and retrieve the board handle

QAPIExtCloseCard

```
void QAPIExtCloseCard ( ULONG cardhandle );
```

The function QAPIExtCloseCard will close the board.

QAPIExtNumOfCards

ULONG **QAPIExtNumOfCards** (void);

It is possible to ask , which used cards are supported by the QLIB with the function

QAPIExtNumOfCards

QAPIGetLastErrorStringEx

The **QAPIGetLastErrorStringEx** use a string to produce a readable Error Message from the Qlib Error code. This will be delivert back from the **QAPIGetLastErrord** alternatively **QAPIGetLastErrorCode** function.

ULONG QAPIGetLastErrorStringEx(char* buffer, **ULONG** buffersize);

Function prototype for Delphi and Vb

Parameter

buffer

Pointer to a buffer of the empty string error.

buffersize

This Parameter is the size of the buffer passed in bytes.

6.2.2 AD/DA advanced functions

QAPIExtReadAD

With the function **QAPIExtReadAD** it is possible to read in a digital value from a input channel of the A/D card.

```
ULONG QAPIExtReadAD(ULONG cardhandle,ULONG channel,ULONG mode);
```

Parameter

cardhandle

Indicates the handle of the open card.

channel

Indicates the channel from wich a digital value will be imported.

mode

Indicates a Parameter dependent to the card.

Mode		Value (hex)	Value (dez)
MODE_DIFFERENTIAL_ENDED or MODE_BI_20V	0x10000+10	0x1000A	65546
MODE_DIFFERENTIAL_ENDED or MODE_BI_10V	0x10000+1	0x10001	65537
MODE_DIFFERENTIAL_ENDED or MODE_BI_5V	0x10000+0	0x10000	65536
MODE_DIFFERENTIAL_ENDED or MODE_BI_4V	0x10000+20	0x10014	65556
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V5	0x10000+6	0x10006	65542
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V	0x10000+12	0x1000C	65548
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V25	0x10000+7	0x10007	65543
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V	0x10000+14	0x1000E	65550
MODE_SINGLE_ENDED or MODE_BI_10V	0x20000+1	0x20001	131073

QAPIExtWriteDA

With this function **QAPIExtWriteDA** will be give out a digital value on a channel of the D/A-Card.

```
void QAPIExtWriteDA(ULONG cardhandle,ULONG channel,ULONG value,ULONG mode);
```

Function prototype for Delphi and VB

Parameter

cardhandle

Indicates the handle of the open card.

channel

Indicates the channel from wich a digital value will be give out.

value

Indicates the Digital value wich will be give out.

mode

Indicates a Parameter dependent to the card.

QAPIExtLatchDA

With this function **QAPIExtLatchDA** will be give out the digital values from all channels.

```
void QAPIExtLatchDA (ULONG cardhandle);
```

Function prototype for Delphi and VB

Parameter

cardhandle

Indicates the handle of the open card.

QAPIExtConvertDWToVoltage

With this function **QAPIExtConvertDWToVoltage** will be convert a digital value to a Analog value..

This function is available for all card with D/A Converter.

float **QAPIExtConvertDWToVoltage** (**ULONG cardhandle**,**ULONG value**,**ULONG mode**);

Function prototype for Delphi and VB

Parameter

cardhandle

Indicates the handle of the open card.

value

Indicates the digital value wich will be convert.

mode

Indicates a Parameter dependent to the card.

Mode		Value (hex)	Value (dez)
MODE_DIFFERENTIAL_ENDED or MODE_BI_20V	0x10000+10	0x1000A	65546
MODE_DIFFERENTIAL_ENDED or MODE_BI_10V	0x10000+1	0x10001	65537
MODE_DIFFERENTIAL_ENDED or MODE_BI_5V	0x10000+0	0x10000	65536
MODE_DIFFERENTIAL_ENDED or MODE_BI_4V	0x10000+20	0x10014	65556
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V5	0x10000+6	0x10006	65542
MODE_DIFFERENTIAL_ENDED or MODE_BI_2V	0x10000+12	0x1000C	65548
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V25	0x10000+7	0x10007	65543
MODE_DIFFERENTIAL_ENDED or MODE_BI_1V	0x10000+14	0x1000E	65550
MODE_SINGLE_ENDED or MODE_BI_10V	0x20000+1	0x20001	131073

QAPIExtConvertVoltageToDW

With this function **QAPIExtConvertVoltageToDW** will be convert a Analog Value to a Digital value.

This function is available for all card with A/D Converter..

ULONG **QAPIExtConvertVoltageToDW** (ULONG **cardhandle**,float **value**,ULONG **mode**);

Function prototype for Delphi and VB

Parameter

cardhandle

Indicates the handle of the open card.

value

Indicates the Analog value wich will be convert.

mode

Indicates a Parameter dependent to the card.

Mode		value (hex)	value (dez)
MODE_UNI_5V	0x+4	0x4	4

6.2.3 Other functions

QAPIExtGetCardInfo

```
LPCARDDATAS QAPIExtGetCardInfo( ULONG cardid );
```

It is possible to get some information about the card with the function QAPIExtGetCardInfo.

QAPIExtGetCardInfoEx

```
ULONG QAPIExtGetCardInfoEx( ULONG cardid LPCARDDATAS lpcd );
```

It is possible to get some information about the card with the function QAPIExtGetCardInfoEx. These will be written into the applications memory

QAPIExtReleaseCardInfo

```
void QAPIExtReleaseCardInfo( LPCARDDATAS carddatas );
```

It is possible with QAPIExtGetCardInfo to get out the asked card information with the function QAPIExtReleaseCardInfo

6.3 Programming with the QLIB (QUANCOM Driver Library)

Make sure that the QLIB (QUANCOM Driver Library) is properly installed. For further information about the installation and how to include the necessary files in your application see the “QLIB” documentation. This chapter describes the special commands that are required to use a QUANCOM board with the QLIB (QUANCOM Driver Library).

6.3.1 Read in the A/D channels under VB

```
Function IsError() As Integer

    Dim buffer As String * 256
    Dim Result As Long

    Result = QAPIGetLastError()

    If (Result <> 0) Then

        Result = QAPIGetLastErrorStringEx(buffer, Len(buffer))

        MsgBox (Left(buffer & " ", Result))

    Else

        End If

    IsError = (Result <> 0)

End Function

' read analog inputs
Sub    ButtonReadAI()

    Dim nDevice As Long
    Dim handle As Long
    Dim advalue As Long
    Dim value As Single
    Dim nMode As Long
    Dim channel as long

    ' find first module

    nDevice = 0

    ' open module

    handle = QAPIExtOpenCard(USBAD8DAC2, nDevice)

    if ( handle = 0 ) then
        MsgBox "Unable to open any USBAD8DAC2 module!"
        exit sub
    end if

    ' read all 8 adc channels

    for channel = 0 to 7

        ' 0 = MODE_BI_5V
        ' 1 = MODE_BI_10V
        ' 2 = MODE_BI_3V3
        ' 6 = MODE_BI_2V5
        ' 7 = MODE_BI_1V25
        ' 4 = MODE_UNI_5V
        ' 3 = MODE_UNI_10V
        ' 5 = MODE_UNI_3V3
        ' 8 = MODE_UNI_2V5
        ' 9 = MODE_UNI_1V25

        nMode = MODE_BI_10V

    next channel

End Sub
```

```

' read a/d value
advalue = QAPIExtReadAD(handle, channel, nMode): If IsError() Then goto terminate_sub

' convert digital value to floating point voltage value
value = QAPIExtConvertDWTToVoltage(handle, advalue, nMode): If IsError() Then goto
terminate_sub

' print voltage to console
debug.print "Channel " & channel & " = " & value & " Volts"

next channel
terminate_sub:
    call QAPIExtCloseCard(handle)
end sub

```

6.3.2 Approach the D/A channels under VB

```

Function IsError() As Integer
    Dim buffer As String * 256
    Dim Result As Long

    Result = QAPIGetLastError()

    If (Result <> 0) Then
        Result = QAPIGetLastErrorStringEx(buffer, Len(buffer))
        MsgBox (Left(buffer & " ", Result))
    Else
        End If

    IsError = (Result <> 0)
End Function

' read analog inputs
Sub    ButtonWriteAO()

    Dim nDevice As Long
    Dim handle As Long
    Dim davalue As Long
    Dim value As Single
    Dim nMode As Long
    Dim channel as long

    ' find first module
    nDevice = 0

    ' open module
    handle = QAPIExtOpenCard(USBAD8DAC2, nDevice)

    if ( handle = 0 ) then
        MsgBox "Unable to open any USBAD8DAC2 module!"
        exit sub
    end if

    ' write to all 2 dac channels
    nMode = MODE_BI_5V

    ' convert float voltage value to digital voltage and write it to dac output 0

```

```

value = 1.0          ' voltage
channel = 0          ' channel

davalue = QAPIExtConvertVoltageToDW(handle, value, MODE_UNI_5V): If IsError() Then GoTo
Call QAPIExtWriteDA(handle, channel, davalue, 0): If IsError() Then GoTo lab_exit

' convert float voltage value to digital voltage and write it to dac output 1

value = 2.55        ' voltage
channel = 1          ' channel

davalue = QAPIExtConvertVoltageToDW(handle, value, MODE_UNI_5V): If IsError() Then GoTo
Call QAPIExtWriteDA(handle, channel, davalue, 0): If IsError() Then GoTo lab_exit

terminate_sub:

call QAPIExtCloseCard(handle)

end sub

```

6.3.3 Read and write the TTL outputs under VB

```

Function IsError() As Integer

Dim buffer As String * 256
Dim Result As Long

Result = QAPIGetLastError()

If (Result <> 0) Then

    Result = QAPIGetLastErrorStringEx(buffer, Len(buffer))

    MsgBox (Left(buffer & " ", Result))

Else

End If

IsError = (Result <> 0)

End Function

' read analog inputs

Sub ButtonTTL()

Dim nDevice As Long
Dim handle As Long
Dim nMode As Long
Dim channel as long
Dim DDR as long
Dim value as long

' find first module

nDevice = 0

' Step 1: Open the module

handle = QAPIExtOpenCard(USBAD8DAC2, nDevice)

if ( handle = 0 ) then
    MsgBox "Unable to open any USBAD8DAC2 module!"
    exit sub
end if

' Step 2: Set the data direction of the lines
'
' The TTL lines of the USBAD8DAC2 module can be grouped in blocks of eight lines.
'
' Bit 0 ( 1 ):    = 1 Select output as data direction for the channels 0...7
' Bit 1 ( 2 ):    = 1 Select output as data direction for the channels 8...15
' Bit 2 ( 4 ):    = 1 Select output as data direction for the channels 16...23
'
' If the corresponding bit is cleared the lines are set to input mode which is the default mode.
'

```

```

' Value written to the DDR          Channels 0...7 Channels 8...15      Channels 16...23
' 0                               Inputs      Inputs      Inputs      Inputs
' 1                               Outputs      Inputs      Inputs      Inputs
' 2                               Inputs      Outputs      Inputs      Inputs
' 3                               Outputs      Outputs      Inputs      Inputs
' 4                               Inputs      Inputs      Inputs      Outputs
' 5                               Outputs      Inputs      Inputs      Outputs
' 6                               Inputs      Outputs      Inputs      Outputs
' 7                               Outputs      Outputs      Outputs      Outputs
'
' Sample:
'
' To set the channels 0...7 and channels 16...23 to output mode and the channels 8...15 to
' input mode
' you have to add the decimal values 1 + 4 = 5 and write this value to the DDR
'
'
' DDR = 1 + 4
'
' Call QAPIExtSpecial(handle, JOB_WRITE_DDR, DDR, 0): If IsError() Then GoTo error_exit
'
' Reading and writing to the lines can be done with the eight different commands QAPIExtReadDI1,
' QAPIExtReadDI8
' QAPIExtReadDI16, QAPIExtReadDI32, QAPIExtWriteDO1, QAPIExtWriteDO8, QAPIExtWriteDO16 and
' QAPIExtWriteDO32.
'
' The definitions of the functions are as following:
'
' lines = QAPIExtReadDI1(handle, channel, mode)      ' mode=1 -> don't latch inputs ( only if supported by hardware )
' lines = QAPIExtReadDI8(handle, channel, mode)      ' mode = 1 -> don't latch inputs ( only if supported by hardware )
' lines = QAPIExtReadDI16(handle, channel, mode)     ' mode = 1 -> don't latch inputs ( only if supported by hardware )
' lines = QAPIExtReadDI32(handle, channel, mode)     ' mode = 1 -> don't latch inputs ( only if supported by hardware )
' call QAPIExtWriteDO1(handle, channel, value, mode) ' mode = 1 -> don't latch outputs ( only if supported by hardware )
' call QAPIExtWriteDO8(handle, channel, value, mode) ' mode = 1 -> don't latch outputs ( only if supported by hardware )
' call QAPIExtWriteDO16(handle, channel, value, mode) ' mode = 1 -> don't latch outputs ( only if supported by hardware )
' call QAPIExtWriteDO32(handle, channel, value, mode) ' mode = 1 -> don't latch outputs ( only if supported by hardware )
'
' Step 3: Reading inputs
' read the a block of lines from channel 8...15
' 0 = channels 0...7    1= channels 8...15    2 = channels 16...23
'
'
' channel = 1
'
' lines = QAPIExtReadDI8(handle, channel, 1)      ' mode = 1 -> don't latch inputs
'
' lines now holds the state of of the channels 8...15
'
' if ( lines and 1 ) then
'     debug.print "channel 8 is high"
' else
'     debug.print "channel 8 is low"
' end if
'
' if ( lines and 2 ) then
'     debug.print "channel 9 is high"
' else
'     debug.print "channel 9 is low"
' end if
'
' if ( lines and 4 ) then
'     debug.print "channel 10 is high"
' else
'     debug.print "channel 10 is low"
' end if
'
' if ( lines and 8 ) then
'     debug.print "channel 11 is high"
' else
'     debug.print "channel 11 is low"
' end if
'
' if ( lines and 16 ) then
'     debug.print "channel 12 is high"
' else

```

```

        debug.print "channel 12 is low"
    end if

    if ( lines and 32 ) then
        debug.print "channel 13 is high"
    else
        debug.print "channel 13 is low"
    end if

    if ( lines and 64 ) then
        debug.print "channel 14 is high"
    else
        debug.print "channel 14 is low"
    end if

    if ( lines and 128 ) then
        debug.print "channel 15 is high"
    else
        debug.print "channel 15 is low"
    end if

    ' Step 4: Writing the outputs
    '
    ' 0 = channels 0...7    1= channels 8...15    2 = channels 16...23
    '
    ' set the lines channel 0, channel 1, channel 2 and channel 7 to high
    '
    ' channel 0:          1
    ' channel 1:          2
    ' channel 2:          4
    ' channel 3:          8
    ' channel 4:         16
    ' channel 5:         32
    ' channel 6:         64
    ' channel 7:        128
    '
    channel = 0

    value = 1 + 2 + 4 + 128

    call QAPIExtWriteD08(handle, channel, value, 0)

    '
    ' 0 = channels 0...7    1= channels 8...15    2 = channels 16...23
    '
    ' set the lines channel 19 and channel 20 to high
    '
    ' channel 16:         1
    ' channel 17:         2
    ' channel 18:         4
    ' channel 19:         8
    ' channel 20:        16
    ' channel 21:        32
    ' channel 22:        64
    ' channel 23:       128
    '
    channel = 2

    value = 8 + 16

    call QAPIExtWriteD08(handle, channel, value, 0)

terminate_sub:

    call QAPIExtCloseCard(handle)

end sub

```

6.3.4 Read Counter 0 and Counter 1 under VB

```

Function ShowError() As Integer

    Dim buffer As String * 256
    Dim Result As Long

    Result = QAPIGetLastError()

```



```

    If (Result <> 0) Then
        Result = QAPIGetLastErrorStringEx(buffer, Len(buffer))
        MsgBox (Left(buffer & " ", Result))
    Else
        End If
    IsError = (Result <> 0)
End Function

' read the counter
Sub    ButtonCounter()
    Dim nDevice As Long
    Dim handle As Long
    Dim nMode As Long
    Dim channel as long
    Dim DDR as long
    Dim value as long

    ' find first module
    nDevice = 0

    ' Step 1: Open the module
    handle = QAPIExtOpenCard(USBAD8DAC2, nDevice)
    if ( handle = 0 ) then
        MsgBox "Unable to open any USBAD8DAC2 module!"
        exit sub
    end if

    '
    ' Step 2: Initialize the counter
    '
    bEnableHWGate = FALSE

    ' Enable Counter 0 ( D-Sub Pin 17 )
    if ( bEnableHWGate ) then
        '
        ' Counting will only take in place if there is high level on the D-Sub Pin 17 line
        '
        Call QAPIExtSetupCounter(handle, 0, MODE_ENABLE_COUNTER_HW_GATE, 0): If ShowError() Then
        GoTo error_exit
    else
        '
        ' Enable counter
        '
        Call QAPIExtSetupCounter(handle, 0, MODE_DEFAULT, 0): If ShowError() Then GoTo error_exit
    end if

    '
    ' Step 3: Reset the counter to zero
    '
    ' Reset Counter 0
    Call QAPIExtResetCounter(handle, 0, 0, 0): If ShowError() Then GoTo error_exit

    ' Reset Counter 1
        Call QAPIExtResetCounter(handle, 1, 0, 0): If ShowError() Then GoTo error_exit

    ' wait one second
    Sleep(1000)

```

```
'
' Step 4: Read the counter ( connect a signal to the lines D-Sub Pin 17-> Channel 0,
' D-Sub Pin 16 -> Channel 1 and D-Sub Pin 18 -> HW Gate Channel 0
'
' There are two possibilities for reading a counter:
'
' 1.) non-destructive, no autoreset, mode parameter MODE_DEFAULT
'
' 2.) destructive reading, after reading the counter will be reset to zero , mode parameter
' MODE_RESET_COUNTER_ON_READ
'
for i% = 0 to 10
    ' read the counter 0, after reading always check for an error with QAPIGetLastError
    ' ( ) , a common error can be ERROR_QLIB_COUNTER_OVERFLOW
    channel = 0
    counter = QAPIExtReadCounter(handle, channel, MODE_RESET_COUNTER_ON_READ, 0):
    If ShowError() Then GoTo error_exit
    debug.print "Counter 0: " & counter
    ' read the counter 0, after reading always check for an error with QAPIGetLastError
    ' ( ) , a common error can be ERROR_QLIB_COUNTER_OVERFLOW
    channel = 1
    counter = QAPIExtReadCounter(handle, channel, MODE_DEFAULT, 0): If ShowError()
    Then GoTo error_exit
    debug.print "Counter 1: " & counter
    ' wait one second
    Sleep(1000)
next i%

terminate_sub:
    call QAPIExtCloseCard(handle)
end sub
```

7. Annex

7.1 Frequently asked questions (FAQ)

7.1.1 General Information

Are there any known problems when using the module through an USB Hub?



No, but we would recommend using the module directly at one USB Port of your PC.

Do I need external power supply or does my USB port drive enough current to run the module safely?

This depends on the used module. Please refer our homepage for more information on your module or contact our technical support.

Is the module USB 1.1 or USB 2.0?

All QUANCOM USB modules are USB 1.1.

7.1.2 Problems with boards running under Windows 98/95 and Windows 2000/NT



Why is the "Control Panel" board configuration dialog "QLIB" empty?

- There is no QUANCOM PCI board in the system.
- There are no drivers installed for a QUANCOM ISA board.



I get the message "QLIBNDRV.SYS not found" or "QLIBNDRV.VXD not found" after installation. What can I do?

- Check that the QLIB is installed properly. For further information about the installation process and the general programming with the QLIB please see the "QLIB" manual which is included on the installation CD.
- If you use a QUANCOM ISA board check if the drivers for the QUANCOM board are installed.



Why do I get the message "Driver QLIBNDRV.SYS" or "Driver QLIBNDRV.VXD" could not be load?

- Check that the QLIB is installed properly. For further information about the installation process and the general programming with the QLIB please read the "QLIB" manual which is included on the installation CD.
- The driver for the QUANCOM board was not loaded. (Control Panel => System)



Windows 2000/NT: Why do I get the message "Driver could not be installed" during the installation?

- The driver's installation has failed, because the QLIB was not installed with administration-rights.
- QLIB-Software was installed on a network drive. Always install the QLIB on your local drive.



Why do I have to restart the driver after every reboot?

- The starting type of the driver is set to "Manual". If you wanted to you are able change this setting on "Automatic" to start the driver on every reboot of the system.



Windows 2000/NT: How can I manually install the driver QLIBNDRV.SYS?
If the QLIBNDRV.SYS failed to install, it may be necessary to install the driver manually.

Please take the following steps to install the driver manually:

- Search on the installation CD "Tools" for the tool **instdrv.exe** in the directory. With this tool you can install and de-install the driver manually.
- Please call this tool with the following command line parameters:

instdrv libndrv d:\directory\qlibndrv.sys .

(Replace **d:\directory** with the drive, where the driver qlibndrv.sys is located.)

- Go to "Start -> Settings ->Control panel ->(Administrative Tools / Windows 2000 only) -> Drivers" change the start type to "**Automatic**", then click on the "**Start**" button. Please restart the system for the changes to become active.

If you sent your QUANCOM board to us, please use the original package or any other suitable package to protect the contents against transport damage. You also need to send us a copy of the original bill and the RMA number.

You can shorten the repair time by sending us an exact failure description, so that a faster failure search is possible. Send your QUANCOM board directly to the service department of QUANCOM Informations-systeme GmbH.