



Admesy USBTMC library manual





All rights reserved. No part of this document may be reproduced, stored in a database or retrieval system, or published in any form or way, electronically, mechanically, by print, photo print, microfilm or any other means without prior written permission from the publisher.

All correspondence regarding copyrights :

Admesy B.V.

Beneluxstraat 7

6014 CC Ittervoort

The Netherlands

Tel : +31 (0)475 600232

Fax : +31 (0) 45 5213156

URL : <http://www.admesy.nl>

Contents

1	General information	4
2	Installing device driver and device	4
2.1	NI-VISA	4
2.2	LibUSB WIN32	5
2.3	Device installation	5
3	Commands	6
3.1	usbtdmc_get_version	6
3.2	usbtdmc_find_devices	6
3.3	usbtdmc_open	6
3.4	usbtdmc_write	7
3.5	usbtdmc_read	7
3.6	usbtdmc_close	7
4	Example	8
4.1	Example using Import libraries	8
4.2	Example using explicit run-time linking	10

1 General information

Admesy creates devices which are compatible with standards like the USBTMC class for Test & Measurement devices. One of the reasons to do so is the choice of drivers for the end user.

The Admesy USBTMC library functions all use NI-VISA or LibUSB WIN32 for communication through USB. All Admesy devices can be directly controlled by Admesy USBTMC.

Note that regarding any used programming language, the commands send to Admesy devices are all in ASCII format and regardless of the programming language used.

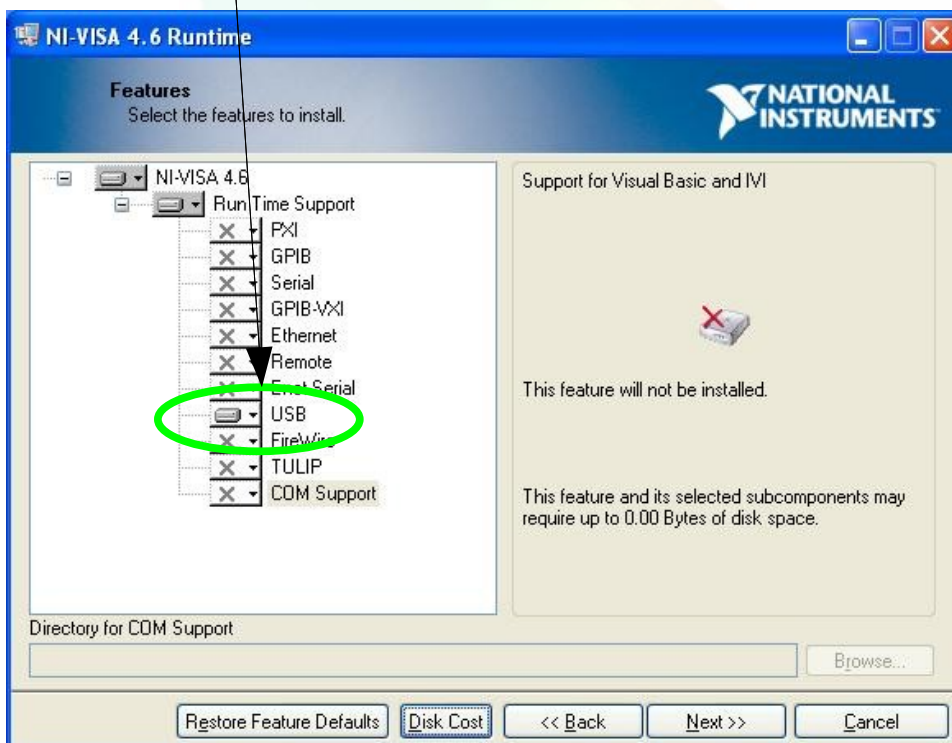
Note: when a function has a return, this will indicate whether there was an error occurred or not. 0 = success

2 Installing device driver and device

To use Admesy USBTMC library NI-VISA or LibUSB WIN32 must be installed. When both are installed please make sure the Admesy device driver is set to NI-VISA, this due to the fact that the Admesy USBTMC library prefers NI-VISA above LibUSB WIN32.

2.1 NI-VISA

[Download NI-VISA runtime](#) and Install the runtime for USB (see below) and follow the steps.



Note: All other modules are not needed to be installed.

2.2 LibUSB WIN32

[Download LibUSB WIN32](#) (current version libusb-win32-filter-bin-0.1.12.2) and Install:

Windows Win98SE, WinME, Win2k, WinXP.

follow the steps, restart and run the test program.

VISTA USERS:

1. Right click on "libusb-win32-filter-bin-x.x.xx.x.exe",
2. Go to Compatibility tab and set **"run this program in compatibility mode for"** select → **windows XP (service pack 2)**,
3. Click on the button → **"Show settings for all users"** → set **"run this program in compatibility mode for" select -> windows XP (service pack 2)**,
4. Now do not forget to set **"Run this program as administrator"** in the privilege level,
5. Click Ok button and Ok button,
6. Right lick on "libusb-win32-filter-bin-x.x.xx.x.exe" and run as administrator,
7. Follow the steps, restart and run the test program.

Note: when not following these steps all USB devices are disconnected and will only will connect when LibUSB is uninstalled. When this problem occurred please start the computer in save mode and uninstall LibUSB via the control panel → Programs and features.

2.3 Device installation

When only NI-VISA installed no need to install an .inf file.

When LibUSB is installed there must be an inf files installed to link the device with LibUSB. This file is provided by Admesy.

When both are installed please make sure to connect the device to the NI-VISA driver via the control panel → device manager → search applicable device → device properties. When not set to NI-VISA driver, update the driver to NI-VISA.

When using the devices on vista no need for an .inf file to be installed, this due to the fact of winUSB. WinUSB will detect the device and the selected driver will communicate via WinUSB to the device.

3 Commands

The following commands can be used to control or retrieve information about the device.

3.1 *usbtmc_get_version*

Program language:

Labview



usbtmc_get_version.vi

This will retrieve the library version number.

C

```
void usbtmc_get_version ( char * version)
```

Function description:

This will retrieve the library version number.

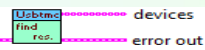
Output:

char * version

3.2 *usbtmc_find_devices*

Program language:

Labview



usbtmc_find_devices.vi

This function will find all USBTMC connected devices.

C

```
int usbtmc_find_devices (char * usbtmcdevices)
```

Function description:

This function will find all USBTMC connected devices.

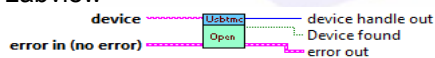
Output:

char * usbtmcdevices

3.3 *usbtmc_open*

Program language:

Labview



usbtmc_open.vi

Open the device by string descriptor:
Usb bus value, vendor id, product id, serial, instr
Example: USB0::0x1781::0x0E95::00006::INSTR

C

```
int usbtmc_open(char * instrDescriptor, unsigned long * handle)
```

Function description:

Open the device by string descriptor: Usb bus value, vendor id, product id, serial, instr
Example: USB0::0x1781::0x0E95::00006::INSTR

Input:

char * instrDescriptor

example: "USB0::0x1781::0x0E95::00006::INSTR"

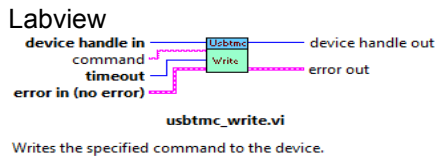
Output:

unsigned long * handle

device handle

3.4 *usbtmc_write*

Program language:



C
int usbtmc_write (unsigned long * handle, char *command, int timeout)

Function description:

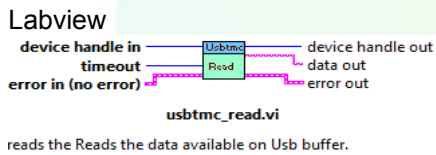
Writes the specified command to the device.

Input:

unsigned long * handle	device handle
char *command	string to write example “:meas:XYZ” , “:set:avg 100”
int timeout	timeout value (ms)

3.5 *usbtc_read*

Program language:



C
int usbtmc_read(unsigned long * handle, char *data, int timeout)

Function description:

Reads the data available on Usb buffer

Input:

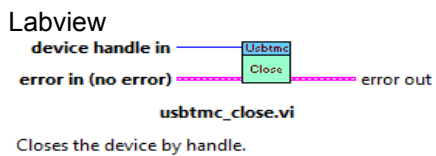
unsigned long * handle	device handle
int timeout	timeout value (ms)

output:

char *data	data from the usb buffer
-------------------	--------------------------

3.6 *usbtc_close*

Program language:



C
int usbtmc_close (unsigned long * handle)

Function description:

Closes the device by handle.

Input:

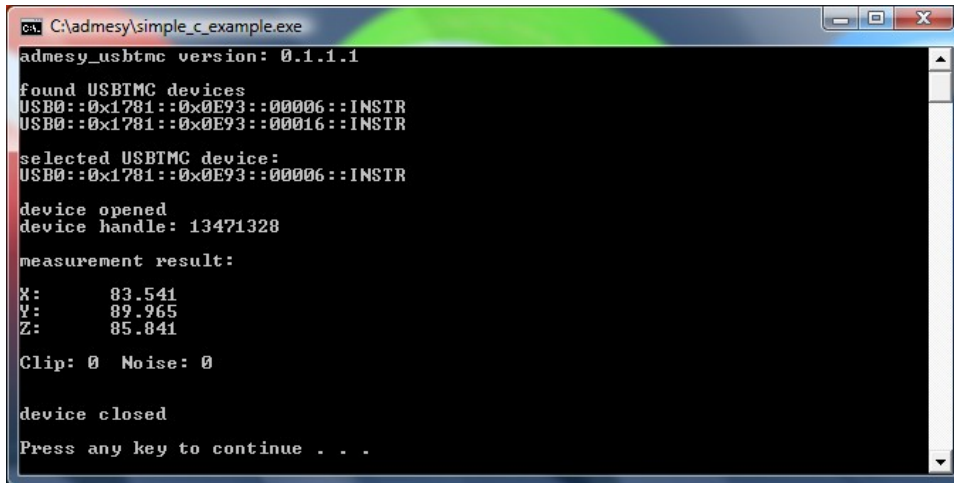
unsigned long * handle	device handle
-------------------------------	---------------

4 Example

In the examples shown below the basics are explained.

4.1 Example using Import libraries

The provided lib is compiled with the GCC compiler the "libadmesy_usbtmp.a" must be included in the linker. When not able to use the .a file, see chapter 4.2 for usage of the admesy_usbtmp.dll. Also include the admesy_usbtmp.h.



```
C:\admesy\simple_c_example.exe
admesy_usbtmp version: 0.1.1.1
found USBTMC devices
USB0::0x1781::0x0E93::00006::INSTR
USB0::0x1781::0x0E93::00016::INSTR
selected USBTMC device:
USB0::0x1781::0x0E93::00006::INSTR
device opened
device handle: 13471328
measurement result:
X:      83.541
Y:      89.965
Z:      85.841
Clip: 0  Noise: 0
device closed
Press any key to continue . . .
```

```
#include <stdio.h>
#include <stdlib.h>
#include <admesy_usbtmp.h> //admesy_usbtmp header file

#define TIMEOUT_USBTMC 5000 //time out value

int main(int argc, char *argv[])
{
    //variables
    char usbtmcdevices[256]="";
    char version[10] = "";
    char meas_result[50] = "";
    unsigned long handle = 0;
    int error = 0;

    //get lib version
    usbtmc_get_version(version);
    printf("admesy_usbtmp version: %s\n", version);
    printf("\n");

    //get devices
    error = usbtmc_find_devices(usbtmcdevices);
    if(error<0)
    {
        printf("Error in usbtmc_find_devices\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    printf("found USBTMC devices \n%s\n", usbtmcdevices);
    printf("\n");

    //select first found device
    strtok(usbtmcdevices, "\n");
    printf("selected USBTMC device: \n%s\n", usbtmcdevices);
    printf("\n");

    //open device
    error = usbtmc_open(usbtmcdevices, &handle);
    if(error<0)
    {
        printf("Error in usbtmc_open\nERROR: %d\n",error);
```

```

        system("PAUSE");
        return 0;
    }
    //see if device handle is valied (not 0)
    if(handle)
    {
        printf("device opened\n");
        printf("device handle: %d\n", handle);
        printf("\n");
    }
    else
    {
        printf("\n");
        printf("No device found\n\nThis program will exit\n");
        system("PAUSE");
        return 0;
    }
    //set averaging
    error = usbtmc_write(&handle, ":sens:aver 1000", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    //set gain
    error = usbtmc_write(&handle, ":sens:gain auto", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    //write measure command
    error = usbtmc_write(&handle, ":meas:XYZ", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    //read the data from the measure command
    error = usbtmc_read(&handle,meas_result,TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_read\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    //all data is ',' seperated
    printf("measurement result:\n\n");
    printf("X:      %s\n", strtok(meas_result, ","));
    printf("Y:      %s\n", strtok(NULL, ","));
    printf("Z:      %s\n", strtok(NULL, ","));
    printf("\n");
    printf("Clip: %s Noise: ", strtok(NULL, ","));
    printf("%s\n", strtok(NULL, ","));
    printf("\n");

    //close the device
    error = usbtmc_close(&handle);
    if(error<0)
    {
        printf("Error in usbtmc_close\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    printf("device closed\n");
    printf("\n");

system("PAUSE");
return 0;
}

```

4.2 Example using explicit run-time linking

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#define TIMEOUT_USBTMC 5000 //time out value

//lib handle
HINSTANCE LibUSBTMC;

int main(int argc, char *argv[])
{
    char usbtmcdevices[256]="";
    char version[10] = "";
    char meas_result[50] = "";
    unsigned long handle = 0;
    int error = 0;

    //load lib and check if loaded
    LibUSBTMC = LoadLibrary("admesy_usbtmc.dll");
    if (LibUSBTMC == NULL)
    {
        printf("Error: admesy_usbtmc.dll not loaded\nThis program will exit\n");
        system("PAUSE");
        return 0;
    }

    //lib functions
    FARPROC get_version, find_devices, Open, Write, Read, Close;
    //var libhandle C functions of admesy_usbtmc.dll
    get_version = GetProcAddress(LibUSBTMC, "usb_tmc_get_version");
    find_devices = GetProcAddress(LibUSBTMC, "usb_tmc_find_devices");
    Open = GetProcAddress(LibUSBTMC, "usb_tmc_open");
    Write = GetProcAddress(LibUSBTMC, "usb_tmc_write");
    Read = GetProcAddress(LibUSBTMC, "usb_tmc_read");
    Close = GetProcAddress(LibUSBTMC, "usb_tmc_close");

    //get lib version
    (*get_version)(version);
    printf("admesy_usbtmc version: %s\n", version);
    printf("\n");

    //find all usb tmc devices devices
    error = (*find_devices)(usbtmcdevices);
    if(error<0)
    {
        printf("Error in usb_tmc_find_devices\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    printf("found USBTMC devices \n%s\n", usbtmcdevices);
    printf("\n");

    //select first found device
    strtok(usbtmcdevices, "\n");
    printf("selected USBTMC device: \n%s\n", usbtmcdevices);
    printf("\n");

    //open device
    error = (*Open)(usbtmcdevices, &handle);
    if(error<0)
    {
        printf("Error in usb_tmc_open\nERROR: %d\n",error);
        system("PAUSE");
        return 0;
    }
    // check if handle is valid (not 0)
    if(handle)
    {
        printf("device opened\n");
    }
}
```

```

        printf("device handle: %d\n", handle);
        printf("\n");
    }
    else
    {
        printf("\n");
        printf("No device found\n\nThis program will exit\n");
        system("PAUSE");
        return 0;
    }
    //set averaging      (avg = 1000)
    error = (*Write)(&handle, ":sens:aver 1000", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n", error);
        system("PAUSE");
        return 0;
    }
    //set gain      (gain = auto)
    error = (*Write)(&handle, ":sens:gain auto", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n", error);
        system("PAUSE");
        return 0;
    }

    //write measure command
    error = (*Write)(&handle, ":meas:XYZ", TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_write\nERROR: %d\n", error);
        system("PAUSE");
        return 0;
    }

    //read the data from the measure command
    error = (*Read)(&handle, meas_result, TIMEOUT_USBTMC);
    if(error<0)
    {
        printf("Error in usbtmc_read\nERROR: %d\n", error);
        system("PAUSE");
        return 0;
    }
    // all data is ',' seperated
    printf("measurement result:\n\n");
    printf("X:      %s\n", strtok(meas_result, ","));
    printf("Y:      %s\n", strtok(NULL, ","));
    printf("Z:      %s\n", strtok(NULL, ","));
    printf("\n");
    printf("Clip: %s Noise: ", strtok(NULL, ","));
    printf("%s\n", strtok(NULL, ","));
    printf("\n");

    //close the device
    error = (*Close)(&handle);
    if(error<0)
    {
        printf("Error in usbtmc_close\nERROR: %d\n", error);
        system("PAUSE");
        return 0;
    }
    printf("device closed\n");
    printf("\n");

    //close the library
    FreeLibrary(LibUSBTMC);

    system("PAUSE");
    return 0;
}

```