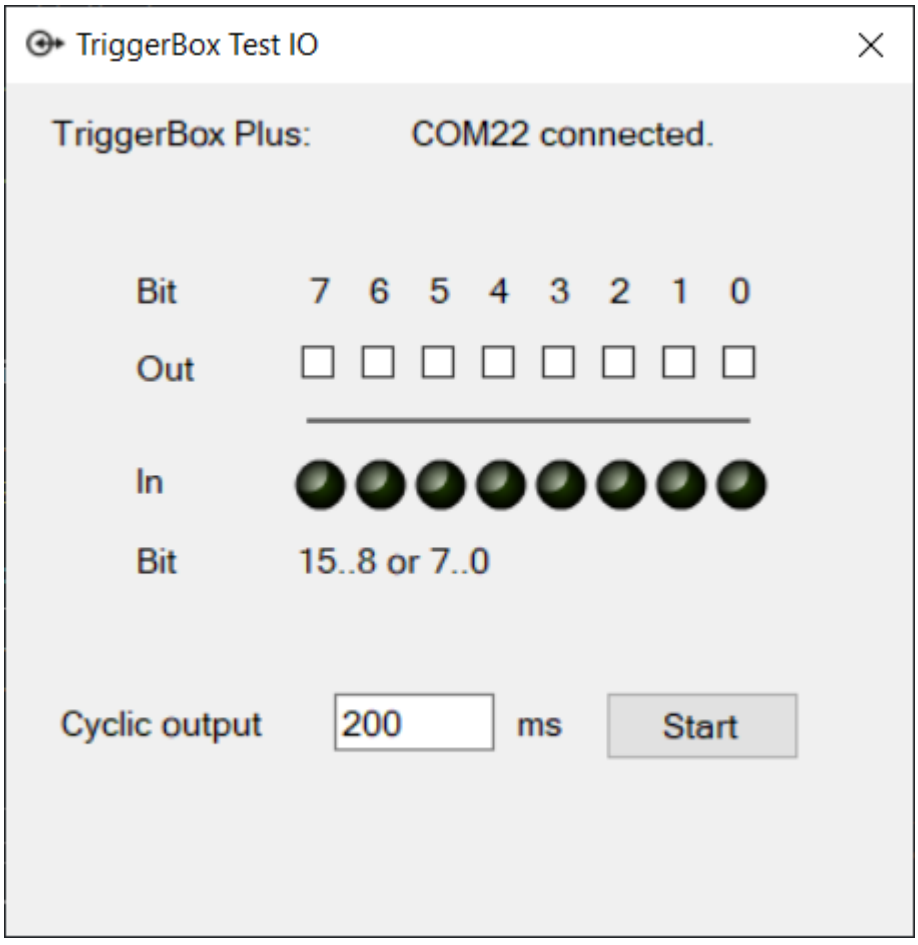
How to use the BrainVision TriggerBox (Plus) application interface

These instructions are only valid for TriggerBox rev. 02 and TriggerBox Plus. There are small differences depending on which of the two shall be controlled. These differences will be pointed out below.

Once the TriggerBox (Plus) software is installed, a serial port communication will be available for various software tools to communicate with the TriggerBox Plus. For TriggerBox rev. 02, a virtual serial port can be used, whereas the TriggerBox Plus has its own serial device driver that works via USB. The COM port can be found in the computer's device manager, or more easily, by connecting the TriggerBox (Plus) and running the TriggerBox Test IO. The TriggerBox Test IO tells you what type of TriggerBox is connected and which COM port it is occupying:



Open the serial port from the application of your choice. For the TriggerBox rev. 02, all settings other than the COM port are irrelevant. You can put any value for all the potential parameters (whether you need to set them at all depends on the software you want to use). For the TriggerBox Plus, the only requirement is to set the baud rate to 2.000.000 b/s. That said, to accommodate both options, simply select the following set of values:

Brain Products GmbH | Zeppelinstrasse 7 | 82205 Gilching, Germany

- Baud rate: 2.000.000 b/s
- Data bits: 8
- Parity: None
- Stop bits: 1
- Flow control: None

Each byte written to the COM port is transmitted to the eight output lines (Bit 0–7) of the TriggerBox (Plus) “Output (to Amp)” connector.

Only for the TriggerBox Plus: At this stage, the current decimal code of these 8 bits is also sent as a LabStreamingLayer (LSL) marker to the local network. Depending on the settings in the Configuration tool, the upper 8 bits are also added in the conversion to a decimal number.

Starting with driver version 1.2.0 the input is event driven. A new byte is available for reading when the state of the TriggerBox “Input (8-15)” lines has changed.

The TriggerBox Plus can read from any input: It is possible to select either bits 0–7 or bits 8–15 in the Configuration tool.

Depending on your setup, you may want to reset the output lines to all zeros or all ones (0x00 or 0xFF in hex code). It may also be necessary to close the port in your software of choice.

Examples

Below, you can find examples for the TriggerBox (Plus) usage in

- C++
- C#
- Python
- MATLAB®
- MATLAB® Psychtoolbox

Note: for all examples you need at least the TriggerBox driver version 1.2.0.

C++ Example

```
#include <windows.h>
#include <iostream>
#include <stdio.h>
#include <tchar.h>

HANDLE hPort;
BOOL   Terminate = FALSE;

// The serial port read thread
ULONG RxThread(void *arg)
{
   BYTE buffer[10000];
   DWORD bytesRead;
   BOOL ok;
   BOOL WaitingOnRead;
   DWORD dwRes;

   try
   {
      OVERLAPPED ovFile = { 0 };
      ovFile.hEvent = CreateEvent(0, true, 0, 0);

      OVERLAPPED ov = { 0 };
      DWORD dwEventMask;
      ov.hEvent = CreateEvent(0, true, 0, 0);
      ok = SetCommMask(hPort, EV_RXCHAR);
      WaitingOnRead = FALSE;

      while (!Terminate)
      {
         if (!WaitingOnRead)
         {
            WaitCommEvent(hPort, &dwEventMask, &ov);
            dwRes = WaitForSingleObject(ov.hEvent, 20);

            // issue a new read request
            ok = ReadFile(hPort, buffer, sizeof(buffer), &bytesRead, &ovFile);
            if (!ok)
            {
               DWORD lastError = ::GetLastError();
               if (lastError != ERROR_IO_PENDING)
               {
                  std::cout << "error " << lastError << " in read thread" << std::endl;
                  Terminate = true;
               }
               else
                  // read operation delayed
                  WaitingOnRead = TRUE;
            }
            else
            {
               // read completed immediately
               if (bytesRead)
               {
                  for (DWORD b = 0; b < bytesRead; b++)
                     std::cout << (int)buffer[b] << std::endl;
               }
            }
         }
         else
         {
            // delayed read operation
            dwRes = WaitForSingleObject(ovFile.hEvent, 20);
            switch (dwRes)
            {
            case WAIT_OBJECT_0:
               // Read completed.
```

```
                if (!GetOverlappedResult(hPort, &ovFile, &bytesRead, FALSE))
                {
                    // Error in communications; report it.
                    DWORD lastError = ::GetLastError();
                    std::cout << "error " << lastError << " in read thread" << std::endl;
                    Terminate = true;
                }
                else
                {
                    // Read completed successfully.
                    if (bytesRead)
                    {
                        for (DWORD b = 0; b < bytesRead; b++)
                            std::cout << (int)buffer[b] << std::endl;
                    }

                    //  Reset flag so that another opertion can be issued.
                    WaitingOnRead = FALSE;
                }
                break;

            case WAIT_TIMEOUT:
                // Operation isn't complete yet.
                // This is a good time to do some background work.
                break;

            default:
                // Error in the WaitForSingleObject; abort.
                // This indicates a problem with the OVERLAPPED structure's
                // event handle.
                DWORD lastError = ::GetLastError();
                std::cout << "error " << lastError << " in read thread" << std::endl;
                Terminate = true;
                break;
            }
        }
    }

    // clean up
    CloseHandle(ovFile.hEvent);
  }
  catch (...)
  {
    std::cout << "exception in read thread" << std::endl;
  }
  ExitThread(0);
  return(0);
}


int _tmain(int argc, _TCHAR* argv[])
{
  BYTE data;
  HANDLE hRcv = 0;
  DWORD  ThreadId = 0;
  int    *ThreadIx = 0;
  OVERLAPPED ov = { 0 };
  ov.hEvent = CreateEvent(0, true, 0, 0);

  // Open the Windows device manager, search for the "TriggerBox VirtualSerial Port" for
  // TriggerBox rev. 02 or for “TriggerBox Plus” for the TriggerBox Plus in "Ports /COM & LPT)"
  // and enter the COM port number as file name.
  // For the usage of serial ports larger than COM9
  // see https://support.microsoft.com/en-us/kb/115831
  hPort = CreateFile(_T("\\\\.\\COM9"),
    GENERIC_WRITE | GENERIC_READ, 0, NULL, OPEN_EXISTING, FILE_FLAG_OVERLAPPED, NULL);
  if (hPort == INVALID_HANDLE_VALUE)
  {
    std::cout << "failed to open COM port" << std::endl;
    return -1;
```

```
    }

    // Entering the baud rate is required for the TriggerBox Plus.
    DCB serialParams = { 0 };
    serialParams.DCBlength = sizeof(serialParams);
    GetCommState(hPort, &serialParams);
    serialParams.BaudRate = 2000000;
    if (!SetCommState(hPort, &serialParams))
    {
        std::cout << "failed to SetCommState" << std::endl;
        return -1;
    }

    // Create and start the read thread
    hRcv = CreateThread(0, 0, (LPTHREAD_START_ROUTINE)&RxThread, &ThreadIx, (DWORD)NULL, &ThreadId);

    // Set the port to an initial state
    data = 0x00;
    WriteFile(hPort, &data, 1, NULL, &ov);
    WaitForSingleObject(ov.hEvent, 20);
    Sleep(10);

    // Set Bit 0, Pin 2 of the Output(to Amp) connector
    data = 0x01;
    WriteFile(hPort, &data, 1, NULL, &ov);
    WaitForSingleObject(ov.hEvent, 20);
    Sleep(10);

    // Reset Bit 0, Pin 2 of the Output(to Amp) connector
    data = 0x00;
    WriteFile(hPort, &data, 1, NULL, &ov);
    WaitForSingleObject(ov.hEvent, 20);
    Sleep(10);

    // Reset the port to its default state
    data = 0xFF;
    WriteFile(hPort, &data, 1, NULL, &ov);
    WaitForSingleObject(ov.hEvent, 20);
    Sleep(10);

    // Terminate the read thread
    Terminate = TRUE;
    WaitForSingleObject(hRcv, (DWORD)2000);
    CloseHandle(hRcv);


    // Close the serial port
    CloseHandle(hPort);

    return 0;
}
```

C# Example

```
using System;
using System.IO.Ports;
using System.Threading;

namespace TriggerBox
{
    class Program
    {
        static void Main(string[] args)
        {
            Byte[] data = { (Byte)0 };

            // Open the Windows device manager, search for the "TriggerBox VirtualSerial Port" for
            // TriggerBox rev. 02 or for “TriggerBox Plus” for the TriggerBox Plus in "Ports /COM & LPT)"
            // and enter the COM port number in the constructor.
            // Entering the baud rate is only required for the TriggerBox Plus.
            SerialPort TriggerBox = new SerialPort("COM6");
            TriggerBox.Open();
            try { TriggerBox.BaudRate = 2000000; } catch { }
            TriggerBox.ReadTimeout = 5000;

            // Attach the data received event handler
            TriggerBox.DataReceived += TriggerBox_DataReceived;

            // Set the port to an initial state
            data[0] = 0x00;
            TriggerBox.Write(data,0,1);
            Thread.Sleep(10);

            // Set Bit 0, Pin 2 of the Output(to Amp) connector
            data[0] = 0x01;
            TriggerBox.Write(data, 0, 1);
            Thread.Sleep(10);

            // Reset Bit 0, Pin 2 of the Output(to Amp) connector
            data[0] = 0x00;
            TriggerBox.Write(data, 0, 1);
            Thread.Sleep(10);

            // Reset the port to its default state
            data[0] = 0xFF;
            TriggerBox.Write(data, 0, 1);
            Thread.Sleep(200);

            // Close the serial port
            TriggerBox.DataReceived -= TriggerBox_DataReceived;
            TriggerBox.Close();
        }

        static void TriggerBox_DataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            SerialPort sp = (SerialPort)sender;
            // get all available bytes from the input buffer
            while (sp.BytesToRead > 0)
            {
                Console.WriteLine(sp.ReadByte());
            }
        }
    }
}
```

Python Example

```
import serial
import time
import threading

Connected = True
PulseWidth = 0.1

# Here, a separate thread for reading from the TriggerBox (Plus) is defined. This is only required if you
want to read data from sources connected to the TriggerBox (Plus).
def ReadThread(port):
    while Connected:
        if port.inWaiting() > 0:
            print ("0x%X"%ord(port.read(1)))


# Open the Windows device manager, search for the "TriggerBox VirtualSerial Port" for TriggerBox rev. 02
or for “TriggerBox Plus” for the TriggerBox Plus in "Ports /COM & LPT)" and enter the COM port number in
the constructor. Entering the baud rate is only required for the TriggerBox Plus.
port = serial.Serial("COM6",baudrate=2000000)

# Start the (optional) read thread. For the TriggerBox Plus, this can be bits 0-7 or 8-15; for TriggerBox
rev. 02 it can only be inputs connected to In 8-15.
thread = threading.Thread(target=ReadThread, args=(port,))
thread.start()

# Set the port to an initial state (‘0x00’ in hex code)
port.write([0x00])
time.sleep(PulseWidth)

# Send the value 1 to the Amp out connector (in hex code)
port.write([0x01])
time.sleep(PulseWidth)

# Reset to 0
port.write([0x00])
time.sleep(PulseWidth)

# It is also possible to use decimal numbers directly
port.write([42])
time.sleep(PulseWidth)

port.write([0])
time.sleep(PulseWidth)

# Or binary code:
port.write([0b00101010])
time.sleep(PulseWidth)

port.write([0b00000000])
time.sleep(PulseWidth)

# Optional: reset the port to its default state
port.write([0xFF])
time.sleep(PulseWidth)

# Terminate the read thread
Connected = False
thread.join(1.0)

# Close the serial port
port.close()
```

MATLAB® Example

```
function TriggerBox()
    PulseWidth = 0.1;
    % Open the Windows device manager, search for the "TriggerBox VirtualSerial Port"
    % for TriggerBox rev. 02 or for “TriggerBox Plus” for the TriggerBox Plus
    % in "Ports /COM & LPT)" and enter the COM port number in the constructor.
    % Entering the baud rate is only required for the TriggerBox Plus.
    port = serialport('COM22', 2000000);

    % To read from the serial port, you can use a callback function:
    % In this example, the function "ReadTriggerBox" is called, whenever
    % there is at least 1 byte of data available:
    configureCallback(port, "byte", 1, @ReadTriggerBox)

    % Set the port to zero (all bits to low):
    write(port, 0, 'uint8');
    pause(PulseWidth);

    % Send the value 1 to the Amp out connector:
    write(port, 1, 'uint8');
    pause(PulseWidth);

    % Send the value 42 to the Amp out connector:
    write(port, 42, 'uint8');
    pause(PulseWidth);

    % Set the port to zero (all bits to low):
    write(port, 0, 'uint8');
    pause(PulseWidth);

    % Optional: reset the port to its default state (all bits to high):
    write(port, 255, 'uint8');
    pause(PulseWidth);

% Callback function that reads in the background (In 8-15 for TriggerBox rev. 02
% or bits 0-7 or bits 8-15 for TriggerBox Plus)
function ReadTriggerBox(src,evt)
    if src.NumBytesAvailable > 0
        data = read(src,src.NumBytesAvailable,'uint8');
        disp(['Data read from TriggerBox (Plus):',num2str(data)])
    end

% Close the serial port:
clear port
```


Brain Products GmbH | Zeppelinstrasse 7 | 82205 Gilching, Germany

MATLAB® Psychtoolbox Example

```
% Open the Windows device manager, search for the "TriggerBox VirtualSerial Port"
% for TriggerBox rev. 02 or for “TriggerBox Plus” for the TriggerBox Plus
% in "Ports /COM & LPT)" and enter the COM port number in the constructor.
% Entering the baud rate is only required for the TriggerBox Plus.
% When using Psychtoolbox, IOPort is an alternative to serialport:
TB = IOPort('OpenSerialPort', 'COM22','BaudRate=2000000');
PulseWidth = 0.1;

% Read data from the TriggerBox
Available = IOPort('BytesAvailable', TB);
if(Available > 0)
    disp(IOPort('Read', TB, 0, Available));
end

% Set the port to zero (all bits to low):
IOPort('Write', TB, uint8(0), 0);
pause(PulseWidth);
% Read data from the TriggerBox
Available = IOPort('BytesAvailable', TB);
if(Available > 0)
    disp(IOPort('Read', TB, 0, Available));
end

% Send the value 1 to the Amp out connector:
IOPort('Write', TB, uint8(1), 0);
pause(PulseWidth);
% Read data from the TriggerBox
Available = IOPort('BytesAvailable', TB);
if(Available > 0)
    disp(IOPort('Read', TB, 0, Available));
end

% Set the port to zero (all bits to low):
IOPort('Write', TB, uint8(0), 0);
pause(PulseWidth);
% Read data from the TriggerBox
Available = IOPort('BytesAvailable', TB);
if(Available > 0)
    disp(IOPort('Read', TB, 0, Available));
end

% Optional: reset the port to its default state (all bits to high):
IOPort('Write', TB, uint8(255), 0);
pause(PulseWidth);
% Read data from the TriggerBox
Available = IOPort('BytesAvailable', TB);
if(Available > 0)
    disp(IOPort('Read', TB, 0, Available));
end

% Then disconnect/close the serial port object from the serial port
IOPort('Close', TB);
```