Teraflash OEM Remote Data Acquisition v22.1

Time-domain measurement data can be accessed remotely via TCP/IPv4. Measurement data means pulse data (time and magnitude) displayed in the upper graph of the Teraflash Host software. If a reference measurement is present, this data is also transmitted (async mode only).

To access these data initialize a TCP connection to the Host-IP on port 6006 (asynchronous mode) or 6007 (synchronous mode). The host immediately starts the data transfer, i.e. the host sends two TCP packages: the first one (fixed length of 6 byte) contains the byte count for the second package. The second package contains the pulse data. As long as the TCP connection is open the data stream sends two TCP packages continuously

every 250ms if connected to port 6006 (asynchronous mode) or
synchronous to the hardware data acquisition if connected to port 6007.

To end the transfer close the TCP connection on the client-side.

Data-format

first TCP package	
type:	String
byte length	6
content	pulse data length [integer]
example	014049

second TCP package	
type	String, lines separated by *CRLF*
byte length	value of first package
content	pulse data [float csv]
example: (first 3 lines)	Time/ps, Signal/nA,Reference/nA 850.000,-0.027478,-0.020780 850.050,0.030787,-0.029909

Basic programming instructions

```
port_async = 6006
port_sync   = 6007
open TCP (server IP, port_sync)
while true // continuous sweeping; for single acquisition: remove while loop
        length = read TCP (bytecount: 6)
        data    = read TCP (bytecount: length)
end while
close TCP (server IP, port_sync)
```

Basic python example

```
// acquire a pulse trace in synchronous mode
import socket
import numpy as np
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('127.0.0.1', 6007))
length_data = sock.recv(6)
pulsedata = sock.recv(int(length_data.decode("utf-8")))
sock.close()

// convert received string to header[String], time[float], magnitude[float]
pulsedata = pulsedata.decode("utf-8").split('\r\n')
header = pulsedata.pop(0)
time = np.zeros(len(pulsedata)-1, dtype = float)
magnitude = np.zeros(len(pulsedata)-1, dtype = float)
for x in range(0,len(pulsedata)-1):
    time[x] = (float(pulsedata[x].split(',')[0]))
    magnitude[x] = (float(pulsedata[x].split(',')[1]))
```