Jaewon's blog

How to import TDT Tank into MATLAB

Posted on Oct 4, 2010

1. Introduction

Tucker-Davis Technologies (TDT) is a company supplying signal processing systems. The signals recorded by their System 3 are stored in a database called TTank and can be read out later. TDT provides application programming interfaces (APIs) for TTank access through ActiveX DLLs and, in any development environment where ActiveX can be incorporated, users can make TTank applications easily.

For example, in MATLAB, you can create a TTank object with the following command. (You need to install OpenDeveloper package before typing the command.)

```
1 TTX = actxcontrol('TTank.X');
```

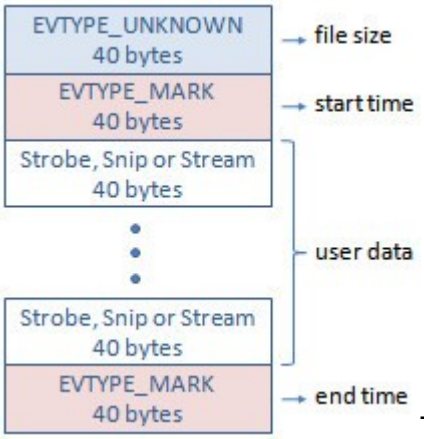
Then you can call any API functions listed in the OpenDeveloper reference manual with the object. (Some code examples are installed with the OpenDeveloper package in C:\TDT\OpenEx\Examples\TTankX_Example\Matlab) If you do not have a TDT system and just want to read TTank, then you can try out TDT NeuroShare Kit. I have not tested it , but I guess it works in a similar way.

However, sometimes it is not convenient to use particular DLLs to access data, since it means that you are bound to the MS Windows platform and cannot read your data without the TDT software package. Considering that you never know how long you will be able to get necessary technical support from the vendor, it is probably not a bad idea to learn how to read the binary format of the TTank directly.

2. Structure of TTank

A TTank can have many blocks. Each block contains data from one continuous recording and consists of 4 files: *.TBK, *.TDX, *.TEV, *.TSQ. Among them, it is TEV and TSQ that contain real data. The other two are used for an indexing purpose.

WARNING: Although TBK and TDK may not be absolutely necessary to retrieve data, most of TDT software becomes unable to read TTanks if those index files do not match with the records in TEV and TSQ. Therefore, DO NOT try to modify the contents of TTanks if you want to use TDT software (e.g., OpenSorter) on them. Rebuilding the index files requires technical support from TDT.



The TSQ file is a heap of 40-byte blocks called event headers (see the right figure). The first two and the last headers are special. The very first one is a EVTYPE_UNKNOWN(0x00000000) type and its size member has the total size of the TSQ file. The second and the last ones are a EVTYPE_MARK(0x00008801) type, and they have the start and the end time of the recording in their timestamp member, respectively. The other headers are all recorded user data and strictly ordered by time.

The structure of the event header is shown below. Those who are not familiar with C/C++ can refer to this variable size information: *long* and *float* (4 bytes), *unsigned short* (2 bytes), *double* and *__int64* (8 bytes).

TSQ event header

```
struct tsqEventHeader
{
   long          size;    // the length of this record in long(4 bytes)
   long          type;    // event type
   long          name;    // store ID, 4 chars cast to long
   unsigned short chan;    // channel
   unsigned short sortcode;
   double        timestamp;
   union
   {
      __int64    fp_loc;  // file pointer location in TEV where A/D samples reside
      double     strobe;
   };
   long format;
   float frequency;
};

// type
#define EVTYPE_UNKNOWN 0x00000000
#define EVTYPE_STRON 0x00000101 // Strobe+, external event codes
#define EVTYPE_STROFF 0x00000102 // Strobe-
#define EVTYPE_SCALAR 0x00000201
#define EVTYPE_STREAM 0x00008101 // Stream, local field potentials
#define EVTYPE_SNIP 0x00008201 // Snip, sorted waveforms and their
timestamps
#define EVTYPE_MARK 0x00008801
#define EVTYPE_HASDATA 0x00008000

// format
#define DFORM_FLOAT 0 // 4 bytes
#define DFORM_LONG 1 // 4 bytes
#define DFORM_SHORT 2 // 2 bytes
#define DFORM_BYTE 3 // 1 byte
#define DFORM_DOUBLE 4
```

Note that some structure members may not have a meaning in some event types. For example, **fp_loc** is valid only for the snip- and the stream-type event headers, because only those types have digitized samples which are separately stored in the TEV file. **fp_loc** indicates the file pointer location in TEV where the sample data points begin. In other event types, **fp_loc** is filled with meaningless numbers or is used for storing strobed codes. The table below summarizes this relationship.

Value or validness of the members in TSQ header (O:valid, X:invalid)

Member	Event type			Note
	Strobe	Snip	Stream	
size	10	10 + (length of A/D data in *long*)		This member is represented in the unit of *long* (4 bytes), so its value is basically 10(=40 bytes; size of the eventheader). If it is larger than 10, the difference indicates the number of A/D samples acquired for this event header. Use the formula in the second row to calculate the
(# of A/D samples)*	N/A	(**size**-10) * [4 / (byte-size of **format**)]		

				number of samples.
channel	X	O	O	The strobe type is an external event and does not have a channel number.
sortcode	X	O	X	Only the snip type does spike-sorting.
fp_loc	X	O	O	These two members share the same memory space, therefore only one of them is valid at a time.
strobe	O	X	X	
* For example, if the value of the **size** is 42 and the **format** is *short*(=2 bytes), then the number of A/D samples in TEV is (42-10) * (4/2) = 64. For the snip type (sorted waveforms), the **format** is *float* whose length is equal to that of *long*, so the number of samples simply becomes **size-**10.				

For the stream type, sample values stored in TEV are voltages, if its format is float. However, if the format is one of the integer types (long, short or byte), raw sample values should be divided by the scaling factor, to be converted into voltages. The scaling factor is a customizable value specified in the data-saving macro. So you should refer to your RPvds circuit, to find this number.

3. tdtread.m: a MATLAB script reading TTank

The script shown below was tested with TDT System 3 Software v72 and OpenEx 2.12. However, using this script is at your own risk. I do not guarantee the accuracy of the results that you will get by using it. It may not work in the future if the binary format of TTank changes. (Note: It works fine with the latest v76 TDT software. 4/23/2013)

Before running the following code, you need to set the values of 4 variables on your own: **tev_path**, **tsq_path**, **store_id1**, and **store_id2**. The first two are the file paths to TEV and TSQ files, and the last two are the Store IDs of the strobed data and the stream data (e.g., ‘Evnt’ and ‘LFPs’), respectively.

tdtread.m (updated: 7/1/2013)

```
   % tev_path = 'path to TEV';
1  % tsq_path = 'path to TSQ';
2  % store_id1 = 'Evnt';  % this is just an example
3  % store_id2 = 'LFPs';  % this is just an example
4
5  % open the files
6  tev = fopen(tev_path);
7  tsq = fopen(tsq_path); fseek(tsq, 0, 'eof'); ntsq = ftell(tsq)/40; fseek(tsq, 0, 'bof');
8
9  % read from tsq
10 data.size      = fread(tsq, [ntsq 1], 'int32', 36); fseek(tsq, 4, 'bof');
11 data.type      = fread(tsq, [ntsq 1], 'int32', 36); fseek(tsq, 8, 'bof');
12 data.name      = fread(tsq, [ntsq 1], 'uint32', 36); fseek(tsq, 12, 'bof');
13 data.chan      = fread(tsq, [ntsq 1], 'ushort', 38); fseek(tsq, 14, 'bof');
14 data.sortcode  = fread(tsq, [ntsq 1], 'ushort', 38); fseek(tsq, 16, 'bof');
15 data.timestamp = fread(tsq, [ntsq 1], 'double', 32); fseek(tsq, 24, 'bof');
16 data.fp_loc    = fread(tsq, [ntsq 1], 'int64',  32); fseek(tsq, 24, 'bof');
17 data.strobe    = fread(tsq, [ntsq 1], 'double', 32); fseek(tsq, 32, 'bof');
18 data.format    = fread(tsq, [ntsq 1], 'int32',  36); fseek(tsq, 36, 'bof');
19 data.frequency = fread(tsq, [ntsq 1], 'float',  36);
20
21 % change the unit of timestamps from sec to millisec
22 data.timestamp(3:end-1) = (data.timestamp(3:end-1) - data.timestamp(2)) *
23 1000;
24
25 % typecast Store ID (such as 'Evnt', 'eNeu', and 'LPFs') to number
26 name = 256.^(0:3)*double(store_id1)';
27
28 % select tsq headers by the Store ID
29 row = (name == data.name);
30
31 % an example of retrieving strobed events
32 EVENTCODE = [data.timestamp(row) data.strobe(row)];
33
34 % an example of reading A/D samples from tev. You can use the same code to
35 read
36 % the snip-type data (sorted waveforms). Just replace the store ID.
37 table = { 'float',  1, 'float';
38         'long',   1, 'int32';
39         'short',  2, 'short';
40         'byte',   4, 'schar'; }; % a look-up table
41 name = 256.^(0:3)*double(store_id2)';
42 row = (name == data.name);
43 first_row = find(1==row,1);
44 format    = data.format(first_row)+1; % from 0-based to 1-based
45
46 LFP.format        = table{format,1};
47 LFP.sampling_rate = data.frequency(first_row);
48 LFP.chan_info     = [data.timestamp(row) data.chan(row)];
49 % For the snip type, you may want the sortcode additionally.
50 % SPIKE.chan_info = [data.timestamp(row) data.chan(row) data.sortcode(row)];
51
52 fp_loc = data.fp_loc(row);
53 nsample = (data.size(row)-10) * table{format,2};
54 LFP.sample_point = NaN(length(fp_loc),max(nsample));
55 for n=1:length(fp_loc)
56     fseek(tev,fp_loc(n),'bof');
```

- Categories: Computer
- Tags: MATLAB, TDT