libTDC >= v1.10 TTM binary format

The binary file is composed by an header, followed by the measures. Numbers are encoed in little endian format.

1. Header format

The header is composed by 10 64-bit words:

Word index	Data type	Description	Example
0	uint64_t	Magic number (constant)	0x e2 8c 9a f0 9f 8c b5 69
1	uint64_t	Header length (in words)	0x 0a 00 00 00 00 00 00 00 = 10
2	uint64_t	Acquistion date and time, unix timestamp in milliseconds	0x 39 6f 01 1c 82 01 00 00 = 2022-07-20 16:27:12 CEST
3	uint64_t	File index (zero based)	0x 00 00 00 00 00 00 00 00
4	uint64_t	*TDC_period* [fs] (Used for LSB calc)	0x 00 9f 24 00 00 00 00 00
5	uint64_t	*LSB_conv_fact_a* (Used for LSB calc)	0x 00 00 00 00 00 00 00 00
6	uint64_t	*LSB_conv_fact_b* (Used for LSB calc)	0x 10 00 00 00 00 00 00 00
7	uint64_t	Total number of TDC channels	0x 11 00 00 00 00 00 00 00
8	uint64_t	Last file. This field is equal to zero if the current file is not the not the last one of the acquistion	0x 01 00 00 00 00 00 00 00
9	uint64_t	If the current file is not the last one, this field is equal to 2**64-1. Otherwise, in this field is stored the number events lost due to bandwidth limitation	0x 00 00 00 00 00 00 00 00 = no loss 0x ff ff ff ff ff ff ff ff = not the last file

1.1 LSB Calculation

The LSB is calculateted using *TDC_period*, *LSB_conv_fact_a*, *LSB_conv_fact_b*

1. If *LSB_conv_fact_a* is equal to zero, the LSB is calculated as:

   LSB = (*TDC_period* / 2** *LSB_conv_fact_b*) fs

2. If *LSB_conv_fact_a* is not zero, the LSB is calculated as:

   LSB = (*TDC_period* / 2** *LSB_conv_fact_b*) * (2**64 / *LSB_conv_fact_a*) fs

2. Measurement format

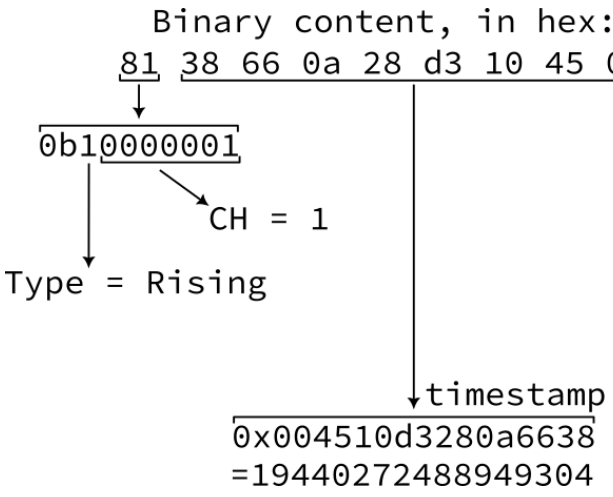
After the header, measurements are stored using 9 bytes (72 bits) for each measure.

The first byte identifies the channel and the type (rising or falling). Most significant bit of the first byte is:

- *1* if event refers to rising event
- *0* if event refers to falling event

The remaining 7 bits of the most significant byte are the channel to which event refers

The 8 less significant bytes are the unsigned 64-bit integer value of the timestamp, expressed in LSB, little-endian encoded:



3. MATLAB import example

The following reads a binary TTM file named 'out.bin' into a Nx3 uint64 matrix called "events_data". It also creates a variable named LSB that contains the LSB value (double).

The three column of the matrix are, in order:

1. TDC channel
2. TDC event type (1 = Rise / 0 = Fall)
3. Timestamps (expressed in LSBs!)

To ensure maximum resolution, the conversion to seconds (i.e. multiplication by LSB) should be performed as last operation (e.g., after the subtraction of two timestamps). Remember to convert the timestamp to double BEFORE multiplying it by LSB.

This script is not performance optimized and can import data at about 1.2 million events per seconds on an average windows machine.

```
clear

% -- Constants --
MAGIC = uint64(7617148963331411170);

% -- File open --
filename = 'out.bin';
fileID = fopen(filename,'r');
s = dir(filename);
filesize = s.bytes;

% -- Read important header fields
if fread(fileID,1,'uint64=>uint64') ~= MAGIC
    error('Invalid input file')
end
header_len = fread(fileID,1,'uint64=>uint64');

% -- Read other header fields
fseek(fileID, 0, 'bof');
header = fread(fileID,header_len ,'uint64=>uint64');

LSB = double(header(5))  * 1e-15 / double(2^header(7));
if header(6) ~= 0
    LSB = LSB * (double(2^64) / double(header(6)));
end

% -- Read all events
events_data = zeros((filesize - header_len * 8) / 9, 3, 'uint64');  % preallocate

% -- Read channel data
fseek(fileID, header_len * 8, 'bof');
events_data(:,1) = fread(fileID,'uint8=>uint64', 8);

% -- Read timestamps data
fseek(fileID, header_len * 8 + 1, 'bof');
events_data(:,3) = fread(fileID,'uint64=>uint64', 1);

% -- Separate rise/fall from channel number
events_data(:,2) = bitshift(events_data(:,1), -7);
```

```
events_data(:,1) = bitand(events_data(:,1), uint64(0b01111111));

if header(9) == 0
    fprintf("Read %d events\n", size(events_data, 1), header(10))
else %if this is the last file, check for lost events
    fprintf("Read %d events (lost %d events)\n", size(events_data, 1), header(10))
end

display('I will now print the first 10 lines/events of the variable "events_data"')

events_data(1:10,:)

display('The first column is the TDC channel,')
display('the second one is the event type (1 = RISING, 0 = FALLING),')
display('the third one is the timestamp expressed in LSBs')

LSB

fprintf("The second event happened %f seconds after the board startup\n", double(events_data(2,3)) * LSB)

fclose(fileID);
clear fileID
```