Deuteron Technologies Ltd
Event File Viewer 9.0

Author: Stefanie Glowinsky

Date: 04/24/2023

Contents

1 Introduction

Deuteron loggers can produce files in two different formats; flat file format and block file format. In flat file format, data files (with extensions ".DT2", ".DT4", ".DT6", ".DT8", or ".DAT") store continuously measured (e.g. neural, audio, motion sensor) data, whereas the special event log file (".NLE") stores one-time events which contain metadata (information about the data), and timestamp and description of each event that occurred. In block file format, events are stored in files alongside continuously recorded data. These files are called "AAAAXXXX.DF1" where AAAA is a 4 letter prefix, and XXXX is the chronological index of the file starting at 0000. Events occurring outside of a recording are stored in files that contain events exclusively, called EVENTXXX.DF1, where XXX is the 3 digit index of the file, starting with 000.

Events can signify many things such as the beginning or end of a recording, LED lights turning on/off, or a stimulus being delivered to the animal. The Event File Reader 9.0 is a tool for parsing, viewing, and saving the event data recorded by data loggers in a convenient and easy to use manner. It can be launched by clicking on the executable to open the user interface, or from the MATLAB command line which can be used to either open the user interface or save the data directly to a CSV file.

Furthermore, information can be retrieved from an event log file using the Event_File_Reader_9_0.dll. Examples showing how to use the DLL are available in Matlab and Python.

2 Getting started

2.1 Setup

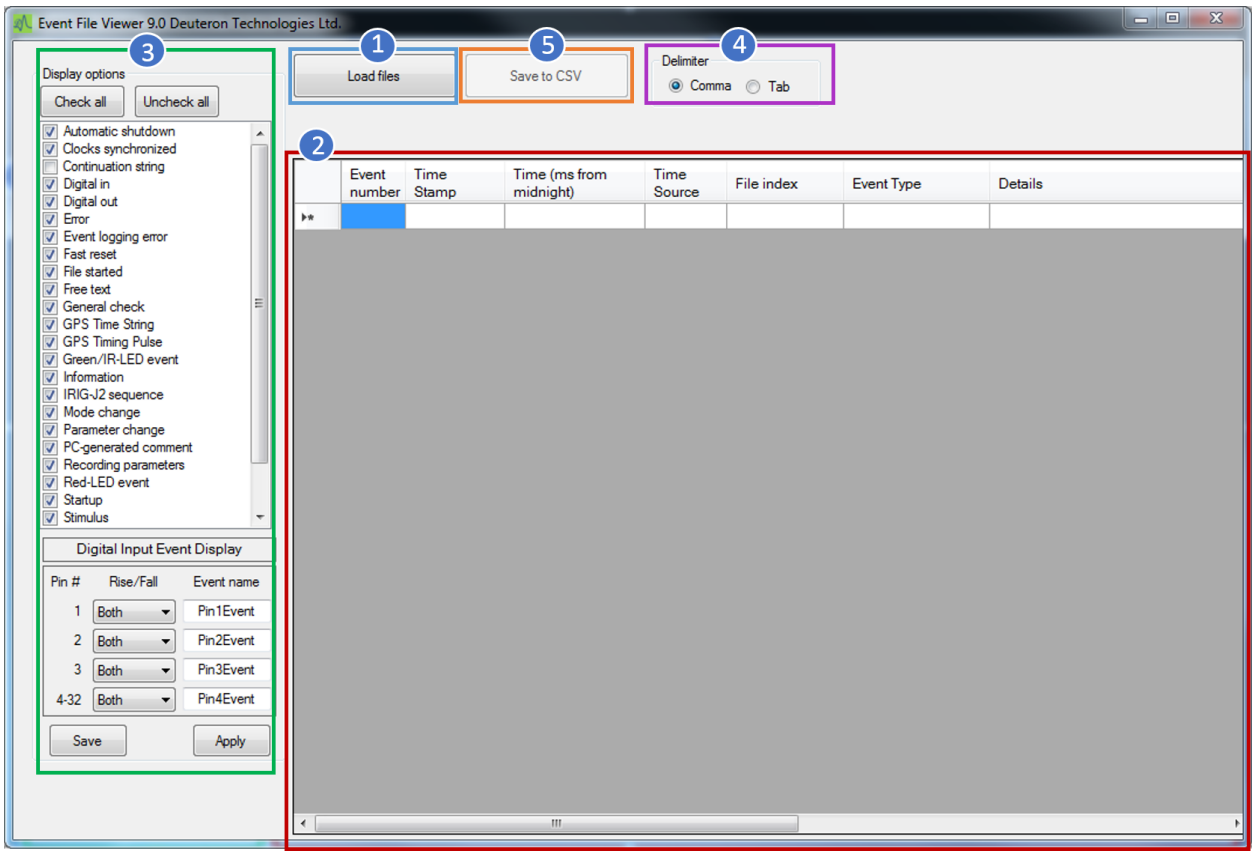
Place the executable and the Settings.ini file in the same directory. Do not directly modify the Settings.ini file or change its name, as this may corrupt it or render the program unable to find it, preventing the program from running correctly.

2.2 Running Event File Reader 9.0

Double clicking on the executable will launch the user interface without initially displaying any files. The user can then choose one (flat format) or many (block format) files to view using the interface. You can also open single event log files directly with the event file viewer by setting Event_File_Reader_9_0 as the default program for opening it and then double clicking on the desired event log file.

3 User Interface

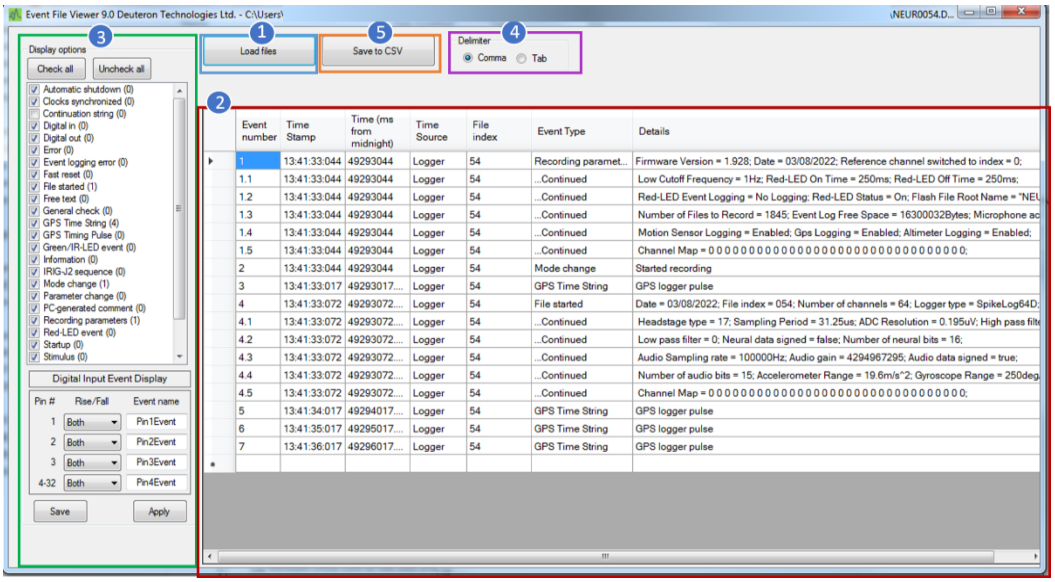
3.1 User Interface overview



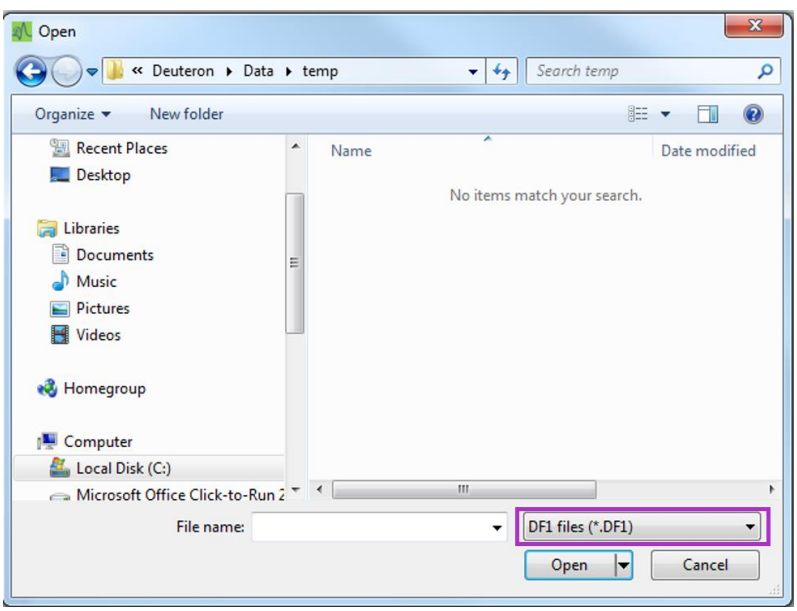
Above is a first look at the GUI before loading an event file. Each of the following controls will be described in greater detail in the following sections:

1. Load file(s)
2. The data table where the event log file data are displayed
3. Controls to choose which kinds of events should be displayed and/or saved to the CSV file
4. Which delimiter should be used for value separation when writing to a CSV file
5. Save data to CSV. Button becomes enabled when a file is loaded.

3.2 Loading Event Log File(s)



Clicking on the "Load files" button (the button labeled 1 in the above image) will open a file dialog (see below) that allows the user to browse for file(s) containing events to load.



If you wish to search for files with the extension ".NLE" (i.e. flat file format), click the dropdown box of the file browser (shown in pink in the figure above) and select the NLE files option.

Upon clicking "OK" after selecting a set of files, the events from those files will immediately be displayed in the table (labelled 2 in the screenshot of the main GUI). Once the files have been loaded into the program, the "Save to CSV" button (labelled 5) will become enabled so the user can save the data from these files.

3.3 Data table

The data table is the table displaying information about the events found in the chosen files (labeled 2 in the figure showing the main GUI). For files in the flat file format, some general information about the file such as the firmware version and the date and time of the creation of the file, will appear as text just above the data table. For block file format, this information will be contained in the events in the table. The data table itself consists of 6 columns which each address different aspects of the event:

Event number – the index of the event in the order in which the events were recorded to the file. When an event traverses multiple lines, the event number resorts to decimals. In the above example, the Recording parameters event (Event number 1) continues on for 5 additional lines which are labeled 1.1 – 1.5.

Time Stamp – gives the time stamp of the event as HH:MM:SS.mmm.

Time (ms from midnight) – the time stamp of the event as milliseconds from midnight. Some event types are measured with sub microsecond resolution while others have only millisecond resolution.

Time Source – If the time of this event was measured according to the logger, it will display “Logger” (in the case of millisecond resolution) or “Logger (Fine)” (sub microsecond resolution). Similarly, if the time of the event was measured according to the transceiver, it will display “Transceiver” (in the case of millisecond resolution) or “Transceiver (Fine)” (microsecond resolution).

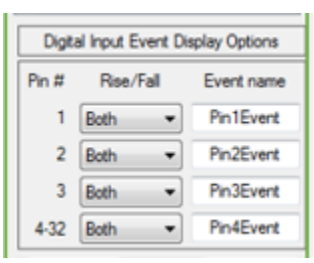
Event Type – there are many different kinds of events. For a description of each type of event, refer to Appendix A.

Detail – description of what has actually occurred in the given event.

3.4 Display control panel

The display control panel on the left hand side allows the user to choose which types of events should be displayed in the table and/or written to the CSV file.

The checklist contains a list of event types (in the section labeled 3 in the GUI screenshot), and each event type name is followed by the number of times it occurs in the current file in brackets. The user can check or uncheck these options individually, or as a group using the check/uncheck all buttons.



The “Digital Input Event Display Options” section (above figure) allows the user to control in greater detail which digital input events should be displayed and how. For each pin (1, 2, 3, or 4+), the user can select from the drop down options whether they would like to display events signifying a digital rising edge, falling edge, both, or neither. The user can also use the text boxes in the “Event name” column to enter a custom name for the events occurring on each pin.

In order for the user's chosen settings to be reflected in the Data table, the user must click "Apply" in the bottom right corner of the display control panel. The "Save" button in the bottom left corner of the display control panel will save the current settings and automatically apply them next time the Event File Reader program is run.

Note: Whereas the "Digital Input Event Display Options" allows the user to choose which pin events to display by pin number and rise/fall status, the "Digital in" checkbox will display, if available, the digital input port status and the digital input event status which is simply an alternative mode of displaying the digital input events. For more information on the different ways of representing the digital input events and how to interpret them, see Appendix A.

3.5 Delimiter Control

The delimiter (labelled 4) control allows the user to choose the type of delimiter that will be used to separate values in the CSV file to which the data are saved. The options are either comma delimited or tab delimited.

3.6 Save to CSV

Clicking on the "Save to CSV" button (labelled 5) will open up a save file dialog, allowing the user to choose a location and name for the CSV file that will be created. Only events that are marked to be displayed will be recorded in the CSV file. To record all events, ensure that you first click "check all" and "apply" in the display options tab.

4 Accessing event log files via DLL

There are 2 DLLs used to access data in an event log file: Event_File_Reader_9_0.dll (henceforth referred to as the main DLL) and Event_File_Reader_Dll_UM.dll which is a wrapper for the main DLL (henceforth referred to as the wrapper DLL). In Matlab, the user will interact directly with the main DLL but in Python the user will interact via the wrapper DLL.

4.1 Matlab

Users can directly interact with the main DLL from Matlab using the code contained in the Event_File_Reader_9_0_Matlab_Example.zip file. The main script for the example is EventFileReaderDllExampleScript.m. The zip file should also contain the dll (Event_File_Reader_9_0.dll) and the Settings.ini file.

Before running the script, the user should modify the following values in the EventFileReaderDllExampleScript .m file:

eventFileReaderDll	The full path of the folder in which the Event_File_Reader_9_0.dll is located
filesetType	Set this to 'Block' for block file format and 'Flat' for flat file format
folderName	the name of the folder where your files to process are located.

If you are using block file format:

filePrefix	The four letter prefix of the data files (e.g. 'NEUR')
minFileIndex	The file index of the first file to process (e.g. for NEUR0001.DF1, this equals 1)

maxFileIndex	The file index of the last file to process (e.g. for NEUR0003.DF1, this equals 3)
IncludeEventFile	If to include the EVENT000.DF1 file, 1 to include, 0 to exclude

In flat file format:

EventFileName	Set this to the name of the event file e.g. 'EventLog.NLE'
IncludeEventFile	Do not modify this value; it should be false.

The example script does the following:

1. Initializes the variables in set by the user
2. Loads the DLL
   For each file:
3. Loads and parses the event data from the file using the DLL.
4. Check if loading in the file was successful. An error code of 0 indicates the files were loaded successfully. If the error code was non-zero, the script will print an error message.
5. Saves the events from tfile in the data struct called EventRecordsStructs
6. Retrieve and print the number of records in the file
7. Once the script has retrieved the event data from all of the files, if the *IncludeEventFile* is set to true it will load the event file and sort the events chronologically
8. If the *IncludeEventFile* is false, the script will concatenate all of the event data since it is already in order.
9. The script will then fix the numbering of the events

After running the script, the events should be listed in chronological order in the struct EventRecords, which contains six fields: EventNumber, TimeStamp, TimeMsFromMidnight, TimeSource, EventType, and Details. These fields are explained in section 3.3 of this document.

Tested in version 2016a.

5.2 Python

Python users must access the main DLL via the wrapper DLL (Event_File_Reader_Dll_UM.dll). All the necessary files are contained in Event_File_Reader_9_0_Python_Example .zip. The user must place the main DLL (Event_File_Reader_9_0.dll) in the same folder as the python.exe executable. Type the following into the python shell:

```
>>> import sys
>>> print(sys.executable)
```

This will print the name of the folder where your python executable is running from, which is the location where you should place the Event_File_Reader_9_0.dll.

The wrapper dll can be placed in any folder.

The main script is called event_reader_python_example_main.py.

In the section titled "User settings", the user should set the following variables:

wrapperDll	the name of the wrapper DLL (Event_File_Reader_Dll_UM.dll) including the full or relative path.
dataFolder	the full path of the folder containing your data files
file_collection_type	"Block" for block file format and "Flat" for flat file format.

If you are using block file format:

include_event_file	Set to True if you want to include EVENT000.DF1
data_file_prefix	Set to the four letter prefix of your data files (e.g. for NEUR0001.DF1, set this to "NEUR"
event_file_prefix	The 5 letter prefix of your event file (e.g. for EVENT000.DF1, set to "EVENT"
data_digit_padding	This refers to the number of digits in neural file names. This is set to 4, do not modify this value.
event_digit_padding	This refers to the number of digits in event file names. This is set to 3, do not modify this value.
extension	This is ".DF1". Do not modify this value.
start_file_index	This is the index of the first file in the collection of files you wish to analyze. E.g. if the first file is NEUR0003.DF1, you would set this value to 3.
end_file_index	This is the index of the last file in the collection of files you wish to analyze. E.g. if the last file is NEUR00010.DF1, you would set this value to 10.
event_start_file_index	This is the index of the first event file in the collection of files you wish to analyze. E.g. if the first file is EVENT000.DF1, you would set this value to 0.
event_start_file_index	This is the index of the last event file in the collection of files you wish to analyze. Typically there will be only 1 event file in which case this value will also be 0.

If you are using flat file format:

Do not modify any additional values

For all data types:

Once the user has entered the above settings, the script is ready to be run. The script does the following:

1. Imports the necessary libraries
2. Initializes the wrapperDLL path, the data folder
3. Creates buffers that the DLL will use to store information
4. Loads the wrapper DLL
5. Loads the events from each of the files chosen by the user and prints "Files were loaded successfully" for each file. If it fails to load a file, it will print the error and exit the script.

6. It will print the number of records in each file
7. If the event file is included in block files, then it will load the event file as well
8. It sorts the events if in block file format and including event file. Otherwise, it will flatten the event records.
9. It renumbers the records to reflect chronological order
10. It prints all the records with all the fields in order.
11. Frees the DLL

Tested in Python version 3.6.5

Appendix A

This appendix describes the different types of events that may appear in a file.

Error An error has occurred in the recording. Errors may be of the following types:
Dropped block type 0 – data transfer to the memory card driver was not fast enough so a single 64 kB block[1] of data was not written to the memory card.
Dropped block type 1 – the memory card was unable to accept data quickly enough so a 64 kB[1] block of data was dropped. If this happens more than once per 10GB of information, the user is probably using an unsuitable memory card.
Dropped block type 2 – a 64 kB[1] block of data was dropped because the logger needed to restart its data transfer hardware.
Motion sensor restart – the motion sensor chip needed to be restarted.

System Running Reserved

Stimulus In loggers that have the optional electrical stimulation module, this event occurs when an electrical stimulus is fired.

Fast reset This event occurs in loggers that support the option to rapidly reset the high pass filter if an input overload occurs.

File started The timestamp of the start of a new 16 MB neural file.

Red-LED/ Green IR-LED event The LED on the logger was turned on or off if the user has elected to record such events.

Free text A custom event the user logged via the LoggerCommand3 program.

[1] The number of milliseconds per 64 kB block depends on the number of the channels in the system in the following table:

Number of channels	16	32	64
Number of ms	65.5	32	16

Digital in A digital input event has occurred. Such an event can be described in two different formats:

1. Hexadecimal representation of the digital input port and event status. The port status, when represented as a binary, describes the position of each of the 32 pins. The event status when represented as a binary describes which pins underwent events (for a given pin, 1 means an event has occurred and 0 means no event has occurred)

Digital in	Digital input port status = 0x73ffff33; Digital input event status = 0x00000080;

For example, in the above event, the input port status is 0x73ffff33 is represented in binary as 0111 0011 1111 1111 1111 1111 0011 0011 indicating that (zero indexed) pins 2, 3, 6, 7, 26, 27 and 31 are off and all other pins are on. The input event status 0x00000080 is represented in binary as
0000 0000 0000 0000 0000 0000 1000 0000 meaning that there was an event only on pin 7 (zero indexed). Since the input port status of pin 7 is 0, this means that the event that occurred was a falling edge.

2. A string describing details of the events including the number of the pin on which it occurred, the nature of the event (rising or falling), and the name of the event.

Digital in	Digital in falling edge on pin number 8. Event name: Pin4Event;

This is the same event as the one above but displayed as a string. It says the event occurred on pin 8 because when displayed this way the pins are 1 indexed. Pin4Event is simply the name the user chose to give to an event on this pin.
The digital input information is always available in format 2, and is available in format 1 as well when the transceiver software has been set up for >4 digital inputs. The controls in the Display Control panel are used to set the display status of digital inputs as format 2, whereas the “Digital in” checkbox controls the display status of the event in format 1, if available.

Digital out A digital output pulse was sent from the transceiver. Details of the event (e.g. rising/falling) are recorded.

IRIG-J2 sequence An IRIG serial time sequence was sent to the digital output terminal of the transceiver.

Mode change The mode (e.g. recording, sleeping, monitoring) of operation of the logger was changed as detailed.

Clocks synchronized **Reserved.**

Recording parameters this record is sent at the start of each new recording session. It contains 17 parameters describing various aspects of the recording including firmware version, date, and channel map.

Parameter change Any parameter has been changed as detailed.

Automatic shutdown	The logger has shut itself down.
GPS Timing Pulse	In systems where the transceiver is equipped with a GPS clock[2], that clock can generate one pulse per second synchronized to GPS time and send each pulse as an event.
GPS Time String	In systems where the transceiver is equipped with a GPS clock[2], the GPS can also send a time string instead of a pulse.
Startup	Occurs when the LoggerCommand3 program initializes a system.
Warning	The message “data buffer is nearly full” appears when the write operation to the memory card has taken longer than usual. It is possible that data in a 64kB block are mistimed.
Information	The message “data buffer is nearly full” indicates that the write operation to the memory card has taken longer than usual but the data were correctly recorded.
PC-generated comment	A comment sent by the PC to the logger. This can contain information about the battery voltage or number of channels.
Tone generated	In instruments that have an audible sounder, a tone was activated. [2]
General check	Periodic system checks of the logger- for e.g. the battery voltage, signal strength, etc

[2] Only some loggers contain this capability