

JEventViewer 2.0

User's Guide

Carl Timmer

Jefferson Lab Experimental Physics Software
and Computing Infrastructure group

26-May-2021

© Thomas Jefferson National Accelerator Facility
12000 Jefferson Ave
Newport News, VA 23606
Phone 757.269.7100

Table of Contents

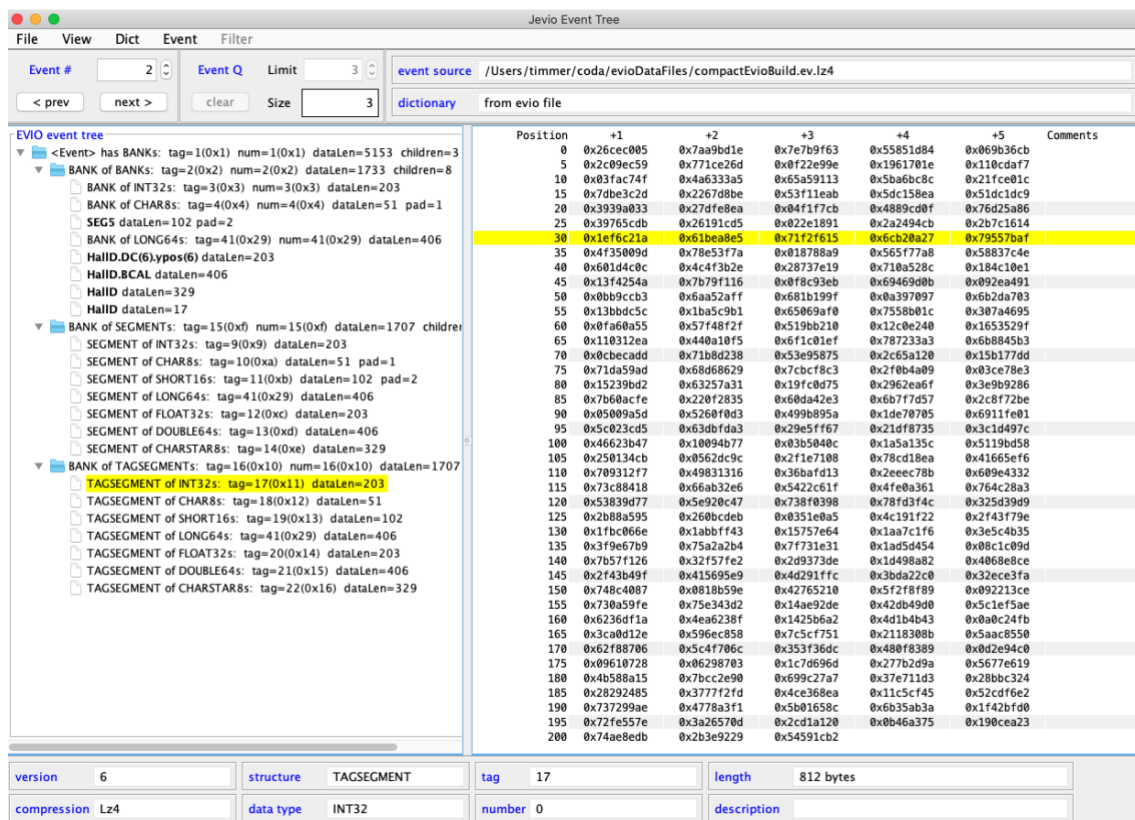
1.	Evio Event Viewing	ii
1.1	Installation	iii
1.2	Prerequisites	iv
1.3	Documentation.....	iv
1.4	Features	iv
2.	File Data Viewing	7
2.1	Searching	8
2.1.1	By Value.....	8
2.1.2	By Location	9
2.1.3	By Page	9
2.1.4	By Evio Record/Block Header.....	9
2.1.5	By Evio Event	9
2.1.6	By Evio Faults	10

Chapter 1

1. Evio Event Viewing

This manual describes a graphical user interface for looking at EVIO format files event-by-event, although it can also look at any file as a list of 32 bit integer (words).

Figure 1.1: Event-viewing gui



This version of the JEventViewer is compatible with all evio formats, including the latest, evio 6.0. To run it, using Java 8 or later, simply execute:

```
java org.jlab.coda.eventViewer.EventTreeFrame
```

Make sure that the jar files, JEventViewer-2.0.jar and all the other jars in java/jars directory, are your CLASSPATH environment variable. The alternative to that is executing the provided script:

```
scripts/jeviodump
```

Note that the script is for CODA users and sets the classpath to:

```
$CODA/common/jar
```

In other words, make sure that your environmental variable CODA is defined and all the jar files in java/jars are in that directory as well.

1.1 Installation

The code can be downloaded from its github site:

```
git clone https://github.com/JeffersonLab/JEventViewer.git
```

The default branch is "2.0" but one can insure that by calling:

```
git checkout 2.0
```

There's the jar file **JEventViewer-2.0.jar** in the **java/jars/java8** directory, already pre-built with Java 8, so one does not need to build it. There's another one in **java/jars/java15** directory built with Java 15.

However, to build it simply do:

```
ant jar
```

Other options can be seen by calling:

```
ant help
```

The output of this command is:

```
help:
[echo] Usage: ant [ant options] <target1> [target2 | target3 | ...]

[echo] targets:
[echo] help      - print out usage
[echo] env       - print out build file variables' values
[echo] compile   - compile java files
[echo] clean     - remove class files
[echo] cleanall  - remove all generated files
[echo] jar       - compile and create jar file
[echo] install   - create jar file and install into 'prefix'
[echo]             if given on command line by -Dprefix=dir',
[echo]             else install into CODA if defined
[echo] uninstall - remove jar file previously installed into 'prefix'
[echo]             if given on command line by -Dprefix=dir',
[echo]             else installed into CODA if defined
[echo] all       - clean, compile and create jar file
[echo] javadoc   - create javadoc documentation
[echo] developdoc - create javadoc documentation for developer
```

```
[echo]      undoc      - remove all javadoc documentation
[echo]      prepare    - create necessary directories
```

Although this is fairly self-explanatory, executing ant is the same as ant compile. That will compile all the java. All compiled code is placed in the generated ./build directory. If the user wants a jar file, execute ant jar to place the resulting file in the ./build/lib directory. The java command in the user's path will be the one used to do the compilation.

1.2 Prerequisites

The other jar files necessary to compile JEventViewer-2.0.jar are in the java/jars directory. They are compiled with Java 8. In addition, there are 2 subdirectories:

- 1) java/jars/java8, which contains all such jars compiled with Java 8, and
- 2) java/jars/java15 which contains all jars compiled with Java 15.

If a jar file is not available in Java 15 use the Java 8 version.

To generate these jar files, go to their respective github sites and follow the directions there:

<https://github.com/JeffersonLab/disruptor>

<https://github.com/lz4/lz4-java>

<https://github.com/JeffersonLab/evio>

<https://github.com/JeffersonLab/et>

<https://github.com/JeffersonLab/cMsg>

1.3 Documentation

Basically, you are now reading the only user documentation in either a pdf or word doc. In the repository, it's located in the **doc/users_guide** directory. There is javadoc that can be generated (ant javadoc or ant developdoc) but would only be useful to a developer or one trying to modify the source code.

1.4 Features

Here's a quick list of the main features:

- Valid event sources are files, cMsg messages, and ET buffers
- Fast compare ability for data from different events
- When receiving events through cMsg or ET, they can be filtered based on their CODA event type (physics, control, etc.) and trigger type if physics event

- View integer data as hex or decimal
- Select dictionary from event source or from separate file containing dictionary
- View the dictionary being used
- Export any evio file in xml format
- View the contents of any file as 32 bit hex integers
- Search for values, positions, evio records/blocks, evio events, or evio errors

In the figure above, starting with the middle of the gui first, the left side shows a tree structure diagram of the whole, single evio event being viewed. Notice that the type of each evio structure is given (bank, segment, tagsegment), along with the type of data it contains, tag, num, size, and # of children. Tag and num are shown in decimal and hex. If a dictionary is being used, the dictionary name is displayed instead of the corresponding structure type, data type, tag, and num values.

The right side, on the other hand, shows the data of any selected bank, segment, or tagsegment that contains a data type and not another container type. Integers can be displayed in hex or decimal.

A fast compare feature is able to compare data from different events. If the current event is changed while viewing the data of its selected structure, and if the new event has a structure with the same hierarchy of tags that the previous selection had, it too is automatically selected. This facilitates comparing the same structure in each successive event by simply hitting the “next” event button.

A dictionary can be loaded from a separate xml format file, or it can come embedded in an evio format file or buffer (cMsg, ET). The viewer allows the user to switch, in the “Dict” menu, between the different dictionaries if more than one is available. Any dictionary being used can be displayed instead of the data.

Selecting an ET system or a cMsg server as an event source, in the “Event” menu, brings up other menus to allow the proper connections to be created and maintained. The only assumptions made are that in a cMsg message, the evio data is contained in the byteArray field. Any dictionary is first looked for in the evio data and if none is found, it is looked for in a String payload item called “dictionary”.

The box in the upper left (under the row of menu buttons), “Event #”, shows the event currently selected (in this case 2) and allows the user to navigate to the desired event.

The box to its right, “Event Q”, shows different things depending on if the data source is a file, cMsg message, or ET event. For files, it shows the total number of events (in this case 3). For cMsg messages and ET events, on the other hand, events are continually arriving. In this case, “Size” shows the number of events currently in an internal queue. “Limit” allows the user to set the size of this internal queue, while “Clear” will remove all events currently in the queue. Once this queue is full, nothing else is added. The “Event #” controls can be used to switch between events in the queue.

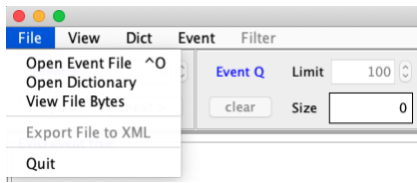
Switching between the different event sources can be done in the “Event” menu item. When selecting a cMsg or ET source, the “Filter” menu is enabled. With this menu, the user can choose to look at control, partially-built physics, physics events, or any combination as well as the selecting the run type of interest.

Notice that above the data, there are boxes containing the event and dictionary sources. Beneath the data are boxes containing information about the selected data structure such as its structure type, data type, tag, num, length in bytes, description, evio version, and the type of data compression if any.

Warning about performance: for large files, make sure they are local to the machine that’s running this program since it uses memory mapping to look at file data. You do not want the performance hit you’ll take for viewing files which are served over the network!

Chapter 2

2. File Data Viewing



The following figure is a screen shot of a file's data obtained by selecting the “View File Bytes” option of the “File” menu of the initial screen shown previously.

Figure 2.1: Data-viewing gui

compactEvioBuild.ev bytes

File

/Users/timmer/coda/evioDataFiles/compactEvioBuild.ev

Word Position +1 +2 +3 +4 +5 Comments

0 0x4556494f 0x00000001 0x0000000e 0x00000002 0x00000000 File Header

5 0x10000506 0x0000026c 0xc0da0100 0x00000000 0x00000000

10 0x00000000 0x0000f4c4 0x00000000 0x00000000 0x00000000

15 0x00000001 0x0000000e 0x00000001 0x00000004 0x00000006

20 0x00000000 0xc0da0100 0x0000022e 0x00000000 0x00000000

25 0x00000000 0x00000000 0x00000000 0x0000022e 0x3c786d6c

30 0x44696374 0x3e0a2020 0x3c62616e 0x6b206e61 0x6d653d22

35 0x48616c6e 0x44220208 0x20202020 0x20202020 0x20202074

40 0x61673d22 0x362d3822 0x20202074 0x70653d22 0x62616e6b

45 0x22203e0a 0x20202020 0x20203c64 0x65736372 0x69707469

50 0x6f6e206e 0x6f726d61 0x743d224e 0x65772046 0x6f726d61

55 0x7422203e 0x68616c6e 0x5f645174 0x61675172 0x616e6765

60 0x3c2f6465 0x73637269 0x7074696f 0x6e3e0a20 0x20202020

65 0x203c6261 0x6e6b206e 0x616d653d 0x22444328 0x25742922

70 0x20202020 0x20202020 0x7461673d 0x22362220 0x6e756d3d

75 0x22342220 0x3e0a2020 0x20202020 0x20202020 0x3c6c6561

80 0x66206e61 0x6d653d22 0x78706f73 0x28256e29 0x22202074

85 0x61673d22 0x3622206e 0x756d3d22 0x3522202f 0x3e0a2020

90 0x20202020 0x20202020 0x3c62616e 0x6b206e61 0x6d653d22

95 0x79706f73 0x28256e29 0x22202074 0x61673d22 0x3622206e

100 0x756d3d22 0x3622202f 0x3e0a2020 0x20202020 0x3c2f6261

105 0x6e6b203e 0x0a202020 0x2020203c 0x62616e6b 0x206e616d

110 0x653d2242 0x43414c22 0x20202020 0x20207461 0x673d2237

115 0x22203e0a 0x20202020 0x20202020 0x20203c6c 0x65616e20

120 0x6e616d65 0x3d227828 0x256e2922 0x20746167 0x3d227272

125 0x206e756d 0x3d22312d 0x3322202f 0x3e0a2020 0x20202020

130 0x3c2f6261 0x6e6b203e 0x0a20203c 0x2f62616e 0x6b203e0a

135 0x20203c64 0x69637445 0x6e747279 0x206e616d 0x653d2253

140 0x45473522 0x20746167 0x3d223522 0x203e0a20 0x20202020

145 0x20203c64 0x65736372 0x69707469 0x6f6e206e 0x6f726d61

150 0x743d224f 0x6c642046 0x6f726d61 0x7422203e 0x74616720

155 0x35206465 0x73637269 0x7074696f 0x6e3c2f64 0x65736372

160 0x69707469 0x6f6e3e0a 0x20203c2f 0x64696374 0x456e7472

165 0x793e0a3c 0x2f786d6c 0x44696374 0x3e0a0000 0x00002856

170 0x00000001 0x0000000e 0x00000002 0x00000008 0x00000006

175 0x00000000 0xc0da0100 0x0000a118 0x00000000 0x00000000

180 0x00000000 0x00000000 0x00000000 0x0000508c 0x0000508c

185 0x00001422 0x00011001 0x000006c6 0x00021002 0x000000cc

190 0x00030b03 0x05c31c98 0x44d1dc2b 0x785cc48b 0x5debb6e2

195 0x1ba99155 0x66051442 0x2a112f81 0x3e29b7a5 0x6601833a

200 0x16de9006 0x5c6bd049 0x061dbfff 0x60d0be21a 0x302705ca

205 0x5da933b7 0x6db1a342 0x32c217ca 0x79f2a5e2 0x7d90e09d

210 0x52ff6ec5 0x447b081c 0x1709daa3 0x207a57d1 0x48f669d9

215 0x1f5010e7 0x06cc6861 0x38d21427 0x5c8015d4 0x51d209f1

220 0x7d96c06e 0x67168a72 0x50252e0a 0x1b1e4532 0x53a74221

225 0x00852086 0x0b2a7cd4 0x1ca38e5a 0x3bcf5a38 0x0b3f5a38

230 0x115eb274 0x52672b4a 0x68d9dc2 0x44e4775b 0x03e0b750

235 0x11c9dbd6 0x2cee1759 0x7705f5c4 0x555759b7 0x4eccd9d0

240 0x557685e5 0x7ce61446 0x066b76fd 0x67d6d90f 0x010503b9

245 0x181b2bb5 0x6bf72675 0x37f90764 0x65eb252a 0x4e6d86e6

250 0x27e9b973 0x2fb40ca5 0x549f388b 0x3de76aac 0x18f9bb4c

255 0x343d0aa6 0x6ea14630 0x575a15dc 0x00c9f153 0x085429d3a

260 0x0be0106e 0x3f62c149 0x208e5adc 0x6931da35 0x549e1398

265 0x15ef9fc7 0x0c31313c 0x5ffbf51b 0x25d68603 0x6444de2b

270 0x73f7e7c7 0x1c580cb1 0x0e225802 0x7181a348 0x24a82bcb

275 0x038e8fdb 0x211d33e9 0x3da50ff3 0x7e2ce4aa 0x73b7d8bc

280 0x614a7e0d 0x63133eff 0x20bb33ac 0x10845f28 0x007a54c1

285 0x1e9a5451 0x10777428 0x568114f4 0x36ae905a 0x3df7318b

290 0x0f9a3b72 0x34dc6f3 0x6f7fc32b 0x34db0cb4 0x6f1558df

295 0x6dab37f9 0x0e023989 0x186fd98d 0x15b8d5c2 0x7cadeadb

300 0x072af9c6 0x415d94ed 0x7dd7f570 0x4495586c 0x12b4c439

305 0x544f35a5 0x696e353c 0x42c7d481 0x32851333 0x70f7ac8b

310 0x0cc44c7a 0x09a62ced 0x50911b0a 0x4796e472 0x6f95c27f

315 0x4150a985 0x3342ca12 0x54fec22f 0x44b3009e 0x6eeec0c1c

File Info

File ID 0x4556494f

File type EVIO_FILE

File split # 1

Header words 56

Record count 2

Index array bytes 0

Evio version 6

Has dictionary true

Has first event false

Has trailer & index true

User header bytes 620

User register 0x0

Trailer position 62660

User int 1 0x0

User int 2 0x0

Search By

Word Value

Word Position

Page Scrolling

Evio Record

Evio Event

Evio Fault

Search For

0xc0da0100

Search Controls

< >

Start Scan Stop

Done

Record Info

Total words 10326

Record # 1

Header words 14

Event count 2

Index array bytes 8

Version 6

Has dictionary false

Is last false

User header bytes 0

Uncompressed bytes 41240

Compression type None

Compressed words 0

User register 1 0x0

User register 2 0x0

Color Key

File

Header

Index array

User header

Record Normal

Header

Index array

User header

Errors

Record with error

Event with error

Evio struct error

Event normal

Word value

Current selection

There are occasions when one wants to examine the raw bytes in a file. This tool will allow one to do just that. It is capable of viewing any file's data, although it's designed specifically to look at evio versions 4 and 6 format data.

Each cell of the table contains 32 bits worth of data displayed in hex. Data can be switched between big and little endian under the "File" menu. The table contains up to 1GB worth of data at one time. For larger files, the next or previous 1GB are loaded when required while scanning through it. On the immediate right of the data is a slider which indicates where the current view is in relation to the part of the file that is currently memory mapped (up to 1GB). On the far right is a color key to highlighted cells.

The figure above is showing an evio 6 format file. All such files have a file header shown in blue. The light blue is the main header of 14 words. Although there is no index in this case, there is a dictionary which is stored in the file header's so-called user header. This is seen highlighted in dark blue. In the "File Info" box on the top left, all values in the file header appear in a table.

When searching for record headers, each one shows up highlighted in green. The light green is the main header of 14 words. The mandatory index of events shows up in medium green. Although not seen above, since it isn't used in evio, any associated user header is shown in dark green. When a record header is found, it's data is shown in the "Record Info" box on the left.

When searching for events, the first 2 words of each are highlighted in cyan. When an event is found, it's data is shown in the "Event Info" box on the left (not seen in the figure above).

2.1 Searching

In order to facilitate finding the data of interest, there are a number of different ways to hunt through it. The control panel on the left has "Search By" radio buttons allowing one to select whether to search by:

1. Looking for a given value
2. Jumping to a given position in the file
3. Scrolling page by page or by blocks of 40 pages
4. Jumping from one evio record/block header to the next
5. Jumping from one evio event to the next
6. Scanning the whole file for evio faults or errors

2.1.1 By Value

Look for a given value by selecting the "Word Value" radio button, typing the value into the "Search For" widget, and then hit the forward or backward search button under "Search Controls". The "Stop" button will be activated since searching a large file (say 20GB) may take extended time. If a search is stopped,

the view position stays where it was when the search was started. If stopped, starting another search starts from the same location. A progress bar is there to estimate how much of the file has been searched.

When a value is found, it is highlighted in gold. Hit the search button again to find the next or previous value. Highlights can be cleared under the “File” menu.

2.1.2 By Location

Look at a given location in the file by selecting the “Word Position” button, typing the position into the “Search For” widget, and then hitting the “Go” button. The view jumps to the given location and the value is selected (but not highlighted). The first position starts at 1, not 0. You can read the position from the table by taking the number in the far left column and adding the number of the heading at the very top of the column.

2.1.3 By Page

The “Page Scrolling” button activates the “<” and “>” buttons which hop through the file page (or view) by page. It also activates the “<<” and “>>” buttons immediately underneath which move through the file in 40 pages at a click.

2.1.4 By Evio Record/Block Header

For evio version 4 files: look for an evio format block header by selecting the “Evio Block” button. The program first looks for the magic # (0xc0da0100) of an evio block header. If found, it checks that the header length is 8 words. If so, it highlights all 8 words in green. All the information contained in that header is also displayed on the left in a panel called “Block Info.”

For evio version 6 files: look for an evio format block header by selecting the “Evio Record” button. The program first looks for the magic # (0xc0da0100) of an evio record header. If found, it checks that the header length is 14 words. If so, it highlights all 14 words in light green. It highlights the index part of the header in medium green, and the user header part in the darkest green. All the information contained in that header is also displayed on the left in a panel called “Record Info” which can be seen in the figure above.

2.1.5 By Evio Event

Look for an evio event (top level evio bank) by selecting the “Evio Event” button. This is less straightforward than looking for record/block headers since there is no universal signature to look for. There are two ways to do the search. The first way is start the search immediately upon loading the file’s data or to first select a position before any events. Then hit the forward button. It is smart enough to hop over any file/record/block headers encountered and uses the length found in the event’s header to be able to find the next one when the forward button is clicked again. The first two words (or header) of each event found in this way is highlighted in cyan and the header information is displayed on the left in a panel called “Event Info” (see figure below).

Event Info	
Length	4
Tag	0xffd1
Num	0
Type	BANK
Data type	UINT32
Padding	0
Bank Type	Prestart event

2.2 Event information panel

The second way to search is to select the known first word of an event with the mouse. Hit the forward button to find subsequent events. Remember that the word immediately after a record/block header is the first word of an event. Hint: selecting the first word of any bank structure (top level or not) will display all of its information

A quick note on the bank type. In CODA online, some tags are reserved for specific purposes. If a selected event has such a reserved tag, its purpose will be shown as the "Bank Type".

2.1.6 By Evio Faults

Look for faults or errors in the evio format by selecting the "Evio Fault" button. Simply hit the "Start Scan" button and this program scans the file from beginning to end (or as far as it can parse) and lists all blocks containing errors in a panel on the left called "Evio Errors" (which can be seen in figure 2.3 below).

The algorithm used to find these errors tries to parse as much of the file as possible. For example, if a block header length does not equal the sum of the lengths of all the events it contains, then the block header length is assumed for the moment to be correct and the event lengths in error. It tries to continue by scanning the next block and stops if it encounters an unrecoverable error or makes it to the end of the file.

Errors that are caught include bad/inconsistent values in a block/event header, wrong endianness of the displayed data, length of block header not consistent with length of contained events, and not enough data to read block/event (usually a bad length), and too large of an event count in a header. The search can go into events themselves to find lower level evio errors.

For an evio version 6 file, it will find inconsistencies between compression type and header values of compression word length and uncompressed data length. Any conflict between the index length and the number of events in a record will be flagged. Of course, if a file contains compressed data, evio events will not be scanned.

To print out suspicious record numbers or record header sizes, one must set the debug flag by hand in the scanFileForErrors() method of the EvioScanner(V6).java file.

Each block in which there is a problem is listed as a button. Click one and it hops to the beginning of that block which will be highlighted in red. Within that block, the ">" and "<" buttons move from event to event. If an event has an error, it is the last event to be accessible through the search buttons and will be highlighted in purple. If the event containing the error has an internal bank or structure with an error, it can also be

accessed through the search buttons and will be highlighted in orange. A corresponding error message (or messages) is displayed at the top of the gui in red text.

Below, a small file with evio format errors has been scanned. It reveals errors in 2 records. The first record is selected showing, in red, a header with an uncompressed data length of 0 even though there is no compression. It also shows the header saying it contains 3 events but there are entries in the index for only 2. Finally, it found an error in the first event, signified by its header in purple. The error is in a sub-structure, highlighted in orange. In this case a little investigation shows that the second bank header word shows padding of 2 for a data type of 32 bit unsigned int, when it should be 0.

Figure 2.3: Error Scanning

The screenshot displays the 'HandCreatedV6.ev bytes' application window. The main area shows a table of word positions and their corresponding hex values, with some rows highlighted in red and orange to indicate errors. The table has columns for Word Position, +1, +2, +3, +4, +5, and Comments. The first record is selected, showing a header with an uncompressed data length of 0. The color key on the right indicates that red highlights the 'Record with error', orange highlights the 'Event with error', and yellow highlights the 'Current selection'.

Word Position	+1	+2	+3	+4	+5	Comments
0	0x4556494f	0x00000001	0x0000000e	0x00000003	0x00000010	
5	0x10000406	0x0000000c	0xc0da0100	0x0070605	0x04030201	File Header
10	0x00000000	0x0000015c	0x00000000	0x00000000	0x0000007c	
15	0x00000002	0x0000000c	0x00000003	0x00000111	0x00000222	
20	0x00000333	0x0000001f	0x00000001	0x0000000e	0x00000003	
25	0x00000008	0x00000006	0x0000000c	0xc0da0100	0x00000000	
30	0x00000000	0x0070605	0x04030201	0x0070605	0x04030201	
35	0x0000001c	0x00000014	0x00000001	0x00000002	0x00000003	
40	0x00000006	0x00015e01	0x00000004	0x00028102	0x00000001	
45	0x00000001	0x00000001	0x00000004	0x00030103	0x00000002	
50	0x00000002	0x00000002	0x00000023	0x00000002	0x0000000e	
55	0x00000003	0x0000000c	0x00000006	0x0000000c	0xc0da0100	
60	0x00000000	0x00000000	0x00000001	0x00000002	0x00000003	
65	0x00000004	0x00000014	0x00000014	0x00000014	0x00000111	
70	0x00000222	0x00000333	0x00000004	0x00018501	0x00010002	
75	0x00030004	0x00056666	0x00000004	0x0001c701	0x01020304	
80	0x05060708	0x09333333	0x00000004	0x00010101	0x00000005	
85	0x00000005	0x00000005	0x00000012	0x00000003	0x0000000e	
90	0x00000000	0x00000010	0x00000206	0x00000000	0xc0da0100	
95	0x00000000	0x00000000	0x00000011	0x00000022	0x00000033	
100	0x00000044	0x0000007c	0x00000002	0x0000000c	0x00000003	

The GUI also includes a 'File Info' section on the left with fields for File ID, File type, File split #, Header words, Record count, Index array bytes, Evio version, Has dictionary, Has first event, Has trailer & index, User header bytes, User register, Trailer position, User int 1, and User int 2. A 'Search By' section allows searching by Word Value, Word Position, Page Scrolling, Evio Record, Evio Event, or Evio Fault. A 'Search For' field contains '0xc0da0100'. The 'Evio Errors' section shows 'Block 1' selected. The 'Search Controls' section has buttons for '<', '>', 'Start Scan', 'Stop', and 'Done'. The 'Record Info' section shows details for the selected record, including Total words, Record #, Header words, Event count, Index array bytes, Version, Has dictionary, Is last, User header bytes, Uncompressed bytes, Compression type, Compressed words, User register 1, and User register 2. A 'Color Key' section on the right lists 'File', 'Header', 'Index array', 'User header', 'Record Normal', 'Header', 'Index array', 'User header', 'Errors', 'Record with error', 'Event with error', 'Evio struct error', 'Event normal', 'Word value', and 'Current selection'.