bufdump - A utility for dumping NSCL event buffers

Ron Fox
bufdump - A utility for dumping NSCL event buffers
by Ron Fox
# Table of Contents

Preface .......................................................................................................................................................iv

1. bufdump quickstart guide.....................................................................................................................1

2. Introduction to bufdump concepts.......................................................................................................4
   2.1. Data sources ................................................................................................................................4
   2.2. Bit patterns ..................................................................................................................................4
   2.3. Buffer filters ...............................................................................................................................5
   2.4. Search patterns ............................................................................................................................5
   2.5. Packets and packet definition files ............................................................................................6
   2.6. Plugins.........................................................................................................................................7

3. Commands on the File menu ................................................................................................................8
   3.1. File→Open...........................................................................................................................8
   3.2. File→Read Packet Defs.............................................................................................................10
   3.3. File→Add Plugin.......................................................................................................................10
   3.4. File→Exit...............................................................................................................................10

4. Commands on the Filter menu ...........................................................................................................11
   4.1. Filter→Filter..........................................................................................................................11
   4.2. Filter→Search... (Ctrl-F) ........................................................................................................13
   4.3. Filter→Search Next (F3) ..........................................................................................................15

5. Commands on the Help menu ............................................................................................................16
   5.1. Help→About..........................................................................................................................16
   5.2. Help→Topics... (F1) ..............................................................................................................16

6. Extending bufdump .............................................................................................................................17
   6.1. Preparing packet definition files ...............................................................................................17
   6.2. Writing formatting plugins for the bufdump program ...............................................................18

bufdump 1daq...................................................................................................................................20
Preface

Every effort has been made to ensure that this manual is correct and accurate. Nonetheless errors may have crept in. If you find an error in this manual, please report it to http://daqbugs.nscl.msu.edu so that we can correct it as soon as possible.

If you find errors in the program or have suggestions for enhancements that will improve bufdump, we welcome your defect reports and suggestions at http://daqbugs.nscl.msu.edu.
Chapter 1. bufdump quickstart guide

This chapter will give you the minimum information you need to start using the bufdump program. In this chapter you will learn:

• How to start bufdump
• How to select a source of buffers
• How to get buffers from the data source
• How to access the program’s online help

Important: The procedure below assumes that the NSCLDAQ utility programs are in the your PATH. At the NSCL this means that the PATH environment variable includes an entry for /usr/opt/utilities/current/bin

Starting bufdump

To start the bufdump program:
1. Login to a system on which the bufdump program has been installed
2. If necessary start a terminal window
3. Type:

   bash$ bufdump

If you have successfully followed these steps, the bufdump window will appear as shown below:
Chapter 1. bufdump quickstart guide

Selecting a data source

This procedure will show you how to select an offline data source for detailed information about selecting data sources see the discussion of the File—Open... command in Chapter 3: The file menu.

1. Click the File menu on the menubar at the top of the bufdump main window.
2. Click the Open... menu entry in the File menu.

This should result in the data source dialog box like the one shown below.

Screen shot of the open data source dialog.
3. Use the lower part of the dialog box to navigate to the directory that has your event files and click on the event file you want to examine.

4. Click the Ok button. The dialog box will be removed from the screen.

**Important:** The procedure below assumes that you have already selected a data source.

---

**Stepping through the event source**

1. Locate the arrow button pointing right at the lower left of the bufdump main window

2. Click the right arrow button. The bufdump program will read the next buffer from the event source, decode the buffer and display the decoded buffer in the middle part of the bufdump main window.

3. Repeatedly clicking the right arrow button will tell bufdump to read subsequent buffers from the data source, decode and display them. Bufdump will hold several buffers in memory. As needed, the middle window will grow scroll bars to allow you to scroll back and forth through the data that has been decoded and displayed.

**Tip:** The bufdump program is quite powerful. If you want to learn more about what bufdump can do, and how to use it like an expert, read the remainder of this manual.
Chapter 2. Introduction to bufdump concepts

This chapter introduces several bits of terminology that you will encounter as you read the remainder of the bufdump documentation. If you are pretty sure you will understand the program without this explanatory chapter, feel free to skip ahead, returning here to clarify the meaning of terms that you may not understand.

In this chapter you will learn about:
- Data sources
- Bit patterns
- Buffer filters
- Search patterns
- Packets and packet definition files
- Plugins

2.1. Data sources

Bufdump must have a source of buffers to format. In this manual, sources of buffers will be called Data Sources. Event sources can be one of the following two types:

*Offline*

Offline event sources are simply event data files that have been acquired by the NSCL data acquisition system.

*Online*

Online event sources are computers that are currently taking data. Note that the time it takes to request a buffer from an online data source will depend on how quickly the readout program is producing buffers.

Event sources are opened by using the File→Open... command. This command is described in detail in Chapter 3: The file menu.
2.2. Bit patterns

Bit patterns are used to specify both filters and search patterns. Both of these terms are described later in this chapter.

A bit pattern is 16 bits that can have any of three states:

1
   Requires a matching word to have a 1 in that bit position.

0
   Requires a matching word to have a 0 in that bit position

x
   Specifies that you don’t care what the value of that bit position is.

Normally bit patterns come in sets. A bit pattern set can either represent a group of bit patterns, any of which can match to trigger the filter or search match, or a sequence, which requires a sequence of words in the buffer to sequentially match all the bit patterns in the set to trigger the filter or search.

2.3. Buffer filters

Buffer filters allow bufdump to skip buffers that don’t match a desired filter criterion. The filter criterion is defined using the Filter—Filter... command. This command is described in detail in Chapter 4: the Filter menu

Buffer filters can restrict the type of buffer that is processed, as well as require a bit pattern match to occur in the body of a buffer. If you think of bufdump as a logic analyzer for NSCL DAQ buffers, you can think of the buffer filter criterion as the trigger for that logic analyzer.

2.4. Search patterns

Search patterns and buffer filters are closely related. Both are used to limit the amount of data you must wade through to locate the data you are interested in.

While buffer filters prevent you from seeing buffers that don’t match the filter criterion, search patterns enable you to locate patterns in buffers that have matched the filter. Search strings can be either textual or
bit patterns. Search patterns are established with the Filter→Search... command, and once established, can be executed again using the Filter→Next command.

Text search patterns can search for exact string matches in the data that is displayed by bufdump. They can also match regular expressions. A regular expression is a string that can itself specify a wide variety of matching conditions. The syntax of regular expressions is powerful and complex. A discussion of regular expression syntax is beyond the scope of this manual. Look at either the unix man page regex(7), or Tcl book sections describing the regexp Tcl command for more information about the structure of regular expressions.

Bit pattern searches can be a collection of bit patterns, any of which may cause a match or a sequence of bit patterns, all of which must be matched in the order specified.

## 2.5. Packets and packet definition files

The NSCL Data acquisition system must often incorporate data from several supported devices into a single event. In order to allow the maintainers of these devices to independently build and maintain data acquisition and analysis software for their devices, the NSCL has adopted a self describing format for the body of an event that allows code to recognize the sections of an event it is responsible for and ignore sections of the event it does not know about.

The core of this self describing structure is the packet structure. A packet is a segment of an event that is created by a logical part of the readout program and can be analyzed independently by an event processor registered in SpecTcl. The format of a packet is shown below:

<table>
<thead>
<tr>
<th>Word</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Self inclusive packet word count</td>
</tr>
<tr>
<td>Id</td>
<td>Describes what packet contains</td>
</tr>
<tr>
<td>Body</td>
<td>Body - 2 words of packet data</td>
</tr>
</tbody>
</table>

The id of a packet identifies what the packet body contains (e.g. data from the Segmented Germanium Array SeGA). Daniel Bazin <bazin@nscl.msu.edu> maintains a registry of these ids. If you want to allocate a new id for a device you are constructing, contact him. The current id registry is available online at the NSCL user information website (http://groups.nscl.msu.edu/userinfo/daq/nscltags.html) where they are referred to as tags.

As far as bufdump is concerned, the significance of this packet structure is that if it knows the packet id assignments, it can represent the body of each event as a series of packets. This representation can identify by name the packets in each event.
Chapter 2. Introduction to bufdump concepts

Bufdump uses three mechanisms to learn about the packet id assignments:

- A system wide packet definition file describes the packet id assignments known at the time the program is distributed. This file is automatically read in by bufdump when it starts.

- If an event file was written using the production readout skeleton, and the author(s) of the readout program used the CDocumentedPacket class to manage the production of their event packets, the event file will contain additional documentation buffers which describe the packets used. If bufdump encounters one of these, in addition to formatting the buffer, it adds any packet definitions that are not already known to its internal database of packet types.

- User written packet definition files can be manually read in by the program. The format of these files is described in Chapter 6: Extending bufdump. How to read them in is described in Chapter 3: The File menu.

### 2.6. Plugins

Plugins are a concept that is closely related to packets and packet definition files described in the previous section. When bufdump encounters a known packet, it displays a formatted version of the header that give the packet id, packet size, and the name of the device the packet came from. By default, however, the body of the packet is dumped as just a set of hexadecimal numbers.

Since the body of each packets type is by the physicist that produced its readout code, bufdump cannot know how to format packet bodies. Device physicists, however may write plugins that can produce a meaningful formatted dump of the packet. Plugins are Tcl code which, when integrated with bufdump, allow it to display the body of a known packet in a more meaningful format. Chapter 3: The File menu describes how to incorporate a plugin in to bufdump. If you want to write a plugin refer to: Chapter 6: Extending bufdump.
Chapter 3. Commands on the File menu

This chapter describes the operations available on the File menu:

When you have finished reading this chapter you will know how to:

• Open a data source
• Read a packet definition file
• Incorporate a plugin.
• Exit the program.

3.1. File ➔ Open...

The File ➔ Open... command allows you to open a data source. Data sources can be either online or offline. An offline data source is just an event file, while an online data source is a computer on the network that is running the NSCL DAQ software.

When you click the File ➔ Open... menu entry, the data source dialog is displayed:
Chapter 3. Commands on the File menu

The numbers in the discussion below refer to the numbers in the screenshot of the data source dialog:

1. The top section of the dialog allows you to select between an online or offline data source by checking or unchecking the Online checkbox. The appropriate areas of the rest of the dialog are disabled or enabled. The spinbox to the right of the checkbox allows you to set the buffersize for the data source if you are not using the default NSCL buffersize of 8192 bytes.

2. This section is only enabled if the Online checkbox is checked. Fill the text box in with the name of the computer on which the online data source (Readout) will be running. In order to use online data sources, you must run bufdump on a system that is running the NSCL data acquisition system. The host you use as the event source must also be running the NSCL Data data acquisition system.

3. The Filter entry is part of the offline data source selection area. It is only enabled if the Online checkbutton is not checked. You can enter a path with wild card characters in the final path element to determine which files will be shown in the file selection part of this area (labelled 5). The screenshot we are annotating only displays the files with the .evt filetype in the directory H:/Wincluster/DAQDocs/2005a/daq/bufdump. If you navigate the directory hierarchy (via the directory navigation box labelled 4), the leading part of this path will automatically change to match the directory you are browsing. After modifying this entry, hit the enter (or return) key to cause the new filter to be applied.

4. The directory navigation box allows you to navigate the filesystem directory hierarchy. If the directory you are examining has subdirectories, they will be listed in this box. You can navigate into a subdirectory by double clicking it. You can navigate up one level in the hierarchy by double clicking the .. entry in this list. You can move to a distant part of the file system hierarchy by editing the filter string as described in 3 above.

5. The file selection box allows you to select a specific event file. If you click on a file in this box, the full path to the file will be copied to the Selection entry labelled 6.
Chapter 3. Commands on the File menu

6. The Selection entry box indicates the name of the offline data source that will be used if you click the Ok button. You may edit this name or select a file in the file selection box (labelled 5).

7. The action area of the dialog has two buttons.

Ok
Accepts the selected data source and dismisses the dialog. Once you click the Ok button, the next buffer button will cause the next data buffer to come from this new data source. Note that any previously open data source will be closed.

Cancel
Dismisses the dialog without changing or opening a new data source.

3.2. File → Read PacketDefs...

Packet definition files extend the set of event packets that are recognized by the buffer dumper. Chapter 2: Introduction to bufdump concepts describes packets and packet definition files. Chapter 6: Extending bufdump describes how to create packet definition files. Note that you only need to use this if bufdump does not recognize some of the packets that are in your events.

Clicking the File → Read PacketDefs... menu command brings up a simple file chooser dialog. The appearance of this dialog will differ from operating system to operating system. Select the packet definition file you want to read and click the appropriate button on the file chooser.

3.3. File → Add Plugin...

The File → Add Plugin... command allows you to incorporate a plugin into the bufdump program. Plugins help bufdump format the bodies of recognized packets within an event. Chapter 2: Introduction to bufdump concepts describes plugins. Chapter 6: Extending bufdump describes how to write plugins.

3.4. File → Exit

Clicking the File → Exit menu command exits the bufdump program.
Chapter 4. Commands on the Filter menu

This chapter describes the options that are available on the Filter menu:

![Filter menu](image)

When you have finished reading this chapter you will know how to:

- Set a buffer filter to select a subset of the buffers in the event file.
- Create a search pattern to locate text, regular expressions or bit patterns within data buffers that have already been read and formatted by bufdump.
- Repeat a previous search of the established search pattern.

If some of the concepts and terminology described above are unfamiliar or unclear to you, please (re)read: Chapter 2: Introduction to bufdump concepts where these concepts are described.

4.1. Filter ➔ Filter...

The Filter ➔ Filter... menu command allows you to establish a filter on the buffers that are processed by bufdump. Filters allow you to specify which buffers are interesting to you. bufdump will skip buffers that do not make the filter criterion you establish via this command.

Since there is never an end of data source on an online data source, exercise care in establishing filters on these sources. If you read the next buffer while a filter is established that can never be matched, bufdump will hang forever ignoring buffers until you abort it.

When you click on Filter ➔ Filter..., the filter creation dialog will pop up. This dialog is shown below:
In the discussion that follows, the numbers refer to the circled numbers on the screenshot.

1. Checking the **Enable Buffer Filtering** checkbox enables a selection of the subset of buffer types. This selection is applied prior to any bit pattern matching on the contents of a buffer. When this checkbox is checked, the remainder of this frame is enabled. Select the desired buffer types in the **Buffer Types** listbox and click on the right arrow button to move the selected buffer types to the **Accepted Types** list. If you no longer wish to accept buffers of a specific type, select that type in the **Accepted Types** listbox and click the left arrow button to move that buffer type back to the **Buffer Types** listbox.

2. This section of the dialog allows you to visually edit a bit pattern. Each \( x \) below represents a bit. Using the up arrow buttons above and the down arrow buttons below each bit you can toggle each bit between the states:

- \( 1 \) Means a match requires a 1 in this bit position
- \( 0 \) Means a match requires a 0 in this bit position
- \( x \) Means that any bit value will match in this position.

3. Once you have finished creating a bit pattern add it to the pattern list list box on the right by clicking the right arrow button.
Chapter 4. Commands on the Filter menu

4. You may alter the order of the bit patterns in this listbox by selecting a pattern and clicking the up or down arrow buttons to move it up or down within the list. If you wish to remove a pattern from the list select it and click the Remove button.

5. Bit pattern matches can be either a Match any criterion where a buffer is accepted if the body of the buffer has a bit pattern that matches any of the bit patterns in the right listbox, or Match Sequence where a match occurs only if a sequence of words matches all of the bit patterns in the ordered sequence of bit patterns in the right listbox. Use these radio buttons to select which type of match you want.

6. This strip of the dialog has three buttons:

   **Ok**
   
   Accepts the filter and dismisses the dialog. Note that the filter is applied the next time you ask bufdump to get a buffer from the data source. Data buffers already accepted and formatted remain displayed.

   **Clear**
   
   Clears the filter. A cleared filter accepts all buffers.

   **Cancel**
   
   Dismisses the dialog without making any changes to the current filter.

4.2. Filter→Search... (Ctrl-F)

The Filter→Search... (Ctrl-F) command allows you to establish a search pattern for the data that has already been formatted by the bufdump program. The command also does an initial search for this pattern.

The search pattern can search either for exact matches to text strings in the formatted buffer, to matches for regular expressions or for bit patterns or sequences of bit patterns.

The menu command brings up the search dialog shown below.
Chapter 4. Commands on the Filter menu

The numbers in the list below refer to the numbers on the figure above:

1. You may enter strings for text searches in this text entry box. Note that text entry is only enabled when a text search is selected (see 6 below).

2. The radio buttons to the right allow you to select between an exact textual search or a search for a regular expression. An example of a regular expression string is [Ee]vent which matches either "Event" or "event". The full syntax definition of regular expressions is powerful and complex. A discussion of this syntax is beyond the scope of this document. Refer either to the unix regex(7) man page or the documentation of the Tcl regexp command in any Tcl book for more information about regular expressions.

3. This section of the dialog allows you to visually edit a bit pattern. This section of the dialog is only enabled if the Bit Pattern Search radio button is selected. Each x below represents a bit. Using the up arrow buttons above and the down arrow buttons below each bit you can toggle each bit between the states:

   1
   Means a match requires a 1 in this bit position

   0
   Means a match requires a 0 in this bit position

   x
   Means that any bit value will match in this position.

Once you have finished creating a bit pattern, it can be added to the set of match patterns by clicking the right arrow button.

4. You may alter the order of bit patterns by selecting a pattern and clicking the up or down arrow buttons here to move it up or down within the list. If you wish to remove a pattern from the list select it and click the Remove button.

5. Bit pattern matches can be either a Match any criterion where a buffer is accepted if the content of the buffer has a bit pattern that matches any of the bit patterns in the right listbox, or Match
Sequence where a match is only declared if a sequence of words matches all of the bit patterns in the ordered sequence of bit patterns in the right listbox. Use these radio buttons to select which type of match you want.

6. These radio buttons allow you to select the search type. Selecting one of them enables the appropriate part of the dialog, and disables the inappropriate part.

7. The buttons in this frame operate as follows:

Ok

Accepts the search pattern, and performs a search from the beginning of the formatted data for the first match on the new search pattern.

Cancel

Dismisses the dialog and neither changes the search pattern nor performs a search.

When a search pattern matches, it is brought into the displayed part of the formatted data and the section that matches is highlighted. This highlighting is not exact, however the first element highlighted corresponds to the start of the match. The picture below shows a match on the bit pattern sequence:

```
0101 1000 xxxx xxxx, 0000 0000 0000 0001:
```

A bit sequence match

4.3. Filter\textgreater \textgreater \textgreater \textgreater Search Next (F3)

The Filter\textgreater \textgreater \textgreater \textgreater Search Next (F3) command repeats the search with the current search pattern. If no search pattern has been established yet, an error message is displayed.
Chapter 5. Commands on the Help menu

The Help menu provides commands that access information about bufdump as well as its extensive online help facility. This chapter describes those commands:

When you are finished reading this chapter you will know:

- How to obtain program identification information
- How to access the online help

5.1. Help → About...

The Help → About... command displays a dialog that provides the program version, license terms and information about the author and owner of the program.

5.2. Help → Topics... (F1)

The Help → Topics... (F1) command accesses the extensive online help bundled into bufdump. If the online help window is already displayed it is brought to the front of the window stacking order displays the introductory help page.
Chapter 6. Extending bufdump

This document describes how to extend bufdump. When you have completed this chapter you will know how to:

• Write a packet definition file
• Write a packet body formatting plugin

6.1. Preparing packet definition files

Packet definition files extend the set of packet ids that are recognized by the bufdump program. When bufdump encounters a recognized packet id, it can do some elementary formatting that identifies the packet contents.

You need to supply a packet definition file if your program uses packet ids that are not in the registered packet id set, and you either have not used the production readout skeleton, or you have, but have not used the CDocumentedPacket class to manage your packets.

A packet definition file is just a text file. By convention, packet definition files have the .def extension. Each line of the text file defines a packet id, and consists of the following fields separated by whitespace:

• Packet short name. This should be a short one word description of the data in the packet.
• Packet id. This is a number that can be expressed in any form normally recognized by C/C++ or Tcl.
• Packet long name. This is a phrase that more fully describes the packet. This phrase is not currently used by bufdump, but may be in the future. The phrase should be quoted in any way acceptable to tcl (e.g. curly braces or double quotes).
• Packet version. This is a string that represents the version of the packet. This is not currently used by the bufdump program.

The text below shows an annotated line, from a packet definition file.

```
s800 0x5800 {S800 Spectrograph} 1.0
```

1. **s800** is the packet short name. This line is the actual definition of the S800 packet from the bufdump default packet definition file

2. **0x5800** is the id of the packet. The leading 0x indicates that this id is expressed as a hexadecimal value.
6.2. Writing formatting plugins for the bufdump program.

The internal structure of a packet is not known to bufdump. By default, therefore, bufdump dumps the body of a packet as a sequence of hexadecimal 16 bit words. Plugins are a mechanism that bufdump provides to teach it how to format the internals of a packet.

A plugin is a Tcl procedure that is associated with a particular packet type. When bufdump encounters a packet whose type is known, either through a packet definition file (standard or user provided), or through analysis of a documentation buffer, it checks to see if a plugin has been registered for that packet type. If so, bufdump hands the body of the packet off to that plugin and receives the formatted body in return.

Let’s look at a very simple plugin:

```tcl
# Plugin to format 0x8000 packets:
#
proc format8000 body {
    set result "Hex  Decimal  Octal\n"
    foreach item $body {
        append result [format "0x%04x  %05d  %06o\n" $item $item $item]
    }
    return $result
}
registerPacketFormatter 0x8000 format8000
```

1. The procedure `format8000` is the plugin procedure. We have adopted the convention that the procedure name will be of the form `format id` where `id` is the hexadecimal id of the packet type this plugin serves. The parameter `body` will contain the body of the packet when the plugin is called by bufdump. `body` is a properly formatted Tcl list of numbers. The size of the body can be determined by applying the Tcl `llength` command to that list.

2. After initializing the variable `result` with a header, this `foreach` loop iterates over each element of the body list appending the hexadecimal, decimal and octal rendition of each body word to the `result` followed by a newline. Note that all formatting must be explicitly provided by the plugin. At this point in time, there is no support for font selection or other forms of highlighting.

3. The return value from the plugin procedure is the string that will be placed verbatim, without interpretation in bufdump’s formatted output.

4. The `registerPacketFormatter` command registers the plugin to process packets with an id of 0x8000. The parameters of the `registerPacketFormatter` are in order:
Chapter 6. Extending bufdump

$id$

The id of the packet that will be processed by the plugin. This can be any numerical representation of the id that is acceptable to Tcl.

$plugin$

The procedure to invoke when the $id$ packet is encountered.

We can see from this example that the incorporation of a plugin is a three step process:

1. Write a plugin procedure that accepts the body of the packet as a parameter and produces as a result the desired formatted representation of the body.

2. In the same file as the plugin, register the plugin procedure by invoking $\text{registerPacketFormatter}$. The call to $\text{registerPacketFormatter}$ must supply the packet id as well as the name of the procedure you wrote in the previous step.

3. Users of bufdump invoke the: $\text{File} \rightarrow \text{Add Plugin...}$ menu command and select the file that provides the plugin.

A cautionary word about plugin distribution. A plugin, like any other program may have defects that you will repair over the life-cycle of the plugin. We recommend against the distribution of copies of plugins within a single institution. Plugins should be installed in some centralized repository and users should be encouraged to add them from the repository so that at any given time, they will receive the most up-to-date versions.

Finally, this plugin interface is primitive. If you have suggestions for enhancing this interface, please feel free to make them via the NSCL bugzilla web pages (http://daqbugs.nscl.msu.edu).
bufdump 1daq

Name
bufdump — Make formatted dumps of NSCL DAQ buffers

Synopsis
bufdump

Description
The bufdump program is a Tcl/Tk script that makes structured dumps of the buffers from NSCL data acquisition system event sources. Event sources can either be online or offline data sources.

The software includes extensive online help as well as a web and pdf based manual at the NSCL documentation website (http://docs.nscl.msu.edu/daq)

Prerequisite Software
bufdump makes use of the following Tcl/Tk packages that are not supplied with the software source.

BWidget
    BWidget - Jeff Hobbs BWidgets.

Iwidgets
    IWidgets - The Itk incremental widget set. Note that this in turn requires the Itk package.

dns
    Distributed name service package in tcllib.

snit
    "Simple Now In Tcl OO System" by Will Duquette. This is also part of tcllib.
Defects and Deficiencies

Several generally useful Tcl snit::type and snit::widget packages are distributed but not documented for independent use. In particular the dataSources and eventData packages contain useful types for accessing NSCL Event data within Tcl scripts. The bufdumpWidgets package contains a pair of useful Tk widgets for displaying formatted event data and unformatted event data.