How to read Train IDs at FLASH

Content

Introduction

At FLASH photons are delivered with a special pulse structure:

- so-called *pulse trains* are delivered with 10 Hz repetition rate
- every *pulse train* can contain between 1 to 800 pulses separated by 1 to 20 µs
- the total length of the *pulse train* can not exceed 800 µs!

Each *pulse train* is identified by an ***Train ID*** (synonyms: ***bunch ID, timestamp***).

Recording this Train ID on the user side allows the correlation of data acquired by the user with all the data saved at FLASH in the DOOCS system, especially with the photon intensity data measured by the GMD system.

Within DOOCS the Train ID is a 32-bit number. For the users it is transmitted via a socket connection over ethernet - TCP socket. The ID is set 20 ms before the train starts and stays 80 ms after the train. To read the Train ID by the user there are 2 options provided by FLASH:

- using a provided program (AdvancedIDServer.exe) running on the user's machine, that synchronizes with the Train ID sender and uses the local clock of the user computer to lower the jitter in reading the ID significantly. The user program connects then locally to the AdvancedIDServer program to get the ID ...
- direct reading of the ID via TCP socket

Basic pinciple of Train ID distribution at Flash1 with TIDS and AIDS

See also this sketch of the sequence with miliseconds attached to it, to see when the ID updates relative to the beam.

Note: If the central Train ID server crashes it will not resume the ID numbers where it was. In most times it will restart with ID 0, sometimes it is set to some high number by hand. For this reason you will see Train IDs with some high bit not as often as expected.

top

Train ID reading with local ID reader (Advanced ID Server) *recommended*

What does *Advanced ID Server* do?

It connects to the TCP based TIDY or TIDS server and continuously gets the Ethernet packets with the IDs. These packets are time stamped with microsecond resolution and stored. This runs all the time. Because we have a lot of data we can correlate the local time on that machine (that generated the local time stamps) and the IDs. If the Ethernet connection lags sometimes, chokes, hick ups and messes around with our packets, even drops some... Who cares. By closely monitoring all data we can define a function to calculate the Train ID from the local time on that machine.

Here it is running on Windows, but of course you can also use it on your Linux machine (build it from the source).

The users can access this server via a socket. Use localhost:58051. Connect your program to that socket and you get immediately the actual Train ID for this very moment. (e.g. in the CMD window with `telnet localhost 58051` ... note:telnet is usually deactivated in win10 and has to be activated) The packet you get looks like this:

```
3437004.37997 O 818 853
```

You get a new calculated ID every time you send any data to the server. So normally your program looks like this and should not close and open the socket all the time:

- Connect to AdvancedIDServer.
- Check the state flag and jitter number in the first packet. Ignore the ID.
- loop:
    - do your measurement
    - get a trigger
    - send something to AIDS and get the Train ID
    - save your aquired data with that ID
- until shift finished or FLASH lost beam

```
3437004.37997 O 818 853
3437005.37997 O 851 889
3437006.35997 O 885 925
```

About the format: More abstract you get a number, dot, a second number. A character. A third and fourth number.

The first number is the Train ID (decimal). The number after the dot is the fractional part of the Train ID. **The Train ID changes from "last ID" to "actual ID " 20 ms before the FEL pulse arrives at the experiment**. The fractional part tells you how far after the point in time did you query. It counts from 0.0 to 0.99999 in 100 ms. Thus you can estimate your timing. If you get xxx.20000 you asked for the Train ID 20 ms after it changed and thus at the time when the FEL hit the target. Accordingly xxx.60000 is 40 ms after the FEL. Make sure that you do not read too close to the change of the ID (not around x.0 and 0.9). So in short: you get the Train ID as **float**.

The char gives basic information about the state of the Advanced ID Server:

- O means Okay, all is well
- S means Stale, our database is quite old and if the state does not change to O soon something is wrong.
- D means Disconnected, we get no new information. Hopefully it will change to O in a moment.

So O means no worries. S and D are warnings, if they stay for more than some seconds your timing might run off.

The third number is the statistical quality of our calculation. Interpret it like something equal to timing jitter with the unit of microseconds. You can use it to monitor your Ethernet link quality. 10000 means 10 ms, everything below will not affect anyone I assume. The last number is also a statistical jitter value.

Hint: Do NOT enable "fast shutter eventIDs" in the Train ID sender program if you intend to use the Advanced ID Server! Or use a TIDY server.

Addresses:

You need to specify a suitable TIDY server for your experimental location on invocation like so:

`AdvancedIDServer.exe hasfcpuexp2.desy.de:58050` *(for Flash1)* or `AdvancedIDServer.exe flash2cpuxgmd2.desy.de:58050` *(for Flash2)*

Resources:

- AdvancedIDServer.exe
- AdvancedIDServer-inline.vi: Example how to use the Advanced ID Server with Labview (LV11)
- AdvancedIDServer on github

top

Direct reading of Train ID

Receiving the Train ID via ethernet is rather simple. Just open a TCP socket to the one of the servers and read from it.
The returned string has the following format: YYMMDD HHMMSS.mmm EVENTID

The following machines can be used:

- hasfcpuexp2.desy.de:58050 (TIDY (*for Flash1*))
- flash2cpuxgmd2.desy.de:58050 (TIDY (*for Flash2*))
- hasfhvctrl.desy.de:58050 (TIDS) (obsolete)

Please see also the sample code for C and LabView 14. Example how to embed the ethernet Train ID code into your Labview project (LV 8.6): BunchID-inline.vi. We have also C++ code which can be used to be integrate into existing projects.

The Train-ID for the train is set ~ 20 ms (for TIDS) before the first photon pulse (at the same time as event 20 or E0) Note the timing differences between TIDS and TIDY in the following sections.

To validate the connection telnet can be used:

```
$ telnet hasfcpuexp2.desy.de 58050
```

You receive for example:

```
150929 180211.495 381469E
150929 180211.595 381469F
150929 180211.695 38146A0
150929 180211.795 38146A1
```

This format probably is:

```
%y%m%d %H%M%S.%3N %X\r\n
```

where `%y` are the last two digits of the year, `%m` month, `%d` day number, `%H` the hour of the day counting from 0 to 23, `%M` minutes, `%S` seconds, `%3N` milliseconds, `%X` the pulse ID in hexadecimal notation. So it is the last column which is the relevant Train ID.

The time and date have leading zeros as necessary to ensure a constant field width. The ID number is not zero padded and thus has a varying field width.

top

Train ID sender program (TIDS) *obsolete*

The sender program is currently running on hasfhvctrl.desy.de. It can be restarted if there are problems (Event_ID_Server). It can be changed to continuously delivers Train IDs (uncheck "fast shutter eventIDs") or deliver IDs only if the fast shutter is open at this moment (check "fast shutter eventIDs" as in the example below).

Address:

- hasfcpuexp2.desy.de:58050

Do not enable "fast shutter eventIDs" if you want to use the Advanced ID Server.

The TIDY: known issues also apply here.

top

Train ID Yelling server (TIDY)

As alternative to the Train ID sender you can use this server. These are the differences:

- **TIDS** versus **TIDY**
- Possible to restrict sending of ID only when the Flash1 fast shutter is open: TIDS
- Sends correct IDs independent of fast shutter: TIDY
- Sends new IDs (20 ms) before the light hits your experiment: TIDS
- Sends new IDs a (5 ms) after the light hits your experiment: TIDY

Note that the TIDY server is especially good for use together with AIDS, as there is no interaction with the fast shutter (which would break AIDS), and the rather late update to new IDs is of no consequence, as you get the IDs as floats anyhow. So the recommodation is to use TIDY together with AIDS and whenever you can, and use TIDS only if you need the fast shutter functionality.

Address:

- hasfcpuexp2.desy.de:58050 *(for Flash1)*
- flash2cpuxgmd2.desy.de:58050 *(for Flash2)*

Known Issues

- If you write code to read the train-ID please make sure you close the socket properly. Otherwise it can happen that you eat up all available connections.
    - If you can not guaranty that the socket will be closed properly the best solution is to use always the same (client-)port number.
    - You can also change your system-settings to let the OS handle it (see tcp_keepalive_time).

- 20.9.18: The .exe always try hasfhvctrl.desy.de first. So start with additional parameter via the commandline.
  AdvancedIDServer.exe hasfcpuexp2.desy.de

top

Short presentation by Fini

- Here a short summary from Fini - held in the groupmeeting Nov 2014