

---

## Table of Contents

简介	1.1
程序安装	1.2
固件升级	1.3
工作原理	1.4
程序运行	1.5
数据结构	1.6
示例教程	1.7

## 简介

DT5730 DPP-PSD 固件有 List、Oscilloscope、Mixed 三种模式，List 模式只记录能量时间，但是其 PSD 功能无法使用，Oscilloscope 模式只能记录波形，Mixed 模式拥有 cfd、QSD 功能，但是缺点时必须记录波形，不想记录波形时候我们可以设置成记录最小波形长度来减少数据量的输出。因此根据我们的测量需求，在当前硬件及固件能力下，我们只能选择 Mixed 模式。

© Hongyi Wu

*updated:* 2019-01-19 14:51:38

## 程序安装

- [ROOT 软件安装](#)
- [CAEN Lib](#)
  - [CAENVMELib](#)
  - [CAENComm](#)
  - [CAENDigitizer](#)
  - [CAENUpgrader](#)
- [检查 CAENUpgrader 安装](#)
- [USB 驱动](#)
- [A2818驱动](#)
- [A3818驱动](#)

本获取经过 Scientific Linux 7 系统测试。建议采用 CentOS 7 或者 Scientific Linux 7。

- 本获取要求 CERN ROOT 6，要求必须安装 FFTW3 库。
- 要求电脑的内存配置较大，建议 16 GB 或者 32 GB 或者以上。

用户可通过 **yum** 安装 **fftw3**，也可自行下载源代码安装。以下为 **CentOS/Scientific Linux** 下 **yum** 安装的命令：

```
yum -y install fftw.x86_64 fftw-devel.x86_64 fftw-libs.x86_64
```

以下为可选安装，安装之后 **ROOT** 可使用功能更多。根据需要安装即可。

```
yum -y install lz4.x86_64 lz4-devel.x86_64
yum -y install gsl.x86_64 gsl-devel.x86_64
yum -y install graphviz.x86_64 graphviz-devel.x86_64
yum -y install ruby.x86_64 ruby-devel.x86_64 ruby-libs.x86_64
yum -y install expect.x86_64 expect-devel.x86_64
yum -y install davix.x86_64 davix-devel.x86_64
yum -y install unuran.x86_64 unuran-devel.x86_64
yum -y install avahi-compat-libdns_sd.x86_64 avahi-compat-libdns_sd-devel.x86_64
yum -y install ftgl.x86_64 ftgl-devel.x86_64
yum -y install glew.x86_64 glew-devel.x86_64
yum -y install mysql++.x86_64 mysql++-devel.x86_64
yum -y install cfitsio.x86_64 cfitsio-devel.x86_64
yum -y install libxml2*
yum -y install binutils-devel.x86_64
yum -y install pythia8.x86_64 pythia8-devel.x86_64
yum -y install redhat-lsb.x86_64
yum -y install R.x86_64
yum -y install R-RInside.x86_64 R-RInside-devel.x86_64 R-Rcpp.x86_64 R-Rcpp-devel.x86_64
```

## ROOT 软件安装

以下示例演示如何将 **ROOT** 安装到 **/opt** 目录下

```
## https://root.cern.ch/building-root
# cmake安装方法,以 6.08.06 为例。 安装之后 .bashrc 中添加 source /opt/root60806/bin/thisroot.sh
tar -zxvf root_v6.08.06.source.tar.gz
mkdir buildroot60806
cd buildroot60806/
cmake -DCMAKE_INSTALL_PREFIX=/opt/root60806 -Dall=ON ../root-6.08.06/
make -j8
make install
rm -rf buildroot60806
```

## CAEN Lib

本程序依赖 CAENVMELib/CAENComm/CAENDigitizer/CAENUpgrader 四个库文件。

其中 CAENVMELib/CAENComm/CAENDigitizer 为获取运行必须的库。CAENUpgrader 用来更新固件。

### CAENVMELib

```
tar -xvzf CAENVMELib-2.50.tgz
cd CAENVMELib-2.50/lib
sh install_x64      #需要ROOT权限
```

### CAENComm

```
tar -xvzf CAENComm-1.2-build20140211.tgz
cd CAENComm-1.2/lib
sh install_x64      #需要ROOT权限
```

### CAENDigitizer

```
tar -zxvf CAENDigitizer_2.11.0_20180212.tgz
sh install_64      #需要ROOT权限
```

### CAENUpgrader

```
tar -xvzf CAENUpgrader-1.6.3-build20170511.tgz
cd CAENUpgrader-1.6.3
./configure
make
make install      #需要ROOT权限
```

## 检查 CAENUpgrader 安装

安装后在终端中输入

```
CAENUpgraderGUI
```

将会弹出 CAEN Upgrader GUI 的图形界面。

## USB 驱动

如果您使用 USB，则需要安装 USB 驱动。

```
tar -xvzf CAENUSBdrvB-1.5.2.tgz
cd CAENUSBdrvB-1.5.2
make
make install      #需要ROOT权限
```

## A2818驱动

如果您使用 A2818，则安装以下驱动。

```
# A2818Drv-1.20-build20161118.tgz
#将该文件夹复制到 /opt 并安装在该位置
tar -zxvf A2818Drv-1.20-build20161118.tgz
cp -r A2818Drv-1.20 /opt          #需要ROOT权限
cd /opt/A2818Drv-1.20            #需要ROOT权限
cp ./Makefile.2.6-3.x Makefile   #需要ROOT权限
make                             #需要ROOT权限

#设置开机自动执行该脚本
#在文件 /etc/rc.d/rc.local 中添加以下一行内容
/bin/sh /opt/A2818Drv-1.20/a2818_load
#或者在开启电脑之后执行以上命令
```

重启电脑后，在终端内输入 `dmesg|grep a2818` 将会看到以下的A2818驱动加载信息

```
a2818: CAEN A2818 CONET controller driver v1.20s
a2818: Copyright 2004, CAEN SpA
pci 0000:05:02.0: enabling device (0000 -> 0003)
pci 0000:05:02.0: PCI INT A -> GSI 19 (level, low) -> IRQ 19
a2818: found A2818 adapter at iomem 0xf7800000 irq 0, PLX at 0xf7900000
a2818: CAEN A2818 Loaded.
a2818: CAEN A2818: 1 device(s) found.
```

## A3818驱动

如果您使用 A3818，则安装以下驱动。安装该驱动时，电脑机箱必须插入 A3818 卡，否则将会报安装失败。

```
tar -zxvf A3818Drv-1.6.1.tgz
cd A3818Drv-1.6.1
make
make install          #需要ROOT权限
```

然后在终端内输入 `dmesg` 将会看到以下的A3818驱动加载信息

```
fuse init (API version 7.14)
CAEN A3818 PCI Express CONET2 controller driver v1.6.0s
Copyright 2013, CAEN SpA
pci 0000:02:00.0: PCI INT A -> GSI 16 (level, low) -> IRQ 16
alloc irq_desc for 33 on node -1
alloc kstat_irqs on node -1
pci 0000:02:00.0: irq 33 for MSI/MSI-X
pci 0000:02:00.0: setting latency timer to 64
Found A3818 - Common BAR at iomem fffff900067d4000 irq 0
Found A3818 with 1 link(s)
found A3818 Link 0 BAR at iomem fffff900067d6000 irq 0
CAEN A3818 Loaded.
CAEN PCIe: 1 device(s) found.
```

需要注意的是每次启动电脑之后均需要重新加载 **A3818** 驱动



# 固件

- [当前固件版本](#)
- [查看固件版本](#)
  - [A2818](#)
  - [A3818](#)
  - [DT5730](#)

[warning] 注意

请确保所使用的插件固件版本与以下一致。

我们尽可能保证采用最新的固件。

## 当前固件版本

```
DT5730 DPP-PSD 4.17_136.16
A2818 1.00
A3818 0.05
```

## 查看固件版本

DT5730/A2818/A3818 查看固件版本采用 CAENUpgraderGUI 程序，DT5730/A2818/A3818 升级固件版本同样采用 CAENUpgraderGUI 程序。即在终端中执行

```
CAENUpgraderGUI
```

升级固件时候，Browse 选择固件之后会弹出一个警告窗口，提示你“You have chosen to use a raw binary file”，点击确认，然后点击右下角的 Upgrade。等待升级结束，将会会有一个窗口提示你重启。

### A2818

如下图，查看 A2818 的固件版本，点击 *Get Fw Rel* 按钮。

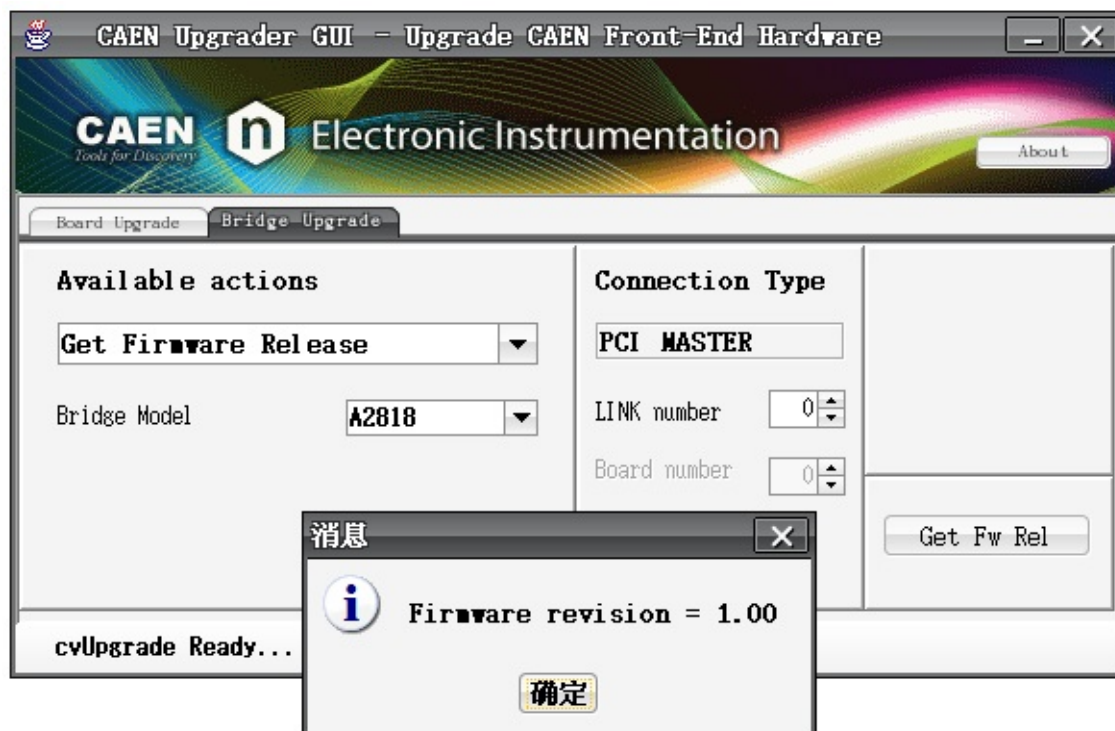


Figure: Get A2818 Version

如果该固件版本不是 当前固件版本 所列版本，则升级固件。

升级界面如下图所示：

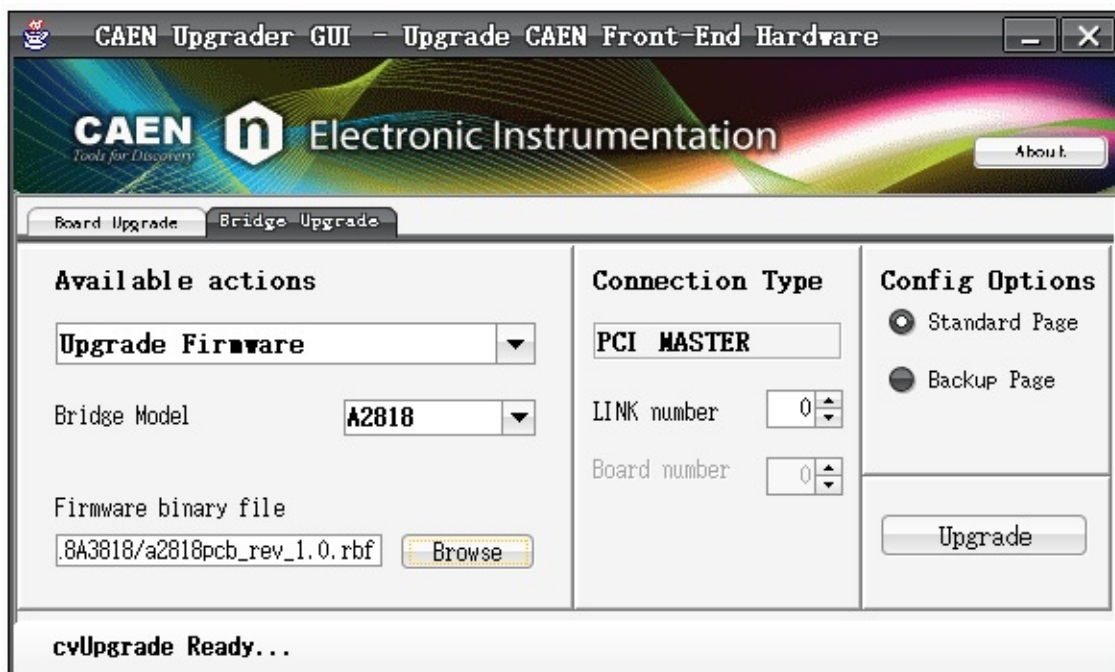


Figure: Update A2818



A3818

如下图，查看 A3818 的固件版本，点击 *Get Fw Rel* 按钮。

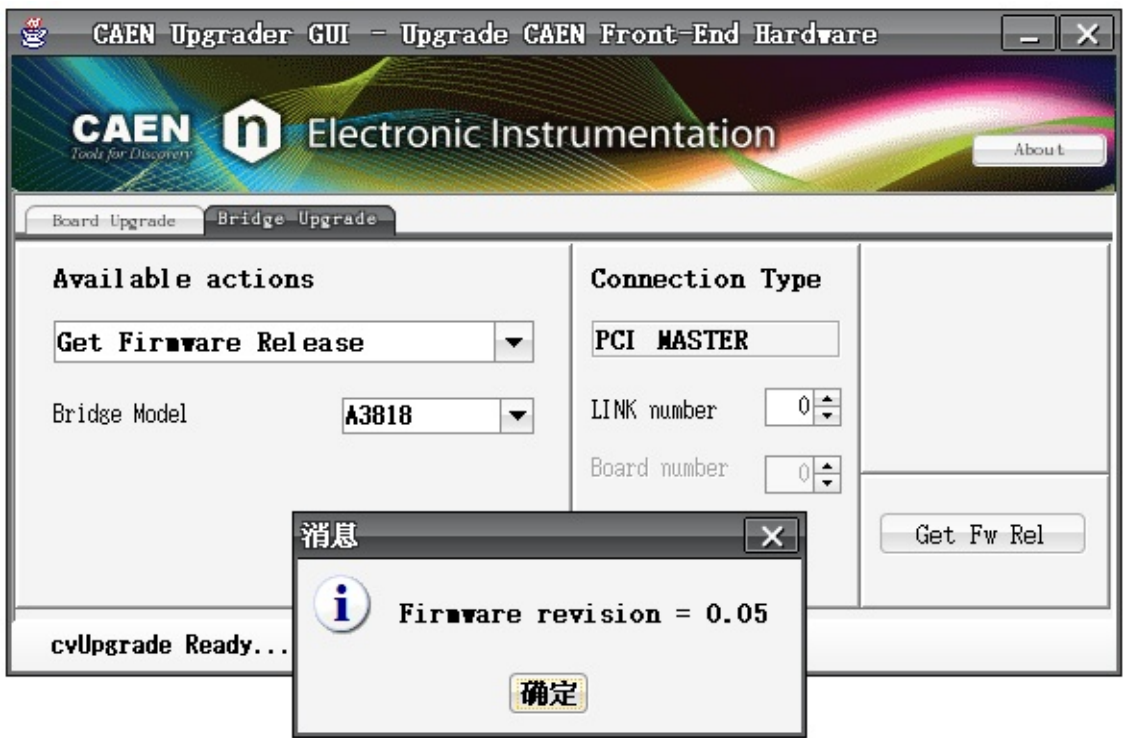


Figure: Get A3818 Version

如果该固件版本不是 当前固件版本 所列版本，则升级固件。

升级界面如下图所示：

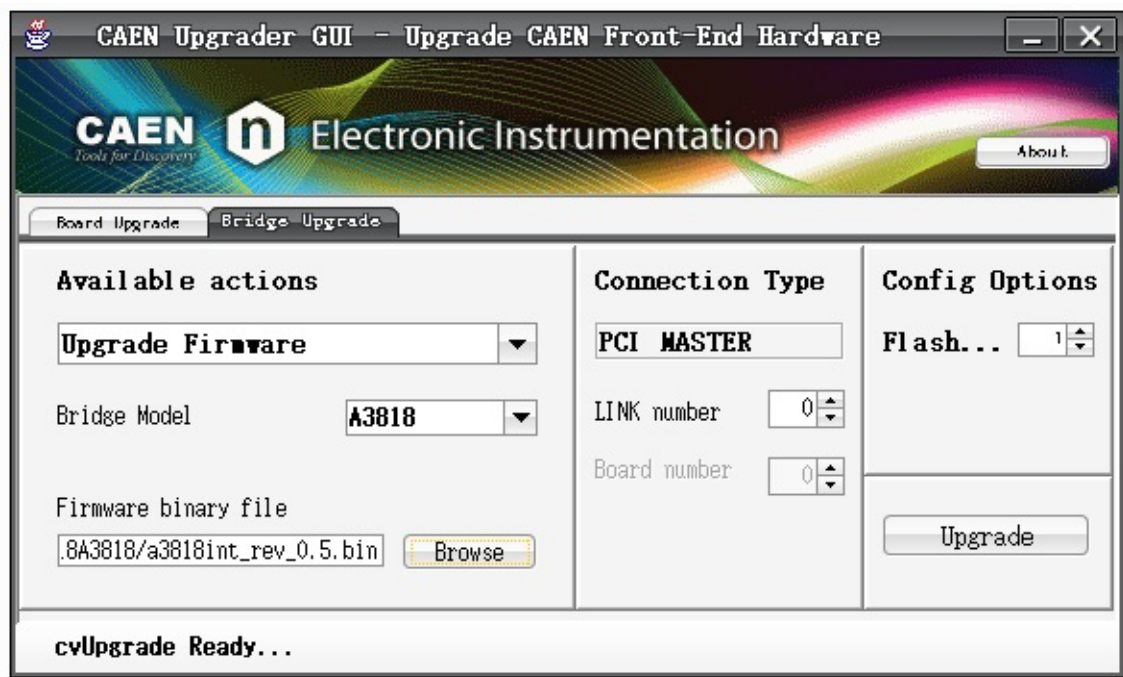


Figure: Update A3818

## DT5730

如下图，查看 DT5730 的固件版本，点击 *Get Fw Rel* 按钮。

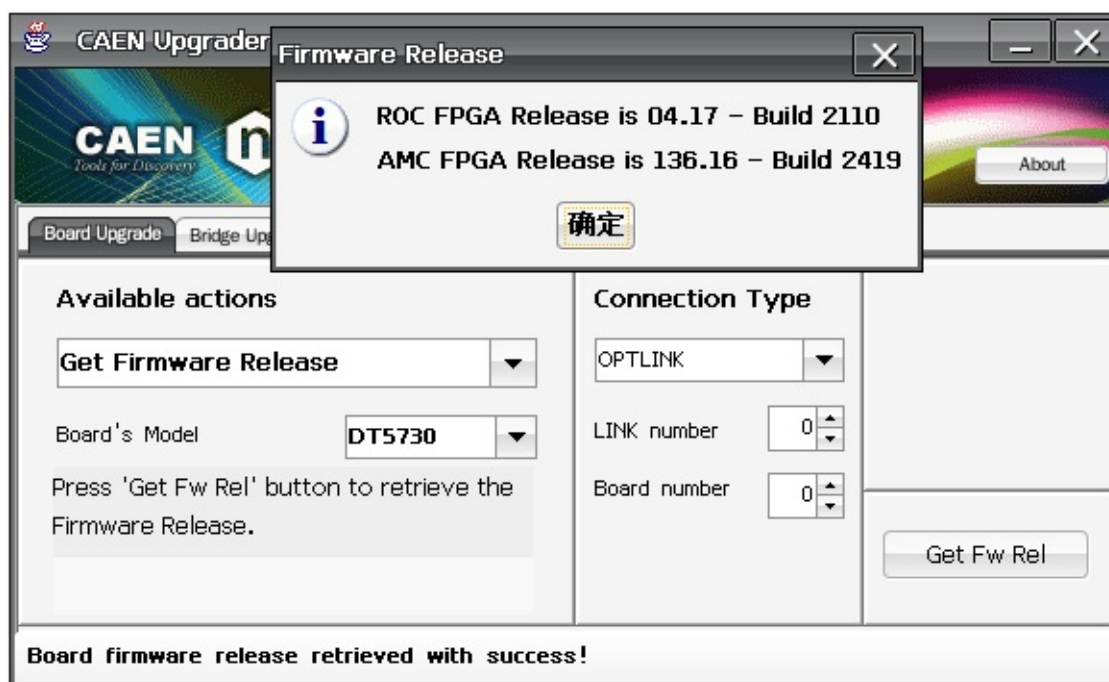


Figure: Get DT5730 Version

如果该固件版本不是 当前固件版本 所列版本，则升级固件。

升级界面如下图所示：

其中连接类型选项中，如果采用光纤连接，则选择 **OPTLINK**，如果采用 **USB** 连接，则选择 **USB** 选项。

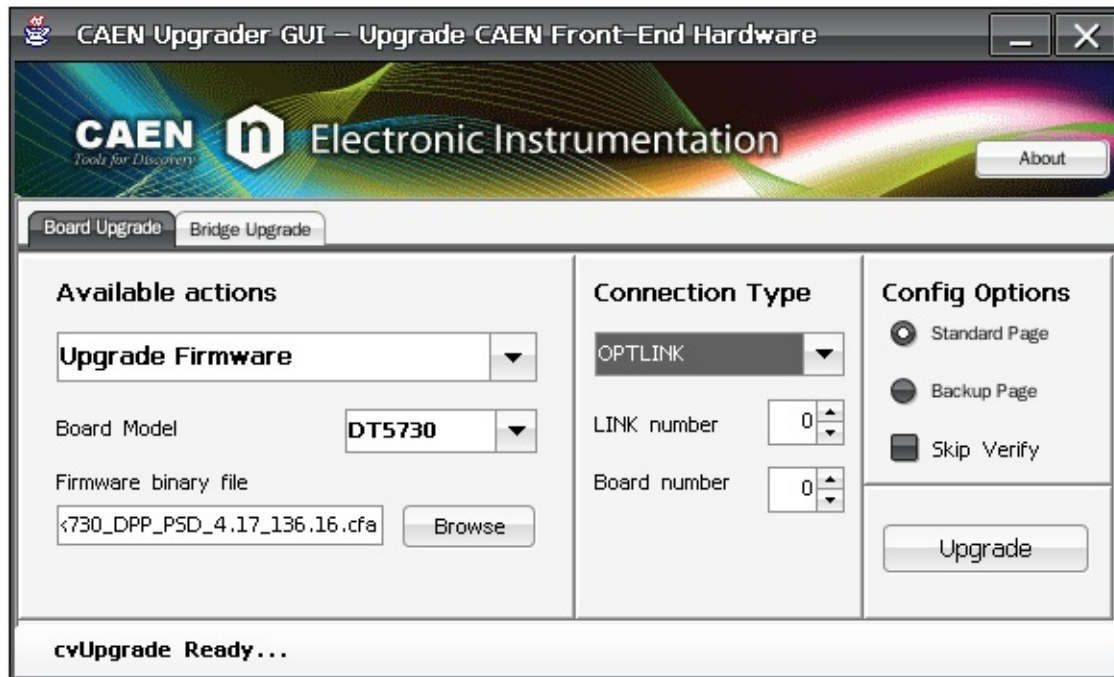


Figure: Update DT5730

工作原理

- 基线计算
- CFD 原理

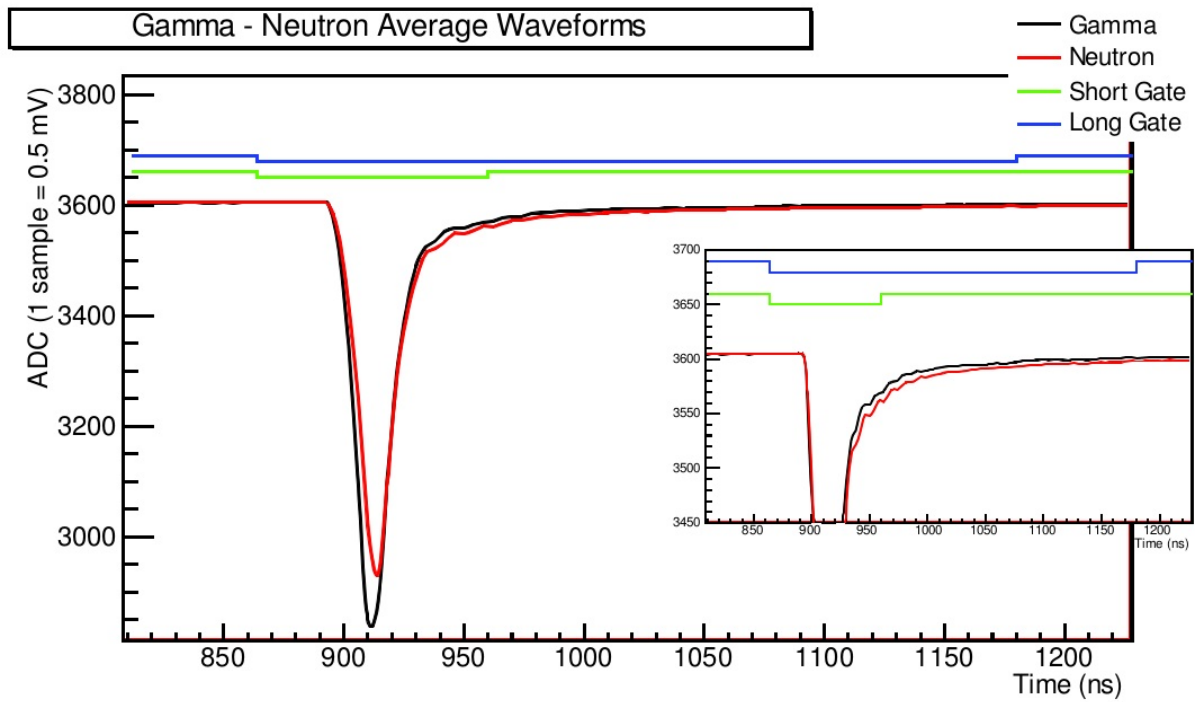


Figure: Typical Gamma Neutron Waveforms

下图为 DPP-PSD 固件的功能框图

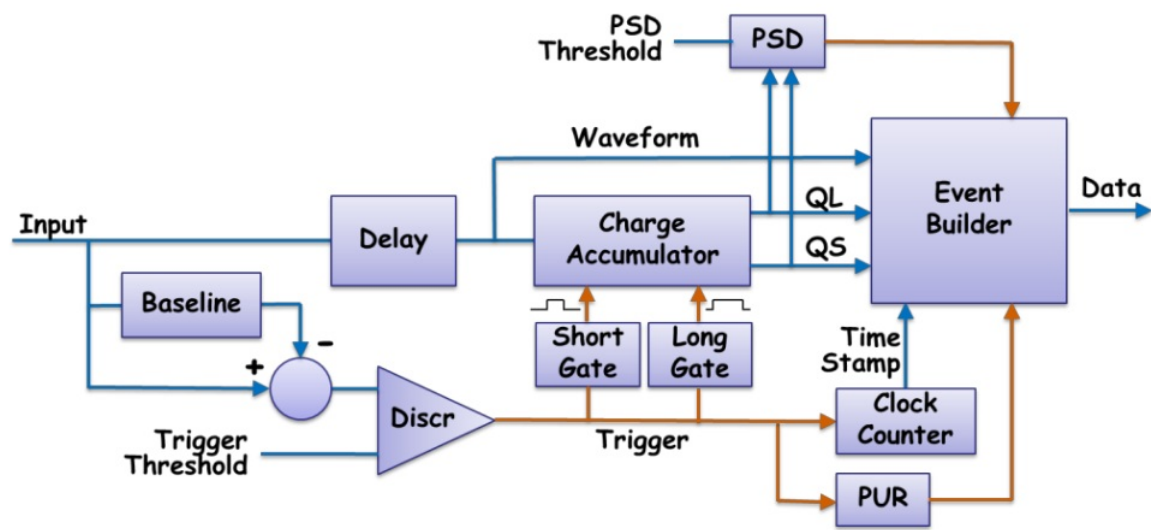


Figure: Functional Block Diagram Of The DPP-PSD

DPP-PSD 固件的目的是计算  $Q_{short}$  和  $Q_{long}$  的两个电荷，计算输入脉冲的双门积分。尾部电荷（慢成分）与总电荷之间的比率给出了用于伽马中子甄别的PSD参数：

$$PSD = (Q_{long} - Q_{short}) / Q_{long}$$

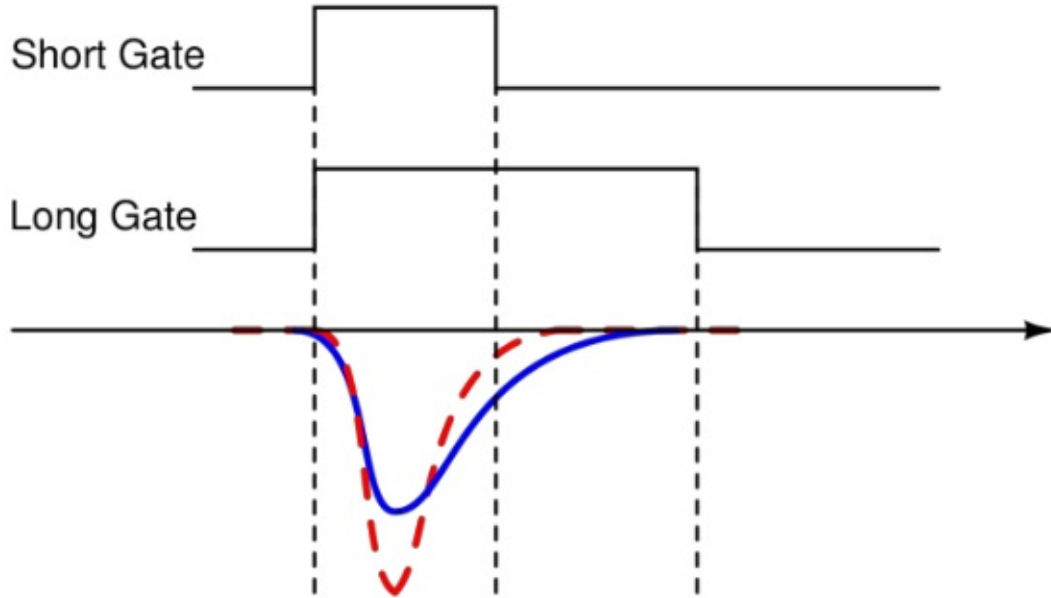


Figure: Long And Short Gate Graphic Position With Respect To A Couple Of Input Pulses

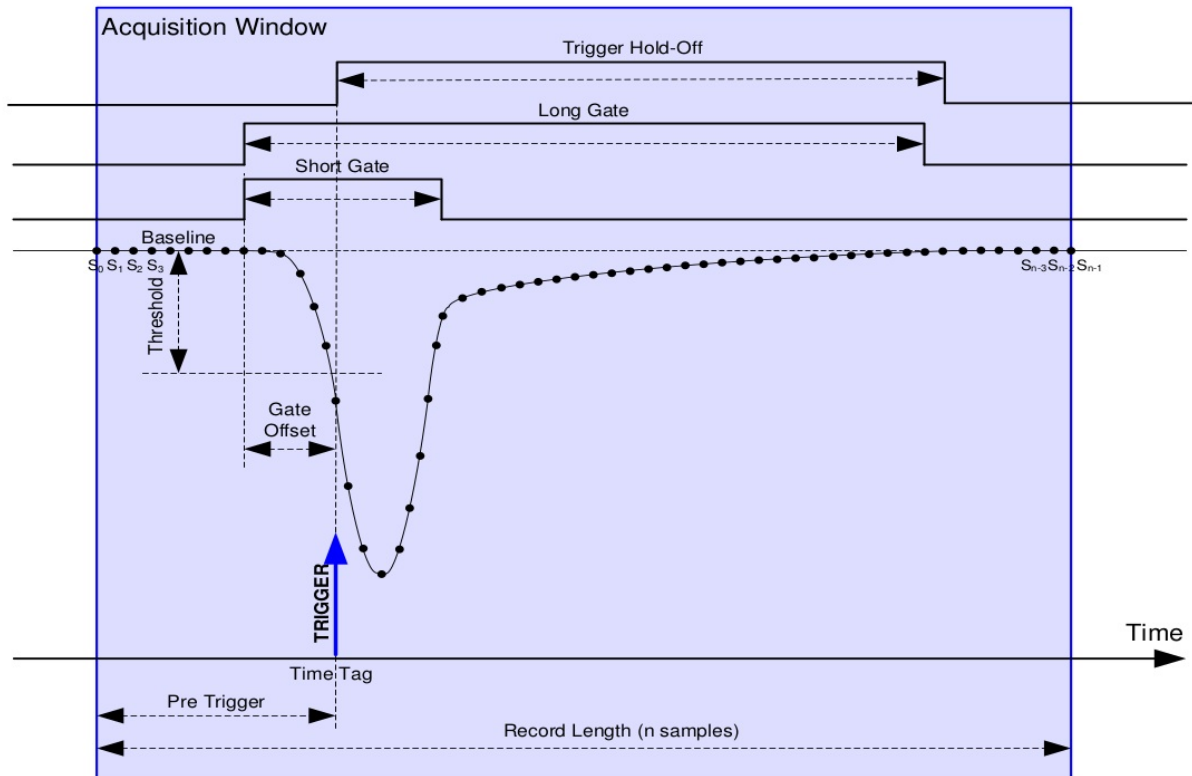


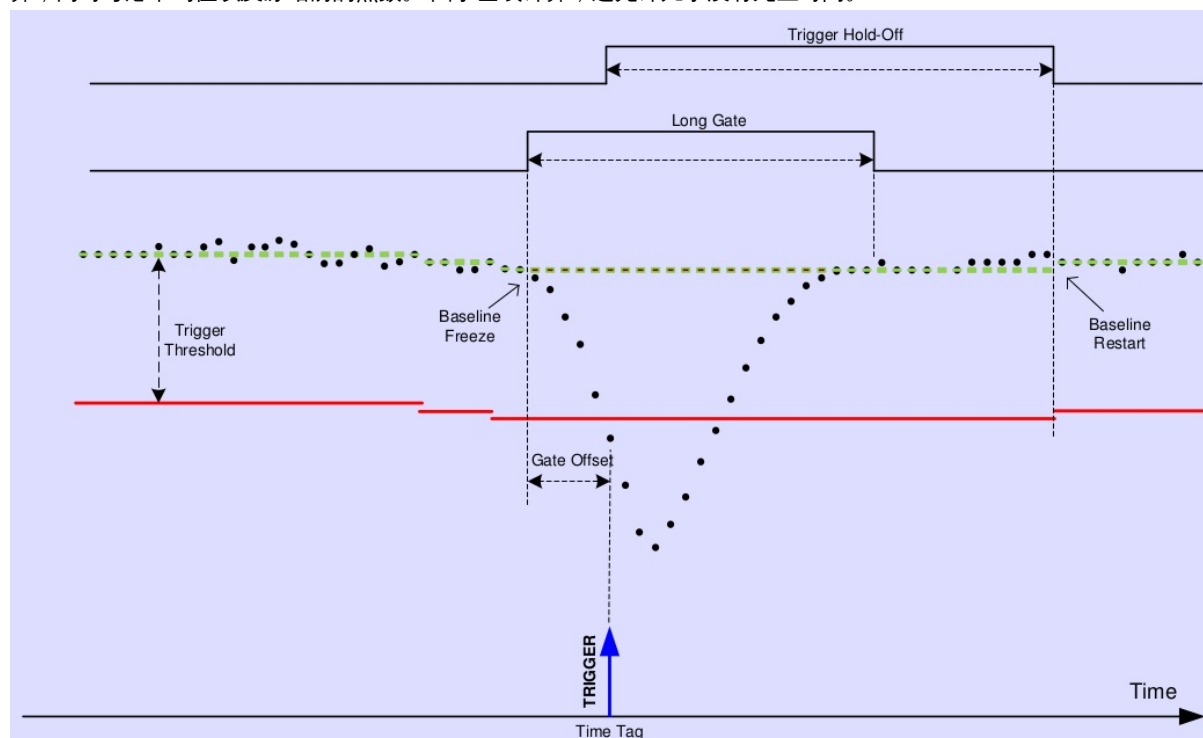
Figure: Diagram Summarizing The DPP-PSD Parameters

## 基线计算

基线计算是 DPP-PSD 固件的一个重要特性，因为它是输入脉冲电荷积分时的参考值。此外，大多数 DPP 参数与基线值相关。

用户可以选择将基线设置固定值，或让 DPP 固件动态计算它。在第一种情况下，用户必须以 LSB 单位设置基线值。对于整个采集运行期间，此值保持固定。在后一种情况下，固件动态地将移动时间窗口内的  $N$  个点的平均值作为基线评估。用户可以为 730 系列为选择 16,64,256,1024。

然后在门开始之前从几个时钟冻结基线的计算，直到长门和触发延迟之间的最大值结束。之后，基线再次重新开始计算，同时考虑平均值以及冻结前的点数。由于基线计算，这允许几乎没有死区时间。



## CFD 原理

在模拟电子学中，传统上使用 CFD（恒比定时甄别）模块来完成时间戳的确定。该技术提取振幅达到全振幅的固定分数的时间作为脉冲的时间戳。

数字 CFD 信号以经典方式实现。输入波形衰减的因子  $f$  等于全振幅的所需定时分数，然后信号被反转并延迟时间  $d$ ，该时间  $d$  等于脉冲从恒定分数水平上升到脉冲峰值所需的时间。将最新的两个信号相加以产生双极性脉冲，CFD 与其零交叉点对应于输入脉冲的分数  $f$ ，该点作为时间触发。如下图所示

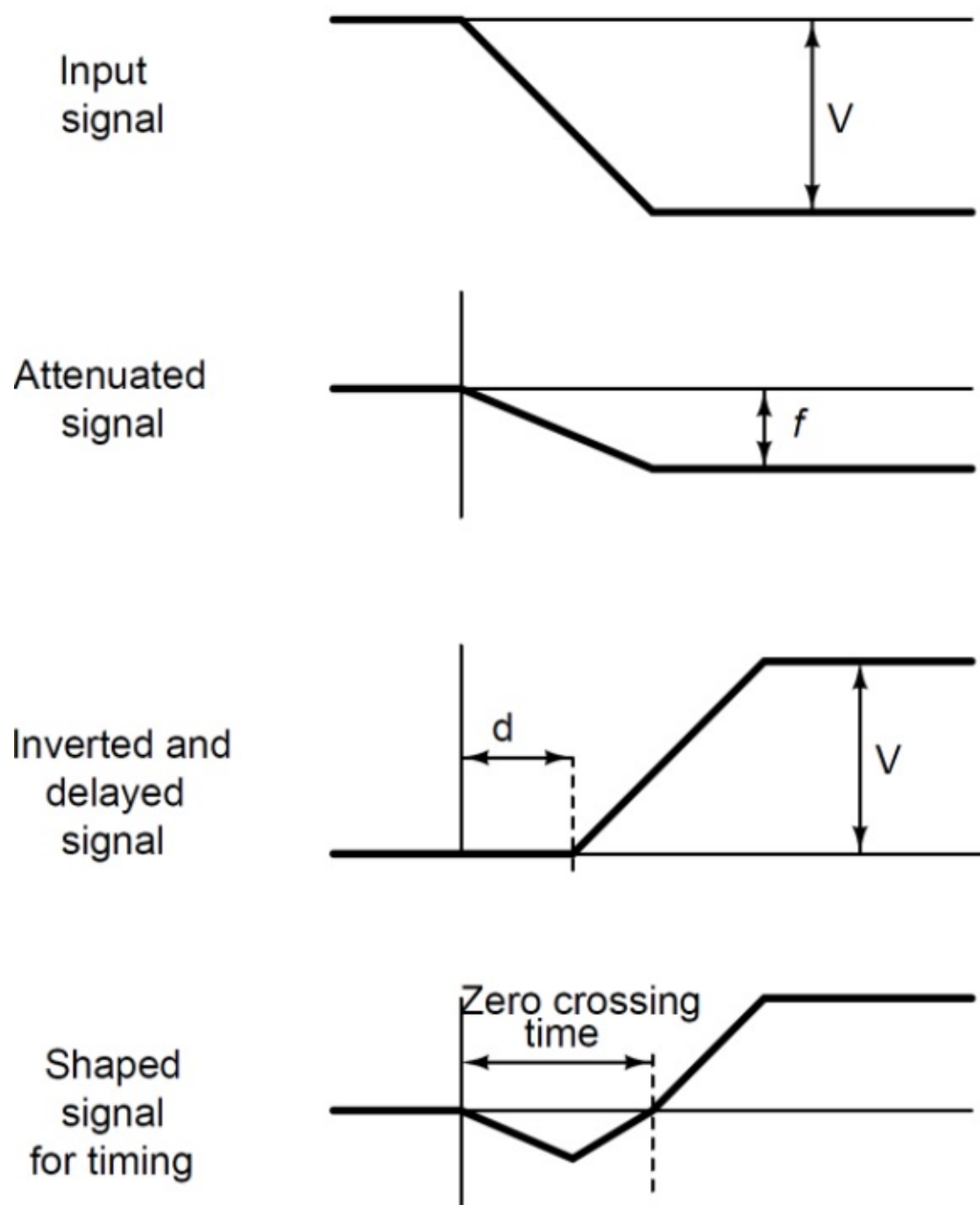


Figure: Classical Implementation Of The Constant Fraction Discriminator

CFD 在数字化中的实现如图所示。输入样本分为两个路径：第一个以采样时钟的步长执行延迟（2 ns），第二个执行衰减。衰减的可能选择是：相对于输入幅度的 25%，50%，75% 和 100%（即无衰减）。CFD 信号被称为动态的中等尺度，即在 730 系列的情况下为道址 8192。

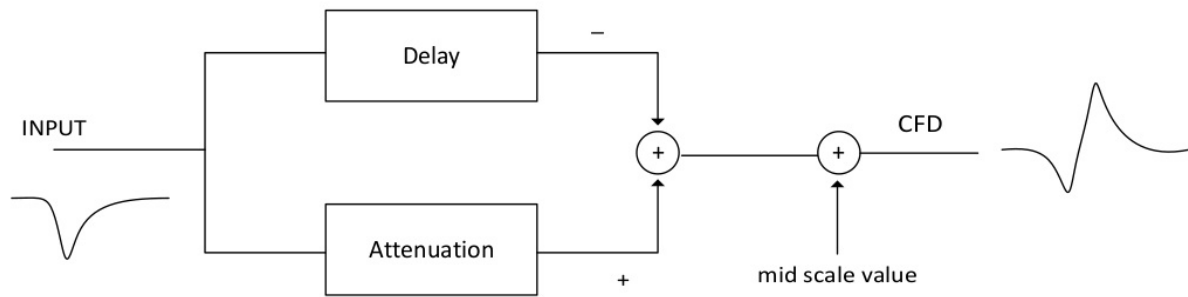


Figure: Implementation Of The Digital CFD In The DPP-PSD Firmware Of 730 Series

来自 CFD 的典型信号如下图所示，其中红点是数字采样点。我们这里定义：过零点前的采样点（SBZC）和过零后的采样点（SAZC）。SBZC 对应于粗略时间戳（Tcoarse），即由标准 PSD 算法评估的触发时间戳。精细时间标记 T 的值，根据公式计算为 SBZC 和 SAZC 的线性插值：

$$T_{\text{fine}} = (\text{midScale} - \text{SBZC}) / (\text{SAZC} - \text{SBZC}) \times T_{\text{sample}}$$

则过零点的时间戳由粗略时间戳加上精细时间戳：AC = Tcoarse + Tfine

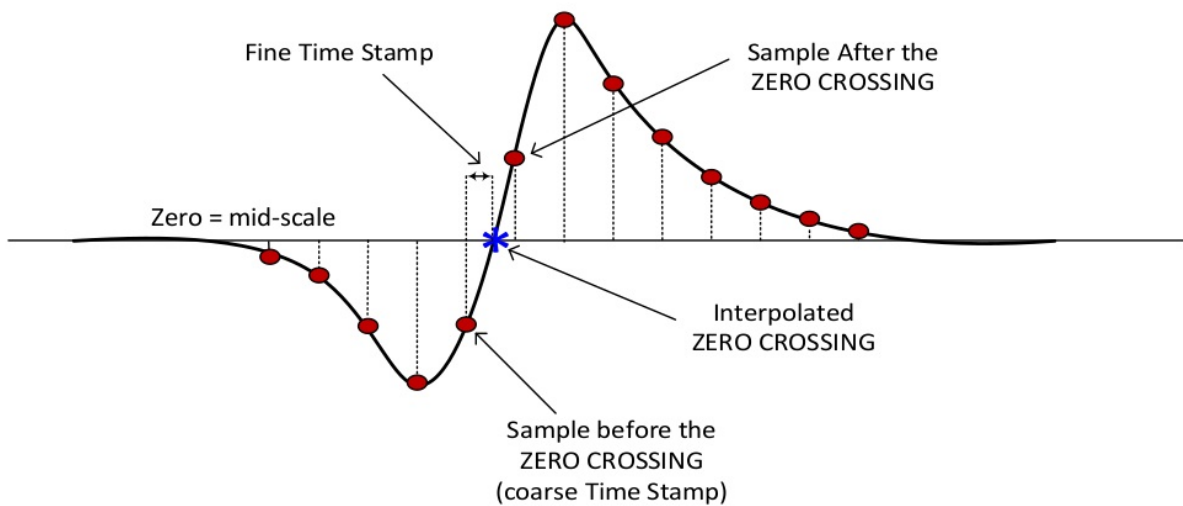


Figure: A typical CFD Signal



## 程序运行

- [程序编译](#)
- [二进制转ROOT](#)
- [按时间排序](#)
- [输入卡GlobalParameters](#)
- [输入卡BoardParameters](#)

## 程序编译

根据编译器版本不同（关键在于编译ROOT时候是否支持C++11，gcc4.8及以上），需要修改CMakeLists.txt文件中的以下内容：

```
## g++ 版本小于 4.8时
##C99 ROOT不支持C++11采用以下两行
set(CMAKE_CXX_FLAGS " -fPIC -W -Wall -s")#
set(CMAKE_C_FLAGS " -fPIC -W -Wall -s")#

## g++ 版本大于 4.8时
##C++11 ROOT支持C++11采用以下两行
set(CMAKE_CXX_FLAGS "-std=c++11 -fPIC -W -Wall -s")#
set(CMAKE_C_FLAGS "-std=c++11 -fPIC -W -Wall -s")#
```

修改好CMakeLists.txt文件之后。

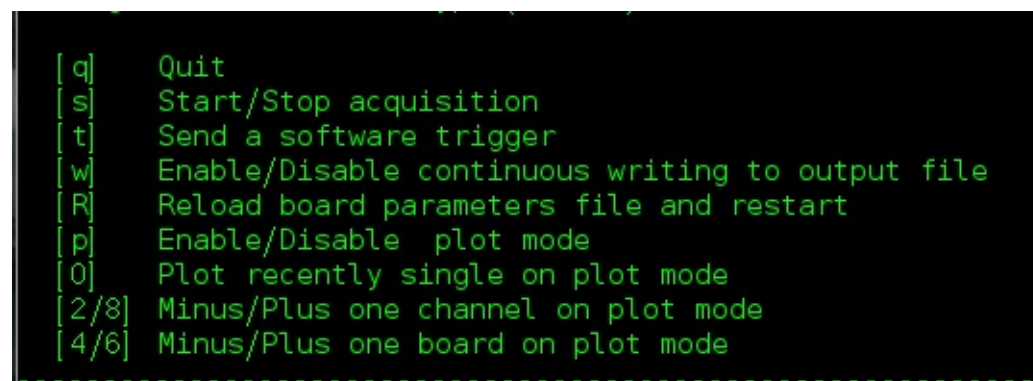
```
# 进入 build 文件夹
cd build

# 执行 cmake 生成 Makefile 文件
cmake ..

# 执行 make 编译成可执行文件
make

# 生成的可执行文件在 bin 文件夹内
cd ../bin
./pkuDigitizer
```

程序启动后，将会显示以下内容：

A terminal window with a black background and green text. It displays a menu of commands for a program. The commands are listed with their corresponding keyboard shortcuts in brackets. The menu is as follows:  
[q] Quit  
[s] Start/Stop acquisition  
[t] Send a software trigger  
[w] Enable/Disable continuous writing to output file  
[R] Reload board parameters file and restart  
[p] Enable/Disable plot mode  
[0] Plot recently single on plot mode  
[2/8] Minus/Plus one channel on plot mode  
[4/6] Minus/Plus one board on plot mode  
The text is aligned to the left, and there is a dashed line at the bottom of the terminal window.

```
[q] Quit
[s] Start/Stop acquisition
[t] Send a software trigger
[w] Enable/Disable continuous writing to output file
[R] Reload board parameters file and restart
[p] Enable/Disable plot mode
[0] Plot recently single on plot mode
[2/8] Minus/Plus one channel on plot mode
[4/6] Minus/Plus one board on plot mode
-----
```

Figure: 程序启动

- 输入`q`退出程序。
- 输入`s`来切换启动/关闭获取。
- 输入`t`表示给一个外部触发信号。
- 输入`w`来切换是否写文本。
- 输入`R`重新读取获取参设。
- 输入`p`来切换是否打开图形监视波形。
- 输入数字`0`表示在监视界面画出监视路最近的一个波形。
- 输入数字`2, 4, 6, 8`用来改变监视路。

获取开启，将会向`run.log`写入开始时间，获取关闭时也会写入结束时间。当开启写数据模式时，先读取`Log`文件夹下的`RunNumber`文件中的数值为当前的运行编号（用在数据文件命名），并使该数值加一保存。在`Log`文件夹下生成当前时刻命名的文件夹(例如20190118150912)，文件夹内复制保存当前获取所用输入卡。并将当前运行编号写进`run.log`。

开启写文本模式时候，获得当前运行编号 `N`，第一次文件名为 `runN_0`,文件大小达到 2 GB 自动保存，打开 `runN_1`，依次类推。

```
run0001_0 run0001_1
run0125_0
run6241_0 run6241_1 run6241_2
```

```
wuhongyi@ScientificLinux: ~/DT5730/code/bin
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

[q] Quit
[s] Start/Stop acquisition
[t] Send a software trigger
[w] Enable/Disable continuous writing to output file
[R] Reload board parameters file and restart
[p] Enable/Disable plot mode
[0] Plot recently single on plot mode
[2/8] Minus/Plus one channel on plot mode
[4/6] Minus/Plus one board on plot mode

-----
Readout Rate=12.81 MB

Board 0:
  Ch 0:  TrgRate=10.969 kHz      PileUpRate=0.00%
  Ch 1:  No Data
  Ch 2:  TrgRate=10.966 kHz      PileUpRate=0.01%
  Ch 3:  No Data
  Ch 4:  TrgRate=10.966 kHz      PileUpRate=0.00%
  Ch 5:  No Data
  Ch 6:  No Data
  Ch 7:  TrgRate=10.966 kHz      PileUpRate=0.02%

Status:
Start acquisition, Not Write .....
```

Figure: 获取开启界面

```
wuhongyi@ScientificLinux:~/DT5730/code/bin
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[q] Quit
[s] Start/Stop acquisition
[t] Send a software trigger
[w] Enable/Disable continuous writing to output file
[R] Reload board parameters file and restart
[p] Enable/Disable plot mode
[0] Plot recently single on plot mode
[2/8] Minus/Plus one channel on plot mode
[4/6] Minus/Plus one board on plot mode
-----
Readout Rate=12.81 MB
Board 0:
  Ch 0:  TrgRate=10.963 kHz      PileUpRate=0.00%
  Ch 1:  No Data
  Ch 2:  TrgRate=10.966 kHz      PileUpRate=0.00%
  Ch 3:  No Data
  Ch 4:  TrgRate=10.963 kHz      PileUpRate=0.00%
  Ch 5:  No Data
  Ch 6:  No Data
  Ch 7:  TrgRate=10.963 kHz      PileUpRate=0.00%
Status:
Start acquisition, Writing file Number: 115
```

Figure: 数据写入界面

## 二进制转ROOT

文件夹 **analysis** 内程序 **raw2root** 用来将输出的二进制文件转成 ROOT 文件。

需要先修改 **main.cc** 中原始数据的路径：

```
char filepath[1024] = "../data"; // 不要以 / 结尾
```

这里修改指向数据文件夹的路径。

具体运行：

```
make          #编译
./raw2root
```

之会提示您输入需要转的文件最小编号跟最大编号。例如我要转文件编号 0000 到 0120 的文件，只需要输入 0 空格 120，然后回车即可。如果里面某些编号文件不存在会自动跳过。同一个运行编号的几个子文件会存在一个 ROOT 文件中，例如 **run0100\_0,run0100\_1,run0100\_2** 数据会转成 **run0100.root**。

## 按时间排序

文件夹 **analysis** 内程序 **timesort** 用来将 ROOT 文件中的事件按照时间戳从小到大进行排序。

需要先修改 **main.cc** 中原始数据的路径：

```
TString filepath = "../../../data";//不要以 / 结尾
```

这里修改指向数据文件夹的路径。

具体运行：

```
make          #编译
./timesort
```

之会提示您输入需要转的文件最小编号跟最大编号。例如我要转文件编号 0000 到 0120 的文件，只需要输入 0 空格 120，然后回车即可。

## 输入卡GlobalParameters

在输入卡**GlobalParameters.txt**中

```
# PathToRawData 后面填写数据文件存放文件夹路径。
# PlotChooseN 后面填写监视路每多少个信号更新一次。该参数仅在开启波形监视时生效。

PathToRawData ../data
PlotChooseN 1000
```

```
# 当采用 USB 通讯时，参数按照以下设置
LinkType CAEN_DGTZ_USB
VMEBaseAddress 0

# 当采用光纤通讯时，参数按照以下设置
LinkType CAEN_DGTZ_PCI_OpticalLink
VMEBaseAddress 0
```

当前输入卡中的其余参数，当前请勿修改。

## 输入卡BoardParameters

在输入卡**BoardParameters.txt**中对每个 channel 的参数进行设置。

```
[RecordLength]
144

#波形记录采样点个数，最小可设置参数为??，数值需要为??的倍数
```

```
[ChannelMask]
11111111

# 通道是否开启的标记，从右往左依次为0-7通道，标记为1表示开启该通道，标记为0表示不启用该通道
```

```
[ChannelDynamicRange]
00000000
```

```
# 通道输入动态范围，从右往左依次为0-7通道，标记0表示2V动态范围，标记1表示0.5V动态范围。
```

```
[TriggerHoldOff]
8
```

```
# 触发保护，在设置时间内，不会接受其它触发，该数值单位为 2 ns
```

其余参数是每个通道单独设置。

以下是所有通道采用同样的参数设置的例子

Channel	POL	Offset	PreTrg	thr	selft	csecs	sgate	lgate	pgate	tvaw	nsbl	discr	cdfd	cfdd	trgc
[COMMON]															
ALL	NEG	-43	30	50	1	0	24	100	8	50	1	0	3	3	1
[INDIVIDUAL]															

以下是通道2/6单独设置，其余通道采用同样的参数设置的例子

Channel	POL	Offset	PreTrg	thr	selft	csecs	sgate	lgate	pgate	tvaw	nsbl	discr	cdfd	cfdd	trgc
[COMMON]															
ALL	NEG	-43	30	50	1	0	24	100	8	50	1	0	3	3	1
[INDIVIDUAL]															
2	POS	43	30	150	1	0	24	100	8	50	1	0	3	3	1
6	POS	43	30	150	1	0	24	100	8	50	1	0	3	3	1

这里重点调节的参数是

- POL 表示输入信号的极性，POS表示正信号，NEG 表示负信号
- Offset 表示偏置，这里提供一个经验参数，如果是负信号，则设置-45 左右，如果是正信号则设置45左右
- PreTrg 表示记录的波形中，触发之前的点的个数
- thr 表示阈值，该数值表示相对基线的道址
- csecs 用来调节增益
- sgate 短门积分门宽，每个点表示2 ns
- lgate 长门积分门宽，每个点表示2 ns
- pgate 积分门起始点相对于触发点提前多少时间，，每个点表示2 ns
- tvaw 当前无用
- nsbl 基线平滑，采用多少个点来平均
- discr 当前选择 0
- cdfd 表示cfd参数的比例因子，有四个档可选
- cfdd 表示cfd的延迟参数，每个点表示2 ns
- trgc 当前选择 1

© Hongyi Wu      updated: 2019-01-19 15:50:56

## 数据结构

- [原始数据](#)
- [一级 ROOT 文件](#)
- [时标排序](#)

## 原始数据

原始数据文件中包含通道编号、QDC 的短门积分、QDC 的长门积分、format 32 bit 数据（当前模式下包含 CFD 数据），波形。

对一个事件，数据依次为 16 bit 通道编号，32 bit 时间戳的低31位，16 bit QDC 的短门积分，16 bit QDC 的长门积分，32 bit 的 format(低 10 bit 的 CFD)，32 bit 的时间戳高 16 位，16 bit 的波形长度，紧接着是 n 个波形点，每个 16 bit。

## 一级 ROOT 文件

原始 ROOT 文件中包含以下 branch，

- ch 通道编号
- qs 短门积分
- ql 长门积分
- format 参数标记，无用
- ts 时间戳，每个数值表示 2 ns
- ft cfd数值，该数值除以 1024 然后乘以 2 之后，单位为 ns，ts 时间戳加上 cfd 时间即可得到亚 ns 时间精度
- size 表示记录的波形长度，即采集了多少个点
- wave 表示波形
- sample 加入的一个branch，为了方便查看波形

如下图，通过在 root 命令行中输入以下命令可查看该文件的 branch，

```
t->Print()
```



```

TFile**      run0102.root
TFile*       run0102.root
KEY: TTree   t;40    PKU Digitizer data
KEY: TTree   t;39    PKU Digitizer data
root [2] t->Print()
*****
*Tree       :t          : PKU Digitizer data *
*Entries    : 8581593 : Total =      5200904316 bytes File Size = 1115730458 *
*           :         : Tree compression factor = 4.66 *
*****
*Br    0 :ch          : ch/S *
*Entries : 8581593 : Total Size= 17168033 bytes File Size = 930045 *
*Baskets : 53 : Basket Size= 1844736 bytes Compression= 18.46 *
*.....*
*Br    1 :qs          : qs/S *
*Entries : 8581593 : Total Size= 17168033 bytes File Size = 14218263 *
*Baskets : 53 : Basket Size= 1844736 bytes Compression= 1.21 *
*.....*
*Br    2 :ql          : ql/S *
*Entries : 8581593 : Total Size= 17168033 bytes File Size = 14308526 *
*Baskets : 53 : Basket Size= 1844736 bytes Compression= 1.20 *
*.....*
*Br    3 :format      : format/i *
*Entries : 8581593 : Total Size= 34332711 bytes File Size = 173533 *
*Baskets : 67 : Basket Size= 1939968 bytes Compression= 197.84 *
*.....*
*Br    4 :ts          : ts/l *
*Entries : 8581593 : Total Size= 68661129 bytes File Size = 15361622 *
*Baskets : 94 : Basket Size= 2129920 bytes Compression= 4.47 *
*.....*
*Br    5 :ft          : ft/S *
*Entries : 8581593 : Total Size= 17168033 bytes File Size = 9915339 *
*Baskets : 53 : Basket Size= 1844736 bytes Compression= 1.73 *
*.....*
*Br    6 :size        : size/S *
*Entries : 8581593 : Total Size= 17168147 bytes File Size = 80268 *
*Baskets : 53 : Basket Size= 1844736 bytes Compression= 213.87 *
*.....*
*Br    7 :wave        : wave[size] /S *
*Entries :8581593 : Total Size= 2506032793 bytes File Size = 1019008401 *
*Baskets : 2159 : Basket Size= 25600000 bytes Compression= 2.46 *
*.....*
*Br    8 :sample      : sample[size] /S *
*Entries :8581593 : Total Size= 2506037119 bytes File Size = 41697308 *
*Baskets : 2159 : Basket Size= 25600000 bytes Compression= 60.10 *
*.....*
root [3] █

```

Figure: raw root data

如下图，通过在 root 命令行中输入以下命令可打印查看某事件的具体数据，

```

t->Show(0)

# 这里的 0 表示查看文件中的第0个事件
# 默认下，数组只显示前 20 个数据，需要如果想要查看更长的数据

t->Show(0,100)

# 后面的 100 表示数组最长打印前 100 个数据

```

```
root [3] t->Show(0)
=> EVENT: 0
ch          = 0
qs          = 484
ql          = 491
format      = 2046820370
ts          = 651313367507
ft          = 879
size        = 144
wave        = 15212,
              15211, 15206, 15205, 15208, 15208, 15214, 15210, 15210, 15209
, 15210,
              15207, 15208, 15210, 15210, 15208, 15208, 15211, 15208, 15213
sample      = 0,
              1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
              11, 12, 13, 14, 15, 16, 17, 18, 19
```

Figure: raw root data show

如下图，通过在 root 命令行中输入以下命令可画出某事件的波形，

```
t->Draw("wave:sample", "Entry$==0")
```

#这里的 0 表示文件中第 0 个事件



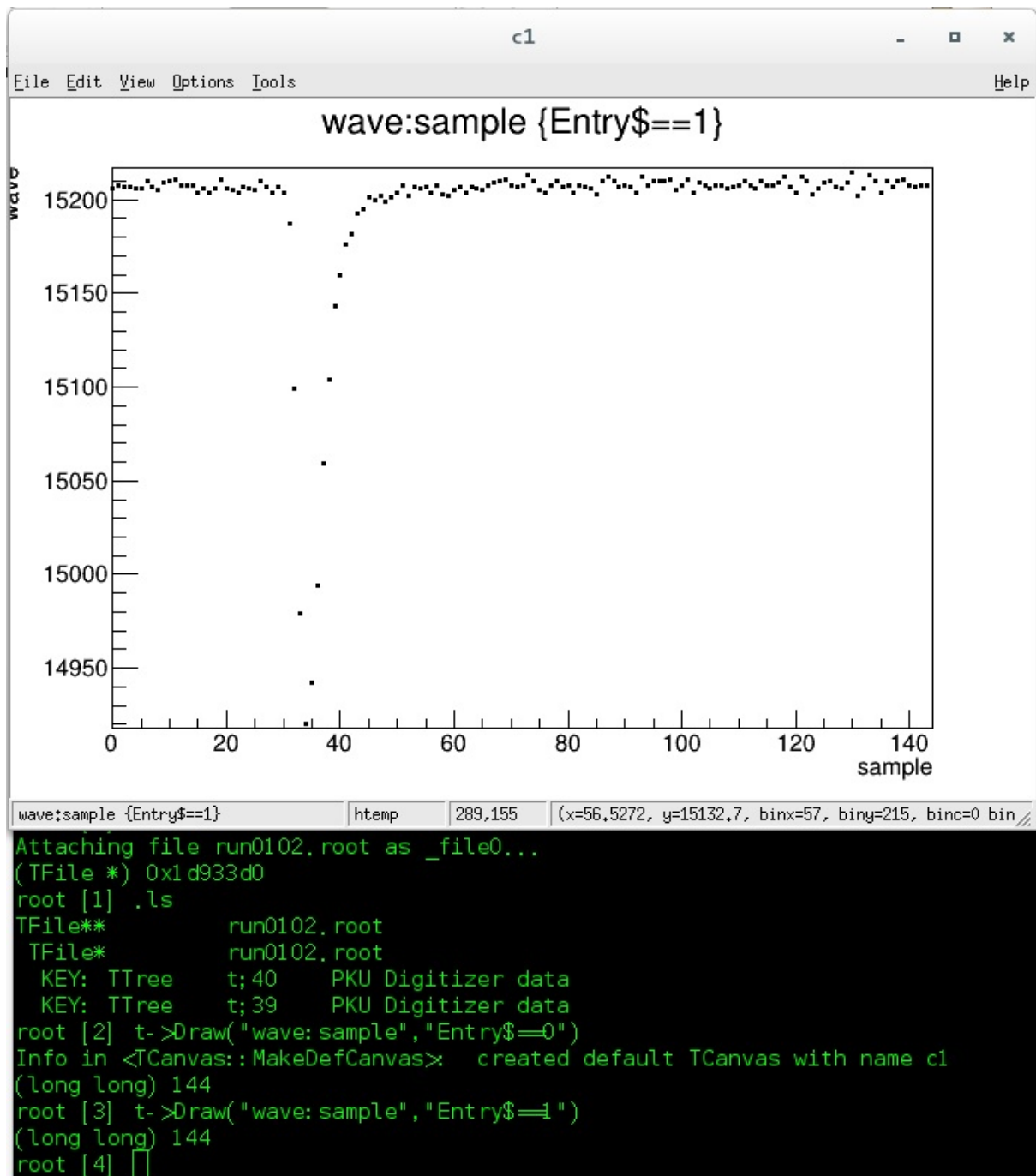


Figure: raw root data draw

## 时标排序

© Hongyi Wu

updated: 2019-01-19 14:34:25

## 示例教程

- [NIM 逻辑信号](#)
- [液闪探测器信号](#)
- [PSD 积分门宽优化](#)

## NIM 逻辑信号

## 液闪探测器信号

## PSD 积分门宽优化

© [Hongyi Wu](#)

*updated:* 2019-01-18 23:51:42