
GDDAQ

发布 *beta*

Hongyi Wu(吴鸿毅)

2019 年 09 月 21 日

1 简介	3
1.1 版本	3
1.1.1 稳定版本	3
1.1.2 准预览版本	3
1.2 关于	4
1.3 文件目录	5
1.4 升级计划	5
1.5 版权声明	5
2 程序安装	7
2.1 软件安装步骤	7
2.2 程序使用说明	9
3 用户指南	11
3.1 数据结构	13
4 机箱	17
4.1 远程控制	17
4.2 插槽	19
5 固件	21
6 数据解码	23
7 图形交互界面	29
7.1 主控制界面	30
7.2 File	31
7.2.1 About	32
7.3 Base	32
7.3.1 Base Setup	32
7.3.2 Trigger Filter	35
7.3.3 Energy	37
7.3.4 CFD	43
7.3.5 QDC	48
7.3.6 Decimation	49
7.3.7 Copy Pars	50
7.3.8 Save2File	51
7.4 Expert	52
7.4.1 Module Variables	52
7.4.2 CSRA	55
7.4.3 Logic Set	58

7.5	Debug	58
7.5.1	Hist & XDT	58
7.5.2	Trace & Baseline	59
7.6	Offline	61
7.6.1	InitData	62
7.6.2	Adjust Par	63
7.6.3	Wave-16	66
7.6.4	Energy-16	67
7.6.5	Orig Energy	67
7.6.6	Calc Energy	69
7.6.7	FF/CFD Thre	70
7.6.8	Energy-FF	72
7.6.9	Energy-CFD	73
7.6.10	QDC	73
7.6.11	FFT	74
7.6.12	Time Diff	75
7.6.13	Simulation(暂未实现)	75
8	非图形交互界面	77
8.1	控制界面	77
8.2	自动运行设置	78
9	在线统计	81
9.1	控制界面	81
10	MakeEvent	83
11	插件前面板	85
11.1	模拟信号输入连接器 (all revisions)	85
11.2	LVDS I/O 端口 (all revisions)	86
11.3	数字 I/O 连接器 (Rev. F only)	87
11.4	前面板 LEDs (all revisions)	89
11.5	3.3V I/O 连接器 (Rev. D only)	90
11.6	GATE Inputs(Rev. D only)	91
11.7	3.3V I/O 连接器 (Rev. B and C only)	92
11.8	标准固件中的数字信号 (all revisions)	92
12	逻辑功能	95
12.1	逻辑概括	95
12.2	Module Fast Trigger(for trigger)	97
12.3	Module Validation Trigger(for control logic)	97
12.4	Channel Validation Trigger(for trigger/control logic)	98
12.5	Veto	98
12.6	System FPGA (coincidence/multiplicity)	99
13	时间同步	103
13.1	独立时钟模式	105
13.2	PXI 时钟模式	106
13.3	菊花链时钟模式	107
13.4	多机箱时钟模式	110
13.4.1	Pixie-16 模块安装	111
13.4.2	Pixie-16 模块上的时钟跳线 (JP101) 设置	111
13.4.3	Pixie-16 背板 I/O 触发模块的电缆连接	111
13.4.4	Pixie-16 背板 I/O 触发模块上的跳线设置	113
14	升级 CPLD	117
14.1	6 pin JTAG connector	117
14.2	update the CPLD	118

15 应用案例	119
16 100M 模块测量飞行时间	121
16.1 TAC 信号特点	121
16.2 实验测量结果 (2017 RIBLL)	122
17 时间分辨	125
17.1 脉冲产生装置	125
17.2 100M 模块	125
17.2.1 2 channel in one module	126
17.2.2 2 channel in one crate	127
17.2.3 2 channel in different crate	127
17.3 250M 模块	127
17.3.1 2 channel in one module	127
17.3.2 2 channel in one crate	127
17.3.3 2 channel in different crate	127
17.4 100M & 250M 模块	127
17.4.1 2 channel in one crate	127
17.4.2 2 channel in different crate	127
18 推荐参数	129
18.1 HPGe	129
18.1.1 100M	129
18.1.2 250M	129
18.2 BGO	130
18.2.1 100M	130
18.3 Si	130
18.3.1 100M	130
18.4 LaBr ₃	131
18.4.1 250M	131
19 在束伽马谱学	133
19.1 实验控制界面	133
19.2 BGO 反康门宽	134
19.3 峰总比	135
20 采购推荐	137
20.1 采集卡	138
20.2 机箱及配套	138
20.3 逻辑模块	140
20.4 34pin 到 16SMB 转接线	140
21 开发者指南	141
22 XIA API	143
23 PKU Code	157
23.1 Decode	157
23.2 GUI	157
23.3 MakeEvent	159
23.4 OnlineStatics	159

Welcome to GDDAQ's guides.

CHAPTER 1

简介



English | 简体中文

1.1 版本

我们建议用户下载稳定版本

1.1.1 稳定版本

稳定版本 Version:2019.09.12

下载最新版本, 请点击: [PKUXIADAQ stable](#)

网页版说明书请访问: [English 简体中文](#)

- reStructuredText 版说明书: [README/](#)
- 离线网页版说明书: [docs/](#)
- pdf 版本说明书: [README_en.pdf](#) [README_ch.pdf](#)

1.1.2 准预览版本

准预览版本 Version:2019.09.12

程序下载请访问: [PKUXIADAQ](#)

网页版说明书请访问: [English 简体中文](#)

- 对本获取程序有任何的意见及建议(功能添加及改进), 欢迎给吴鸿毅 (wuhongyi@qq.com) 发邮件。
 - 我们将会尽快完善软件的中英文使用说明书, 当前主要以操作演示来讲解软件的使用。
-

1.2 关于

本说明书仅适用于 XIA LLC 的 Pixie-16 系列采集卡

- 本程序由 北京大学实验核物理组 开发。
- 最早的图形界面程序是基于 NSCL DDAS Nscope 开发。
- 特别感谢 谭辉 (XIA LLC) 对我们开发的支持。

技术指导:

- Zhihuan Li 李智焕
- Hui Tan 谭辉 (XIA LLC)
- Wolfgang Hennig(XIA LLC)

软件主要开发者:

- 2015 - 2016
 - Jing Li 李晶 (lijinger02@126.com)
- 2016 - now
 - Hongyi Wu 吴鸿毅 (wuhongyi@qq.com)

说明书主要撰写者:

- Diwen Luo 罗迪雯
- Hongyi Wu 吴鸿毅
- Xiang Wang 王翔

本程序的开发得到以下单位的支持:

- XIA LLC
 - 中国科学院近代物理研究所 (IMP)
 - 中国原子能科学研究院 (CIAE)
 - 香港大学 (HKU)
 - 山东大学 (威海)(SDU)
 - ...
-

本程序适用于 XIA Pixie16 系列采集卡, 支持 100/250/500 MHz 采集卡(具体支持型号可查看图形软件中的 File->About), 最大支持 8 个机箱同步运行, 即 1600+ 路信号同时采集。本程序包要求使用 CERN ROOT6 版本。要求采用 1920x1080 及以上分辨率显示屏。

本程序的设计兼容 100/250/500 MHz 采集卡的混合使用, 只需在 cfgPixie16.txt 添加相应采样率采集卡的固件位置即可, 程序在线能够自动识别采集卡类型并加载相应的固件。当前我们测试了绝大多数类型的采集卡, 因此默认可运行我们测试过类型的采集卡, 如要支持其它类型, 请联系 XIA LLC 获取对应固件或者联系吴鸿毅 (wuhongyi@qq.com)。

1.3 文件目录

用户使用程序包中包含以下文件/文件夹:

- Decode(将原始二进制数据转为 ROOT)
- docs(使用说明书, 网页版)
- **firmware(固件)**
 - firmware/firmware.md(历史各版本固件说明)
- GUI(图形软件)
- MakeEvent(事件重构程序, 可选)
- NOGUI(非图形软件)
- OnlineStatics(在线监视程序)
- parset(参数设置文件)
- PlxSdk.tar.gz(Plx9054 驱动)
- README(markdown 版本说明书)
- README.md(主页介绍)
- README.pdf(pdf 版本说明书)
- software(非标准驱动, 吴鸿毅修改)
- TestTool(开发者测试工具, 用户不需要!!!)

1.4 升级计划

- 当前基于 ROOT GUI 开发的主控制界面复杂度高, 用户修改难度大。其它用户不容易基于其发展适合自己的版本。
- 我们也在开发基于网页控制的获取在线/离线分析程序:
 - Django
 - ZeroMQ
 - JSROOT
 - ...

1.5 版权声明

CHAPTER 2

程序安装

本程序安装要求

- **CERN ROOT 6**

- GCC >= 4.8

- FFTW3

本程序测试过的系统包括 CentOS7 / Scientific Linux 7.2/7.3/7.4

危险: 图形界面程序与非图形界面程序不能同时运行!

图形界面程序与非图形界面程序不能同时运行!

图形界面程序与非图形界面程序不能同时运行!

2.1 软件安装步骤

- 删除个人目录下的老版本 PKUXIADAQ 文件夹
- 将本程序包解压缩到个人目录中 (\$HOME)
- 设置环境变量
- 编译 Plx9054 驱动
- 编译 pixie16 驱动 API(该 API 被吴鸿毅修改过, 非官方标准驱动)
- 编译图形化获取软件
- 编译非图形化获取软件
- 编译在线监视程序
- 编译数据转换程序
- 编译事件重构程序 (可选)

```
## 设置环境变量

# 在 .bashrc 文件中添加
export PLX_SDK_DIR=$HOME/PKUXIADAQ/PlxSdk

# 将 PKUXIADAQ.tar.gz(或者 PKUXIADAQ-master.tar.gz) 放置到 /home 下的个人目录下, 即 ~/ 位置
tar -zxvf PKUXIADAQ.tar.gz # 解压缩
或者
tar -zxvf PKUXIADAQ-master.tar.gz
mv PKUXIADAQ-master PKUXIADAQ

# 得到 PKUXIADAQ 目录
```

```
## 编译 Plx9054 驱动

# 打开新终端
cd ~
cd PKUXIADAQ/
rm -rf PlxSdk # 删除可能存在的未删除驱动, 如果没有该目录则不用执行该行命令
tar -zxvf PlxSdk.tar.gz
cd PlxSdk/PlxApi/
make clean
make
# 成功后你将会看到 Library "Library/PlxApi.a" built successfully

cd ../../Samples/ApiTest/
make clean
make
# 成功后你将会看到 Application "App/ApiTest" built successfully

cd ../../Driver/
./builddriver 9054

# 成功后你将会看到 Driver "Plx9054/Plx9054.ko" built sucessfully
```

```
## 编译 pixie16

cd ~
cd PKUXIADAQ/software/
make clean
make

# 只要没报错, 并且该文件夹内生成 libPixie16App.a libPixie16Sys.a
```

```
# 修改设置参数
cd ~
cd PKUXIADAQ/parset/

# 修改 cfgPixie16.txt 文件。
# 其中 CrateID 后面的数值表示机箱编号, 该值允许 0-15。如果单机箱则随意设置 (一般就采用默认的 0), 如果多个机箱同步运行务必让每个机箱的该编号设置为不同的数值。
#SettingPars 后面为参数设置文件, 写入要采用的参数配置文件即可。
#ModuleSlot 后面第一个数值表示插件个数, 如果有 3 个插件则为 3。之后的数字未为每个插件在机箱的插槽位置 (插槽位置从 2 开始计数), 有三个插件则之后分别为 2 3 4。
#AutoRunModeTimes 后面数值为自动运行模式下自动切换的时间

# 参数 ModuleSampingRate 与 ModuleBits 只对离线模式生效, 当主界面采用 Offline 模式初始化时则读取该参数。
```

(下页继续)

(续上页)

```
# 修改 Run.config 文件, 该文件中第一行为原始数据存放路径, 第二行为文件名。
# 修改 RunNumber 文件, 该文件中的数值为运行的 run number。
```

```
## 编译图形化获取软件
```

```
cd ~
cd PKUXIADAQ/GUI/
make clean
make
```

```
## 编译非图形化获取软件
```

```
cd ~
cd PKUXIADAQ/NOGUI/
make clean
make
```

```
## 编译在线监视程序
```

```
cd ~
cd PKUXIADAQ/OnlineStatics/

make clean
make
```

```
## 编译数据转换程序
```

```
cd ~
cd PKUXIADAQ/Decode/

# 修改 UserDefine.hh, 按照程序中的说明修改即可

make clean
make
```

```
## 编译事件重构程序
```

```
cd ~
cd PKUXIADAQ/MakeEvent/

# 修改 UserDefine.hh, 按照程序中的说明修改即可

make clean
make
```

2.2 程序使用说明

- 开机机箱后重启电脑(电脑必须晚于机箱开启)
- 开启机箱后 ROOT 权限下加载 Plx9054 驱动
- 正常获取

```
## ROOT 权限下加载 Plx9054 驱动

cd ~
cd PKUXIADAQ/PlxSdk/Bin/
su # 输入 ROOT 密码
./Plx_load 9054
# 将会看到加载成功的提示
exit # 退出 ROOT 权限
```

```
## 启动图形界面程序

cd ~
cd ~/PKUXIADAQ/GUI
./pku

# 将会弹出图形化界面
# 可选择 Online/Offline Mode 然后按 Boot 初始化
# 等待初始化成功后，可修改输出数据文件路径，文件名，run number。按 Complete 按钮确认。
# 此时 LSRunStart 按钮变为可操作。即可开始按 Start，之后第二次按即为 Stop。
#Online Statistics 选项选择表示发送在线统计
#Update Energy Monitor 每选择一次则从插件内部读取一次能谱信息并发送给在线程序（频繁选择会影响获取）
```

```
## 启动非图形界面程序
```

```
cd ~
cd ~/PKUXIADAQ/NOGUI
./pku
```

根据提示输入控制命令

```
## 启动在线监视程序
```

```
cd ~
cd PKUXIADAQ/OnlineStattics/
./online
```

```
# 将会弹出图形化界面
# 按 RunStart 开始启动监视，每 3 秒更新一次每路的输入率、输出率。（开启机箱后第一次启用该程序需要在获取开启之后）
# 监视界面右下角有对写入硬盘使用量的监视。
```

#EnergyMonitor 页面用来查看能谱。由于插件内部寄存器大小限制，该能谱与实际能谱地址范围存在差别。

```
## 执行数据转换程序
```

```
cd ~
cd PKUXIADAQ/Decode/

# 在上一轮获取结束之后，我们便可将上一轮数据转为 ROOT 文件
./decode xxx
# xxx 为运行 run number
```

CHAPTER 3

用户指南

User's Manual Digital Gamma Finder (DGF) PIXIE-16 Version 1.40, October 2009

Pixie-16 User Manual Version 3.00 August 21, 2018

重要: Pixie-16 专为单个指数衰减信号而设计。

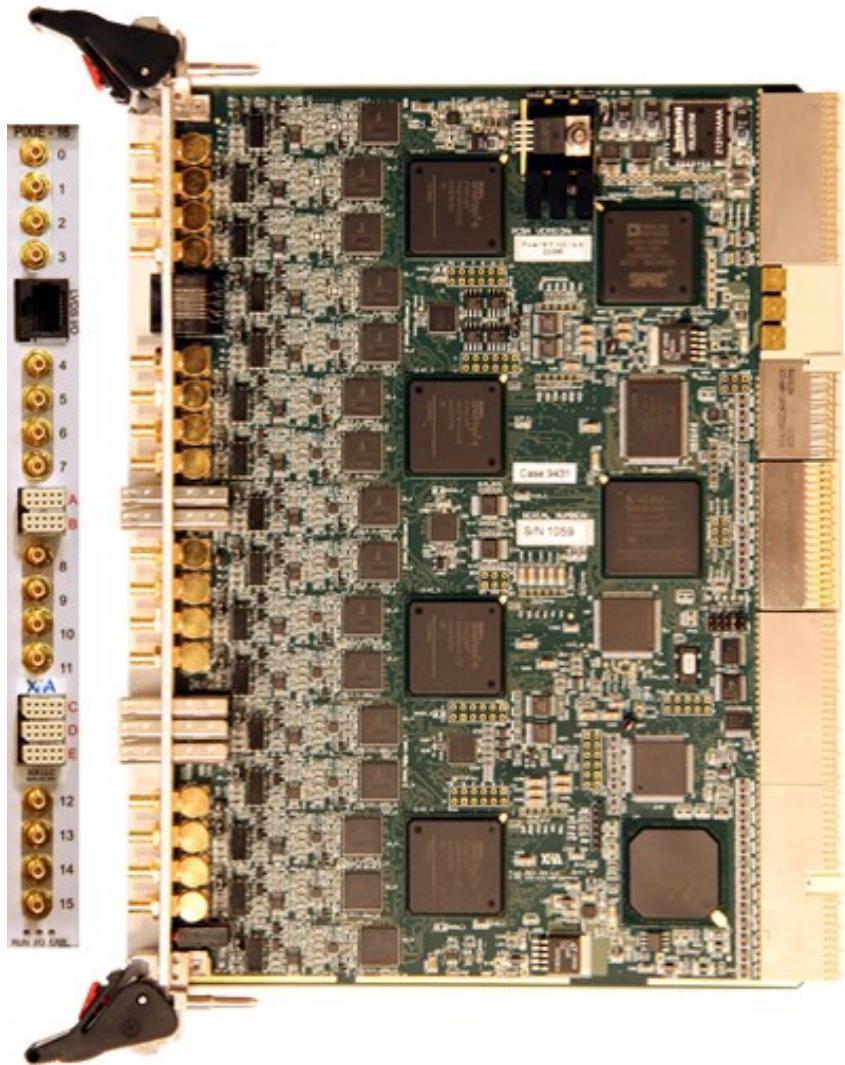
可以通过特定的参数设置来匹配步进脉冲 (step pulses) 或短的非指数脉冲 (short non-exponential pulses)。

来自复位前置放大器 (reset preamplifiers) 的楼梯类型 (staircase type) 信号通常需要先进行交流耦合 (AC coupled) 处理。

危险: 如果安装了 50 欧姆输入端接跳线并且未使用 1: 4 衰减，则建议探测器输出信号的幅度不要超过 +/-3.5V。

不要热插拔！

为避免人身伤害和/或损坏 DGF-Pixie-16，在从机箱中取出 DGF-Pixie-16 之前，请务必关闭机箱电源！



DGF Pixie-16 是一款基于 CompactPCI/PXI 标准的 16 通道全数字波形采集和谱仪卡，可快速读取主机数据。它将波形数字化的谱学和在线脉冲形状分析功能相结合。Pixie-16 几乎可以接收来自任何辐射探测器的信号。输入信号将被 12/14/16 位 100/250/500 MSPS ADC 数字化。每个通最高可以将长度 163.8 μ s 的波形存储在 FIFO 中。

波形可用于板载脉冲形状分析，也可通过向核心处理软件中定制添加用户功能。主机系统可以读出波形，时间戳和脉冲形状分析的结果，以进行进一步的离线处理。脉冲高度计算为 16 位精度，在线最多可以记录分组为 32K bin 的能谱。Pixie-16 支持符合测量，可以实现复杂的触发模式。

通过 CompactPCI/PXI 背板到主机的数据读出速率最高可达 109 Mbyte/s。标准 PXI 背板以及其它定制背板连接用于在几个 Pixie-16 模块之间分配时钟和触发信号，以进行组操作。通过将 Pixie-16 模块与市售的 CompactPCI/PXI 处理器，控制器或 I/O 模块组合在同一机箱中，可以构建完整的数据采集和处理系统。

Pixie-16 是一种用于伽马射线或其它辐射探测器阵列的波形采集和 MCA 直方图的仪器：

- **100 MSPS**

- 高纯锗探测器/分块的高纯锗探测器
- 闪烁体/PMT 组合: NaI, CsI, BGO 和许多其它组合
- 气体探测器
- 大面积硅探测器/硅条探测器

- **250 MSPS**

- 闪烁体
- 溴化镧

- **500 MSPS**

- 闪烁体
- 溴化镧

Pixie-16 模块必须在定制的 6U CompactPCI/PXI 机箱中运行，在 CompactPCI/PXI 标准 1 中未包含在特定电压下提供高电流。目前 XIA 提供 14 插槽机箱。将主机（或远程 PXI 控制器）放入机箱的系统插槽（插槽 1）中。在机箱断电下将 Pixie-16 模块放入任何可用的插槽（插槽 2-14）。安装模块后，打开机箱电源（Pixie-16 模块不支持热插拔）。如果使用远程控制，请务必在打开机箱电源后再启动电脑主机。

3.1 数据结构

Event header as the first 4 words
RevD(12-bit,100MHz),RevF(14-bit,100MHz)

Index	Data					
	[31]	[30:17]	[16:12]	[11:8]	[7:4]	[3:0]
0	Finish Code	Event Length	Header Length	CrateID	SlotID	Chan#
1	[31:0]					
	EVTTIME_LO[31:0]					
2	[31]	[30:16]			[15:0]	
	CFD forced trigger bit	CFD Fractional Time[14:0] x 32768			EVTTIME_HI[15:0]	
3	[31]	[30:16]			[15:0]	
	Trace Out-of-Range Flag	Trace Length			Event Energy	

(EVTTIME_LO[31:0]+EVTTIME_HI[15:0]x2³²+(CFD_Fractional_Time[14:0]/32768))x10ns

Finish Code: 0-good event,1- pileup event

CFD forced trigger bit: 0- valid ,1-invalid (Threshold was set too high)

Trace Out-of-Range Flag: 0- trace in range, 1- trace is out of range



Event header as the first 4 words RevF(12/14/16-bit,250MHz)

Index	Data					
0	[31]	[30:17]	[16:12]	[11:8]	[7:4]	[3:0]
	Finish Code	Event Length	Header Length	CratID	SlotID	Chan#
1	[31:0] EVTTIME_LO[31:0]					
2	[31]	[30]	[29:16]	[15:0]		
	CFD forced trigger bit	CFD trigger source bit	CFDFractionalTime[13:0]x16384	EVTTIME_HI[15:0]		
3	[31]	[30:16]	[15:0]			
	Trace Out-of-Range Flag	Trace Length	Event Energy			

((EVTTIME_LO[31:0]+EVTTIME_HI[15:0]x2³²)x2 - CFD trigger source bit + (CFD_Fractional_Time[13:0]/16384))x4ns

Finish Code: 0-good event,1- pileup event

CFD forced trigger bit: 0- valid ,1-invalid (Threshold was set too high)

CFD trigger source bit:

Trace Out-of-Range Flag: 0- trace in range, 1- trace is out of range



Event header as the first 4 words RevF(12/14-bit,500MHz)

Index	Data					
0	[31]	[30:17]	[16:12]	[11:8]	[7:4]	[3:0]
	Finish Code	Event Length	Header Length	CratID	SlotID	Chan#
1	[31:0] EVTTIME_LO[31:0]					
2	[31:29]	[28:16]	[15:0]			
	CFD trigger source bits[2:0]	CFD Fractional Time[12:0] x 8192	EVTTIME_HI[15:0]			
3	[31]	[30:16]	[15:0]			
	Trace Out-of-Range Flag	Trace Length	Event Energy			

(EVTTIME_LO[31:0]+EVTTIME_HI[15:0]x2³²)x10 + ((CFD_Fractional_Time[12:0]/8192)+ CFD trigger source bits[2:0]-1)x2ns

Finish Code: 0-good event,1- pileup event

CFD trigger source bits:

Trace Out-of-Range Flag: 0- trace in range, 1- trace is out of range



如果开启了波形记录，则波形数据将紧跟在事件头的最后。由于原始 ADC 数据点是 12/14/16 位数，因此将两个 12/14/16 位数字打包成一个 32 位字。由于事件头具有可变长度 (4,6,8,10,12,14,16 或 18 个字)，具体取决于各种输出数据选项的选择，事件长度，事件长度和波形长度记录在事件头的前 4 个字 (words) 来引导输出数据流。

- CAEN 插件在采集波形时，一个插件所有通道每个事件波形采集长度只能设置成相同的。
- XIA 插件可以每个通道单独设置数据长度。例如：ch-0 设置采 3000 个点，ch-1 设置采 2000 个点，ch-2 设置采 5000 个点，ch-3 设置成不采集波形……。另外，还可每个通道选择是否记录 baseline sum、QDC、external timestamps。
- XIA 还可设置对 pile up 事件的处理。例如 pile up 事件记录波形，其它事件则不记录波形。意味着一个 channel 不同事件其数据长度也不一定相等。
- 在读取数据时候，CAEN 插件每次拿到的数据都是完整事件的数据。而 XIA 插件 FIFO 的读指针、写指针是独立的、同时进行的，所以每次我们要取数据时候都需要先查询下当前 FIFO 有多少数据，

然后再指定该次读取多少数据。因为 FIFO 读、写是独立的，因而查询到的数据量是该时刻拥有的数据量，这个数据量基本都不是完整事件的数据量（还有部分是当前事件正在写入的）。每个事件除了有 4 words 的 header 及后面紧跟着的数据外并没有事件标记符，当然这样设计最大的好处就是速度快。

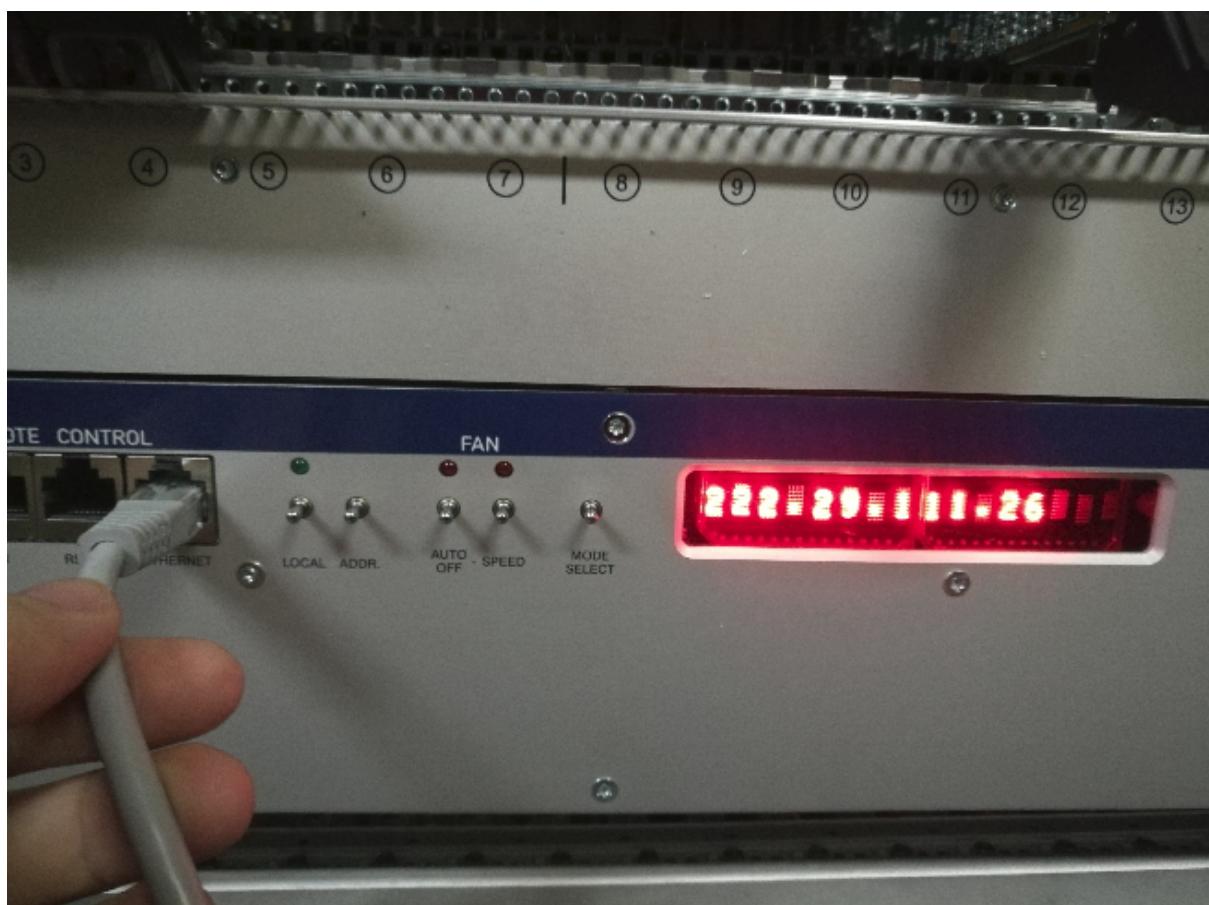
基于以上事实，两个公司的产品对用户写时时监视获取率、在线监视程序等的实现方式存在较大差异。

- CAEN：由于 CAEN 一个插件上每个通道的采样数相同，因而他们每个事件的长度是一致的。在拿到数据时即可存文件或者 decode 出每个事件的信息用来统计时时获取速率，也可将数据发送到在线程序。
- XIA：由于其上每个通道数据长度可单独设置，这是 XIA 插件的灵活性。又 FIFO 查询到的数据量是当前时刻的数据量，这样每次取到的数据都不是完整事件的数据。
- 对于多个插件的系统，获取时候一直循环读取每一个插件上的数据，如果将所有插件的数据存在同一个文件内，将造成数据的错乱。如果每个插件的数据单独存在一个文件内，那么下一次读取的该插件的数据中开头的不完整事件的数据刚好接上一次最后一个不完整事件的数据，可解决这个问题。
- 由于每次拿到的不是完整数据，因此也不好 decode 当前数据来获得每个事件的信息（从开始就 decode 记录每次数据的不完整事件的情况也可解决该问题，但是这样 decode 需要额外的时间）。但是 XIA 内部 DSP 上统计着信号输入量、获取接收量、活时等信息。可频繁调用函数读取 DSP 上的统计信息来监视时时获取速率。但是频繁调用该函数会对获取的 I/O 有一点影响。几秒钟调用一次该函数，这个影响基本可以忽略。
- 由于每次拿到的不是完整数据，因此共享内存的在线监视也不容易实现，如若要实现，也不是没办法（从开始就 decode 记录每次数据的不完整事件的情况也可解决该问题，但是这样 decode 需要额外的时间）。
- 下面给出一些困难的解决思路：牺牲 XIA 插件的灵活性，对一个插件设置成每个通道采集波形长度相同（这样就与 CAEN 一样了），因为每个通道的每个事件数据的长度相等，读取数据时候查询到当前的数据量之后，然后只读取事件长度整数倍的最大数据，这样每次拿到的都是完整的事件数据了，可参照 CAEN 的实现方式。
- 对采波形时候，可采用“伪”在线监视获取情况，即获取文件每隔一段时间保存一次，用上一轮的数据作为“在线”数据。

CHAPTER 4

机箱

4.1 远程控制



如上图中将机箱网口接入网线，右侧显示屏中会快速闪过所分配的IP，例如这里IP为222.29.111.26。如果没看清该IP，则拔出网线重新连接。

UEP6000/PL500 - Mozilla Firefox

UEP6000/PL500 | 222.29.111.26 | 搜索 | 星 | 自 | 下 | 家 | 三

[UEP6000/PL500](#) W-IE-NE-R

MAIN POWER VME SYSRESET FAN SLOWER FAN FASTER

Global Status

Power Supply Status	OFF
Fan Tray Status	OK
Fan Speed	0 RPM
Fan Speed (mean)	0 RPM
Fan Temperature	69°F

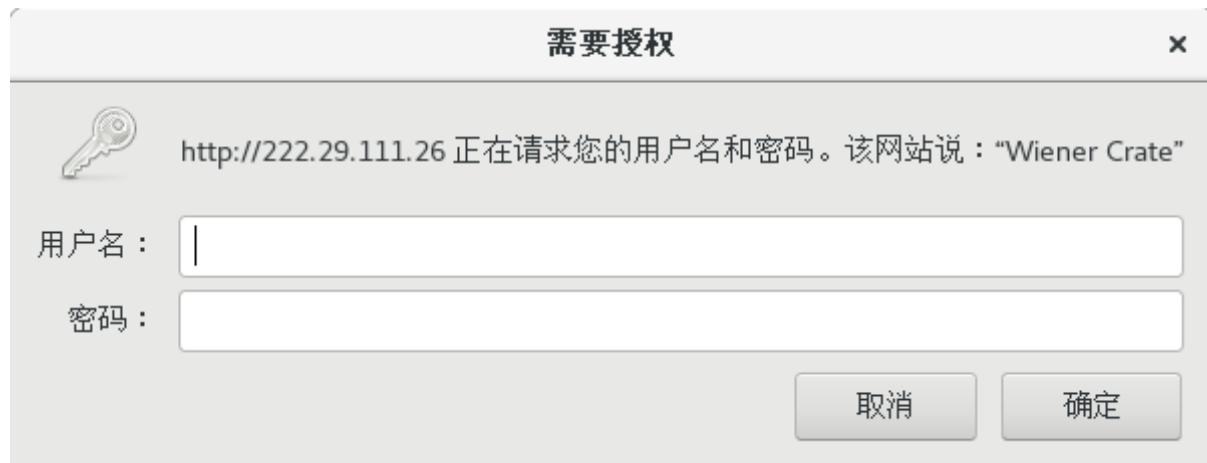
Output Voltages

Channel	Name	Voltage	Current	Status
U0	+5V5	0.00V	0A	OK
U1	+12V	0.0V	0.0A	OK
U2	+5V0	0.00V	0.0A	OK
U3	+3V3	0.00V	0A	OK
U5	-12V	0.0V	0.0A	OK
U6	-6V0	0.00V	0.0A	OK
U7	+1V8	0.00V	0A	OK

External Temperature Sensors and Voltage Inputs

1	2	3	4	5	6	7	8
66°F		66°F		66°F			

如上图，在浏览器中输入 IP，则可进入该控制页面。图中展示的是机箱关闭状态。



按钮 **MAIN POWER** 用来控制机箱开启及关闭，首次点击会弹出以上登陆框。

输入用户名“private”，默认密码“private”

The screenshot shows a web-based monitoring interface for the UEP6000/PL500 system. At the top, there's a header bar with the title "UEP6000/PL500 - Mozilla Firefox". Below it is a navigation bar with a back button, forward button, address bar containing "222.29.111.26", a search bar with placeholder "搜索", and various browser control icons.

Below the navigation bar, there are four buttons: "MAIN POWER", "VME SYSRESET", "FAN SLOWER", and "FAN FASTER". To the right of these buttons, the text "W-IE-NE-R" is displayed.

The main content area is divided into sections:

- Global Status:**

Power Supply Status	ON
Fan Tray Status	OK
Fan Speed	3300 RPM
Fan Speed (mean)	3290 RPM
Fan Temperature	64°F
- Output Voltages:**

Channel	Name	Voltage	Current	Status
U0	+5V5	5.49V	6A	OK
U1	+12V	12.0V	0.2A	OK
U2	+5V0	5.00V	0.2A	OK
U3	+3V3	3.31V	5A	OK
U5	-12V	12.0V	0.0A	OK
U6	-6V0	6.00V	4.5A	OK
U7	+1V8	1.81V	1A	OK
- External Temperature Sensors and Voltage Inputs:**

1	2	3	4	5	6	7	8
64°F		66°F		64°F			

上图为机箱开启后的监视的参数。

其中，右上角的按钮 **FAN SLOWER** 和 **FAN FASTER** 用来调节风散的转速。

4.2 插槽

机箱共有 14 个插槽，底下分别标有数字 1-14。其中插槽 1 为控制器插槽，2-14 为采集卡插槽。

CHAPTER 5

固件

北京大学定制固件

在标准固件的基础上添加了以下功能：

- **100MHz 12 bit(pixie16_rev12b100m_firmware_release)**
 - 标准固件
- **100MHz 14 bit(pixie16_revfpku_14b100m_firmware_release_09272018)**
 - MultiplicityMaskHigh[31]=0 和 1 时候 RJ45 口与前面板 A 均能输出 multiplicity 结果。
 - 当计算的能量为负数时，该值设置为 0。
 - pileup 事件能量保留，不设置为 0。
 - 在记录波形模式下，waveform 的 buffer 满了的时候，插件不 busy，header 继续记录，该情况下，输出的数据没有波形。
 - 在采集波形时候，增加了降频输出的功能，采取的策略为可选择 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 频率的输出，即多少个点保留一个点。保留的点是平均后的值。
 - 删除一些非必要的等待来加速事件处理。
 - 删除 DSP 中非必要的处理：包括堆积检测、超量程波形检测等
- **100MHz 14 bit(pixie16_revfpku_14b100m_dsp_update_05082018)**
 - MultiplicityMaskHigh[31]=0 和 1 时候前面板均能输出 multiplicity 结果。
 - 当计算的能量为负数时，该值设置为 0。
 - pileup 事件能量保留，不设置为 0。
 - 在记录波形模式下，waveform 的 buffer 满了的时候，插件不 busy，header 继续记录，该情况下，输出的数据没有波形。
 - 在采集波形时候，增加了降频输出的功能，采取的策略为可选择 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 频率的输出，即多少个点保留一个点。保留的点是平均后的值。
- **250MHz 12bit(pixie16_revf_12b250m_firmware_release)**
 - 标准固件
- **250MHz 14bit(pixie16_revfpku_14b250m_release_06092019)**
 - MultiplicityMaskHigh[31]=0 和 1 时候 RJ45 口与前面板均能输出 multiplicity 结果。

- 当计算的能量为负数时，该值设置为 0。
- pileup 事件能量保留，不设置为 0。
- **250MHz 16bit(pixie16_revfpku_16b250m_release_06092019)**
 - MultiplicityMaskHigh[31]=0 和 1 时候 RJ45 口与前面板均能输出 multiplicity 结果。
 - 当计算的能量为负数时，该值设置为 0。
 - pileup 事件能量保留，不设置为 0。
- **500MHz 14bit(pixie16_revf_14b500m_firmware_release)**
 - 标准固件

CHAPTER 6

数据解码

Decode 程序用来将同一轮数据不同采集卡采集的数据转为一个 ROOT 文件。用户的物理分析以本程序产生的 ROOT 文件为基准。

用户首先需要修改 **UesrDefine.hh** 文件中的定义

```
#define RAWFILEPATH "/home/wuhongyi/data/"      //原始二进制文件的路径
#define RAWFILENAME "data"                         //原始文件的文件名
#define ROOTFILEPATH "/home/wuhongyi/data/"        //要生成 ROOT 文件的路径

#define TimesHist 3600   // second 直方图参数，设置比该轮运行时间长即可

#define Crate0
#define Crate0num 5    //该机箱中使用插件数
const int Crate0SamplingRate[Crat0num] = {100,100,100,250,250}; //分别指定每个插件的采样率 100/250/500 三种采样率 0 为跳过该插件
```

用户需要修改：

- 原始二进制文件存放目录
- 生成 ROOT 文件存放目录
- 文件名
- 机箱中使用采集卡个数
- 每个采集卡对应的采样频率，如果采样频率设置为 0，则忽略该采集卡的数据

修改之后执行以下命令编译程序：

```
make clean
make
```

编译成功之后将生成一个可执行文件 **decode**，程序运行方式：

```
./decode [RuNnumber]
```

其中 *[RuNnumber]* 为想要转换的文件运行编号。

例如：

```
./decade 3
```

ROOT File Branch:

- sr(short): sample rate , 100/250/500 , This value is specified in UesrDefine.hh. / 该数值由 UesrDefine.hh 中指定
- pileup(bool): 堆积标记。
- outofr(bool): 是否超量程标记。
- cid(short): 机箱编号
- sid(short): 采集卡所在插槽编号
- ch(short): 采集卡通道
- evte(unsigned short): 梯形算法的能量
- ts(long int 64 bit): 时间戳
- ets(long int 64 bit): 外部时间戳
- cfd(short): cfd 数值
- cfdft(bool): cfd 数值是否有效
- cfds(short): cfd source , 仅适用于 250/500 MHz 采集卡
- trae(unsigned int): 能量梯形上升段积分
- leae(unsigned int): 能量梯形下降段积分
- gape(unsigned int): 能量梯形平台段积分
- base(double): 能量梯形算法的基线
- qs(unsigned int): 八个 QDC 面积的积分
- ltra(unsigned short): 波形采集点数
- data(unsigned short): 波形数据
- dt(unsigned short): 为了方便直接查看每个波形, 添加了一个数值从 0 - N-1 的数组
- nevt(unsigned int): 该事件在本文件中的编号

下图展示一个文件中的 Branch 定义:

```
[wuhongyi@ScientificLinux data]$ root data_R0595.root
root [0]
Attaching file data_R0595.root as _file0...
(TFile *) 0x1f6edd0
root [1] .ls
TFile**      data_R0595.root
TFile*       data_R0595.root
  KEY: TTree   tree;175          PKU XIA Pixie-16 Data
  KEY: TTree   tree;174          PKU XIA Pixie-16 Data
root [2] tree->Print()
*****
*Tree    :tree     : PKU XIA Pixie-16 Data
*Entries : 1123666 : Total = 13993888785 bytes File Size = 3708728891 *
*      :           : Tree compression factor = 3.77
*****
*Br   0 :sr      : sr/S
*Entries : 1123666 : Total Size= 2263185 bytes File Size = 167039 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 13.53
*.....
*Br   1 :pileup   : pileup/0
*Entries : 1123666 : Total Size= 1140233 bytes File Size = 30956 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 36.71
*.....
*Br   2 :outofr   : outofr/0
*Entries : 1123666 : Total Size= 1140233 bytes File Size = 23284 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 48.81
*.....
*Br   3 :cid      : cid/S
*Entries : 1123666 : Total Size= 2263364 bytes File Size = 27637 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 81.76
*.....
*Br   4 :sid      : sid/S
*Entries : 1123666 : Total Size= 2263364 bytes File Size = 175529 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 12.87
*.....
*Br   5 :ch       : ch/S
*Entries : 1123666 : Total Size= 2263185 bytes File Size = 284004 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 7.96
*.....
*Br   6 :evte     : evte/s
*Entries : 1123666 : Total Size= 2263543 bytes File Size = 1631103 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 1.39
*.....
*Br   7 :ts       : ts/L
*Entries : 1123666 : Total Size= 9020679 bytes File Size = 3066162 *
*Baskets : 349 : Basket Size= 51200 bytes Compression= 2.94
*.....
*Br   8 :ets      : ets/L
*Entries : 1123666 : Total Size= 9021032 bytes File Size = 73195 *
*Baskets : 349 : Basket Size= 51200 bytes Compression= 123.15
*.....*
```

```

*Br   9 :cfds      : cfd/S
*Entries : 1123666 : Total Size=    2263364 bytes File Size =    2191516 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   1.03   *
*.....
*Br  10 :cfdft     : cfdft/0
*Entries : 1123666 : Total Size=    1140054 bytes File Size =    30291 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=  37.51   *
*.....
*Br  11 :cfds      : cfd/S
*Entries : 1123666 : Total Size=    2263543 bytes File Size =    269187 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   8.39   *
*.....
*Br  12 :trae      : trae/i
*Entries : 1123666 : Total Size=    4510879 bytes File Size =   2642761 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   1.71   *
*.....
*Br  13 :leae      : leae/i
*Entries : 1123666 : Total Size=    4510879 bytes File Size =   2886877 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   1.56   *
*.....
*Br  14 :gape      : gape/i
*Entries : 1123666 : Total Size=    4510879 bytes File Size =   2982289 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   1.51   *
*.....
*Br  15 :base      : base/D
*Entries : 1123666 : Total Size=    9021385 bytes File Size =   3984210 *
*Baskets :    349 : Basket Size=     51200 bytes Compression=   2.26   *
*.....
*Br  16 :qs        : qs[8]/i
*Entries : 1123666 : Total Size=  35989106 bytes File Size =  23047394 *
*Baskets :    354 : Basket Size=    174592 bytes Compression=   1.56   *
*.....
*Br  17 :ltra      : ltra/s
*Entries : 1123666 : Total Size=    2263543 bytes File Size =   230891 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   9.79   *
*.....
*Br  18 :data      : data[ltra]/s
*Entries : 1123666 : Total Size= 6945634237 bytes File Size = 3206301603 *
*Baskets :   1689 : Basket Size= 25600000 bytes Compression=   2.17   *
*.....
*Br  19 :dt        : dt[ltra]/s
*Entries : 1123666 : Total Size= 6945630851 bytes File Size = 457034676 *
*Baskets :   1689 : Basket Size= 25600000 bytes Compression= 15.20   *
*.....
*Br  20 :nevt      : nevt/I
*Entries : 1123666 : Total Size=    4510879 bytes File Size =   1581484 *
*Baskets :    175 : Basket Size=     51200 bytes Compression=   2.85   *
*.....

```

每轮数据转换结束，本文件夹内均会生成一个 **txt** 文件，该文件统计了采集卡每个通道的以下信息：

- Mod: 采集卡标记，从 0 开始
- Channel: 采集卡通道标记，0 - 15
- OutOfRange: 信号幅度超出模数转换模块范围的事件数
- Pileup: 标记为堆积的事件数
- CfdForcedTrigger: cfd 强制触发事件数 (cfд 未过阈值)
- Energy->0: 计算梯形能量小于 0 的事件数 (计算结果小于 0 的直接被标记为 0 了)

- WaveformCount: 记录波形的事件数
- TotalEvent: 总的输出事件数

每轮数据转换结束，本文件夹内均会生成一个 **ROOT** 文件，该文件统计了采集卡每个通道的计数率.

CHAPTER 7

图形交互界面

配置好 **parset** 内参数文件

进入 GUI 目录，执行以下命令即可弹出主控制界面

```
./pku
```

7.1 主控制界面



主界面最上方是 File、Base、Expert、Monitor、Offline 五个下拉栏。里面的子菜单如下所示：

- File

- Exit
- About
- **Base**
 - Base Setup
 - Trigger Filter
 - Energy
 - CFD
 - QDC
 - Decimation
 - Copy Pars
 - Save2File
- **Expert**
 - Module Variables
 - CSRA
 - Logic Set
- **Debug**
 - Hist & XDT
 - Trace & Baseline
- **Offline**
 - Adjust Par
 - Simulation(暂未实现)

开启获取主界面之后，选择 **Online Mode** 选项表示在线模式，需要连接机箱，该模式下可使用所有的功能（包括离线分析功能）。不选择 **Online Mode** 选项则表示开启离线模式，可设置、修改获取参数，分析已采集波形。

选择、或者不选择 **Online Mode** 选项之后，按 **Boot** 按钮开启初始化过程，可看到最下方 *Information* 栏目中的状态变化。

系统初始化成功后，再次确认 *Setup* 栏中的获取文件存放路径、文件名、文件编号是否有问题，如果有问题则直接修改，确认之后按 **Complete**。

确认 *Setup* 栏中的信息之后，*Control* 栏中的主按钮 **RunStart** 则开启，此时点击该按钮，获取则开启，按钮状态更改为 **RunStop**，再次点击该按钮，获取结束，运行的 *Run Num* 号码自动加一。再次点击 **RunStart** 开启下轮获取。

当前，获取之前可通过最上方的下拉栏里面的子菜单来调节、修改参数。获取数据时请勿操作 *Control* 栏之外的所有选项。

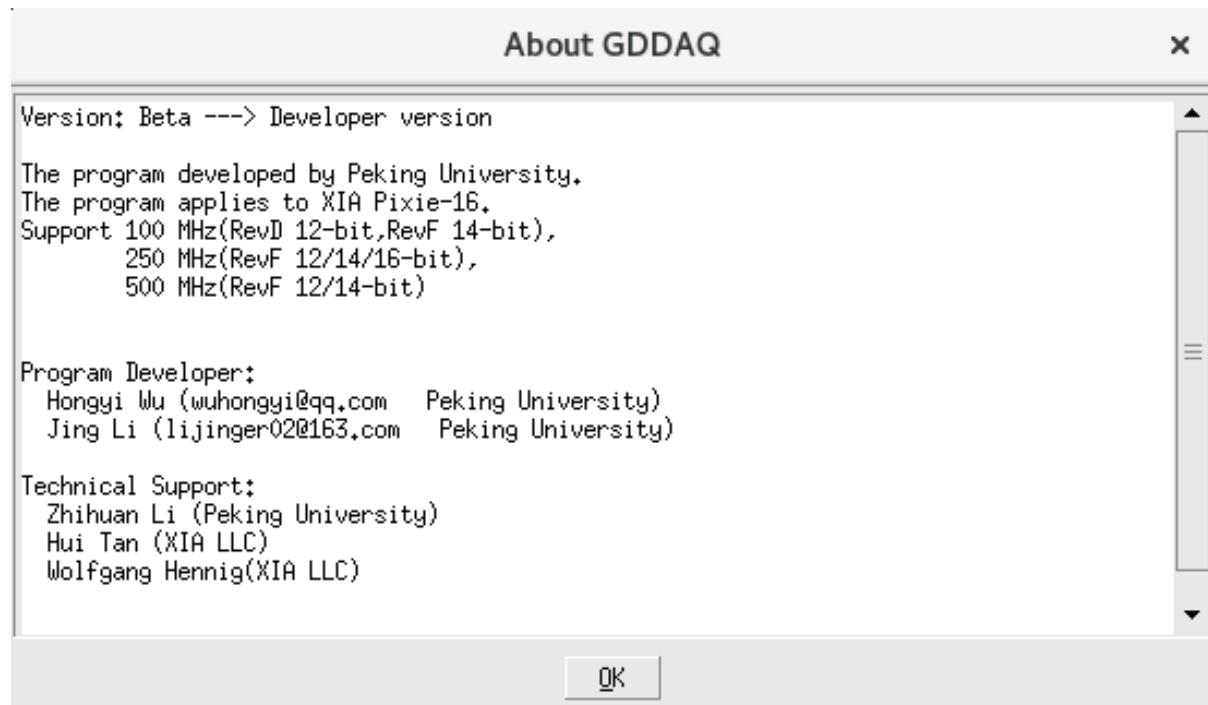
Control 栏内的 **Online Statistics** 选项开启则获取每 **3 s** 向 *OnlineStatistics* 程序发送时时每路信号的输入率、输出率信息。

点击 **Update Energy Monitor** 选项一次，则将所有采集卡内部寄存器中每路的一维能谱发送到 **Online Statistics** 程序，发送该信息会导致一定的死时间，请不要频繁点击该选项。

7.2 File

本下拉栏内容没有实际用途。

7.2.1 About



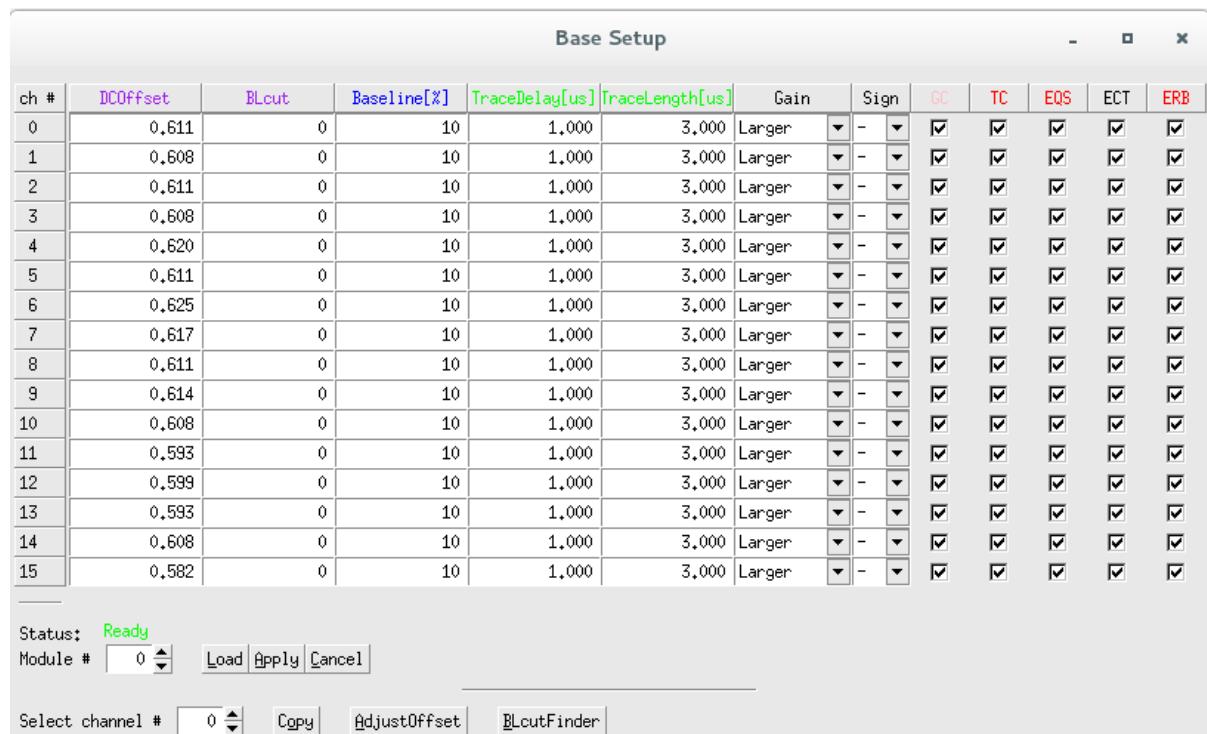
软件开发者介绍。之后将会添加主程序基本操作说明。

7.3 Base

本下拉栏中调节内容为基础内容，任何使用 Pixie16 获取系统的人员都应该熟悉并掌握其调节技巧。

7.3.1 Base Setup

控制界面



界面下方的 Status 显示为 绿色的 Ready 时表示可操作该界面，否则需要等待。

界面中 Module 后面的参数用来选择调节的采集卡模块，Load 表示读取该采集卡的参数数值，Apply 表示将界面中的数值写入采集卡。

界面下方的 Select Channel 后面的参数表示选择用来将界面上该通道的参数复制给其它通道，点击后面 Copy 完成复制，然后需要 Apply 来将参数写入采集卡。

Base Setup 页面控制每个通道的模拟增益，偏置和极性。单击顶部控制栏中 **Monitor** 的 **Trace & Baseline** 可以查看从 ADC 读取的信号，同时调整这些参数。该页面可以显示模块的一个或全部 16 个通道。您可以在 **Hist & XDT** 页面中设置每个模块每个通道的采样间隔以捕获更长的时间帧。单击 **Draw** 以更新图形。

来自探测器的脉冲应落在 0 到 16383 (例如 14 位) 的范围内，基线在 ~1638 处允许漂移和/或下冲，并且在上限处没有削波。如果存在削波，请调整增益和偏置，或单击 **AdjustOffset** 按钮让软件将 DC 偏置设置为适当的水平。

由于 FPGA 中的触发/滤波电路仅作用于正极性脉冲，因此负脉冲在输入 FPGA 处进行极性反转，并且 ADC 波形显示中显示的波形包括此可选的反转。因此，设置通道的极性，使得来自探测器的脉冲以正幅度（上升沿）出现。

在 **Base Setup** 选项卡中，您可以设置在列表模式运行中获取的波形的总跟踪长度和预触发跟踪延迟。

跟踪延迟 (trace delay) 不能长于跟踪长度 (trace length)，并且对于每个 Pixie-16 通道，跟踪延迟和跟踪长度的最大值也有限制。

参数介绍

- 选项 *Gain* 为增益调节，用户可选择 Larger 或者 Small 档，具体每个采集卡这两档所对应的增益参数用户可自行测试或者咨询厂家。
- 选项 *Sign* 选择输入信号的极性，输入正信号选择 “+”，输入负信号则选择 “-”。
- 选项 *GC* 表示是否记录该通道数据，选择表示记录该通道数据，不选择表示不记录。
- 选项 *ECT* 选择表示开启 CFD 触发功能。否则，则采用快梯形的前沿甄别。

红色的 *TC*, *EQS*, *ERB* 用来选择输出哪些原始数据:

- 选项 *TC* 选择表示记录波形, 此时 *TraceDelay*、*TraceLength* 生效, 不选择则表示不记录波形。
- 选项 *EQS* 选择表示记录八个 QDC 的积分, 不选择则不记录。
- 选项 *ERB* 选择表示记录能量梯形的三部分面积积分及梯形计算的基线数值。

绿色的 *TraceDelay*、*TraceLength* 为输出数据的点数, 该参数除以采集卡的标称采样率即为波形实际输出数据点数:

- *TraceDelay* 表示触发前的采集波形长度。
- *TraceLength* 表示整个波形采集长度。

需要特别说明的是采用降频模式时, 实际波形长度为 *TraceDelay* \times 2^N / *TraceLength* \times 2^N (*N* 为降频参数)

蓝色的 *Baseline* 用来调节基线位置, 通过电压的补偿将基线调节到用户预期的位置:

- *Baseline* 可调节范围为 0 - 100, 表示波形基线落在满量程的位置百分比。例如垂直精度 14 bit 采集卡, 该参数设置为 10 意味着降基线补偿调节到满量程 16384 道的 10% 左右即 1638 附近。

紫色的 *DCOffset*、*BLcut* 用户不需要修改, 采用自动调节参数即可。本子菜单中修改了 *Baseline*、*Gain*、*Sign* 之后, 需要按最下方的 **AdjustOffset**, 之后再按 **BLcutFinder** 来自动调节这两个参数。

重要信息

注解: 500 MHz 模块中的波形

对于 500 MHz Pixie-16 模块, ADC 以 500 MHz 运行, 但在 FPGA 中以 100 MHz 时钟记录波形, 每 10 ns 间隔捕获 5 个 ADC 样本。此外, 从 FPGA 到板上外部 FIFO 的数据打包是一次传输两组 5 个 ADC 样本。因此, 波形长度应为 20 ns 的倍数, 即 20 ns, 40 ns, …… 例如, 波形长度为 500 ns, 波形延迟为 200 ns。

小技巧: 好通道 (Good channel)

只有标记为好的通道才会记录其事件。

此设置与通道自身触发的能力无关。

可以有一个触发通道, 其数据被丢弃。

未标记为好的通道将不会自动进行偏置调整。

基线测量

当没有检测到脉冲时, Pixie-16 不断地进行基线测量, 并且在脉冲高度重建期间保持从能量滤波器输出中减去基线平均值。与平均值相差超过 *BaselineCut* 值的基线测量将被拒绝, 因为它们可能被低于触发阈值的小脉冲污染。

可以在 **Trace & Baseline** 页面中查看每个通道的一系列基线测量值, 在 **BASELINE** 面板中可以构建基线直方图, 以验证基线切割 (Baseline Cut) 不会拒绝落入基线分布 (理想情况下为高斯测量主峰值的测量)。

通常, 将基线切割 (Baseline Cut) 保持为默认值就足够了。

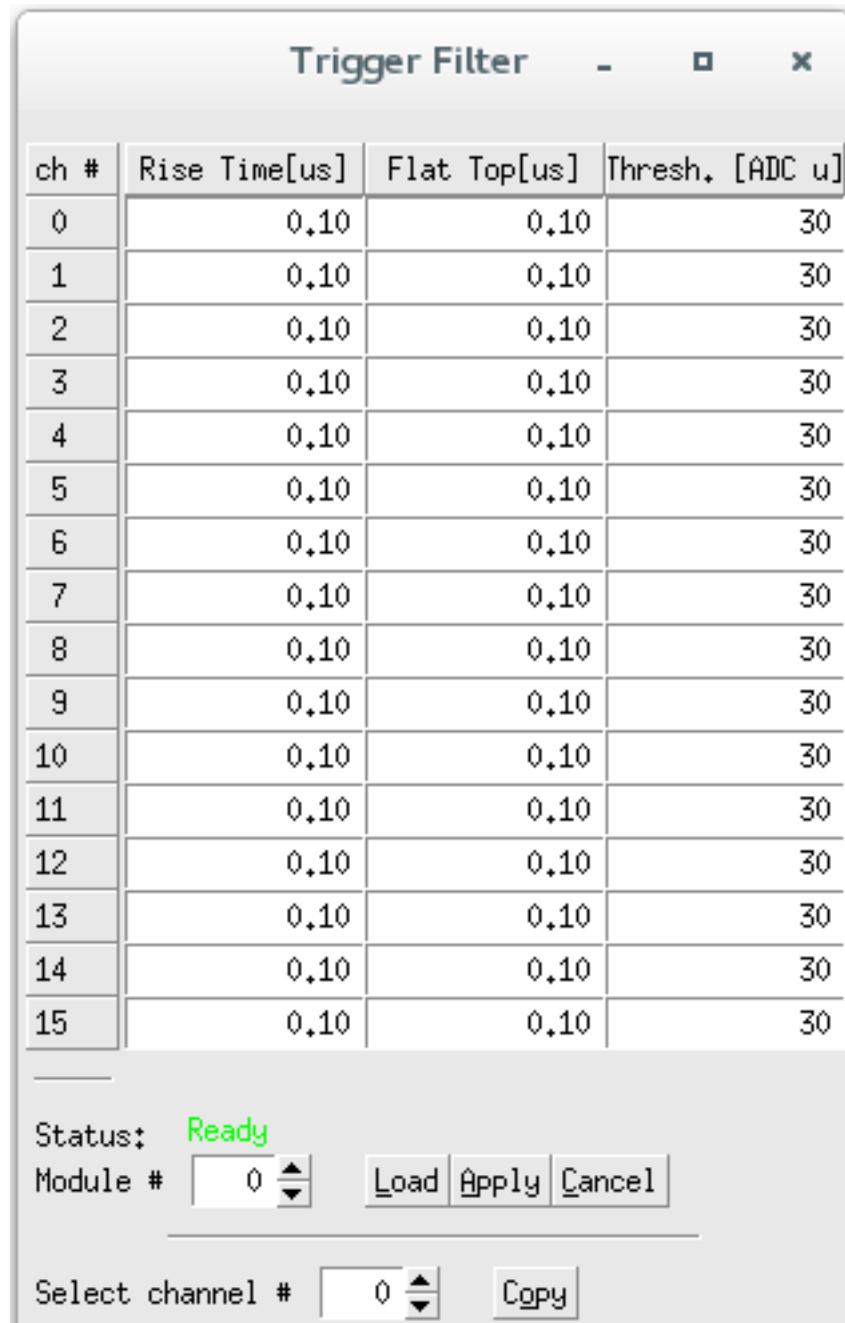
注意: 由于基线计算考虑了指数衰减, 因此如果基线显示中没有明显的脉冲, 说明我们的设置满足了

- a) 正确设置了衰减时间
- b) 探测器脉冲是真正的指数衰减

基线百分比是用于自动偏置调整的参数。通过单击 **AdjustOffses** 按钮，将设置偏置，使 ADC 波形显示中看到的基线降至整个 ADC 范围的基线百分比（例如，对于 12 位 ADC 和基线百分比 = 基线的 10%）落在 ADC 整个量程 4096 个 bin 中的 409 个 bin）。

7.3.2 Trigger Filter

控制界面



界面下方的 Status 显示为 绿色的 Ready 时表示可操作该界面，否则需要等待。底下按钮的操作同上。

- 参数 *Rise Time* 表示触发滤波上升时间。
- 参数 *Flat Top* 表示触发滤波平顶时间。

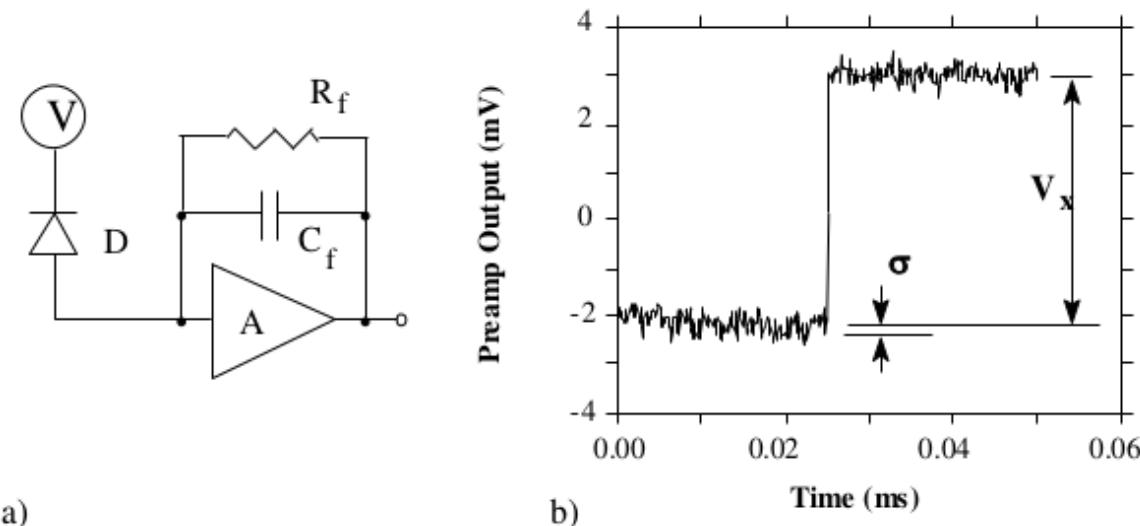
- 参数 *Thresh.* 表示阈值，该数值的设置是相对 fast filter 波形。

以下为参数的一般经验：

- 较长的触发滤波上升时间平均更多采样点，因此允许设置较低的阈值而不触发噪声。
- 通常应将阈值设置得尽可能低，略高于噪声水平。
- 较长的触发滤波平顶时间则可以更容易地触发缓慢的上升脉冲。

数字滤波

粒子能量探测器，包括诸如 Si (Li), HPGe, HgI₂, CdTe 和 CZT 的固态探测器，通常采用电荷敏感前置放大器，如图所示。这里，探测器 **D** 加偏置电压 **V**，并连接到 前置放大器 **A** 的输入端，后者具有反馈电容 **C_f** 和反馈电阻 **R_f**。



通过滤波来降低测量中的噪声。传统的模拟电路使用微分电路和多个积分电路的组合将前置放大器输出步骤（如图 (b) 所示）转换为三角形或半高斯脉冲，其振幅（相对于基线）与 **Vx** 成比例，从而与伽马射线的能量成比例。

数字滤波从一个稍微不同的角度进行。这里的信号已经数字化，不再是连续的。相反，它是一个离散值字符串，如下图所示。图实际上只是上图 (b) 的一个子集，图中的信号由 Tektronix 544 TDS 数字示波器以 10 MSPS (每秒百万采样数) 进行数字化。鉴于此数据集和某种算术处理，确定 **Vx** 的明显方法是对当前处理点之前的点取某种平均值，然后从当前处理点之后点的平均值中减去它。也就是说，如下图中所示，计算标记为“Length”的两个区域的平均值（“Gap”区域被省略，因为这里的信号变化很快），并将其差作为 **Vx** 的度量。因此，**Vx** 值可从以下方程式中得出：

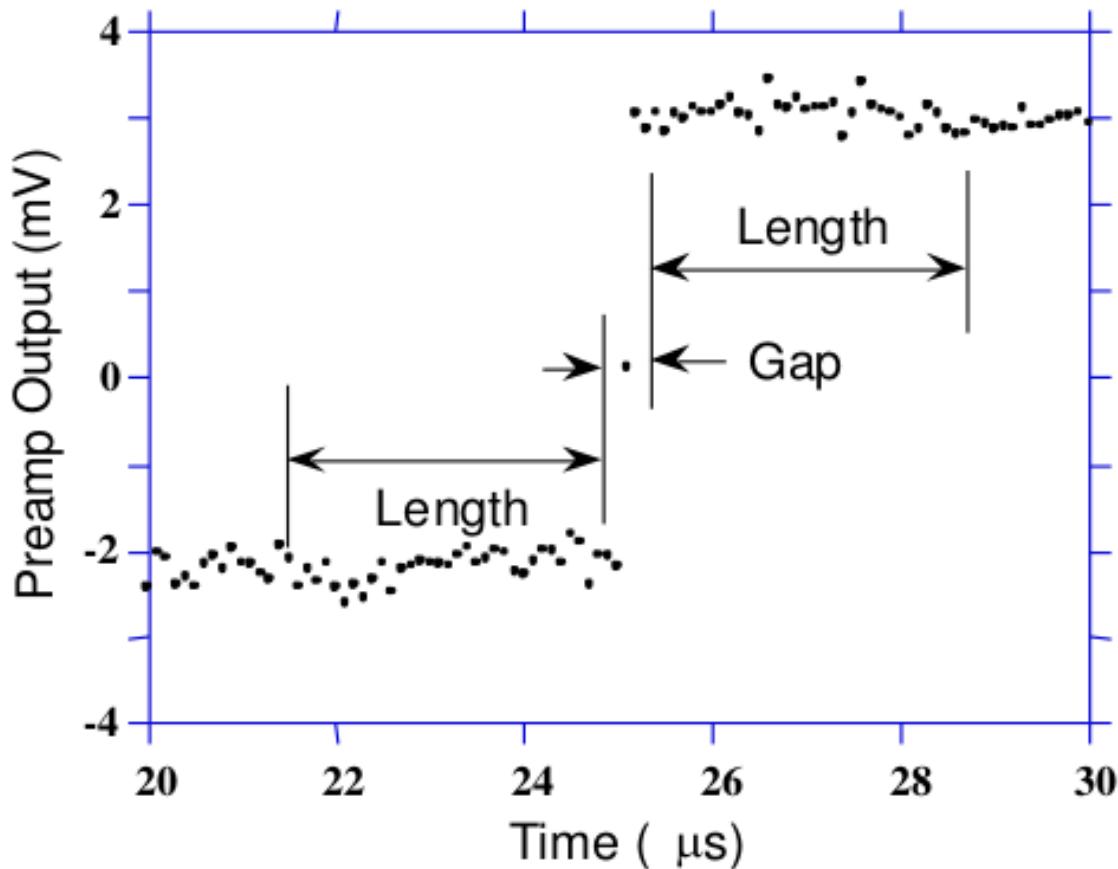
$$V_{x,k} = - \sum_{i(before)} W_i V_i + \sum_{i(after)} W_i V_i$$

其中，加权常数 **Wi** 的值决定计算的平均值类型。两组权重的值之和必须单独归一化。

不同数字信号处理算法之间的主要区别在于两个方面：使用哪组权重 **Wi 以及如何选择区域来计算方程。**

因此，例如，当靠近当前处理点的区域使用较大的权重值，而远离当前处理点的数据使用较小的值时，方程式生成“尖点样”滤波。当权重值为常量时，将获得三角形（如果间隙为零）或梯形滤波。尖点滤波背后的概念是，由于最接近台阶的点携带了关于其高度的最多信息，因此它们在平均过程中应该是最强大的权重。如何选择滤波长度会导致时间变化（长度随脉冲变化）或时间不变（所有脉冲的长度相同）滤波。传统的模拟滤波是不随时间变化的。时变滤波背后的概念是，由于伽马射线随机到达，它们之间的长度也相应变化，因此可以通过将长度设置为脉冲间隔来最大限度地利用可用信息。

原则上，最佳过滤是通过使用尖点样权重 (cusp-like weights) 和时变滤波长度选择来完成的。然而，这种方法存在严重的成本问题，无论是在实时评估所需的总算能力方面，还是在以脉冲为基础生成（通常由存储系数）归一化 **Wi** 集所需的电路的复杂性方面。



Pixie-16 采用了不同的方法，因为它针对高速处理进行了优化。

它实现了一个固定长度的滤波算法，所有的 W_i 值均相等，实际上对每个新的信号值点 k 重新计算这个和。因此，所实现的方程是：

$$LV_{x,k} = - \sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^k V_i$$

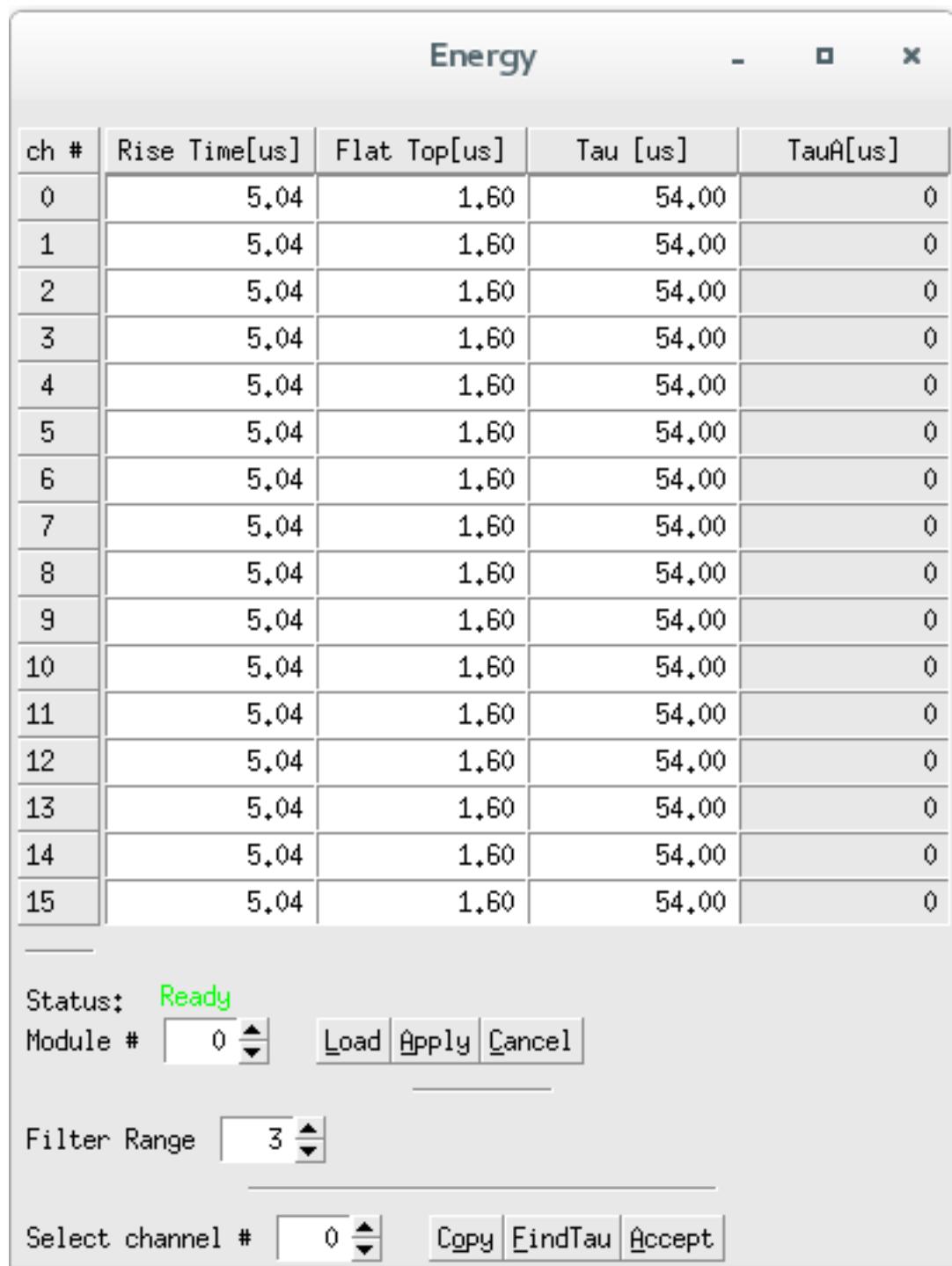
其中过滤长度为 L ，间隙为 G 。系数 L 乘以 $V_{x,k}$ ，因为这里的权重总和未归一化。适应这一因素是微不足道的。

虽然这种关系很简单，但仍然非常有效。首先，这是三角形（或梯形，如果 $g \neq 0$ ）滤波的数字等价物，这是模拟行业的高速处理标准。第二，理论上可以证明，如果信号中的噪声在阶跃上下为白噪声（即高斯分布），这通常是用于高信号率处理的短整形时间的情况，那么方程中的平均值实际上在最小二乘意义上给出 \mathbf{Vx} 的最佳估计。当然，这就是为什么三角过滤在高速率下更受欢迎的原因。

原则上，带时变滤波器长度的三角形滤波既可以获得较高的分辨率，也可以获得更高的吞吐量，但其代价是电路要复杂得多，并且与速率相关的分辨率，这对于许多类型的精确分析来说是不可接受的。在实践中，XIA 的设计可以复制最佳模拟整形器的能量分辨率，同时使其吞吐量增加一倍，这为该方法的有效性提供了实验验证。

7.3.3 Energy

控制界面



界面下方的 Status 显示为 绿色的 Ready 时表示可操作该界面，否则需要等待。底下按钮的操作同上。

- 参数 Rise Time，请参考 *Trapezoidal Filtering* 部分
- 参数 Flat Top，请参考 *Trapezoidal Filtering* 部分
- 参数 Tau，请参考 *Baselines and Preamp. Decay Times* 部分
- 参数 filter range，请参考 *Filter Range* 部分

能量计算的最关键参数是信号衰减时间 Tau。在计算能量时，用来补偿先前脉冲的下降沿。您可以直接为每个通道输入 Tau，也可以单击“FindTau”，让软件自动确定衰减时间。

单击“Accept”将找到的值应用到通道。(如果近似值不变，软件找不到更好的值。)

在高计数率下，脉冲以较高的频率重叠。为了精确地计算这些脉冲的能量或脉冲高度，而不需要等到它们完全衰减回基线水平，Pixie-16 中计算当前脉冲的脉冲高度时采用的脉冲高度计算算法使用衰减时间来计算和消除之前脉冲重叠得指数衰减尾的贡献。

危险：单指数衰减常数

假设脉冲只有一个指数衰减常数。如果脉冲具有多个衰减常数，则可以使用起主要作用的脉冲衰减的衰减常数，但会降低脉冲高度计算的精度。

以下重要参数的一般经验如下：

- 能量滤波平顶时间应大于最长脉冲上升时间。
- 可以改变能量滤波的上升时间，以平衡分辨率和吞吐量。
- 一般来说，能量分辨率随着能量滤波的上升时间的增加而提高，直到当较长的滤波只在测量中增加更多的噪声时达到最佳值。
- 能量滤波区时间 TD 约为 $2 \times (T_{rise} + T_{flat})$ ，泊松统计的最大吞吐量为 $1/(TD \times e)$ 。对于 HPGe 探测器，上升时间为 4-6 us，平顶 1 us 通常是合适的。
- 选择允许设置最佳能量滤波的上升时间的最小能量滤波补偿 (Filter Range)。较大的滤波步长允许较长的滤波总长度之和，但会增加能量滤波的上升时间和平顶时间的可能值的梯度，并增加相对于脉冲上升沿锁定能量滤波输出的抖动。这通常只对非常快的脉冲很重要。

滤波步长

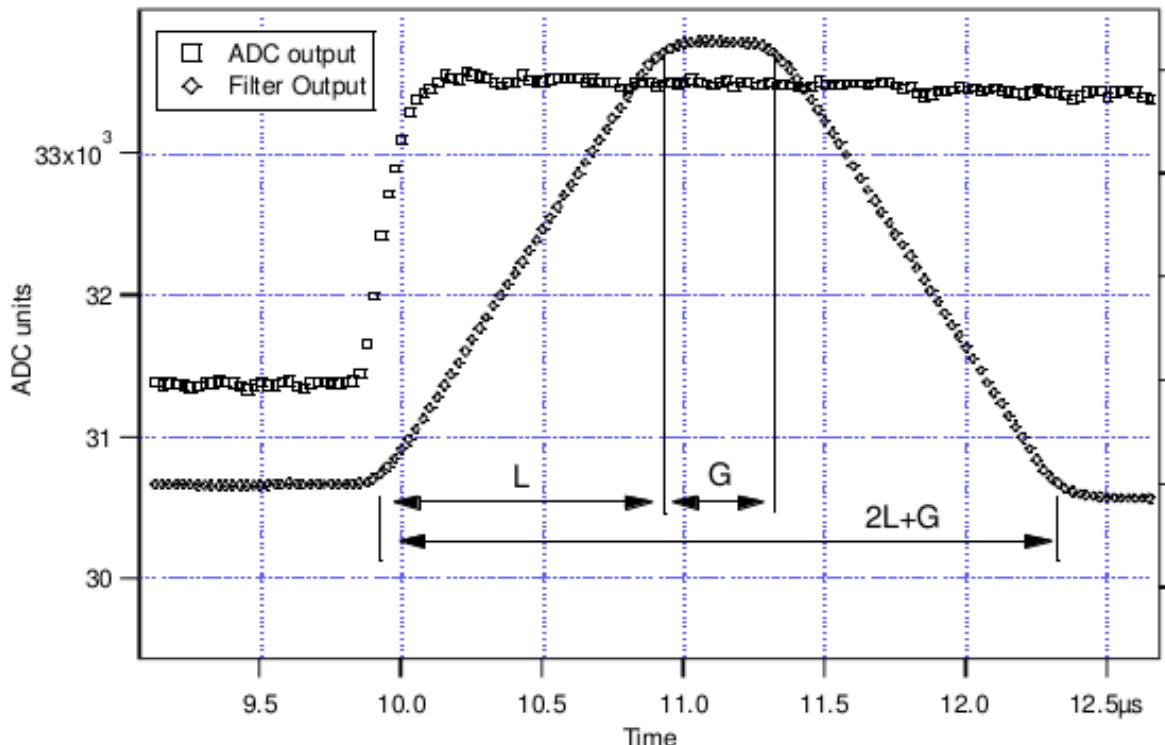
为了适应从数十纳秒到数十微秒各种上升时间的能量滤波器，滤波器在 FPGA 中具有不同的时钟抽取(滤波器范围)。ADC 采样速率为 2 ns, 4 ns 或 10 ns，具体取决于所使用的硬件版本，但在更高的时钟抽取中，几个 ADC 采样在进入能量滤波逻辑之前进行平均。在过滤器范围 1 中，2 个样本被平均，在过滤器范围 2 中 4 个样本，依此类推。由于上升时间和平顶的总和限制为 127 个抽取时钟周期，因此滤波时间粒度和滤波时间仅限于下表中列出的值。

Filter range	Filter granularity	max. $T_{rise}+T_{flat}$	min. T_{rise}	min. T_{flat}
1	0.02μs	2.54μs	0.04μs	0.06μs
2	0.04μs	5.08μs	0.08μs	0.12μs
3	0.08μs	10.16μs	0.16μs	0.24μs
4	0.16μs	20.32μs	0.32μs	0.48μs
5	0.32μs	40.64μs	0.64μs	0.96μs
6	0.64μs	81.28μs	1.28μs	1.92μs

Filter range	Filter granularity	max. $T_{rise}+T_{flat}$	min. T_{rise}	min. T_{flat}
1	0.016μs	2.032μs	0.032μs	0.048μs
2	0.032μs	4.064μs	0.064μs	0.096μs
3	0.064μs	8.128μs	0.128μs	0.192μs
4	0.128μs	16.256μs	0.256μs	0.384μs
5	0.256μs	32.512μs	0.512μs	0.768μs
6	0.512μs	65.024μs	1.024μs	1.536μs

梯形滤波

从这一点开始，仅考虑梯形滤波，因为它是根据公式 $LV_{x,k} = -\sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^k V_i$ 在 Pixie-16 模块中实现的。将长度 $L = 1 \mu s$ 和平顶 $G = 0.4 \mu s$ 的滤波器应用于伽马射线事件的结果如下图所示。滤波器输出形状明显为梯形，上升时间等于 L ，平顶等于 G ，对称下降时间等于 L 。基带宽度是滤波器降噪特性的一阶测量值，此时为 $2L+G$ 。



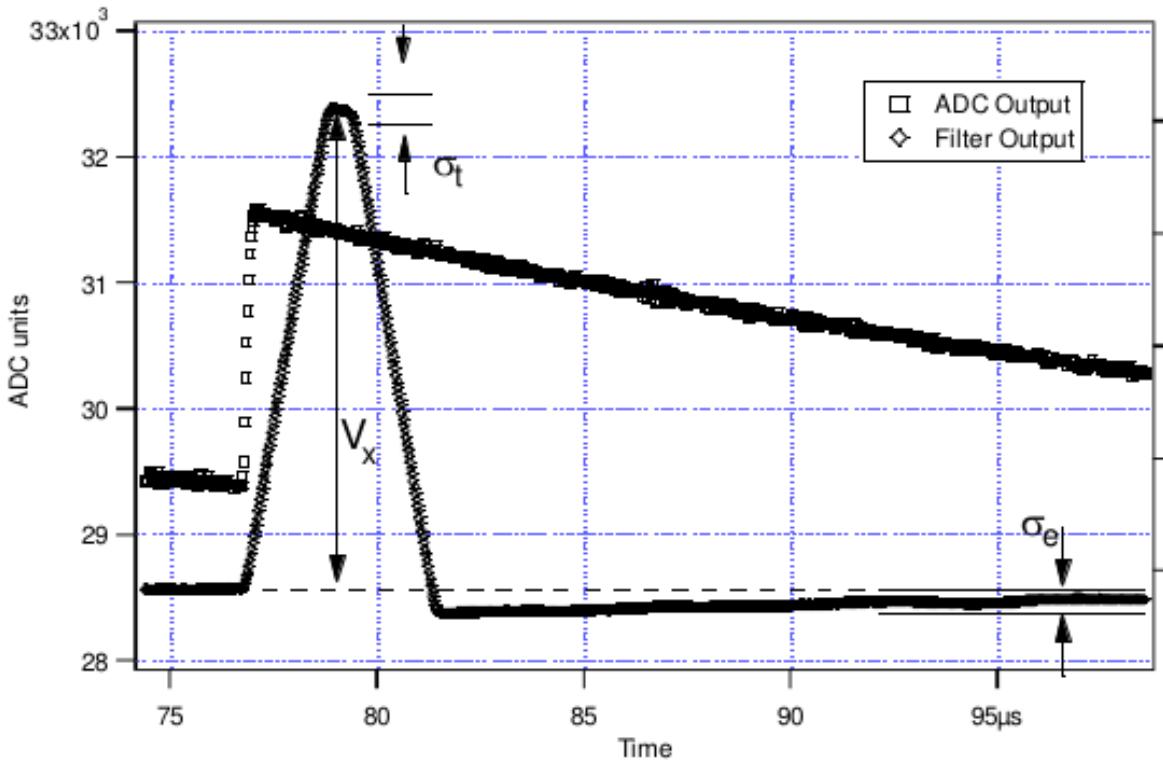
这在比较 Pixie-16 模块的噪声性能和模拟滤波放大器时提出了几个重点：

- 首先，半高斯滤波器通常由成形时间指定。- 它们的上升时间通常是这以时间的两倍，并且它们的脉冲不对称，因此基带宽度约为成形时间的 5.6 倍或上升时间的 2.8 倍。
- 因此，半高斯滤波器通常具有比具有相同上升时间的三角滤波器稍好的能量分辨率，因为它具有更长的滤波时间。- 这通常适用于通过将三角形上升时间拉伸一点来提供三角和半高斯滤波的放大器，因此真正的三角形上升时间通常是所选半高斯上升时间的 1.2 倍。- 当其能量分辨率与具有相同标称上升时间的数字系统相比时，这也为模拟系统带来明显的优势。

数字形梯形脉冲的一个重要特征是在基带宽度 $2L+G$ 时极其尖锐地终止。这可以与模拟滤波脉冲进行比较，模拟滤波脉冲其尾部可能持续高达上升时间的 40%，这是由于模拟滤波器的有限带宽引起的现象。从下面可以看出，这种尖锐的终止使数字滤波器在无堆积吞吐量方面具有明确的速率优势。

基线与前放衰减时间

图中显示了较长时间间隔内的事件以及当没有伽马射线脉冲时滤波器如何处理区域中的前置放大器噪声。



可以看出，滤波器的效果是减小波动的幅度并降低它们的高频含量。该区域称为基线，因为它建立了要测量伽马射线峰值幅度 V_x 的参考电平。基线的波动具有标准偏差 σ_e ，其被称为系统的电子学噪声，该数字取决于所使用的滤波器的上升时间。在这种噪声的基础上，伽马射线峰值会产生额外的噪声项，即 Fano 噪声，这是由于伽马射线在探测器中被吸收时产生的电荷量 Q_x 的统计涨落引起的。此 Fano 噪声 σ_f 与电子噪声偶和，因此测量 V_x 时的总噪声 σ_t 来自：

$$\sigma_t = \sqrt{\sigma_f^2 + \sigma_e^2}$$

Fano 噪声仅是探测器材料的特性。另一方面，电子学噪声可能来自前置放大器和放大器。然而，当前置放大器和放大器设计良好且匹配良好时，放大器的噪声贡献基本上可以忽略不计。然而，在数字脉冲处理器的混合模拟-数字环境中实现这一点是一项非常重要的任务。

使用 RC 型前置放大器时，前置放大器的斜率很少为零。每一步都以指数方式衰减回前置放大器的 DC 电平。在这种衰减过程中，基线显然不是零。这可以在上图中看到，其中脉冲之后指数衰减期间的滤波器输出低于初始水平。另外请注意一点，平顶区域向下倾斜。

使用衰减常数 τ 可以将基线映射回 DC 级别。这允许精确测定伽马射线能量，即使脉冲位于前一个脉冲的下降斜率上。作为前置放大器的一个特征， τ 的值必须由用户和获取程序确定并设置到模块中。

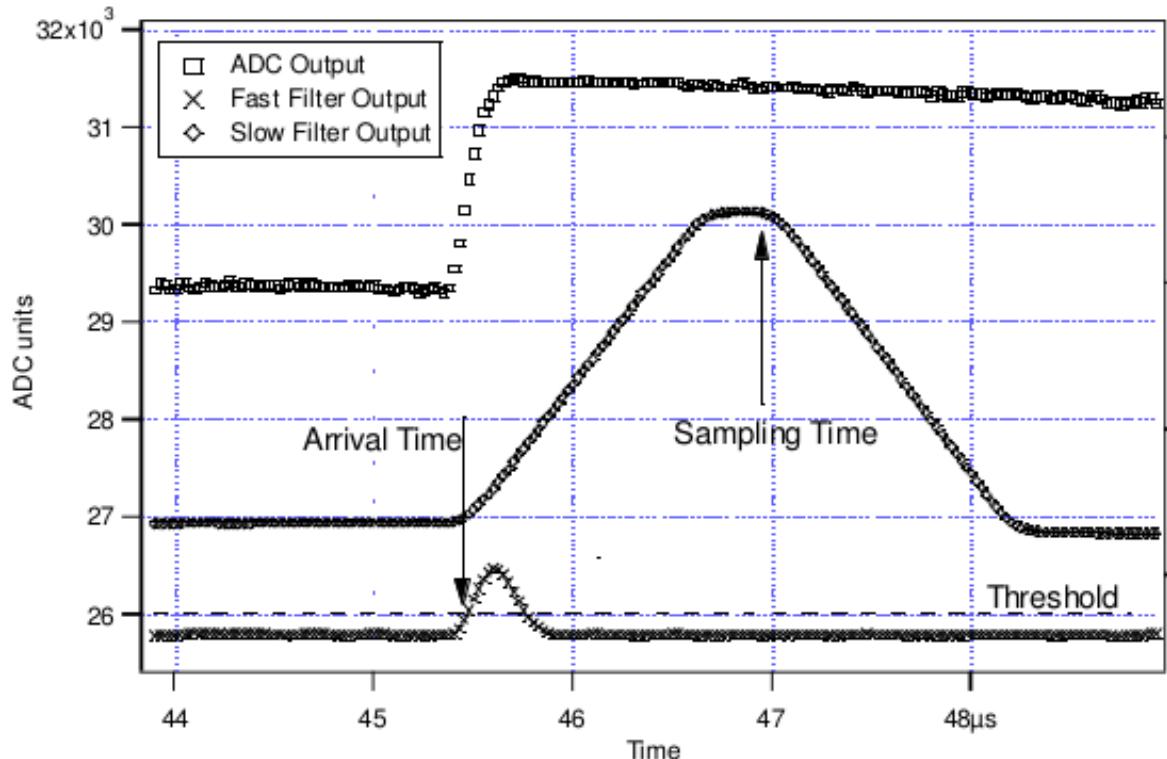
堆积检测

如上所述，目标是为检测到的每一条伽马射线捕获 V_x 值，并使用这些值构建一个能谱。

注解：此过程在数字和模拟系统之间也存在显着差异。在模拟系统中，峰值必须“捕获”到模拟存储设备（通常是电容器）中，并“保持”直到数字化为止。然后，该数字值用于更新存储位置以构建所需的能谱。在此模数转换过程中，系统对其它事件无效，这会严重降低系统吞吐量。即使是单通道分析仪系统也会在此阶段引入显著的死时间，因为它们必须等待一段时间（通常为几微秒）才能确定是否满足窗口条件。

数字系统在这方面效率更高，因为滤波器输出的值已经是数字值。所需要的只是获取滤波器加和数值，重建能量 V_x ，并将其添加到能谱中。在 Pixie-16 中，滤波器加和数值在 FPGA 中不断更新，并被捕获到事件缓冲器中。重建能量并增加能谱由 DSP 完成，因此 FPGA 可以立即采集新数据（除非缓冲区已满）。这是数字系统中增强吞吐量的重要来源。

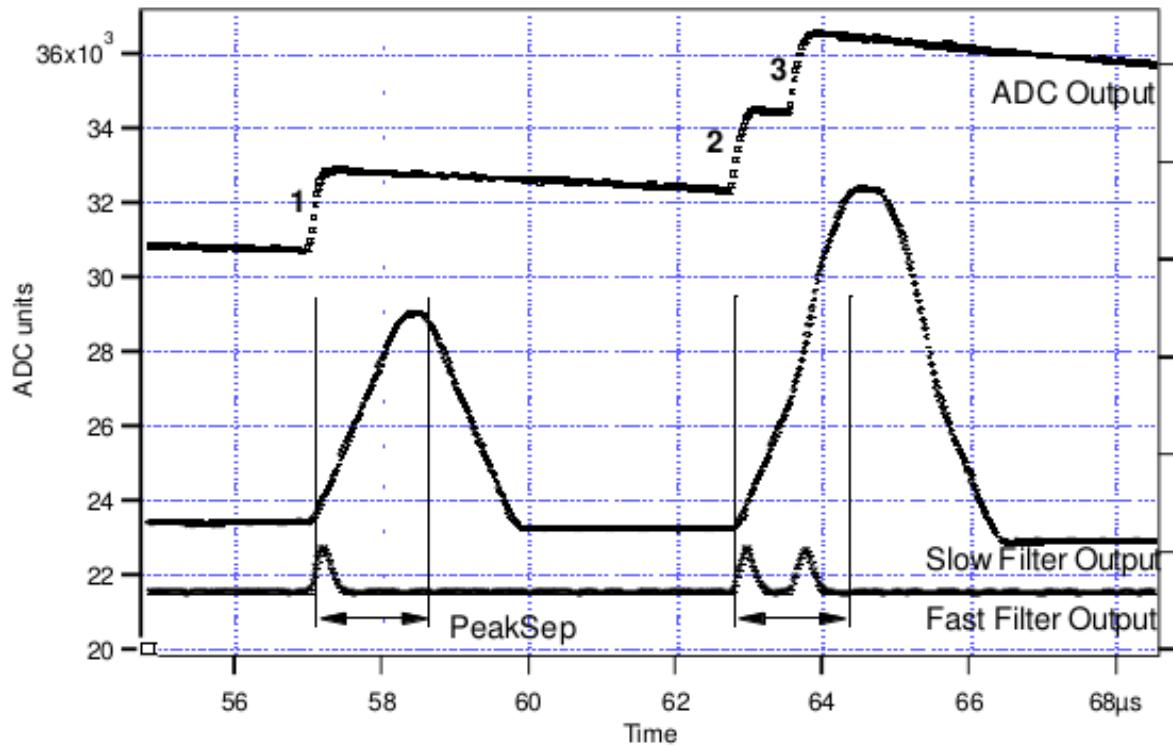
Pixie-16 模块中的峰值检测和采样如下图所示进行处理。图中实现了两个梯形滤波器，快速滤波器和慢速滤波器。快速滤波器用于检测伽马射线的到达，慢速滤波器用于测量 V_x ，在较长的滤波器上升时间内降低噪声。快速滤波器的滤波器长度 $L_f = 0.1 \mu s$ ，间隙 $G_f = 0.1 \mu s$ 。慢滤波器的 $L_s = 1.2 \mu s$ ， $G_s = 0.35 \mu s$ 。



通过将快速滤波器输出与用户设置的数字常数 **THRESHOLD** 进行数字比较来检测伽马射线步进（在前置放大器输出中）的到达。越过阈值则开始延迟线等待 **PEAKSAMP** 个时钟周期，到达适当的时间来采样慢速滤波器的值。由于数字滤波过程是确定性的，**PEAKSAMP** 仅取决于快速和慢速滤波器常数的值。

在 **PEAKSAMP** 之后捕获的慢滤波器值则是慢速数字滤波器对 V_x 的估计。使用延迟线允许甚至在 **PEAKSAMP** 间隔内对多个脉冲进行采样（尽管滤波器值本身不是单个脉冲高度的正确表示）。

捕获的值 V_x 将仅是相关伽马射线能量的有效测量，条件是滤波后的脉冲在时间上与其前一个和后一个相邻脉冲足够好地分开，使得它们的峰值幅度不会因梯形滤波器的作用而失真。也就是说，如果脉冲没有堆积。通过参考下图可以理解相关问题，图中示出了通过各种间隔分开的 3 个伽马射线。快速滤波器的滤波器长度 $L_f = 0.1 \mu s$ ，间隙 $G_f = 0.1 \mu s$ 。慢滤波器的 $L_s = 1.2 \mu s$ ， $G_s = 0.35 \mu s$ 。



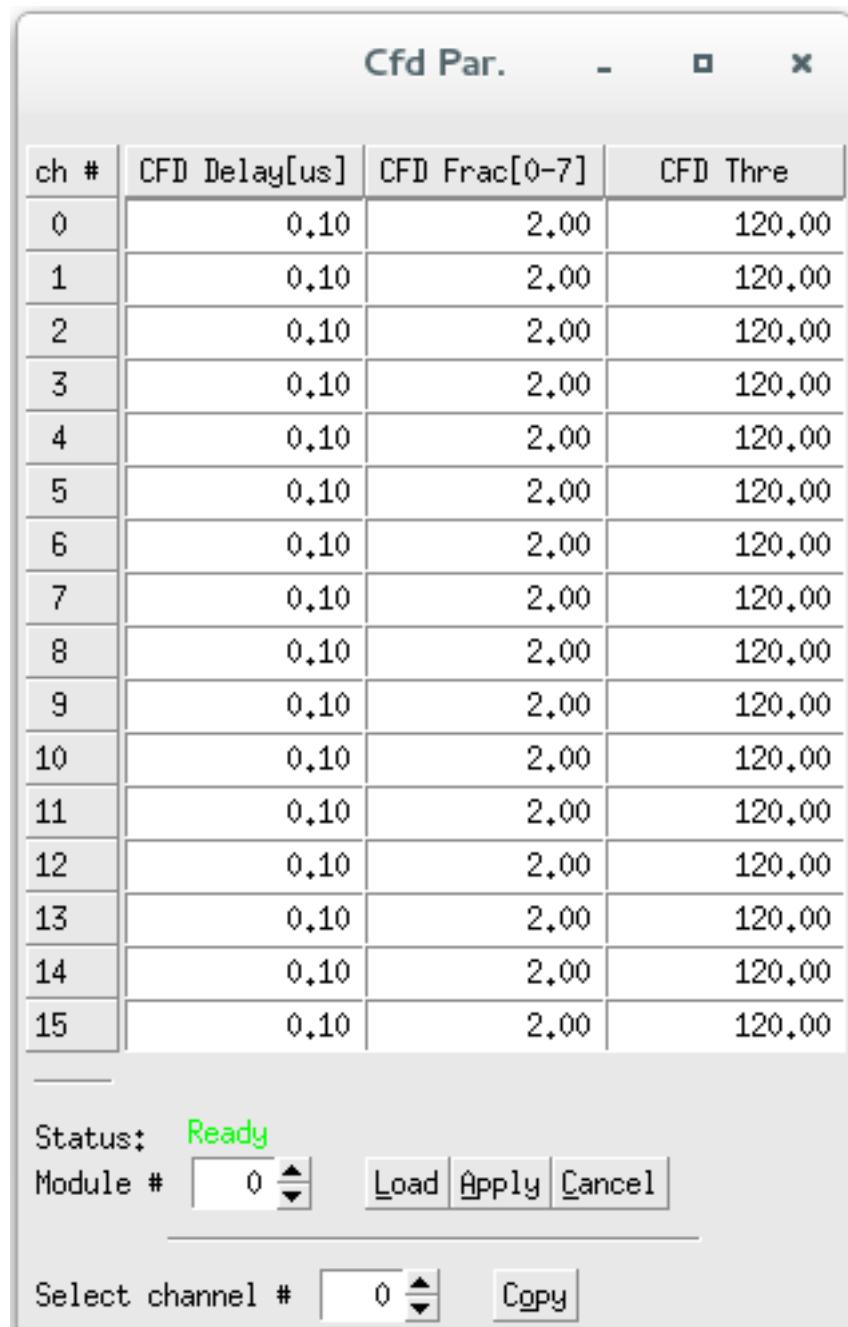
由于梯形滤波器是线性滤波器，因此其对一系列脉冲的输出是其系列中各个成员的输出的线性和。当一个脉冲的上升沿位于其邻居的峰值（特别是采样点）之下时，发生堆积。因此，在图中，峰 1 和 2 被充分分离，使得峰 2 的前沿在脉冲 1 的峰值之后下降。因为梯形滤波器函数是对称的，这也意味着脉冲 1 的后沿也不会落在脉冲 2 的峰中。为此，两个脉冲必须至少间隔 $L+G$ 。因此，在本示例中看到峰值 2 和 3，其间隔小于 1.0 μs ，具有 1.2 μs 的上升时间。

这导致了一个重要的观点：脉冲是否遭受缓慢的堆积主要取决于所使用的过滤器的上升时间。在给定的平均信号速率下发生的堆积量将随着上升时间的增加而增加。

由于快速滤波器上升时间仅为 0.1 μs ，因此这些伽马射线脉冲不会在快速滤波器通道中堆积。因此，Pixie-16 模块可以通过在脉冲到达时间之后测量间隔 PEAKSEP 的快速滤波器来测试慢速通道堆积。如果在该间隔中没有检测到第二个脉冲，则没有后沿堆积并且脉冲可以用于采集。PEAKSEP 通常设置为接近 $L+G+1$ 的值。脉冲 1 通过此测试，如上图所示。然而，脉冲 2 未通过 PEAKSEP 测试，因为脉冲 3 低于 1.0 μs 。注意，通过梯形滤波器的对称性，如果脉冲 2 由于脉冲 3 而被拒绝，则脉冲 3 由于脉冲 2 而被类似地拒绝。

7.3.4 CFD

控制界面



TODO

100 / 250 MHz 模块

以下 CFD 算法在 100 MHz (版本 B, C, D 和 F) 和 250 MHz (版本 F) Pixie-16 模块的信号处理 FPGA 中实现。

假设数字化波形可以用数据序列 $\text{Trace}[i]$ 表示, $i = 0, 1, 2, \dots$ 。首先, 数字化波形的快速滤波器响应 (FF) 计算如下:

$$FF[i] = \sum_{j=i-(FL-1)}^i \text{Trace}[j] - \sum_{j=i-(2 \times FL+FG-1)}^{i-(FL+FG)} \text{Trace}[j]$$

这里, FL 称为快速长度, FG 称为数字梯形滤波器的快速间隙。CFD 计算如下:

$$CFD[i + D] = FF[i + D] \times (1 - w/8) - FF[i]$$

其中 D 称为 CFD 延迟长度, 而 w 称为 CFD 缩放因子 ($w = 0, 1, \dots, 7$)。

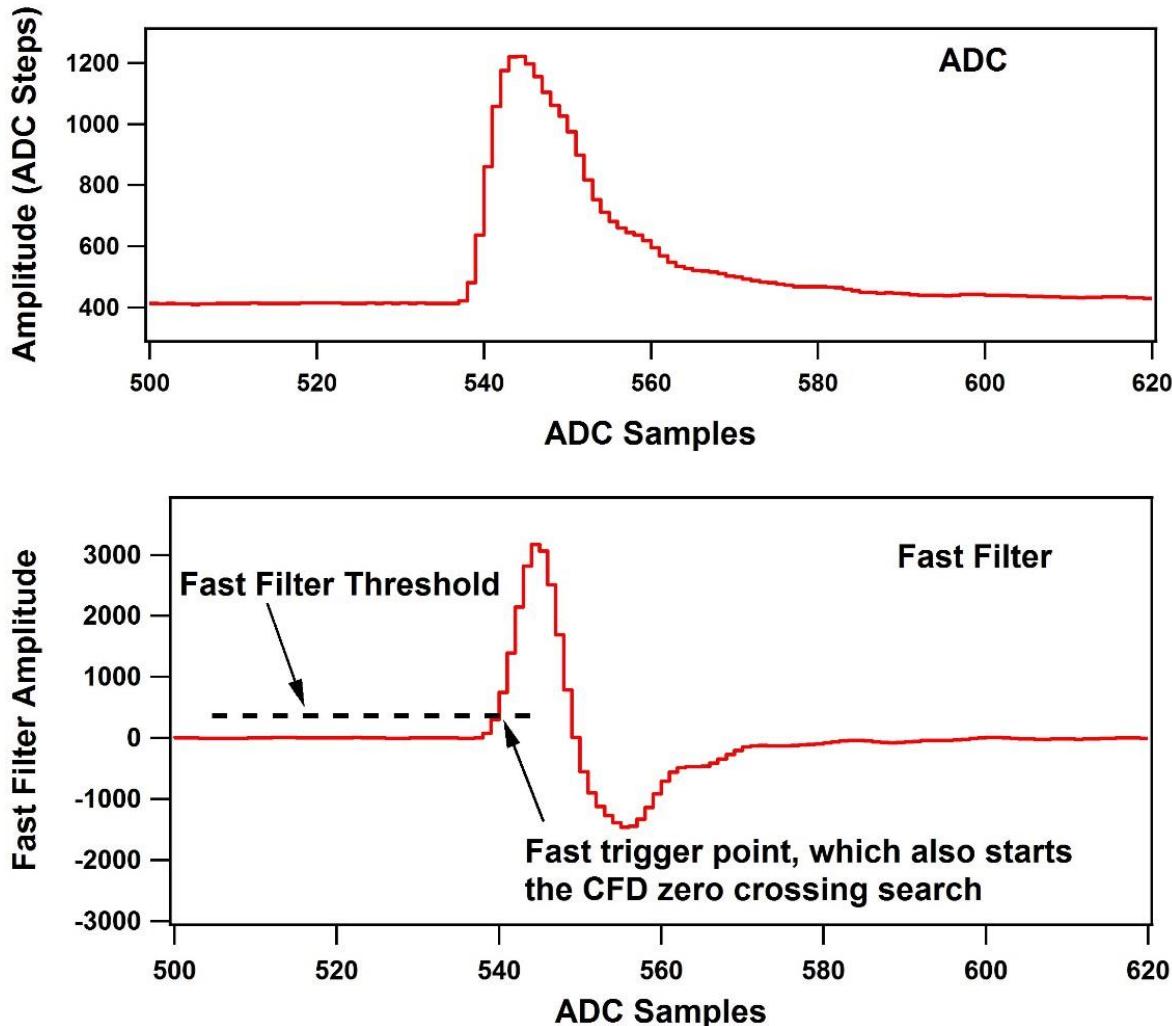
CFD 过零点由 $CFD[i] \geq 0$ 和 $CFD[i + 1] < 0$ 来确定。时间戳被标记在跟踪点 i , 分数时间 f 由过零点前后两个 CFD 响应幅度之比得出。

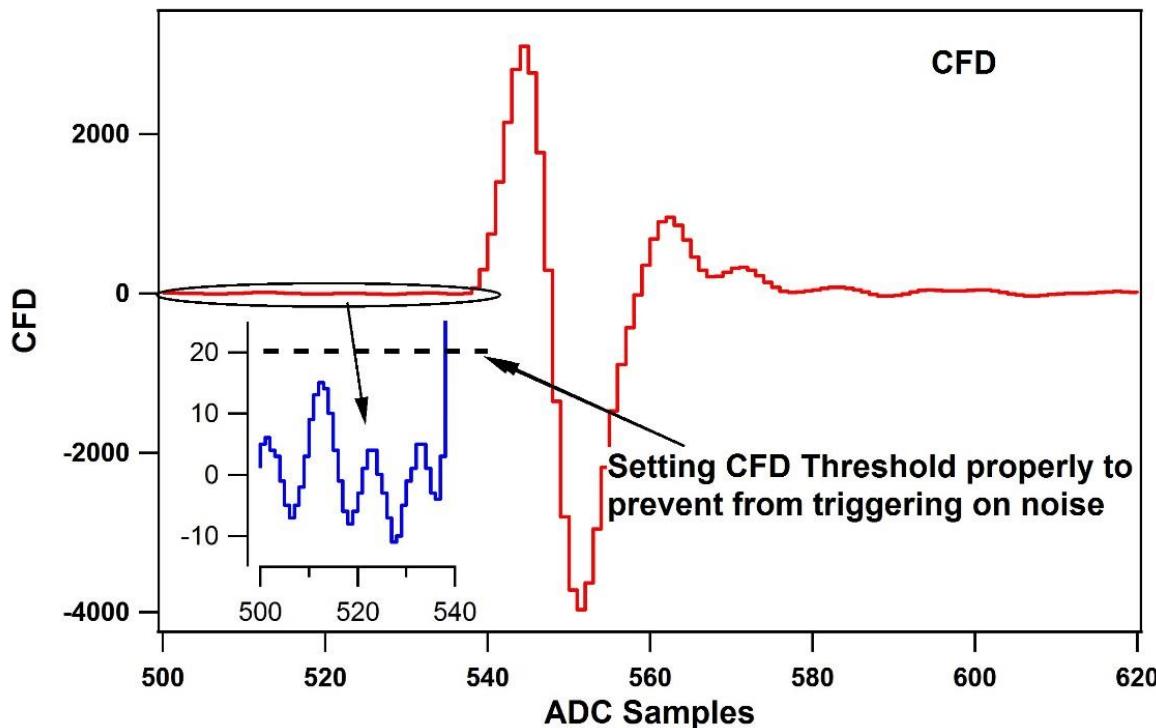
$$f = \frac{CFDout1}{CFDout1 - CFDout2}$$

其中 CFDout1 是过零点之前的 CFD 响应幅度, CFDout2 是过零点之后的 CFD 响应幅度 (分母中使用减法, 因为 CFDout2 为负数)。Pixie-16 DSP 按以下方式计算 CFD 最终值, 并将其存储在输出数据流中以进行在线或离线分析。

$$CFD = \frac{CFDout1}{CFDout1 - CFDout2} \times N$$

其中 N 是比例因子, 分别对应于 100 MHz 模块的 32768 和对应于 250 MHz 模块的 16384。





上图分别显示了原始采集波形，其快速滤波器响应和 CFD 响应。

最上图显示了原始 ADC 采集波形。在使用公式 $FF[i]$ 计算原始 ADC 采集波形上的快速滤波器响应之后，将快速滤波器响应与快速滤波器阈值进行比较，如中间图所示。快速滤波器响应超过快速滤波器阈值的 ADC 采样称为快速触发点，它也开始搜索 CFD 过零点。

CFD 响应是使用公式 $CFD[i + D]$ 计算的，并显示在上面最后一张图中（对于固件中的实际实现，快速滤波器响应 FF 在用于计算 CFD 响应之前会稍有延迟。在快速触发后有足够的 CFD 响应点来寻找过零点）。为了防止实际触发之前 CFD 响应中的噪声导致 CFD 触发过早，使用称为 CFD 阈值的 DSP 参数来抑制那些由噪声引起的过零点。但是，如果在快速触发后的某个时间段内（通常为 32 个时钟周期）找不到过零点（例如，由于不必要的高阈值），则会发出强制 CFD 触发，并在事件标头中设置标志来表示此事件记录的 CFD 时间是无效的。

但是，事件将仍然具有有效的时间戳记，当快速过滤器超过触发阈值时，该时间戳将由快速过滤器触发器锁存。前述的 CFD 参数对应于以下 DSP 参数。

CFD Parameters	DSP Parameters
FL	FastLength
FG	FastGap
Fast Filter Threshold	FastThresh
D	CFDDelay
W	CFDScale (valid values: 0, 1, 2, ... and 7)
CFD Threshold	CFDThresh

注解: 250 MHz

在 250 MHz Pixie-16 模块中，事件时间戳以 125 MHz 时钟滴答计数，即 8 ns 间隔进行计数，并且也以 8 ns 间隔捕获两个连续的 250 MHz ADC 样本。

CFD 触发也以 125 MHz 的频率运行，但是 CFD 过零点仍报告为两个相邻的 250 MHz ADC 采样之间的分数时间，由 FPGA 在一个 125 MHz 时钟周期内对其进行处理。

但是，CFD 过零点可能处于捕获的 250 MHz ADC 波形的奇数或偶数时钟周期中。

因此，固件在输出数据流中输出“CFD trigger source”位，以指示 CFD 过零点是处于捕获的 250 MHz ADC 波形的奇数还是偶数时钟周期中。

注解：100 MHz

在 100 MHz Pixie-16 模块中，事件时间戳，CFD 触发和 ADC 波形捕获都使用相同的 100 MHz 时钟执行。因此，无需报告 100 MHz Pixie-16 模块的“CFD trigger source”。

500 MHz 模块

上一节中讨论的针对 100 MHz 和 250 MHz Pixie-16 模块的 CFD 算法也可以采用以下格式编写：

$$CFD(k) = w \cdot \left(\sum_{i=k}^{k+L} a(i) - \sum_{i=k-B}^{k-B+L} a(i) \right) - \left(\sum_{i=k-D}^{k-D+L} a(i) - \sum_{i=k-D-B}^{k-D-B+L} a(i) \right)$$

其中 $a(i)$ 是 ADC 跟踪数据， k 是索引， w , B , D 和 L 是 CFD 参数。

在调整参数 w 、 B 、 D 和 L 的能力方面，在 500 MHz Pixie-16 模块中实现的 CFD 算法与在 100 MHz 和 250 MHz Pixie-16 模块中实现的 CFD 算法相比是特殊的。

其原因是在 500 MHz 的 Pixie-16 模块中，以 500 MHz 的速度进入 FPGA 的 ADC 数据首先以 1:5 的比率减慢，换言之，FPGA 以 100 MHz 的速率，即每 10 ns 捕获 5 个 ADC 采样。然后，FPGA 通过首先建立 ADC 样本的和，然后计算延迟和非延迟和之间的差异，直到找到过零点，试图在该 10 ns 内找到任意两个相邻 2 ns ADC 样本之间的 CFD 触发点。然而，在 500 MHz 的 Pixie-16 模块中，FPGA 没有足够的资源来为 5 个 ADC 采样并行地构建可变延迟的和。因此，500 MHz 模块的 CFD 算法是使用一组固定 CFD 参数来实现的，如表 *Fixed CFD Parameter Values for 500 MHz Pixie-16 Modules*。实验表明，这些固定参数对 LaBr3(Ce) 探测器的性能最好。

CFD Parameters	Fixed Values for 500 MHz Modules
w	1
B	5
D	5
L	1

500 MHz Pixie-16 模块给出的 CFD 时间由两部分组成：在 5 个 ADC 样本内进行移位，以及在两个 ADC 样本之间发生 CFD 过零点的时间。由于以下定义了 3-bit CFD trigger source[2:0]，因此报告了 5 个 ADC 采样中的移位。

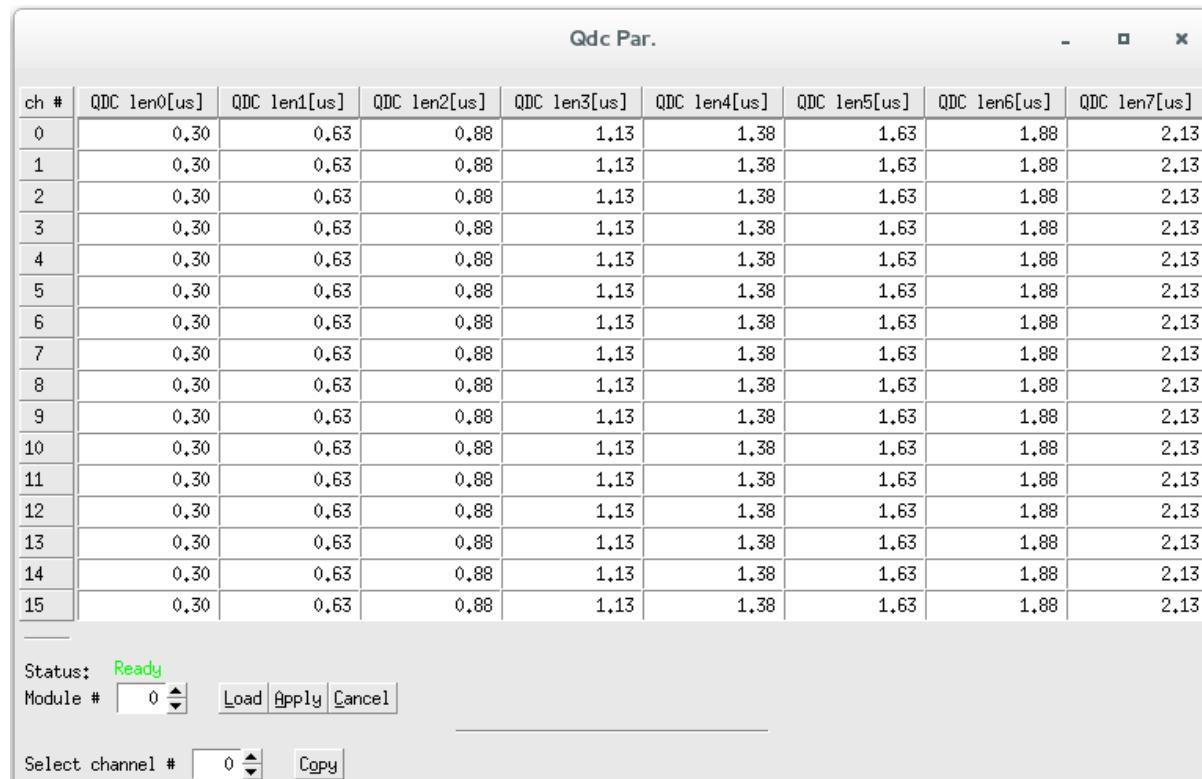
CFD Trigger Source [2:0]	Zero Crossing Point (ZCP) Location
000	ZCP occurred between the 5th ADC sample of the previous 5-sample group and the 1st ADC sample of the current 5-sample group
001	ZCP occurred between the 1th ADC sample of the current 5-sample group and the 2nd ADC sample of the current 5-sample group
010	ZCP occurred between the 2nd ADC sample of the current 5-sample group and the 3rd ADC sample of the current 5-sample group
011	ZCP occurred between the 3rd ADC sample of the current 5-sample group and the 4th ADC sample of the current 5-sample group
100	ZCP occurred between the 4th ADC sample of the current 5-sample group and the 5th ADC sample of the current 5-sample group
101	Not used
110	Not used
111	CFD trigger is forced, so CFD time is invalid

CFD 分数时间如下：

$$CFD = \frac{CFDout1}{CFDout1 - CFDout2} \times 8192$$

7.3.5 QDC

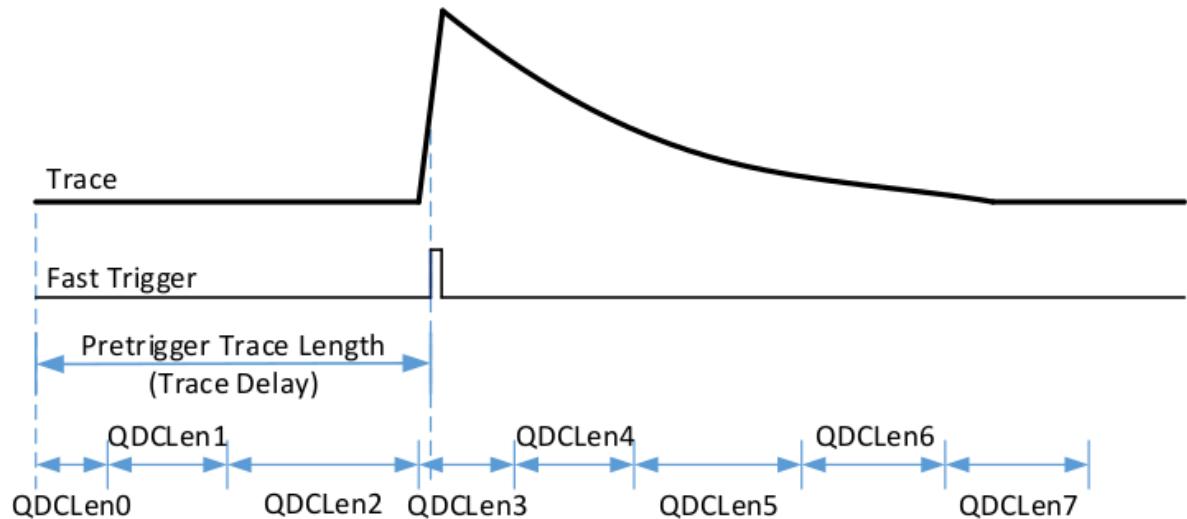
控制界面



在 Pixie-16 模块的信号处理 FPGA 中为每个通道计算八个 QDC 积分，每个长度可以具有不同的长度，如果用户要求，则将这些积分写入列表模式输出数据流。

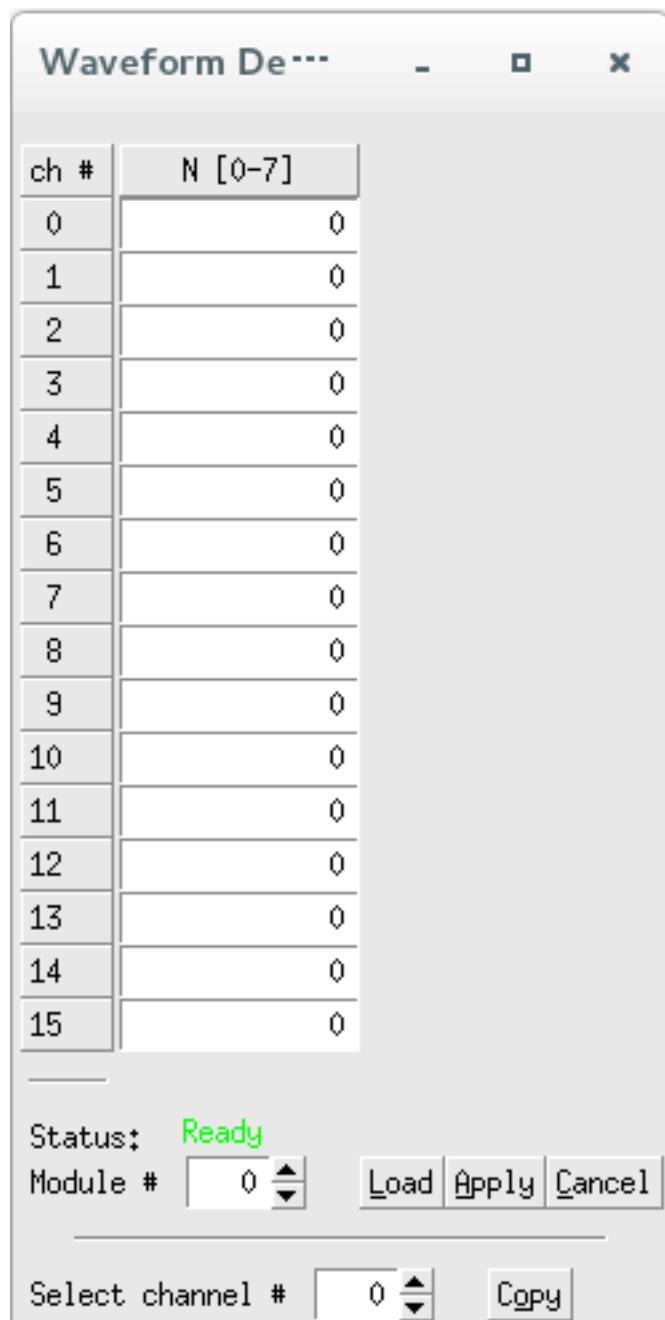
QDC 的积分起点从记录波形的点开始，该波形点比触发点早 *Pre-trigger Trace Length* 或 *Trace Delay*，触发模式是 CFD 触发还是通道快速触发取决于 CFD 触发模式是否开启。

连续一个接一个地计算八个 QDC 积分，但它们并不重叠。八个间隔全部过去后，QDC 积分的记录结束。



7.3.6 Decimation

控制界面

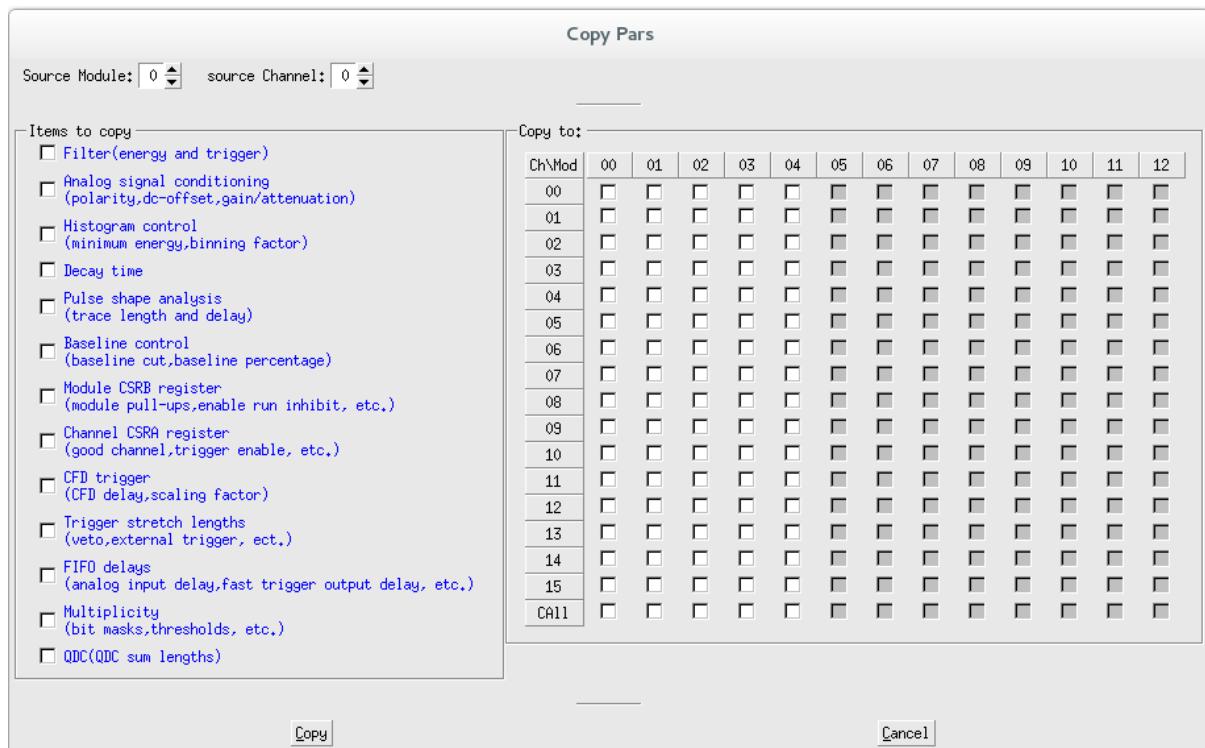


在 PKU 固件中，对于 100 MHz 模块定制了波形降频输出。

在采集波形时候，增加了降频输出的功能，采取的策略为可选择 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128 (分别对应参数 0-7) 频率的输出，即多少个点保留一个点。保留的点是平均后的值。

7.3.7 Copy Pars

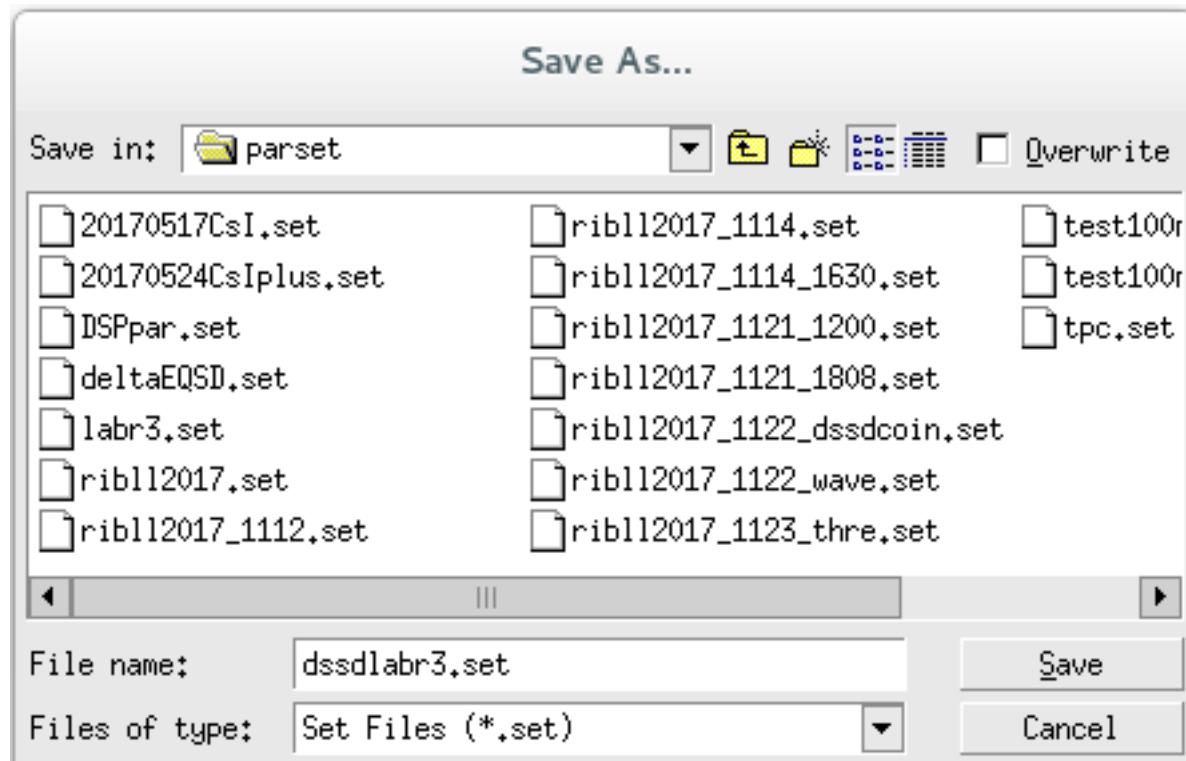
控制界面



该窗口用来快速进行通道之间的参数拷贝。

7.3.8 Save2File

控制界面



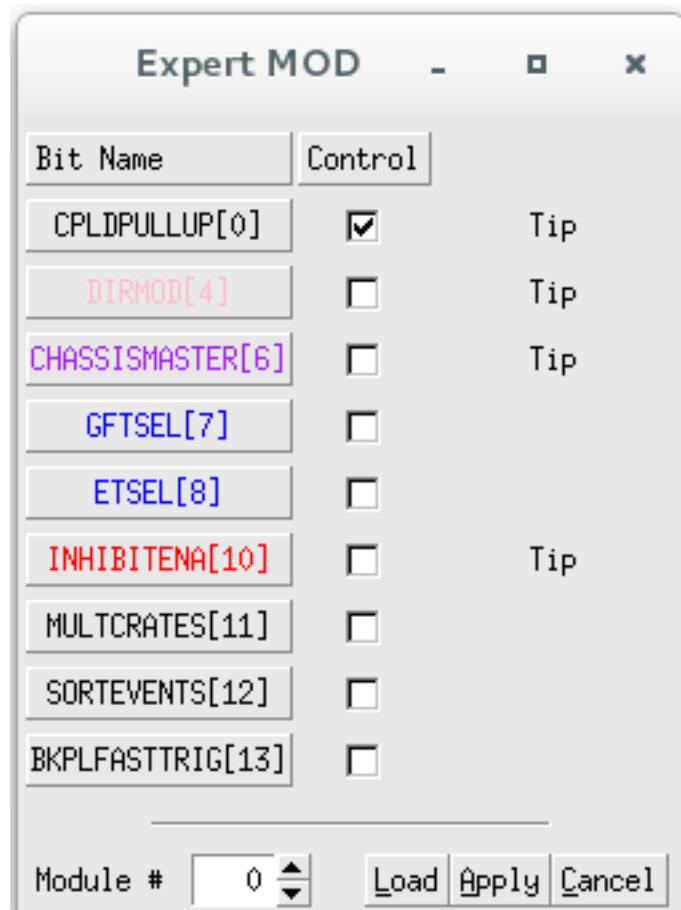
正如你所知道的一样，DSP 参数被保存在一个后缀为“.set”的文件中，当你配置好参数之后，应该保存它以便于之后使用。

7.4 Expert

本下拉栏中调节内容为高阶内容，需要对获取逻辑有一定基础的人学习掌握。

7.4.1 Module Variables

控制界面



除了分配全局时钟信号 (global clock signal) 外，Pixie-16 后方 I/O 触发模块还可以共享全局触发并运行同步信号。全局触发信号包括全局验证触发 (global validation trigger) 和全局快速触发 (global fast trigger)，以及 Pixie-16 FPGA 数据存储缓冲器的满标志信号。运行同步信号包括可以在多个机箱之间共享的同步运行开始和停止信号。

为了能够分配此类全局触发器和运行同步信号，必须正确设置某些 Pixie-16 参数。控制触发器分配和运行同步的参数是模块控制寄存器 B (ModCSRB)。

ModCSRB 是一个 32 位参数，其中 32 位中的每个位控制 Pixie-16 模块的不同操作模式。

注解：触发分配和运行同步

对于安装在主机箱中的 System Director 模块，ModCSRB 的位 0、4、6 和 11 应该设置为 1 (选中并启用)。

对于从机箱中安装的 master 模块，应将 ModCSRB 的位 0、6 和 11 设置为 1 (选中并启用)。

对于安装在从机箱和主机箱中的常规模块，ModCSRB 的第 11 位应设置为 1 (选中并启用)。

寄存器定义

模块控制寄存器 B 作用于整个模块。

• bit 0 - MODCSRB_CPLDPULLUP

- 通过板载 CPLD 为背板上的 PXI 触发线启用上拉 (pullups)。
- 使用上拉时 (pullups)，这些 PXI 触发线默认为逻辑高电平状态。

- 仅当一个模块主动将线拉至逻辑低状态时，该线才会处于低状态。
- 因此，通过这些 PXI 触发线传输的信号为低电平有效信号。
- 注意：每个机箱仅对一个模块启用此位（例如机箱主模块）

- **bit 4 - MODCSRB_DIRMOD**

- 将此模块设置为 Director 模块，以便它可以通过后 I/O 触发模块向所有机箱发送触发，波形和事件头 DPM 满信号并运行同步信号。
- 这里的触发包括快速触发和验证触发
- 注意：仅对所有机箱中的一个模块启用此位（例如，多机箱配置中的 System Director 模块）

- **bit 6 - MODCSRB_CHASSISMASTER**

- 将此模块设置为机箱主模块，以便它可以将触发，波形和事件头 DPM 满信号发送并运行同步信号到本机箱的背板。
- 这里的触发包括快速触发和验证触发
- 注意：每个机箱仅对一个模块启用此位（例如机箱主模块）

- **bit 7 - MODCSRB_GFTSEL**

- 选择外部快速触发源（=1：外部验证触发，=0：外部快速触发。用来在 Pixie-16 前面板输入连接器上交换这两个信号）

- **bit 8 - MODCSRB_ETSEL**

- 选择外部验证触发源（=1：外部快速触发，=0：外部验证触发，用来在 Pixie-16 前面板输入连接器上交换这两个信号）

- **bit 10 - MODCSRB_INHIBITENA**

- 启用（=1）或禁用（=0）使用外部 INHIBIT 信号。
- 器用该功能后，处于逻辑高电平状态的外部 INHIBIT 信号将阻止运行开始，直到该外部 INHIBIT 信号变为逻辑低电平状态。

- **bit 11 - MODCSRB_MULTCRATES**

- 将此模块设置为以多机箱模式（=1）或以本地机箱模式（=0）运行。
- 如果模块以多机箱模式运行，它将使用波形和事件头 DPM 满信号以及运行同步信号，这些信号是在多个机箱中生成和分配的。
- 如果模块以本地机箱模式运行，它将使用波形和事件头 DPM 满信号并运行在本地机箱中生成的同步信号。

- **bit 12 - MODCSRB_SORTEVENTS**

- 在将事件存储在外部 FIFO 中之前，根据事件的时间戳对来自 Pixie-16 模块的所有 16 个通道的事件进行排序（=1）或不对事件进行排序（=0）。
- 注意：所有 16 个通道必须具有相同的 DAQ 参数设置才能使用此功能

- **bit 13 - MODCSRB_BKPLFASTTRIG**

- 启用（=1）或禁用（=0），将 16 个本地快速触发发送到机箱背板上的 16 条线路。
 - 注意：机箱的每个 PCI 总线段（bus segment）中只有一个模块可以启用此选项（不限于机箱主模块，例如，每个 PCI 总线段中的任何模块）
-

7.4.2 CSRA

控制界面

Ch #	FTS	MSE	GC	CSE	BDA	SP	CTV	HE	TC	EQS	ECT	MVT	ERB	CVT	IR	NPR	IPR	NTL	GTS	CVS	MVS	ETS
0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
2	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
3	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
4	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
5	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
6	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
7	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
8	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
9	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
10	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
11	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
12	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
13	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
14	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
15	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>						
CA11	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Module #

- 黄色 FTS、GTS 组合来选择 channel fast trigger:
 - 两个均不选时为 local fast trigger
 - 选择 FTS 时为 latched module fast trigger
 - FTS 不选、GTS 选上时为 latched channel validation trigger
- 蓝色 MSE、CSE、MVT、CVT 用来选择 module/channel validation trigger:
 - MVT 为是否开启 module validation trigger
 - CVT 为是否开启 channel validation trigger
 - MSE 选择 module validation trigger 来源于 System FPGA 还是前面板 module GATE
 - CSE 选择 channel validation trigger 来源于 System FPGA 还是前面板 channel GATE
- 粉红色 NPR、IPR 组合选择 pileup 事件的处理:
 - 两个均不选时则记录所有事件，堆积事件能量值为无效
 - NPR 选择 IPR 不选时不记录堆积事件
 - NPR 不选 IPR 选择时堆积事件记录波形、不堆积时候不记录波形
 - 两个均选择时只记录堆积事件
- 绿色 CTV、CVS、MVS 用来选择 module/channel veto:
 - MVS 选择 module veto 来源于前面板 module GATE 还是 module validation trigger
 - CVS 选择 channel veto 来源于前面板 channel GATE 还是 channel validation trigger
 - CTV 为是否开启 channel trigger veto
- 红色为基础设置
 - 黑色 NTL 是否保留超出量程的波形
 - 黑色 ETS 是否记录外部时钟的数据
- 剩余的 BDA 不选，HE 不管

寄存器定义

通道控制寄存器 A 分别用于每个通道

- **bit 0 - CCSRA_FTRIGSEL**

- 通道快速触发选择 (=1: 系统 FPGA 的模块快速触发; =0: 选择取决于另一位的值 CCSRA_GROUPTRIGSEL: 如果 CCSRA_GROUPTRIGSEL=1, 则选择系统 FPGA 的通道验证触发, 如果 CCSRA_GROUPTRIGSEL=0, 选择此通道的本地快速触发)

- **bit 1 - CCSRA_EXTTRIGSEL**

- 模块验证触发选择 (=1: 来自 Pixie-16 前面板模块门 LVDS 连接器的模块门输入; =0: 来自系统 FPGA 的模块验证触发)

- **bit 2 - CCSRA_GOOD**

- 将此通道设置为“好”通道 (=1) 或“不好”通道 (=0)。
 - **当某个通道设置为“不好”通道时, 它仍会生成本地快速触发, 可用于多重性计算等, 但是此通道不会记录列表模式数据或 MCA 数据, 并且不会更新其基线数值。

- **bit 3 - CCSRA_CHANTRIGSEL**

- 通道验证信号选择 (=1: 来自 Pixie-16 前面板通道门 LVDS 连接器的通道门输入; =0: 来自系统 FPGA 的通道验证触发)

- **bit 4 - CCSRA_SYNCDATAACQ**

- 选择此通道的同步数据采集级别 (=1: 系统中任何 Pixie-16 模块的任何通道的波形或事件头 DPM 满时, 停止采集数据; =0: 仅当此 Pixie-16 模块的此通道的波形或事件头 DPM 已满时停止采集数据)

- **bit 5 - CCSRA_POLARITY**

- 选择此通道的输入信号极性 (=1: 反转输入信号的极性; =0: 不反转输入信号的极性)
 - 请注意, 在 Pixie-16 中, 信号处理需要正上升输入信号。因此, 如果输入信号具有负下降沿, 则应通过将此 CCSRA_POLARITY 位设置为 1 将其反相

- **bit 6 - CCSRA_VETOENA**

- 启用 (=1) 或禁用 (=0) 此通道的否决 (veto)。
 - 如果启用否决, 则该模块的否决信号 (请参见下面的位 20 CCSRA_MODVETOSEL) 或通道否决信号 (请参见下面的位 19 CCSRA_CHANVETOSEL) 将否决该通道的快速触发
 - 但是, 如果否决被禁用, 则即使存在任何一个否决信号, 该通道的快速触发也不会被任何一个否决信号否决

- **bit 7 - CCSRA_HISTOE**

- 启用 (=1) 或禁用 (=0) 板载 MCA 存储器中脉冲能量值的直方图。
 - 但是, 当前的 Pixie-16 固件始终会对板载 MCA 存储器中的脉冲能量值进行直方图绘制。
 - 因此, 此 CCSRA_HISTOE 目前基本上未使用

- **bit 8 - CCSRA_TRACEENA**

- 在列表模式下为此通道运行启用 (=1) 或禁用 (=0) 波形记录

- **bit 9 - CCSRA_QDCENA**

- 启用 (=1) 或禁用 (=0) 列表模式下为此通道运行 QDC 积分记录
 - 每个事件共有 8 个 QDC 积分

- **bit 10 - CCSRA_CFDMODE**

- 在列表模式下为此通道运行启用 (=1) 或禁用 (=0) CFD 触发
 - CFD 触发用于锁存事件到达时间或时间戳的亚采样时间

- **bit 11 - CCSRA_GLOBTRIG**
 - 启用 (=1) 或禁用 (=0) 此通道的模块验证触发的要求
 - 如果启用，则仅当模块验证触发与通道快速触发重叠时，才会记录该通道的事件
- **bit 12 - CCSRA_ESUMSENA**
 - 启用 (=1) 或禁用 (=0) 以列表模式为此通道运行原始能量总和及基线值的记录
 - 每个事件共有三个原始能量总和及一个基线值。
 - 请注意，基线值以 32 位 IEEE 浮点数 (IEEE 754) 的格式存储
- **bit 13 - CCSRA_CHANTRIG**
 - 启用 (=1) 或禁用 (=0) 此通道的通道验证触发的要求
 - 如果启用，则仅当通道验证触发与通道快速触发重叠时，才会记录该通道的事件
- **bit 14 - CCSRA_ENARELAY**
 - 通过输入继电器在此通道中的输入信号的两个衰减或增益之间切换 (=1: 关闭输入继电器，不会导致输入信号衰减; =0: 打开输入继电器，会导致输入信号衰减为 1/4)
- **bit 15/16 - CCSRA_PILEUPCTRL/CCSRA_INVERSEPILEUP**
 - 控制列表模式运行的正常堆积拒绝 (位 15) 和反向堆积拒绝 (位 16):
 - 1) 位 [16:15] = 00, 记录所有事件
 - 2) 位 [16:15] = 01, 仅记录非堆积事件，即拒绝堆积事件
 - 3) 位 [16:15] = 10, 记录堆积事件的所有内容，但即使启用了波形记录，也不会记录非堆积事件的波形，即仅记录事件头
 - 4) 位 [16:15] = 11, 仅记录堆积事件，即拒绝非堆积事件
 - 在所有情况下，如果事件堆积，则不会为该事件计算能量
- **bit 17 - CCSRA_ENAENERGYCUT**
 - 启用 (=1) 或禁用 (=0) “no traces for large pulses” 功能
 - 启用后，如果事件能量大于 DSP 参数 EnergyLow 中设置的值，则不会记录跟踪
- **bit 18 - CCSRA_GROUPTRIGSEL**
 - 选择通道快速触发，此位与 CCSRA_FTRIGSEL 位 (位 0) 一起使用：如果 CCSRA_FTRIGSEL=1，则此 CCSRA_GROUPTRIGSEL 位无效；否则，该位无效。如果 CCSRA_FTRIGSEL=0，则如果 CCSRA_GROUPTRIGSEL=1，则从系统 FPGA 中选择通道验证触发，如果 CCSRA_GROUPTRIGSEL=0，则选择该通道的本地快速触发
- **bit 19 - CCSRA_CHANVETOSEL**
 - 通道否决信号选择 (=1: 来自系统 FPGA 的通道验证触发; =0: 来自 Pixie-16 前面板通道门 LVDS 连接器的通道门输入)
- **bit 20 - CCSRA_MODVETOSEL**
 - 模块否决信号选择 (=1: 来自系统 FPGA 的模块验证触发; =0: 来自 Pixie-16 前面板模块门 LVDS 连接器的模块门输入)
- **bit 21 - CCSRA_EXTTSENA**
 - 在此通道的列表模式运行期间，启用 (=1) 或禁用 (=0) 事件头中的 48 位外部时钟时间戳的记录

7.4.3 Logic Set

控制界面

Logic Trigger

ch #	ExtTrigStr[us]	ExtDelayLen[us]	ExtTrigoutDel[us]	VetoStr[us]	ChafTrigStr[us]	astTriBalLen[us]	Left[0xFFFF]	Itself[0xFFFF]	Right[0xFFFF]	ItselfCoin[0-7]	RightCoin[0-7]	LeftCoin[0-7]	MultiThre[0-31]	Sel Coin/Multi	Sel ChValidTrig
0	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
1	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
2	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
3	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
4	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
5	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
6	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
7	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
8	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
9	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
10	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
11	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
12	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
13	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
14	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0
15	1.50	0.20	0.00	0.30	0.00	0.10	0000	0000	0000	0	0	0	0	0	0

Status: Ready
Module #: 0 ▾ Load Apply Cancel

Select channel #: 0 ▾ Copy

FastTrigBackplaneLeft[0-FFFF] 0000 FastTrigBackplaneRight[0-FFFF] 0000 Front Panel Monitor Disable Group 000 Ch 00 TestSig 00

Channel Validation Trigger(System FPGA group trigger): (ExtFastTrigGate => 'FT' AND Ext_Fast_Trig_In(module fast trigger)) [Choose 'System FPGA' or 'Front panel channel GATE' in CSRA]

Ch 0-3:	ExtFastTrig[0]	GroupTri 0_0	LocalModule Ch	Ch 00	RightModule Ch	Ch 00	LeftModule Ch	Ch 00
Ch 4-7:	ExtFastTrig[1]	GroupTri 1_0	LocalModule Ch	Ch 00	RightModule Ch	Ch 00	LeftModule Ch	Ch 00
Ch 8-11:	ExtFastTrig[2]	GroupTri 2_0	LocalModule Ch	Ch 00	RightModule Ch	Ch 00	LeftModule Ch	Ch 00
Ch 12-15:	ExtFastTrig[3]	GroupTri 3_0	LocalModule Ch	Ch 00	RightModule Ch	Ch 00	LeftModule Ch	Ch 00

Module Fast Trigger:

Source	current module (Ext_FastTrig_In)
Current Module	Ext_FastTrig_Sel(front panel TTL)
Int_FastTrig_Sgl	Ch 00

Module Validation Trigger(System FPGA): [Choose 'System FPGA' or 'Front panel module GATE' in CSRA]

source	current module (Ext_ValidTrig_In)
Current Module	Ext_ValidTrig_Sel(front panel TTL)
Int_ValidTrig_Sgl	Ch 00

Module Fast/Validation Trigger ChanTrig_Sel:

Channel Trigger:	Ch 00
------------------	-------

Module/Channel Veto: [Choose 'front panel module/channel GATE' or 'module/channel validation trigger' in CSRA, channel veto need choose enable/disable]

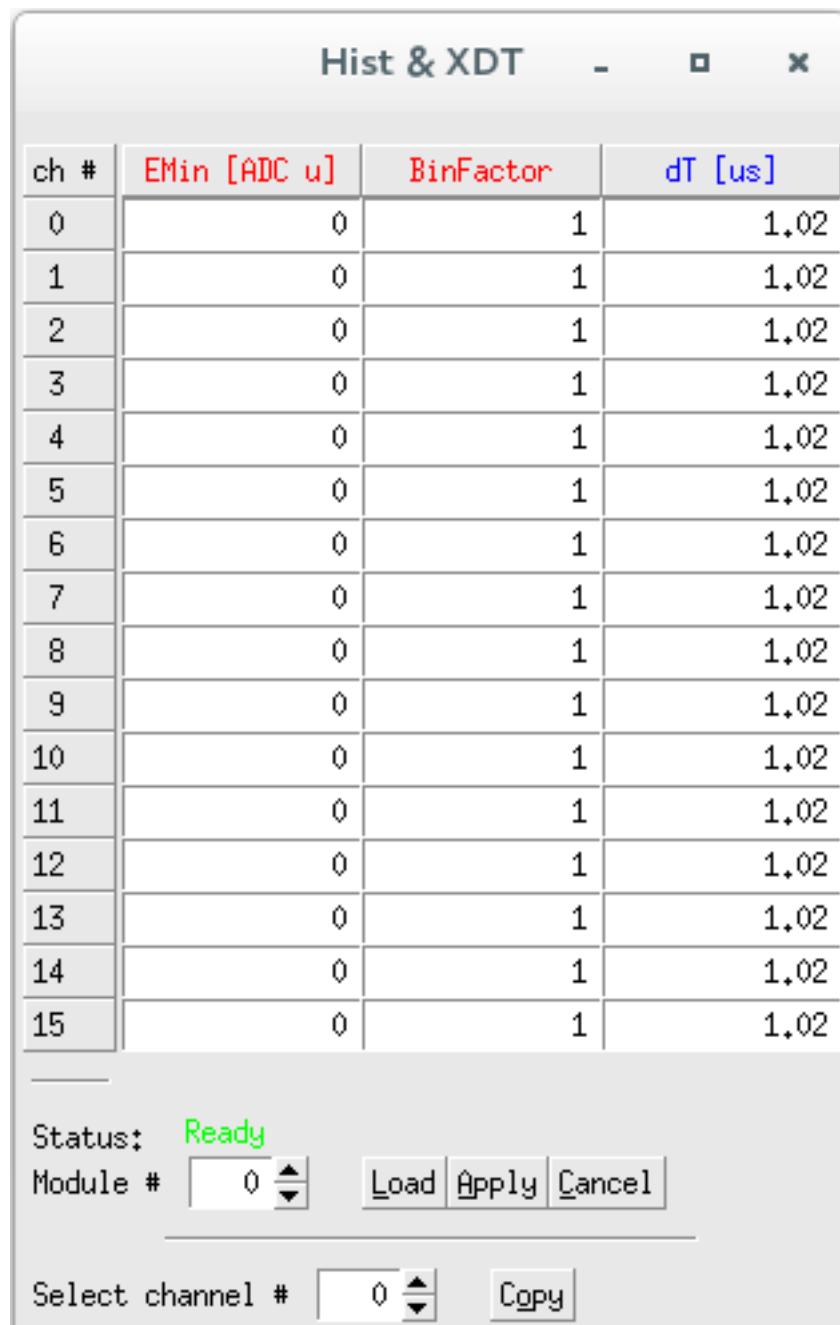
			Range[us]
1	Fast trigger stretch length	FT门宽	0.01-40.95
2	Fast trigger delay length	FT延迟	0-5.11
3	Extern delay	采集信号延迟	0-5.11
4	External trigger stretch length		0.01-40.95
5	Channel trigger stretch length		0.01-40.95
6	Veto stretch length		0.01-40.95

7.5 Debug

本下拉栏中调节内容为监视波形噪声水平、基线分布等。

7.5.1 Hist & XDT

控制界面



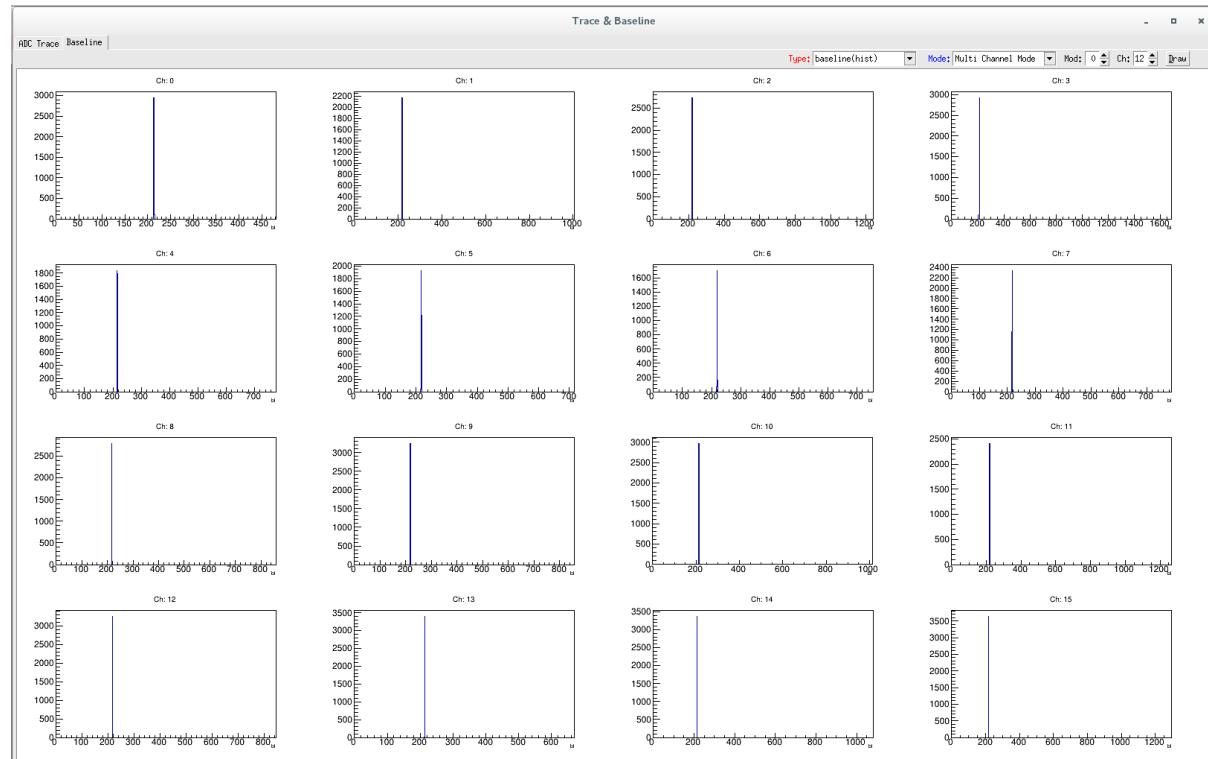
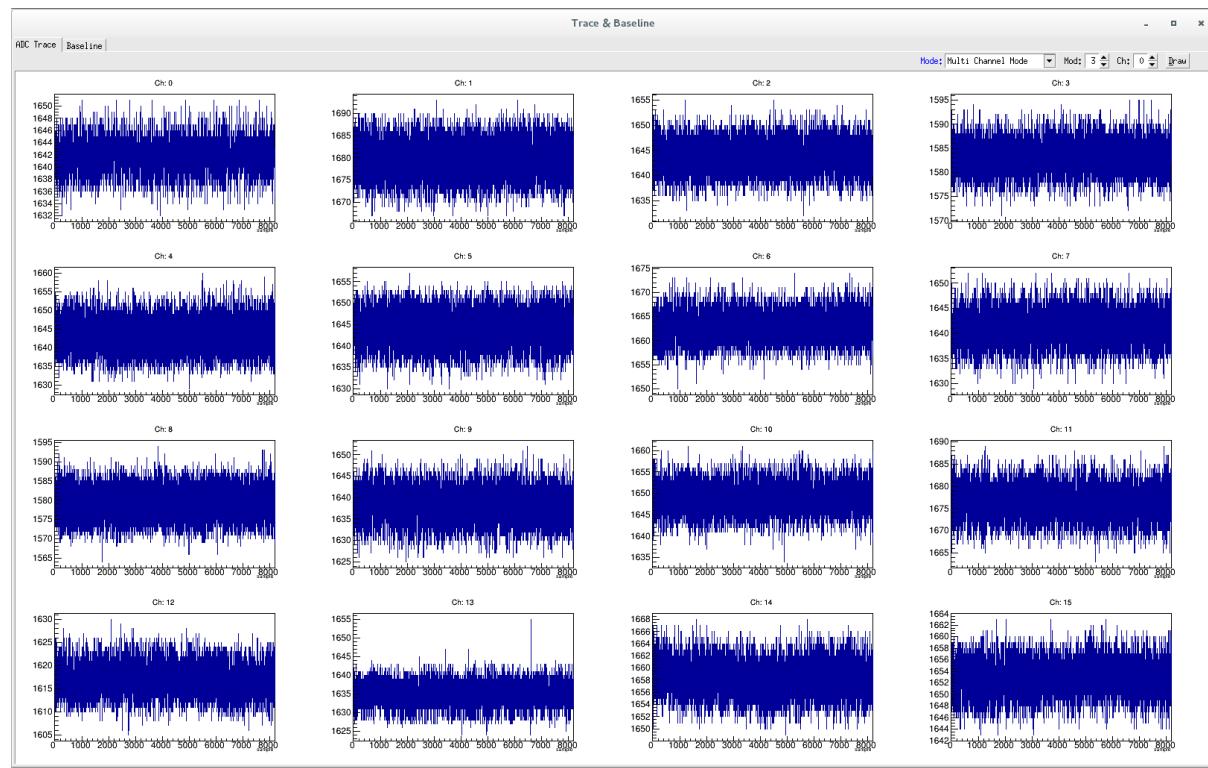
Binning factor 控制能谱中 MCA bin 的数量。能量计算为 16 位数字，原则上允许 64K MCA bin。

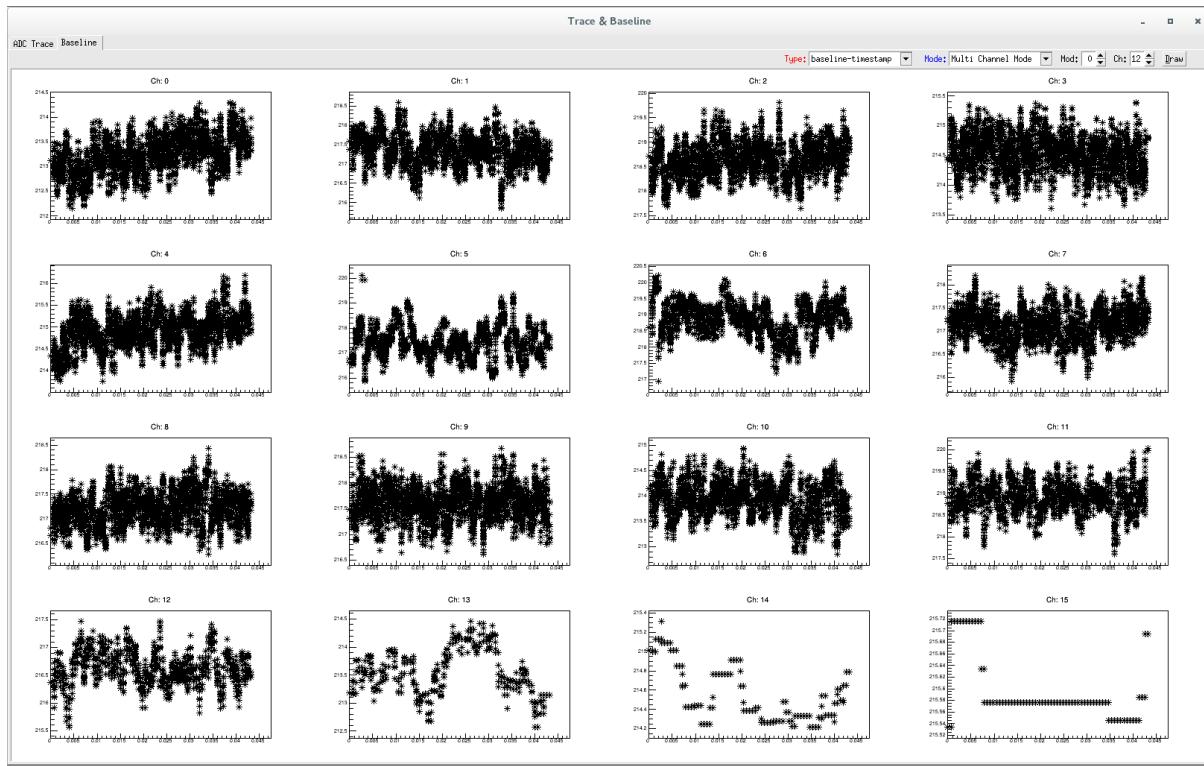
但是，每个通道的能谱内存限制在 32K 的 bin 数，因此在构建柱状图之前，计算的能量值除以系数 $2^{binning\ factor}$ 。Binning factor 通常设置为 1，但对于低计数率和宽峰值，将其设置为较大的值可能会有用，以获得具有较少 bin 数但每个 bin 中更多的计数。

E_{min} 是为将来的函数预留的，用于在能谱填充之前从计算的能量值中减去一个常数“最小能量”，从而基本上切断能谱的低地址部分。

7.5.2 Trace & Baseline

控制界面



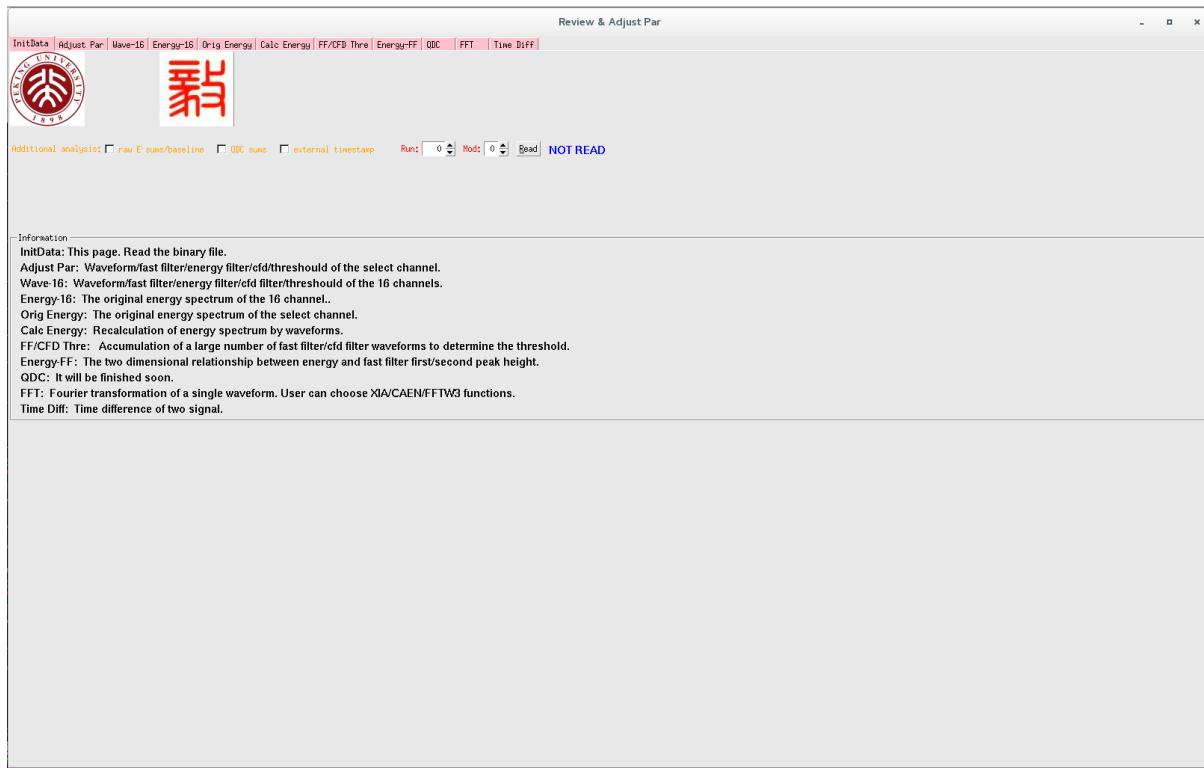


◦◦ TODO ◦◦

7.6 Offline

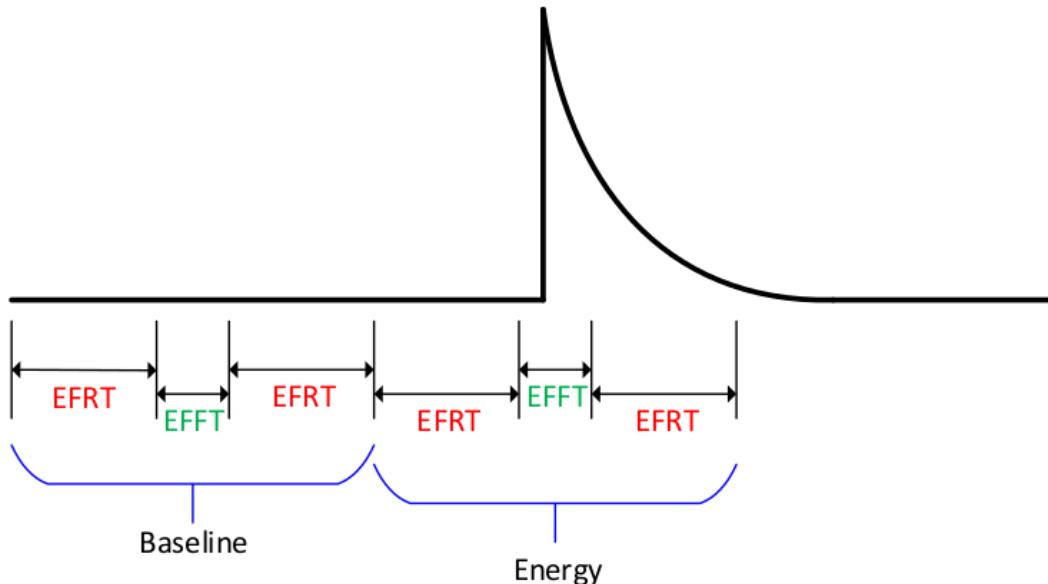
本下拉栏中为离线参数优化调节。

7.6.1 InitData



- Run 选择要读取的文件运行编号, Mod 选择要读取第几个采集卡, 按钮 Read 将文件主要信息(道址、能量、波形位置等)载入内存。
- Additional analysis: 三个选项中, 选择表示读取该文件数据到内存中时包括该信息。只有读取了该数据, 才能启用一些分析方法。但是前提是数据采集时候需要记录该信息。
- InitData: This page. Read the binary file.
- Adjust Par: Waveform/fast filter/energy filter/cfd/threshold of the select channel.
- Wave-16: Waveform/fast filter/energy filter/cfd filter/threshold of the 16 channels.
- Energy-16: The original energy spectrum of the 16 channel..
- Orig Energy: The original energy spectrum of the select channel.
- Calc Energy: Recalculation of energy spectrum by waveforms.
- FF/CFD Thre: Accumulation of a large number of fast filter/cfd filter waveforms to determine the threshold.
- Energy-FF: The two dimensional relationship between energy and fast filter first/second peak height.
- QDC: It will be finished soon.
- FFT: Fourier transformation of a single waveform. User can choose XIA/CAEN/FFTW3 functions.
- Time Diff: Time difference of two signal.

7.6.2 Adjust Par



$$\text{Net Energy} = \text{Energy} - \text{Baseline}$$

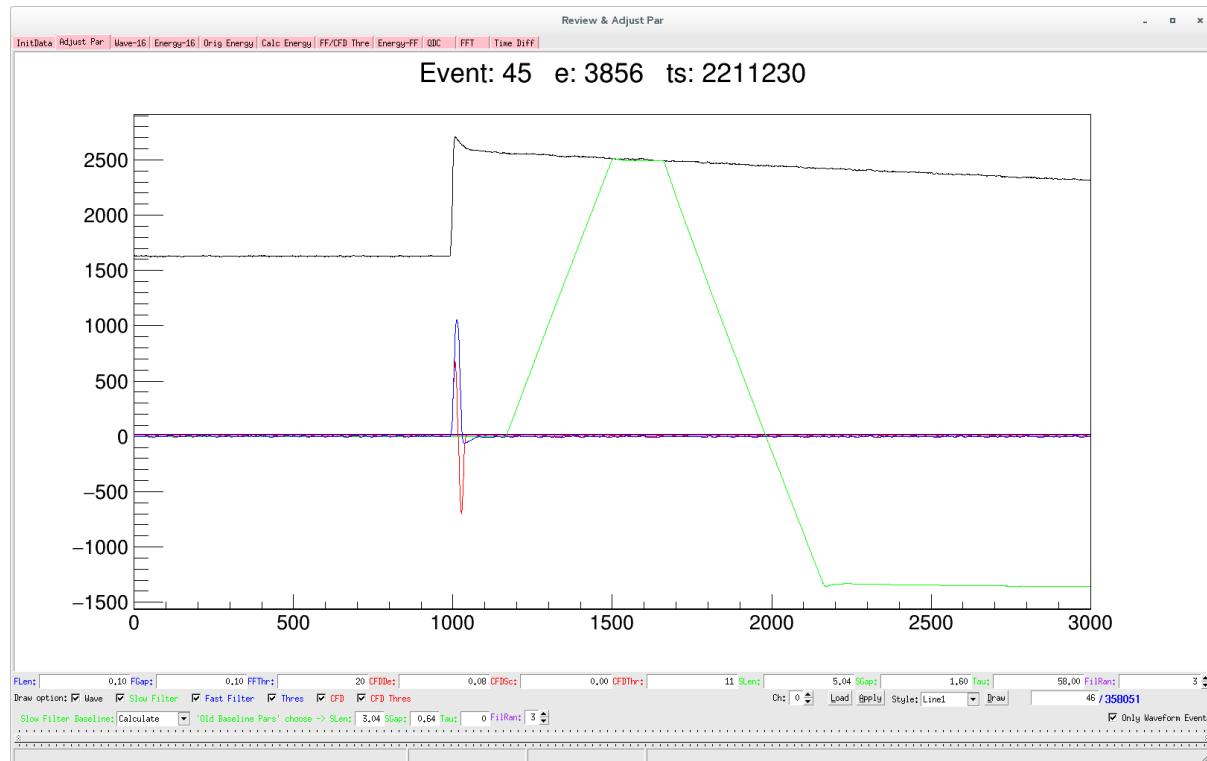
要通过采集的波形离线计算 fast filter、slow filter cfd 曲线，对采集的波形有以下要求。如上图中，计算的能量是算法的能量与算法的基线的差，要得到正确的梯形，那么前提是前面有足够的点来计算基线。

In the figure, EFRT stands for Energy Filter Rise Time and EFFT stands for Energy Filter Flat Top.

To compute energy filter response offline, the ideal settings are:

- Total trace length > $2 \times (2 \times \text{EFRT} + \text{EFFT})$
- Pre-trigger trace (Trace-delay) length > $(3 \times \text{EFRT} + \text{EFFT})$

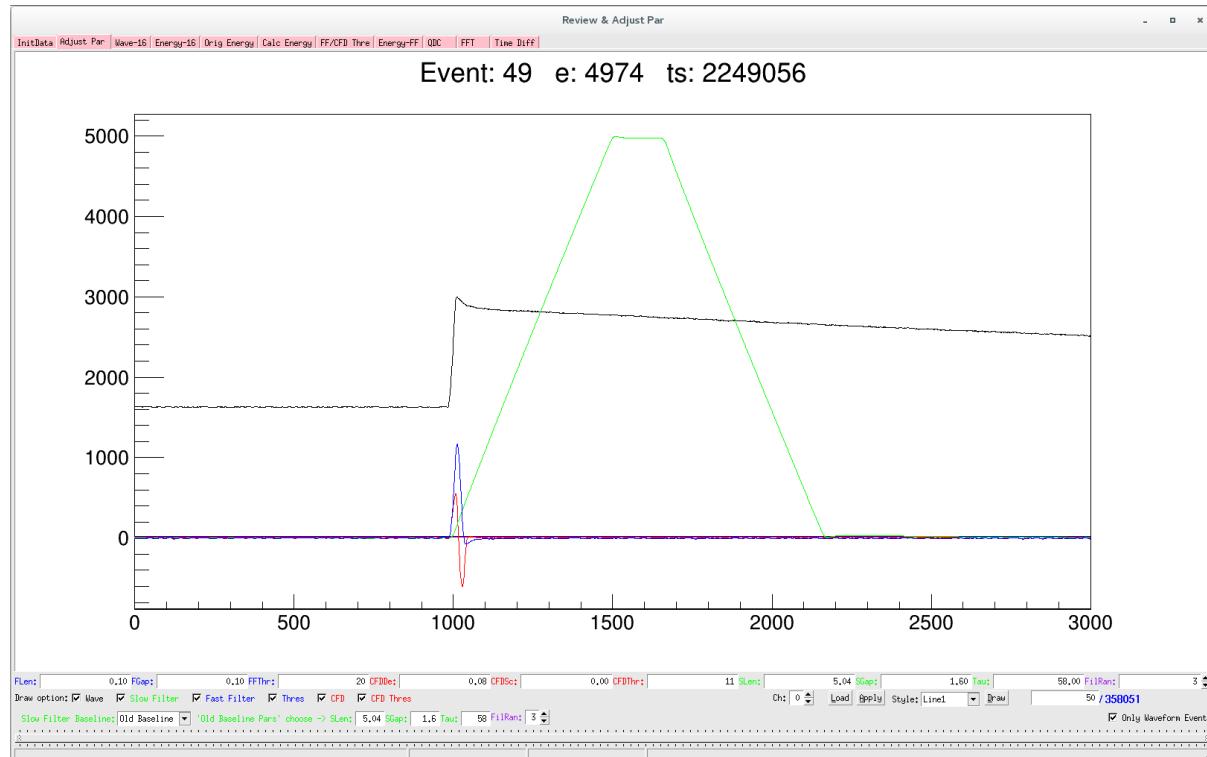
当然，这只是计算梯形的一个方法，如果我们记录了每个事件的能量梯形的基线，并且采用 pre-trigger 部分点的平均值作为波形左侧的无限延伸，那么就不受 Pre-trigger trace length > $(3 \times \text{EFRT} + \text{EFFT})$ 条件的限制了。下面的页面中，当采用 Old Baseline 方法来计算能量梯形时，有个前提是 pre-trigger trace length 至少需要有 200 个点，因为波形左侧延伸采用前 200 个点来平均。



当采集的波形 pre-trigger trace length $> 3 \times \text{EFRT} + \text{EFFT}$ 时，pre-trigger trace 提供足够多的点来计算基线，SF BL 算法可选择 Calculate，否则需要选择 Old Baseline 算法。选择 Old Baseline 算法的前提是记录数据的时候，选择开启记录梯形的 baseline，并且 InitData 页面的 raw E sums/baseline 选项开启。当选择 Old Baseline 算法时，之后的四个选项参数生效，该四个参数为该数据采集时候所用的能量梯形的参数。

上图中绿色曲线为典型的不满足 pre-trigger trace length $> 3 \times \text{EFRT} + \text{EFFT}$ 时，采用的 Calculate 算法造成的结果。图中显示 pre-trigger trace length 为 10 us，EFRT 为 5.04 us，EFFT 为 1.60 us。

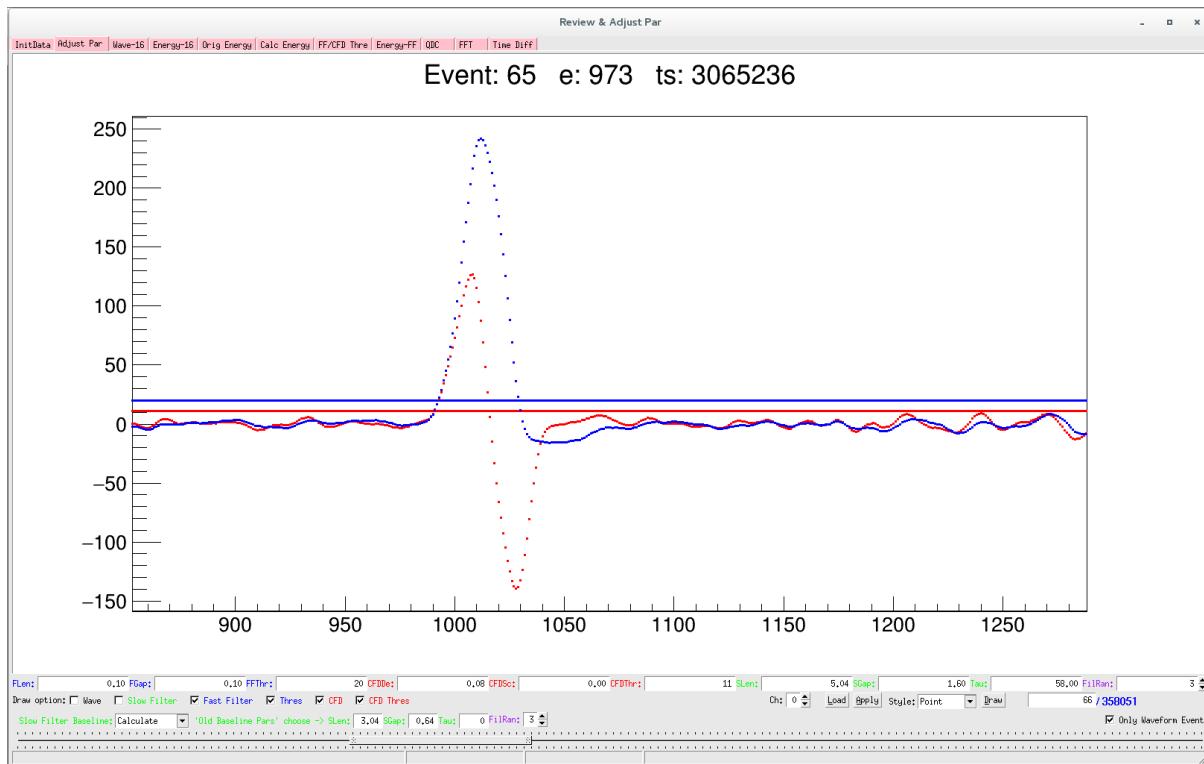
此时，应该采用下图所示的 Old Baseline 算法。



用户可选择查看波形的通道，按钮 Load 可读取并显示当前的参数设置情况，当修改以上的参数时候，需

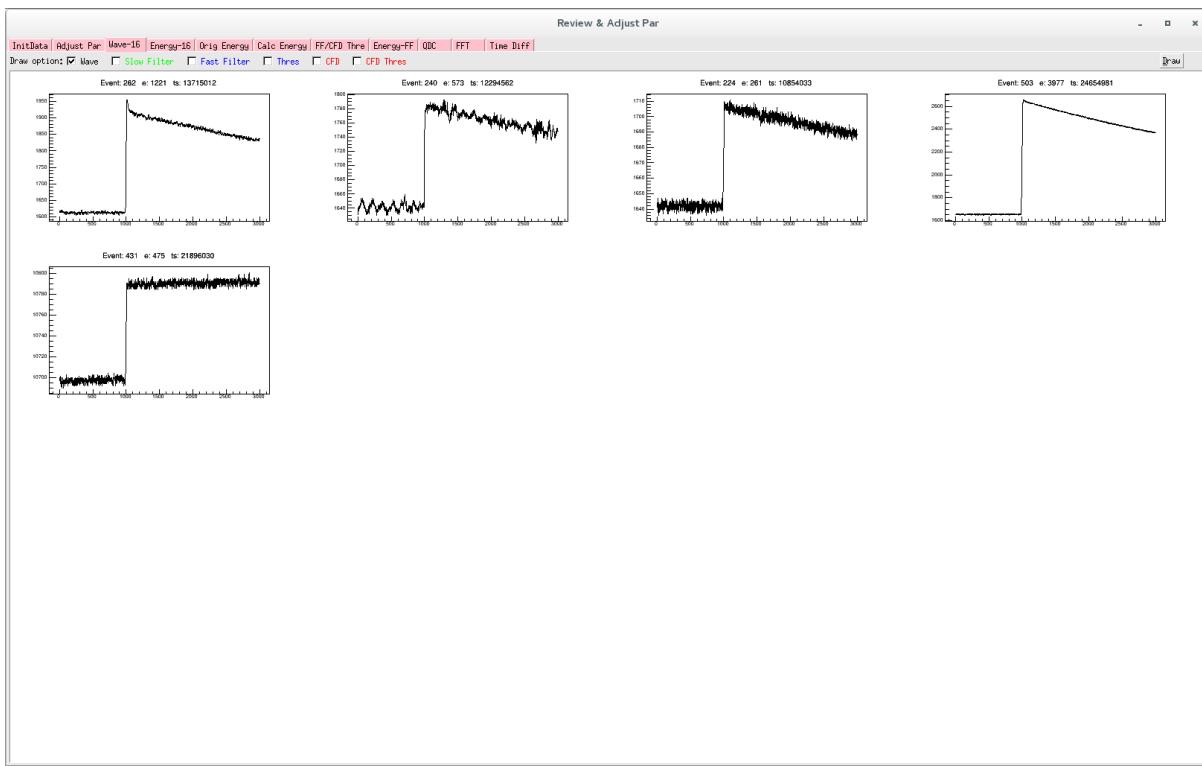
要按 Apply 按钮使之生效。按钮 Draw 用来显示下一个该通道的事件波形。

用户可选择同时显示 Wave / Slow Filter/ Fast filter / Thres / CFD / CFD Thres 中的多个波形。或者选择曲线的绘画样式。



上图展示了显示 fast filter、Thres、CFD、CFD Thres 四个波形，图样采用点显示方式。最低端的水平条两端可以拖动，用户可拉动来控制波形横坐标的显示范围，如图中显示 800 - 1300 的点。该情况下，点击 Draw 按钮，将会保持该指定的坐标范围。

7.6.3 Wave-16

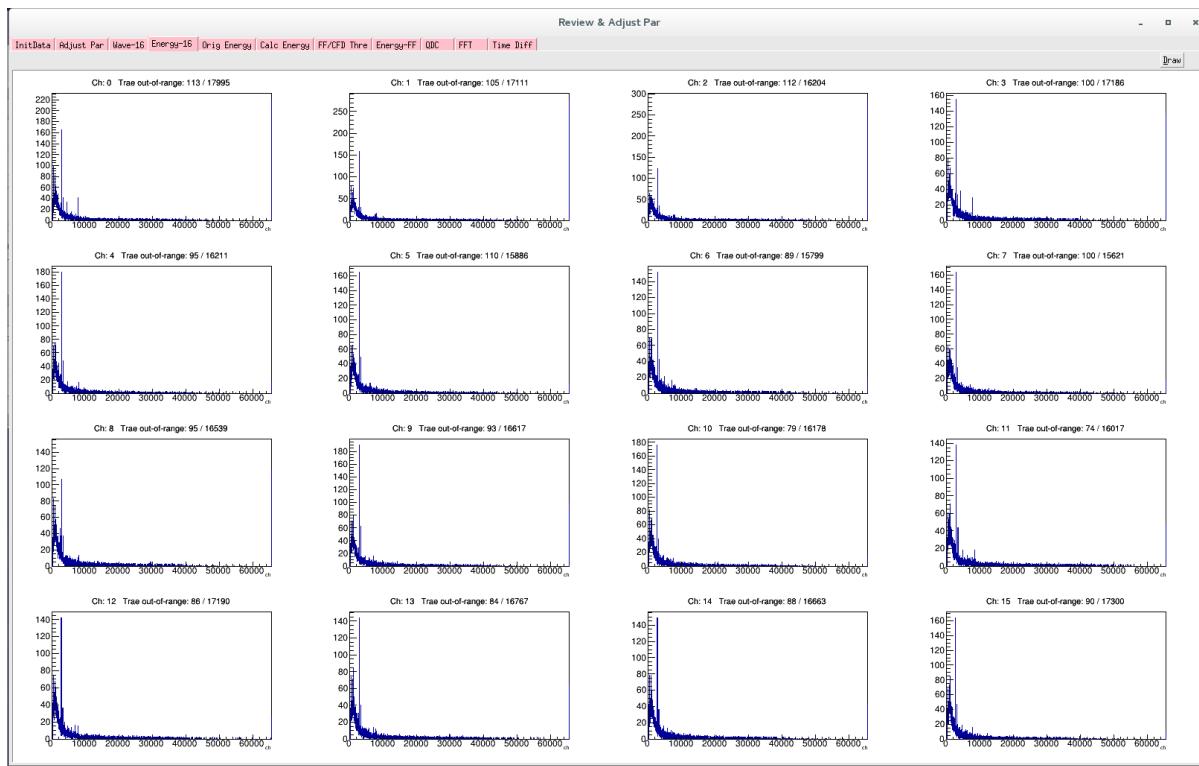


该页面用于同时查看 16 通道的原始波形、filter 波形，阈值等。用户可选择同时显示 Wave / Slow Filter/ Fast filter / Thres / CFD / CFD Thres 中的多个波形。

用户可通过该页面，快速查看该采集卡所有通道的波形是否正常，参数设置是否合理。点击按钮 Draw 一次，则显示所有通道下一个波形。

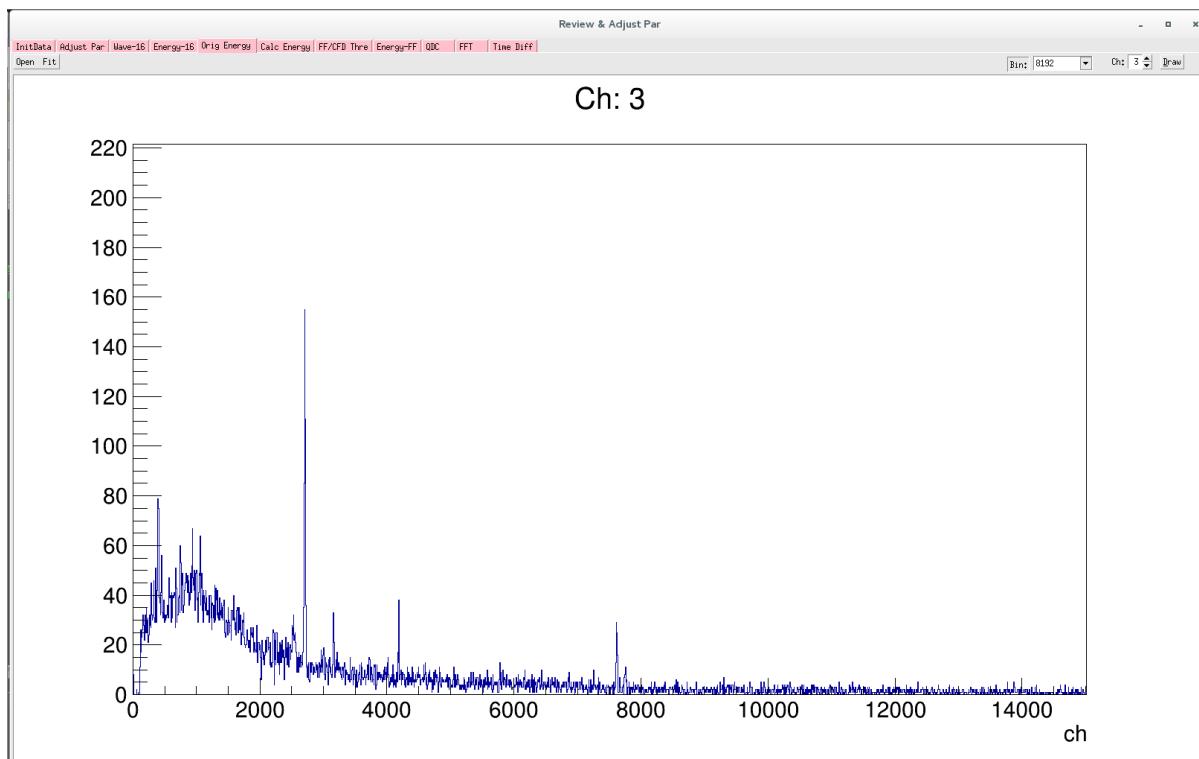
需要注意的一点，本页面的 Slow Filter 波形需要在采集的波形 *pre-trigger trace* 长度大于 $3 \times EFRT + EFFT$ 时才是正确的。

7.6.4 Energy-16

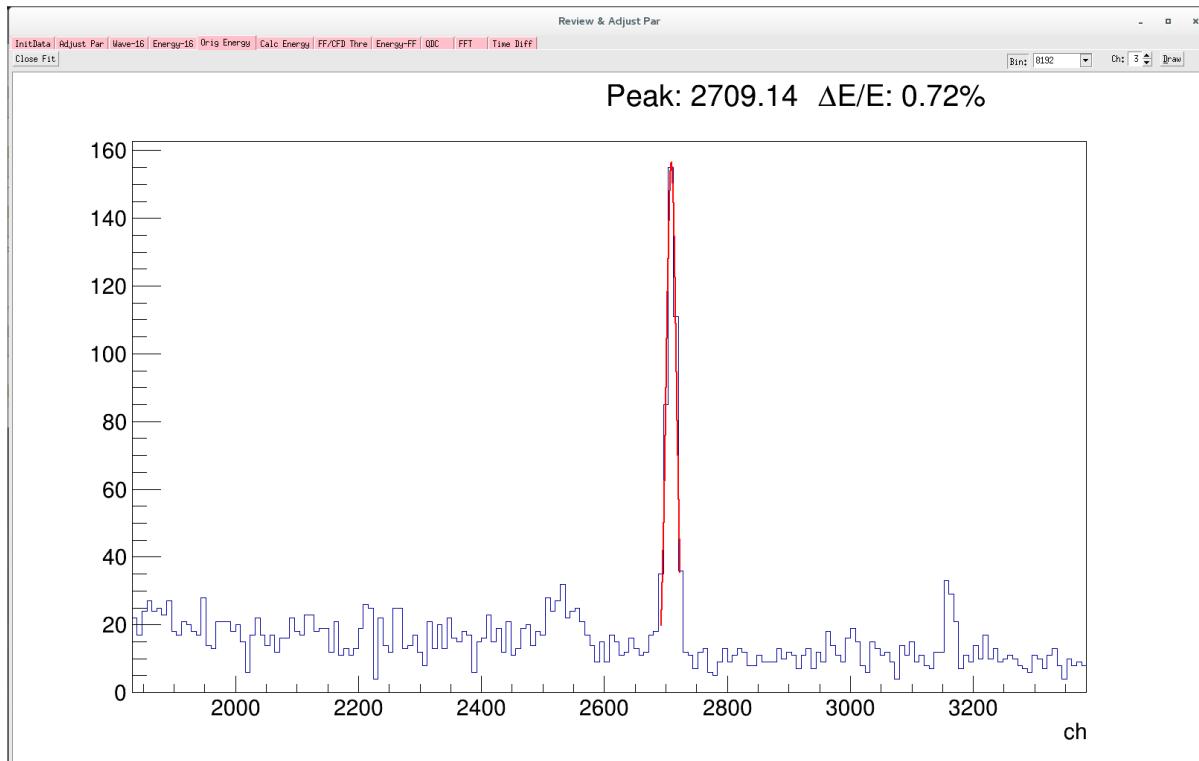


该界面用于同时查看 16 通道的一维能谱。点击右上角的按钮 Draw 即可。

7.6.5 Orig Energy

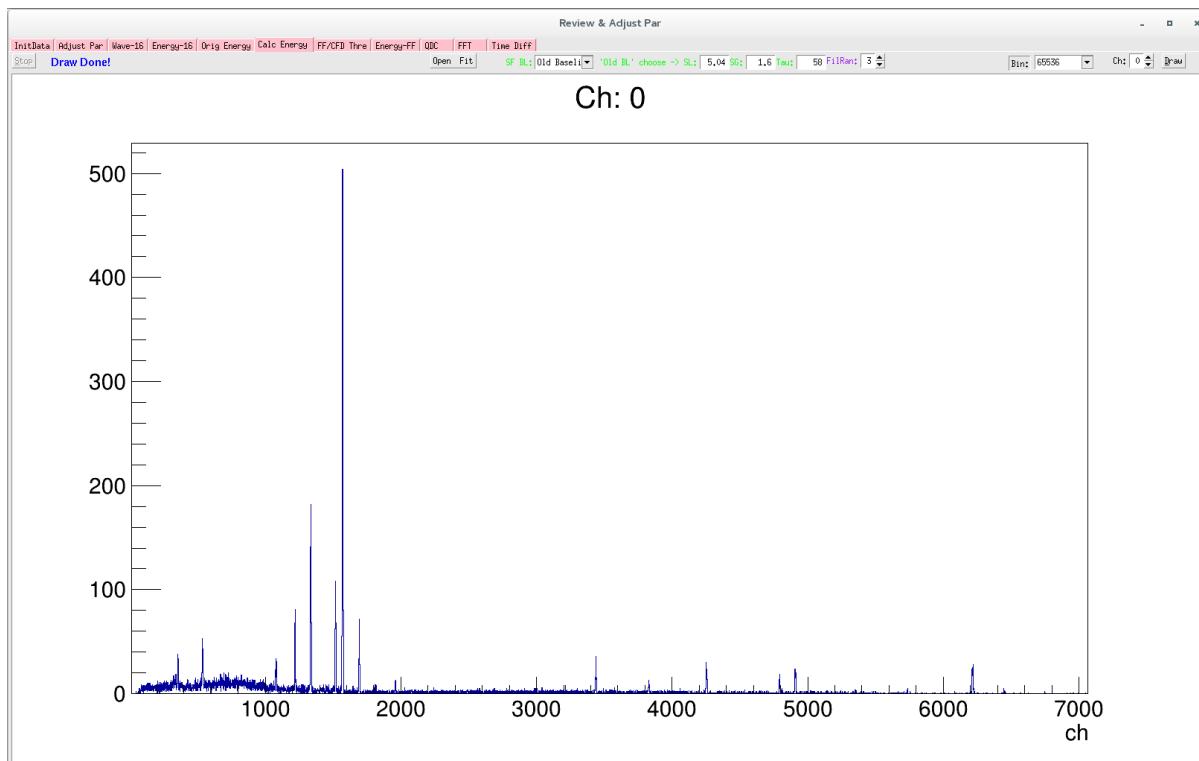


该页面用来快速查看某通道的能谱。用户选择能谱的分 Bin 数，该数值表示将 0 - 65536 道分成多少份。选择查看通道。然后按 Draw 按钮即可。



左上角的 Open Fit 按钮用来快速高斯拟合看能量分辨。点击按钮，开启拟合模式，再次点击按钮则关闭该功能。将鼠标移动到直方图的蓝线上，鼠标十字将会变成三角箭头。三角箭头的鼠标点击直方图中的两个位置，两点所在区间即为拟合区间，则可查看能量分辨。

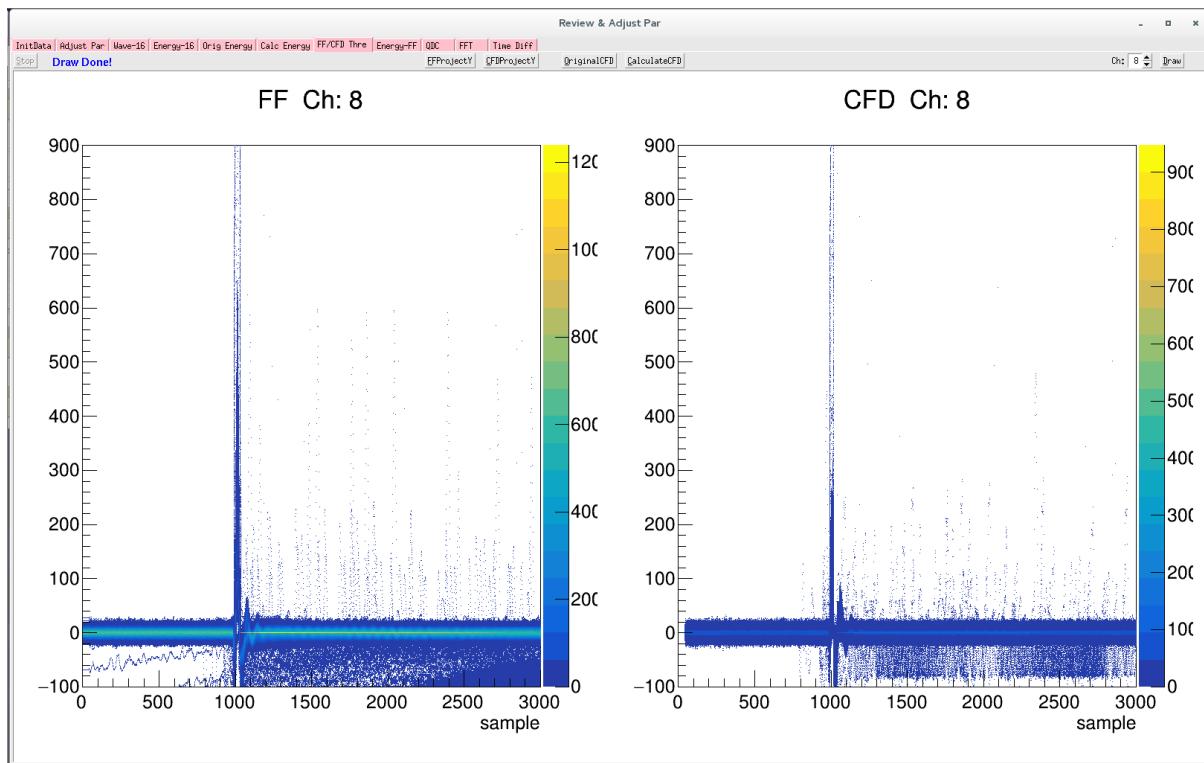
7.6.6 Calc Energy



该页面利用采集的波形重新计算能量。同 Adjust Par 页面一样，SF BL 算法可选择 Calculate 算法或者 Old Baseline 算法。

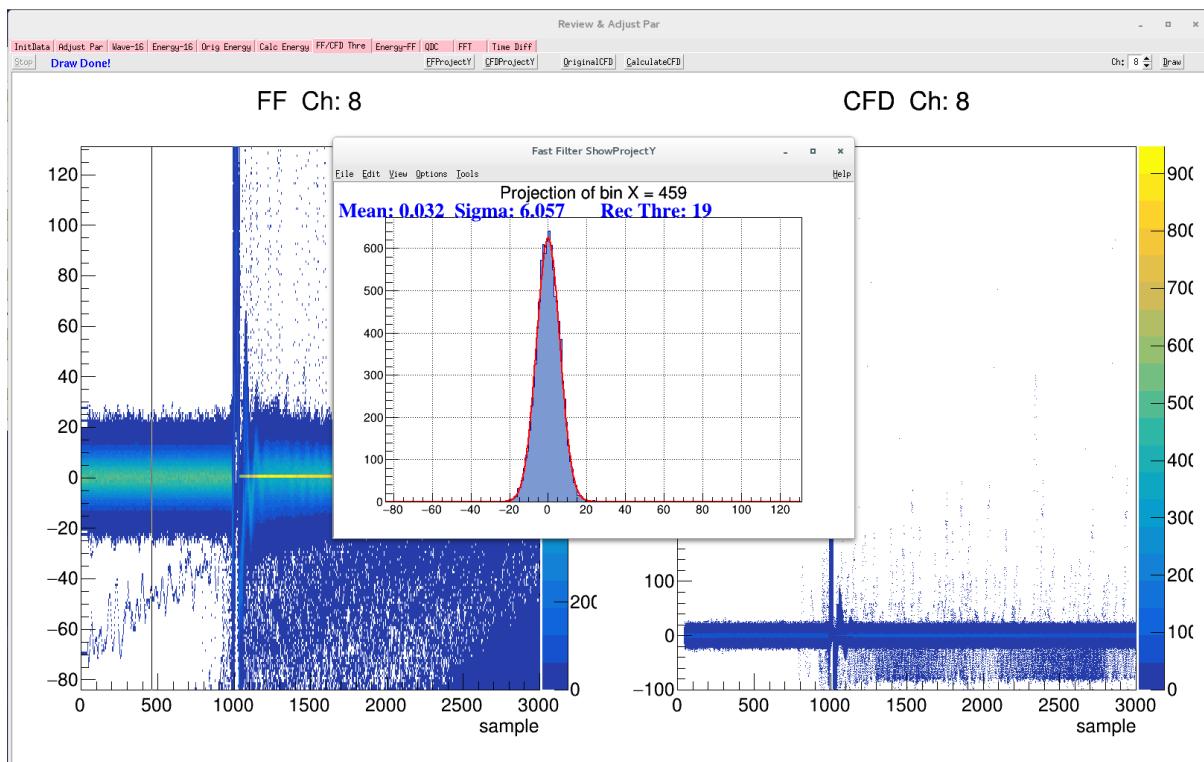
计算能量采用的 fast filter、energy filter 参数采用采集卡的设置参数，用户需要选择能量 0-65536 分成多少个 bin，可选择 1024/2048/4096/8192/16384/32768/65536，选择计算的通道，然后按按钮 Draw 即开始计算，左上角将会显示计算的进度，也可以按按钮 Stop 提前终止计算。当计算终止时，画板上将显示能谱。

7.6.7 FF/CFD Thre

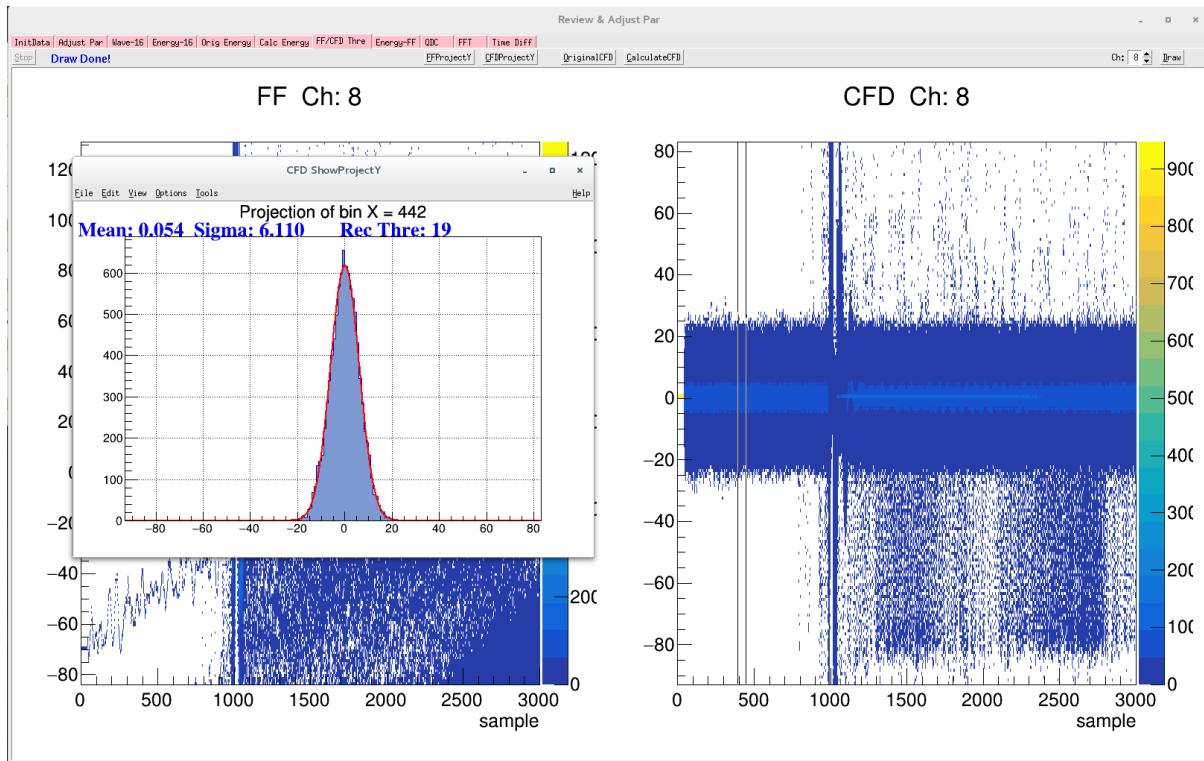


该界面用于 fast filter 波形、cfd filter 波形的累加。用户选择查看通道，然后按 Draw 按钮则开始进入计算，左上角可时时监视进度，也可按 Stop 按钮提前终止计算。计算结束得到如上图所示。

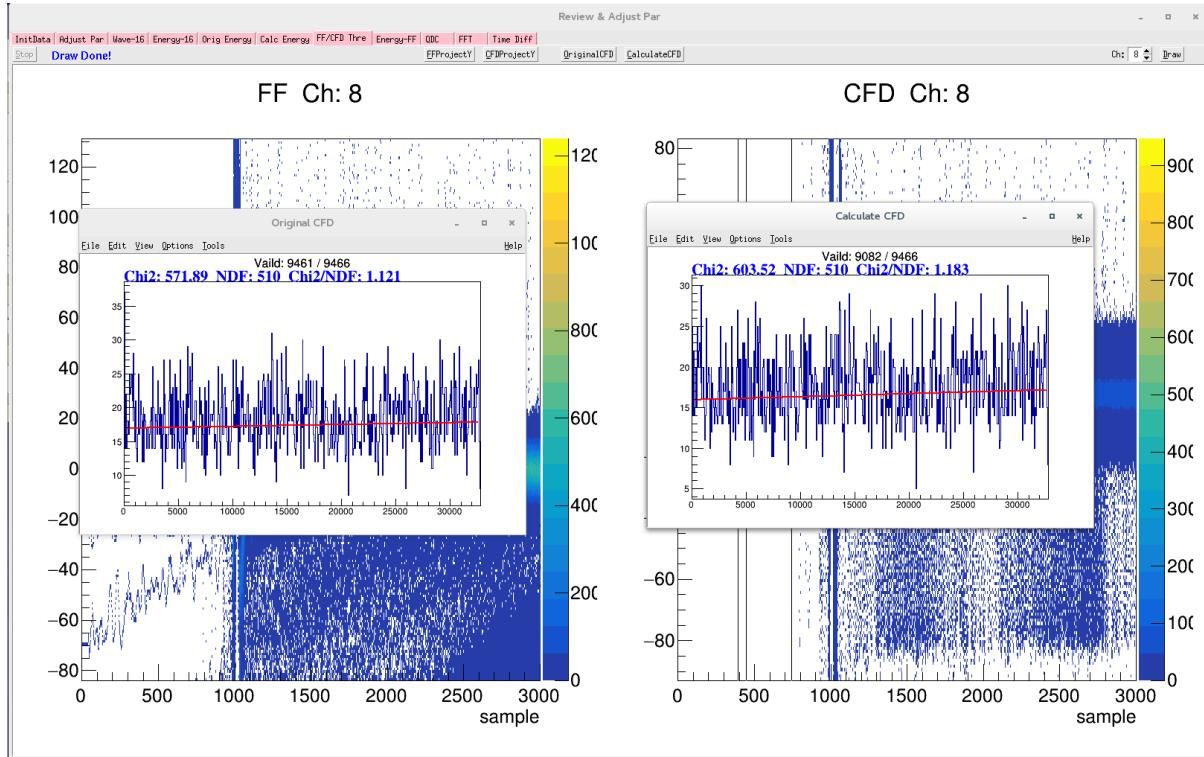
上方按钮 FFProjectY、CFDProjectY、OriginalCFD、CalculateCFD 分别可弹出子画板。



点击按钮 FFProjectY，则开启查看 fast filter 投影图，再次点击则关闭该功能。开启功能时，将鼠标放在二维图上，左右移动鼠标，Fast Filter ShowProjectY 子画板则显示鼠标指向的该位置的投影分布。触发前的该分布，也表征噪声的水平。

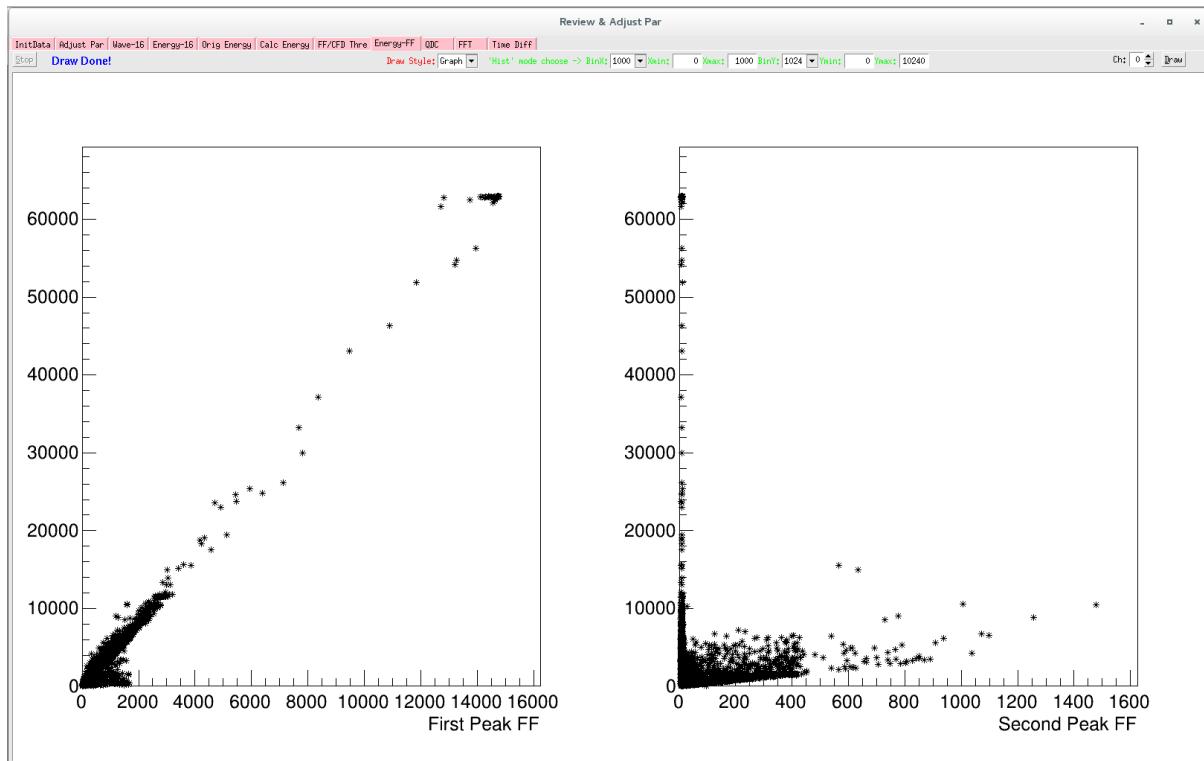


同理，按钮 CFDProjectY 功能如上图所示。



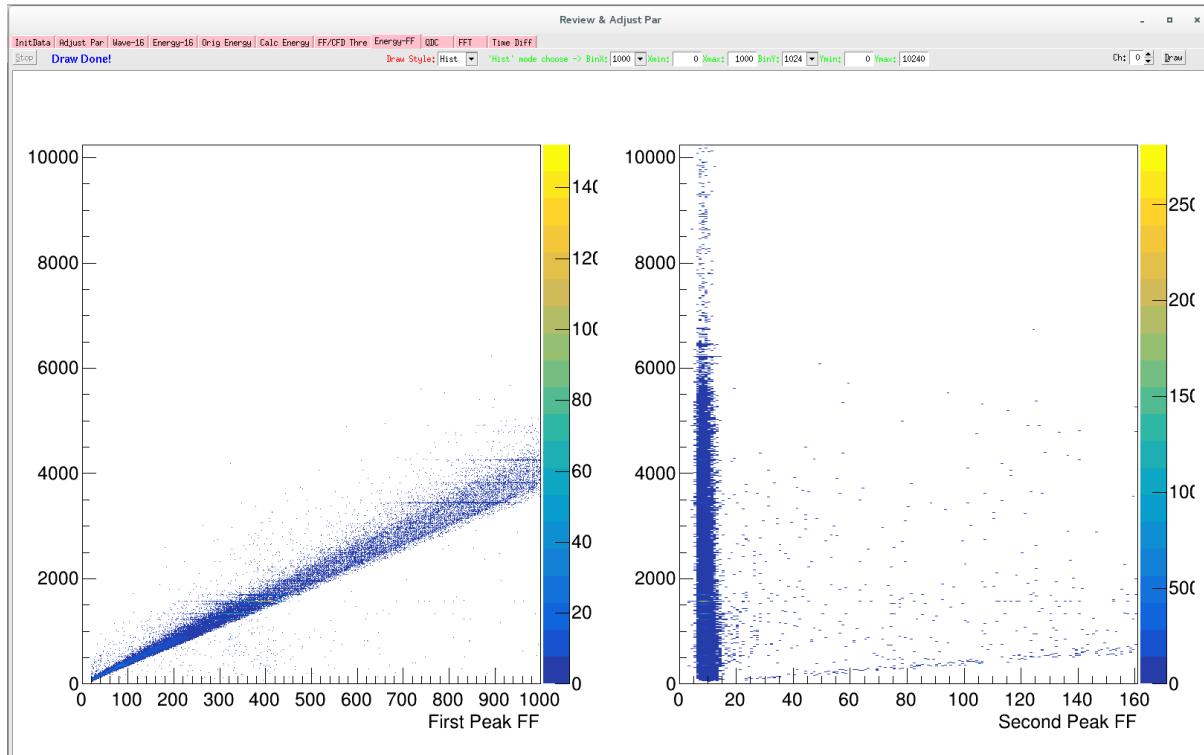
点击按钮 OriginalCFD，则展示左图中原始数据中 CFD 数值的分布。点击按钮 CalculateCFD，则展示右图中通过离线波形计算的结果，计算所用参数为当前的参数。对于一个合适的 CFD 参数设置，该 CFD 分布该是平均分布的。

7.6.8 Energy-FF



该界面是能量与 fast filter 峰高的二维关联图。用于确定合适的阈值。左图是能量与 fast filter 的二维关联，它们应该有个较好的线性关系，右图为能量与 fast filer 中抛除梯形部分剩余中最大值的二维关联，抛除梯形部分剩余分布的最大值表征噪声水平，能量跟该值应该是没有关联的。

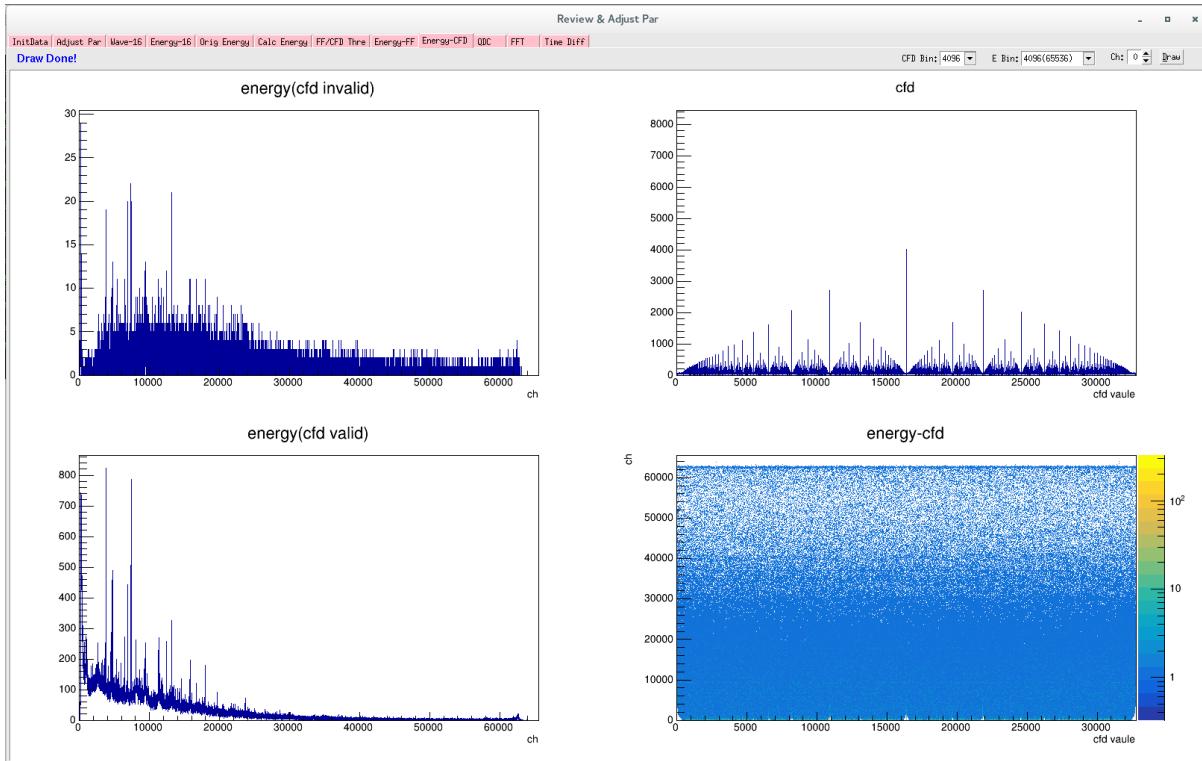
首先 Draw Style 选择 Graph，即二维散点图模式。选择查看通道，然后按 Draw 按钮则开始进入计算，左上角可时时监视进度，也可按 Stop 按钮提前终止计算。计算结束得到如上图所示。



二维散点图并不能很直观显示展示数据点的密度分布，因此 Draw Style 选择 Hist 模式，选择 X、Y 轴的

分 bin 数即范围，然后同样按 Draw 按钮开始计算。结果如上图所示，右图反映了噪声的水平。

7.6.9 Energy-CFD



- 左上图为 cfd 无效时的能谱。
- 左下图为 cfd 有效时的能谱。
- 右上图为 CFD 谱。
- 右下图为能量与 CFD 的二维关联图。

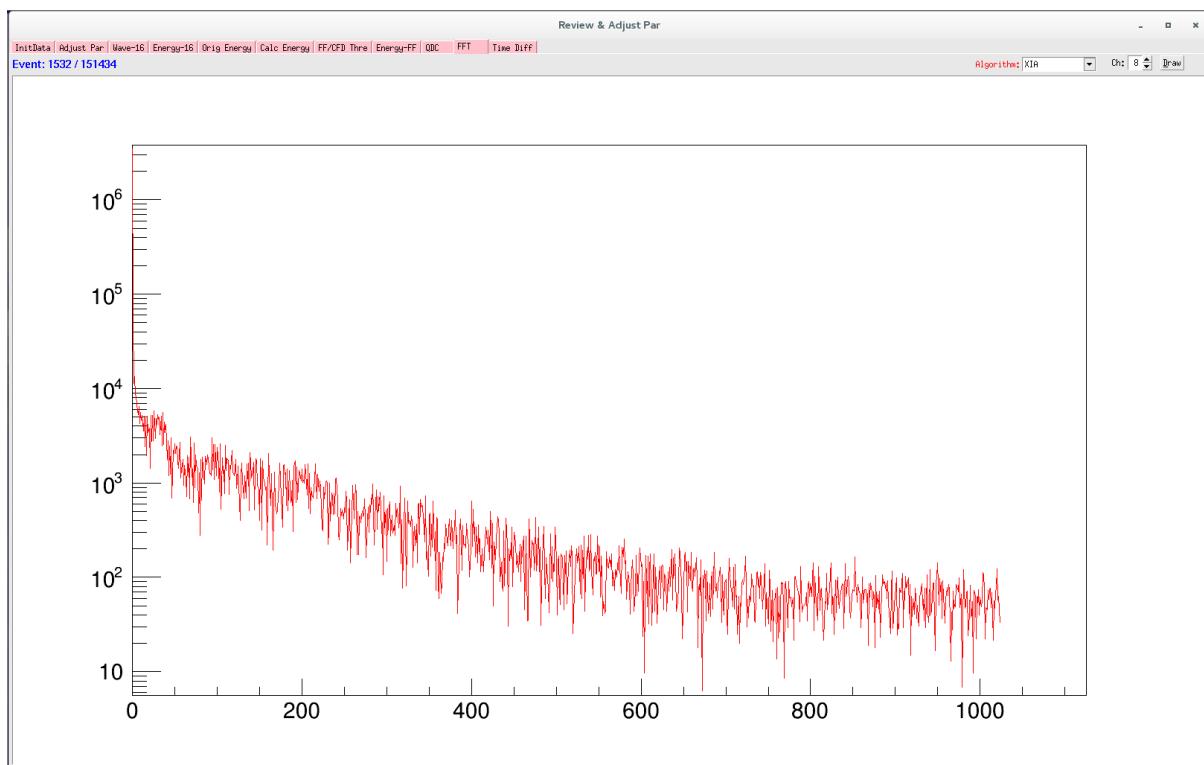
首先选择能量与 CFD 二维关联图中 bin 数。其中 CFD 分 bin 可选择 4096, 2084, 1024；能量可选择 bin 数与道址范围。之后选择查看通道，然后按 Draw 按钮开始进入计算。

7.6.10 QDC

to do not completed

QDC TODO 功能未完成

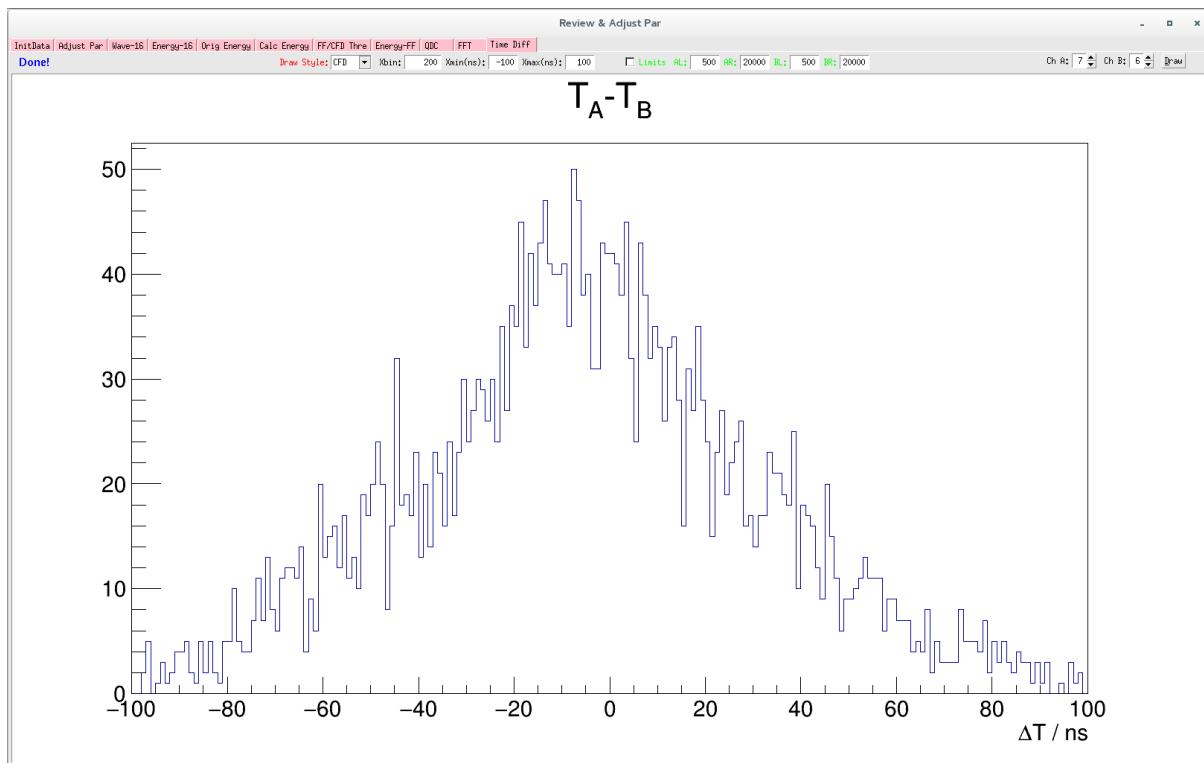
7.6.11 FFT



该界面用于快速查看波形的傅立叶变换。用户可以选择不同的算法，例如 XIA、fftw3、CAEN(HANNING)、CAEN(HAMMING)、CAEN(BLACKMAN)、CAEN(RECT)。选择查看通道。然后按 Draw 按钮即可，每点击一次该按钮，则显示下一个结果。

the ADC trace display also includes the option to view a FFT of the acquired trace. This is useful to diagnose noise contributions.

7.6.12 Time Diff



该界面用于快速查看两路信号的时间分辨。用户可以选择查看 CFD 算法过零点的时间差或者 fast filter 过阈值的时间差。Xbin 表示横坐标分 bin 数, Xmin 表示横坐标的最小值, Xmax 表示横坐标的最大值。通过 Ch A、Ch B 来选择想要查看的两个通道。然后按 Draw 按钮即可。

选项 Limits 选择则开启能量范围约束。选择该选项后,之后的四个参数 AL、AR、BL、BR 才生效,其分别表示 Ch A/B 能量道址的左右范围,只有能量落在这个区间的事件才填充到直方图中。用户可通过 Orig Energy 页面来选择合适的能量道址区间。

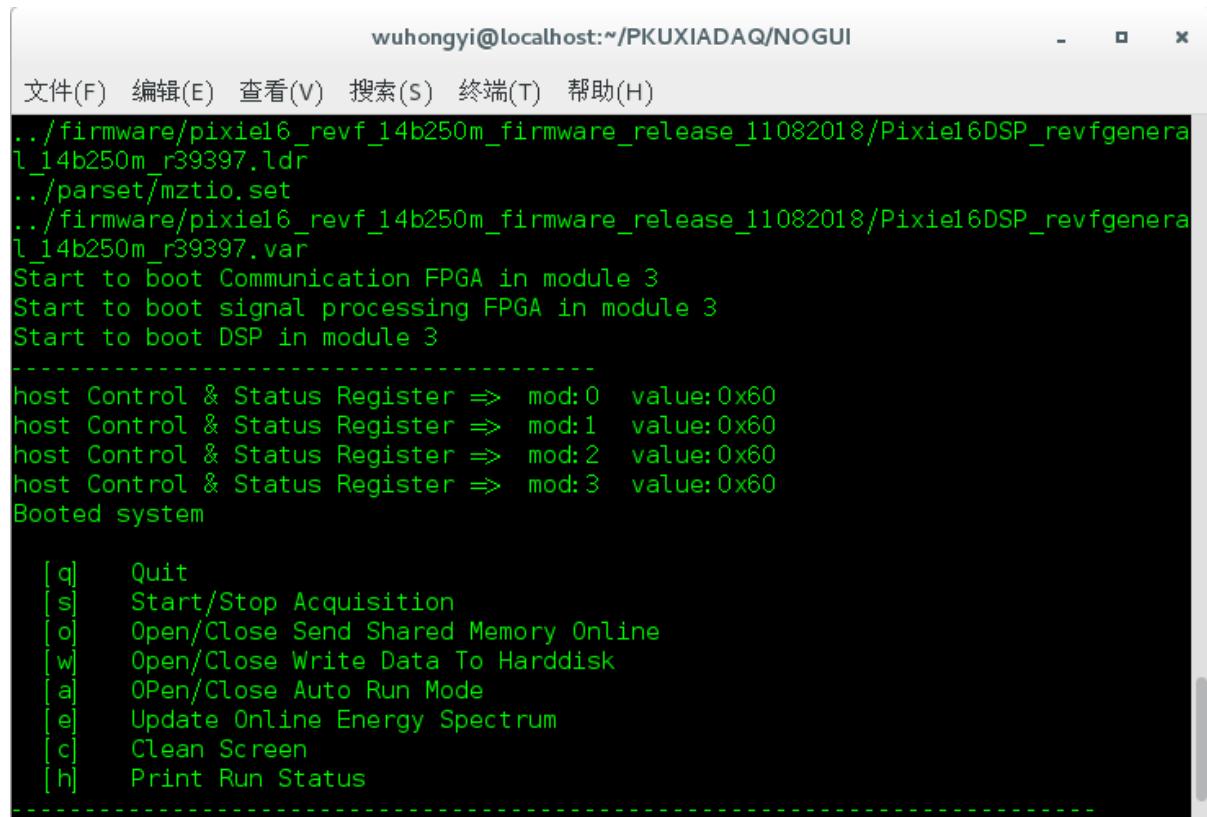
7.6.13 Simulation(暂未实现)

通过模型产生不同类型探测、不同信噪比的波形,辅助使用者学习参数优化调节的。

非图形交互界面

8.1 控制界面

当开启程序之后，将自动加载固件进行初始化，等待初始化结束后，出现以下界面



```
wuhongyi@localhost:~/PKUXIADAQ/NOGUI
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
.../firmware/pixie16_revf_14b250m_firmware_release_11082018/Pixie16DSP_revfgenera
l_14b250m_r39397.ldr
.../parset/mztio.set
.../firmware/pixie16_revf_14b250m_firmware_release_11082018/Pixie16DSP_revfgenera
l_14b250m_r39397.var
Start to boot Communication FPGA in module 3
Start to boot signal processing FPGA in module 3
Start to boot DSP in module 3
-----
host Control & Status Register => mod:0 value:0x60
host Control & Status Register => mod:1 value:0x60
host Control & Status Register => mod:2 value:0x60
host Control & Status Register => mod:3 value:0x60
Booted system
-----
[ q] Quit
[ s] Start/Stop Acquisition
[ o] Open/Close Send Shared Memory Online
[ w] Open/Close Write Data To Harddisk
[ a] Open/Close Auto Run Mode
[ e] Update Online Energy Spectrum
[ c] Clean Screen
[ h] Print Run Status
```

以下是所有命令的功能，

- | | |
|-----|----------------|
| [q] | 退出程序 |
| [s] | 控制获取的开始、结束 |
| [o] | 控制在线共享内存的开启、关闭 |

(下页继续)

(续上页)

- | | |
|-----|----------------|
| [w] | 控制数据写入硬盘的开启、关闭 |
| [a] | 控制自动运行模式的开启、关闭 |
| [e] | 刷新在线监视中的能谱 |
| [c] | 清除屏幕，显示本命令提示 |
| [h] | 输出运行控制参数 |

下图是典型的运行界面信息，可以清晰看到当前的运行模式。

```
s
created the directory /home/wuhongyi/data/0320.
open: /home/wuhongyi/data/0320/data_R0320_M00.bin
open: /home/wuhongyi/data/0320/data_R0320_M01.bin
open: /home/wuhongyi/data/0320/data_R0320_M02.bin
open: /home/wuhongyi/data/0320/data_R0320_M03.bin
RUN START
SHM Open!
Running No. 320
Auto Run Mode: CLOSE
Send Shared Memory Online: OPEN
Write Data To Harddisk: OPEN
a
The times for each run: 180 s
h

[ q]    Quit
[ s]    Start/Stop Acquisition
[ o]    Open/Close Send Shared Memory Online
[ w]    Open/Close Write Data To Harddisk
[ a]    Open/Close Auto Run Mode
[ e]    Update Online Energy Spectrum
[ c]    Clean Screen
[ h]    Print Run Status
-----
Running No. 320
Auto Run Mode: OPEN - 180 s per run
Send Shared Memory Online: OPEN
Write Data To Harddisk: OPEN
```

8.2 自动运行设置

当开启自动运行模式时，将会按照用户所设定的时间自动切换到下一轮运行。

时间参数设置在文件 parset/cfgPixie16.txt

```
# Only use in NOGUI, unit: second
AutoRunModeTimes    180
```


在线统计

9.1 控制界面

修改 **OnlineStatics** 中的文件 **PixieOnline.config**, 其中第一行为原始二进制文件存放路径, 第二行为文件名。通过该两行参数来监视每个文件时时大小及硬盘占用量。

通过执行以下命令, 开启在线监视主界面:

```
./online
```

检查二进制文件路径、文件名是否有问题, 如果没问题则点击按钮 **Complete**, 之后点击 **RunStart** 则开启在线监视, 在线监视每 3 秒刷新一次。可时时监视每路的触发率、每路的实际事件输出率。

监视界面如下:

GDDAQ Online																									
File		CountRate		Alert		EnergyMonitor		Setup		File Path:		File Name: data		Complete		RunStop		R0520 M11							
Monitor																									
ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s					
00	0	0	00	0	0	00	0	0	00	0	0	00	0	0	00	0	0	00	0	0					
01	0	0	01	0	0	01	0	0	01	0	0	01	0	0	01	0	0	01	0	0					
02	0	0	02	0	0	02	0	0	02	0	0	02	0	0	02	0	0	02	0	0					
03	0	0	03	0	0	03	0	0	03	0	0	03	0	0	03	0	0	03	0	0					
04	0	0	04	0	0	04	0	0	04	0	0	04	0	0	04	0	0	04	0	0					
05	0	0	05	0	0	05	0	0	05	0	0	05	0	0	05	0	0	05	0	0					
06	0	0	06	0	0	06	0	0	06	0	0	06	0	0	06	0	0	06	0	0					
07	0	0	07	0	0	07	0	0	07	0	0	07	0	0	07	0	0	07	0	0					
08	0	0	08	0	0	08	0	0	08	0	0	08	0	0	08	0	0	08	0	0					
09	0	0	09	0	0	09	0	0	09	0	0	09	0	0	09	0	0	09	0	0					
10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0					
11	0	0	11	0	0	11	0	0	11	0	0	11	0	0	11	0	0	11	0	0					
12	0	0	12	0	0	12	0	0	12	0	0	12	0	0	12	0	0	12	0	0					
13	0	0	13	0	0	13	0	0	13	0	0	13	0	0	13	0	0	13	0	0					
14	0	0	14	0	0	14	0	0	14	0	0	14	0	0	14	0	0	14	0	0					
15	0	0	15	0	0	15	0	0	15	0	0	15	0	0	15	0	0	15	0	0					
ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s	ch #	InRate/s	OutRate/s					
00	0	0	00	0	0	00	0	0	00	0	0	00	0	0	00	0	0	M00	0	100					
01	0	0	01	0	0	01	0	0	01	0	0	01	0	0	01	0	0	M01	0	100					
02	0	0	02	0	0	02	0	0	02	0	0	02	0	0	02	0	0	M02	0	100					
03	0	0	03	0	0	03	0	0	03	0	0	03	0	0	03	0	0	M03	0	100					
04	0	0	04	0	0	04	0	0	04	0	0	04	0	0	04	0	0	M04	0	100					
05	0	0	05	0	0	05	0	0	05	0	0	05	0	0	05	0	0	M05	0	100					
06	0	0	06	0	0	06	0	0	06	0	0	06	0	0	06	0	0	M06	0	100					
07	0	0	07	0	0	07	0	0	07	0	0	07	0	0	07	0	0	M07	0	100					
08	0	0	08	0	0	08	0	0	08	0	0	08	0	0	08	0	0	M08	0	100					
09	0	0	09	0	0	09	0	0	09	0	0	09	0	0	09	0	0	M09	0	100					
10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	M10	0	100					
11	0	0	11	0	0	11	0	0	11	0	0	11	0	0	11	0	0	M11	0	100					
12	0	0	12	0	0	12	0	0	12	0	0	12	0	0	12	0	0	M12	0	100					
13	0	0	13	0	0	13	0	0	13	0	0	13	0	0	13	0	0								
14	0	0	14	0	0	14	0	0	14	0	0	14	0	0	14	0	0								
15	0	0	15	0	0	15	0	0	15	0	0	15	0	0	15	0	0								
														Used 6706		Available 9246		Use% 43%							

CHAPTER 10

MakeEvent

本转换程序的使用前提，插件必须从第一个插槽开始，中间不留空插槽。

MakeEvent 程序用来快速将数据组装成与传统 VME 获取数据类似的结构，方便实验时的初步物理分析，最终的物理分析不能以本程序产生的数据为基准。

用户首先需要修改 **UesrDefine.hh** 文件中的定义

```
#define OUTFILEPATH "/home/wuhongyi/data/"
#define RAWFILEPATH "/home/wuhongyi/data/"
#define RAWFILENAME "data"

// 设置插件个数
#define BOARDNUMBER 5
```

用户需要修改：

- 原始 ROOT 文件的路径
- 生成的事件结构 ROOT 文件的存放路径
- 文件名
- 使用采集卡个数

修改之后执行以下命令编译程序：

```
make clean
make
```

编译成功之后将生成一个可执行文件 **event**，程序运行方式：

```
./event [RunNnumber] [windows]
```

其中 **[RunNnumber]** 为想要转换的文件运行编号，**[windows]** 为事件的时间窗，单位为 ns。

ROOT File Branch:

- sr: 采样率，该事件中该通道数值不为 0 表示探测到信号。
- adc: 能量

- outofr: 标记是否超模数转换的量程
- qdc: QDC 的八段积分
- tdc: 时间
- cfd: cfd 数值
- cfdft: 标记 CFD 数值是否有效
- cfds: 仅适用于 250/500 MHz 采集卡, cfd source

TODO 这里添加一个 Branch 截图。。。

CHAPTER 11

插件前面板

每个 Pixie-16 模块的前面板上都有 16 个模拟信号输入连接器，一个 LVDS I/O 端口，五个数字 I/O 连接器以及前面板底部附近的三个 LED。另外，在前面板的顶部手柄上贴有显示 Pixie-16 型号的标签（例如 P16L-250-14，表示 Pixie-16 250 MHz 14bit 模块）。Pixie-16 模块的序列号（例如，S/N 1100）位于前面板的底部手柄处。

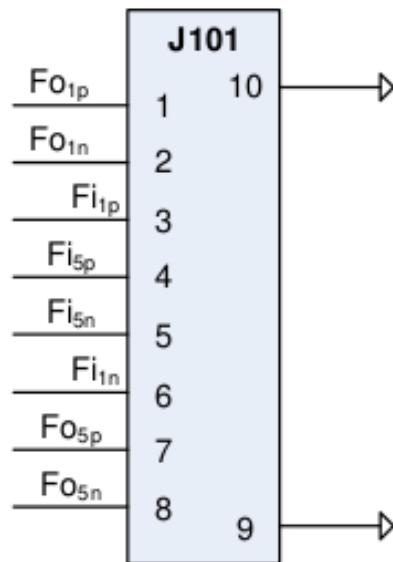
11.1 模拟信号输入连接器 (all revisions)

- 连接器标签
 - 16 通道：0 至 15
- 连接器类型
 - SMB Jack

每个 Pixie-16 模块均接受 16 个模拟输入信号，每个输入连接器均为 SMB 插孔（插头触点）连接器。

11.2 LVDS I/O 端口 (all revisions)

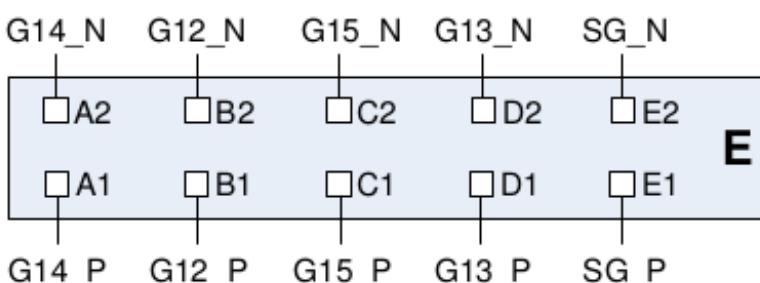
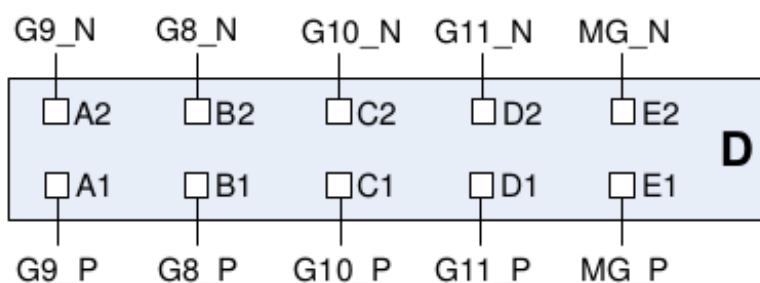
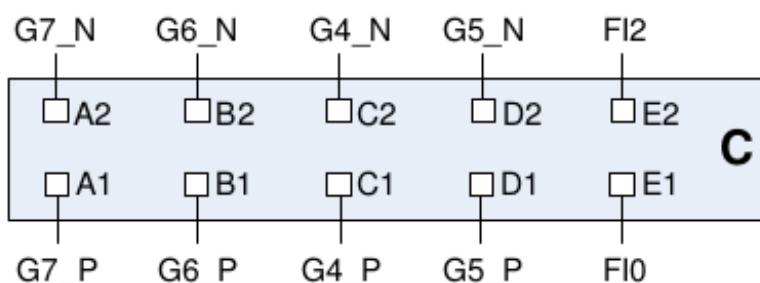
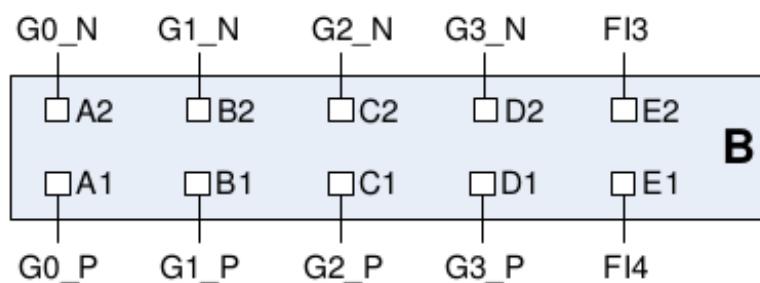
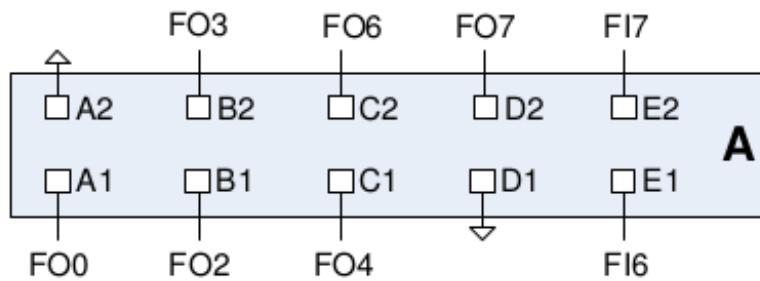
Connected Signals	4 LVDS pairs (Fo_{1p}/Fo_{1n} , Fi_{1p}/Fi_{1n} , Fi_{5p}/Fi_{5n} , Fo_{5p}/Fo_{5n} , see below for pin layout)
Signal Direction	Input or Output, software configurable
Cable Type	Cat 5 or Cat 6 (the same ones used for Ethernet)



每个 Pixie-16 模块在其前面板上均配有一个 LVDS I/O 端口。LVDS 代表低压差分信号。LVDS I/O 连接器是 RJ45 连接器，这意味着可以使用相同的 Cat 5 或 Cat 6 以太网电缆将信号连接到该 I/O 端口或从该 I/O 端口连接信号。但是，此 Pixie-16 I/O 端口没有以太网连接。

可从此 I/O 端口获得四个差分信号对，即引脚对 Fo_{1p} / Fo_{1n} , Fi_{1p} / Fi_{1n} , Fi_{5p} / Fi_{5n} , and Fo_{5p} / Fo_{5n} 。每对都可以配置为输入或输出信号。

11.3 数字 I/O 连接器 (Rev. F only)



Pixie-16 Rev. F 模块的前面板上装有五个 har-link 连接器，用作数字 I/O 连接器。HARTING 的 2mm 间距 har-link 连接器专为高速数据传输而设计，速率高达 2 Gbit/s。其电磁干扰屏蔽如图 2mm pitch har-link [2]

connector from HARTING 所示，可确保在电磁污染环境下的出色性能。



每个 har-link 连接器有 2 行，每行有 5 个引脚，并使用从 A 到 E 的五个红色字母之一标记。表 *Rev. F Module's Digital I/O Connectors* 中定义了连接到这五个连接器的每个引脚的信号。

Connector Type	har-link® (HARTING, 2mm pin spacing)
FI ₀ , FI ₂ , FI ₃ , FI ₄ , FI ₆ , FI ₇	TTL digital input signals (max. 5V)
FO ₀ , FO ₂ , FO ₃ , FO ₄ , FO ₆ , FO ₇	Digital outputs for test/debug purpose (TTL 5V)
Gx_P/Gx_N (x=0-15)	Channel Gate Inputs (0-15 for 16 channels) (LVDS format)
MG_P/MG_N	Module Gate Input (LVDS format)
SG_P/SG_N	Spare Gate Input (LVDS format)

其中，FI₀, FI₂, FI₃, FI₄, FI₆, FI₇ 是六个 TTL 数字输入信号。它们可以是诸如全局快速触发 (global validation trigger)，全局验证触发 (global validation trigger)，外部时钟 (external clock)，运行禁止 (run inhibit) 等信号。每个输入引脚的具体用法取决于下载到 Pixie-16 模块的特定固件 (标准固件支持的输入信号请参见上表)。六个数字输出信号 FO₀, FO₂, FO₃, FO₄, FO₆, FO₇ 连接到 Pixie-16 的系统 FPGA 上的六个测试输出引脚，可用于帮助用户进行系统设置。这些测试引脚连接到 Pixie-16 的各种内部信号，以洞悉系统的当前状态。

通道门输入 (16 通道为 0-15) 是 LVDS 格式的输入信号，可独立控制 Pixie-16 模块 16 个通道中每个通道的数据采集。

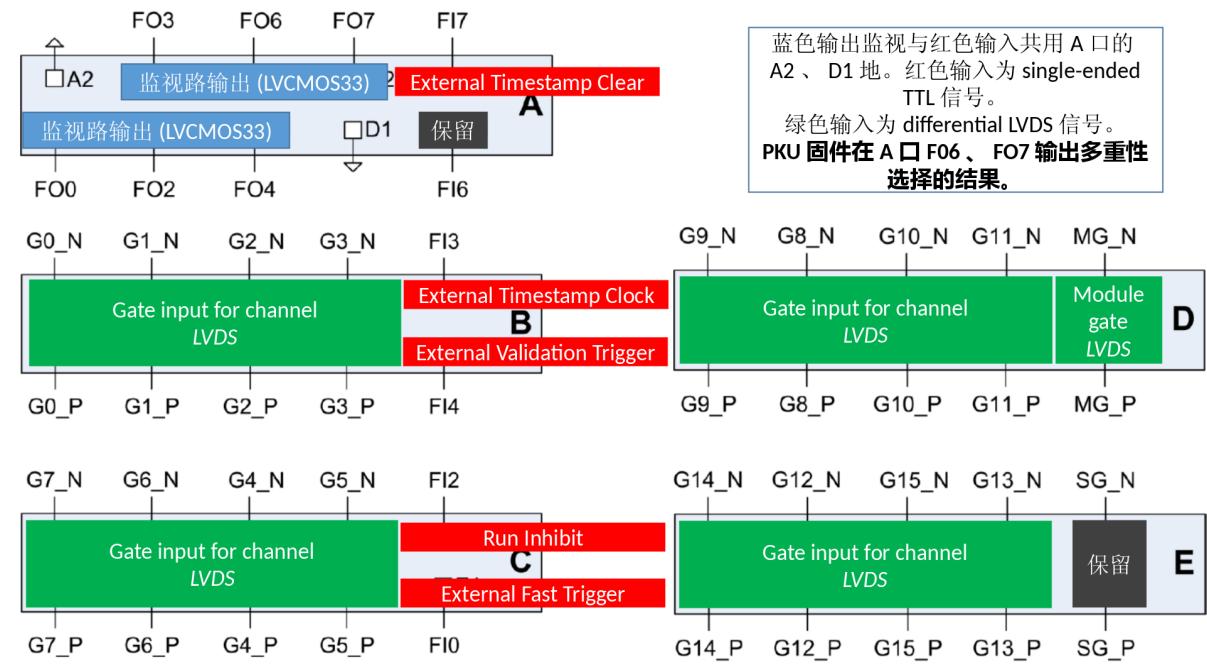
通道门信号是电平敏感信号，即当通道门信号的电平为逻辑高电平 (1) 时，该门信号有效；当通道门信号的电平为逻辑低电平 (0) 时，未使用门信号。在正常情况下，将通道门信号设置为否决 (veto) 给定通道中的数据采集，即，在该通道中快速触发到达时，如果通道门信号为逻辑高 (1)，则该快速触发 (fast filter) 被否决，因此将其丢弃。但是，可以通过软件在 FPGA 中设置相应的寄存器来反转这种逻辑。在这种情况下，将通道门信号设置为验证给定通道中的数据采集，即在该通道中的快速触发时，仅当通道门信号为逻辑高电平 (1) 时，该触发才会被触发并记录事件。

模块门控输入是 LVDS 格式信号，用于门控 Pixie-16 模块所有 16 个通道中的数据采集。它也是一个对电平敏感的信号，即当模块门信号的电平为逻辑高电平 (1) 时，门信号有效；当模块的电平为逻辑低电

平 (0) 时，未使用门信号。在正常情况下，如果模块门信号为逻辑高 (1)，则将模块门信号设置为否决 (veto) 所有 16 个通道中的数据采集，即在 16 个通道中的任何一个通道中快速触发到达时，则该通道的快速触发已被否决，因此将被丢弃。但是，可以通过软件在 FPGA 中设置相应的寄存器来反转这种逻辑。在这种情况下，将模块门信号设置为验证所有 16 个通道中的数据采集，即，在 16 个通道中的任何一个快速触发时，仅当模块门信号为逻辑高电平 (1) 时，该通道的快速触发被接受以记录事件。

备用门输入是 LVDS 格式的信号，保留给特殊应用。

此类应用程序通常需要开发定制固件以支持 Pixie-16 系统的特殊功能。



11.4 前面板 LEDs (all revisions)

在 Pixie-16 前面板底部附近，有三个 LED。它们从左到右分别标记为 RUN，I/O 和 ERR。它们分别对应三种不同的颜色，分别为 ** 绿色，黄色和红色 **。

LED Name	Color	Function
RUN	Green	ON when run is in progress, and OFF if run is stopped or not started yet
I/O	Yellow	Flashing when there is I/O activity on the PCI bus between the Pixie-16 module and host computer
ERR	Red	ON when there is no more space in the External FIFO for storage of list mode event data, and OFF when there is sufficient space to store at least one more list mode event data (ON does not indicate any actual error condition. Rather, it simply indicates the External FIFO's FULL condition)

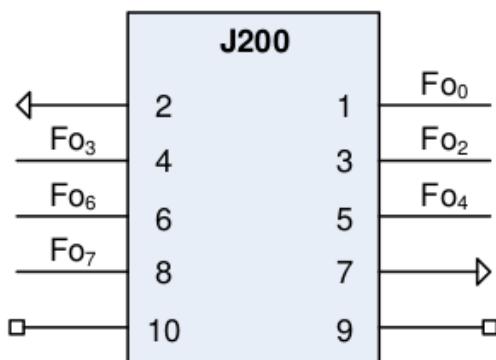
当 Pixie-16 模块中的运行正在进行时，**RUN LED** 将亮起，而当运行停止或尚未开始时，则熄灭。

当 Pixie-16 模块和主机之间的 PCI 总线上有 I/O 活动时，**I/O LED** 将闪烁。

实际上，**ERR LED** 并不表示 Pixie-16 模块中的任何错误情况。相反，它用于指示 Pixie-16 模块的外部 FIFO 是否已满。当外部 FIFO 中没有足够的空间来存储列表模式事件数据时，该指示灯将亮起；如果有足够的空间来存储至少一个列表模式事件数据，则该指示灯将熄灭。当外部 FIFO 已满时，无法再将列表模式事件数据写入其中，直到主机软件通过 PCI 总线读出外部 FIFO 中的部分数据为止。

11.5 3.3V I/O 连接器 (Rev. D only)

Connector Type	Single-ended, 2mm pin spacing
FO ₀ , FO ₂ , FO ₃ , FO ₄ , FO ₆ , FO ₇	Digital outputs for test/debug purpose (3.3V)



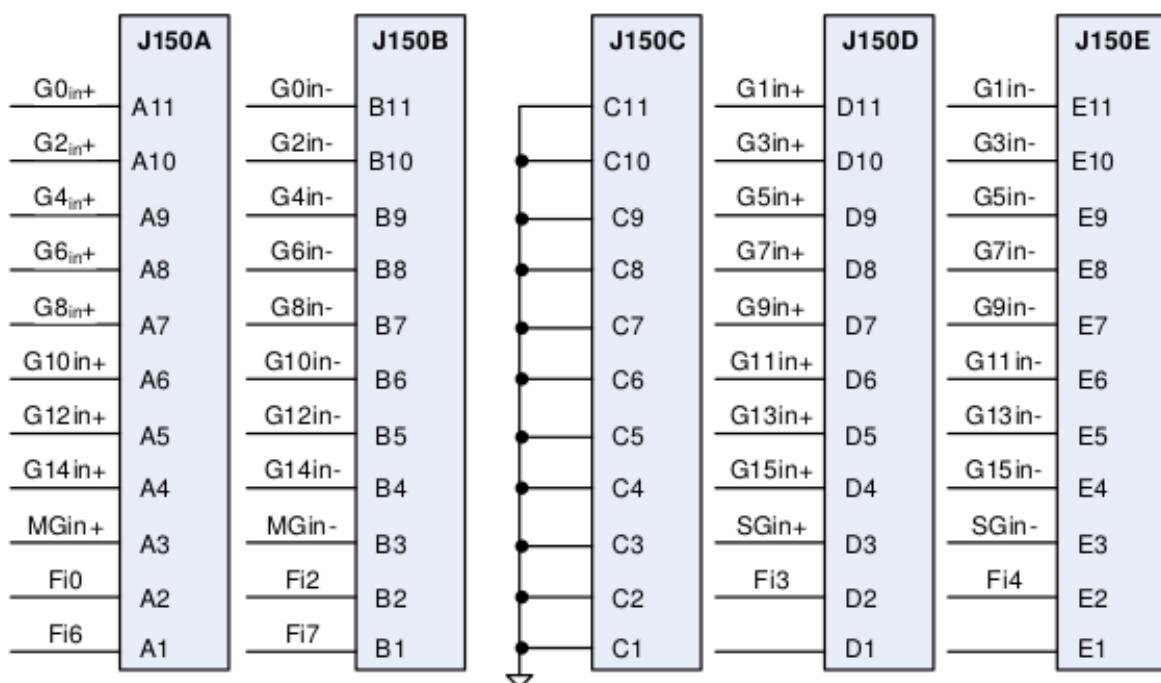
在版本 D Pixie-16 模块上，通道 7 和通道 8 的模拟输入 SMB 连接器之间是 3.3V I/O 连接器 (J200)。它具有 10 个 2mm 间距的单端引脚。引脚 # 1、3、4、5、6 和 8 连接到来自 Pixie-16 模块的系统 FPGA 的六个数字输出信号，即 FO₀、FO₂、FO₃、FO₄、FO₆、FO₇，主要用于测试和调试。# 2 和 7 引脚为接地引脚，而 # 9 和 10 引脚未使用。

11.6 GATE Inputs(Rev. D only)

Connector Type	Amphenol FCI® 55 Position Header, 2mm pin spacing
FI ₀ , FI ₂ , FI ₃ , FI ₄ , FI ₆ , FI ₇	TTL digital input signals (max. 5V)
G _{xin+} /G _{xin-} (x=0-15)	Channel Gate Inputs (0-15 for 16 channels) (LVDS format)
MG _{in+} /MG _{in-}	Module Gate Input (LVDS format)
SG _{in+} /SG _{in-}	Spare Gate Input (LVDS format)



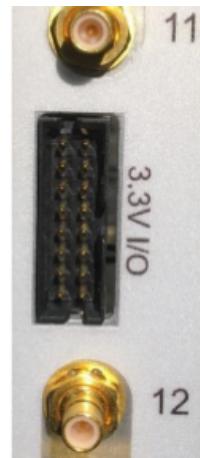
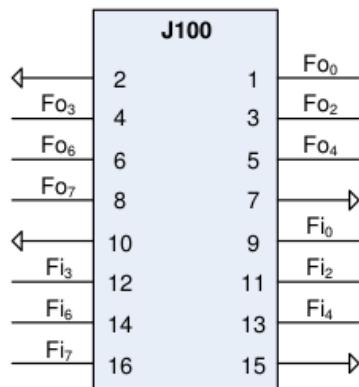
在版本 D Pixie-16 模块上，通道 11 和通道 12 的模拟输入 SMB 连接器之间是门输入连接器。此连接器是 2mm 针间距的 Amphenol FCI® 55 Position Header。这 55 个引脚的布局如图 *Rev. D Module's GATE INPUTS Connector* 所示。中间引脚列 (J150C) 的 11 个引脚均接地。在门输入连接器的前 8 行中，来自 A/B 列 (J150A/J150B) 或 D/E 列 (J150D/J150E) 的每对差分引脚对应于一个通道的输入门，其格式为 LVDS，例如 G_{xin+}/G_{xin-}(x=0-15)。J150A3/J150B3 上的差分引脚对是模块栅极输入信号 MG_{in+}/MG_{in-}。通道门输入信号可用于否决或验证给定通道自己的触发信号。模块门输入信号在整个模块级别上起作用，即，它可用于否决或验证给定模块的所有 16 个通道自己的触发信号。J150D3/J150E3 上的差分引脚对是备用输入信号 SG_{in+}/SG_{in-}。备用门输入信号可用于需要定制固件的特殊应用。



在 Rev. D Pixie-16 模块上，TTL 数字输入信号（最大 5V），即 FI₀, FI₂, FI₃, FI₄, FI₆, FI₇，分布在门输入连接器的底部两行中，如图 *Rev. D Module's GATE INPUTS Connector* 所示。

11.7 3.3V I/O 连接器 (Rev. B and C only)

Connector Type	Single-ended, 2mm pin spacing
FI ₀ , FI ₂ , FI ₃ , FI ₄ , FI ₆ , FI ₇	TTL digital input signals (max. 5V)
FO ₀ , FO ₂ , FO ₃ , FO ₄ , FO ₆ , FO ₇	Digital outputs for test/debug purpose (3.3V)



在 B 和 C 版本的 Pixie-16 模块上，在通道 11 和通道 12 的模拟输入 SMB 连接器之间是 3.3V I/O 连接器 (J100)。它具有 16 个单端引脚，间距为 2mm。引脚 # 1、3、4、5、6 和 8 连接到来自 Pixie-16 模块的系统 FPGA 的六个数字输出信号，即 FO₀, FO₂, FO₃, FO₄, FO₆, FO₇，主要用于测试和调试。引脚 # 2、7、10 和 15 是接地引脚。引脚 # 9、11、12、13、14 和 16 连接到六个 TTL 数字输入信号 (最大 5V)，即 FI₀, FI₂, FI₃, FI₄, FI₆, FI₇。

11.8 标准固件中的数字信号 (all revisions)

Pixie-16 的标准固件通过其前面板 I/O 连接器支持数字信号的输入和输出，这已在前面进行了讨论。

TTL digital input signals	Connected signals in standard firmware	Direction	Description
FI ₀	EXT_FASTTRIG	Input	External fast trigger signal
FI ₂	INHIBIT	Input	Run inhibit signal
FI ₃	EXT_TS_CLK	Input	External timestamp clock signal
FI ₄	EXT_VALIDTRIG	Input	External validation signal
FI ₆	not used		
FI ₇	EXT_TS_CLR	Input	External timestamp clear signal

表 *TTL Digital Input Signals* 显示了 Pixie-16 标准固件支持的五个 TTL 数字输入信号。

其中，信号 EXT_TS_CLK 和 EXT_TS_CLR 用于 Pixie-16 中的外部时间戳记，即 Pixie-16 接受外部时钟信号（为了避免出现时钟信号完整性问题，不建议该外部时钟的频率超过 20 MHz），使用 48 位计数器对此类时钟信号进行计数，并在发生事件触发时将此类计数器值输出到列表模式数据流。

通过将两个系统记录的事件的外部时间戳相关联，外部时间戳可用于将 Pixie-16 数据获取系统与另一个数据获取系统同步。

当在 Pixie-16 模块中启用了同步要求时，外部系统会使用 INHIBIT 信号来禁止在 Pixie-16 系统中运行数据采集。这是一个对电平敏感的信号，即当 INHIBIT 信号为逻辑高电平时，Pixie-16 的运行将不会开始。只有当 INHIBIT 信号变为逻辑低电平时，Pixie-16 才会开始运行。在运行期间，如果 INHIBIT 信号返回逻辑高电平，则运行将中止。

EXT_FASTTRIG 信号是外部快速触发信号，可用于代替本地快速触发来在 Pixie-16 模块中记录事件。**EXT_VALIDTRIG** 信号是外部验证信号，可用于验证 Pixie-16 模块中的事件。

Connector J101 Pins	Connected signals in standard firmware	Direction	Description
Fo _{1p} /Fo _{1n}	not used		
Fi _{1p} /Fi _{1n}	LVDS_VALIDTRIG	Input	External validation trigger signal in LVDS format
Fi _{5p} /Fi _{5n}	LVDS_FASTTRIG	Input	External fast trigger signal in LVDS format
Fo _{5p} /Fo _{5n}	SYNC_LVDS_FP	Output	Pixie-16 synchronization output signal in LVDS format (to synchronize with other DAQ systems)

表 *Connector J101 LVDS I/O Port Signals* 显示了 Pixie-16 连接器 J101 LVDS I/O 端口信号。此 J101 LVDS I/O 端口可以使用常规以太网电缆进行连接，但没有以太网连接。该 J101 端口可提供的四对 LVDS 对中，当前未使用一对，输入使用两对，输出使用一对。**LVDS_VALIDTRIG** 是 LVDS 格式的外部验证触发信号，**LVDS_FASTTRIG** 是 LVDS 格式的外部快速触发信号。**SYNC_LVDS_FP** 是来自 Pixie-16 模块的输出信号，用于向外部数据采集系统指示 Pixie-16 系统的同步状态，以便可以同步两个数据采集系统。

TTL digital output signals	Connected signals in standard firmware		Direction	Description	
FO ₀	FTRIG_DELAY	FTRIG_DELAY	Output	Delayed local fast trigger of one of the 16 channels	Delayed local fast trigger of one of the 16 channels
FO ₂	FTRIG_VAL	VETO_CE	Output	Validated, delayed local fast trigger one of the 16 channels	Stretched veto trigger of one of the 16 channels
FO ₃	ETRIG_CE	LDPMFULL	Output	Stretched external global validation trigger of one of the 16 channels	Module level dual port memory (DPM) full status flag
FO ₄	CHANTRIG_CE	SDPMFULL	Output	Stretched channel validation trigger of one of the 16 channels	System level dual port memory (DPM) full status flag
FO ₆	FTIN_OR	FTIN_OR	Output	OR of 16 local fast triggers	OR of 16 local fast triggers
FO ₇	TEST_SEL	TEST_SEL	Output	Selected test signal	Selected test signal

表 *TTL Digital Output Signals* 列出了六个 Pixie-16 TTL 数字输出信号。可以通过软件设置选择两组六个输出信号（请参见 TrigConfig0 的位 [14:12] 和 [19:16]）。可以通过软件设置进一步选择最后的输出信号 TEST_SEL。有关这些信号的更多详细信息将在本手册的后续部分中提供。

CHAPTER 12

逻辑功能

12.1 逻辑概括

对于某路信号的每个事件是否被有效记录取决于：

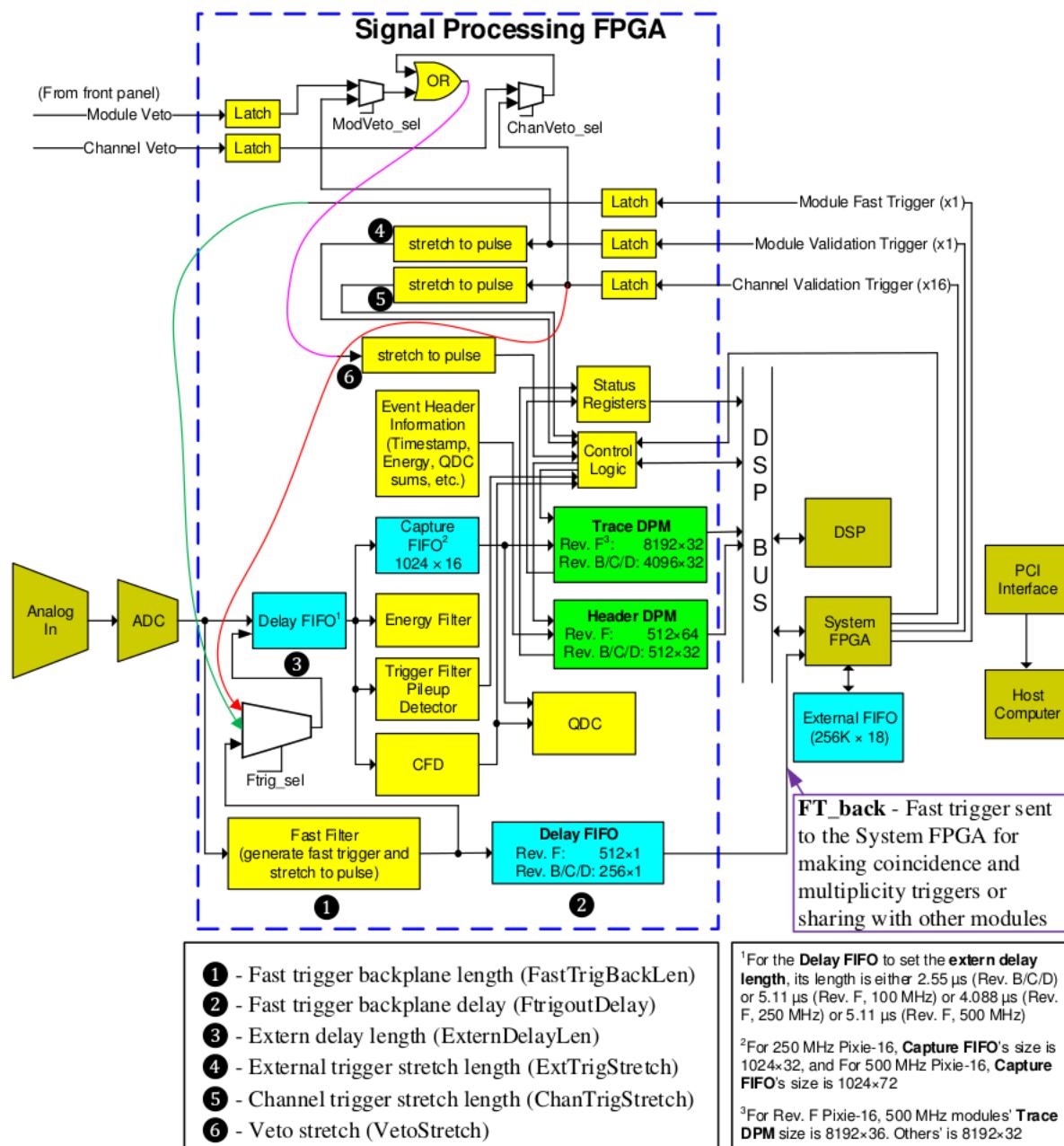
- Fast trigger select (一级 trigger)
- Control logic (二级 trigger)

快速触发的选择 (Fast trigger select)：

- 快速滤波器 (Local fast filter)
- 通道有效触发 (Channel validation trigger)
- 模块快速触发 (Module fast trigger)

控制逻辑 (Control logic)：

- 模块有效触发 (Module validation trigger)
- 通道有效触发 (Channel validation trigger)
- 否决 (Veto)
- 堆积 (Pileup)
- ...



如图所示，输入模拟脉冲首先由 ADC 数字化，然后进入信号处理 FPGA 中的信号处理电路，每个信号处理电路来自 Pixie-16 模块的 4 个通道的 ADC 数据。

数字化数据流首先被送入两个分支：快速过滤器 (fast filter) 生成快速触发 (fast triggers) 以发送到系统 FPGA，延迟 FIFO 可用于补偿快速触发 (fast triggers) 和外部触发 (external triggers) 之间的延迟。

然后，通过延迟 FIFO 的数字化数据流分为四个部分：

- 能量滤波器，采集能量滤波器在 PeakSample 时间的能量；
- 触发滤波器，检测脉冲并进行堆积检查；
- 捕获 FIFO，当检测到有效脉冲时，ADC 数据在流入跟踪双端口存储器 (DPM) 之前根据跟踪延迟参数值延迟 ADC 数据；
- CFD 电路，其中生成 CFD 触发以触发 QDC，锁存时间戳和记录波形的计算。

信号处理 FPGA 中的控制逻辑利用本地快速触发 (local fast trigger)，CFD 触发 (CFD trigger)，否决 (VETO) 和外部触发 (external triggers) 来确定是否以及何时将波形数据流入 Trace DPM 并将事件信息写入 Header DPM。DSP 通过状态寄存器监视 DPM 的状态，并通过 DSP 总线和系统 FPGA 将事件数据移入外部 FIFO(External FIFO)。

触发展宽长度 (Trigger Stretch Lengths)

- 外部触发展宽 (**External trigger stretch**) 用于展宽模块有效触发 (module validation trigger) 脉冲。
- 通道触发展宽 (**Channel trigger stretch**) 用于展宽通道有效触发 (channel validation trigger) 脉冲。
- **Veto 展宽 (Veto stretch)** 用于展宽该通道的否决脉冲。
- 快速触发背板长度 (**Fast trigger backplane length**) 用于展宽要发送到系统 FPGA 的快速触发脉冲，其中此快速触发可以发送到背板以与其它模块共享，或者可以用于进行符合或多重触发。

FIFO 延迟 (FIFO Delays)

- 外部延迟长度 (**External delay length**) 用于延迟输入 ADC 波形和本地快速触发，以便补偿外部触发脉冲的延迟到达，例如：模块有效触发，通道有效触发等。
- 快速触发背板延迟 (**Fast trigger backplane delay**) 用于在将快速触发脉冲发送到系统 FPGA 之前将其延迟，以便通过背板与其它模块共享或进行符合或多重触发。

Parameters	Range	
	100 or 500 MHz	250 MHz
<i>External trigger stretch</i>	0.01 – 40.95 μs	0.008 – 32.76 μs
<i>Channel trigger stretch</i>	0.01 – 40.95 μs	0.008 – 32.76 μs
<i>Veto stretch</i>	0.01 – 40.95 μs	0.008 – 32.76 μs
<i>Fast trigger backplane length</i>	0.01 – 40.95 μs	0.008 – 32.76 μs
<i>External delay length</i>	0 – 2.55 μs (Rev. B/C/D) 0 – 5.11 μs (Rev. F)	0 – 4.088 μs
<i>Fast trigger backplane delay</i>	0 – 2.55 μs (Rev. B/C/D) 0 – 5.11 μs (Rev. F)	0 – 4.088 μs

12.2 Module Fast Trigger(for trigger)

Module fast trigger 有以下四种来源可供选择：

- **Ext_FastTrig_In(来源于本插件)**
 - Ext_FastTrig_Sel(前面板 TTL 输入)
 - Int_FastTrig_Sgl(内部某路 FT)
 - FTIN_Or(内部 FT 的 OR)
 - LVDS_FastTrig_FP(前面板网口输入)
 - ChanTrig_Sel(内部某路的 valid trigger)(与 module validation trigger 共用一个设置)
- FT_LocalCrate_BP(本机箱中指定插件送出的 trigger)
- FT_In_BP(多机箱中指定机箱上指定插件发送的 trigger)
- FT_WiredOr(本机箱中所有插件发送出 trigger 的 OR)

12.3 Module Validation Trigger(for control logic)

Module validation trigger 有以下来源可供选择：

- **Ext_ValidTrig_In(来源于本插件)**
 - Ext_ValidTrig_Sel(前面板 TTL 输入)
 - Int_ValidTrig_Sgl(内部某路 FT)

- FTIN_Or(内部 FT 的 OR)
- LVDS_ValidTrig_FP(前面板网口输入)
- ChanTrig_Sel(内部某路的 valid trigger)(与 module fast trigger 共用一个设置)
- ET_LocalCrate_BP(本机箱中指定插件送出的 trigger)
- ET_In_BP(多机箱中指定机箱上指定插件发送的 trigger)
- ET_WiredOr(本机箱中所有插件发送出 trigger 的 OR)
- 前面板 module GATE 输入 LVDS 信号

12.4 Channel Validation Trigger(for trigger/control logic)

Channel validation trigger 来源有以下选择:

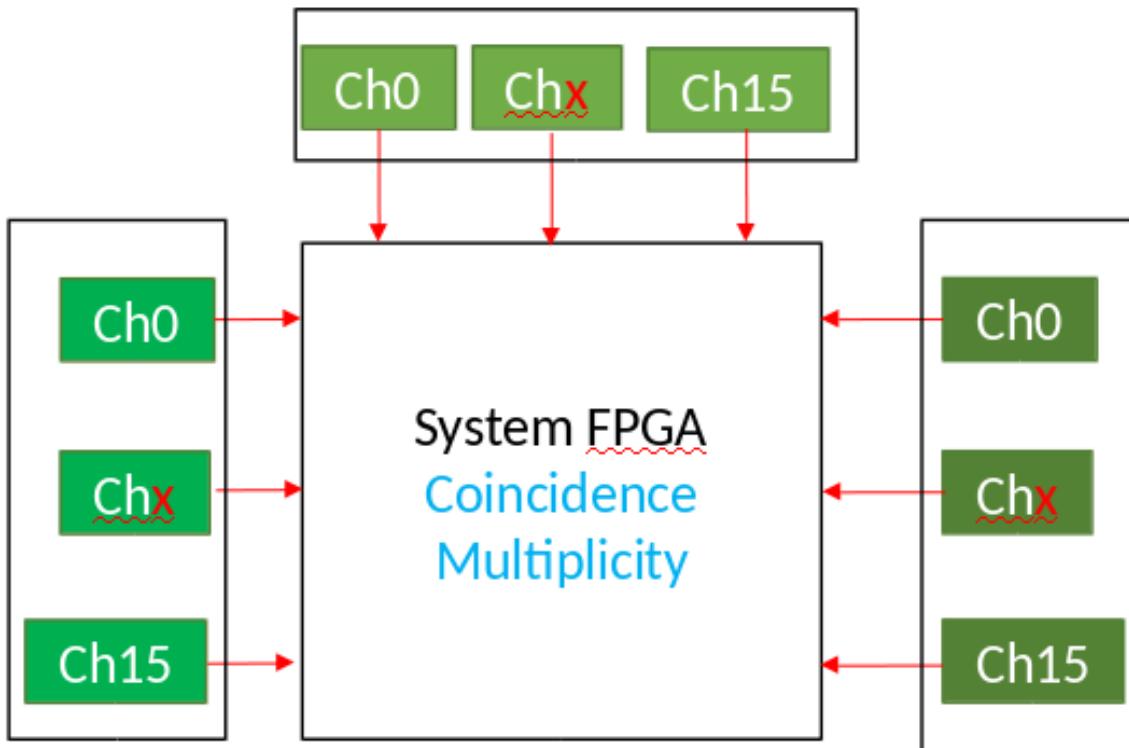
- 每路独立设置, 来源于多重性选择
- 每路独立设置, 来源于符合
- 每 4 路共用一个设置, 来源于左、中、右插件某路的 FT
- 每 4 路共用一个设置, 来源于自身 FT 与 Ext_FastTrig_In 的符合
- 每路独立设置, 前面板 channel GATE 输入 LVDS 信号 (与前面板 Veto 共用一个输入口)

12.5 Veto

来源于 ModuleVeto 与 ChannelVeto 的 OR:

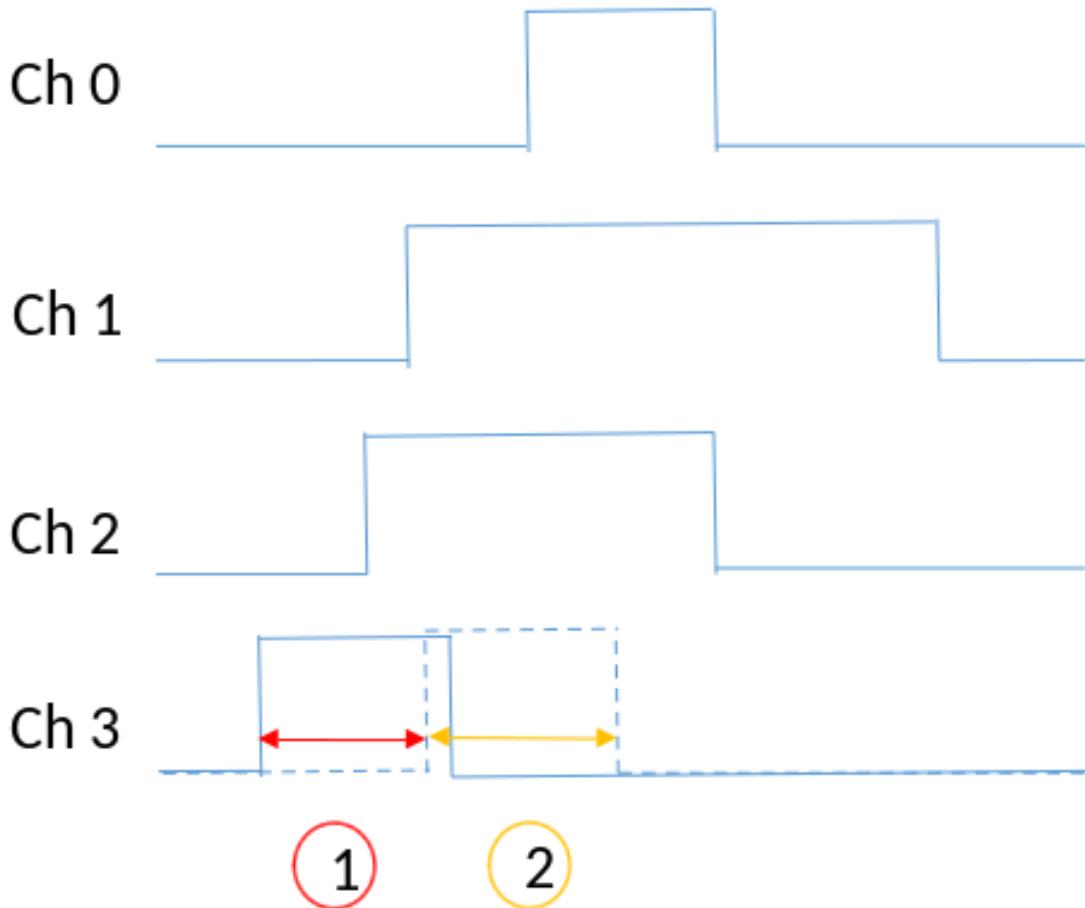
- **ModuleVeto 来源有两个选择:**
 - Module Validation Trigger
 - 前面板 Module Gate
- **ChannelVeto 来源有两个选择:**
 - Channel Validtion Trigger
 - 前面板 Gate input for channel (与前面板 Channel validation trigger 共用一个输入口)

12.6 System FPGA (coincidence/multiplicity)



Multiplicity: 对设置的该 channel 来说，左邻插件、自身插件、右邻插件共 48 路，可以选择参与多重性选择的路数

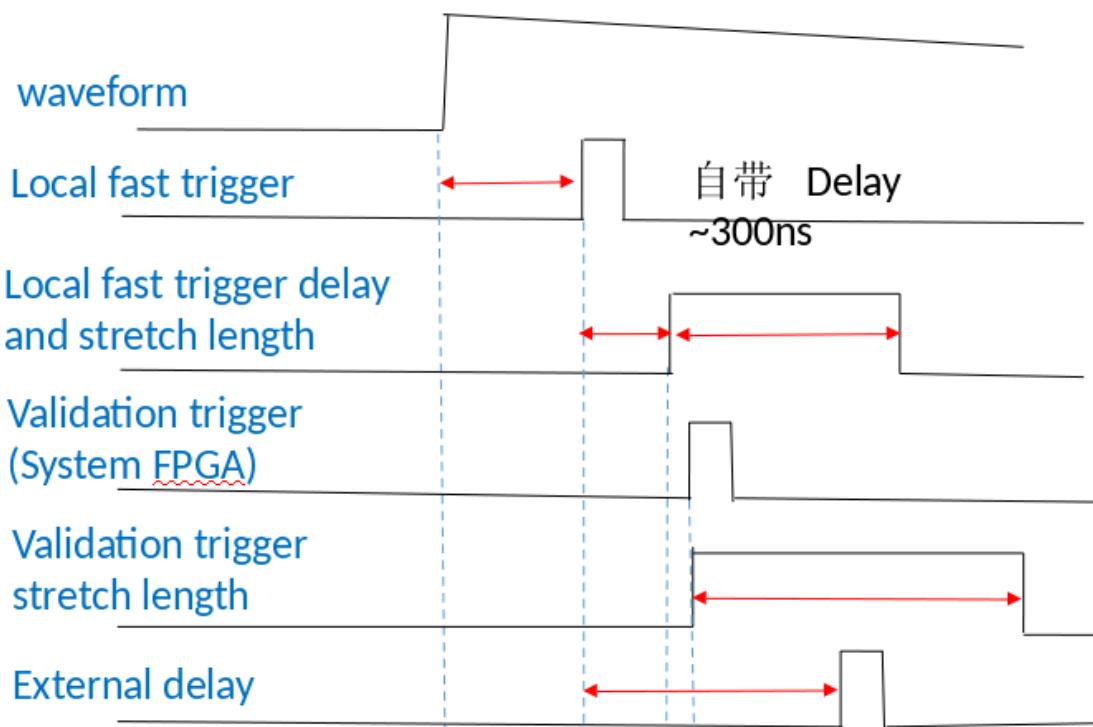
Coincidence: 对设置的该 channel 来说，左邻插件、自身插件、右邻插件，每个插件均满足设置的符合条件，才能给出符合触发



其它插件的 fast filter trigger 通过机箱背板传到该插件需要大约 100 ns 左右。因此通过调节门宽、延迟来保证符合、多重性选择的合理设置。

- Fast trigger stretch length: 设置 fast trigger 门宽,
- fast trigger delay length: 设置 fast trigger 延迟。

Control logic (module/channel validation trigger)

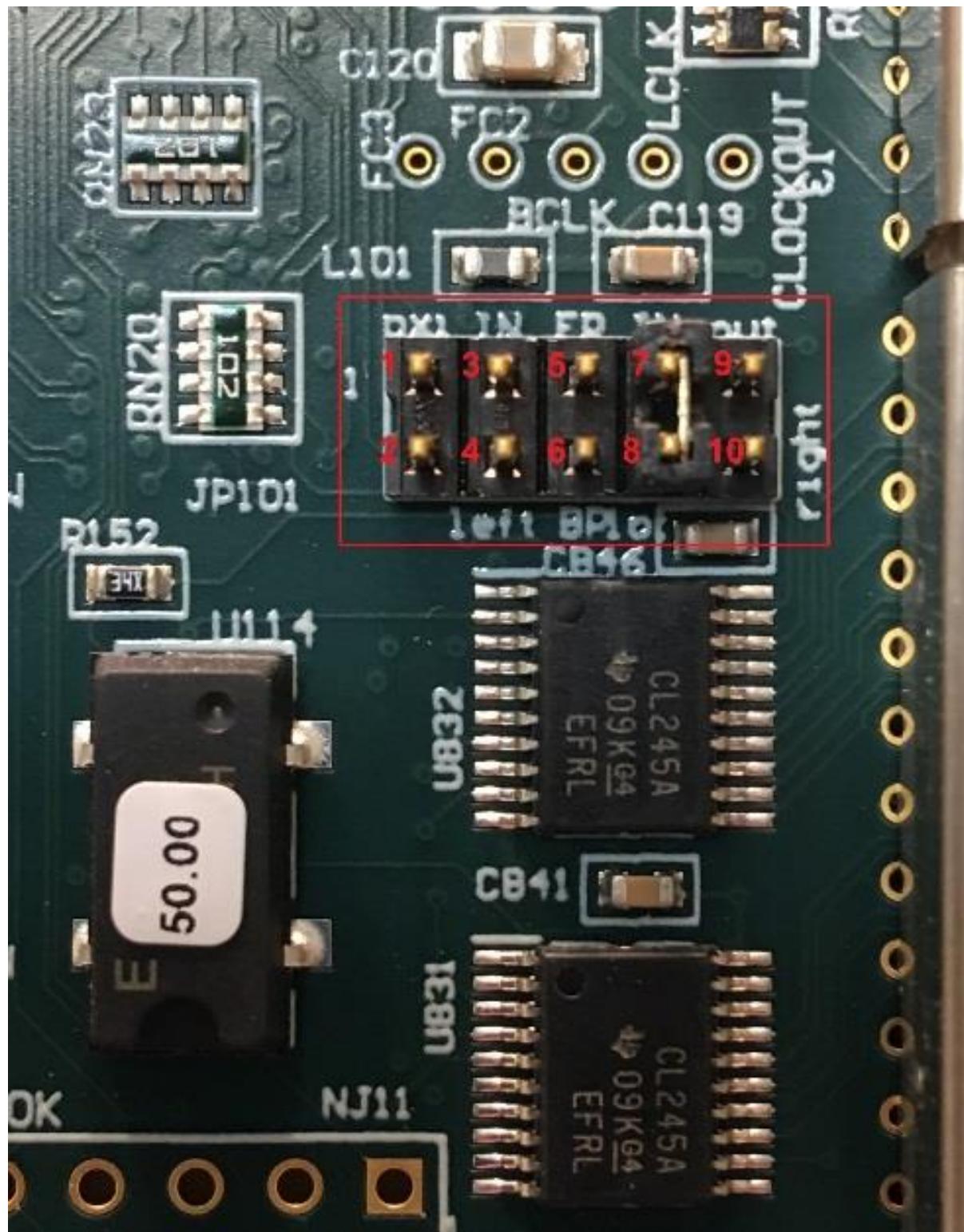


特别需要注意信号经过背板传输大约需要时间 100 ns。

CHAPTER 13

时间同步

当许多 Pixie-16 模块作为一个系统一起运行时，可能需要在它们之间同步时钟和定时器，并在模块之间分配触发。还需要确保在所有模块中同步启动和停止运行。所有这些信号都分布在 Pixie-16 机箱的 PXI 背板。

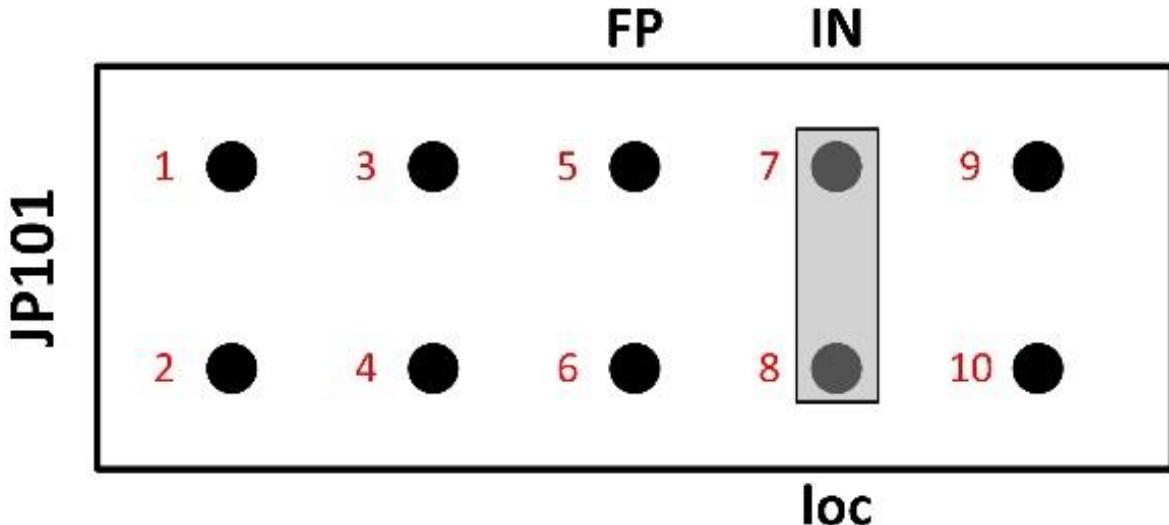


在多模块系统中，将有一个时钟主机和多个时钟从机或中继器。可以通过在电路板右下角附近的跳针 JP101 上设置分流器来选择模块的时钟功能。10 针跳针 JP101 如图所示，顶部标有红色标记。提供分流器以连接适合于每种所选时钟分配模式的引脚。下面描述了四种时钟分配模式，独立时钟模式，PXI 时钟模式，菊花链时钟模式和多机箱时钟模式。

警告：在 250 MHz 或 500 MHz Pixie-16 模块中，FPGA 中的信号处理时钟频率分别降频为 125 MHz 或 100 MHz，以便更容易地实现设计。无论何时重新初始化模块，该降频都可能导致不同的时钟相位，从而导致给定的 250 MHz 或 500 MHz Pixie-16 模块中每个通道的时间截偏移量不同。可能需要校准

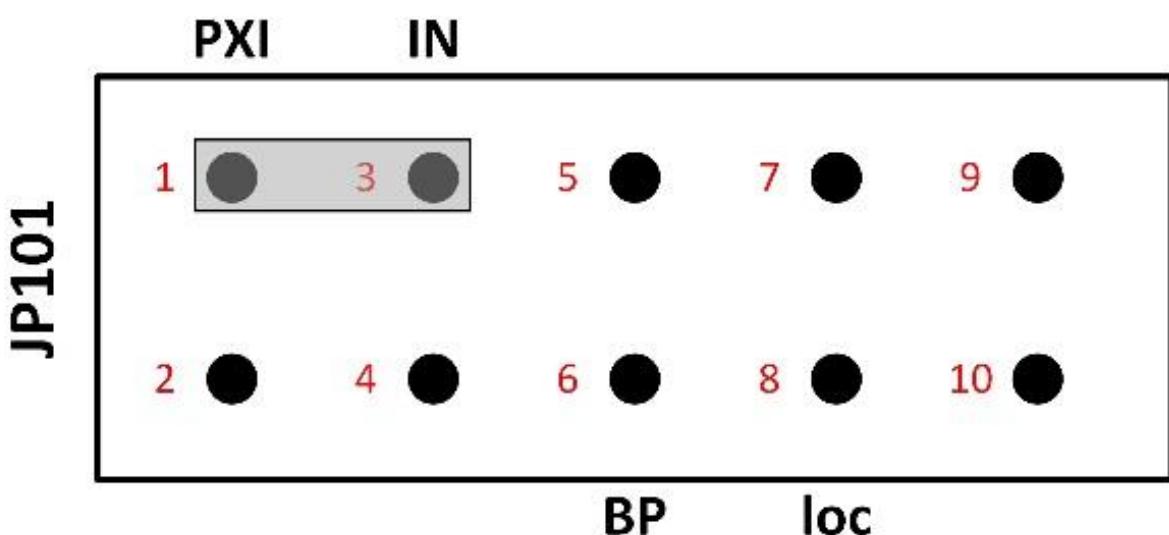
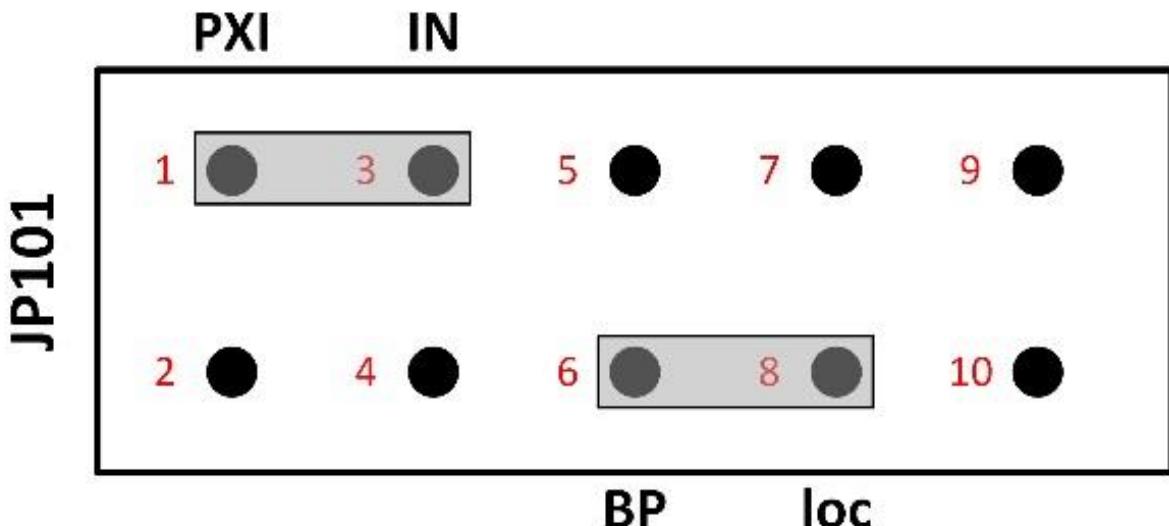
来量化每个通道的不同时间戳偏移。

13.1 独立时钟模式



如果系统中仅使用一个 Pixie-16 模块，或者模块之间的时钟不需要同步，则应将模块设置为单独的时钟模式。将 JP101 的引脚 7（时钟输入）与分流器连接到引脚 8（loc-IN）。这将使用 Pixie-16 模块的 50 MHz 本地晶体振荡器作为时钟源。

13.2 PXI 时钟模式



Top: PXI clock master (slot 2)

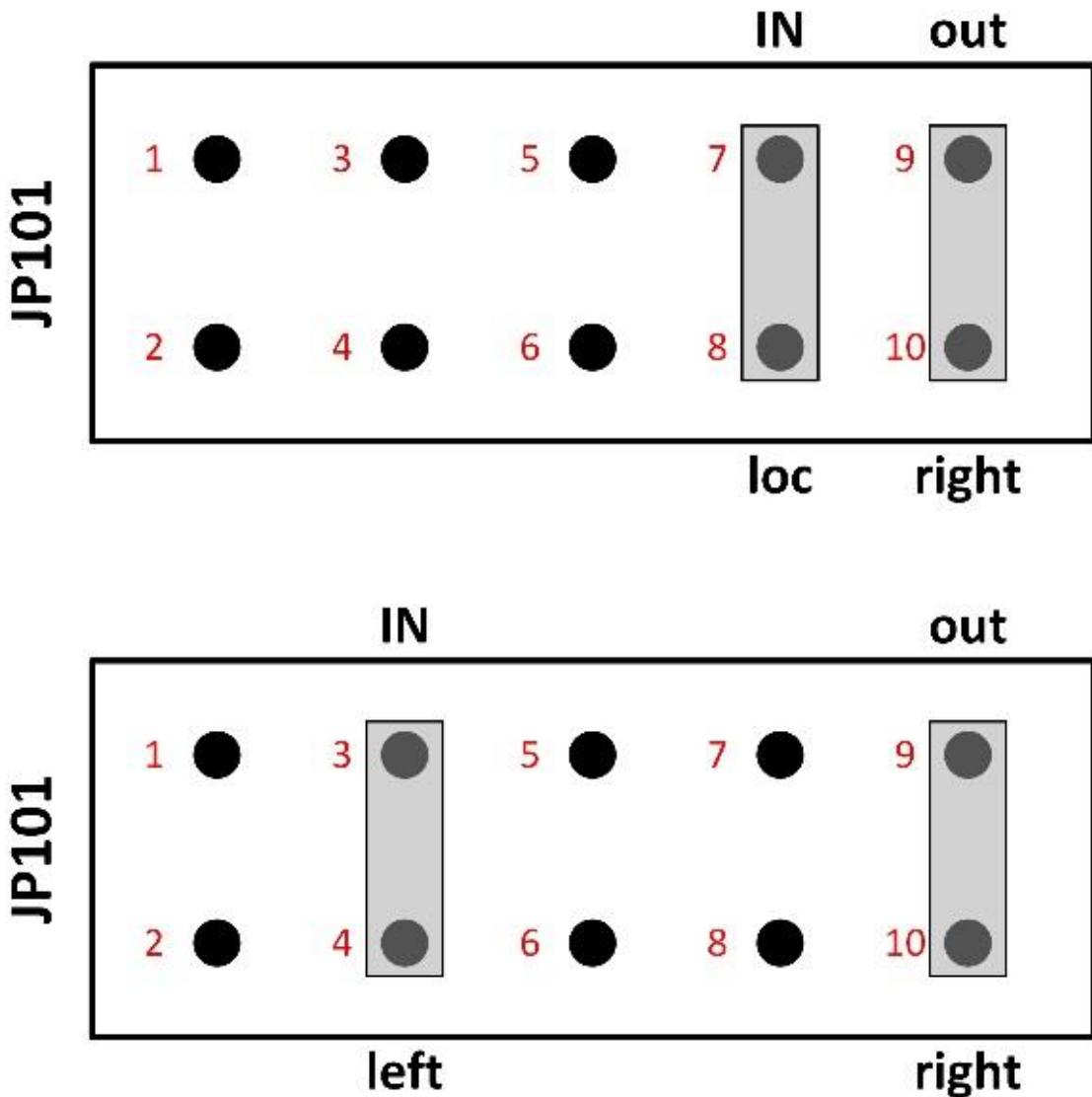
Bottom: PXI clock recipient (slots 3-14)

在多个 Pixie-16 模块之间分配时钟的首选方法是使用分布在背板上的 PXI 时钟。该时钟默认在背板上生成，是一个 10 MHz 的时钟信号，然后由扇出缓冲区重复该信号，并通过具有最小偏差（每个插槽的走线长度相等）的专用线连接到每个机箱插槽。虽然 10 MHz 的速度太慢，不足以成为 Pixie-16 的有用时钟，但它可以被安装在 2 号插槽中的 Pixie-16 模块通过 JP101 上适当的分流器设置发出的本地时钟信号所覆盖。

通过连接 JP101 的插脚 6 和 8 (LOC-BP)，插槽 2 中的 Pixie-16 模块可以配置为 PXI 时钟主机。所有模块，包括时钟主机，都应设置为通过连接 JP101 (PXI-IN) 上的插脚 1 和 3 来接收 PXI 时钟。这样，来自 Pixie-16 时钟主机的 50 MHz 时钟通过几乎相同的时钟相位的背板分配给所有 Pixie-16 模块。

与菊花链时钟模式相比，PXI 时钟模式的另一个优点是，除了必须安装在插槽 2 中的 Pixie-16 主模块外，其它 Pixie-16 模块可以安装在 Pixie-16 机箱的任何其他插槽中。相反，当使用菊花链时钟模式时，所有 Pixie-16 模块必须相邻安装，即模块之间不允许有间隙。

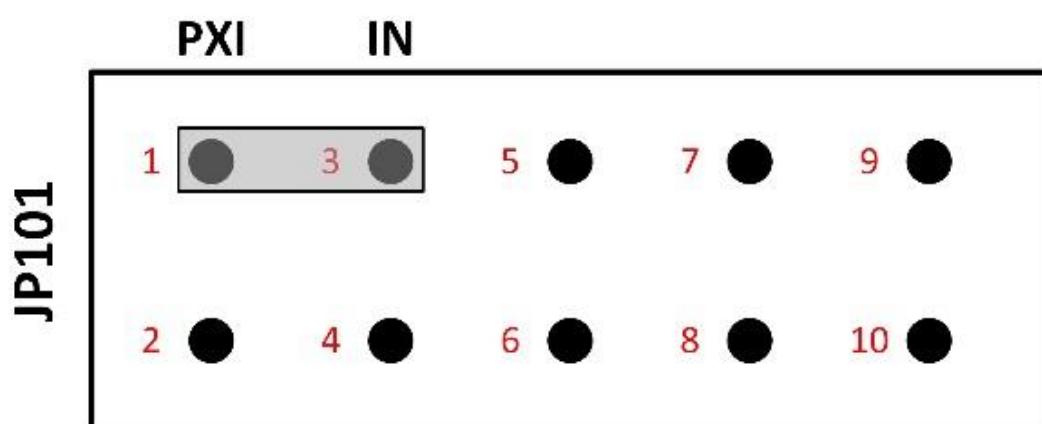
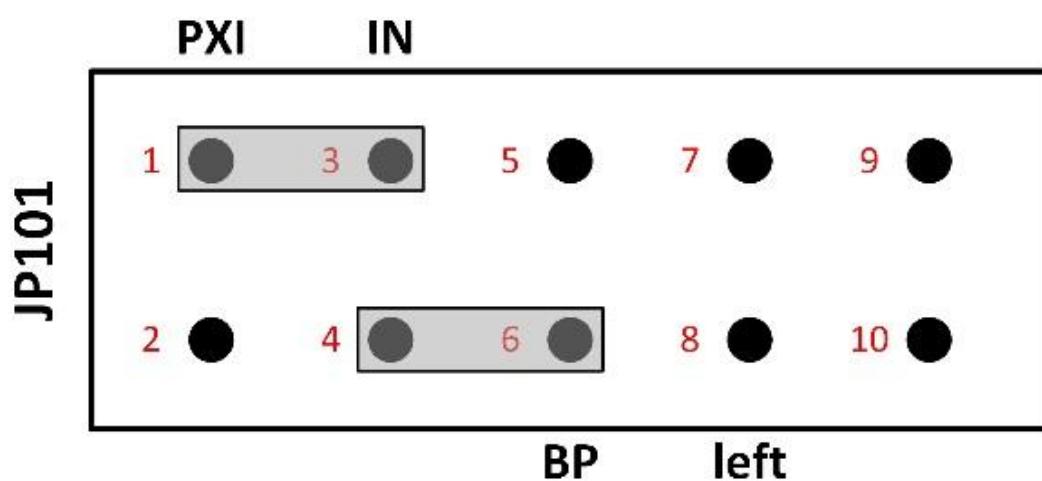
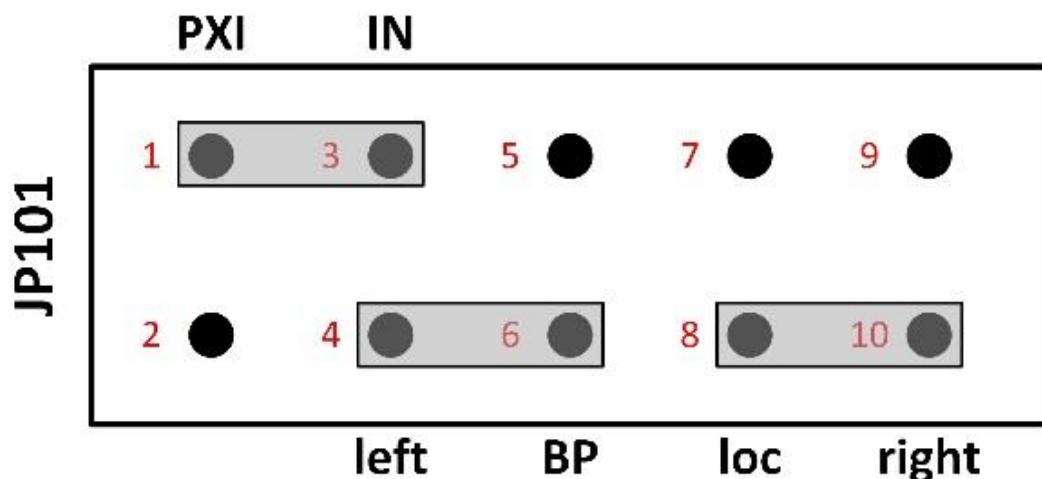
13.3 菊花链时钟模式



Top: Daisy-chain clock master (leftmost module)
Bottom: Daisy-chain clock repeater

时钟分配的另一种选择是将时钟从一个模块串到另一个模块，每个模块重复时钟信号并将其发送到右侧的邻居。这需要一个主模块，位于 Pixie-16 模块组最左边的插槽中。主模块使用其本地晶体振荡器作为输入，并将其输出发送到右侧 (loc-IN, out-right)。机箱中的其它 Pixie-16 模块应配置为时钟中继器，使用来自左邻居的信号作为输入，并将其输出发送至右 (left - IN, out - right)。但是，正如前面提到的，模块之间必须没有插槽间隙。

13.4 多机箱时钟模式



Top: System Director Module

Middle: Crate Master Module

Bottom: Regular Module

在多机箱系统中，可以使用专用的触发器和时钟分配卡（即 XIA 提供的 Pixie-16 背板 I/O 触发器模块）在这些机箱之间分配全局时钟信号。

两个机箱之间的时钟分布示例如下所示。

13.4.1 Pixie-16 模块安装

多个 Pixie-16 模块可安装在两个 14 槽 Pixie-16 机箱中，机箱 1 和机箱 2。为了进行时钟分配，机箱 1 被称为主机箱，其中所有 Pixie-16 模块的系统范围全局时钟都起源于主机箱，而机箱 2 被称为从机箱，从主机箱接收全局时钟。

安装在主机箱插槽 2 中的 Pixie-16 模块被指定为系统控制器模块，其本地 50 MHz 晶体振荡器充当系统范围全局时钟的源。通过 Pixie-16 背板 I/O 触发模块，将时钟信号从系统控制器模块分配到双机箱系统中的所有 Pixie-16 模块。

安装在从机箱插槽 2 中的 Pixie-16 模块称为机箱主模块，该模块负责接收来自主机箱的全局时钟，并通过背板上的长度匹配记录道将该时钟发送给该机箱中的所有模块。系统控制器模块还负责将全局时钟发送到主机箱中的所有模块。因此，它也是一个机箱主模块。这两个机箱中的其它模块是常规模块。表中显示了双机箱系统中不同类型的模块。

Crate #	1				
Slot #	2	3	...	13	14
Module	System Director Module / Crate Master Module	Regular Module	...	Regular Module	Regular Module
Crate #	2				
Slot #	2	3	...	13	14
Module	Crate Master Module	Regular Module	...	Regular Module	Regular Module

13.4.2 Pixie-16 模块上的时钟跳线 (JP101) 设置

对于双机箱系统中使用相同全局时钟信号的所有 Pixie-16 模块，所有模块中的时钟跳线 (JP101) 应根据表 *Clock Jumper JP101 Settings in a 2-crate System* 和图 *multi-crate clock mode* 进行设置。

System Director Module	Connect pins 1 and 3, 4 and 6, 8 and 10.
Crate Master Module	Connect pins 1 and 3, 4 and 6.
Regular Module	Connect pins 1 and 3.

13.4.3 Pixie-16 背板 I/O 触发模块的电缆连接

Pixie-16 背板 I/O 触发模块安装在每个机箱的背面，机箱背面安装了 6U 卡槽。图 *rear I/O trigger modules* 显示一个 Pixie-16 背板 I/O 触发模块分别安装在控制器或主模块的正后方，用于在多个 Pixie-16 机箱之间共享时钟、触发和运行启动或停止同步信号。背板的背面有连接器 J3、J4 和 J5，但没有 J1 和 J2，因为它不需要使用 CompactPCI 或 PXI 通信。

通常，安装了 J3、J4、J5 连接器的背板的第一个插槽是应安装 Pixie-16 背板 I/O 触发模块的插槽。安装模块时，请确保顶部和底部轨道与触发模块对齐，以避免损坏背板插针。

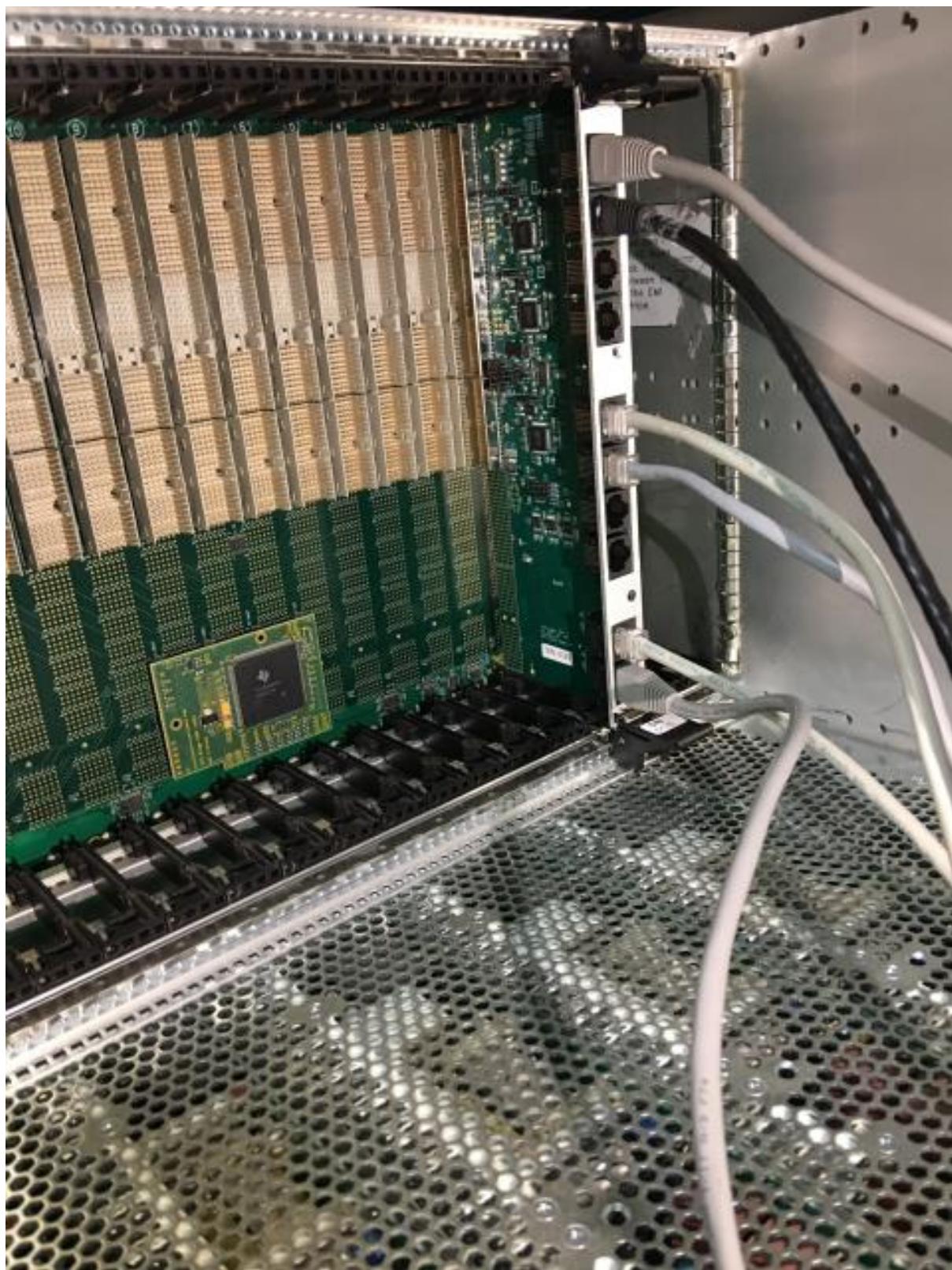
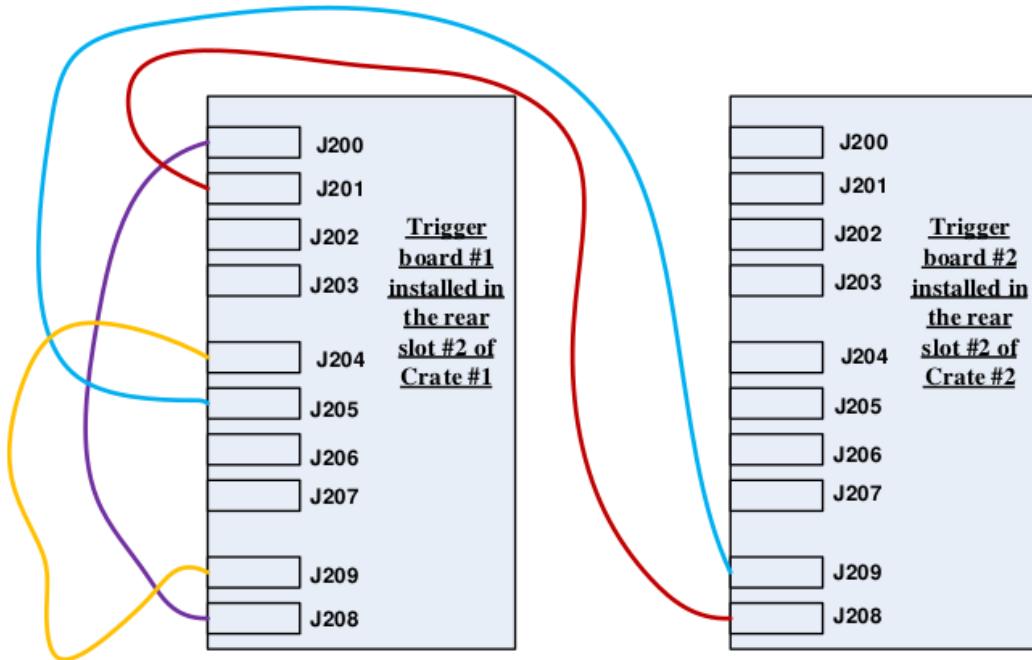


图 *Cable connections between two Pixie-16 rear I/O trigger modules* 显示安装在两个机箱中的两个 Pixie-16 背板 I/O 触发模块之间的电缆连接。所有连接电缆均为 5 类或 6 类以太网电缆，其长度应相同，以尽量减小两个机箱中 Pixie-16 模块之间的时钟相位差。



13.4.4 Pixie-16 背板 I/O 触发模块上的跳线设置

触发模块 1 安装在机箱 1 的后槽 2 中。如前所述，后槽 2 位于机箱背面，位于机箱前槽 2 的正对面。在将触发模块安装到后槽 2 中时应小心，避免弯曲背板背面的任何针脚，因为这可能导致 3.3 V 针脚与相邻接地针脚短路，从而损坏整个背板。

请注意，当面向模块顶部时，触发模块上所有跳线的插脚编号从右到左计数，即底板连接器 J3 到 J5 在左侧（只有 JP1 例外，它是垂直方向的，应从下到上计数）。跳线右侧画有一个很小的“1”标签，指示插脚 1。图 *Pin numbering for the jumpers on the Pixie-16 rear I/O trigger module* 显示红色框中的管脚“1”。

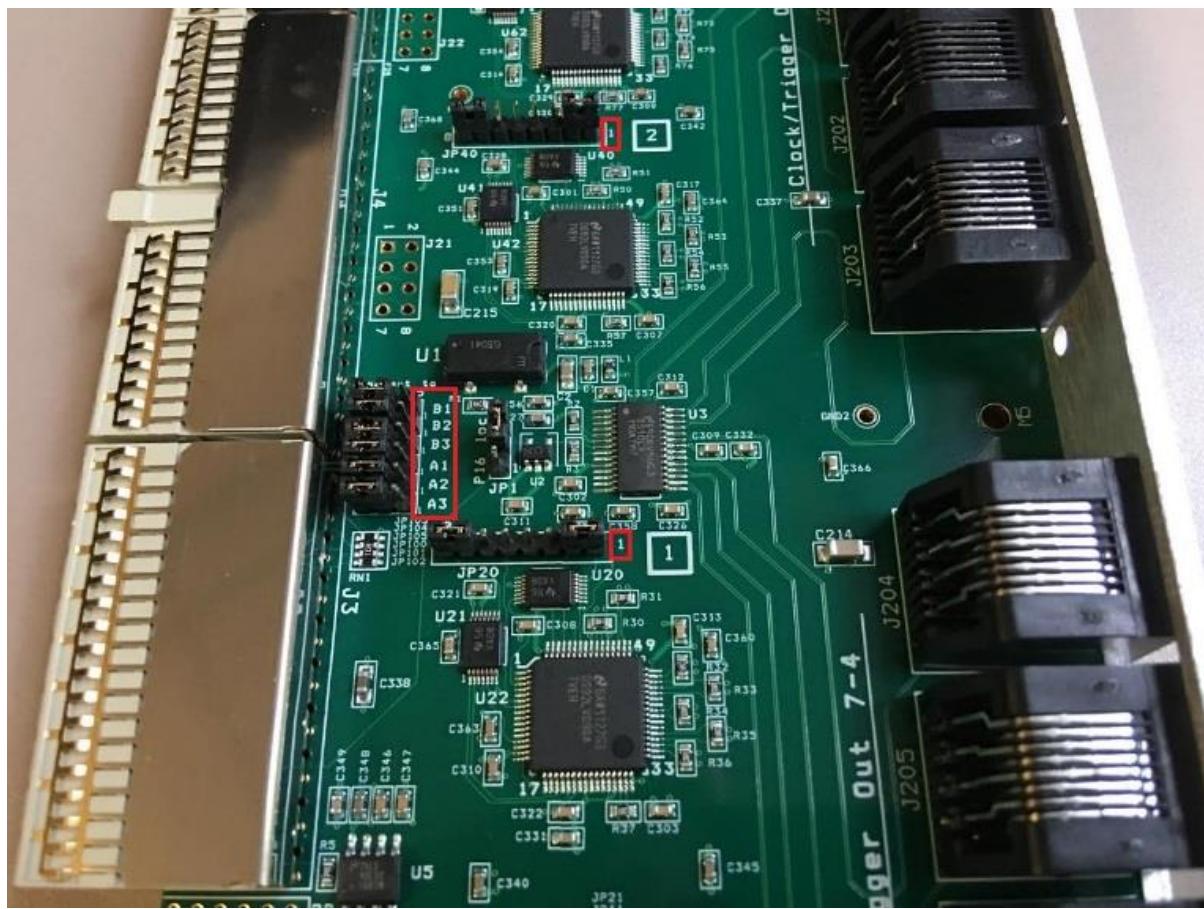
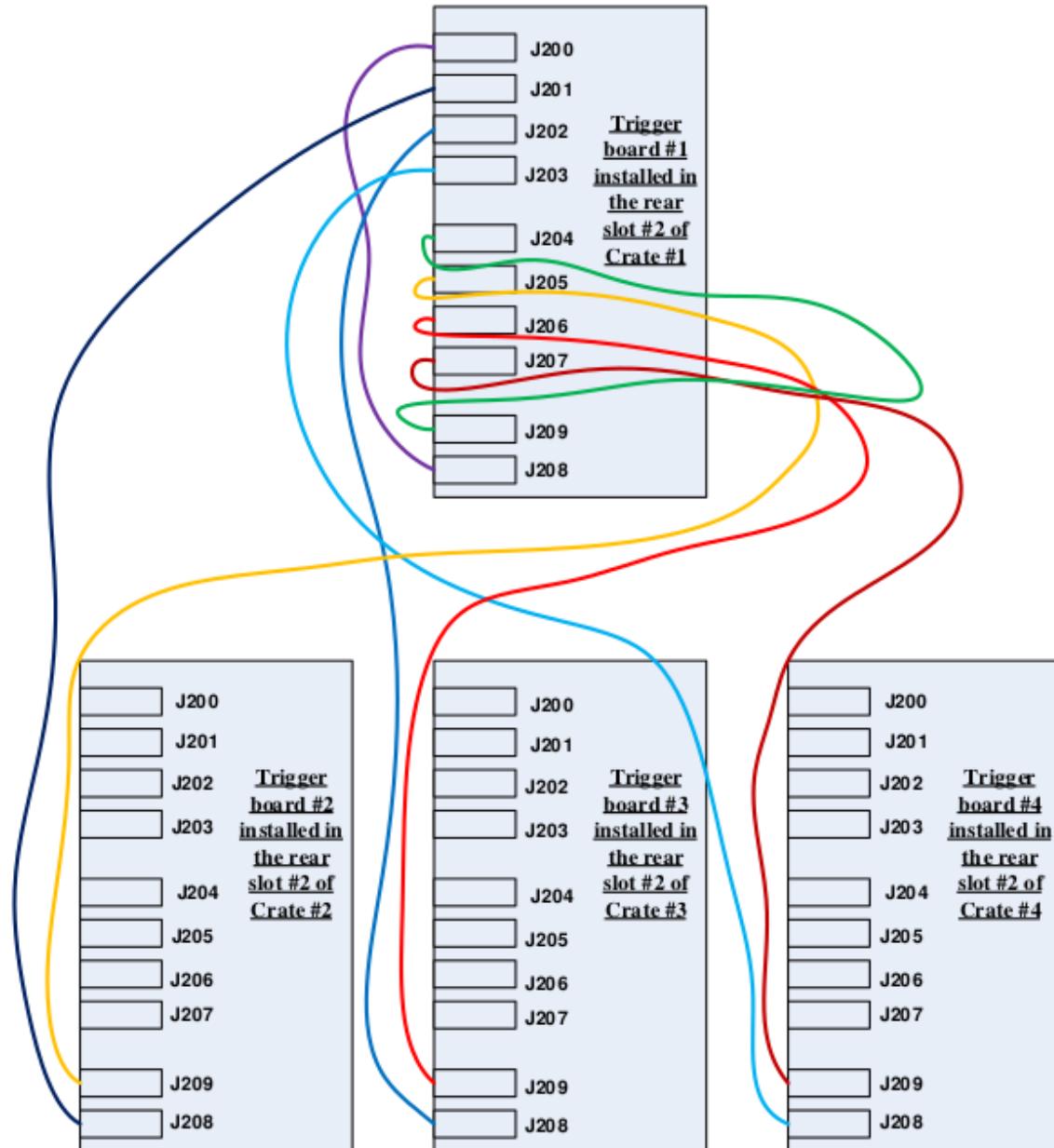


表 *Rear I/O Trigger Module #1's Jumper Settings* 显示了双机箱系统中 Pixie-16 背板 I/O 触发模块 1 的跳线设置。

JP1	Connect pins 1 and 2 for “P16”
JP20	Connect pins 2 and 3, 6 and 7
JP40	Connect pins 2 and 3, 6 and 7
JP60	Connect pins 1 and 2, 7 and 8
JP21	Connect pins 2 and 3
JP41	Connect pins 2 and 3
JP61	Connect pins 1 and 2
JP100	Connect pins 2 and 3 (connect to J4)
JP101	Connect pins 2 and 3 (connect to J4)
JP102	Connect pins 2 and 3 (connect to J4)
JP103	Connect pins 2 and 3 (connect to J4)
JP104	Connect pins 2 and 3 (connect to J4)
JP105	Connect pins 2 and 3 (connect to J4)

触发模块 2 安装在机箱 2 的后槽中。表 *Rear I/O Trigger Module #2's Jumper Settings* 显示了双机箱系统中 Pixie-16 背板 I/O 触发模块 2 的跳线设置。

JP1	Connect pins 2 and 3 for “loc”
JP20	Connect pins 1 and 2, 7 and 8
JP40	Connect pins 1 and 2, 7 and 8
JP60	Connect pins 2 and 3, 6 and 7
JP21	Don't connect any pin
JP41	Don't connect any pin
JP61	Connect pins 1 and 2
JP100	Connect pins 2 and 3 (connect to J4)
JP101	Connect pins 2 and 3 (connect to J4)
JP102	Connect pins 2 and 3 (connect to J4)
JP103	Connect pins 2 and 3 (connect to J4)
JP104	Connect pins 2 and 3 (connect to J4)
JP105	Connect pins 2 and 3 (connect to J4)



请注意，如果总共有四个机箱，安装在这四个机箱中的四个 Pixie-16 背板 I/O 触发模块之间的电缆连接应遵循图中所示的连接方法。对于 Pixie-16 背板 I/O 触发模块上的跳线设置，触发模块 1 和 2 应分别使用与双机箱系统触发模块 1 和 2 相同的跳线设置，而触发模块 3 和 4 应使用与触发模块 2 相同的跳线设置。

升级 CPLD

当您在使用更新、更强大的计算机时遇到初始化 (boot) Pixie-16 模块的问题。

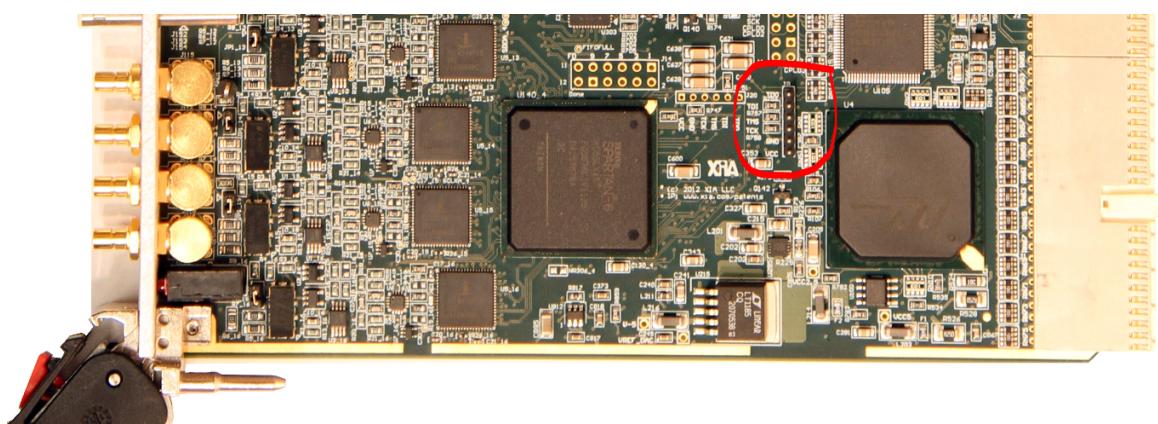
根本原因是 XIA 通过 API 软件延迟的方式来确保在将固件比特流下载到 Pixie-16 模块时出现适当的延迟。

如何使用更好、或者多核的更好的计算机，这样的延迟在软件中无法保证。因此，用户需要对 CPLD 固件进行更新，以确保使用硬件方法满足此类延迟。

请注意，此 CPLD 代码应用于更新所有 Pixie-16 模块，100 MHz，250 MHz 或 500 MHz。

14.1 6 pin JTAG connector

You should now try to locate the 6-pin vertical Pixie-16 CPLD JTAG header J8 (circled in red in the screenshot below).



The next step is to identify each of the 6 pins of the Pixie-16 CPLD JTAG header J8. When facing the top side of the Pixie-16 board, the 6 pins of the J8 connector can be identified as follows from top to bottom: **TDO**, **TDI**, **TMS**, **TCK**, **GND**, and **VCC**. It is very important to match these pins with the colored cables from the green PCB connector that is plugged into the Xilinx Platform USB programmer.

One issue here is that the Pixie-16 CPLD JTAG header J8 has 2 mm pin spacing, and the colored cables from the green PCB connector of the Xilinx programmer might have too large header to be plugged into the J8 pins directly. In that case, an adapter might be needed to convert those colored cable headers to 2 mm pin spacing first.

14.2 update the CPLD

Turn on your Pixie16 crate and the LED on the Xilinx Platform Cable USB should turn green. If the LED does not turn green check your cable connection. The grey INIT cable is purposely left unconnected. Your start menu should now include the folder Xilinx ISE Design Suite 14.7. Run the 32 bit or 64 bit version of “iMPACT”. Click on **File->Initialize Chain** and the program should find the **xc2c256 CPLD**. Click on “**No**” in the window “**Auto Assign Configuration Files Query**”, then click on “**Cancel**”. Left click on the Xilinx part in the window once, then right click and select “**Assign New Configuration Files**” and **select the new “jed” file**. Left click on the Xilinx part, right click and then click on “**Program**”. A device programming properties window appears. Ensure that only “**Verify**” and “**Erase before Programming**” are checked, then click on “**OK**”. The program should now say “**program succeeded**” in blue at the bottom of the window. The CPLD is now reprogrammed and the Pixie16 is ready to be tested.

I would suggest testing out your first board before reprogramming the CPLDs on all of your boards. Once you are in the batch mode of reprogramming and you have the cable connected (the USB programmer LED is green), you only need to click on “**Program**” to reprogram CPLD on your next Pixie16.

CHAPTER 15

应用案例

本章节介绍一些实验应用案例、测试结果等。

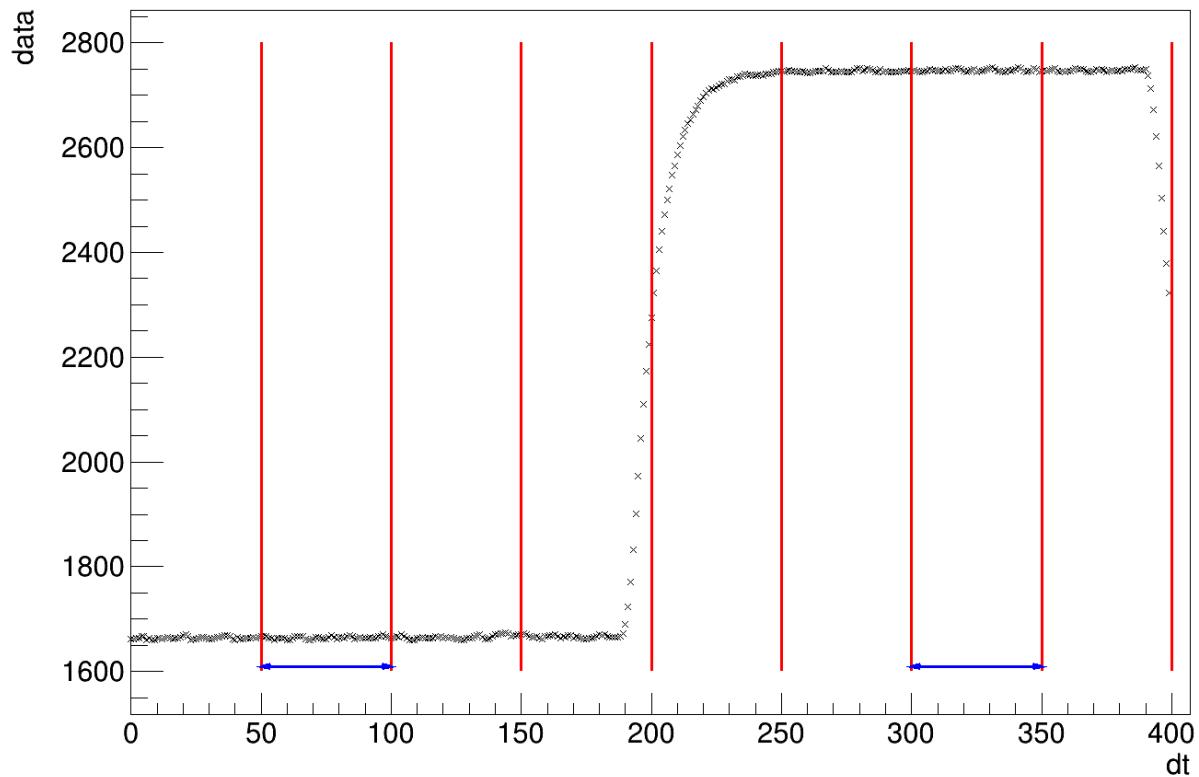
CHAPTER 16

100M 模块测量飞行时间

飞行时间的测量是核物理实验中的基本需求，在数字化获取系统中，100M 模块的本征分辨约为 100-200 ps，250M 模块本征分辨约为 40ps，500M 模块本真分辨约为 20ps。当采样率不足以直接测量飞行时间时，需要先将 TOF 信息转为 TAC 信号，通过 TAC 的幅度来表示时间。下面将展示如何通过 100M 模块来实现高精度的飞行时间测量。

16.1 TAC 信号特点

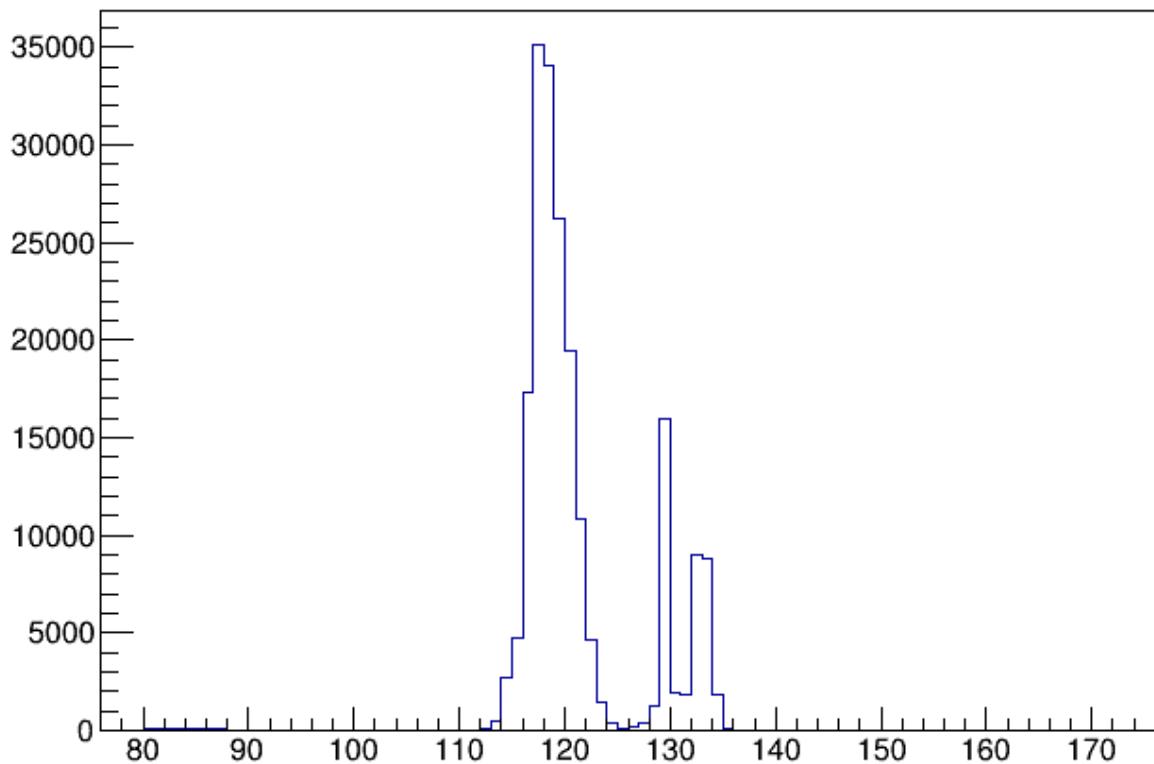
- TOF 信号经过 TAC 转为下图所示宽度 2us 的方波，其高度表示时间
- XIA slow filter 计算能量适用于单指数下降波形，该 TAC 信号无法通过梯形算法给出准确的高度值
- 因此这里采用 QDC 方法，trigger 前 2us 为 QDC 积分起点，每 500 ns 为一个积分时间窗
 - pre-trigger 2 us, all QDCs gate width 0.5 us
- 采用蓝色箭头所示两个积分窗的积分数值差作为飞行时间
 - $TOF = QDC[6] - QDC[1]$ ($QDC[0], QDC[1], \dots, QDC[7]$)
- 以上参数为我们测量的一组参数，该参数能够得到与 CAEN V1290 相当的时间分辨。该组参数避开了波形的上升段以及采样点 150 附近的反射峰



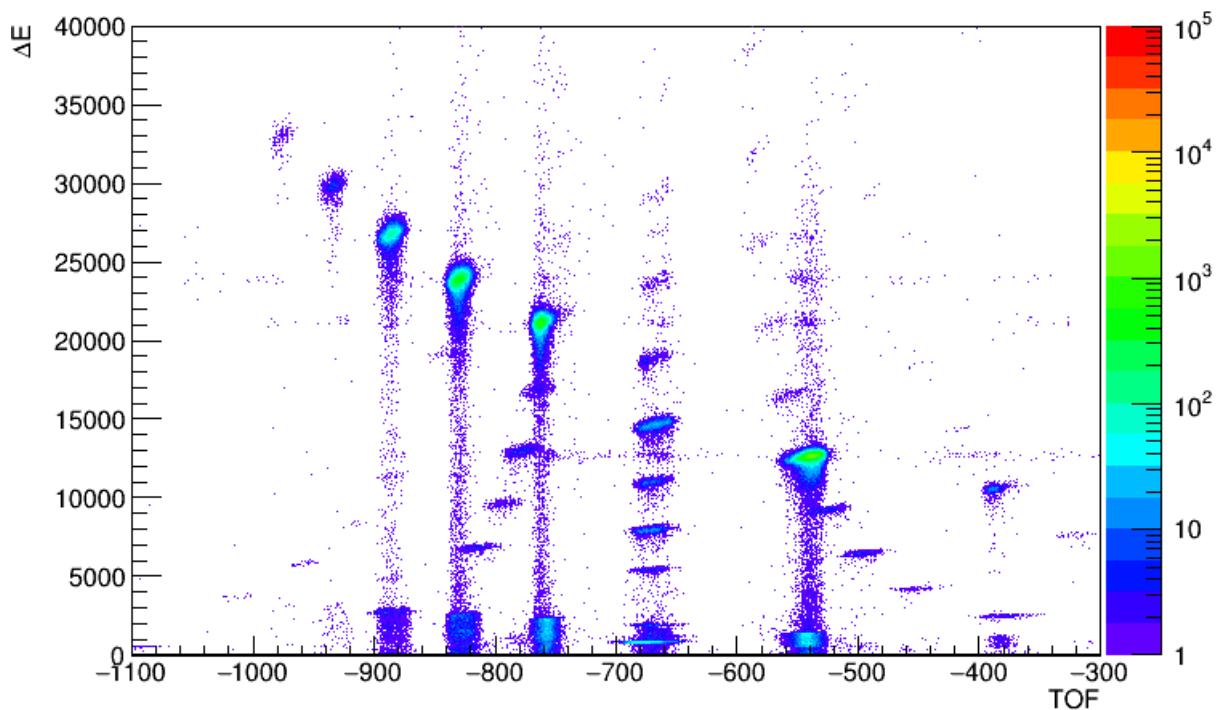
16.2 实验测量结果 (2017 RIBLL)

下图为 TAC 信号与硅探测器的时间差 (10 ns/channel)，我们可以发现 TAC 信号输出存在一个延迟，而这个延迟与我们选择的 TAC 量程有关，在事件重构时需要注意该时间差

tdc[1][3]-tdc[1][8] {qdc[1][3][1]>0}



下图为实验测量的 TOF-DeltaE 谱，飞行时间来源于两个薄塑料闪烁体探测器，DeltaE 来于硅探测器。



时间分辨

17.1 脉冲产生装置

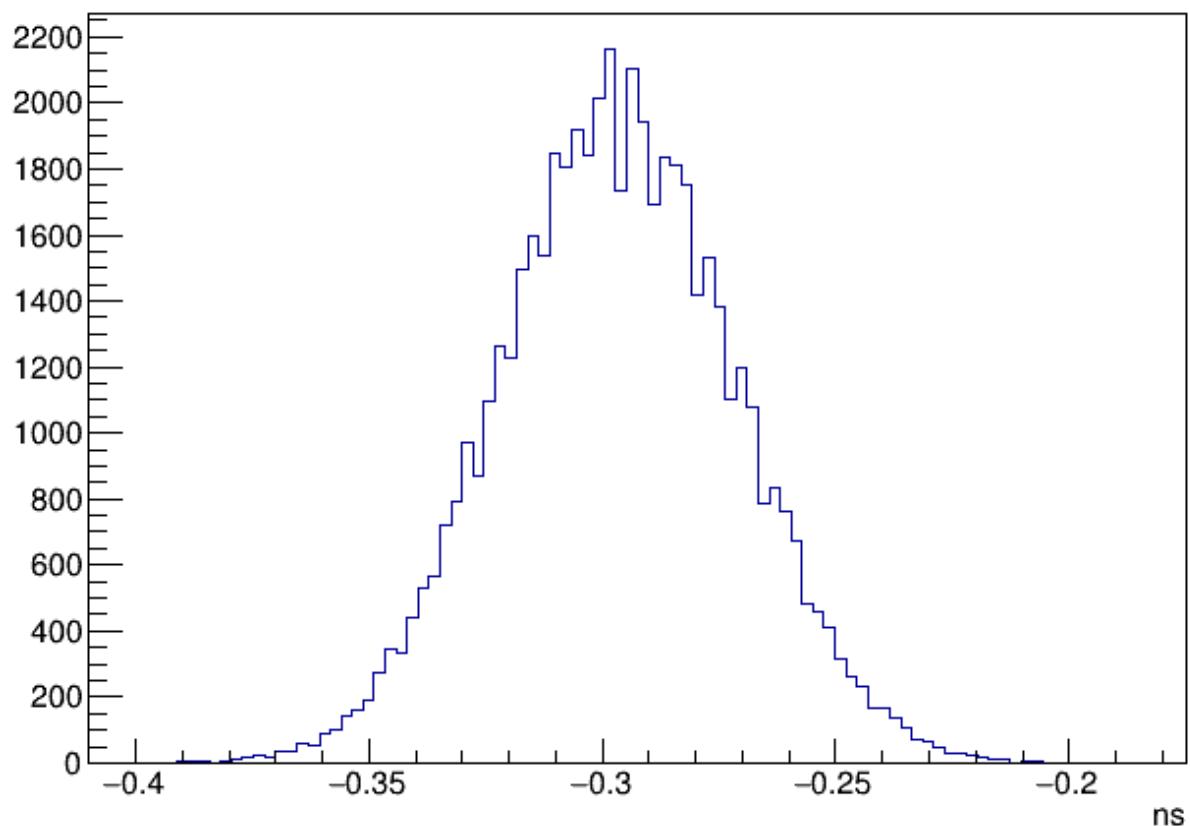
- Model PB-5
 - <https://www.berkeleynucleonics.com/model-pb-5>

17.2 100M 模块

- Width: 100ns 设置为最小值, 平台时间
- Rise time: 0.1 us 上升时间
- Fall time: 100 us 衰减时间常数
- Rate: 1.0 kHz 频率
- Delay: 250ns 任意设置
- Ampl: 10.0 V 根据需要设置
- Polarity: Pos 根据需要设置
- Pulse Top: Flat
- Atten: 1X 根据需要设置
- PB-5 Pulse: ON 开启脉冲输出

脉冲发生器产生信号经过分路器一分为二。

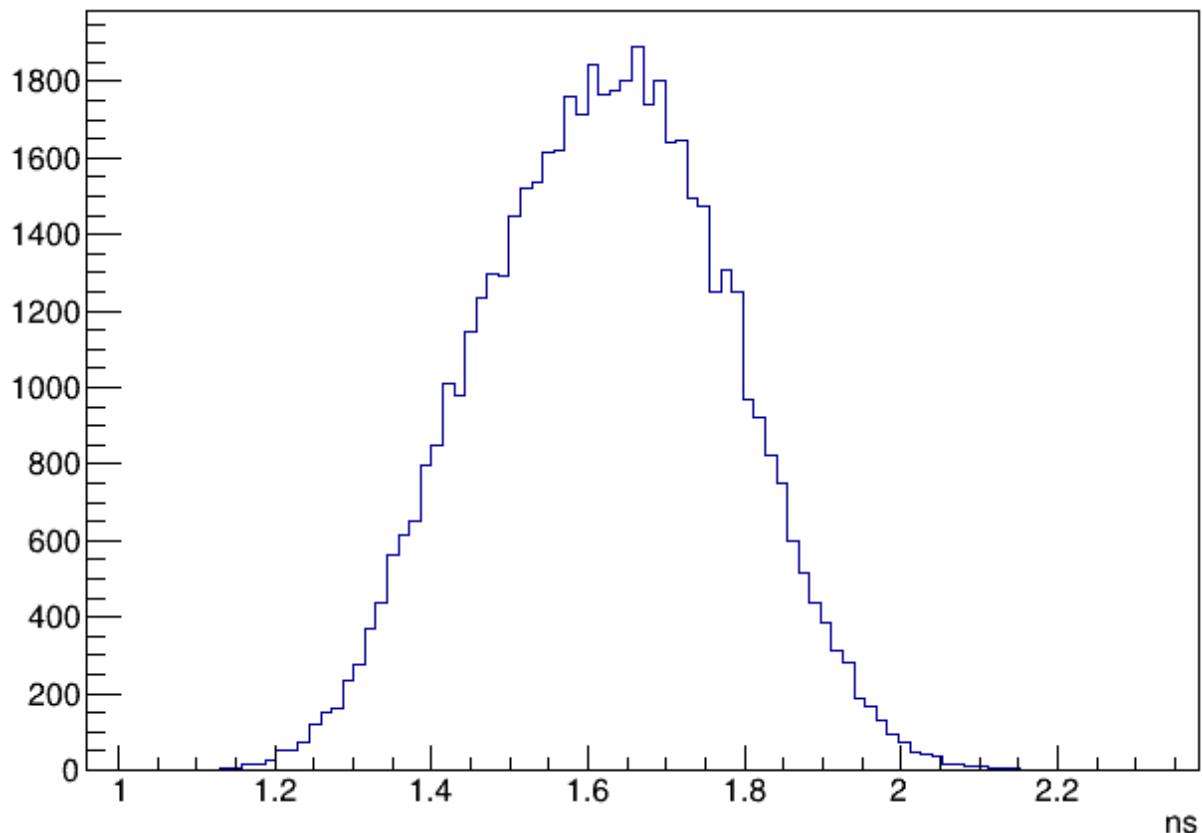
17.2.1 2 channel in one module



本征分辨约为 60 ps

17.2.2 2 channel in one crate

17.2.3 2 channel in different crate



本征分辨约为 360 ps

17.3 250M 模块

17.3.1 2 channel in one module

17.3.2 2 channel in one crate

17.3.3 2 channel in different crate

17.4 100M & 250M 模块

17.4.1 2 channel in one crate

17.4.2 2 channel in different crate

CHAPTER 18

推荐参数

本页参数仅供参考

18.1 HPGe

18.1.1 100M

- **fast filter**
 - FL: 0.1
 - GF: 0.1
- **slow filter**
 - SL: 5.04
 - SG: 1.2/1.6
 - Range: 3
 - TAU: 以实际测量为准
- **cfd filter**
 - dealy: 0.02
 - scale: 5

18.1.2 250M

- **fast filter**
 - FL: 0.13
 - GF: 0.13
- **slow filter**
 - SL: 5.04
 - SG: 1.2

- Range: 3
- TAU: 以实际测量为准

- **cfd filter**

- dealy: 0.08
 - scale: 0
-

18.2 BGO

18.2.1 100M

- **fast filter**

- FL: 0.06
- GF: 0.0

- **slow filter**

- SL:
- SG:
- Range: 1
- TAU: 以实际测量为准

- **cfd filter**

- dealy: 0.08
 - scale: 0
-

18.3 Si

18.3.1 100M

- **fast filter**

- FL: 0.1
- GF: 0.0

- **slow filter**

- SL: 3.04
 - SG: 0.24
 - Range: 2
 - TAU: Based on actual measurements
-

18.4 LaBr3

18.4.1 250M

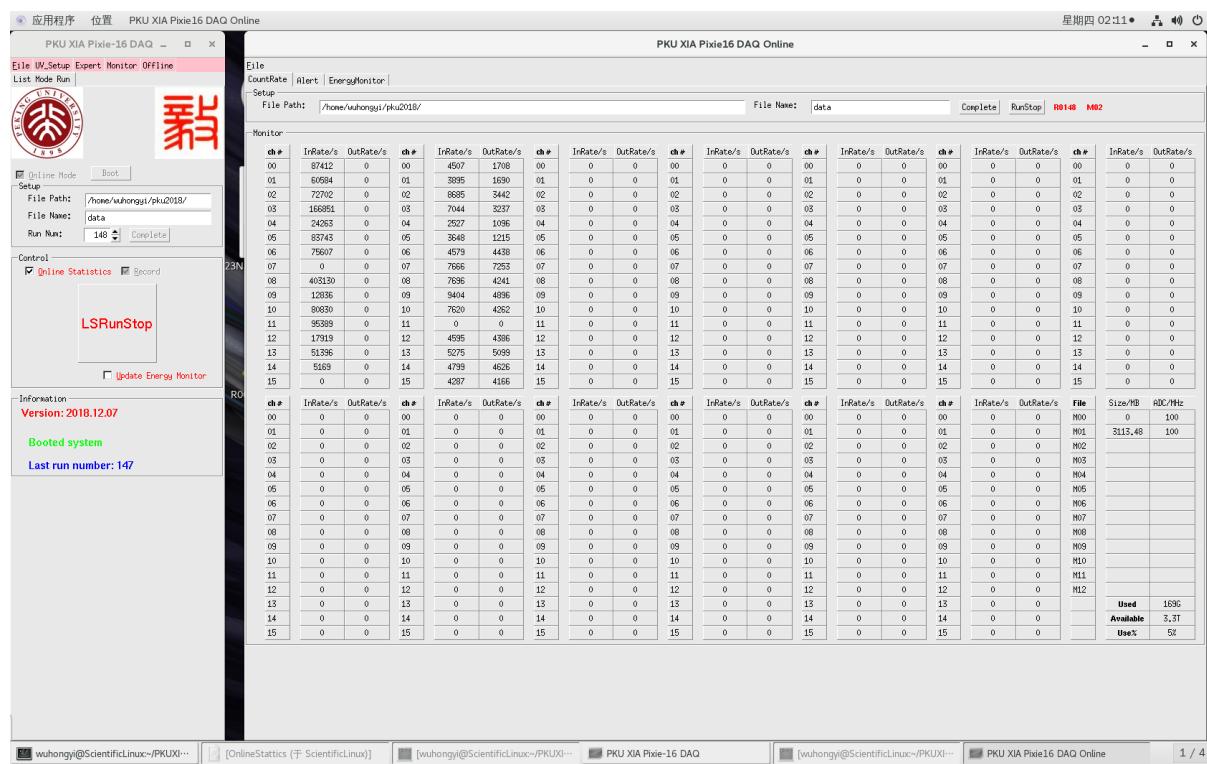
- **fast filter**
 - FL: 0.08
 - GF: 0.016
- **slow filter**
 - SL: 0.144/0.128
 - SG: 0.048
 - Range: 1
 - TAU: 以实际测量为准 (0.023)
- **cfd filter**
 - dealy: 0.024
 - scale: 0
- **Trace**
 - Delay: 1.496
 - Length: 3.504

CHAPTER 19

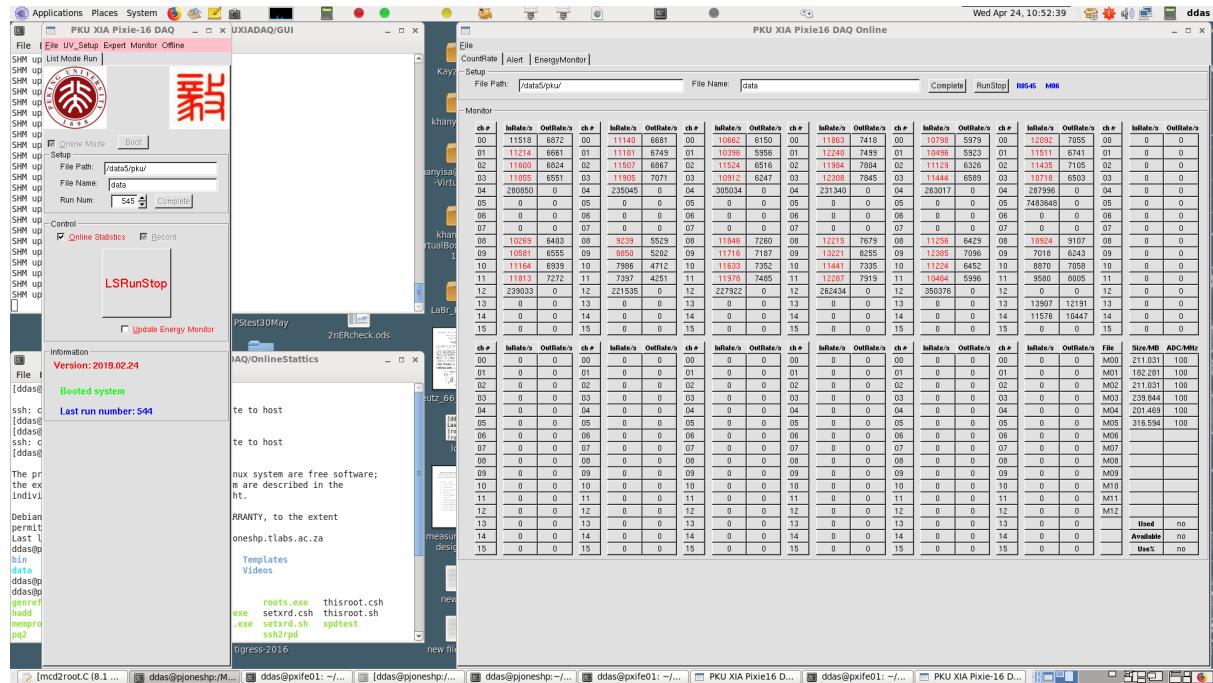
在束伽马谱学

19.1 实验控制界面

实验控制界面如下图所示

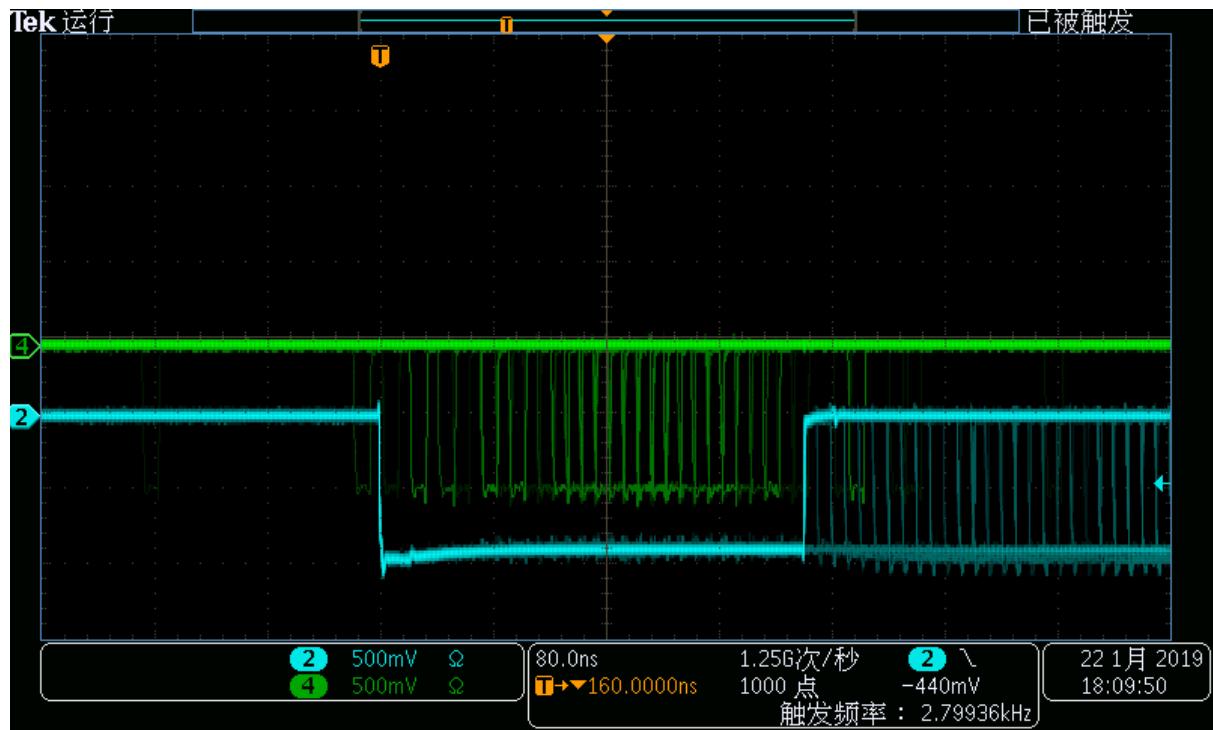


5 HPGe + 2 Clovers + 2LEPS



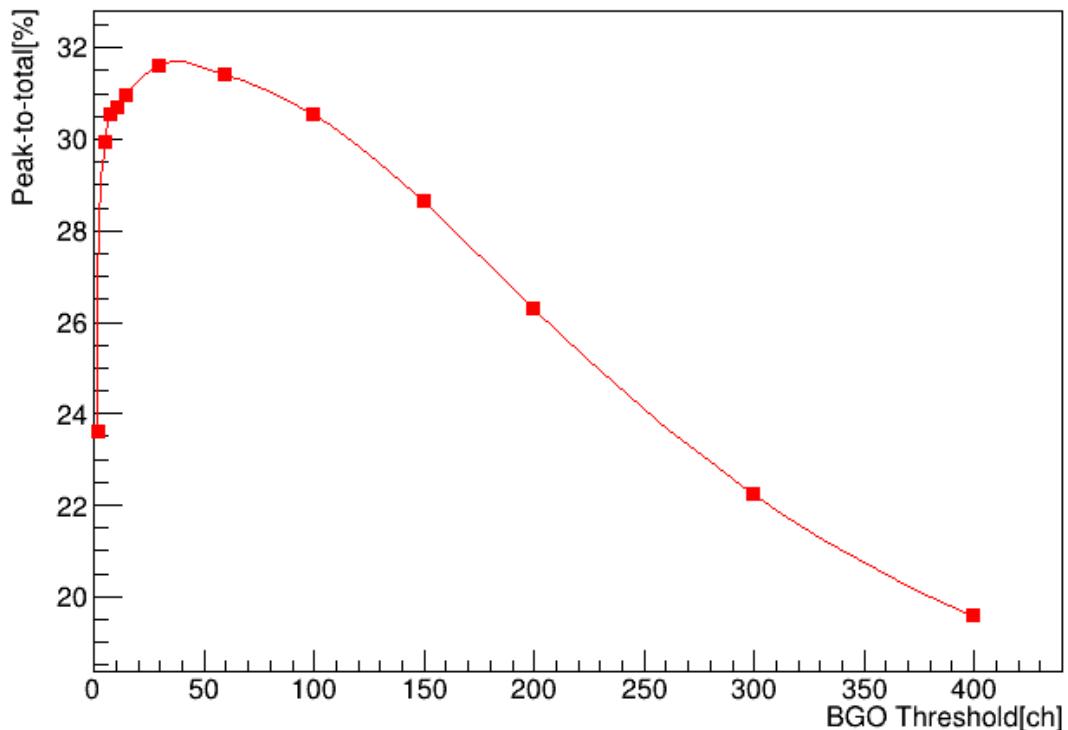
11 Clovers + 4 channel LEPS + 2 LEPS

19.2 BGO 反康门宽



19.3 峰总比

Peak-to-total without BGO: 13.89%



CHAPTER 20

采购推荐

特别声明：北京大学实验核物理组与 XIA LLC 当前中国区代理北京华恒鑫达科技发展有限公司不存在任何形式的合作。任何打着北京大学实验核物理组名义的宣传都是非法的，请大家谨防上当受骗。关于产品硬件的性能、固件能力等信息，以本说明书为准。PKUXIADAQ 长期进行版本更新，请大家时常关注是否有新版本发布。如有任何疑问，请与吴鸿毅联系 (wuhongyi@qq.com)



XIA LLC 推出的 Pixie16 系列采集卡，拥有 100M 12/14 bit、250M 12/14/16 bit、500M 12/14 bit 采样率/垂直精度的采集卡，PKUXIADAQ 支持以上任意采样率/垂直精度卡的混合使用。

20.1 采集卡

- **100M 14bit**
 - HPGe/Clover
 - BGO
 - Si
 - CsI
 - gas det
 - ...
- **250M 14/16 bit、500 M 14 bit**
 - LaBr3
 - Plastic scintillator
 - Liquid scintillator
 - BaF2
 - ...

对于快时间信号的测量，推荐优先购买 500 M 14 bit，其次是购买 250M 16bit/14 bit

20.2 机箱及配套

Pixie16 需要 XIA 定制的专用机箱。需要使用这个机箱的原因是通用的机箱电流不足以支持那么多插件的运行。另外，此机箱背板为定制背板，因此能实现及其复杂、高效的逻辑运算。

电脑控制器可采用光纤控制器或者同轴电缆控制器，这两个的区别在于，同轴电缆控制器线缆长度不超过 7 米（千万不要买 1 米或者 3 米的，机箱与电脑的摆放位置会被极大约束），也就是说采集电脑必须紧靠着机箱，而光纤长度选择范围较大，电脑可远离机箱。选择哪个方案区别仅在这个长度上，对数据传输效率、软件控制上没有本质的区别。但是光纤跟同轴电缆的差别在于光纤实现了光电隔离，获取系统跟控制电脑的地线是隔离的，而同轴电缆获取系统跟控制电脑地线是在一起的，这个有没有影响，取决于具体实验对噪声的灵敏情况，实验终端的地线情况等另外，同轴电缆的比较便宜。

另外，控制器也可以选择嵌入式电脑，即一个小电脑嵌入在机箱最左侧，采集卡的数据直接传输到该电脑。采用该方案需要注意的几点是，PKUXIADAQ 仅支持 LINUX 操作系统，因此选择该方案需要购买 linux 操作系统的嵌入式电脑。另外，嵌入式电脑的硬盘容量较小，一般来说，需要配合外部的磁盘阵列（或者服务器）来使用才能满足核物理实验的数据存储需求（如果仅仅是简单测试那就不需要磁盘阵列，直接外挂一个 USB3.0 的移动硬盘即可），该方案适合于固定实验终端的配套获取使用，例如 MSU/NSCL 的做法，将数据通过以太网实时传到服务器上。这个方案需要一定的技术考虑，主要是需要一个网络架构来保证数据的传输不会影响数据计数率，也就是数据传输引起的死时间不会影响系统性能，这个需要比较专业的网络技术部门来提供技术支持。

另外，关于电脑控制器是采用 PCI 还是 PCIe 的，这个取决于电脑主板上的插槽，购买前请先考虑好要使用的控制电脑。但是现在采购的电脑，如果没有特别说明，默认采购到的将是只有 PCIe 接口的电脑。如果是 PCIe 的，还需要注意插槽的宽度，插槽分为 x1, x4, x8, x16，而且是想下兼容的，例如，一个 x4 宽度的控制器，可以插入 x4, x8, x16 的插槽，但是无法插入 x1 的插槽。

这里给出几个选项供大家参考

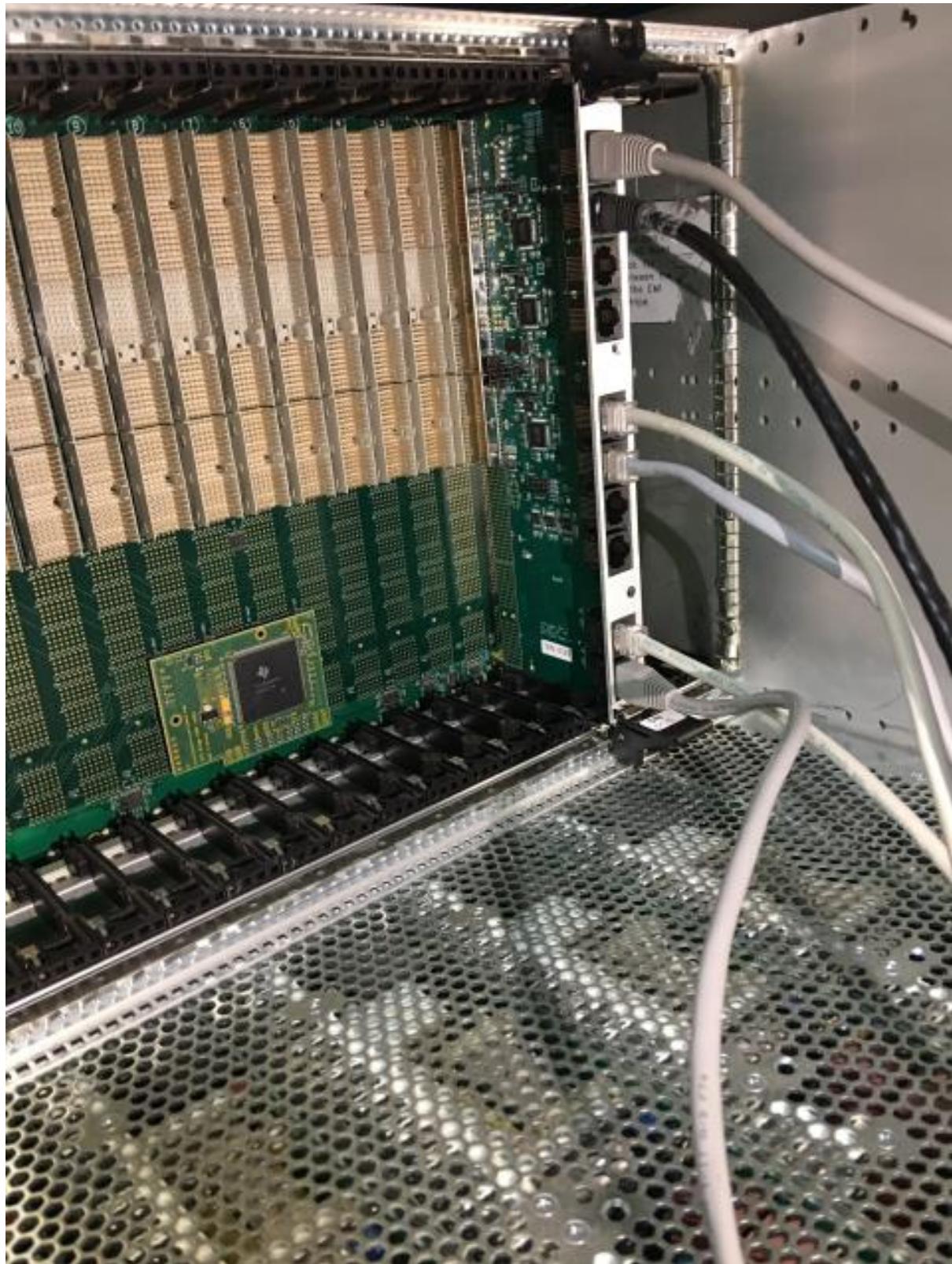
电脑端 PCIe 控制器

- PXI-8360/PCIe-8361 机箱端使用 PXI8360 卡，电脑端使用 PCIe8361 x1 卡，同轴电缆链接，记得指定要 7 米的电缆。
- PXI-8360/PCIe-8362 机箱端使用 PXI8360 卡，电脑端使用 PCIe8362 x1 卡，同轴电缆链接，记得指定要 7 米的电缆。

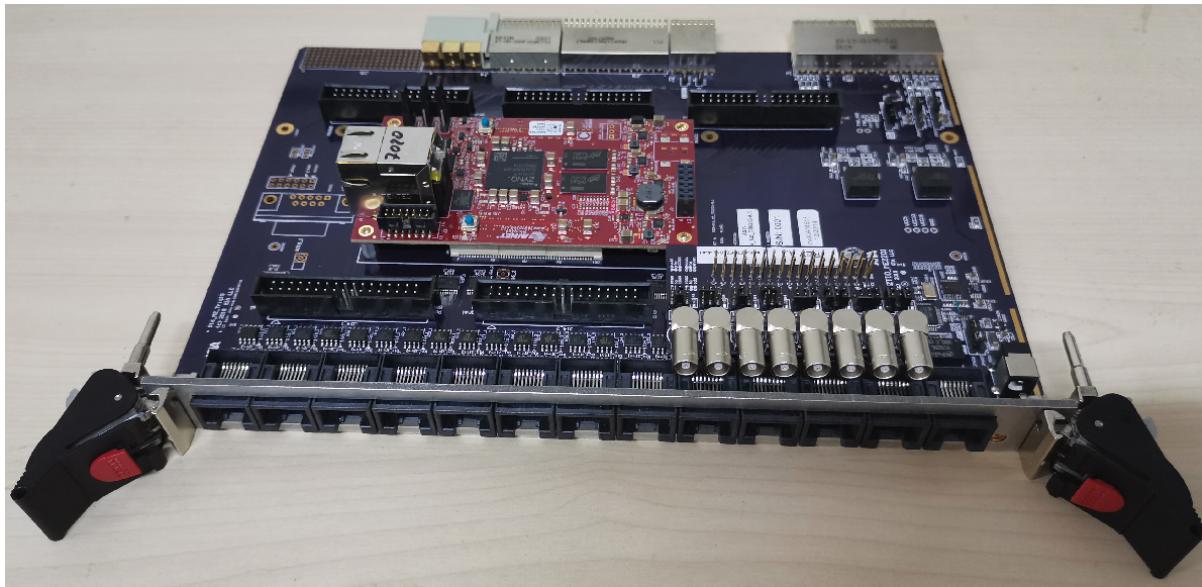
电脑端 PCI 控制器

- PXI-8368/PCI-8366 机箱端使用 PXI8368 卡，电脑端使用 PCI8366 卡，光纤链接，光纤长度可选择 10/30/200 米。

如果需要多个机箱的同步使用，那就需要给每个机箱配套一个时间同步模块，该模块插在机箱背板，通过网线来实现不同机箱中的时间同步、触发信号共享等功能。



20.3 逻辑模块



MZTIO + MZTIO-MEZZ01 需要复杂实验触发逻辑的用户需要采购该配套模块。

PKU 固件中，通过前面板的网口将多重性选择的结果输出，然后通过外部的可编程逻辑插件来实现逻辑运算，最后将外部逻辑通过网口送到采集卡。外部逻辑信号与每个通道的自触发逻辑进行与运算作为有效触发信号。这样就能避免记录大量无用的数据。

外部可编程逻辑插件推荐使用 XIA 的 MZTIO 触发逻辑模块（该模块是我们与 XIA LLC 讨论开发的，相比 CAEN 的类似功能模块使用上更容易）。为了方便通过示波器来观测触发逻辑与采集门之间的关系、触发逻辑与 veto 的关系等，实现在束可视化远程调节采集逻辑，PKU 固件将通过机箱背板把监视信号传输到 MZTIO 模块，然后推荐使用配套的 8 通道转接板，监视信号通过 LEMO 连接器连接到示波器。当然，此 8 通道转接板也可作为信号输入使用，可通过该转接板将外部信号输入 MZTIO 模块参与触发逻辑的运算。

因此，采用该模块，可以取代传统电子学中的逻辑插件，并且具有可调节范围更广，逻辑缺陷少的优点。

20.4 34pin 到 16SMB 转接线

生产中... 预计 10 月中旬能够拿到产品。

由于核物理实验中双面硅微条路数在 200-300 路，规模较大的可达到 500-600 路。而硅信号的输出一般都是 34 pin 的连接器，Pixie16 采集卡的输入是 SMB 连接器，现阶段为 BNC 转 SMB 连接线，在硅探测器上使用起来及其不方便，因此必须采用一个合适的转接器来实现。

CHAPTER 21

开发者指南

本章节介绍 Pixie16 开发中使用的 Pixie-16 API 函数及获取程序的基本原理。
为用户提供基于我们获取程序开发的可能。

CHAPTER 22

XIA API

It from **Programmer's Manual Digital Gamma Finder (DGF) PIXIE-16 Version 1.40, October 2009**

```
// Configure modules for communication in PXI chassis
// Use this function to configure the Pixie-16 modules in the PXI chassis.
// NumModules is the total number of Pixie-16 modules installed in the system.
// →PXISlotMap is the pointer to an array that must have at least as many entries as
// →there are Pixie-16 modules in the chassis.
// PXISlotMap serves as a simple mapping of the logical module number and the
// →physical slot number that the modules reside in. The logical module number runs
// →from 0. For instance, in a system with 5 Pixie-16 modules, these 5 modules may
// →occupy slots 3 through 7. The user must fill PXISlotMap as follows: PXISlotMap =
// →{3, 4, 5, 6, 7 ...} since module number 0 resides in slot number 3, etc. To find
// →out in which slot a module is located, any piece of subsequent code can use the
// →expression PXISlotMap[ModNum], where ModNum is the logic module number.
// OfflineMode is used to indicate to the API whether the system is running in
// →OFFLINE mode (1) or ONLINE mode (0). OFFLINE mode is useful for situations where
// →no Pixie-16 modules are present but users can still test their calls to the API
// →functions in their application software.
// This function must be called as the first step in the boot process. It makes
// →the modules known to the system and "opens" each module for communication.
// The function relies on an initialization file (pxisys.ini) that contains
// →information about the Host PC's PCI buses, including the slot enumeration scheme.
// →XIA's software distribution normally puts this file under the same folder as
// →Pixie-16 software installation folder. However, the user has the flexibility of
// →putting it in other folders by simply changing the definition of the string
// →PCISysIniFile_Windows or PCISysIniFile_Linux in the header part of the file
// →pixie16sys.c, depending on which operating system is being used.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16InitSystem (
    unsigned short NumModules,      // total number of Pixie16 modules in the
    //system
    unsigned short *PXISlotMap,     // an array containing the PXI slot number for
    //each pixie16 module
    unsigned short OfflineMode ); // specify if the system is in offline mode
```

```
// Release user virtual addressees assigned to modules
// Use this function to release the user virtual addressees that are assigned to
// →Pixie-16 modules when these modules are initialized by function
// →Pixie16InitSystem. This function should be called before a user's application
// →exits.
```

(续上页)

```
// If ModNum is set to less than the total number of modules in the system, only
// the module specified by ModNum will be closed. But if ModNum is equal to the
// total number of modules in the system, e.g. there are 5 modules in the chassis
// and ModNum = 5, then all modules in the system will be closed altogether. Note
// that the modules are counted starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ExitSystem (
    unsigned short ModNum ); // module number
```

```
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadModuleInfo (
    unsigned short ModNum, // module number
    unsigned short *ModRev, // returned module revision
    unsigned int *ModSerNum, // returned module serial number
    unsigned short *ModADCBits, // returned module ADC bits
    unsigned short *ModADCMSPS ); // returned module ADC sampling rate
```

```
// Boot modules so that they can be set up for data taking
// Use this function to boot Pixie-16 modules so that they can be set up for data
// taking. The function downloads to the Pixie-16 modules the communication FPGA
// configurations, signal processing FPGA configurations, trigger FPGA
// configurations (Revision A modules only), executable code for the digital signal
// processor (DSP), and DSP parameters.
// The FPGA configurations consist of a fixed number of words depending on the
// hardware mounted on the modules; the DSP codes have a length which depends on
// the actual compiled code; and the set of DSP parameters always consists of 1280
// 32-bit words for each module. The host software has to make the names of those
// boot data files on the hard disk available to the boot function.
// If ModNum is set to be less than the total number of modules in the system,
// only the module specified by ModNum will be booted. But if ModNum is equal to
// the total number of modules in the system, e.g. there are 5 modules in the
// chassis and ModNum = 5, then all modules in the system will be booted.
// The boot pattern is a bit mask (shown below) indicating which on-board chip
// will be booted. Under normal circumstances, all on-board chips should be booted,
// i.e. the boot pattern would be 0x7F. For Rev-B, C, D modules, bit 1, i.e., "Boot
// trigger FPGA", will be ignored even if that bit is set to 1.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16BootModule (
    char *ComFPGAConfigFile, // name of communications FPGA configuration
    file
    char *SPFPGAConfigFile, // name of signal processing FPGA
    configuration file
    char *TrigFPGAConfigFile, // name of trigger FPGA configuration file
    char *DSPCodeFile, // name of executable code file for digital
    signal processor (DSP)
    char *DSPParFile, // name of DSP parameter file
    char *DSPVarFile, // name of DSP variable names file
    unsigned short ModNum, // pixie module number
    unsigned short BootPattern ); // boot pattern bit mask
```

```
// Acquire ADC traces in single or multiple modules
// Use this function to acquire ADC traces from Pixie-16 modules. Specify the
// module using ModNum. If ModNum is set to be less than the total number of
// modules in the system, only the module specified by ModNum will have its ADC
// traces acquired. But if ModNum is equal to the total number of modules in the
// system, then all modules in the system will have their ADC traces acquired.
// After the successful return of this function, the DSP's internal memory will be
// filled with ADC trace data. A user's application software should then call
// another function Pixie16ReadSglChanADCTrace to read the ADC trace data out to
// the host computer, channel by channel.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16AcquireADCTrace (
    unsigned short ModNum ); // module number
```

```

// Read ADC trace data from a channel in a module
// Use this function to read ADC trace data from a Pixie-16 module. Before calling
// this function, another function Pixie16AcquireADCTrace should be called to fill
// the DSP internal memory first. Also, the host code should allocate appropriate
// amount of memory to store the trace data. The ADC trace data length for each
// channel is 8192. Since the trace data are 16-bit unsigned integers (actually
// only the lower 14-bit contains real data due to the on-board 14-bit ADC), two
// consecutive 16-bit words are packed into one 32-bit word in the DSP internal
// memory. So for each channel, 4096 32-bit words are read out first from the DSP,
// and then each 32-bit word is unpacked to form two 16-bit words.
// Specify the module using ModNum and the channel on the module using ChanNum.
// Note that both the modules and channels are counted starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadSglChanADCTrace (
    unsigned short *Trace_Buffer, // trace data
    unsigned int Trace_Length,   // trace length
    unsigned short ModNum,      // module number
    unsigned short ChanNum );   // channel number

```

```

// Transfer data between host and DSP internal memory
// Use this function to directly transfer data between the host and the DSP
// internal memory of a Pixie-16 module. The DSP internal memory is split into two
// blocks with address range 0x40000 to 0x4FFFF for the first block and address
// range 0x50000 to 0x5FFFF for the second block. Within the first block, address
// range 0x40000 to 0x49FFF is reserved for program memory and shouldn't be
// accessed directly by the host. Address range 0x4A000 to 0x4A4FF is used by the
// DSP I/O parameters which are stored in the configuration files (.set files) in
// the host. Within this range, 0x4A000 to 0x4A33F can be both read and written,
// but 0x4A340 to 0x4A4FF can only be read but not written. The remaining address
// range (0x4A500 to 4FFFF) in the first block and the entire second block (0x50000
// to 0x5FFFF) should only be read but not written by the host. Use Direction = 1
// for read and Direction = 0 for write.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16IMbufferIO (
    unsigned int *Buffer,           // buffer data
    unsigned int NumWords,          // number of buffer words to read or write
    unsigned int Address,           // buffer address
    unsigned short Direction,       // I/O direction
    unsigned short ModNum );        // module number

```

```

// Transfer data between host and DSP external memory
// Use this function to directly read data from or write data to the on-board
// external memory of a Pixie-16 module. The valid memory address is from 0x0 to
// 0x7FFFF (32-bit wide). Use Direction = 1 for read and Direction = 0 for write.
// The external memory is used to store the histogram data accumulated for each of
// the 16 channels of a Pixie-16 module. Each channel has a fixed histogram length
// of 32768 words (32-bit wide), and the placement of the histogram data in the
// memory is in the same order of the channel number, i.e. channel 0 occupies
// memory address 0x0 to 0x7FFF, channel 1 occupies 0x8000 to 0xFFFF, and so on.
// NOTE: another function Pixie16ReadHistogramFromModule can also be used to read
// out the histograms except that it needs to be called channel by channel.
// In Rev-A modules, part of the external memory is also used to store the list
// mode data in ping-pong buffering mode. This function can be used to read list
// mode data from the buffers.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16EMbufferIO (
    unsigned int *Buffer,           // buffer data
    unsigned int NumWords,          // number of buffer words to read or write
    unsigned int Address,           // buffer address
    unsigned short Direction,       // I/O direction
    unsigned short ModNum );        // module number

```

```

// Start a list mode data acquisition run
// Use this function to start a list mode data acquisition run in Pixie-16 modules.
// List mode run is used to collect data on an event-by-event basis, gathering
// energies, timestamps, pulse shape analysis values, and waveforms, for each event.
// Runs will continue until a preset number of events are reached or the user
// terminates the run by calling function Pixie16EndRun. Once the run is progress, 145
// if the run is set to terminate after a given number of events have been
// accumulated, another function, Pixie16CheckRunStatus, should be called to check
// if the run has finished. To start the data acquisition this function has to be
// called for every Pixie-16 module in the system. If all modules are to run

```

(续上页)

```

// Use mode=NEW_RUN (=1) to erase histograms and statistics information before_
// launching the new run. Note that this will cause a start up delay of up to 1_
// millisecond. Use mode=RESUME_RUN (=0) to resume an earlier run. This mode has a_
// start up delay of only a few microseconds.
// For Rev-A modules, currently there are 4 list mode run types supported. They_
// are 0x100 (general purpose run), 0x101 (without waveforms), 0x102 (without_
// auxiliary data) and 0x103 (energy and timestamp only).
// For Rev-B, C, D modules, there are only one list mode run type supported, that_
// is, 0x100. However, different output data options can be chosen by enabling or_
// disabling different CHANCSRA bits.
// Histograms and statistics data are updated incrementally from run to run_
// provided RESUME_RUN mode is used.
// ModNum is the module number which starts counting at 0. If ModNum is set to be_
// less than the total number of modules in the system, only the module specified_
// by ModNum will have its list mode run started. But if ModNum is set to equal to_
// the total number of modules in the system, then all modules in the system will_
// have their runs started together.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16StartListModeRun (
    unsigned short ModNum,           // module number
    unsigned short RunType,         // run type
    unsigned short mode );          // run mode

```

```

// Start a MCA histogram mode data acquisition run
// Use this function to begin a data acquisition run that accumulates energy_
// histograms, one for each channel. It launches a data acquisition run in which_
// only energy information is preserved and histogrammed locally to each channel.
// Call this function for each Pixie-16 module in the system. The last module_
// addressed will allow the actual data acquisition to begin. Histogram run can be_
// self-terminating when the elapsed run time exceeds the preset run time, or the_
// user can prematurely terminate the run by calling Pixie16EndRun. On completion,_
// final histogram and statistics data will be available.
// Use mode=NEW_RUN (=1) to erase histograms and statistics information before_
// launching the new run. Use mode=RESUME_RUN (=0) to resume an earlier run.
// ModNum is the module number which starts counting at 0. If ModNum is set to be_
// less than the total number of modules in the system, only the module specified_
// by ModNum will have its histogram run started. But if ModNum is set to be equal_
// to the total number of modules in the system, then all modules in the system_
// will have their runs started together.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16StartHistogramRun (
    unsigned short ModNum,           // module number
    unsigned short mode );          // run mode

```

```

// Check status of a data acquisition run
// Use this function to check the run status of a Pixie-16 module while a list_
// mode data acquisition run is in progress. If the run is still in progress_
// continue polling.
// If the return code of this function indicates the run has finished, there might_
// still be some data in the external memory (Rev-A modules) or external FIFO (Rev-
// B, C, D modules) that need to be read out to the host. In addition, final run_
// statistics and histogram data are available for reading out too.
// In MCA histogram run mode, this function can also be called to check if the run_
// is still in progress even though it is normally self-terminating.
// ModNum is the module number which starts counting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16CheckRunStatus (
    unsigned short ModNum );        // Pixie module number

```

```

// Stop a data acquisition run
// Use this function to end a histogram run, or to force the end of a list mode_
// run. In a multi-module system, if all modules are running synchronously, only_
// one module needs to be addressed this way. It will immediately stop the run in_
// all other module in the system.

```

(下页继续)

(续上页)

```
// ModNum is the module number which starts counting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16EndRun (
    unsigned short ModNum ); // Pixie module number
```

```
// Compute input count rate
// Use this function to calculate the input count rate on one channel of a Pixie-
// →16 module. This function does not communicate with Pixie-16 modules. Before
// →calling this function, another function, Pixie16ReadStatisticsFromModule, should
// →be called to read statistics data from the module first.
// *Statistics is a pointer to an array whose size is exactly 448 unsigned integer
// →words (32-bit). The *Statistics array is filled with data from a Pixie-16 module
// →after calling function Pixie16ReadStatisticsFromModule. ModNum is the module
// →number which starts counting at 0. ChanNum is the channel number which starts
// →counting at 0.
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeInputCountRate (
    unsigned int *Statistics,
    unsigned short ModNum,
    unsigned short ChanNum );
```

```
// Compute output count rate of a channel
// Use this function to calculate the output count rate on one channel of a Pixie-
// →16 module. This function does not communicate with Pixie-16 modules. Before
// →calling this function, another function, Pixie16ReadStatisticsFromModule, should
// →be called to read statistics data from the module first.
// *Statistics is a pointer to an array whose size is exactly 448 unsigned integer
// →words (32-bit). The *Statistics array is filled with data from a Pixie-16 module
// →after calling function Pixie16ReadStatisticsFromModule. ModNum is the module
// →number which starts counting at 0. ChanNum is the channel number which starts
// →counting at 0.
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeOutputCountRate (
    unsigned int *Statistics,
    unsigned short ModNum,
    unsigned short ChanNum );
```

```
// Compute live time that a channel accumulated in a run
// Use this function to calculate the live time that one channel of a Pixie-16
// →module has spent on data acquisition. This function does not communicate with
// →Pixie-16 modules. Before calling this function, another function,
// →Pixie16ReadStatisticsFromModule, should be called to read statistics data from
// →the module first.
// *Statistics is a pointer to an array whose size is exactly 448 unsigned integer
// →words (32-bit). The *Statistics array is filled with data from a Pixie-16 module
// →after calling function Pixie16ReadStatisticsFromModule. ModNum is the module
// →number which starts counting at 0. ChanNum is the channel number which starts
// →counting at 0.
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeLiveTime (
    unsigned int *Statistics,
    unsigned short ModNum,
    unsigned short ChanNum );
```

```
// Compute number of events processed by a channel
// Use this function to calculate the number of events that have been processed by
// →a Pixie-16 module during a data acquisition run. This function is only used by
// →Rev-A modules. This function does not communicate with Pixie-16 modules. Before
// →calling this function, another function, Pixie16ReadStatisticsFromModule, should
// →be called to read statistics data from the module first.
// *Statistics is a pointer to an array whose size is exactly 448 unsigned integer
// →words (32-bit). The *Statistics array is filled with data from a Pixie-16 module
// →after calling function Pixie16ReadStatisticsFromModule. ModNum is the module
// →number which starts counting at 0. ChanNum is the channel number which starts
// →counting at 0.
```

(下页继续)

(续上页)

```
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeProcessedEvents (
    unsigned int *Statistics,
    unsigned short ModNum );
```

```
// Compute real time that a channel accumulated in a run
// Use this function to calculate the real time that a Pixie-16 module has spent
// on data acquisition. This function does not communicate with Pixie-16 modules.
// Before calling this function, another function, Pixie16ReadStatisticsFromModule,
// should be called to read statistics data from the module first.
// *Statistics is a pointer to an array whose size is exactly 448 unsigned integer
// words (32-bit). The *Statistics array is filled with data from a Pixie-16 module
// after calling function Pixie16ReadStatisticsFromModule. ModNum is the module
// number which starts counting at 0. ChanNum is the channel number which starts
// counting at 0.
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeRealTime (
    unsigned int *Statistics,
    unsigned short ModNum );
```

```
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16complexFFT (
    double *data,
    unsigned int length );

// Test one bit of a 16-bit unsigned integer
PIXIE16APP_EXPORT unsigned short PIXIE16APP_API APP16_TstBit (
    unsigned short bit,
    unsigned short value );

// Set one bit of a 16-bit unsigned integer
PIXIE16APP_EXPORT unsigned short PIXIE16APP_API APP16_SetBit (
    unsigned short bit,
    unsigned short value );

// Clear one bit of a 16-bit unsigned integer
PIXIE16APP_EXPORT unsigned short PIXIE16APP_API APP16_ClrBit (
    unsigned short bit,
    unsigned short value );

// Set one bit of a 32-bit unsigned integer
PIXIE16APP_EXPORT unsigned int PIXIE16APP_API APP32_SetBit (
    unsigned short bit,
    unsigned int value );

// Clear one bit of a 32-bit unsigned integer
PIXIE16APP_EXPORT unsigned int PIXIE16APP_API APP32_ClrBit (
    unsigned short bit,
    unsigned int value );

// Test one bit of a 32-bit unsigned integer
PIXIE16APP_EXPORT unsigned int PIXIE16APP_API APP32_TstBit (
    unsigned short bit,
    unsigned int value );
```

```
// Program on-board DACs
// Use this function to reprogram the on-board digital to analog converters (DAC)
// of the Pixie-16 modules. In this operation the DSP uses data from the DSP
// parameters that were previously downloaded.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16SetDACS (
    unsigned short ModNum );
```

```
// Program on-board signal processing FPGAs
// Use this function to program the on-board signal processing FPGAs of the Pixie-
// →16 modules. After the host computer has written the DSP parameters to the DSP_
// →memory, the DSP needs to write some of these parameters to the FPGAs. This_
// →function makes the DSP perform that action.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ProgramFippi (
    unsigned short ModNum );
```

```
// Adjust DC-offsets in single or multiple modules
// Use this function to adjust the DC-offsets of Pixie-16 modules. Specify the_
// →module using ModNum. If ModNum is set to be less than the total number of_
// →modules in the system, only the module specified by ModNum will have its DC-
// →offsets adjusted. But if ModNum is set to be equal to the total number of_
// →modules in the system, then all modules in the system will have their DC-offsets_
// →adjusted.
// After the DC-offset levels have been adjusted, the baseline level of the_
// →digitized input signals will be determined by the DSP parameter BaselinePercent._
// →For instance, if BaselinePercent is set to 10(%), the baseline level of the_
// →input signals will be ~ 1638 on the 14-bit ADC scale (minimum: 0; maximum:_
// →16383).
// The main purpose of this function is to ensure the input signals fall within_
// →the voltage range of the ADCs to ensure all input signals can be digitized by_
// →the ADCs properly.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16AdjustOffsets (
    unsigned short ModNum );
```

```
// Acquire baselines from a module
// Use this function to acquire baselines from Pixie-16 modules. Specify the_
// →module using ModNum. If ModNum is set to be less than the total number of_
// →modules in the system, only the module specified by ModNum will have its_
// →baselines acquired. But if ModNum is set to be equal to the total number of_
// →modules in the system, then all modules in the system will have their baselines_
// →acquired.
// After the successful return of this function, the DSP's internal memory will be_
// →filled with baselines data. Users should then call another function_
// →Pixie16ReadSglChanBaselines to read the baselines data out to the host computer,_
// →channel by channel.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16AcquireBaselines (
    unsigned short ModNum );           // module number
```

```
// Read baselines from a channel in a module
// Use this function to read baselines data from a Pixie-16 module. Before calling_
// →this function, another function Pixie16AcquireBaselines should be called to fill_
// →the DSP internal memory first. Also, the host code should allocate appropriate_
// →amount of memory to store the baseline data. The baselines data length for each_
// →channel is 3640. In the DSP internal memory, each baseline data is a 32-bit IEEE_
// →floating point number. After being read out to the host, this function will_
// →convert each baseline data to a decimal number. In addition to baseline values,_
// →timestamps corresponding to each baseline were also returned after this function_
// →call.
// Specify the module using ModNum and the channel on the module using ChanNum._
// →Note that the modules and channels are counted starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadSglChanBaselines (
    double *Baselines,                // returned baselines values
    double *TimeStamps,               // time stamp for each baseline value
    unsigned short NumBases,          // number of baseline values to read
    unsigned short ModNum,            // module number
    unsigned short ChanNum );         // channel number
```

```
// Ramp Offset DACs of a module and record the baselines
```

(下页继续)

(续上页)

```
// Use this function to execute the RAMP_OFFSETDACS control task run. Each Offset_
→DAC has 65536 steps, and the RAMP_OFFSETDACS control task ramps the DAC from 0_
→to 65335 with a step size of 64, i.e., a total of 1024 steps. At each DAC step,_
→the control task computes the baseline value as the representation of the signal_
→baseline and stores it in the DSP memory. After the control task is finished,_
→the stored baseline values are read out to the host computer and saved to a_
→binary file called "rampdacs.bin" in the form of IEEE 32-bit floating point_
→numbers. Users can then plot the baseline values vs. DAC steps to determine the_
→appropriate DAC value to be set in the DSP in order to bring the input signals_
→into the voltage range of the ADCs. However, this function is no longer needed_
→due to the introduction of function Pixie16AdjustOffsets.
// If ModNum is set to less than the total number of modules in the system, only_
→the module specified by ModNum will start the RAMP_OFFSETDACS control task run._
→But if ModNum is equal to the total number of modules in the system, e.g. there_
→are 5 modules in the chassis and ModNum = 5, then all modules in the system will_
→start the RAMP_OFFSETDACS control task run. Note that the modules are counted_
→starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16RampOffsetDACS (
    double *DCValues,           // returned DC offset values
    unsigned short NumDCVals,   // number of DC values to read
    unsigned short ModNum );
```

```
// Execute special control tasks
// Use this function to call special control tasks. This may include programming_
→the Fippi or setting the DACs after downloading DSP parameters.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ControlTaskRun (
    unsigned short ModNum,       // Pixie module number
    unsigned short ControlTask, // Control task number
    unsigned int Max_Poll );   // Timeout control in unit of ms for control_
→task run
```

```
// Find the Baseline Cut values of a module
// Use this function to find the Baseline Cut value for one channel of a Pixie-16_
→module. The baseline cut value is then downloaded to the DSP, where baselines_
→are captured and averaged over time. The cut value would prevent a bad baseline_
→value from being used in the averaging process, i.e., if a baseline value is_
→outside the baseline cut range, it will not be used for computing the baseline_
→average. Averaging baselines over time improves energy resolution measurement.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16BLcutFinder (
    unsigned short ModNum,       // Pixie module number
    unsigned short ChanNum,      // Pixie channel number
    unsigned int *BLcut );       // BLcut return value
```

```
// Find the exponential decay time of a channel
// Use this function to find the exponential decay time constant (Tau value) of_
→the detector or preamplifier signal that is connected to one channel of a Pixie-
→16 module. The found Tau value is returned via pointer *Tau.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16TauFinder (
    unsigned short ModNum,       // Pixie module number
    double *Tau );              // 16 returned Tau values, in [F]
```

```
// Write a MODULE level parameter to a module
// Use this function to write a module parameter to a Pixie-16 module.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16WriteSglModPar (
    char *ModParName,           // the name of the module parameter
    unsigned int ModParData,     // the module parameter value to be written to_
→the module
    unsigned short ModNum );    // module number
```

```
// Read a MODULE level parameter from a module
// Use this function to read a module parameter from a Pixie-16 module.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadSglModPar (
    char *ModParName,           // the name of the module parameter
    unsigned int    *ModParData, // the module parameter value to be read from
    →the module
    unsigned short ModNum );   // module number
```

```
// Write a CHANNEL level parameter to a module
// Use this function to write a channel parameter to a Pixie-16 module.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16WriteSglChanPar (
    char *ChanParName,          // the name of the channel parameter
    double ChanParData,         // the channel parameter value to be written
    →to the module
    unsigned short ModNum,      // module number
    unsigned short ChanNum );   // channel number
```

```
// Read a CHANNEL level parameter from a module
// Use this function to read a channel parameter from a Pixie-16 module.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadSglChanPar (
    char *ChanParName,          // the name of the channel parameter
    double *ChanParData,         // the channel parameter value to be read from
    →the module
    unsigned short ModNum,      // module number
    unsigned short ChanNum );   // channel number
```

```
// Read histogram data from a module
// Use this function to read out the histogram data from a Pixie-16 module's
// histogram memory. Before calling this function, the host code should allocate
// appropriate amount of memory to store the histogram data. The default histogram
// length is 32768. Histogram data are 32-bit unsigned integers.
// Specify the module using ModNum and the channel on the module using ChanNum.
// Note that both the modules and channels are counted starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadHistogramFromModule (
    unsigned int    *Histogram,    // histogram data
    unsigned int    NumWords,     // number of words to be read out
    unsigned short ModNum,       // module number
    unsigned short ChanNum );    // channel number
```

```
// Read run statistics data from a module
// Use this function to read out statistics data from a Pixie-16 module. Before
// calling this function, the host code should allocate appropriate amount of
// memory to store the statistics data. The number of statistics data for each
// module is fixed at 448. Statistics data are 32-bit unsigned integers.
// Specify the module using ModNum. Note that the modules are counted starting at
// 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadStatisticsFromModule (
    unsigned int    *Statistics,   // run statistics data
    unsigned short ModNum );     // module number
```

```
// Read histogram data from a module and save to a file
// Use this function to read histogram data from a Pixie-16 module and save the
// data to a file. New data will be appended to the end of the file. So the same
// file name can be used for multiple modules and the data from each module will be
// stored in the order that this function is called.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16SaveHistogramToFile (
    char *FileName,             // histogram data file name
    unsigned short ModNum);    // module number
```

```
// Parse a list mode data file to get events information
// Use this function to parse the list mode events in the list mode data file. The
// number of events for each module will be reported.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16GetModuleEvents (
    char *FileName, // the list mode data file name (with complete
    // path)
    unsigned int *ModuleEvents ); // receives number of events for each module
```

```
// Get detailed events information from a data file
// Use this function to retrieve the detailed information of each event in the
// list mode data file for the designated module. Before calling this function to
// get the individual events information, another function Pixie16GetModuleEvents
// should be called first to determine the number of events that have been recorded
// for each module. If the number of events for a given module is nEvents, a memory
// block *EventInformation should be allocated with a length of (nEvents*68):
// EventInformation = (unsigned long *)malloc(sizeof(unsigned long) * nEvents *
// 68);
// where 68 is the length of the information records of each event (energy,
// timestamps, etc.) and has the following structure.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16GetEventsInfo (
    char *FileName, // the list mode data file name (with
    // complete path)
    unsigned int *EventInformation, // to hold event information
    unsigned short ModuleNumber); // the module whose events are to be
    // retrieved
```

```
// Read trace data from a list mode data file
// Use this function to retrieve list mode trace from a list mode data file. It
// uses the trace length and file location information obtained from function
// Pixie16GetEventsInfo for the selected event.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadListModeTrace (
    char *FileName, // list mode data file name
    unsigned short *Trace_Data, // list mode trace data (16-bit words)
    unsigned short NumWords, // number of 16-bit words to be read out
    unsigned int FileLocation); // the location of the trace in the file
```

```
// Read histogram data from a histogram data file
// Use this function to read histogram data from a histogram data file. Before
// calling this function, the host code should allocate appropriate amount of
// memory to store the histogram data. The default histogram length is 32768.
// Histogram data are 32-bit unsigned integers.
// Specify the module using ModNum and the channel on the module using ChanNum.
// Note that both the modules and channels are counted starting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadHistogramFromFile (
    char *FileName, // the histogram data file name (with complete
    // path)
    unsigned int *Histogram, // histogram data
    unsigned int NumWords, // number of words to be read out
    unsigned short ModNum, // module number
    unsigned short ChanNum); // channel number
```

```
// Read DSP parameters from modules and save to a file
// Use this function to save DSP parameters to a settings file. It will first read
// the values of DSP parameters on each Pixie-16 module and then write them to the
// settings file. Each module has exactly 1280 DSP parameter values (32-bit
// unsigned integers), and depending on the value of PRESET_MAX_MODULES (defined in
// pixie16app_defs.h), the settings file should have exactly (1280 * PRESET_MAX_
// MODULES * 4) bytes when stored on the computer hard drive.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16SaveDSPParametersToFile (
    char *FileName ); // the DSP parameters file name (with complete
    // path)
```

```
// Load DSP parameters to modules from a file
// Use this function to read DSP parameters from a settings file and then download_
// the settings to Pixie-16 modules that are installed in the system. Each module_
// has exactly 1280 DSP parameter values (32-bit unsigned integers), and depending_
// on the value of PRESET_MAX_MODULES (defined in pixie16app_defs.h), the settings_
// file should have exactly (1280 * PRESET_MAX_MODULES * 4) bytes when stored on_
// the computer hard drive.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16LoadDSPParametersFromFile (
    char *FileName ); // the DSP parameters file name (with complete_
// path)
```

```
// Copy DSP parameters from a module to others
// Use this function to copy DSP parameters from one module to the others that are_
// installed in the system.
// BitMask is bit pattern which designates which items should be copied from the_
// source module to the destination module(s).
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16CopyDSPParameters (
    unsigned short BitMask, // copy items bit mask
    unsigned short SourceModule, // source module
    unsigned short SourceChannel, // source channel
    unsigned short *DestinationMask ); // the destination module and channel_
// bit mask
```

```
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadMSGFile (
    char *ReturnMsgStr );
```

```
// Convert a decimal into IEEE 32-bit floating point number
PIXIE16APP_EXPORT unsigned int PIXIE16APP_API Decimal2IEEEFloating(double_
//DecimalNumber);

// Convert an IEEE 32-bit floating point number to a decimal
PIXIE16APP_EXPORT double PIXIE16APP_API IEEEFloating2Decimal(unsigned int_
//IEEEFloatingNumber);
```

```
// Read data from external FIFO and save to a file
// Use this function to read data from the external FIFO of a module. This_
// function can only be used for Pixie-16 Revision-B, C, and D modules.
// This function first checks the status of the external FIFO of a Pixie-16 module,
// and if there are data in the external FIFO, this function then reads list mode_
// data (32-bit unsigned integers) from the external FIFO. So this function_
// essentially encapsulates both functions Pixie16CheckExternalFIFOStatus and_
// Pixie16ReadDataFromExternalFIFO within one function. The number of words that_
// are read from the external FIFO is recorded in variable*nFIFOWords.
// The function also expects setting the value of a variable called "EndOfRunRead"
// to indicate whether this read is at the end of a run (1) or during the run (0)._
// This is necessary since the external FIFO needs special treatment when the host_
// reads the last few words from the external FIFO due to its pipelined structure.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16SaveExternalFIFODataToFile (
    char *FileName, // list mode data file name
    unsigned int *nFIFOWords, // number of words read from external FIFO
    unsigned short ModNum, // module number
    unsigned short EndOfRunRead); // indicator whether this is the end of run_
// read
```

```
// Read from or write to registers on a module
// Use this function to read data from or write data to a register in a Pixie-16_
// module.
// Specify the module using ModNum. Note that the modules are counted starting at_
// 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16RegisterIO (
```

(下页继续)

(续上页)

```
unsigned short ModNum,           // the Pixie module to communicate to
unsigned int address,           // register address
unsigned short direction,       // either MOD_READ or MOD_WRITE
unsigned int *value );          // holds or receives the data
```

```
// Read Control & Status Register value from a module
// Use this function to read the host Control & Status Register (CSR) value.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadCSR (
    unsigned short ModNum,
    unsigned int *CSR );
```

```
// Write to Control & Status Register in a module
// Use this function to write a value to the host Control & Status Register (CSR).
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16WriteCSR (
    unsigned short ModNum,
    unsigned int CSR );
```

```
// Check status of external FIFO of a module
// Use this function to check the status of the external FIFO of a Pixie-16 module.
// While a list mode data acquisition run is in progress. The function returns the
// number of words (32-bit) that the external FIFO currently has. If the number of
// words is greater than a user-set threshold, function
// Pixie16ReadDataFromExternalFIFO can then be used to read the data from the
// external FIFO. The threshold can be set by the user to either minimize reading
// overhead or to read data out of the FIFO as quickly as possible.
// *nFIFOWords returns the number of 32-bit words that the external FIFO currently
// has.
// ModNum is the module number which starts counting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16CheckExternalFIFOStatus (
    unsigned int *nFIFOWords,
    unsigned short ModNum );
```

```
// Read data from external FIFO of a module
// Use this function to read data from the external FIFO of a module. This
// function can only be used for Pixie-16 Revision-B, C, and D modules.
// This function reads list mode data from the external FIFO of a Pixie-16 module.
// The data are 32-bit unsigned integers. Normally, function
// Pixie16CheckExternalFIFOStatus is called first to see how many words the
// external FIFO currently has, then this function is called to read the data from
// the FIFO.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ReadDataFromExternalFIFO (
    unsigned int *ExtFIFO_Data, // To receive the external FIFO data
    unsigned int nFIFOWords,   // number of words to read from external FIFO
    unsigned short ModNum ); // module number
```

```
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ComputeFastFiltersOffline (
    char *fileName,           // the list mode data file name (with
    // complete path)
    unsigned short ModuleNumber, // the module whose events are to be
    // analyzed
    unsigned short ChannelNumber, // the channel whose events are to be
    // analyzed
    unsigned int FileLocation, // the location of the trace in the file
    unsigned short RcdTraceLength, // recorded trace length
    unsigned short *RcdTrace, // recorded trace
    double *fastfilter, // fast filter response
    double *cfd ); // cfd response
```

```

PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16ComputeSlowFiltersOffline (
    char          *FileName,           // the list mode data file name (with_
→complete path)
    unsigned short ModuleNumber,      // the module whose events are to be_
→analyzed
    unsigned short ChannelNumber,     // the channel whose events are to be_
→analyzed
    unsigned int   FileLocation,      // the location of the trace in the file
    unsigned short RcdTraceLength,    // recorded trace length
    unsigned short *RcdTrace,         // recorded trace
    double         *slowfilter );    // slow filter response

```

```

// Add by Hongyi Wu
PIXIE16APP_EXPORT int PIXIE16APP_API HongyiWuPixie16ComputeSlowFiltersOffline (
    char          *FileName,           // the list mode data file name (with_
→complete path)
    unsigned short ModuleNumber,      // the module whose events are to be_
→analyzed
    unsigned short ChannelNumber,     // the channel whose events are to be_
→analyzed
    unsigned int   FileLocation,      // the location of the trace in the file
    unsigned short RcdTraceLength,    // recorded trace length
    unsigned short *RcdTrace,         // recorded trace
    double         *slowfilter,        // slow filter response
    unsigned int   bl,
    double         sl,
    double         sg,
    double         tau,
    int            sfr,
    int            pointtbl );

```


CHAPTER 23

PKU Code

本节介绍程序的主要思路。

DOTO 需要补充框图帮助理解程序!!!

23.1 Decode

- decoder.cc
- **decoder.hh**
 - 读取二进制文件
- **main.cc**
 - 主程序
- Makefile
- r2root.cc
- **r2root.hh**
 - 保存 ROOT 文件
- **UserDefine.hh**
 - 用户定义参数

23.2 GUI

- Base.cc
- **Base.hh**
 - 子界面，基线、极性、增益、波形长度、数据记录等参数调节
- Cfd.cc
- **Cfd.hh**
 - 子界面，CFD 参数调节

- CopyPars.cc
- **CopyPars.hh**
 - 子界面, 参数复制
- Csra.cc
- **Csra.hh**
 - 子界面, 方便快速调节每通道的控制寄存器
- Decimation.cc
- **Decimation.hh**
 - 子界面, 降频参数设置
- Detector.cc
- **Detector.hh**
 - 数据采集循环主体
- Energy.cc
- **Energy.hh**
 - 子界面, 梯形参数调节界面
- ExpertMod.cc
- **ExpertMod.hh**
 - 子界面, 采集卡模块参数设置
- Global.cc
- **Global.hh**
 - 全局函数
- HistXDT.cc
- **HistXDT.hh**
 - 子界面, 设置记录的一维能谱的最小值、bin 宽及 DSP 抓波形时的部长
- LogicTrigger.cc
- **LogicTrigger.hh**
 - 子界面, 逻辑参数调节
- main.cc
- MainFrame.cc
- **MainFrame.hh**
 - 主控制界面
- MainLinkdef.h
- Makefile
- Offline.cc
- **Offline.hh**
 - 离线分析主界面, 离线分析功能代码
- OfflineData.cc
- **OfflineData.hh**
 - 离线分析读取文件数据

- pkuFFTW.cc
- **pkuFFTW.hh**
 - 基于 FFTW3 封装类
- Qdc.cc
- **Qdc.hh**
 - 子界面，用于 QDC 积分门窗的调节
- ReadChanStatus.cc
- **ReadChanStatus.hh**
 - 子界面，查看 DSP 中抓取的波形及 baseline
- Simulation.cc
- **Simulation.hh**
 - 未实现
- Table.cc
- **Table.hh**
 - 基类，用于参数调节界面
- TriggerFilter.cc
- **TriggerFilter.hh**
 - 子界面，fast filter 参数调节界面
- **wuReadData.hh**
 - 模版函数，用来读取输入卡

23.3 MakeEvent

- **main.cc**
 - 主程序
- Makefile
- sort.cc
- **sort.hh**
 - 事件组装
- **UserDefine.hh**
 - 用户定义参数

23.4 OnlineStatics

- Linkdef.hh
- **main.cc**
 - 主程序
- Makefile
- Online.cc

- **Online.hh**
 - 在线监视界面
- PixieOnline.config