
Table of Contents

README	1.1
INSTALL	1.2
Users Guide	1.3
firmware	1.3.1
Decode	1.3.2
GUI	1.3.3
OnlineStatics	1.3.4
MakeEvent	1.3.5
FrontPanel	1.3.6
Programmers Guide	1.4
XIA API	1.4.1
PKU code	1.4.2

README

最新版本号 Version:2018.05.12

程序下载请访问(私有仓库，暂未开放下载) <https://github.com/wuhongyi/PKUXIADAQ>

网页版说明书请访问 <http://wuhongyi.cn/PKUXIADAQ/>

markdown 版本说明书请访问 README/

离线网页版说明书请访问 docs/

pdf版本说明书请访问 README.pdf

- 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，当前基本以操作演示讲解软件的使用为主。
-

This manual applies only to XIA LLC Pixie-16

- This program is developed by the **experimental nuclear physics group of Peking University**. 本程序由北京大学实验核物理组开发。
- The earliest graphical interface development of this program is based on NSCL DDAS Nscope(<http://docs.nscl.msu.edu/daq/newsite/ddas-1.1/nscope.html>). 最早的图形界面程序是基于 NSCL DDAS Nscope 开发。
- Thanks to Hui Tan's(XIA LLC) support for our development. 特别感谢 谭辉(XIA LLC) 对我们开发的支持。

Technical adviser 技术指导:

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

Software Developer 软件主要开发者:

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

Principal author of the instruction 说明书主要撰写者:

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

The development of this program is supported by the following 本程序的开发得到以下单位的支持：

- XIA LLC
- Institute of Modern Physics, Chinese Academy of Sciences 中国科学院近代物理研究所(IMP)
- China Institute of Atomic Energy 中国原子能科学研究院(CIAE)
- The University of Hong Kong 香港大学(HKU)

- Shandong University, Weihai 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 添加各类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。当前我们只有100/250 MHz 14 bit 的采集卡，因此默认可运行该类型的采集卡，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)。

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。新版本升级中，暂时不可用)
 - OnlineStatics(在线监视程序)
 - parset(参数设置文件)
 - PlxSdk.tar.gz(Plx9054 驱动)
 - README(说明书 markdown 版)
 - README.md(github首页介绍)
 - README.pdf(pdf版本说明书)
 - software(pixie16驱动API，非官方标准，已经被吴鸿毅修改)
 - TestTool(开发者测试工具，用户不需要！！！)
-

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：
 - FastCGI
 - JSROOT
 - web
 - ...

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Scientific Linux 7.2/7.3/7.4

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

```

```

##编译图形化获取软件

#修改设置参数
cd ~
cd PKUXIADAQ/parset/
#修改cfgPixie16.txt文件。
#其中CreateID 后面的数值表示机箱编号，时值允许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。
#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。

#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。
```

```

cd ~
cd PKUXIADAQ/GUI/
make clean
make

```

```

##编译在线监视程序

cd ~

```

```
cd PKUXIADAQ/OnlineStatics/
#修改 PixieOnline.config 文件中的参数
#第一行为获取数据文件存放路径
#第二行为获取文件名

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
```

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）
```

```
##启动在线监视程序

cd ~
cd PKUXIADAQ/OnlineStatics/
./online
#将会弹出图形化界面
#查看上面的原始数据文件夹路径、文件名是否正确。按 Complete 确认。
#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 表示 Run Number
```

Guide

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

- 这里需要介绍跳线接法
- 原始数据定义
- 算法原理
- . . .

Do Not Hot-Swap!

To avoid personal injury, and/or damage to the DGF-Pixie-16, always turn off crate power before removing the DGF-Pixie-16 from the crate!

The DGF Pixie-16 is a 16-channel all-digital waveform acquisition and spectrometer card based on the CompactPCI/PXI standard for fast data readout to the host. It combines spectroscopy with waveform digitizing and the option of on-line pulse shape analysis. The Pixie-16 accepts signals from virtually any radiation detector. Incoming signals are digitized by 12/14/16-bit 100/250/500 MSPS ADCs. Waveforms of up to 163.8 μ s in length for each channel can be stored in a FIFO.

The waveforms are available for onboard pulse shape analysis, which can be customized by adding user functions to the core processing software. Waveforms, timestamps, and the results of the pulse shape analysis can be read out by the host system for further off-line processing. Pulse heights are calculated to 16-bit precision and can be binned into spectra with up to 32K channels. The Pixie-16 supports coincidence spectroscopy and can recognize complex hit patterns.

Data readout rates through the CompactPCI/PXI backplane to the host computer can be up to 109 Mbyte/s. The standard PXI backplane, as well as additional custom backplane connections are used to distribute clocks and trigger signals between several Pixie-16 modules for group operation. A complete data acquisition and processing systems can be built by combining Pixie-16 modules with commercially available CompactPCI/PXI processor, controller or I/O modules in the same chassis.

The Pixie-16 is an instrument for waveform acquisition and MCA histogramming for arrays of gamma ray or other radiation detectors such as

- 100 MSPS
 - Segmented HPGe detectors.
 - Scintillator/PMT combinations: NaI, CsI, BGO and many others.
 - Gas detector.
 - Silicon strip detectors.
- 250 MSPS
 - Scintillator
 - LaBr₃
- 500 MSPS

- Scintillator
- LaBr3

The Pixie-16 modules must be operated in a custom 6U CompactPCI/PXI chassis providing high currents at specific voltages not included in the CompactPCI/PXI standard 1 . Currently XIA provides a 14-slot chassis. Put the host computer(or remote PXI controller) in the system slot (slot 1) of your chassis. Put the Pixie-16 modules into any free peripheral slot (slot 2-14) with the chassis still powered down. After modules are installed, power up the chassis (Pixie-16 modules are not hot swappable). If using a remote controller, be sure to boot the host computer after powering up the chassis.

FIRMWARE

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 14 bit(pixie16_revfpku_14b100m_dsp_update_05082018)
 - MultiplicityMaskHigh[31]=0和1 时候前面版均能输出 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 14bit(pixie16_revf_14b250m_firmware_release_04222018)
 - 前面板多重性MultiplicityMaskHigh[31]=0和1 时候前面版均能输出 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。

Decode

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 ROOTFILEPATH "/home/wuhongyi/data/" //要生成ROOT文件的路径
#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 Crate0
#define Crate0num 5 //该机箱中使用插件数
const int Crate0SamplingRate[Crate0num] = {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：

```
./decode 3
```

ROOT File Branch :

- sr(short):sample rate，100/250/500，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):timestamps
- ets(long int 64 bit):外部时标
- cfd(short):cfд数值
- cfdft(bool):cfд数值是否有效
- cfds(short):cfд 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 :cfd    : cfd/S
*Entries : 1123666 : Total Size= 2263364 bytes File Size = 2191516 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 1.03 *
*.....
*Br  10 :cfdfit : cfdfit/0
*Entries : 1123666 : Total Size= 1140054 bytes File Size = 30291 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 37.51 *
*.....
*Br  11 :cfds   : cfds/S
*Entries : 1123666 : Total Size= 2263543 bytes File Size = 269187 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 8.39 *
```

```
.....*
*Br 12 :trae    : traе/i
*Entries : 1123666 : Total Size= 4510879 bytes File Size = 2642761 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 1.71 *
*.....*
*Br 13 :leae    : leае/i
*Entries : 1123666 : Total Size= 4510879 bytes File Size = 2886877 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 1.56 *
*.....*
*Br 14 :gape    : gаре/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    : lтra/s
*Entries : 1123666 : Total Size= 2263543 bytes File Size = 230891 *
*Baskets : 175 : Basket Size= 51200 bytes Compression= 9.79 *
*.....*
*Br 18 :data    : data[lтra]/s
*Entries : 1123666 : Total Size= 6945634237 bytes File Size = 3206301603 *
*Baskets : 1689 : Basket Size= 25600000 bytes Compression= 2.17 *
*.....*
*Br 19 :dt      : dt[lтra]/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

GUI

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
```

主控制界面



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

- File
 - Exit
 - About
- UV_Setup
 - Base Setup
 - Trigger Filter
 - Energy
 - CFD
 - QDC
 - Decimation
 - Copy Pars
 - Save2File
- Expert
 - Module Variables
 - CSRA
 - Logic Set
- Monitor
 - 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 **LSRunStart** 则开启，此时点击该按钮，获取则开启，按钮状态更改为 **LSRunStop**，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 **LSRunStart** 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。

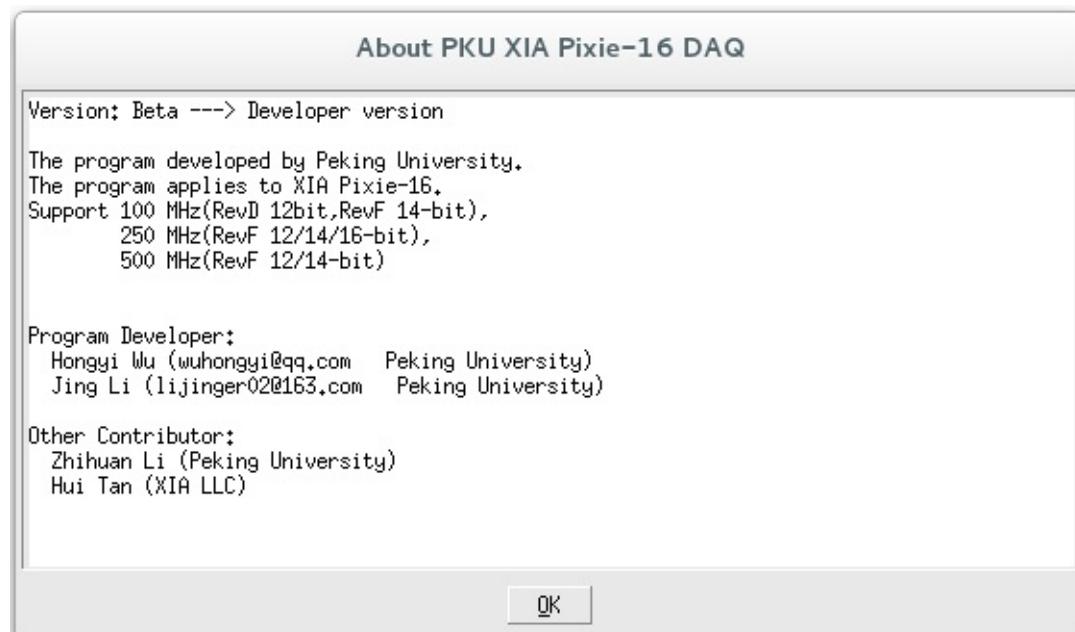
Control 栏内的 **Online Statistics** 选项开启则获取每 3 s 向 *OnineStatistics* 程序发送时时每路信号的输入率、输出率信息。

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。

File 下拉栏

本下拉栏内容没有实际用途。

About

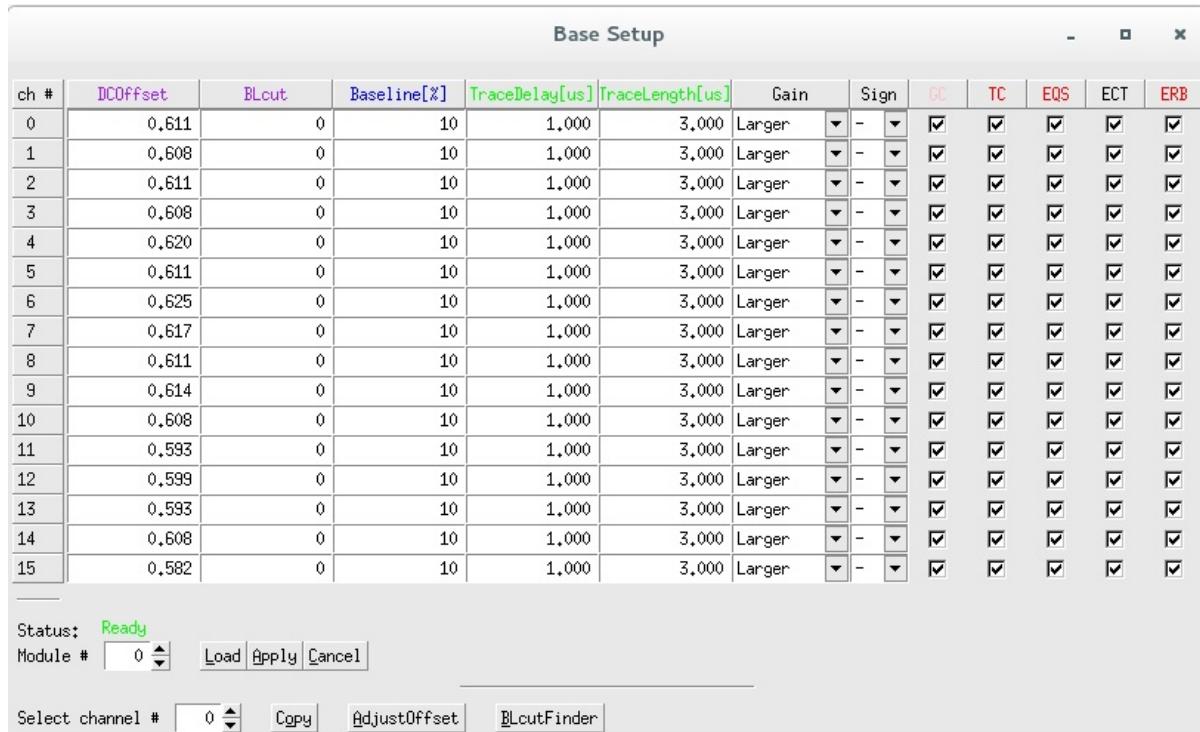


软件开发者介绍。之后将会添加主程序基本操作说明。

UV_Setup 下拉栏

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。

Base Setup



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。

- 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 表示是否开启该通道，选择表示开启该通道，不选择表示不开启。
- 选项 ECT 选择表示开启CFD功能。

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 x 2^N / TraceLength x 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。

- Good channel. Only channels marked as good will have their events recorded. This setting has no bearing on the channel's capability to issue a trigger. There can be a triggering channel whose data are discarded. Channels not marked as good will be excluded from the automatic offset adjustment.
- Capture trace. When this box is checked, trace will be captured and recorded for each event, along with other list mode information, e.g. timestamp, energy, etc.
- Capture QDC sums. When this box is checked, eight QDC sums will be recorded for each event. QDC sums are consecutive sums of the list mode trace.
- Enable CFD trigger. Check this box to enable this channel's CFD trigger. Otherwise, regular trapezoidal fast trigger will be used.
- Capture raw energy sums and baseline. Check this box to record raw energy sums and baseline value for each event.

The **Base Setup** page controls the analog gain, offset and polarity for each channel. It is useful to click on **Trace & Baseline** in the top control **Monitor** bar to view the signal read from the ADCs while adjusting these parameters. The display shows one or all 16 channels of a module; you can set the sampling interval for each block to capture a longer time frame in **Hist & XDT** page. Click **Draw** to update the graph.

Pulses from the detector should fall in the range from 0 to 12/14/16-bit -1, with the baseline at ~400 to allow for drifts and/or undershoots and no clipping at the upper limit. If there is clipping, adjust the Gain and Offset or click on the Adjust Offsets button to let the software set the DC offsets to proper levels.**TODO** 这段还需要进一步修改

Since the trigger/filter circuits in the FPGA only act on rising pulses, negative pulses are inverted at the input of the FPGA, and the waveforms shown in the ADC trace display include this optional inversion. Thus set the channel's Polarity such that pulses from the detector appear with positive amplitude (rising edge).

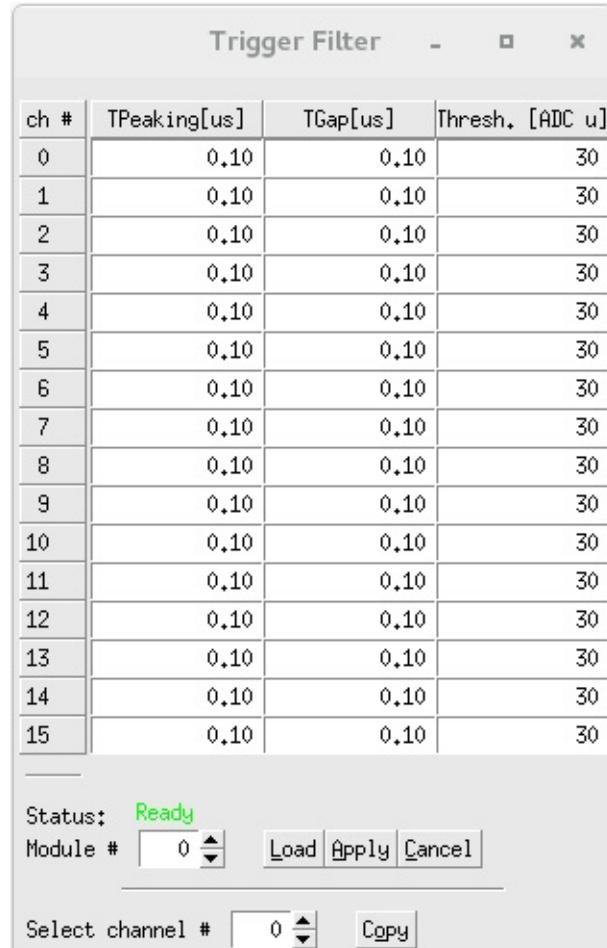
In the **Base Setup** tab, you can set the total trace length and the pre-trigger trace delay for the waveforms to be acquired in list mode runs.

The Pixie-16 constantly takes baseline measurements when no pulse is detected and keeps a baseline average to be subtracted from the energy filter output during pulse height reconstruction. Baseline measurements that differ from the average by more than Baseline Cut will be rejected as they are likely contaminated with small pulses below the trigger threshold. You can click on the Acquire Baseline button to view a series of baseline measurements for each channel, and in the single channel view you can build a histogram of baselines to verify that the Baseline Cut does not reject measurements falling into the main (ideally Gaussian) peak in the baseline distribution. Usually, it is sufficient to keep Baseline Cut at its default value.

Note: Since the baseline computation takes into account the exponential decay, no pulses should be noticeable in the baseline display if a) the decay time is set correctly and b) the detector pulses are truly exponential.

Baseline Percent is a parameter used for automatic offset adjustment; by clicking on the Adjust Offsets button, offsets will be set such that the baseline seen in the ADC trace display falls at the Baseline Percent fraction of the full ADC range (e.g. for Baseline Percent = 10% the baseline falls at ADC step 1638 out of 16384 total).

Trigger Filter



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

TODO 这里需要补充一个示意图，还有计算公式。

Assume the digitized waveform stream can be represented by data series $\text{Trace}[i]$, $i=0,1,2,\dots$. First the fast filter response of the digitized waveform is computed as follows:

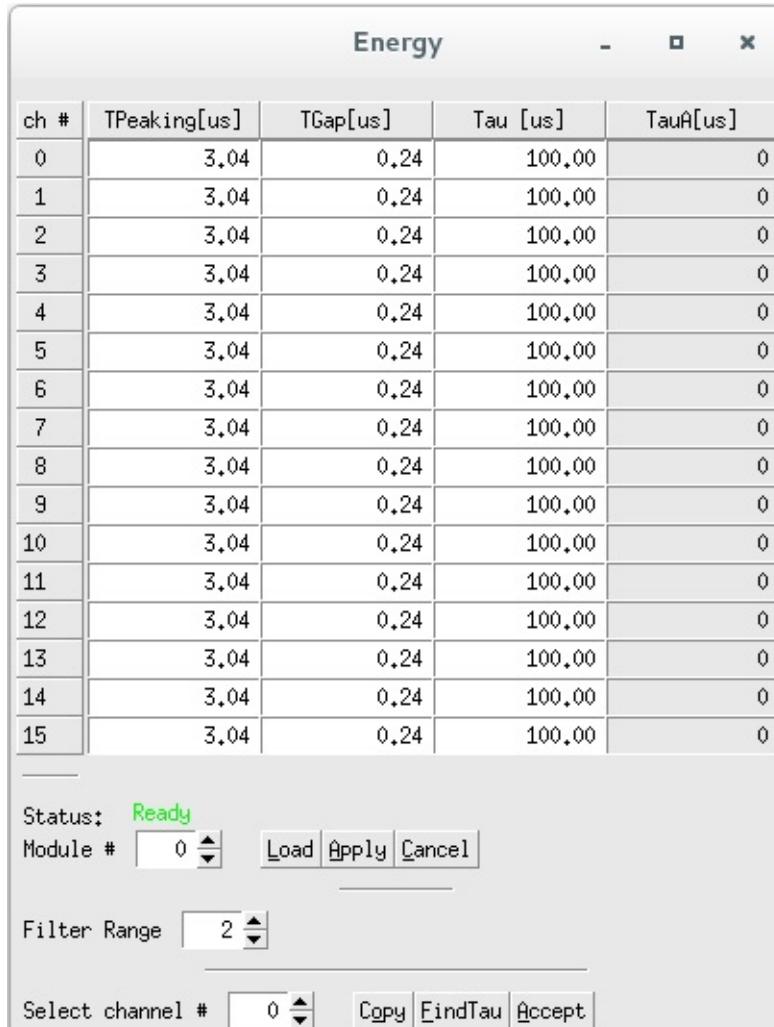
TODO公式

Where FL is called the fast length and FG is called the fast gap of the digital trapezoidal filter.

- A longer trigger filter rise time averages more samples and thus allows setting lower thresholds without triggering on noise.
- Typically the threshold should be set as low as possible, just above the noise level.

- A longer trigger filter flat top time makes it easier to detect slow rising pulses.

Energy



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。

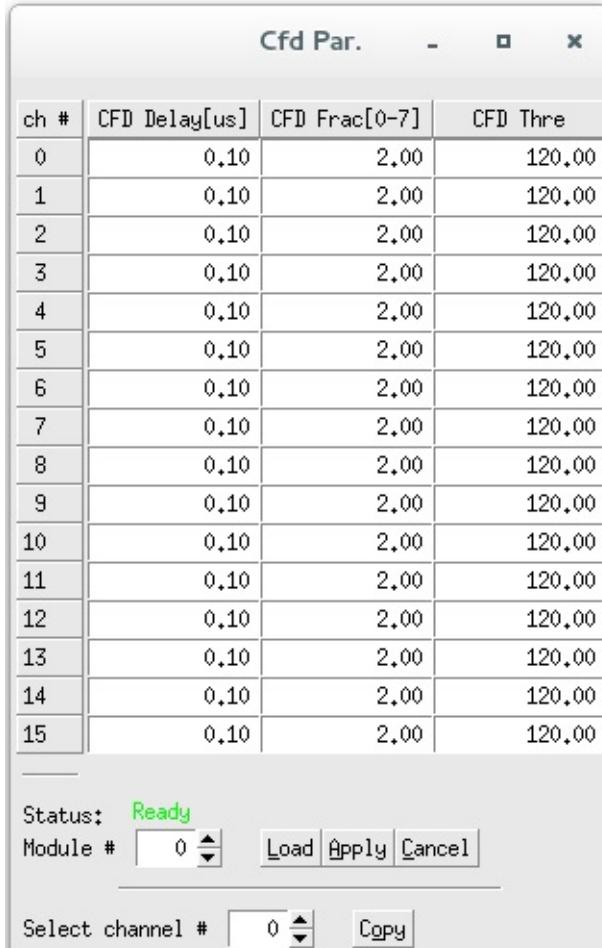
。 。 TODO。 。

The most critical parameter for the energy computation is the signal decay time Tau. It is used to compensate for the falling edge of a previous pulse in the computation of the energy. You can either enter Tau directly for each channel, or enter an approximate value in the right control, select a channel, and click Find it to let the software determine the decay time automatically. Click Accept it to apply the found value to the channel. (If the approximate value is unchanged, the software could not find a better value.) **TODO** 这段还需要进一步修改

- The energy filter flat top time should be about the same as the pulse rise time.
- The energy filter rise time can be varied to balance resolution and throughput. Typically, energy resolution increases with the length of the filter rise time, up to an optimum when longer filters only add more noise into the measurement. The filter dead time is about $TD = 2 \times (T_{rise} + T_{flat})$, and the maximum throughput for Poisson statistics is $1/(TD \cdot e)$. For HPGe detectors, a rise time of 4-6 μ s is usually appropriate.

- Choose the smallest energy filter range that allows setting the optimum energy filter rise time. Larger filter ranges allow longer filter sums, but increase the granularity of possible values for the energy filter rise time and flat top time and increase the jitter of latching the energy filter output relative to the rising edge of the pulse. This is usually only important for very fast pulses.

CFD



.. TODO ..

Then the CFD is computed as follows:

TODO公式

Where D is called the CFD delay length and W is called the CFD scaling factor.

The CFD zero crossing point (ZCP) is then determined when $CFD[i] > 0$ and $CFD[i+1] < 0$. The timestamp is latched at Trace point i, and the fraction time f is given by the ratio of the two CFD response amplitudes right before and after the ZCP.

TODO公式

Where $CFDout1$ is the CFD response amplitude right before the ZCP and $CFDout2$ is the CFD response amplitude (absolute value since $CFDout2$ is negative) right after the ZCP. The Pixie-16 DSP computes the CFD final value as shown below and stored it in the output data stream for online or offline analysis.

Valid CFD Scale values and corresponding CFD scaling factors are **TODO CFD参数对应表格**

QDC

ch #	QDC len0[us]	QDC len1[us]	QDC len2[us]	QDC len3[us]	QDC len4[us]	QDC len5[us]	QDC len6[us]	QDC len7[us]
0	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
1	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
2	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
3	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
4	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
5	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
6	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
7	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
8	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
9	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
10	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
11	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
12	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
13	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
14	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13
15	0.30	0.63	0.88	1.13	1.38	1.63	1.88	2.13

Status: **Ready**

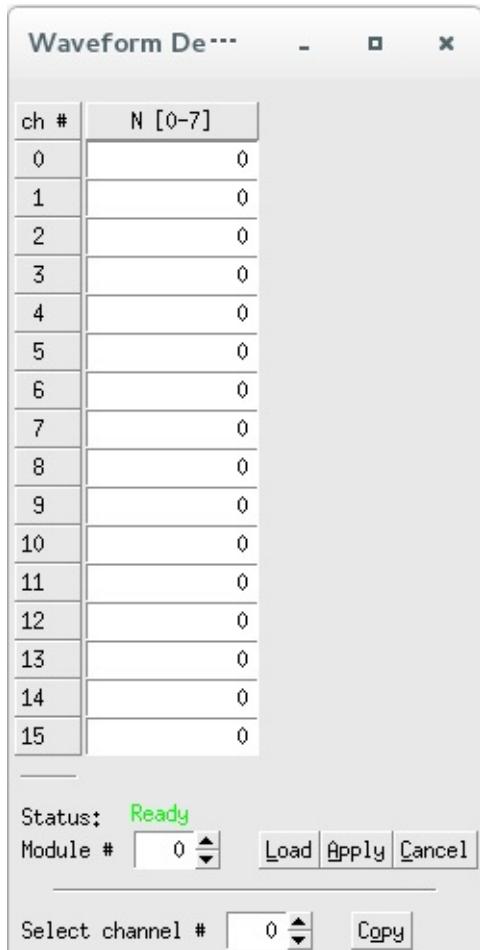
Module #

Select channel #

。 。 TODO。 。

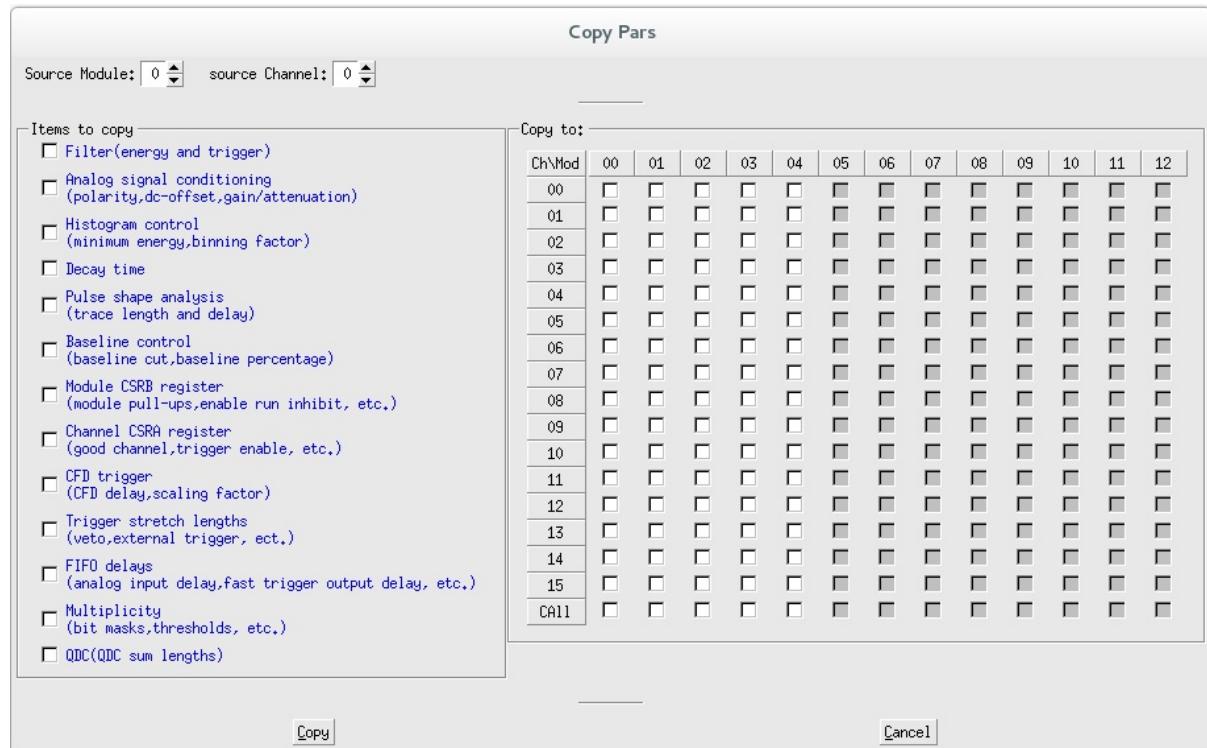
Eight QDC sums, each of which can have different length varying from 10 ns to 327.68 μ s, are computed in the signal processing FPGA of a Pixie-16 module for each channel and the sums are written to the output data stream if the user requests so. The recording of QDC sums starts at the waveform point which is Pre-trigger Trace Length earlier than the trigger point, which is either the CFD trigger or local fast trigger depending on whether or not CFD trigger mode is enabled. The eight QDC sums are computed one by one continuously, but they are not overlapping. Both the QDC sum length and the Pre-trigger trace length are set by the user. The recording of QDC sums ends when the eight intervals have passed.

Decimation



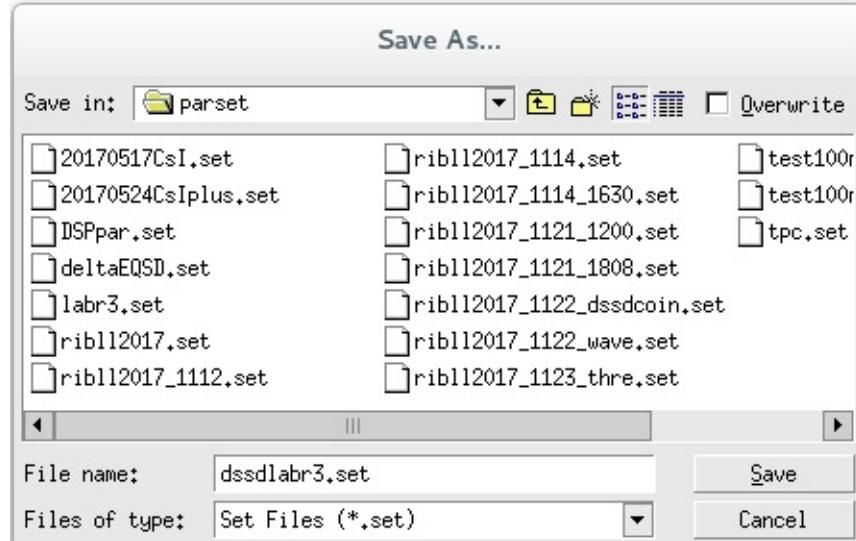
。 。 TODO。 。

Copy Pars



。 。 TODO。 。

Save2File



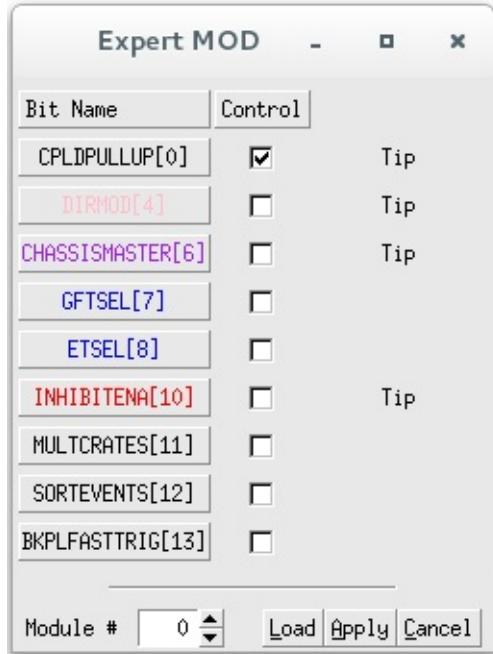
。 。 TODO。 。

Expert 下拉栏

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。

。 。 TODO。 。

Module Variables



.. TODO ..

Module Control Register B affecting the module as a whole.

- a) Enable pullups for backplane bus lines. This should be enabled for only one module in the crate.
- b) Connect trigger signals to backplane. You can set the module to share triggers over the backplane with other modules, unless you run each module (or channel) independently.
- c) Accept external trigger and run inhibit signals. Enable this option to let this module accept external trigger and run inhibit signals and then put the signals on the backplane so that all modules can see the same signals. This should be enabled for only one module in the crate.
- d) Crate master module (multiple crates only). This option is only used when multiple Pixie-16 crates communicate with each other. By enabling this option, the master module in each crate is responsible for sending synchronization or trigger signals to certain backplane lines. This should be enabled for only one module in the crate.
- e) Enable run inhibit signal input. This option is only applicable to the module which has the “Accept external trigger and run inhibit signals” option enabled. This should be enabled for only one module in the crate.
- f) Multiple crates. This option is only used when multiple Pixie-16 crates communicate with each other.

CSRA

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 checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" 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 checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input 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"/>							
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
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"/>											
Cal1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input 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"/>							

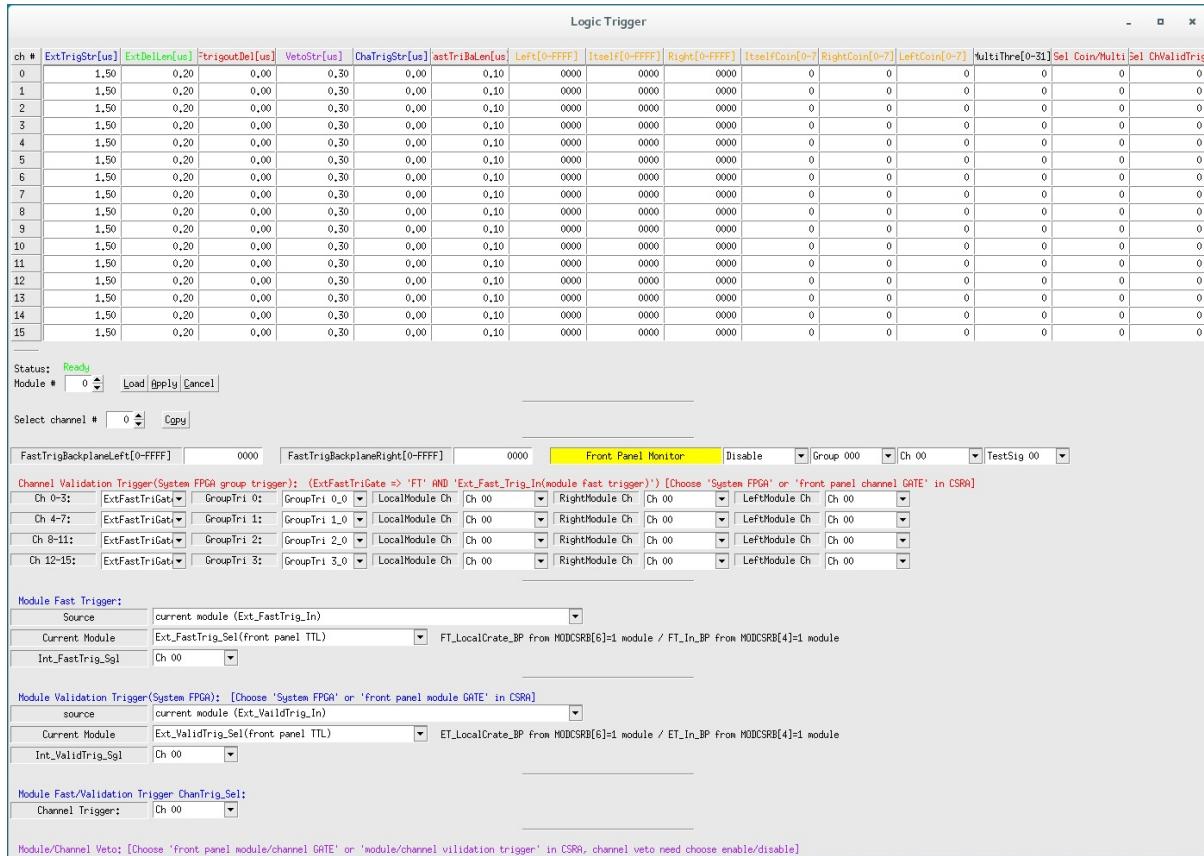
Module #

。 。 TODO。 。

Channel Control Register A affecting each channel individually

- Good channel. Only channels marked as good will have their events recorded. This setting has no bearing on the channel's capability to issue a trigger. There can be a triggering channel whose data are discarded. Channels not marked as good will be excluded from the automatic offset adjustment.
- Histogram energies. When this box is checked, pulse height (energy) computed for each event will be incremented to an energy histogram in the MCA memory.
- Capture trace. When this box is checked, trace will be captured and recorded for each event, along with other list mode information, e.g. timestamp, energy, etc.
- Capture QDC sums. When this box is checked, eight QDC sums will be recorded for each event. QDC sums are consecutive sums of the list mode trace.
- Enable CFD trigger. Check this box to enable this channel's CFD trigger. Otherwise, regular trapezoidal fast trigger will be used.
- Require global external trigger for validation. Check this box to require global external trigger to validate events for this channel.
- Capture raw energy sums and baseline. Check this box to record raw energy sums and baseline value for each event.
- Require channel external trigger for validation. Check this box to require channel external trigger to validate events for this channel.
- Enable pileup rejection. Check this box to enable pileup rejection for this channel. Otherwise, pulses will still be recorded even if they are piled up. **TODO** 这段还需要进一步修改

Logic Set

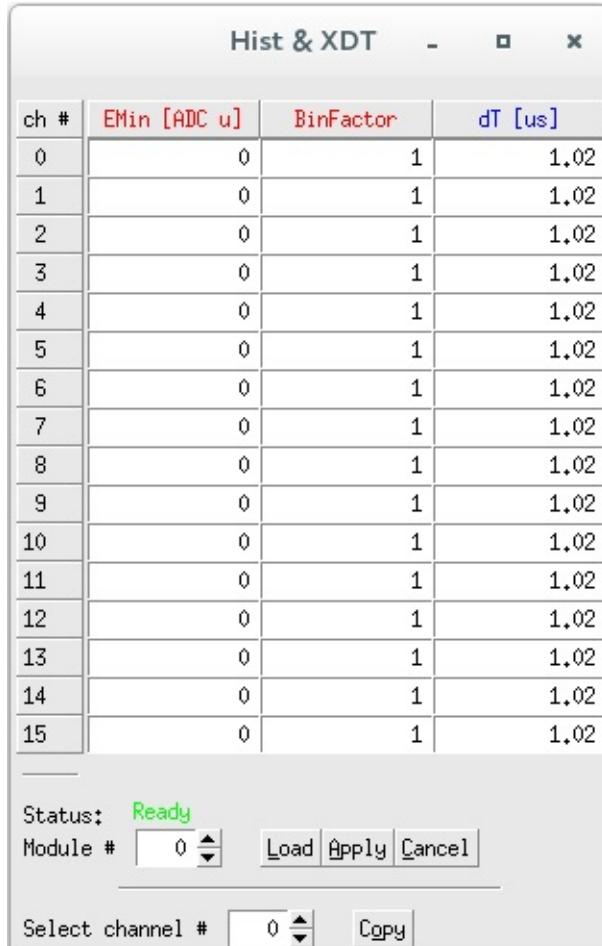


◦◦ TODO◦◦

Monitor 下拉栏

本下拉栏中调节内容为监视波形噪声水平、基线分布等。

Hist & XDT



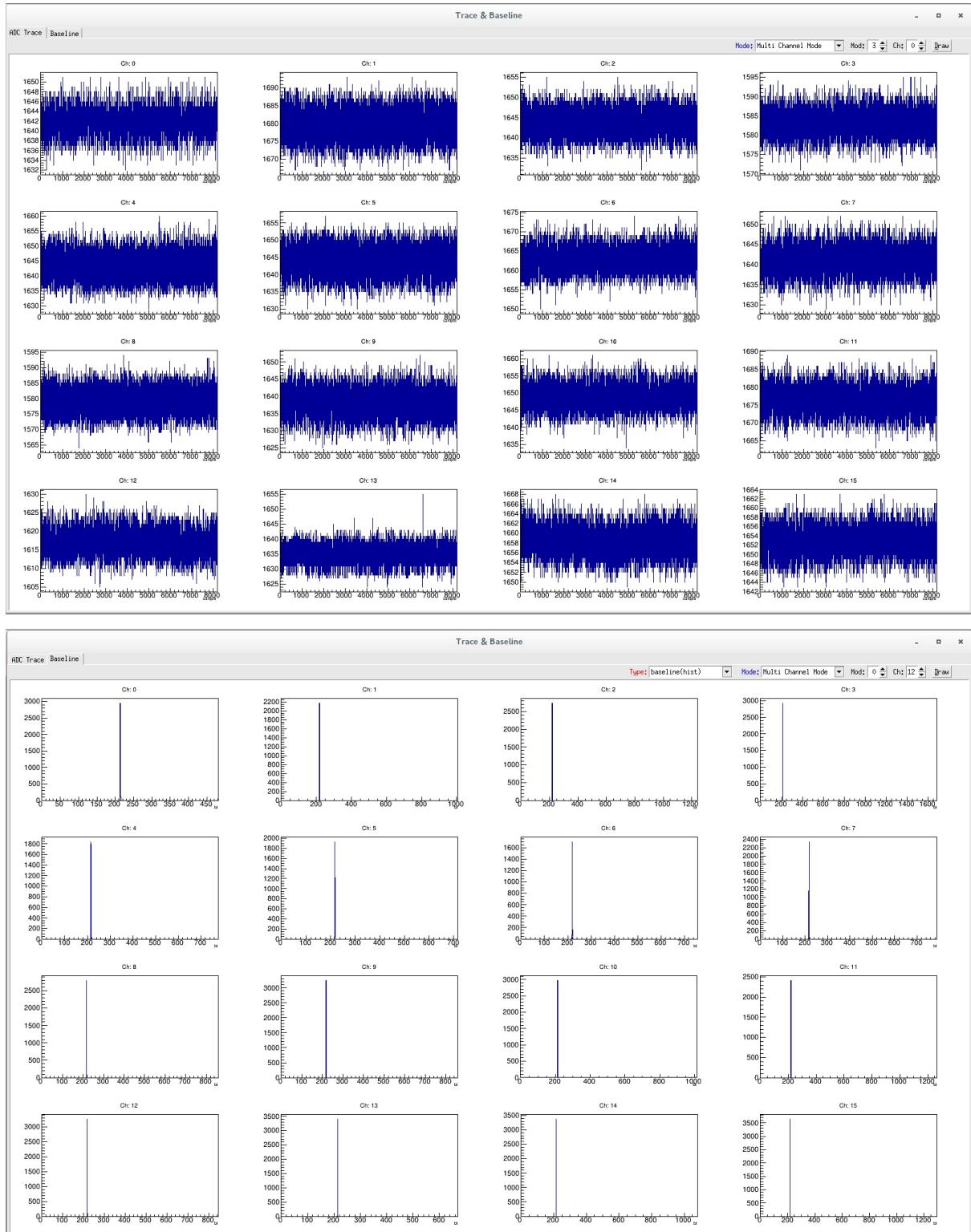
.. TODO ..

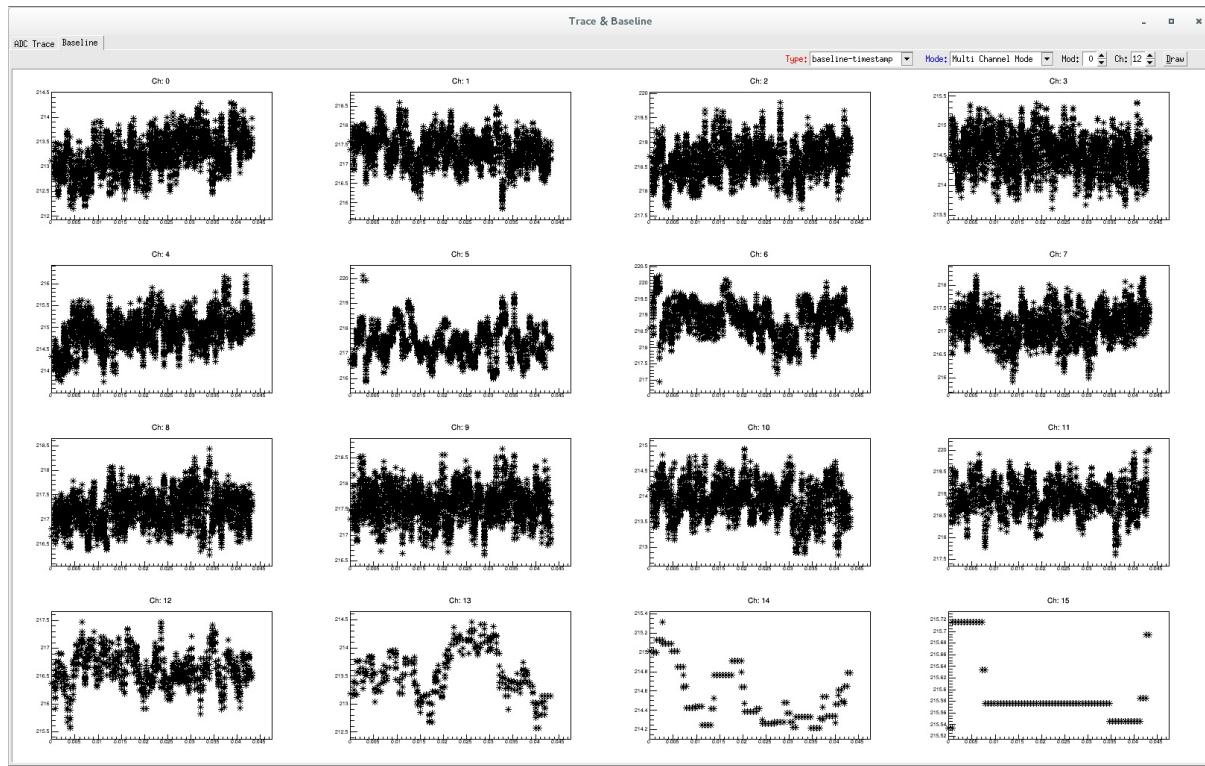
The binning factor controls the number of MCA bins in the spectrum. Energies are computed as 16 bit numbers, allowing in principle 64K MCA bins.

However, spectrum memory for each channel is limited to 32K bins, so computed energy values are divided by 2 binning factor before building the histogram. Binning Factor is usually set to 1, but for low count rates and wide peaks, it might be useful to set it to a larger value to obtain a spectrum with fewer bins, but more counts per bin.

Emin is reserved for a future function to subtract a constant “minimum energy” from the computed energy value before binning to essentially cut off the lower end of the spectrum.

Trace & Baseline



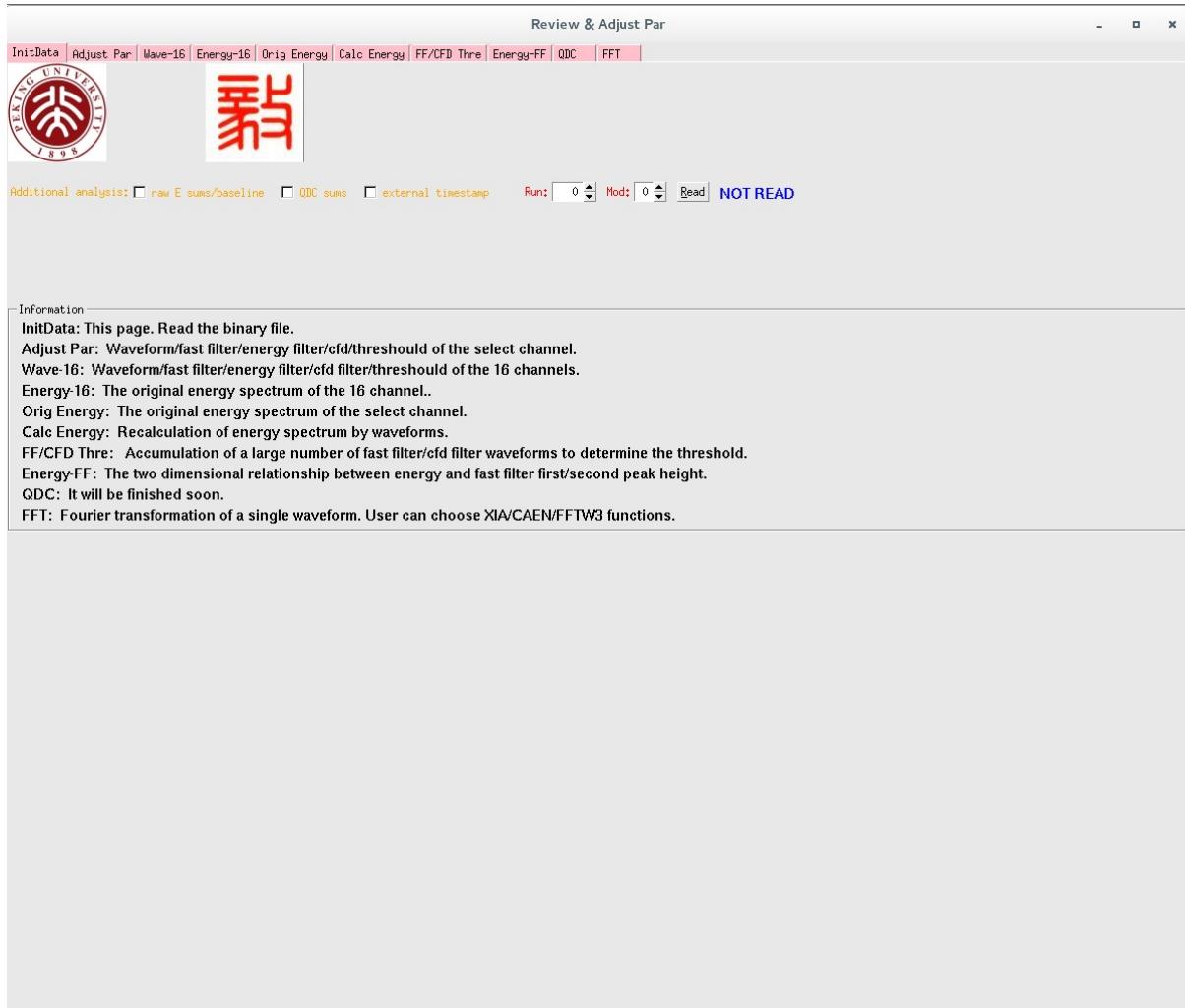


◦◦ TODO ◦◦

Offline 下拉栏

本下拉栏中为离线参数优化调节。

Adjust Par



- InitData: This page. Read the binary file.
 - 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(地址、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。
- 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.

需要添加一个页面功能，选择两路快速看时间分辨。类似于**TechnoAP**界面中的功能。

。。。 TODO。。。 这里需要较大篇幅来介绍。每个子页面的用法需要详细介绍。

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

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。

Online Stattics

修改 **OnlineStattics** 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：

PKU XIA Pixie16 DAQ Online																													
File		CountRate		Alert		EnergyMonitor		Setup		File Path:		File Name:		Complete		RunStop	R0092	M05											
<hr/>																													
Monitor																													
00	0	0	00	113	83	00	25	15	00	0	0	00	0	0	00	00	0	0											
01	0	0	01	26	23	01	18	17	01	0	0	01	0	0	01	00	0	0											
02	0	0	02	22	22	02	22	22	02	0	0	02	0	0	02	00	0	0											
03	0	0	03	23	24	03	22	22	03	0	0	03	0	0	03	00	0	0											
04	0	0	04	19	18	04	21	20	04	0	0	04	0	0	04	00	0	0											
05	0	0	05	25	24	05	19	20	05	0	0	05	0	0	05	00	0	0											
06	0	0	06	25	24	06	21	24	06	0	0	06	0	0	06	00	0	0											
07	0	0	07	26	26	07	19	22	07	0	0	07	0	0	07	00	0	0											
08	0	0	08	13	13	08	23	22	08	0	0	08	0	0	08	00	0	0											
09	0	0	09	14	14	09	23	22	09	0	0	09	0	0	09	00	0	0											
10	0	0	10	17	13	10	21	19	10	0	0	10	0	0	10	00	0	0											
11	0	0	11	21	20	11	17	18	11	0	0	11	0	0	11	00	0	0											
12	0	0	12	21	16	12	14	16	12	0	0	12	0	0	12	00	0	0											
13	0	0	13	27	21	13	14	14	13	0	0	13	0	0	13	00	0	0											
14	0	0	14	27	19	14	13	13	14	0	0	14	0	0	14	00	0	0											
15	0	0	15	35	22	15	14	13	15	0	0	15	0	0	15	00	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												
00	0	0	00	0	0	00	0	0	00	0	0	00	0	0	00	00	0	0											
01	0	0	01	0	0	01	0	0	01	0	0	01	0	0	01	00	0	0											
02	0	0	02	0	0	02	0	0	02	0	0	02	0	0	02	00	0	0											
03	0	0	03	0	0	03	0	0	03	0	0	03	0	0	03	00	0	0											
04	0	0	04	0	0	04	0	0	04	0	0	04	0	0	04	00	0	0											
05	0	0	05	0	0	05	0	0	05	0	0	05	0	0	05	00	0	0											
06	0	0	06	0	0	06	0	0	06	0	0	06	0	0	06	00	0	0											
07	0	0	07	0	0	07	0	0	07	0	0	07	0	0	07	00	0	0											
08	0	0	08	0	0	08	0	0	08	0	0	08	0	0	08	00	0	0											
09	0	0	09	0	0	09	0	0	09	0	0	09	0	0	09	00	0	0											
10	0	0	10	0	0	10	0	0	10	0	0	10	0	0	10	00	0	0											
11	0	0	11	0	0	11	0	0	11	0	0	11	0	0	11	00	0	0											
12	0	0	12	0	0	12	0	0	12	0	0	12	0	0	12	00	0	0											
13	0	0	13	0	0	13	0	0	13	0	0	13	0	0	13	00	0	0											
14	0	0	14	0	0	14	0	0	14	0	0	14	0	0	14	00	0	0											
15	0	0	15	0	0	15	0	0	15	0	0	15	0	0	15	00	0	0											

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，cfds source

TODO 这里添加一个**Branch**截图。。

Front Panel

本页内容待更新。。。。

LVDS I/O port(J101)

普通网口

当前标准固件中还没定义

J155 (letter "A")

输出信号

3.3V I/O port，输出阻抗是 50 欧

A2、D1 地 A1、B1、B2、C1、C2、D2 输出信号

J151-J154

输入信号

differential LVDS signals

- 16 channel gate input
 - 1 module gate input
 - 1 not use current
-

J151-J155

输入信号

single-ended TTL external input signals

FI0、FI2、FI3、FI4、FI6、FI7

group 000

FTRIG_DELAY 采集延迟时间 只要fast filter 过阈值就会产生 FTRIG_VAL 基本同上，有效采集时候才有信号 GLBETRIG_CE stretched external global validation trigger CHANETRIG_CE stretched channel validation trigger，开启CSRA bit13。采集延迟必须在这个时间窗内才能采集到看到的信号实际起始位置在 100 ns，意味着System FPGA 处理时间需要 100 ns？信号宽度由 ChanTrigStretch 控制。

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。

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
lotMap 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 physi
cal 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 slo
t a module is located, any piece of subsequent code can use the expression PXISlotM
ap[ModNum], where ModNum is the logic module number.
// OfflineMode is used to indicate to the API whether the system is running in OFFL
INE mode (1) or ONLINE mode (0). OFFLINE mode is useful for situations where no Pix
ie-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 informa
tion about the Host PC's PCI buses, including the slot enumeration scheme. XIA's so
ftware distribution normally puts this file under the same folder as Pixie-16 softw
are installation folder. However, the user has the flexibility of putting it in oth
er 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 whic
h 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 ea
ch 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 Pi
xie-16 modules when these modules are initialized by function Pixie16InitSystem. Th
is function should be called before a user's application exits.
// If ModNum is set to less than the total number of modules in the system, only th
e module specified by ModNum will be closed. But if ModNum is equal to the total nu
mber 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 81
92. Since the trace data are 16-bit unsigned integers (actually only the lower 14-b
it 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, 4
096 32-bit words are read out first from the DSP, and then each 32-bit word is unpa
cked to form two 16-bit words.
```

```
// Specify the module using ModNum and the channel on the module using ChanNum. Not
e 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 interna
1 memory of a Pixie-16 module. The DSP internal memory is split into two blocks wit
h 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 0x4
9FFF 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 store
d in the configuration files (.set files) in the host. Within this range, 0x4A000 t
o 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 b
y 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 exter
nal memory of a Pixie-16 module. The valid memory address is from 0x0 to 0x7FFFF (3
2-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 o
// ut 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 mod
// e data in ping-pong buffering mode. This function can be used to read list mode dat
// a 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 energi
// es, timestamps, pulse shape analysis values, and waveforms, for each event. Runs wi
// ll continue until a preset number of events are reached or the user terminates the
// run by calling function Pixie16EndRun. Once the run is progress, 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 star
// t the data acquisition this function has to be called for every Pixie-16 module in
// the system. If all modules are to run synchronously, The last module addressed will
// release all others and the acquisition starts then. The first module to end the ru
// n will immediately stop the run in all other modules.
// Use mode=NEW_RUN (=1) to erase histograms and statistics information before laun
// ching 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 dela
// y 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 d
// ata) and 0x103 (energy and timestamp only).
// For Rev-B, C, D modules, there are only one list mode run type supported, that i
// s, 0x100. However, different output data options can be chosen by enabling or disab
// ling different CHANCSRA bits.
// Histograms and statistics data are updated incrementally from run to run provide
// d RESUME_RUN mode is used.
// ModNum is the module number which starts counting at 0. If ModNum is set to be l
// ess than the total number of modules in the system, only the module specified by Mo
// dNum will have its list mode run started. But if ModNum is set to equal to the tota
// l number of modules in the system, then all modules in the system will have their r
// uns 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.
// ModNum is the module number which starts counting at 0.
PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16EndRun (
    unsigned short ModNum );        // Pixie module number

```

```

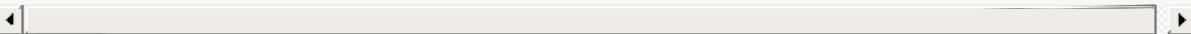
// Compute input count rate
// Use this function to calculate the input count rate on one channel of a Pixie-16

```

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.`

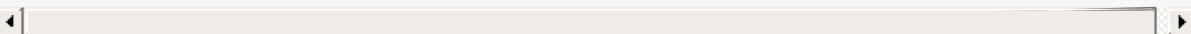
```
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeOutputCountRate (
    unsigned int    *Statistics,
    unsigned short ModNum,
    unsigned short ChanNum );
```



`// 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 );
```

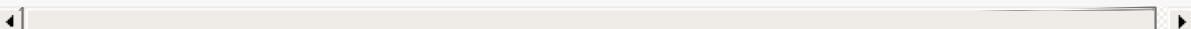


`// Compute number of events processed by a channel`
`// Use this function to calculate the number of events that have been processed by`

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.

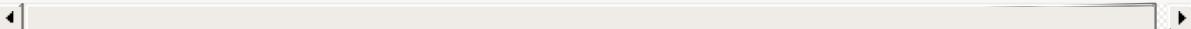
```
PIXIE16APP_EXPORT double PIXIE16APP_API Pixie16ComputeProcessedEvents (
    unsigned int    *Statistics,
    unsigned short ModNum );
```



// 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 Pixie16ReadSglC
// hanBaselines 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
// 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  $\mu$ s
```

```
// 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

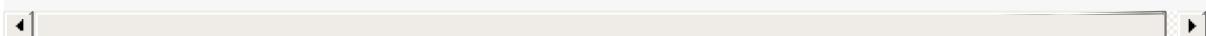
```

```

// 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

```

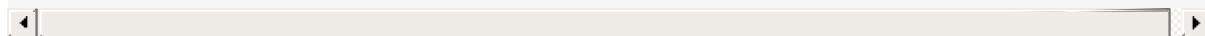


```
// Read histogram data from a module and save to a file
// Use this function to read histogram data from a Pixie-16 module and save the data to a file. New data will be appended to the end of the file. So the same file name can be used for multiple modules and the data from each module will be stored 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 DecimalNu
mber);

// Convert an IEEE 32-bit floating point number to a decimal
PIXIE16APP_EXPORT double PIXIE16APP_API IEEEFloating2Decimal(unsigned int IEEEFloat
ingNumber);

```

```

// 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 dat
a (32-bit unsigned integers) from the external FIFO. So this function essentially e
ncapsulates both functions Pixie16CheckExternalFIFOStatus and Pixie16ReadDataFromEx
ternalFIFO 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). Thi
s 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

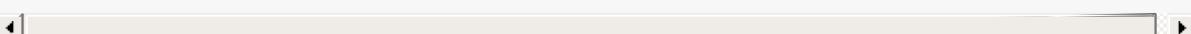
```

```

// 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 mo
dule.
// Specify the module using ModNum. Note that the modules are counted starting at 0.

PIXIE16APP_EXPORT int PIXIE16APP_API Pixie16RegisterIO (
    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 Pixie16CheckExternalFIFOSstatus (
    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 Pixie16CheckExternalFIFOS
// status 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
    te path)
    unsigned short ModuleNumber,       // the module whose events are to be analyzed
    ed
    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
    te path)
    unsigned short ModuleNumber,       // the module whose events are to be analyzed
    ed
    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          pointtobl );
```

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！！！

Decode

- decoder.cc
 - 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

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
 - 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

MakeEvent

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

OnlineStattics

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

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