Pixie-4 Express

User Manual

Version 4.47

February 15, 2018

Hardware Revision: B

Software Revision: 4.47

XIA LLC
31057 Genstar Rd
Hayward, CA 94544 USA
Email: support@xia.com
Tel: (510) 401-5760; Fax: (510) 401-5761
http://www.xia.com/

Table of Contents

Safety

Please take a moment to review these safety precautions. They are provided both for your protection and to prevent damage to the Pixie module and connected equipment. This safety information applies to all operators and service personnel.

Specific Precautions

Observe all of these precautions to ensure your personal safety and to prevent damage to either the Pixie module or equipment connected to it.

Power Source

The Pixie module is powered through a PXI Express (PXIe) chassis. Please refer to the chassis manual for the correct AC voltage connections. The chassis must be powered down to insert and remove the module.

User Adjustments/Disassembly

To avoid personal injury, and/or damage, always turn off power before accessing the Pixie module's on-board switches and jumpers.

Detector and Preamplifier Damage

Because the Pixie module does not provide power for the detector or preamplifier there is little risk of damage to either resulting from the Pixie module itself. Nonetheless, please review all instructions and safety precautions provided with these components before powering a connected system.

Voltage Ratings

Signals on the analog inputs (gold SMA connectors) must not exceed ± 3.5V. Exceptions apply for certain attenuation and termination settings, see Appendix.

Signals on the digital inputs (gold MMCX connector and 10-pin 2mm har-link connector) must not exceed 3.3V.

Servicing and Cleaning

To avoid personal injury, and/or damage to the Pixie module or connected equipment, do not attempt to repair or clean these units. These modules are warranted against all defects for one (1) year. Please contact the factory or your distributor before returning items for service.

Warranty Statement

XIA LLC warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, XIA LLC, at its option, will either repair the defective products without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify XIA LLC of the defect before the expiration of the warranty period and make suitable arrangements for the performance of the service.

This warranty shall not apply to any defect, failure or damage caused by improper uses or inadequate care. XIA LLC shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than XIA LLC representatives to repair or service the product; or b) to repair damage resulting from improper use or connection to incompatible equipment.

THIS WARRANTY IS GIVEN BY XIA LLC WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED OR IMPLIED. XIA LLC AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITYOR FITNESS FOR A PARTICULAR PURPOSE. XIA'S RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. XIA LLC AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER XIA LLC OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Contact Information:

XIA LLC
31057 Genstar Rd.
Hayward, CA 94544 USA

Telephone:	(510) 401-5760
Downloads:	http://support.xia.com
Hardware Support:	support@xia.com
Software Support:	support@xia.com

Manual Conventions

The following conventions are used throughout this manual

Convention	Description	Example
»	The **»** symbol leads you through nested menu items and dialog box options.	The sequence **File»Page Setup»Options** directs you to pull down the **File** menu, select the **Page Setup** item, and choose **Options** from the sub menu.
Bold	Bold text denotes items that you must select or click on in the software, such as menu items, and dialog box options.	...click on the **MCA** tab.
[Bold]	Bold text within [ ] denotes a command button.	**[Start Run]** indicates the command button labeled Start Run.
`monospace`	Items in this font denote text or characters that you enter from the keyboard, sections of code, file contents, and syntax examples.	`Setup.exe` refers to a file called "setup.exe" on the host computer.
"window"	Text in quotation refers to window titles, and quotations from other sources	"Options" indicates the window accessed via **Tools»Options**.
Italics	Italic text denotes a new term being introduced , or simply emphasis	*peaking time* refers to the length of the slow filter. ...it is important first to set the energy filter Gap so that SLOWGAP to *at least one unit greater than* the preamplifier risetime...
<Key> <Shift-Alt-Delete> or <Ctrl+D>	Angle brackets denote a key on the keybord (not case sensitive). A hyphen or plus between two or more key names denotes that the keys should be pressed simultaneously (not case sensitive).	<W> indicates the W key <Ctrl+W> represents holding the control key while pressing the W key on the keyboard
Bold italic	Warnings and cautionary text.	***CAUTION:*** *Improper connections or settings can result in damage to system components.*
CAPITALS	CAPITALS denote DSP parameter names	SLOWLEN is the length of the slow energy filter
SMALL CAPS	SMALL CAPS are used for panels/windows/graphs in the GUI.	…go to the MCADISPLAY panel and you see…

1 Introduction

The Digital Gamma Finder (DGF) family of digital pulse processors features unique capabilities for measuring both the amplitude and shape of pulses in nuclear spectroscopy applications. The DGF architecture was originally developed for use with arrays of multi-segmented HPGe gamma-ray detectors, but has since been applied to an ever broadening range of applications.

The DGF Pixie-4 Express is a 4-channel all-digital waveform acquisition and spectrometer card based on the CompactPCI/PXI Express (PXIe) standard for fast data readout to the host. It combines spectroscopy with waveform capture and on-line pulse shape analysis. The Pixie-4 Express accepts signals from virtually any radiation detector with exponentially decaying pulses, and accommodations can be made for other shapes as well. Incoming signals are digitized by 12-16 bit, 125-500 MSPS ADCs. Waveforms of 8.0-32.0 μs in length for each event can be captured in a first level FIFO, and stored in 256 MB of on-board SDRAM memory organized as a fast FIFO with interrupt driven DMA readout to the host PC. The waveforms are available for onboard pulse shape analysis, which can be customized by adding user functions to the core processing code. 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 32Ki channels. The Pixie-4 Express supports coincidence spectroscopy and can recognize complex hit patterns.

Data readout rates through the CompactPCI/PXI Express backplane to the host computer can reach several hundred MB/s (theoretical max for x4 connection is 800 MB/s). Multiple modules can be read out in parallel with a suitable chassis and host PC. The PXI backplane is also used to distribute clocks and trigger signals between several Pixie-4 Express modules for group operation. With a large variety of CompactPCI/PXI Express processor, controller or I/O modules being commercially available, complete data acquisition and processing systems can be built in a small form factor.

1.1 Pixie-4 Express Features

- Designed for
  - high precision γ-ray spectroscopy with HPGe detectors,
  - timing with fast scintillators (NaI, LaBr3, etc),
  - pulse shape analysis to extract time, position, and/or particle type in segmented or strip detectors, phoswich detectors, or neutron detectors
  - coincidence acquisition
- 12-16 bit, 125-500 MSPS ADC
  Standard versions: 16 bit, 125 MSPS - 14 bit 500 MSPS
- Programmable gain and input offset.
- Programmable pileup inspection criteria include trigger filter parameters, threshold, and rejection criteria.
- Triggered synchronous waveform acquisition across channels and modules.
- Simultaneous amplitude measurement and pulse shape analysis for each channel.
- On-board MCA memory
- Supports x4 PCIe data transfers (several hundred MBytes/second).
- Dead time free acquisition up to the PCIe transfer rate using 256 MB SDRAM memory buffer
- Configurable digital inputs and outputs
- User can customize software, DSP and FPGA firmware, and front end hardware

1.2 Specifications

Front Panel I/O	
Signal Input (4x)	4 analog inputs. Switch selectable input impedance: 50Ω and 2kΩ. Switch selectable input attenuation 1:8 and 1:1 for either impedance. Input range: After termination and attenuation, up to ±2.5V DC can be added to compensate signal DC offsets. After DC offset compensation, signals in the range from 0V to (2V/analog gain) are accepted by the ADC. (See below for analog gain values)
Logic Input/Output	General Purpose I/O connected to programmable logic: 1 MMCX coaxial connector and 1 high density 10-pin connector (single ended or differential)
Backplane I/O	
Clock Input/Output	Distributed 10 and 100 MHz clocks on PXIe backplane.
Triggers	Wired-or bussed trigger on PXIe backplane for synchronous event acquisition.
Synchronization	Wired-or SYNC signal distributed through PXIe backplane to synchronize timers and run start/stop to 50ns.
Veto	Global logic level to suppress event triggering.
Data Interface	
PCI Express	x4 connection to host PC. Theoretical bandwidth 800 MB/s per slot Actual bandwidth: ~495 MB/s PXIe backplane and PC hard drive bandwidth may limit usable data rate.
Digital Controls	
Gain	Coarse gain: 125 MSPS variant: 8 gain settings from 1.6 to 22.6 500 MSPS variant: 2 gain settings: 2 and 5 Other variants: please contact XIA Digital gain: +/-20% gain adjustment for channel matching
Offset	DC offset adjustment from –2.5V to +2.5V, in 65535 steps.
Shaping	Trapezoidal filter with peaking times 0.048 to 63.4 us Adjustable flat top to eliminate ballistic deficit effects
Trigger	Digital trapezoidal trigger filter with adjustable threshold. Rise time and flat top set independently.
Coincidence	Programmable coincidence window: 40 to 1016 ns Reject unwanted hit patterns of the 4 channels
Data collection	MCA limits and number of bins Waveform lengths and pre-trigger delay

Data Outputs	
Spectrum	1024-32768 bins per channel, 32 bit deep (4.2 billion counts/bin). Additional memory for sum spectrum for clover detectors.
Statistics	Real time, counting time, filter dead time, input and throughput counts.
List mode event data	Pulse height (energy), time stamps, pulse shape analysis results, waveform data (up to 4Ki samples) and ancillary data like hit patterns. Buffered in 256MB FIFO.

Table 1-1. Specifications for the Pixie-4 Express

1.3 System Requirements

The digital spectroscopy system considered here consists of a host computer, one or more Pixie-4 modules in a PXIe chassis, and a gamma ray detector with appropriate power supplies.

1.3.1 PXIe Chassis

The Pixie-4 Express can be operated in a peripheral PXIe or PXIe/PXI hybrid slot of any standard 3U PXIe chassis.

1.3.2 Host Computer

The Pixie module communicates with a host computer through a PCI Express (PCIe) interface. The host computer is usually either an embedded PC installed in the PXIe chassis or a remote desktop or laptop that is linked to the chassis via a PCIe bridge.

The host computer must have the following minimum capabilities

- ~100 MB of disk space for operation software
- Sufficient disk space for acquired data
- Windows 7 (32 bit or 64 bit)[1].
  Operation under Linux is possible, please contact XIA for details.
- There are no minimum processor requirements, but a clock rate of 1 GHz or more and memory of 512 MB or more are recommended.

1.3.3 Drivers and Software

Communication between the Pixie module and the host PC is facilitated by Jungo WinDriver (version 11.2). Drivers are provided by the XIA software distribution.

The Pixie Viewer, the default user interface for the Pixie modules, is based on Wavemetrics' Igor Pro. Version 6.22 or higher is required for the Pixie Viewer. Alternative interfaces are LabView, ROOT (under Linux), or command line C programs; demo code is provided on request.

[1] Currently software is still compatible with Windows XP

1.3.4 Detector Signals

The Pixie-4 Express is designed for fast rising, exponentially decaying signals. Step pulses and short non-exponential pulses can be accommodated with specific parameter settings. Staircase type signals from reset preamplifiers generally need to be AC coupled.

Detector signals must not exceed ± 3.5V. Exceptions apply for certain attenuation and termination settings (see Appendix).

1.3.5 Power Requirements

The Pixie-4 Express consumes roughly 25W, requiring the following currents from the PXIe chassis:

3.3V	2.4 A	(3 A)
12V	1.5 A	(2 A)
5V	0.1A	(1 A)

Numbers in brackets show the minimum currents supplied by the chassis per PXIe standard

1.3.6 Connectors and Cabling

The Pixie-4 Express uses SMA connectors for the analog inputs from the detectors. SMA to BNC adapter cables are provided with the module.

A MMCX connector is used for a multi-function digital input/output. A MMCX to BNC adapter cable is provided with the module.

A 10pin har-link connector is used for 10 additional digital inputs and outputs. The pin pitch is 2mm. Matching cables are e.g. Harting 33 27 243 1000 002.

1.4 Software and Firmware Overview

Two levels of software are employed to operate the Pixie modules: a user interface for setup and data acquisition, and a driver layer that handles communication between the user interface and the module. In addition, firmware code is downloaded to the module for the on-board pulse processing functions.

For installation of the software and an introduction to the user interface, please refer to the following chapter of this manual. For details on the driver layer, please refer to the programmer's manual.

1.5 Support

A unique benefit of dealing with a small company like XIA is that the technical support for our sophisticated instruments is often provided by the same people who designed them. Our customers are thus able to get in-depth technical advice on how to fully utilize our products within the context of their particular applications.

Please read through the following sections before contacting us. Contact information is listed in the first few pages of this manual.

2 Installation

2.1 Hardware Setup

Place the Pixie-4 Express modules into any peripheral PXIe or PXIe/PXI hybrid slot with the chassis still powered down, then power up the chassis (Pixie-4 Express modules are not hot swappable). If using a remote controller, be sure to boot the host computer *after* powering up the chassis.

2.2 Software Installation

When the host computer is powered up the first time after installing the controller and Pixie−4 Express modules in the chassis, it will detect new hardware and try to find drivers for it. (A Pixie-4 Express module will be detected as a new device every time it is installed in a new slot.) While there is no required order of installation of the driver software, the following sequence is recommended (users with embedded host computer skip to step 4):

1. If you have a remote controller, first install the driver software for the controller itself. Otherwise, skip to step 4.
   Unless directed otherwise by the manufacturer of the controller, this can be done with or without the controller and Pixie-4 Express modules installed in the host computer and/or chassis. If the modules are installed, ignore attempts by Windows to install drivers until step 7.
   NI controllers come with a multi-CD package called "Device Driver Reference CD". For simplicity it is recommended to install the software on these CDs in the default configuration.
2. Unless already installed, power down the host computer, install the controller in both the host computer and chassis, and power up the system again (chassis first).
3. Windows will detect new hardware (the controller) and should find the drivers automatically. Verify in Window's device manager that the controller is properly installed and has no "resource conflicts".
4. Unless already installed, power down the host computer and install the Pixie-4 Express modules in the chassis. Check the input switch settings for the appropriate signal termination: 50 Ω or 2 kΩ (see section 10.1 for details). Then power up the system again (chassis first). Ignore attempts by Windows to install drivers until step 7.
5. Install Igor Pro, version 6.22 or higher.
   (If installing Igor Pro version 7, be sure to use the 32bit version)
6. Install the Pixie-4 Express software provided by XIA.
   The CD-ROM with the Pixie-4 Express software distribution contains the installation program (for version XXX)

   Pixie-4e_XXX_setup.exe

   Run the setup program and follow the instructions shown on the screen to install the software to the default folder selected by the installation program, or to a custom folder. Driver installation is rather lengthy (the PC may pause for extended periods

of time) and may require acknowledgment of unsigned drivers. The installation folder will contain the IGOR control program (Pixie.pxp), its source code (XIA_Functions.ipf and XIA_Panels.ipf), online help files and 9 subfolders (Configuration, Data, Doc, Drivers, DSP, Firmware, MCA, PixieClib, and PulseShape). Make sure you keep this folder organization intact, as the Pixie Viewer relies on this. Feel free, however, to add folders and subfolders at your convenience.

7. Windows will detect new hardware (the Pixie-4 Express modules) and should find the drivers automatically. If not, direct it to the "drivers" directory in the Pixie-4 Express software distribution installed in step 6. Verify in Window's device manager that the modules are properly installed as "Pixie4e RevB (…)" under Jungo devices and have no "resource conflicts".

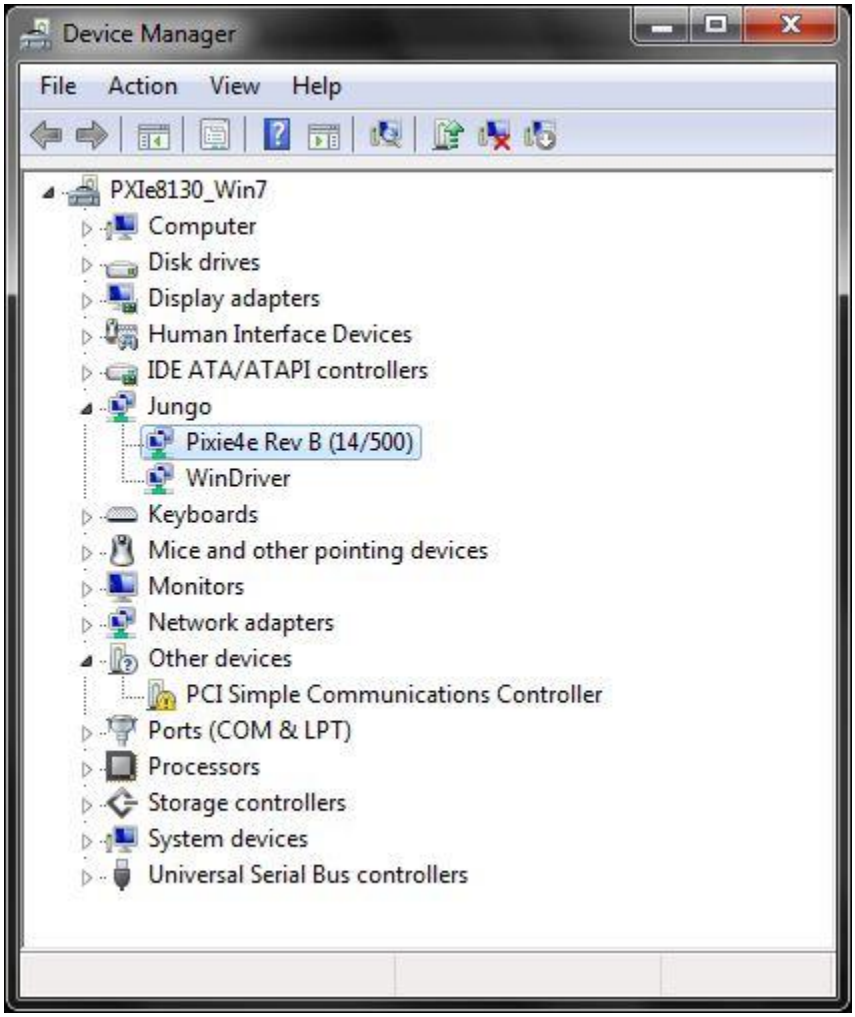


Figure 2-1: Screenshot of Windows device manager with Pixie-4 Express. On 64 bit Windows, the name changes to "Pixie-4e"

2.3 Getting Started

To start the Pixie Viewer, double-click on the file "Pixie.pxp" in the installation folder.

The Pixie Viewer offers the following online help options and short cuts:

- All panels have a [**Help**] button that opens a help window with description of the panel's controls.
- When hoovering over most controls, a brief explanation is displayed in the lower left of the Igor window.
- Frequently used panels can be opened using the functions keys F2-F12. See top menu **XIA** for a list of panels.
- In graphs, use <ctrl>-i to toggle a cursor bar (place cursor on plot to see data point details)
- In graphs, left click and drag to define an area, then right click into the area for a zoom/expand menu.

After IGOR loaded the Pixie Viewer, the START UP panel should be prominently displayed in the middle of the desktop. In the panel, first specify the **number** N of Pixie modules in the system. The **chassis type** should be PXIe-1062 for any PXIe chassis. Then specify the serial numbers of the modules – this allows addressing the modules from 0 to N-1 independent of the physical slot location in the chassis.

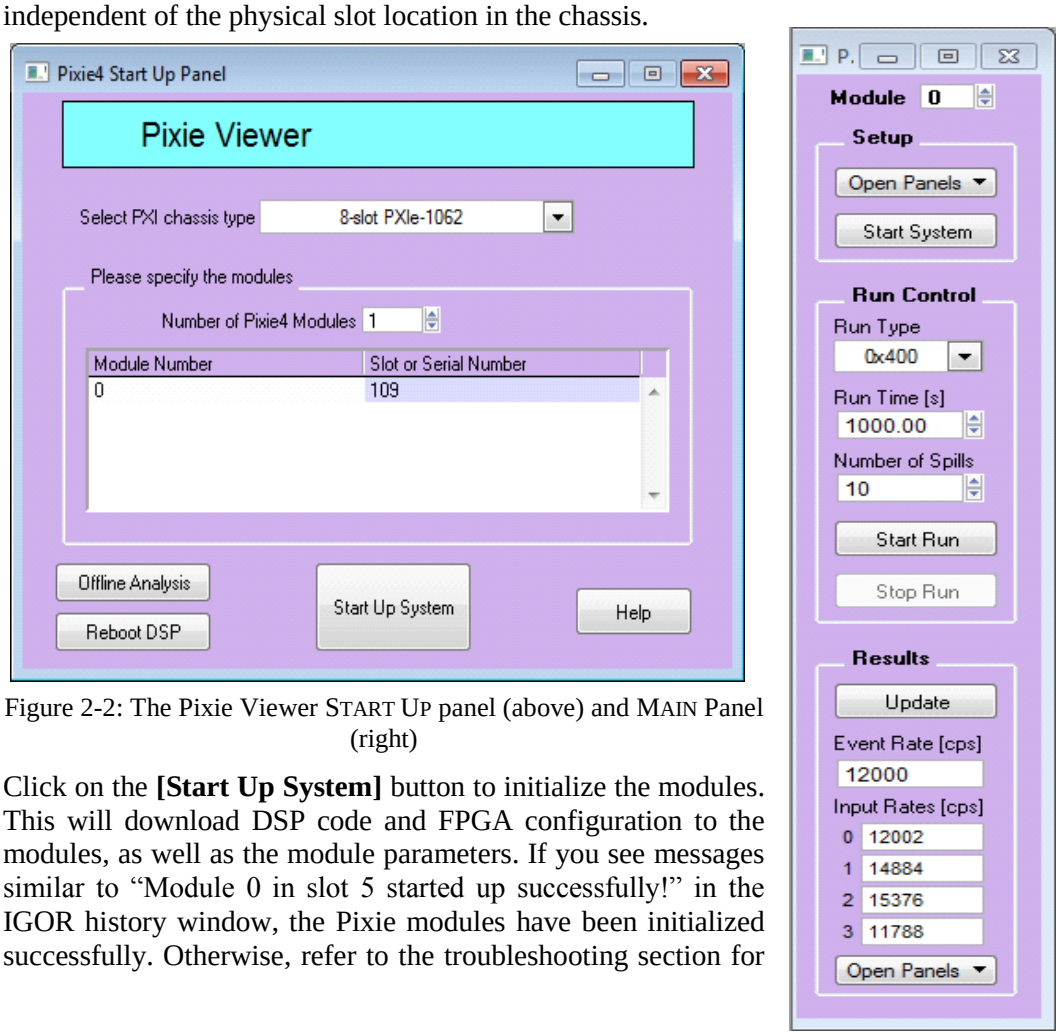


Figure 2-2: The Pixie Viewer START UP panel (above) and MAIN Panel (right)

Click on the **[Start Up System]** button to initialize the modules. This will download DSP code and FPGA configuration to the modules, as well as the module parameters. If you see messages similar to "Module 0 in slot 5 started up successfully!" in the IGOR history window, the Pixie modules have been initialized successfully. Otherwise, refer to the troubleshooting section for

possible solutions. If you want to try the software without a chassis or modules attached, click on [**Offline Analysis]**.

After the system is initialized successfully, you will see the Main control panel that serves as a shortcut to the most common actions and from which all other panels are called. Its controls are organized in three groups: **Setup, Run Control**, and **Results**.

In the **Setup** group, the [**Start System**] button opens the START UP panel in case you need to reboot the modules. The **Open Panels** popup menu leads to four panels where parameters and acquisition options are entered. They are described in more detail in section 3 and in the online help. To get started, select **Parameter Setup**, which will open (or bring to front) the PARAMETER SETUP panel shown in Figure 2-3. For most of the actions the Pixie Viewer interacts with one Pixie module at a time. The number of that module is displayed at the top of the Main panel and the top right of the PARAMETER SETUP panel. Proceed with the steps below to configure your system.

Note: The [**More]** or **[Less]** button next to the [**Help]** button on the bottom of the PARAMETER SETUP panel can be used to hide some controls. This may be helpful to first-time Pixie users who want to focus on the most essential settings.

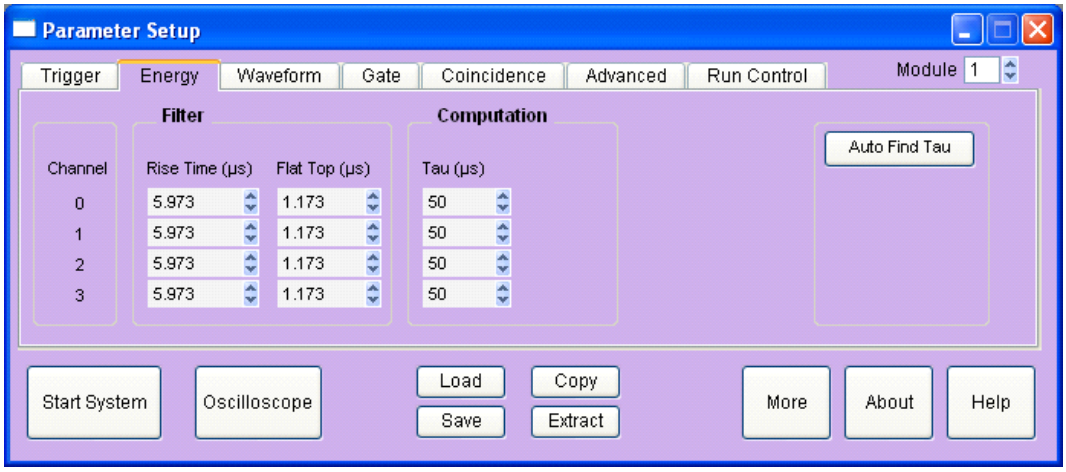


Figure 2-3: The PARAMETER SETUP Panel, Energy tab shown

For an initial setup, go through the following steps:

1. If not already visible, open the PARAMETER SETUP panel by selecting **Parameter Setup** from the **Open Panel** popup menu in the MAIN panel.
2. At the bottom of the PARAMETER SETUP panel, click on the [**Oscilloscope]** button. This opens a graph that shows the untriggered signal input. (Figure 2-4)
   In the OSCILLOSCOPE panel, click [**Refresh]** to update the display. The pulses should fall in the range 0-16Ki[2] for modules with 14 bit ADCs, 0-64Ki for modules with 16 bit ADCs. If no pulses are visible or if they are cut off at the upper or lower range of the display, click [**Adjust Offsets]** to automatically set the DC offset. If the pulse amplitude is too large to fall in the display range, decrease the **Gain**. If the pulses have falling leading edges, toggle the **Invert** checkbox. For the 500 MHz

[2] In this manual, we use the IEC notation "Ki" for 1024, and the SI notation "k" for 1000.

version, if the signal looks unusually noisy, click on the [**Calibrate]** button to automatically match gain and offset of the 2 ADC cores.

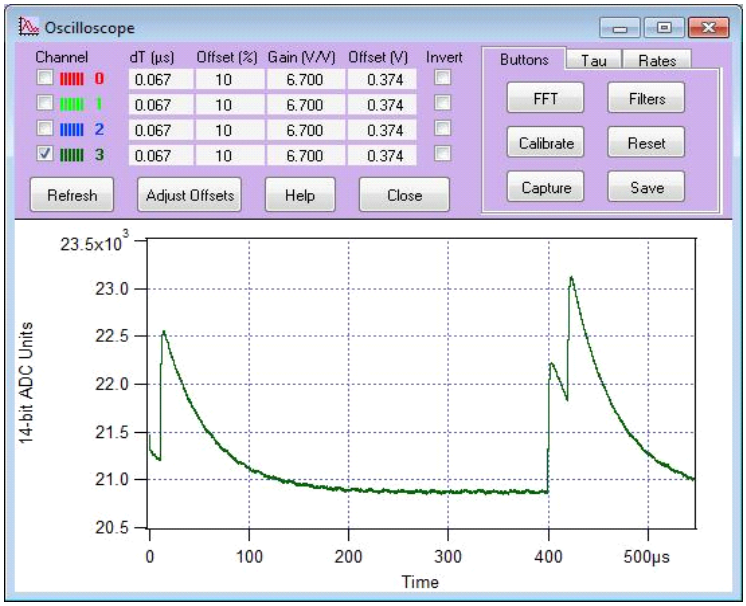


Figure 2-4: OSCILLOSCOPE panel

3. In the **Energy** tab of the PARAMETER SETUP panel, input an estimated preamplifier exponential RC decay time for **Tau**, and then click on [**Auto Find Tau]** to determine the actual Tau value for all channels of the current module. You can also enter a known good Tau value directly in the **Tau** control field, or use the controls in the OSCILLOSCOPE to manually fit Tau for a pulse.
4. Save the Igor experiment using **File -> Save Experiment As** from the top menu. This saves the current state of the interface with all open panels and the settings for file paths and slot numbers (the settings independent of module parameters). Igor will also prompt you to ...
5. … save the modified parameter settings to file. (To do so at any other time, click on the [**Save]** button at the bottom of the PARAMETER SETUP panel to open a save file dialog.) Create a new file name to avoid overwriting the default settings file.
6. Click on the **Run Control** tab, set **Run Type** to "0x301 MCA Mode", **Poll time** to 1 second, and **Run time** to 30 seconds or so, then click on the [**Start Run]** button. A spinning wheel will appear occasionally in the lower left corner of the screen as long as the system is waiting for the run to finish. If you click the [**Update]** button in the MAIN panel, the count rates displayed in the Results group are updated.
7. After the run is complete, select **MCA Spectrum** from the **Open Panels** popup menu in the Results group of the MAIN panel. The MCA SPECTRUM graph shows the MCA histograms for all four channels. You can deselect other channels while working on only one channel. After defining a range in the spectrum with the cursors and setting the fit option to **fit peaks between cursors**, you can apply a Gauss fit to the spectrum by selecting the channels to be fit in the **Fit** popup menu. You can alternatively enter the fit limits using the **Min** and **Max** fields in the table or by specifying a **Range** around the tallest peak or the peak with the highest energy. To scale the spectrum in keV, enter the appropriate ratio in the field **keV/bin**.

At this stage, you may not be able to get a spectrum with good energy resolutions. You may need to adjust some settings such as energy filter rise time and flat top as described in section 3.5.

3 Navigating the Pixie Viewer

3.1 Overview

The Pixie Viewer consists of a number of graphs and control panels, linked together by the MAIN control panel. The Pixie Viewer comes up in exactly the same state as it was when last saved to file using **File->Save Experiment**. This preserves settings such as the file paths and the slot numbers entered in the START UP panel. However, the Pixie module itself loses all programming when it is switched off (powered down). When the Pixie module is switched on again, all programmable components need code and configuration files to be downloaded to the module. Clicking on the [**Start Up System]** button in the START UP panel performs this download. Below we describe the concepts and principles of using the Pixie Viewer. Detailed information on the individual controls can be found in the Online Help for each panel. The operating concepts are described in sections 4-7.

The controls in the MAIN control panel are organized in three groups: **Setup**, **Run Control**, and **Results**. In the **Setup** and **Results** groups, popup menus lead to the panels and graphs indicated in Figure 3.1

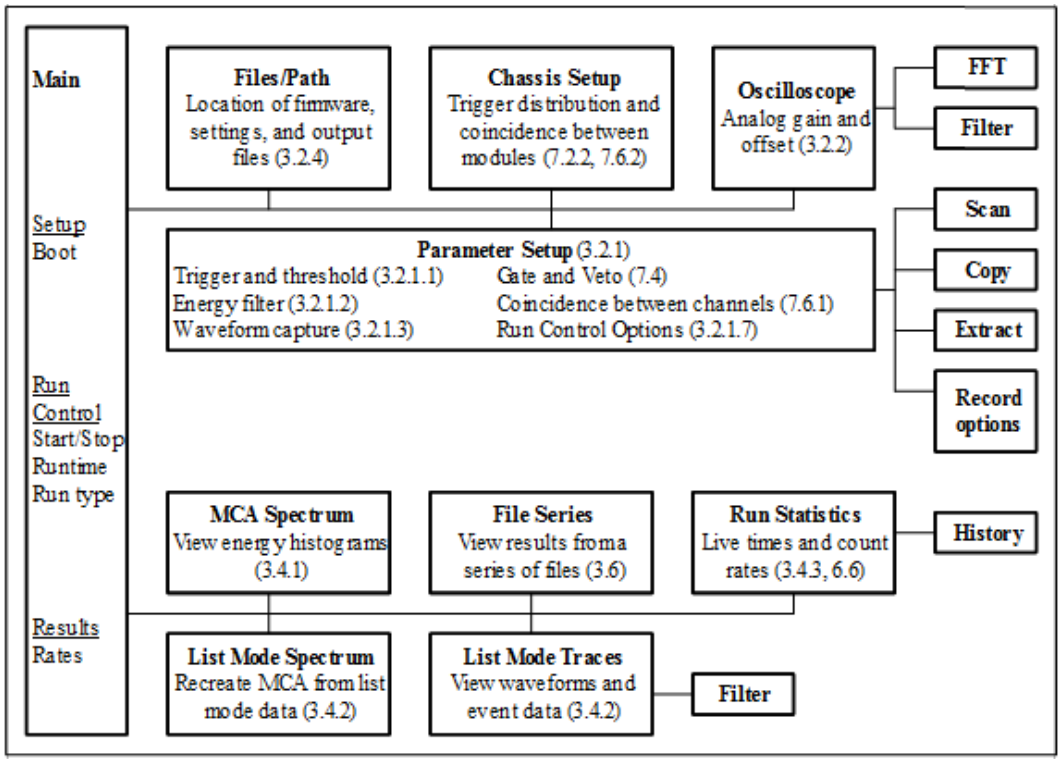


Figure 3-1: Block diagram of the major panels in the Pixie Viewer. Numbers in brackets point to the corresponding section in the user manual. All panels are described in detail in the online help.

3.2 Setup Group

In the setup group, there is a button to open the START UP panel, which is used to boot the modules. The **Open Panels** popup menu leads to one of the following panels: PARAMETER SETUP, OSCILLOSCOPE, CHASSIS SETUP, FILES/PATHS

3.2.1 PARAMETER SETUP Panel

The PARAMETER SETUP panel is divided into 7 tabs, summarized below. Settings for all four channels of a module are shown in the same tab. At the upper right is a control to select the module to address. At the bottom of the panel is a [**More]** button, which will make all advanced panel controls visible as well.

The Pixie modules being digital systems, all parameter settings are stored in a settings file. This file is separate from the Igor experiment file, to allow saving and restoring different settings for different detectors and applications. Parameter files are saved and loaded with the corresponding buttons at the bottom of the PARAMETER SETUP panel. After loading a settings file, the settings are automatically downloaded to the module. At module initialization, the settings are automatically read and applied to the Pixie module from the last saved settings file.

In addition there are buttons to copy settings between channels and modules, and to extract settings from a settings file. Two large buttons at the lower left duplicate the buttons to call the START UP panel and the OSCILLOSCOPE.

3.2.1.1 Trigger Tab

The **Trigger** tab contains controls to set the trigger filter parameters and the trigger **threshold**, together with checkboxes to enable or disable trigger and to control trigger distribution (see section 7.2.1). Except for the threshold, the trigger settings have rarely to be changed from their default values.

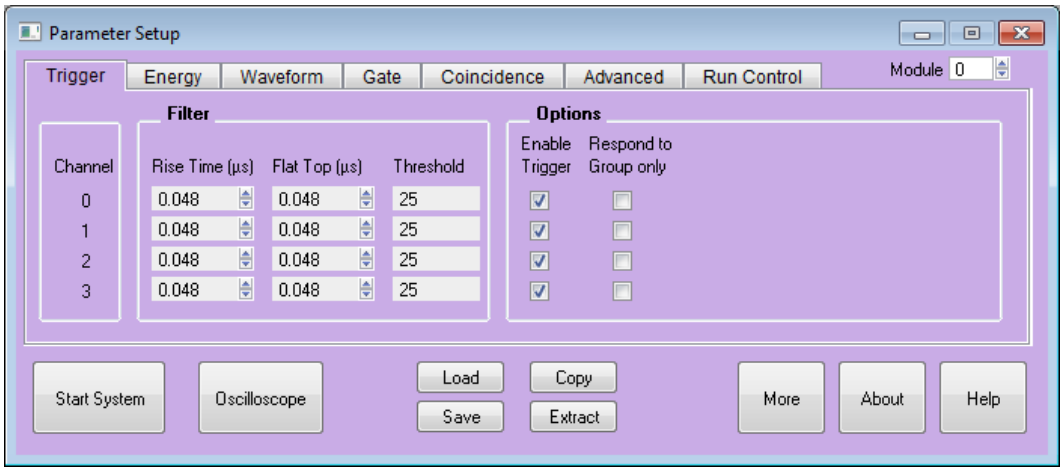


Figure 3-2: The Trigger tab of the PARAMETER SETUP panel.

The threshold value corresponds to ¼ of the pulse height in ADC steps, e.g. with a threshold of 20, triggers are issued for pulses above 80 ADC steps. This relation is true if the trigger filter **rise time** is large compared to the pulse rise time and small compared to the pulse decay time. A pulse shape not meeting these conditions has the effect of raising the effective threshold. For a modeled behavior of the trigger, you can open displays from the OSCILLOSCOPE and the LIST MODE TRACES panels that show trigger filter and threshold

computed from acquired waveforms using the current settings. The threshold value is scaled with the trigger filter **rise time**, therefore it is not limited to integer numbers.

3.2.1.2 Energy Tab

The **Energy** tab contains the settings for the energy filter and the subsequent computation. These settings are most important for obtaining the best possible energy resolution with a Pixie system. The energy filter **rise time** (or peaking time) essentially sets the tradeoff between throughput and resolution: longer filter **rise times** generally improve the resolution by averaging out noise (up to a certain optimum) but reduce the throughput because more time is required to measure each pulse. The pulse decay time **Tau** is used to compensate for the decay of a previous pulse in the computation of the pulse height. You can enter a known good value, or click on [**Auto Find Tau]** to let the Pixie Viewer determine the best value.

The advanced controls in this tab contain functions to modify the energy computation and to acquire a series of measurements with varying filter settings and decay times to find the best settings. For a detailed description of the filter operation, see section 6.

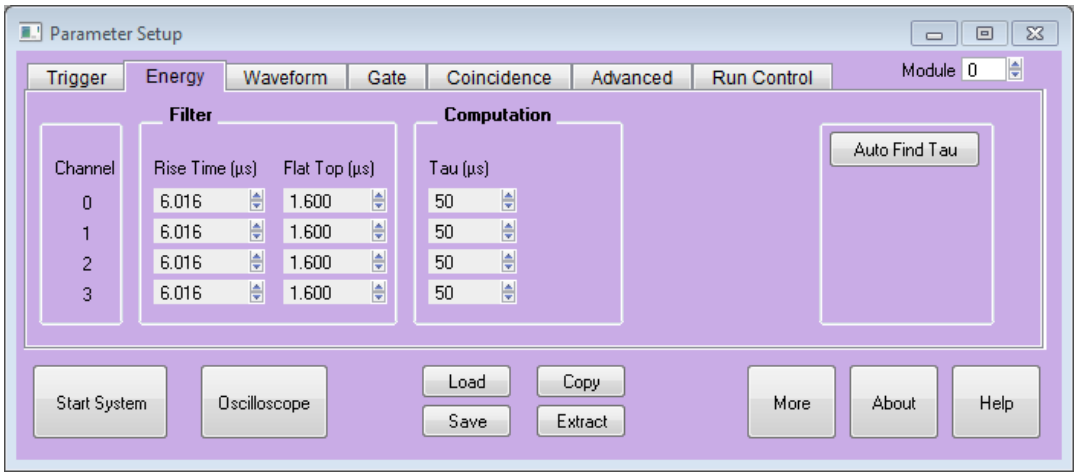


Figure 3-3: The Energy tab of the PARAMETER SETUP Panel.

3.2.1.3 Waveform Tab

The **Waveform** tab contains the controls to set the length and pre-trigger delay of the waveforms to be acquired. Advanced options include parameters for online pulse shape analysis

3.2.1.4 Gate Tab

The **Gate** tab contains the controls to set the window for gating acquisition with external signals. We define *Veto* as a signal distributed to all modules and channels, but each channel is individually enabled to require or ignore this signal. *Veto* is active during the validation of a pulse (after pileup inspection), an energy filter rise time plus flat top after the rising edge. With suitable external logic, the decision to veto a pulse can be made from information obtained at the rising edge of the pulse (e.g. multiplicity from several channels) and therefore this function is also sometimes called Global First Level Trigger (GFLT). In contrast, *Gate* signals are understood as individual signals for each channel, and they are active at the rising edge of the detector pulse.

For a detailed description of the *Veto* and *Gate* operation, see section 7.4.

3.2.1.5 Coincidence Tab

The **Coincidence** tab contains the controls to set the acceptable hit pattern, and the coincidence window around the rising edge during which channels can contribute to the hit pattern. There is a checkbox for each possible hit pattern. For example, if the checkbox with pattern 0100 is checked, events with a hit in channel 2 and no others are accepted. Selecting multiple checkboxes accepts combinations of hit patterns, e.g. any event with exactly one channel hit.

For a detailed description of the coincidence operation, see section 7.2.1. Controls for coincidences between modules are located in the CHASSIS SETUP Panel and described in section 7.2.2.

3.2.1.6 Advanced Tab

The **Advanced** tab contains the controls for modifying the pileup inspection, histogram accumulation, baseline measurements, and ADC calibration. The ADC used on 500 MHz versions of the Pixie-4 Express actually consists of two ADC cores on a single IC, which need to be calibrated for matched gain, offset and phase. Normally, these calibration settings are read from the module's non-volatile memory at boot time, but sometimes, for example at temperature changes, it may be required to recalibrate the cores. An indication of mismatch are systematic offsets between odd and even samples. These controls are repeated in the OSCILLOSCOPE panel.

3.2.1.7 Run Control Tab

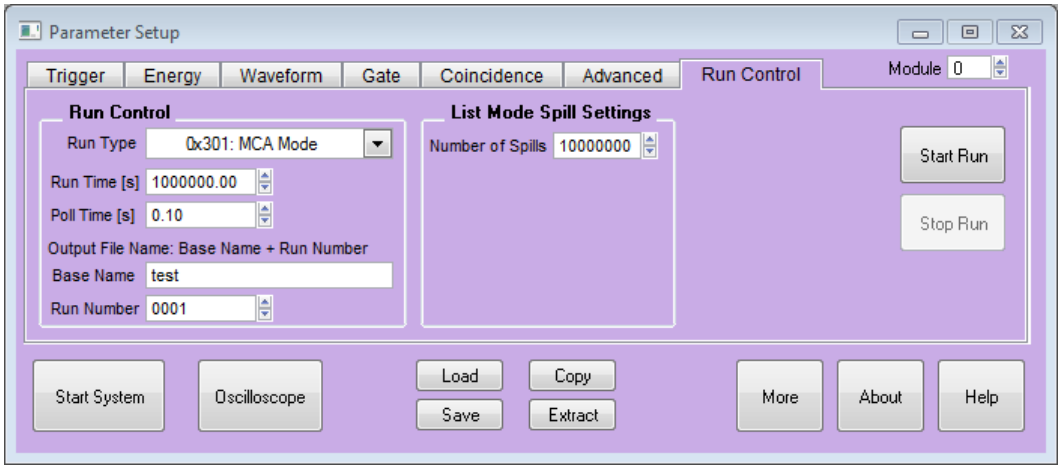


Figure 3-4: The Run Control tab of the PARAMETER SETUP Panel.

The **Run Control** tab defines the settings for data acquisition. The "**Run Type**" popup menu selects MCA or list mode runs, see section 4 for a detailed description. In addition, there are controls

- to set the run time (length of data acquisition as measured by Igor),
- to set the polling time (period for checking how many buffers of list mode data have been written to disk and/or if run time is reached),
- to specify the data file name (a base name plus 4-digit run number that can be made to increment automatically), and

- to specify the number of spills in list mode runs. (In list mode runs, data is transferred in 2MB buffers to the host PC. We call each such buffer transfer a spill. The number of spills thus sets the amount of data to collect.)

The [**Start Run]** and [**Stop Run]** buttons from the MAIN control panel are duplicated here as well.

Advanced options include settings for synchronizing acquisition between modules, controls to set a timeout for each spill, and the spill readout mode, and a button to open a panel with advance record options.

3.2.2 OSCILLOSCOPE

As mentioned in section 2.3, the OSCILLOSCOPE (Figure 2.3) is used to view untriggered traces as they appear at the ADC input and to set all parameters relating to the analog gain and offset. There are controls titled

- **dT [us],** which sets the time between samples in the oscilloscope (there are always 8192 samples in the oscilloscope window),
- **Offset [%],** which sets the target DC-offset level for automatic adjustment,
- **Gain (V/V),** which sets the analog gain before digitization, and
- **Offset (V),** which directly sets the offset voltage.

The traces from different channels are not acquired synchronously but one after the other. Therefore even if coincident signals are connected to the Pixie-4 Express inputs, the Oscilloscope will show unrelated pulses for each channel.

There are also buttons and controls to

- open a display of the [**FFT]** of the input signal, which is useful to diagnose noise sources
- open a display of the waveforms of the trigger [**filter]** and energy filter computed from the traces in the oscilloscope
- repeat the action of the [**Refresh]** button until a pulse is [**captured]**. This is useful for low count rates.
- **Fit** the pulses in the OSCILLOSCOPE with an exponential decay function to determine the decay time Tau, and to **accept** the fit value for the module settings.
- View the current input count rate and the current fraction of time the signal is out of range. These values are updated in the DSP every ~2-3ms independent of whether a run is in progress or not. Their precision is in the order of 5-10%, or 50 cps.
- **[Calibrate]** the ADC gain and offset matching of its two cores. Calibrations are reset at every power cycle or reboot of the module, or by clicking the [**Reset]** button. The process started with this button will measure the mismatch, then modify the gain and offset match in an iterative process.
- **[Save]** ADC waveforms to file in Igor text format (ASCII with header and footer scaling info)

3.2.3 FILES/PATHS

The firmware files, DSP files and settings files are defined in the FILES/PATHS panel. Changes will take effect at the next reboot, e.g. when clicking the [**Start Up System]** or

[**Reboot DSP]** buttons in this panel or in the START UP panel. There is also a button to set the files and paths to the default, relative to the home path of the file Pixie.pxp.

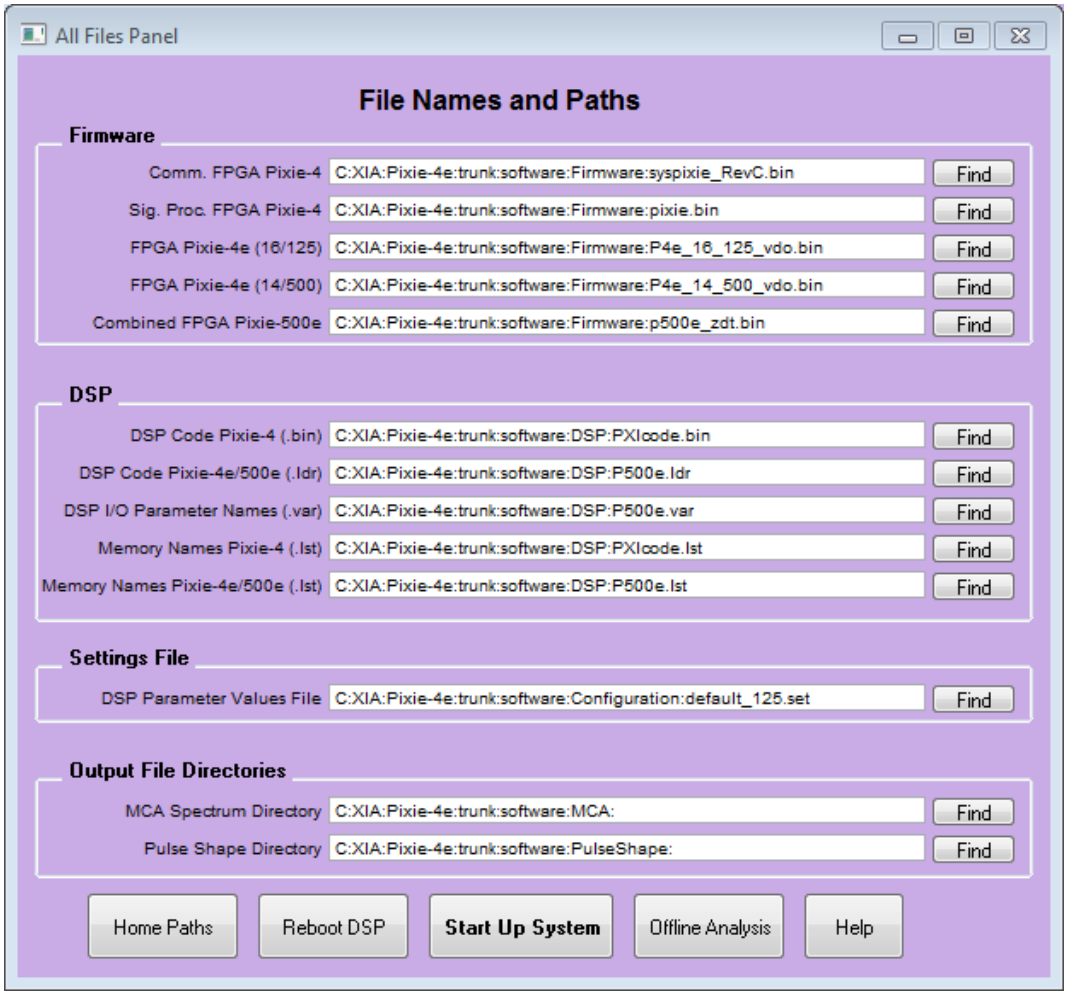


Figure 3-5: The FILES/PATHS Panel.

3.2.4 CHASSIS SETUP

The CHASSIS SETUP panel is used to set parameters that affect the system as a whole. Examples are trigger distribution between modules, coincidence settings between modules, and the operation of the Pixie-4 Express's front panel input. See sections 7.2.2 and 7.6.2 for details.

3.3 Run Control Group

The Run Control group in the MAIN control panel has the most essential controls to start and stop runs, and to define or monitor the run time and the number of spills. For more options, use the **Run Control** tab of the PARAMETER SETUP panel.

3.4 Results Group

The Results group of the MAIN control panel displays the count rates of the current or most recent run. Click Update to refresh these numbers. Note that the **event rate** is the sum of

the output count rates of the four channels in run types 0x400 and 401, but the rate of 4-channel events in Run Type 0x402. .

The popup menu **Open Panels** leads to panels to view the output data from the data acquisition in detail. These panels are the MCA SPECTRUM display, THE LIST MODE TRACES display, the LIST MODE SPECTRUM display, the RUN STATISTICS, and a panel to display results from a series of files.

3.4.1 MCA SPECTRUM

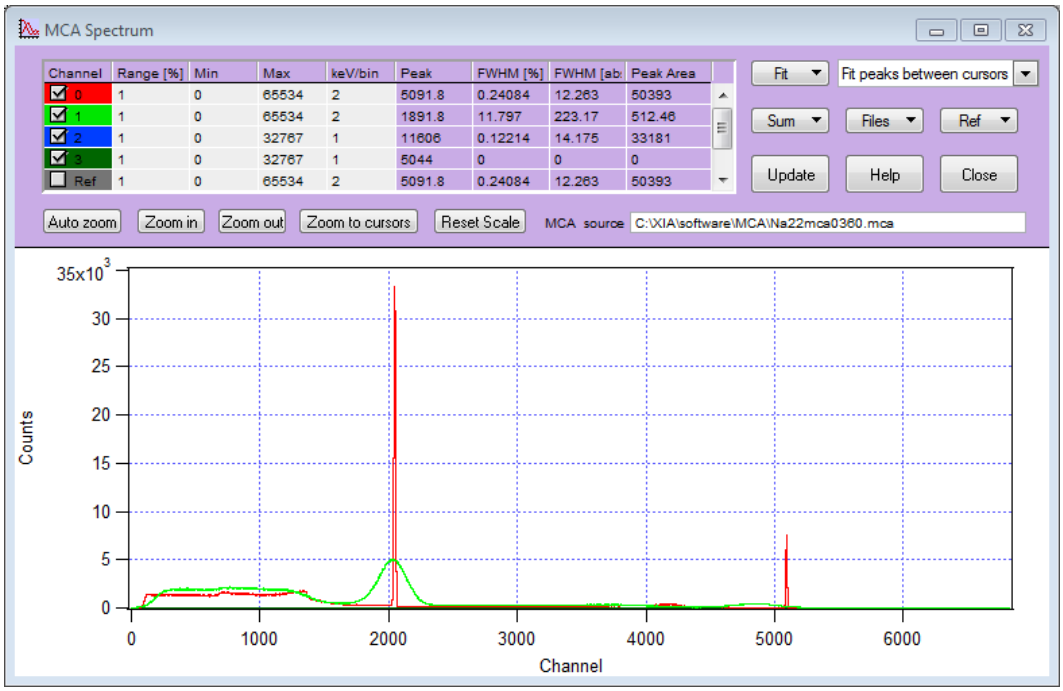


Figure 3-6: The MCA SPECTRUM display

The MCA SPECTRUM display shows the spectra accumulated in on-board memory or from a .mca file saved at the end of a run. Spectrum analysis is limited to fitting peaks with a Gaussian and computing the peak resolution. There are several options to define the fit range, as described in the online help. Spectra can be saved as text files or .CHN files for import into other applications.

3.4.2 LIST MODE TRACES and LIST MODE SPECTRUM

The LIST MODE TRACES display shows the data from the binary list mode files (.bin or .b##). If waveforms were collected, they are shown in the graph section of the panel. Event and channel header information – energy, time stamps, and hit patterns as described in section 4.1.2 – are shown in the fields above the graph section. Key information bits of the hit pattern are decoded in checkboxes below the hexadecimal value. Resizing of the panel may be required to see all features.

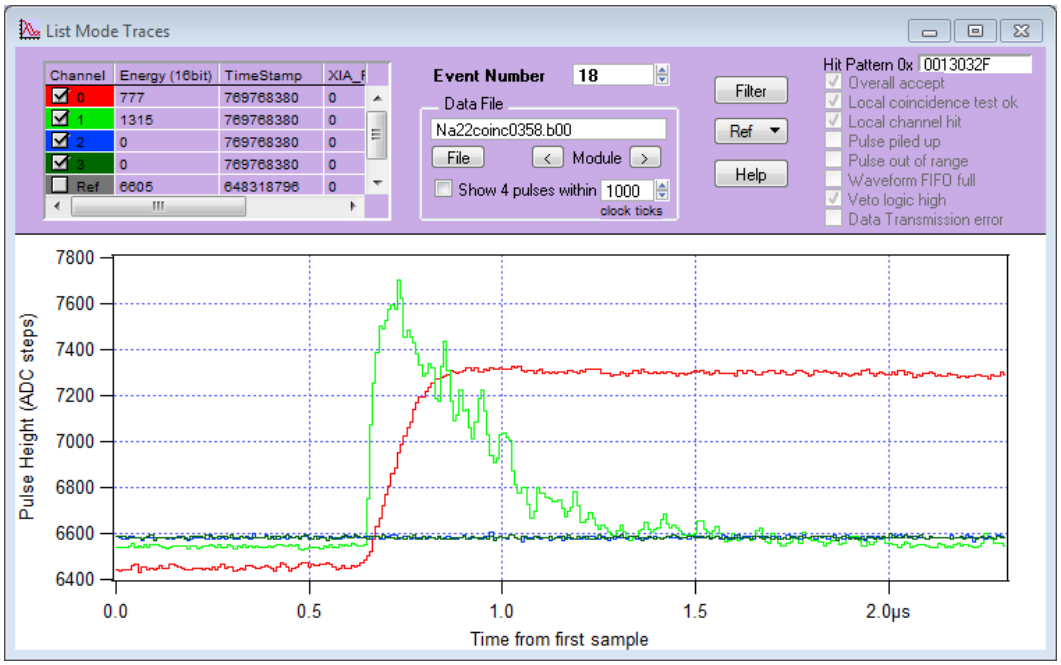


Figure 3-7: The LIST MODE TRACES display

To view event data, first specify a data file with the [**Find]** button. The file for module 0 of the most recent run is selected by default. Arrow buttons allow changing of both module number of the Pixie Viewer and the file name at the same time. You can then select an event to view by entering its number in the **Event Number** field. Events are stored as single-channel records in run type 0x400 or 4-channel records in run type 0x402. To display data from coincident pulses in run type 0x400, check the box **Show 4 pulses** and enter a coincidence window in clock ticks. You will have to increment the **Event Number** up to 4 times to get new data. The waveform from the current event is displayed in bold.

The **Ref** popup menu allows one of the current 4 channels to be saved for comparison; check the corresponding dark gray box in the table to add its waveform to the plot.

The button [**Digital Filters]** opens a new plot that shows the response of the trigger filter and energy filter computed from the list mode waveforms. This plot is more precise than the related graph opened from the OSCILLOSCOPE since it uses the same full rate data as the filters implemented in the module, not the reduced rate sampled at the OSCILLOSCOPE's **dT**. However, unless long list mode traces are acquired or energy filters are short, there may not be sufficient data to compute the energy filter properly.

The LIST MODE SPECTRUM display is a plot similar to the MCA SPECTRUM, but it is computed from the energies saved in the list mode data file. Since energies are stored there in full 16 bit precision, binning can be made finer than in the MCA SPECTRUM, which is limited to 32Ki bins. See the online help for a detailed description of the controls. Note that invalid events will have energy=0, which causes a large spike in the first bin of the spectrum. Set **Emin** to a nonzero value to hide this spike. If desired, a range of events can be specified to histogram, rather than all events in the file.

3.4.3 RUN STATISTICS

The RUN STATISTICS panel shows the counting times and count rates measured by the Pixie-4 Express. The numbers can be updated by clicking the [**Update]** button and read

from or saved to **Files**. For a detailed description of the definition of these values, see section 6.7.

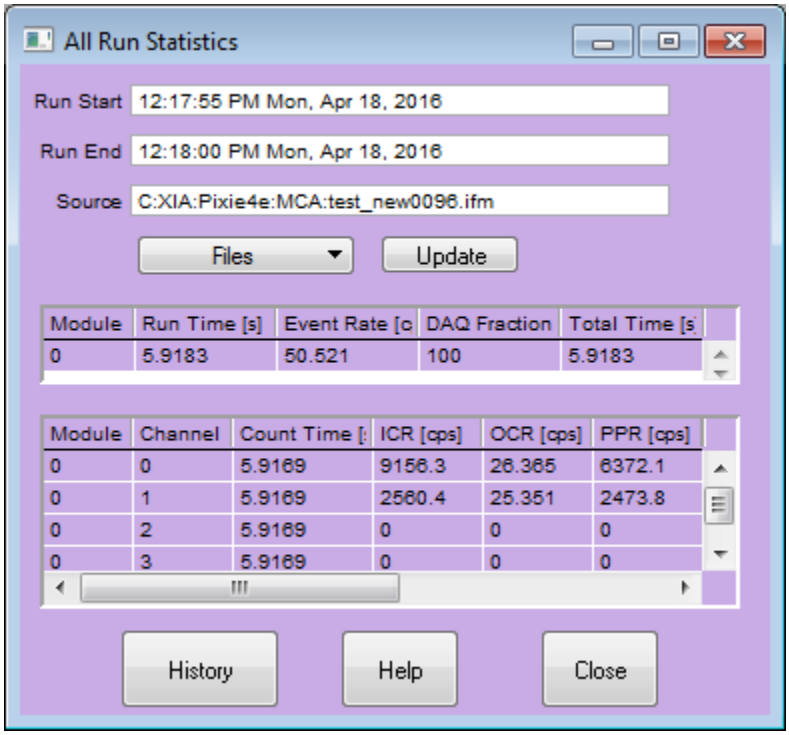


Figure 3-8: The RUN STATISTICS panel.

3.4.4 File Series

See section 3.6 for a more detailed description

3.5 Optimizing Parameters

Optimization of the Pixie-4 Express's run parameters for best resolution depends on the individual systems and usually requires some degree of experimentation. The Pixie Viewer includes several diagnostic tools and settings options to assist the user, as described below.

3.5.1 Noise

For a quick analysis of the electronic noise in the system, you can view a Fourier transform of the incoming signal by selecting OSCILLOSCOPE **»** [**FFT]**. The graph shows the FFT of the untriggered input sigal of the Oscilloscope. By adjusting the **dT** control in the OSCILLOSCOPE and clicking the [**Refresh]** button, you can investigate different frequency ranges. For best results, remove any source from the detector and only regard traces without actual events. If you find sharp lines in the 10 kHz to 1 MHz region you may need to find the cause for this and remove it. If you click on the [**Apply Filter]** button, you can see the effect of the energy filter simulated on the noise spectrum.

3.5.2 Energy Filter Parameters

The main parameter to optimize energy resolution is the energy filter rise time. Generally, longer rise times result in better resolution, but reduce the throughput. Optimization should begin with scanning the rise time through the available range. Try 2µs, 4µs, 8µs, 11.2µs,

take a run of 60s or so for each and note changes in energy resolution. Then fine tune the rise time.

The flat top usually needs only small adjustments. For a typical coaxial Ge-detector we suggest to use a flat top of 1.2µs. For a small detector (20% efficiency) a flat top of 0.8µs is a good choice. For larger detectors flat tops of 1.2µs and 1.6µs will be more appropriate. In general the flat top needs to be wide enough to accommodate the longest typical *signal rise time* from the detector. It then needs to be wider by one filter clock cycle than that minimum, but at least 3 filter clock cycles. Note that a filter clock cycle ranges from 0.026 to 0.853µs, depending on the filter range, so that it is not possible to have a very short flat top together with a very long filter rise time.

The Pixie Viewer provides a tool to create a file series where the energy filter parameters are modified for each file in the series. See section 3.6 for more details.

3.5.3 Threshold and Trigger Filter Parameters

In general, the trigger threshold should be set as low as possible for best resolution. If too low, the input count rate will go up dramatically and "noise peaks" will appear at the low energy end of the spectrum. If the threshold is too high, especially at high count rates, low energy events below the threshold can pass the pile-up inspector and pile up with larger events. This increases the measured energy and thus leads to exponential tails on the (ideally Gaussian) peaks in the spectrum. Ideally, the threshold should be set such that the noise peaks just disappear.

The settings of the trigger filter have only minor effect on the resolution. However, changing the trigger conditions might have some effect on certain undesirable peak shapes. A longer trigger rise time allows the threshold to be lowered more, since the noise is averaged over longer periods. This can help to remove tails on the peaks. A long trigger flat top will help to trigger better on slow rising pulses and thus result in a sharper cut off at the threshold in the spectrum.

3.5.4 Decay Time

The preamplifier decay time τ is used to correct the energy of a pulse sitting on the falling slope of a previous pulse. The calculations assume a simple exponential decay with one decay constant. A precise value of τ is especially important at high count rates where pulses overlap more frequently. If τ is off the optimum, peaks in the spectrum will broaden, and if τ is very wrong, the spectrum will be significantly blurred.

The first and usually sufficiently precise estimate of τ can be obtained from the Auto Find routine in the **Energy** tab of the PARAMETER SETUP panel. Measure the decay time several times and settle on the average value.

Fine tuning of τ can be achieved by exploring small variations around the fit value (± 2-3%). This is best done at high count rates, as the effect on the resolution is more pronounced. The value of τ found through this way is also valid for low count rates. Manually enter τ, take a short run, and note the value of τ that gives the best resolution.

Pixie users can also use the fit routines in the OSCILLOSCOPE to manually find the decay time through exponentially fitting the untriggered input signals. Another tool is to create a file series where τ is modified for each file in the series. See section 3.6 for more details.

3.5.5 Baselines and ADC calibration

Between detector pulses, the Pixie module continuously measures baselines, which is ultimately used to correct for the DC offset. Multiple baseline measurements can be

averaged to reduce noise, and a threshold can be set to exclude the occasional bad measurement from the average. The controls to set these parameters are located in the **Advanced** tab of the PARAMETER SETUP panel. The optimum values depend on the detector used; but usually the defaults are good estimates and resolutions only improve slightly with manual fine tuning.

The 500 MHz ADC used on some variants of the Pixie-4 Express is actually a combination of two 250 MHz ADC cores on a single IC. For best performance, the two cores have to be calibrated to match in gain, offset and phase. Default ADC calibration values are stored on an on-board EEPROM and are applied to the ADCs at boot time. It may happen that the default values are not suitable, e.g. due to significant temperature drifts. This would manifest itself as a distinct offset between even and odd samples in the waveforms. In such a case, the ADCs can be recalibrated with a routine called from a button in the OSCILLOSCOPE or in the **Advanced** tab of the PARAMETER SETUP panel.

3.6 File Series

3.6.1 File Series to break up long data acquisition runs

When taking long data acquisitions, it may be beneficial to break up the run into smaller sub runs. This helps to save data in case of power failure or system crashes, since only the most recent sub run is lost. Also list mode files tend to get large and unwieldy for analysis in longer runs, and 32bit operating systems may impose a 4 GB limit.

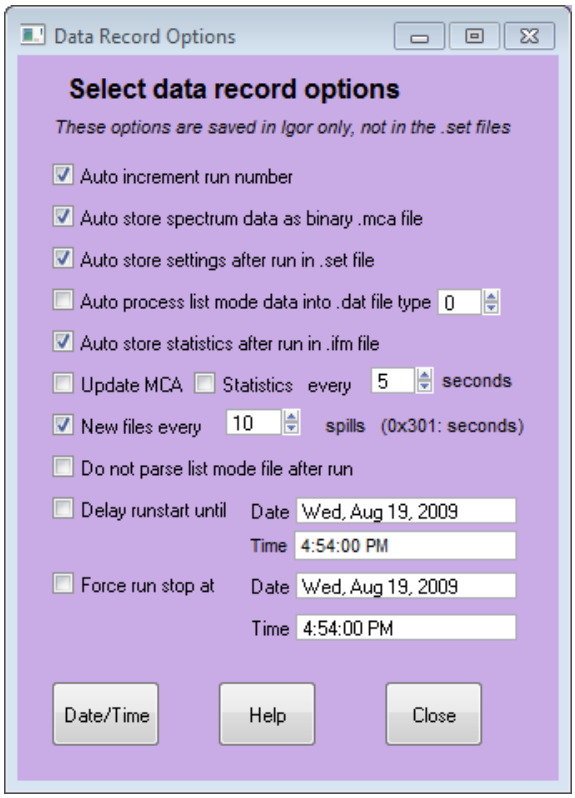


Figure 3-9: The DATA RECORD OPTIONS panel with checkboxes set to acquire a series of files.

The Pixie Viewer thus has a method to create a series of files at specified intervals. In the DATA RECORD OPTIONS panel, opened with the **[Record]** button in the **Run Control** tab

of the PARAMETER SETUP panel, there is a checkbox named **New files every**, followed by a control field to enter an interval N. If checked, every N spills during the run the data file is closed, spectra, settings and statistics are saved, and future data is saved to a new file with incremented run number. It is recommended to enable the automatic increment and auto-store options as shown in Figure 3.9 as well.

Spectra and run statistics are cumulative in these subsequent files for run type 0x400-0x403. In MCA mode, the new file interval is every N seconds. [3]

3.6.2 File Series to scan filter parameters

With some modifications, the mechanism to create file series described in section 3.6.1 can also be used to scan through a range of energy filter or decay time settings. This is equivalent to starting an MCA run with initial settings, stopping the run, incrementing the energy filter rise time, restarting the run, and so on. The file series will thus contain spectra for a whole range of settings, which can be analyzed manually or with the routine described in section 3.6.3.

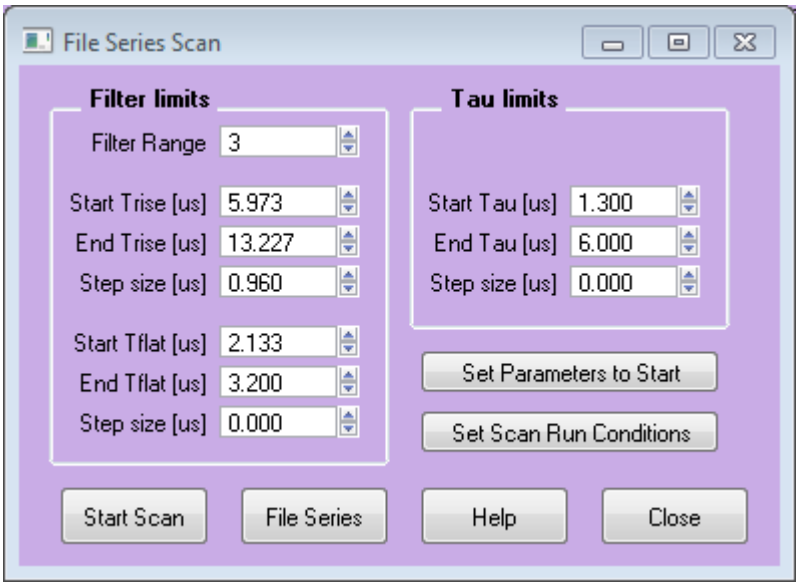


Figure 3-10: The FILE SERIES SCAN panel to acquire a series of files in which energy filter parameters and Tau are varied within user defined limits.

To set up such a parameter scan, open the FILE SERIES SCAN panel (PARAMETER SETUP **» Energy** tab **» [Scan Settings]**) shown in Figure 3.10. A control field named **Filter Range** is repeated from the Energy tab. In three groups of controls, you can set the start, end, and step size for varying the energy filter **rise time**, the energy filter **flat top**, and **Tau**. If the step size is zero, that parameter will not be varied.

Two buttons assist in setting up the initial conditions: **[Set Parameters to Start]** sets the current values of the energy filter and Tau to the start value defined in the FILE SERIES SCAN panel. If you omit to click this button, the file series will begin with the current value; this is useful to resume a file series. **[Set Scan Run Conditions]** will set the checkboxes

[3] In legacy run types 0x100-0x301, a new run is started with the new file (statistics and spectra start from 0, equivalent to manually clicking first the Stop Run button and then the Start Run button.

in the DATA RECORD OPTIONS panel to the values required for the scan, and set the run time to the total time required (interval N in the DATA RECORD OPTIONS panel times the number of settings).

At the bottom of the panel, the button **[Start Scan]** starts the file series. This is a different button from the standard **[Start Run]** button, because it is starting a run which is modifying parameters. All the updates during a run work the same as in a standard run, though; and the run can be stopped with the standard **[Stop Run]** button. When the run is complete, click on the **[File Series]** button to open the panel described in section 3.6.3

3.6.3 File Series Analysis

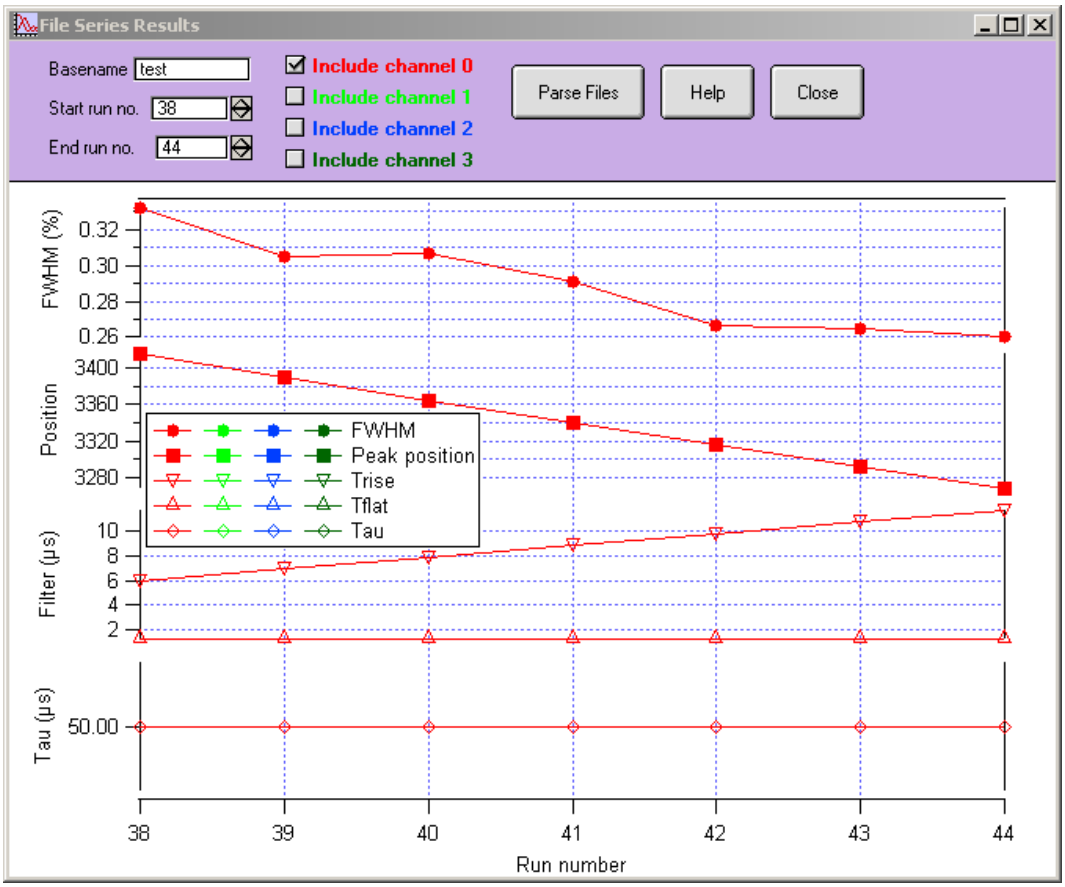


Figure 3-11: The FILE SERIES RESULTS plot to analyze a series of files from a parameter scan.

To analyze a series of .mca files, you can use the FILE SERIES RESULTS panel. Enter the **base name** and the **start** and **end** run numbers of the series, then click **[Parse Files]**. Start and end are inclusive, i.e. for start = 1 and end =13, the parsing covers files base0001-base0013. An .ifm file is required to read the values for Tau and the filter settings. The parsing routine reads the spectra and fits peaks with the options set in the MCA SPECTRUM. Thus make sure the fit range is set appropriately. In the plot, the peak position and resolution is plotted as a function of run number, together with the filter settings and tau for each channel selected.

4 Data Acquisition and Data Structures

4.1 Run Types

There are two major types of data acquisition runs: MCA runs and List mode runs. MCA runs only collect spectra and run statistics, List mode runs acquire data on an event-by event basis, but also collect spectra and run statistics. List mode runs come in several variants (see below), storing different amounts of data per event.

The output data are stored in three different memory blocks. The MCA block resides in a dedicated spectrum memory. List mode data is buffered in 256 MB of SDRAM organized as a FIFO and streamed continuously to the host PC memory. Run statistics are kept in local memory by the on-board FPGA.

4.1.1 MCA Runs

If only energy spectra are of interest, an MCA run should be used. For each event, this type of run only calculates pulse heights (energies). The energy values are then used to increment the MCA spectrum. The run continues until the host computer stops data acquisition, either by reaching the run time set in the Pixie Viewer, or by a manual stop from the user (the module does not stop by itself).

There is no data transferred between the Pixie module and the host PC, except for the occasional manual update of MCA spectra and run statistics. By design, the MCA memory does not "fill up" – each event simply increments a bin in the spectrum.

4.1.2 List Mode Runs

If, on the other hand, data should be collected on an event-by-event basis, including energies, time stamps, pulse shape analysis values, and wave forms, a list mode run should be used. In list mode, pulse heights are still histogrammed into MCA spectra, e.g. for monitoring purposes. The list mode data is continuously transferred from the Pixie module to the host PC.

There are currently three types of list mode runs implemented in the Pixie-4 Express: General purpose (0x400), text only (0x401), and group mode (0x402):

- **General purpose (0x400) list mode runs** collect energies, time stamps, optional PSA results, and wave forms in single channel event records. 32Ki MCA histograms are accumulated in on-board memory. This mode is suitable for systems where interactions between channels are either not relevant or are to be extracted offline from saved data. Even if multiple channels see a pulse at the same time, each channel is recorded independently and individually. Nevertheless, there is the option to share triggers between channels and modules and to apply coincidence tests, and the hit status of all 4 channels is recorded in each channel's record. Offline processing can be used to match time stamps and group events if desired.
- **Text only (0x401) list mode runs** collect energies, time stamps, and optional PSA results. 32Ki MCA histograms are accumulated in on-board memory. Data is transferred internally in the same way as in run type 0x400, but the C library writes results to a text file in ASCII format. Since no waveforms are recorded in the output file, waveforms are not acquired internally even if the tracelengths are set to a

nonzero value. Pulse shape analysis implemented in the FPGA operates on the incoming data stream and creates valid values even with zero tracelengths specified in the settings.

- **Group mode (0x402) list mode runs** collect energies for each channel and an energy sum, time stamps, optional PSA results, and wave forms in 4-channel event records. 16Ki MCA histograms are accumulated in on-board memory for each channel and the sum energy. The assumption is that channels are related and whenever one channel sees a pulse, all channels should be recorded. Consequently, the following options must be enabled in the settings:
    - same trace length and energy filter length for all channels
    - MCA binning factor should be 2 or more to ensure the energies are binned into the 16Ki MCA without being cut off

  The options of gating, vetoing, accepting out-of-range or piled up pulses apply to the whole set of 4 channels. So for example, if one out of 4 channels is piled up, nothing is recorded for any channel unless pileup rejection is turned off for that channel.

  Count rates should be limited to 100kcps or so (with 500ns waveforms). The channels' count time are linked together; whenever one channel is out of range of has full event buffers, all channels stop counting. Group trigger and sum MCA are enabled no matter what the channel settings are (0x402 overrides these settings).

List mode runs halt data acquisition either when a preset time is reached, or when a preset number of "spills" have been collected, as determined by the Pixie Viewer. A spill here means 2 MB of data read from the SDRAM FIFO. Unlike the Pixie-4, the Pixie-4 Express never stops the acquisition for data readout. List mode data is buffered in the SDRAM FIFO, and moved to the host PC on one end while being written by the firmware on the other end. When a 2 MB spill has been moved to PC memory, an interrupt is issued to save the data to file and free up the memory block. At that time, also a data error check can be performed. Given the data bandwidth of the PXIe interface, it is rather unlikely for the SDRAM to fill up, except for situations with very high rates and very long waveforms. (If the SDRAM actually does fill up, data acquisition is paused but as soon as the host frees up SDRAM memory by moving data and storing it to disk, the acquisition continues. The red LED on the module's front panel indicates such a condition. Another indication is a large difference in total time and count time)

When the Pixie Viewer ends the data acquisition, there may be data in the SDRAM that has not yet been saved to file. The readout thus continues for a short while after the DSP stops collecting new data, adding one or more spills to the file. The preset number of spills is therefore to be understood as a minimum request.

4.1.2.1 Pulse Shape Analysis

Pulse shape analysis comes in several varieties, executing algorithms by XIA (enabled by selecting options in the standard firmware/software) or algorithms programmed by users as plug-in code for the DSP. In the current firmware/software, the following algorithms are available from XIA:

- Accumulation of 2 sums (baseline subtracted) near the rising edge of the pulse.
- Capture of amplitude (maximum minus baseline) near the rising edge of the pulse.
- Computation of the ratio of the 2 sums

Please contact XIA for details.

4.1.2.2 Compressed Data Formats

The output data of list mode runs can be reduced by using one of the compressed formats described below. The key differences are that as less data is recorded for each event, there is room for more events in the SDRAM FIFO, less time is spent per event to read out data to the host computer, and data files are smaller. These compressed data formats will be developed as necessary.

4.1.3 Summary of Run Types

Table 4-1summarizes the Run Types described above. The Pixie Viewer also shows Run Types 0x100-103, these are for the Pixie-4 only. Please refer to the Pixie-4 user manual for details.

Run Type	Output data	RUNTASK	Files created
MCA Mode	Spectra in MCA memory	0x301	. mca binary spectra .set binary settings (optional) .ifm ASCII run statistics etc (optional)
List Mode (standard)	Energies, time stamps, PSA values, and wave forms in List mode memory. Spectra in MCA memory	0x400	.b## binary list mode data . mca binary spectra (optional) .set binary settings (optional) .ifm ASCII run statistics etc (optional)
List Mode (text, no traces)	Energies, time stamps, PSA values in List mode memory. Spectra in MCA memory	0x401	_m#.dt3 ASCII list mode data . mca binary spectra (optional) .set binary settings (optional) .ifm ASCII run statistics etc (optional)
List Mode (group)	Energies, time stamps, PSA values in List mode memory. Spectra in MCA memory	0x402	.b## binary list mode data . mca binary spectra (optional) .set binary settings (optional) .ifm ASCII run statistics etc (optional)

Table 4-1: Summary of run types and data files.

4.2 Output Data Structures

4.2.1 MCA Histogram Data Structure

In most run types, the MCA memory uses 32Ki words (32-bit deep) per channel, i.e. total 128Ki words. If spectra of less than 32Ki length are requested, only part of the 32Ki will be filled with data. In run type 0x402, each channel's MCA is 16Ki words, followed by a 16Ki sum energy MCA. The memory can be read out via the PCIe data bus at any time, though not at the full burst rate.

The total MCA memory size on the Pixie-4 Express is 512Ki words. It can be reorganized for special applications (e.g., 2D spectra or channel sum spectra).

If enabled, the histogram data is automatically read and saved to file at the end of the run. The file has the extension .mca and contains 128Ki binary numbers (32 bit unsigned, low byte first).

4.2.2 List Mode Data Structures

The list mode data in the SDRAM FIFO consists of a series of data records for each event. For each module, the host readout process creates an individual file for these records. The extension of these files is .b## (## = 2 digit module number) in run types 0x400 and 0x402 and _m#.dt3 (# = 1-2 digit module number) in Run Type 0x401. The records can be written by the DSP in a number of formats. User code should access the data in the file header to navigate through the data. The file should only be read when the run has ended.

4.2.2.1 File Headers in Run Type 0x400 and 0x402

In Run Types 0x400 and 0x402, the.b## file always starts with a file header of length BUFHEADLEN. Currently, BUFHEADLEN is 32, and the 32 words (16 bit unsigned integer, low byte first) are:

Word #	Variable	Description
0	BlkSize	Block size (16-bit words)
1	ModNum	Module number
2	RunFormat	Format descriptor = Run Type
3	ChanHeadLen	Channel Header Length
4	CoincPat	Coincidence pattern
5	CoincWin	Coincidence window in 8ns clock ticks
6	MaxCombEventLen	Maximum length of traces plus headers from all 4 channels (in blocks)
7	BoardVersion	Module type and revision
8	EventLength0	Length of traces from channel 0 plus header (in blocks)
9	EventLength1	Length of traces from channel 1 (in blocks)
10	EventLength2	Length of traces from channel 2 (in blocks)
11	EventLength3	Length of traces from channel 3 (in blocks)
12	SerialNumber	Serial number of that module
13--31	unused	reserved

Table 4-2: File header data format, total 32 words (16bit).

4.2.2.2 Event Data in Run Type 0x400

Following the file header, in Run Type 0x400 the single channel event records are stored in sequential order. Each event starts out with a channel header of length ChanHeadLen. Currently, ChanHeadLen=32, and the 32 words (16 bit) are:

Word #	Variable	Description
0	EvtPattern	Hit pattern.
1	EvtInfo	Event status flags.
2	NumTraceBlks	Number of blocks of Trace data to follow the header
3	NumTraceBlksPrev	Number of blocks of Trace data in previous record (for parsing back)
4	TrigTimeLO	Trigger time, low word
5	TrigTimeMI	Trigger time, middle word
6	TrigTimeHI	Trigger time, high word
7	TrigTimeX	Trigger time, extra 8 bits
8	Energy	Pulse Height
9	ChanNo	Channel number
10	User PSA Value	Result of User specific pulse shape analysis
11	XIA PSA Value	Result of standard XIA pulse shape analysis
12--15	Extended PSA Values	
16--31	reserved	

Table 4-3: Channel header for Run Type 0x400, total 32 words (16bit).

The hit pattern is a bit mask, which tells which channels were recorded detected within the specified coincidence window plus some additional status information, as listed in table 4.4. The channel header may be followed by waveform data. An offline analysis program can recognize this by reading the number of waveform blocks from the NumTraceBlks word. The block size is defined in the file header.

Bit #	Description
EvtPattern	
0..3	If set, indicates that data for channel 0..3 have been recorded[10]
4..7	4: Logic level of FRONT panel input 5: Result of LOCAL acceptance test 6: Logic level of backplane STATUS line, 7: Logic level of backplane TOKEN line (= result of global coincidence test), see section 7
8..11	If set, indicates that channel 0..3 has been hit in this event[4] (i.e. if zero, energy reported is invalid or only an estimate)
12..15	If set, indicates that the GATE input of channel 0..3 has been high at time of fast trigger

[4] As event records are for a single channel at a time, only one bit in [0..3] is set. If there was a coincident pulse in any other channel, the corresponding hits in [8..11] are set. However, recording of those other channels follows those channels' rules. For example, if a channel is piled up it will only be recorded if pileup rejection is turned off. Event records thus may show coincidence patterns with more channels than actually being recorded.

EvtInfo	
0	Coincidence test result
1	Logic level of backplane VETO line
2	If set, indicates event is piled up
3	If set, indicates waveform FIFO full
4	If set, indicates this channel was hit (else the event was recorded based on distributed trigger)
5	If set, indicates that the GATE input of this channel has been high at time of fast trigger
6	If set, indicates this channel was out of ADC range at time of fast trigger
7..14	reserved
15	If set, indicates a data transmission error has been detected for this event. Parts of header and waveform may be corrupted

Table 4-4: Event pattern and Event info in Run Type 0x400, total 32 bits.

4.2.2.3 Event Data in Run Type 0x401

The .dt3 files created in Run Type 0x401 list energies, channel numbers, 48 bit time stamps, and 6 PSA values as tab delimited values, one event per line. This is the same format as created with the AutoProcessListModeData option set to 3. See below for an example.

Data is transferred from the Pixie module to the PC in the binary format described under Run Type 0x400. The PC parses through the data on the fly and saves the extracted event values to file. The data error routine must be enabled for this Run Type.

```
Module:     0
Run Type:   1025
Run Start Time (s):     33.912160
Event Channel TimeStamp Energy RT   Apeak Bsum  Q0 Q1 PSAval
0   0   16956079848  286   4369   8738   808   1361  2434   0
1   0   16956109724  899   4369   8738   739   628   65466  65535
…
```

Example of .dt3 file content

The headers have the following meanings:

TimeStamp	32 bit time stamp (2ns units)
Energy	Pulse Height
RT*	unused, reserved for rise time
Apeak*	peak amplitude
Bsum*	pre-trigger baseline sum
Q0*	first sum on rising edge of pulse
Q1*	second sum on rising edge of pulse
PSAval*	DSP computed ratio of Q sums

* = only meaningful when PSA firmware variant is active

4.2.2.4 Event Data in Run Type 0x402

In Run Type 0x402, following the file header, the 4-channel event records are stored in sequential order. Each event starts out with an event header of length ChanHeadLen. Currently, ChanHeadLen=32, and the 32 words (16 bit) are:

Word #	Variable	Description
0	EvtPattern	Hit pattern.
1	EvtInfo	Event status flags.
2	NumTraceBlks	Number of blocks of Trace data to follow the header (all channels)
3	NumTraceBlksPrev	Number of blocks of Trace data in previous record (for parsing back)
4	TrigTimeHI	Event trigger time, high word
5	TrigTimeX	Event trigger time, extra 8 bits
6	Energy_sum	Sum of channel energies
7	NumUserDataBlks	Number of blocks of user header data to follow
8	LocalTimeLO_0	Local trigger time, low word (ch. 0)
9	LocalTimeMI_0	Local trigger time, middle word (ch. 0)
10	Energy_0	Pulse Height (ch. 0)
11	NumTraceBlks_0	Number of blocks of Trace data to follow the header (ch. 0)
12	LocalTimeLO _1	Local trigger time, low word (ch. 1)
13	LocalTimeMI _1	Local trigger time, middle word (ch. 1)
14	Energy_1	Pulse Height (ch. 1)
15	NumTraceBlks_1	Number of blocks of Trace data to follow the header (ch. 1)
16	LocalTimeLO _2	Local trigger time, low word (ch. 2)
17	LocalTimeMI _2	Local trigger time, middle word (ch. 2)
18	Energy_2	Pulse Height (ch. 2)
19	NumTraceBlks_2	Number of blocks of Trace data to follow the header (ch. 2)
20	LocalTimeLO _3	Local trigger time, low word (ch. 3)
21	LocalTimeMI _3	Local trigger time, middle word (ch. 3)
22	Energy_3	Pulse Height (ch. 3)
23	NumTraceBlks_3	Number of blocks of Trace data to follow the header (ch. 3)
24	EvtInfo_01	Event status flags channel 0,1
25	EvtInfo_23	Event status flags channel 2,3
26	TrigTimeLO	Event trigger time, low word
27	TrigTimeMI	Event trigger time, middle word
28--31	reserved	reserved

Table 4-5: Channel header for Run Type 0x402, total 32 words (16bit)..

The hit pattern and event status flags are the same as in Table 4-4 above, except that the flags refer to "at least one channel", essentially an OR of the 4-channel flags (table 4.7). The additional event info words *EvtInfo_01*, *EvtInfo_23* record the individual status flags.

Bit #	Description
EvtPattern	
0..3	If set, indicates that data for channel 0..3 have been recorded
4..7	4: Logic level of FRONT panel input 5: Result of LOCAL acceptance test 6: Logic level of backplane STATUS line, 7: Logic level of backplane TOKEN line (= result of global coincidence test), see section 7
8..11	If set, indicates that channel 0..3 has been hit in this event (i.e. if zero, energy reported is invalid or only an estimate)
12..15	If set, indicates that the GATE input of channel 0..3 has been high at time of fast trigger
EvtInfo	
0	Coincidence test result
1	Logic level of backplane VETO line
2	If set, indicates event is piled up for at least one channel
3	If set, indicates waveform FIFO full for at least one channel
4	If set, indicates at least one channel was hit
5	If set, indicates that the GATE input of at least one channel has been high at time of fast trigger
6	If set, indicates at least one channel was out of ADC range at time of fast trigger
7..14	reserved
15	If set, indicates a data transmission error has been detected for this event. Parts of header and waveform may be corrupted
EvtInfo_01, EvtInfo_23	
0	reserved
1	reserved
2	If set, indicates event is piled up (ch.0 or 2)
3	If set, indicates waveform FIFO full (ch.0 or 2)
4	If set, indicates this channel was hit (ch.0 or 2)
5	If set, indicates that the GATE input of this channel has been high at time of trigger (ch.0 or 2)
6	reserved
7	reserved
8	reserved
9	reserved
10	If set, indicates event is piled up (ch.1 or 3)
11	If set, indicates waveform FIFO full (ch.1 or 3)
12	If set, indicates this channel was hit (ch.1 or 3)
13	If set, indicates that the GATE input of this channel has been high at time of trigger (ch.1 or 3)
14	reserved
15	reserved

Table 4-6: Event pattern and Event info in Run Type 0x402, total 32 bits, and channel specific Event info bits in EvtInfo_01 and EvtInfo_23, 32 bits each.

4.2.2.5 File Footer in Run Type 0x400 and 0x402

At the end of the file, an "end of run" (EOR) record is appended. This is created by the DSP so that the software can recognize the end of the data stream to be saved. Its content is shown in Table 4.5. Parsing code should check the EvtInfo word to detect this record, otherwise it will be interpreted as a zero-energy event for channel 1 (usually harmless).

Word #	Variable	Description
0	EvtPattern	EOR pattern (low): 0x0002
1	EvtInfo	EOR pattern (high): 0x0100
2	NumTraceBlks	Number of blocks of Trace data to follow the header: 0
3	NumTraceBlksPrev	Number of blocks of Trace data in previous record (for parsing back)
4--31	reserved	

Table 4-7: EOR record at end of file, total 32 words (16bit).

4.2.2.6 Special List Mode Records

In some variants of the firmware, there can also be special records with additional information. These are listed in the tables below:

Word #	Variable	Description
0	EvtPattern	RSR pattern (low): 0x0004
1	EvtInfo	RSR pattern (high): 0x0100
2	NumTraceBlks	Number of blocks of statistics data to follow the header: **4**
3	NumTraceBlks Prev	Number of blocks of Trace data in previous record (for parsing back)
4--31	reserved	

Word #	Variables (Ch.0)	Variables (Ch.1)	Variables (Ch.2)	Variables (Ch.3)
0..3	COUNTTIMEX0	COUNTTIMEX1	COUNTTIMEX2	COUNTTIMEX3
4_7	...A0	...A1	...A2	...A3
8_11	...B0	...B1	...B2	...B3
12_15	...C0	...C1	...C2	...C3
16_19	FASTPEAKSX0	FASTPEAKSX1	FASTPEAKSX2	FASTPEAKSX3
20_23	...A0	...A1	...A2	...A3
24_27	...B0	...B1	...B2	...B3
28_31	FTDTX0	FTDTX1	FTDTX2	FTDTX3
32_35	...A0	...A1	...A2	...A3
36_39	...B0	...B1	...B2	...B3
40_43	SFDTX0	SFDTX1	SFDTX2	SFDTX3
44_47	...A0	...A1	...A2	...A3
48_51	...B0	...B1	...B2	...B3
52_55	...C0	...C1	...C2	...C3
56_59	GCOUNTX0	GCOUNTX1	GCOUNTX2	GCOUNTX3
60_63	...A0	...A1	...A2	...A3
64_67	...B0	...B1	...B2	...B3
68_71	NOUTX0	NOUTX1	NOUTX2	NOUTX3
72_75	...A0	...A1	...A2	...A3
76_79	...B0	...B1	...B2	...B3
80_83	GDTX0	GDTX1	GDTX2	GDTX3
84_87	...A0	...A1	...A2	...A3
88_91	...B0	...B1	...B2	...B3
92_95	...C0	...C1	...C2	...C3
96_99	reserved	reserved	reserved	reserved
100_103	reserved	reserved	reserved	reserved
104_107	reserved	reserved	reserved	reserved
108_111	reserved	reserved	reserved	reserved
112_115	RecordTimeMI_0	RecordTimeMI_1	RecordTimeMI_2	RecordTimeMI_3
116_119	RecordTimeLO_0	RecordTimeLO_1	RecordTimeLO_2	RecordTimeLO_3
124_127	reserved	reserved	reserved	reserved

Table 4-8: Run Statistics record containing the DSP parameter names for Count Time, Number of output counts etc. Records can be triggered by a pulse on the Veto input. Four blocks of parameter values immediately follow the header block. The individual 16 bit values can be combined to full 48 or 64 bit numbers as described in the Programmer's Manual. The RecordTime is equivalent to the lower 32 bit of an event time stamp.

4.2.3 List Mode Data Values

4.2.3.1 List Mode Time Stamps

In the Pixie-4 Express, there is a 56-bit time counter. It is incremented at a rate of 125 MHz by 4 ticks, so that the unit of the LSB is 2ns. Hence, the 56-bit word can span a time interval of over 800 days before rolling over. The time counter is reset to zero at boot time or at a run start with the "synchronize clocks" option selected.

The counter is latched in the firmware at an event trigger or other notable events. It is reported in the list mode data stream as up to four 16bit words. There are currently three types of time stamps:

- TrigTime in Run Types 0x400 and 0x402 is reported as (TrigTimeLO, TrigTimeMI, TrigTimeHI, TrigTimeX). This is the time at which the event was latched into the front end buffers. It has a fixed delay of a couple of clock cycles from the detected rising edge of the triggering channel.
- LocalTime_# in Run Type 0x402 is reported as (LocalTimeLO_#, LocalTimeMI_#). This is the time of the last detected rising edge in that channel. These local times can be used to compute time-of-arrival differences between channels. If a channel has seen no trigger in this event, the local time stamp (and energy) is unchanged from the previous event.
  Note that waveform capture occurs at TrigTime. Therefore LocalTime should *not* be used to adjust time of arrival extracted from waveforms.
  In most cases, the reported lower 32 bits can be directly combined with the upper 24 bits from the TrigTime. However, there is a small chance that the higher bits rolled over between latching of LocalTime and TrigTime. This will result in an unusually large difference between LocalTime and TrigTime and can be corrected by subtracting 2^{32}.
- RecordTime_# in the special run statistcs record is reported as (RecordTimeLO_#, RecordTimeMI_#). This is the time at which run statistics were latched by the Veto signal.
  In most cases, the reported lower 32 bits can be directly combined with the upper 24 bits from the TrigTime. However, there is a small chance that the higher bits rolled over between latching of LocalTime and TrigTime. This will result in an unusually large difference between LocalTime and TrigTime and can be corrected by subtracting 2^{32}.

4.2.3.2 List Mode Energy

- The energy reported in list mode data is the result of the pulse height measurement. This 16 bit number is *not* simply the difference between baseline and maximum sample of the pulse, but the result of the energy filter corrected for the exponential decay of the signal.
- The energy is also histogrammed in MCA memory. However, since the MCA has only 32Ki bins, the 16bit energy value is divided by 2 (or other user defined power of 2) before incrementing the MCA: bin (E/2) is incremented for measured energy E.
- If a channel is triggered by a group trigger without a local trigger from a rising edge in this channel, the energy captured is still the one from the last local trigger. The DSP then sets such energies to zero unless the "estimate Energy" option is set.

4.2.3.3 List Mode PSA Values

PSA values are described in a separate manual specific to the PSA functions implemented for a specific variant.

5 Hardware Description

The Pixie-4 Express is a 4-channel unit designed for gamma-ray spectroscopy and waveform capturing. It incorporates four functional building blocks, which we describe below. This section concentrates on the functionality aspect. Technical specification can be found in section 1.2. Figure 5.1 shows the functional block diagram of the Pixie-4 Express.

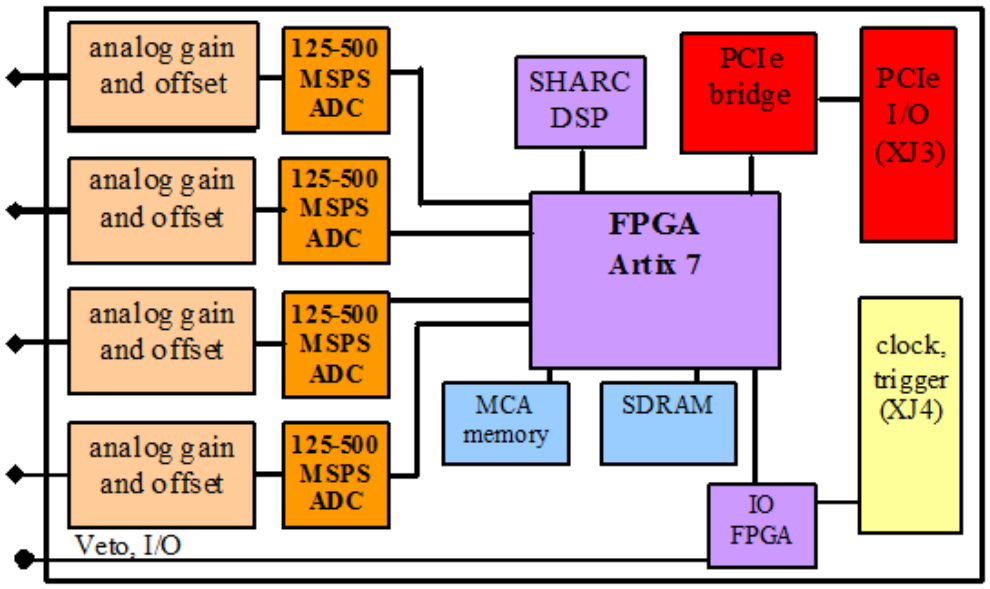


Figure 5-1: Functional block diagram of the Pixie-4 Express front-end data acquisition and signal processing card.

5.1 Analog Signal Conditioning

Each analog input has its own signal conditioning unit. The task of this circuitry is to adapt the incoming signals, which are DC coupled, to the input voltage range of the ADC, which spans 2 V. Input signals are adjusted for offsets, and there is a computer-controlled gain stage with switches and/or relays. Fine tuning of the gain is achieved by multiplying the calculated energy values with digital gain factors in the digital signal processor (DSP). Four options of termination and attenuation are selected by manual switches at the front end of the module. This circuitry is implemented on a small daughtercard; it can be replaced with a variant with different termination, gain, bandwidth etc for specific applications.

The ADC is not a peak sensing ADC, but acts as a waveform digitizer. In order to avoid aliasing, we remove the high frequency components from the incoming signal prior to feeding it into the ADC. The anti-aliasing filter, cuts off sharply at the Nyquist frequency, namely half the ADC sampling frequency.

Though the Pixie-4 Express can work with many different signal forms, best performance is to be expected when sending the output from a charge integrating preamplifier directly to the Pixie-4 Express without any further shaping.

5.2 Pulse Processing

Real time pulse processing is implemented in a field programmable gate array (FPGA) which also incorporates a first level FIFO memory for each channel. The data stream from the ADCs is sent to these units at the full ADC sampling rate. While modern FPGAs can capture high speed data streams, internal processing is limited by the complexity of the logic. Therefore, the FPGA on the Pixie-4 Express processes the data stream at 125 MHz. For 250 MSPS (or 500) MSPS variants each channel's 16 (or 14) bit data stream is "de-serialized" into a 32 (or 56) bit data stream at 125 MHz. Using a pipelined architecture, the signals are processed at this high rate, without the help of the on-board DSP.

The processing applies digital filtering to perform essentially the same action as a shaping amplifier. The important difference is in the type of filter used. In a digital application it is easy to implement finite impulse response filters, and we use a trapezoidal filter. The flat top will typically cover the rise time of the incoming signal and makes the pulse height measurement less sensitive to variations of the signal shape.

The first two processing elements in the FPGA are thus a fast filter for triggering and a slow filter for pulse height (energy) measurements. For a detailed description, see section 6. These filters run continuously. Triggers are issued at each detected rising edge, latch time stamps, and are used for the other processes. The energy filter sums are latched the appropriate time after each trigger.

A third processing element is a pileup inspector. This logic ensures that if a second pulse is detected too soon after the first, so that it would corrupt the first pulse height measurement, both pulses are flagged as piled up. The pileup inspector is, however, not very effective in detecting pulse pileup on the rising edge of the first pulse, i.e. in general pulses must be separated by their rise time to be effectively recognized as different pulses. Therefore, for high count rate applications, the pulse rise times should be as short as possible, to minimize the occurrence of pileup peaks in the resulting spectra.

The fourth processing component is the FIFO memory, which is organized in two blocks. A smaller delay FIFO (2Ki samples) buffers ADC data to position captured waveforms appropriately for the user defined pre-trigger delay. A larger storage FIFO (8Ki samples) captures waveforms of the user defined trace length.

Up to 500 events and 8Ki samples of waveforms are buffered in the FPGA. For each event, a complete set of time stamps, energy filter sums, pileup inspection flags, coincidence information and waveforms are stored. Waveforms from closely following events may overlap, i.e. portions of the same ADC data is stored once but read twice for subsequent events. User defined acceptance settings specify if an event is considered valid (e.g. only accept events without pileup).

The last processing element are a number of counters that maintain the run statistics such as counting time, filter dead time, number of triggers, etc.

5.3 Digital Signal Processor (DSP) and Event Building

The pulse processing described above runs independently in every channel of the Pixie module. On a module-wide level, additional logic is implemented to distribute triggers and apply a coincidence test. See section 7 for details. The result of the coincidence test is fed back to every processing channel.

The DSP manages the flow of channel data into the SDRAM buffer where it is waiting for PCIe readout by the host PC. Whenever a channel has an event in its buffer, the DSP will

read the raw data from the FPGA and based on the event status flags determine if the event is to be recorded. (At this point, there is also the option of executing customized user DSP code to modify results and the acceptance decision.) If the event is acceptable, the DSP computes the pulse height in a few floating point operations, and includes it in the event header data sent to the SDRAM. The captured waveform data is normally not touched by the DSP; the DSP only enables a direct FPGA-internal transfer from the channel processing block to the SDRAM interface block, at a rate of 1GByte/s.

The DSP also controls the overall operation of the Pixie-4 Express. The host computer communicates with the DSP via the PCIe interface. Reading and writing data to DSP memory does only temporarily pause its operation, and can occur even while a measurement is underway. The host sets variables in the DSP memory and if necessary calls DSP functions to apply them to the FPGA. Through this mechanism all gain and offset DACs are set and the filter settings are applied to the FPGA. The FPGA then processes the data without support from the DSP, once it has received the filter settings.

In this scheme, the greatest processing power is located in the FPGA, processing the incoming waveforms from the ADCs in real time and producing, for each valid event, a small set of distilled data from which pulse heights and arrival times can be reconstructed. The computational load for the DSP is much reduced, as it has to react only on an event-by-event basis and has to work with only a small set of numbers for each event.

5.4 PCI Express Interface

The PCI Express (PCIe) interface through which the host communicates with the Pixie-4 Express is implemented in a PCIe endpoint IC which is linked to the FPGA by a local bus. The host computer can read and write a number of registers in the IC, and through it, in the FPGA. The IC can also perform DMA transfers to the host computer's memory, and issue interrupt requests to the host computer.

The FPGA links the PCIe IC with the DSP and the on-board memory. The host can read out the memory without interrupting the operation of the DSP. This allows updates of the MCA spectrum or list mode data while a run is in progress.

A dedicated I/O FPGA distributes triggers and coincidence signals to other modules using the PXIe backplane connections and the front panel connectors.

6 Theory of Operation

6.1 Digital Filters for γ-ray Detectors

Energy dispersive detectors, which include such solid state detectors as Si(Li), HPGe, HgI_2, CdTe and CZT detectors, are generally operated with charge sensitive preamplifiers as shown in Figure 6.1 (a). Here the detector D is biased by voltage source V and connected to the input of preamplifier A which has feedback capacitor C_f and feedback resistor R_f.

The output of the preamplifier following the absorption of an γ-ray of energy E_x in detector D is shown in Figure 6.1 (b) as a step of amplitude V_x (on a longer time scale, the step will decay exponentially back to the baseline, see section 6.3). When the γ-ray is absorbed in the detector material it releases an electric charge $Q_x = E_x/\varepsilon$, where ε is a material constant. Q_x is integrated onto C_f, to produce the voltage $V_x = Q_x/C_f = E_x/(\varepsilon C_f)$. Measuring the energy E_x of the γ-ray therefore requires a measurement of the voltage step V_x in the presence of the amplifier noise σ, as indicated in Figure 6-1 (b). Scintillator detectors read out with a photomultiplier tube generate pulses in a different mechanism, but for the most part they can still be described as fast rise followed by exponential decay, so the processing described below equally applies.

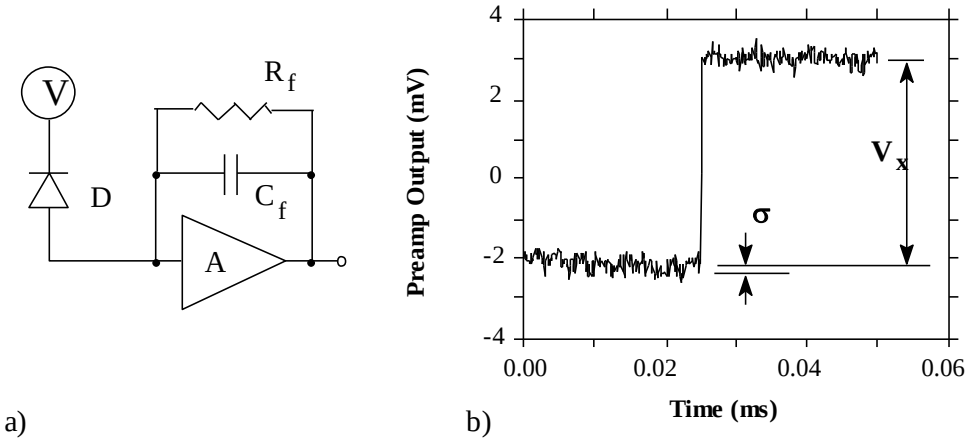


Figure 6-1: (a) Charge sensitive preamplifier with RC feedback; (b) Output on absorption of a □-ray.

Reducing noise in an electrical measurement is accomplished by filtering. Traditional analog filters use combinations of a differentiation stage and multiple integration stages to convert the preamp output steps, such as shown in Figure 6-1 (b), into either triangular or semi-Gaussian pulses whose amplitudes (with respect to their baselines) are then proportional to V_x and thus to the γ-ray's energy.

Digital filtering proceeds from a slightly different perspective. Here the signal has been digitized and is no longer continuous. Instead it is a string of discrete values as shown in Figure 6-2. Figure 6-2 is actually just a subset of Figure 6-1 (b), in which the signal was digitized by a Tektronix 544 TDS digital oscilloscope at 10 MSPS (mega samples per second). Given this data set, and some kind of arithmetic processor, the obvious approach to determining V_x is to take some sort of average over the points before the step and subtract

it from the value of the average over the points after the step. That is, as shown in Figure 6-2, averages are computed over the two regions marked "Length" (the "Gap" region is omitted because the signal is changing rapidly here), and their difference taken as a measure of V_x. Thus the value V_x may be found from the equation:

$$V_{x,k} = -\sum_{i(before)} W_i V_i + \sum_{i(after)} W_i V_i \qquad (1)$$

where the values of the weighting constants W_i determine the type of average being computed. The sums of the values of the two sets of weights must be individually normalized.

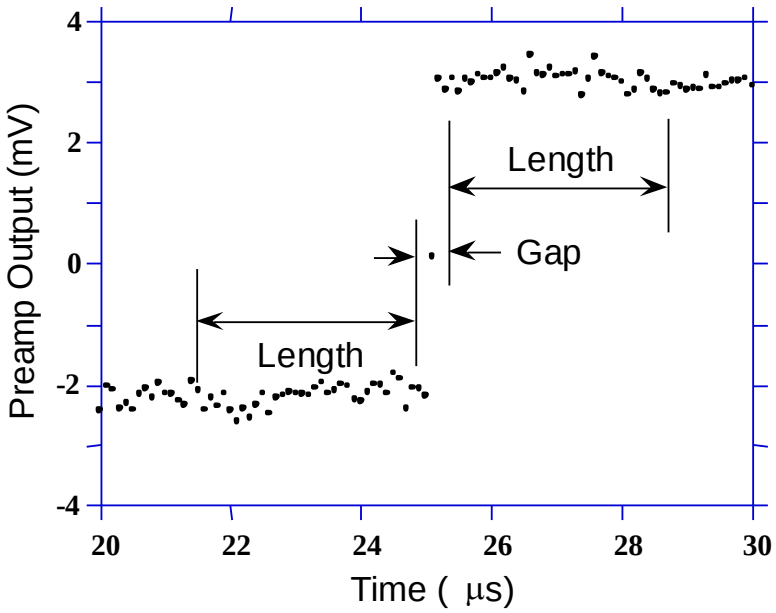


Figure 6-2: Digitized version of the data of Figure 6.1 (b) in the step region.

The primary differences between different digital signal processors lie in two areas: what set of weights W_i is used and how the regions are selected for the computation of Eqn. 1. Thus, for example, when larger weighting values are used for the region close to the step while smaller values are used for the data away from the step, Eqn. 1 produces "cusp-like" filters. When the weighting values are constant, one obtains triangular (if the gap is zero) or trapezoidal filters. The concept behind cusp-like filters is that, since the points nearest the step carry the most information about its height, they should be most strongly weighted in the averaging process. How one chooses the filter lengths results in time variant (the lengths vary from pulse to pulse) or time invariant (the lengths are the same for all pulses) filters. Traditional analog filters are time invariant. The concept behind time variant filters is that, since the γ-rays arrive randomly and the lengths between them vary accordingly, one can make maximum use of the available information by setting the length to the interpulse spacing.

In principle, the very best filtering is accomplished by using cusp-like weights and time variant filter length selection. There are serious costs associated with this approach however, both in terms of computational power required to evaluate the sums in real time and in the complexity of the electronics required to generate (usually from stored coefficients) normalized W_i sets on a pulse by pulse basis.

The Pixie-4 Express takes a different approach because it was optimized for high speed operation. It implements a fixed length filter with all W_i values equal to unity and in fact computes this sum afresh for each new signal value k. Thus the equation implemented is:

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

where the filter length is L and the gap is G. The factor L multiplying $V_{x,k}$ arises because the sum of the weights here is not normalized. Accommodating this factor is trivial.

While this relationship is very simple, it is still very effective. In the first place, this is the digital equivalent of triangular (or trapezoidal if $G \neq 0$) filtering which is the analog industry's standard for high rate processing. In the second place, one can show theoretically that if the noise in the signal is white (i.e. Gaussian distributed) above and below the step, which is typically the case for the short shaping times used for high signal rate processing, then the average in Eqn. 2 actually gives the best estimate of V_x in the least squares sense. This, of course, is why triangular filtering has been preferred at high rates. Triangular filtering with time variant filter lengths can, in principle, achieve both somewhat superior resolution and higher throughputs but comes at the cost of a significantly more complex circuit and a rate dependent resolution, which is unacceptable for many types of precise analysis. In practice, XIA's design has been found to duplicate the energy resolution of the best analog shapers while approximately doubling their throughput, providing experimental confirmation of the validity of the approach.

6.2 Trapezoidal Filtering in a Pixie Module

From this point onward, we will only consider trapezoidal filtering as it is implemented in a Pixie module according to Eqn. 6.2. The result of applying such a filter with Length L=1μs and Gap G=0.4μs to a γ-ray event is shown in Figure 6.3. The filter output is clearly trapezoidal in shape and has a rise time equal to L, a flattop equal to G, and a symmetrical fall time equal to L. The basewidth, which is a first-order measure of the filter's noise reduction properties, is thus 2L+G.

This raises several important points in comparing the noise performance of the Pixie module to analog filtering amplifiers. First, semi-Gaussian filters are usually specified by a *shaping time*. Their rise time is typically twice this and their pulses are not symmetric so that the basewidth is about 5.6 times the shaping time or 2.8 times their rise time. Thus a semi-Gaussian filter typically has a slightly better energy resolution than a triangular filter of the same rise time because it has a longer filtering time. This is typically accommodated in amplifiers offering both triangular and semi-Gaussian filtering by stretching the triangular rise time a bit, so that the *true* triangular rise time is typically 1.2 times the selected semi-Gaussian rise time. This also leads to an apparent advantage for the analog system when its energy resolution is compared to a digital system with the same nominal rise time.

One important characteristic of a digitally shaped trapezoidal pulse is its extremely sharp termination on completion of the basewidth 2L+G. This may be compared to analog filtered pulses whose tails may persist up to 40% of the rise time, a phenomenon due to the finite bandwidth of the analog filter. As we shall see below, this sharp termination gives the digital filter a definite rate advantage in pileup free throughput.

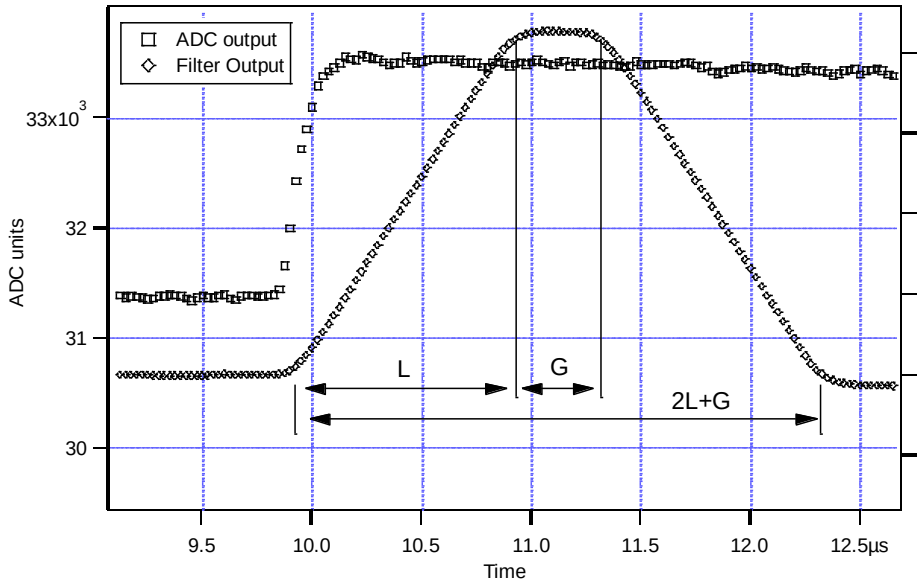


Figure 6-3: Trapezoidal filtering of a preamplifier step with L=1 s and G=0.4 s.

6.3 Baselines and Preamplifier Decay Times

Figure 6.4 shows an event over a longer time interval and how the filter treats the preamplifier noise in regions when no γ-ray pulses are present. As may be seen the effect of the filter is both to reduce the amplitude of the fluctuations and reduce their high frequency content. This region is called the *baseline* because it establishes the reference level from which the γ-ray peak amplitude V_x is to be measured. The fluctuations in the baseline have a standard deviation σ_e which is referred to as the *electronic noise* of the system, a number which depends on the rise time of the filter used. Riding on top of this noise, the γ-ray peaks contribute an additional noise term, the *Fano noise*, which arises from statistical fluctuations in the amount of charge Q_x produced when the γ-ray is absorbed in the detector. This Fano noise σ_f adds in quadrature with the electronic noise, so that the total noise σ_t in measuring V_x is found from

$$\sigma_t = \mathrm{sqrt}(\sigma_f^2 + \sigma_e^2) \qquad (3)$$

The Fano noise is only a property of the detector material. The electronic noise, on the other hand, may have contributions from both the preamplifier and the amplifier. When the preamplifier and amplifier are both well designed and well matched, however, the amplifier's noise contribution should be essentially negligible. Achieving this in the mixed analog-digital environment of a digital pulse processor is a non-trivial task, however.

With a RC-type preamplifier, the slope of the preamplifier is rarely zero. Every step decays exponentially back to the DC level of the preamplifier. During such a decay, the baselines are obviously not zero. This can be seen in Figure 6-4, where the filter output during the exponential decay after the pulse is below the initial level. Note also that the flat top region is sloped downwards.

Using the decay constant τ, the baselines can be mapped back to the DC level. This allows precise determination of γ-ray energies, even if the pulse sits on the falling slope of a previous pulse. The value of τ, being a characteristic of the preamplifier, has to be determined by the user and host software and downloaded to the module.

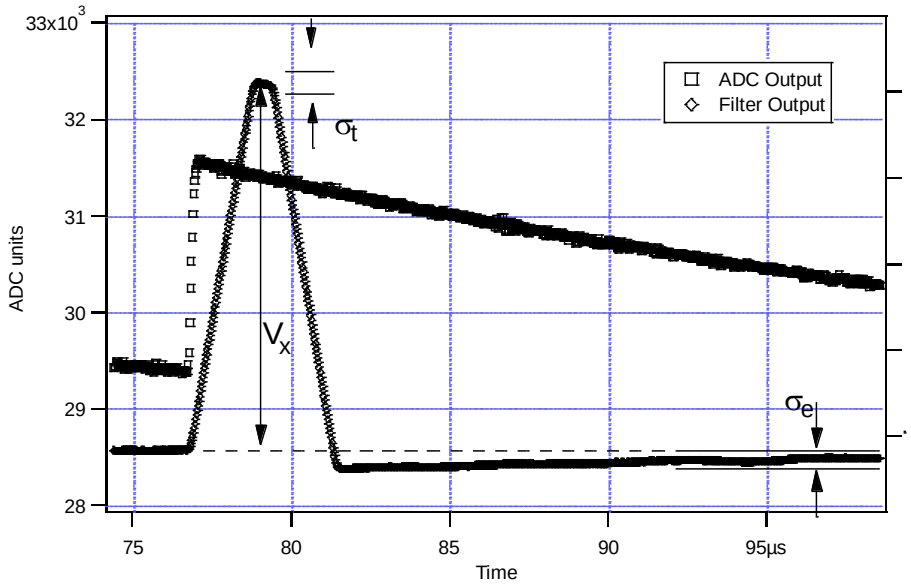


Figure 6-4: A γ-ray event displayed over a longer time period to show baseline noise and the effect of preamplifier decay time.

6.4 Thresholds and Pile-up Inspection

As noted above, we wish to capture a value of V_x for each γ-ray detected and use these values to construct a spectrum. This process is also significantly different between digital and analog systems. In the analog system the peak value must be "captured" into an analog storage device, usually a capacitor, and "held" until it is digitized. Then the digital value is used to update a memory location to build the desired spectrum. During this analog to digital conversion process the system is dead to other events, which can severely reduce system throughput. Even single channel analyzer systems introduce significant deadtime at this stage since they must wait some period (typically a few microseconds) to determine whether or not the window condition is satisfied.

Digital systems are much more efficient in this regard, since the values output by the filter are already digital values. All that is required is to take the filter sums, reconstruct the energy V_x, and add it to the spectrum. In the Pixie-4 Express, the filter sums are continuously updated in the FPGA (see section 5.2), and are captured into event buffers. Reconstructing the energy and incrementing the spectrum is done by the DSP, so that the FPGA is ready to take new data immediately (unless the buffers are full). This is a significant source of the enhanced throughput found in digital systems.

The peak detection and sampling in a Pixie module is handled as indicated in Figure 6.5. Two trapezoidal filters are implemented, a *fast filter* and a *slow filter*. The fast filter is used to detect the arrival of γ-rays, the slow filter is used for the measurement of V_x, with reduced noise at longer filter rise times. The fast filter has a filter length L_f = 0.1µs and a gap G_f =0.1µs. The slow filter has L_s = 1.2µs and G_s = 0.35µs.

The arrival of the γ-ray step (in the preamplifier output) is detected by digitally comparing the fast filter output to THRESHOLD, a digital constant set by the user. Crossing the threshold starts a delay line to wait PEAKSAMP clock cycles to arrive at the appropriate time to sample the value of the slow filter. Because the digital filtering processes are deterministic, PEAKSAMP depends only on the values of the fast and slow filter constants.

The slow filter value captured following PEAKSAMP is then the slow digital filter's estimate of V_x. Using a delay line allows to stage sampling of multiple pulses even within a PEAKSAMP interval (though the filter values themselves are then not correct representations of a single pulse's height).

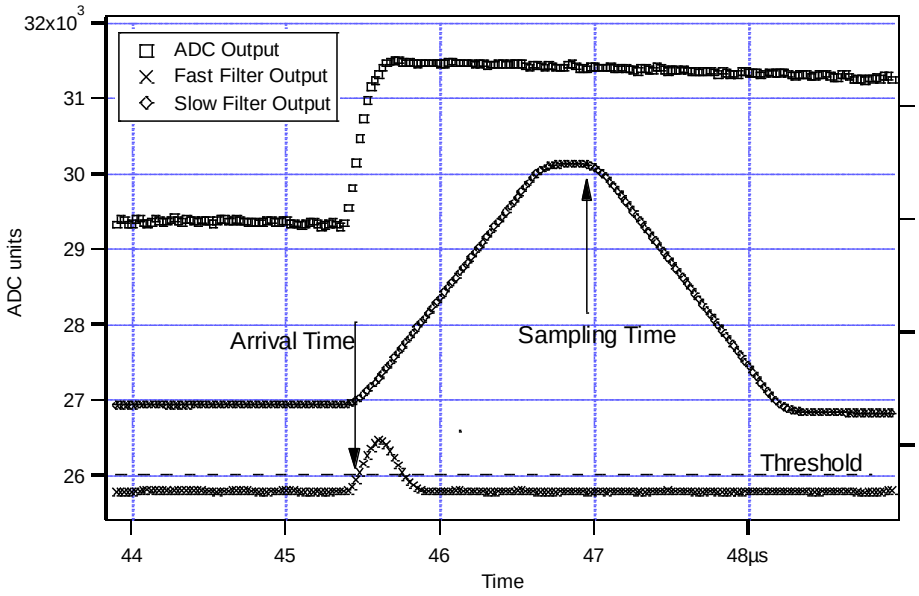


Figure 6-5: Peak detection and sampling in a Pixie module.

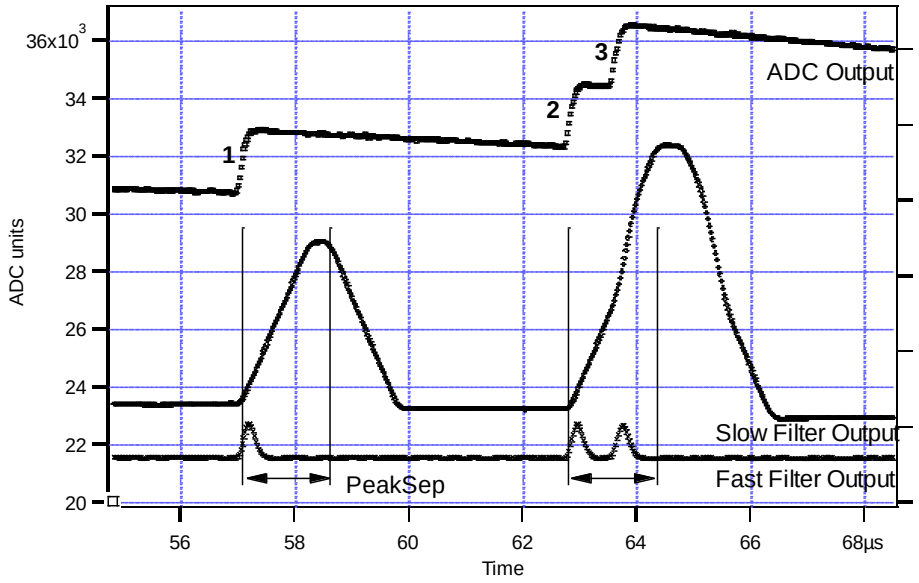


Figure 6-6: A sequence of 3 γ-ray pulses separated by various intervals to show the origin of pileup and demonstrate how it is detected by the Pixie module.

The value V_x captured will only be a valid measure of the associated γ-ray's energy provided that the filtered pulse is sufficiently well separated in time from its preceding and succeeding neighbor pulses so that their peak amplitudes are not distorted by the action of the trapezoidal filter. That is, if the pulse is not *piled up*. The relevant issues may be understood by reference to Figure 6-6, which shows 3 γ-rays arriving separated by various

intervals. The fast filter has a filter length L_f = 0.1μs and a gap G_f =0.1μs. The slow filter has L_s = 1.2μs and G_s = 0.35μs.

Because the trapezoidal filter is a linear filter, its output for a series of pulses is the linear sum of its outputs for the individual members in the series. Pileup occurs when the rising edge of one pulse lies under the peak (specifically the sampling point) of its neighbor. Thus, in Figure 6.6, peaks 1 and 2 are sufficiently well separated so that the leading edge of peak 2 falls after the peak of pulse 1. Because the trapezoidal filter function is symmetrical, this also means that pulse 1's trailing edge also does not fall under the peak of pulse 2. For this to be true, the two pulses must be separated by at least an interval of L + G. Peaks 2 and 3, which are separated by less than 1.0 μs, are thus seen to pileup in the present example with a 1.2 μs rise time.

This leads to an important point: whether pulses suffer slow pileup depends critically on the rise time of the filter being used. The amount of pileup which occurs at a given average signal rate will increase with longer rise times.

Because the fast filter rise time is only 0.1 μs, these γ-ray pulses do not pileup in the fast filter channel. The Pixie module can therefore test for slow channel pileup by measuring the fast filter for the interval PEAKSEP after a pulse arrival time. If no second pulse occurs in this interval, then there is no trailing edge pileup and the pulse is validated for acquisition. PEAKSEP is usually set to a value close to L + G + 1. Pulse 1 passes this test, as shown in Figure 6.6. Pulse 2, however, fails the PEAKSEP test because pulse 3 follows less than 1.0 μs. Notice, by the symmetry of the trapezoidal filter, if pulse 2 is rejected because of pulse 3, then pulse 3 is similarly rejected because of pulse 2.

6.5 Filter Range

To accommodate a wide range of energy filter rise times from tens of nanoseconds to tens of microseconds, the filters are implemented in the FPGA with different clock decimations (filter ranges). The ADC sampling rate is always 8ns (2ns or 4ns in 500 MSPS or 250 MSPS variants), but in higher clock decimations, several ADC samples are averaged before entering the energy filtering logic. In filter range 1, 2^1 samples are averaged, 2^2 samples in filter range 2, and so on. Since the sum of rise time and flat top is limited to 127 decimated clock cycles, filter time granularity and filter time are limited to the values listed in Table 6.1.

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

Table 6-1: Filter clock decimations and filter time granularity

6.6 Data Capture Process

The data capture in the Pixie-4 Express is based on the principle that for every detected rising edge, one record is assembled from the continuously running processes for waveform capture and energy filters. As some of the processes are not finished by the time of the rising edge, input data or capture signals are delayed appropriately. For example, incoming ADC data is delayed for the waveform capture by the user specified pre-trigger delay. The signal to capture energy filter sums is sent through a delay line of length (energy filter rise time plus energy filter flat top) to capture the output after filtering.

Consequently, for every rising edge, the following information is latched into front end buffers:

- 56 bit time stamp of latch signal
- 32 bit time stamp of last rising edge in this channel
- Energy filter sums for last rising edge in this channel
- Pileup inspection flags
- Coincidence flags
- Starting address of waveform memory

and the (delayed) waveform data begins to flow into the waveform memory, for the user specified length of trace. The front end buffers hold 500 such records and the waveform memory holds 8Ki samples.

When the front end buffers are not empty, a flag is raised for the DSP. On this flag, the DSP reads one record and checks if it is to be recorded per the user defined pileup and coincidence conditions. If so, the DSP computes final energies, increments the MCA histogram, and writes the channel header to the FPGA to send to the SDRAM list mode data stream. Following the header, the waveform data is moved from waveform memory to the SDRAM, starting at the recorded starting address. If the event is piled up or otherwise rejected, it is cleared from the front end buffer without recording.

Closely following rising edges still capture one record per edge, with the limitation of one record per 1/8 of a decimated clock cycle in filter range 3 and higher. If such events are piled up, the energy will be not a valid measure of the pulse height and waveforms may overlap from pulse to pulse, but some of the information in the record may still be useful for offline re-analysis.

6.7 Dead Time and Run Statistics

6.7.1 Definitions

Dead time in the Pixie-4 Express data acquisition can occur at several processing stages. For the purpose of this document, we distinguish three types of dead time (described below), each with a number of contributions from different processes.

Please note: There is a conceptual difference between momentary dead time (associated with a pulse) and cumulative dead time (sum of dead time contributions during an acquisition). Their relation is not trivial.

Live time is often used to describe the portion of the overall time during which the system was not dead. However, since dead time can occur on several levels, this term is prone to misunderstandings and not used here.

6.7.1.1 Dead time associated with each pulse

1. Filter dead time

At the most fundamental level, the energy filter implemented in the FPGA requires a certain amount of pulse waveform (the "filter time") to measure the energy. Once a rising edge of a pulse is detected at time T0, the FPGA computes three filter sums using the waveform data from T- (a energy filter rise time before T0) to T1 (a flat top time plus filter rise time after T0), see section 6.4 and figure 6.7. If a second pulse occurs during this time, the energy measurement will be incorrect. Therefore, processing in the FPGA includes pileup rejection which enforces a minimum distance between pulses and validates a pulse for recording only if no more than one pulse occurred from T0 to T1. Consequently, each pulse creates a dead time Td = (T1 – T0) equal to the filter time. This dead time, simply given by the time to measure the pulse height, is unavoidable unless pulse height measurements are allowed to overlap (which would produce false results).

Assuming randomly occurring pulses, the effect of dead time on the output count rate is governed by Poisson statistics for paralyzable systems with pileup rejection[5]. This means the output count rate OCR (valid pulses) is a function of filter dead time Td and input count rate ICR given by

$$OCR = ICR * \exp(-ICR* 2 * Td), \quad (4)$$

which reaches a maximum $OCR_{max} = ICR_{max}/e$ at $ICR_{max} = 1/(2*Td)$. Simply speaking, the factor 2 for Td comes from the fact that not only is an event E2 invalid when it falls into the dead time of a previous event E1, but E1 is rejected as piled up as well. This filter dead time is accumulated in the SFDT counter in each processing channel.

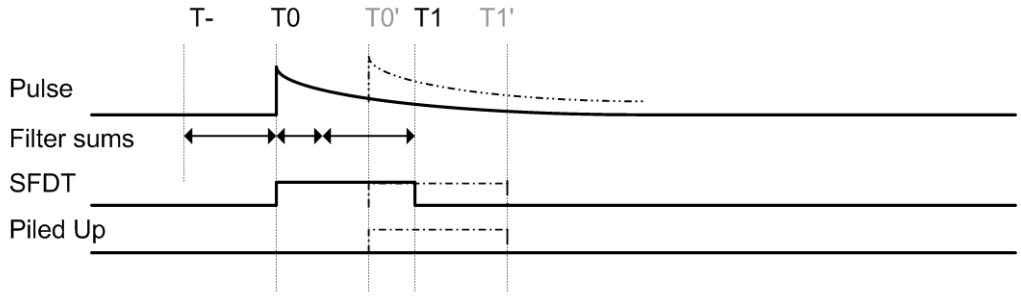


Figure 6-7: Filter dead time. A pulse arriving at T0 will incur slow filter dead time (for energy measurement) until T1. At T1, the pileup status is latched – for a single pulse, it is logic low and the event is accepted. A second pulse arriving at T0' will extend the dead time and cause the pileup status to be logic high. Unless pileup rejection is disabled, both events are rejected.

2. Fast trigger dead time (FTDT)

A second type of dead time only affects the trigger filter. Triggers are issued when the trigger filter output goes above the trigger threshold set by the user. However, the trigger filter output will remain above threshold for a finite amount of time, depending on the length of the trigger filter and the rise time of the input signal. During this time, no second trigger can be issued[6]. Therefore triggers are not counted during this time, and when

[5] G. Knoll, Radiation and Measurement, J Wiley & Sons, Inc, 2000, chapters 4 and 17.

[6] The MAXWIDTH parameter can be used to define a maximum acceptable time over threshold and thus to reject events piled up "on the rising edge".

computing the input count rate, the time lost has to be taken into account. FTDT is thus purely a correction for the computation of the input count rate.

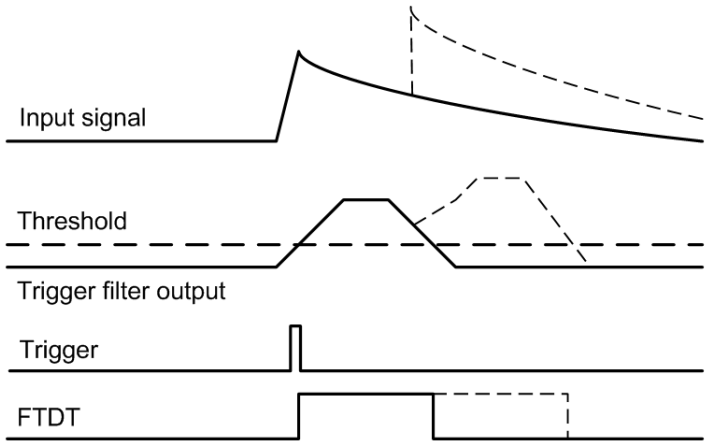


Figure 6-8: Fast Trigger Dead Time (FTDT). A second pulse is not detected if the trigger filter output is still above threshold.

3. Other

In the Pixie-4 and Pixie-500, there was additional dead times associated with reading out the data, since only one event at a time was stored in the FPGA. In the Pixie-4 Express, up to 500 events (and/or total 8Ki waveform samples) are buffered in the FPGA. Thus new events are accepted while captured ones are read out and processed further, and these types of dead time are eliminated. If the buffers fill up, the channel pauses acquisition and stops the count time counter.

6.7.1.2 Dead time associated with external conditions

There are three dead time effects that originate from outside the trigger/filter FPGA. The first two have the effect of stopping the Pixie-4 Express count time counter, the last is counted separately.

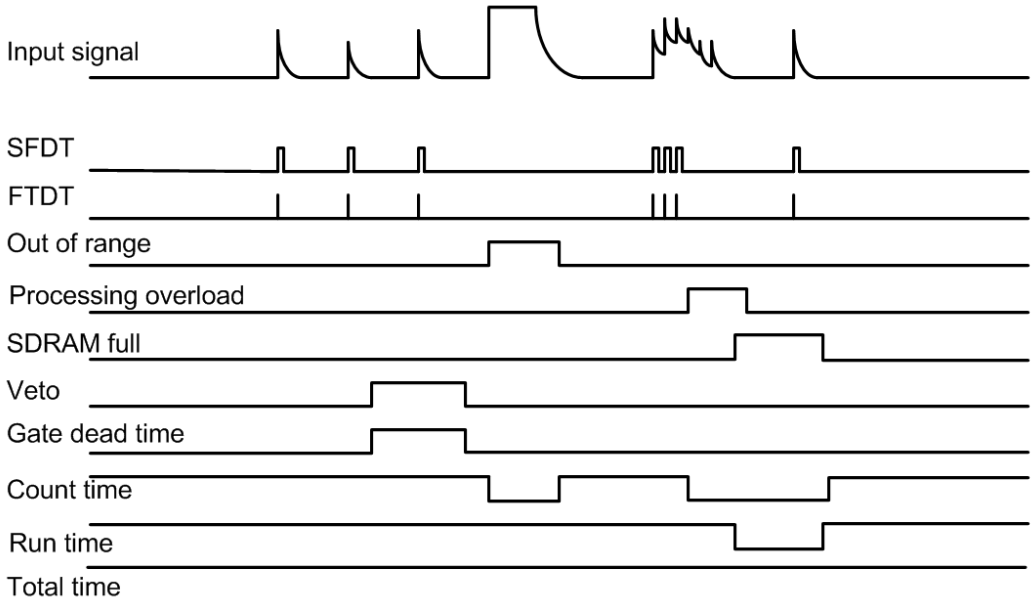


Figure 6-9: The count time counter is stopped when the signal is out of range and when events are rejected because of a processing backlog (e.g. local buffer memory full or SDRAM not read out by host). SFDT and FTDT are only counted when the count time is on. The gate dead time is counted in a separate counter, but also only when the count time is on. Run time and total time are always on unless the run is stopped (see below).

1. Signal out of range

When detector gains or offsets drift, or an unusual large pulse is generated in the detector, the analog input of the ADC may go out of range. In this condition, the FPGA can not accumulate meaningful filter sums and thus is considered dead. This condition persists during the actual out-of-range time and several filter times afterwards until the bad ADC samples are purged from filter memory. The count time counter is stopped during the out-of-range condition because no triggers can be issued and no pulses are counted.

2. On-board pulse processing limit

The on-board pulse processing by the DSP computes the pulse height (energy) from raw energy filter sums, which is then stored in list mode memory and/or binned into spectrum memory. In the Pixie-4 Express, the computation itself takes only a few DSP cycles, but the readout and other overhead amounts to approximately 1 microsecond per pulse to read raw sums, compute the energy, and send it to the memory SDRAM. Waveforms are transferred within the FPGA at ~1 GB/s. Every microsecond of captured waveform thus takes ¼ of a microsecond for SDRAM data transfer (½ of a microsecond for 250 MSPS ADCs, 1 microsecond for 500 MSPS ADCs). The measured processing limit is roughly 800,000 pulses/s if a single channel is processed per event in MCA mode. This rate is much higher than the maximum throughput set by Poisson statistics for most typical filter times. In List mode with nonzero waveforms, the limit is strongly dependent on the length of the

captured waveform. However, bursts of pulses may still exceed the processing rate momentarily, fill the buffers, and so prevent the channel from acquiring more data. Thus the count time counter is also stopped during such buffer full (processing overload) conditions.

3. *Gate* or *Veto*

If an external signal prohibits acquisition using the *Gate* or *Veto* signals, the channel is also dead (disabled on purpose). As further described in section 7.4, the use of these signals may depend on the application:

- On one hand they may be used to reject an individual pulse (e.g. externally summing multiplicities from several channels and issuing a short validation pulse at the right time in the validation process). In this case the actual length of the pulse does not correspond to a dead time. The *Veto* input is set up for this purpose and we call this mode of operation GFLT (global first level trigger for validation). The GCOUNT parameter counts the number of *Veto* pulses and is likely more useful to describe the experiment.
- On the other hand *Gate* or *Veto* may block validation of events for certain amounts of time (e.g. changing sources or activating beams). In this case they should be counted clock cycle by clock cycle as dead time. Both the *Veto* and the *Gate* inputs are available for this purpose, *Veto* as a global signal for the whole system, Gate as a dedicated signal for each channel. *Veto* acts at the time of pulse validation, *Gate* acts at the time of the rising edge of the pulse. However, the *Veto* input can be routed to replace the *Gate* input as a user option.
- In a third class of application, the acquisition may only be of interest when *Gate* or *Veto* are on (or off). The pulse rejection logic would be similar to the second case, but count times and count rates should only be counted when *Gate*/*Veto* are on (or off) as count rates would be different in on and off times. (In the second case one would be more interested in an overall count time and average count rate and additionally the time inhibited by *Gate* or *Veto* to make corrections.)

The appropriate way to count *Gate* or *Veto* dead time may thus depend on the experiment. See below (GDT) for the current methods implemented in the firmware.

6.7.1.3 Dead time associated with host readout

The final type of dead time comes from the readout of data from Pixie-4 Express memory to the host PC. In MCA mode, this is limited to the access arbitration for the spectrum memory. The memory has only a single port for both the increments according to the pulse height computed by the DSP and for readout to the host PC, arbitrated by the FPGA. While the host is reading the memory, spectrum increments are queued in a buffer (2iK long). At the maximum count rate, it will take the DSP at least (2Ki * processing time) to fill the buffer and correspondingly longer at lower count rates while the host readout typically takes ~30 ms. Thus host readout dead time is usually not an issue in MCA runs unless rates are high and spectra are read very frequently.

In list mode runs, the data is buffered in a large SDRAM memory organized as a FIFO. In a major improvement compared to the Pixie-4, the Pixie-4 Express therefore never stops the acquisition for data readout. The host PC can read the memory from one end at the rate set by the PCIe interface (max. 800 MB/s) while new data is added on the other end. Given the data bandwidth of the PXIe interface, it is rather unlikely for the SDRAM to fill up, except for very high rates at very long waveforms. (If the SDRAM actually does fill up, data acquisition is *paused* but as soon as the host frees up SDRAM memory by reading and

storing data to disk, the acquisition continues. Any such pause is counted as dead time by turning off the count time counter, see Figure 6-9.)

At the current firmware and SDRAM operation rate, the SDRAM reads and writes data at about 0.5 GB/s, for a combined average throughput of ~0.25 GB/s, and the PXI Express to the host PC readout matches that rate.[7] With the buffering of up to 500 events in each channel's first processing stage, and of up to 256 MB in the SDRAM, temporary bursts of pulses creating higher data rates can still be captured without loss of data. There is also an option to bypass the SDRAM with a smaller, in-FPGA FIFO which allows data transfers to the host PC at roughly 0.5 GB/s (in current firmware). We note that streaming high rate ADC data in real time, for example to 4 x 2 bytes x 500 MSPS = 4 GB/s, is still beyond the capacity of the PCIe x4 interface used by the Pixie-4 Express. Few, if any, PXIe crates, controllers, and/or hard drives can accommodate such rates. Streaming ADC data is therefore only possible if some kind of data compression is applied, please contact XIA for details.

6.7.2 Count time and dead time counters

The Pixie-4 Express firmware has been optimized to reduce the dead time as much as possible, and a number of counters measure the remaining dead times as well as the number of counts to provide information for dead time correction. The result of these counters is stored in the following DSP output variables:

TOTAL TIME

The TOTAL TIME is an attempt to measure the real laboratory time during which the Pixie module was requested to take data. It essentially counts the time from the command to start a data acquisition to the command to end it. The TOTAL TIME includes the time spent for run start initialization and host readout. However, since it is based on the PXIe chassis' internal clock (a part with typically 50 ppm accuracy) and only updated periodically (~1 ms), it may not be as precise as a "laboratory wall clock" over long time spans (e.g. the host PC's internal clock). Also, it does not take into account the time required to send commands from the PC to the module.

RUN TIME

The RUN TIME variable tracks the time during which the DSP on the Pixie module was "switched on" for data acquisition. The usefulness of this variable is limited. It may be less than the TOTAL TIME because it omits the time the SDRAM is full and waiting for readout (during which the data acquisition is paused in all channels). It is larger than the time an individual channel is ready to take data because it does not account for the dead time from the pileup inspection, out-of-range condition and energy filter in each individual channel. Thus its main uses are to compute an average event rate (total output counts of all channels / RUN TIME), and to compute the fraction of time the SDRAM was not full and acquisition not paused, which is the "DAQ Fraction" displayed in the Pixie Viewer: RUN TIME / TOTAL TIME.

COUNT TIME

The COUNT TIME is counted in the FPGA independently for each channel and measures the time the channel is ready for acquisition. The COUNT TIME counter starts when the

[7]However, writing data to hard disk is usually much slower, in the order of 0.1 GB/s depending on the particular system. Advances system architectures may improve this data rate, for example multiple hard drive arrays.

DSP finished all setup routines at the beginning of a run, omits the times the ADC signal is out of range, each channel's local 500-event buffer is full, or the SDRAM memory is full and ends when the DSP encounters an end run condition (e.g. host stop). Internally, the "counter on" signal is called LCE. It is thus the time during which triggers are counted and can cause recording (or pile up) of data, the best available measurement of the time the channel was active. The difference between COUNT TIME and RUN TIME can be used to determine how long the local 500-event buffers were full and waiting for readout or other events prevented the channel from data taking (e.g. out of range).

FTDT (fast trigger dead time)

The fast trigger dead time counts the time the trigger filter is unable to issue triggers because the trigger filter output is already above threshold (and can not recognize a second pulse). It does not include the time triggers have been "paused" for a short time after a first trigger (an advanced user option to suppress double triggering), because the concept is that all triggers occurring during the pause are counted as only one trigger. When computing the input count rate, one should divide the number of triggers counted (FASTPEAKS) by the difference (COUNT TIME – FTDT) since triggers are not counted during FTDT.

SFDT (slow filter dead time)

The slow filter dead time counts the time new triggers will not lead to the recording of new data. This is the time the pileup inspection is taking place and the summation of energy filter sums is in progress (section 6.7.1.1). In case pileup inspection is inverted or disabled, there is no contribution to SFDT.

GDT (*GATE* dead time)

The dead time from *Veto* /GFLT is counted separately from SFDT for each channel. As mentioned above and further described in section 7.4, the use of these signals may depend on the application.

In the current firmware, the time during which GDT is counted depends on several user options on signal source and polarity. The source options result in a signal GCE to be counted, the polarity selects whether to count while GCE is high or low, as listed in the following table.

Use Veto	Gate Mode	GCE	Count @ Fall	GDT incremented
0	0	(*Veto* OR *Gate**) AND LCE	0	GCE high
1	0	*Gate** AND LCE	0	GCE high
0	1	(*Veto* OR *Gate**)	0	GCE high
1	1	*Gate**	0	GCE high
0	0	(*Veto* OR *Gate**) AND LCE	1	GCE low
1	0	*Gate** AND LCE	1	GCE low
0	1	(*Veto* OR *Gate**)	1	GCE low
1	1	*Gate**	1	GCE low

* shaped and delayed as described in section 7.4

With the assumption that normally only either *Gate* or *Veto* are used, GCE is either the OR of *Veto* and *Gate* (the latter shaped and delayed as described in section 7.4), or only *Gate* if the *Veto* input is used as the source for *Gate* (to avoid double counting). Unless in Gate Mode, GCE is only on while the count time is incremented (LCE=1), which means it is only counted while data is being acquired. In Gate Mode, GCE is independent of LCE and in turn LCE is only on while GCE=1, so prohibit all acquisition and counters while the acquisition is gated. GCE is counted as GDT either while it is high or while it is low. Any rejection due to *Veto* or *Gate* has to be configured independently.

For the case that the Veto input is used for a GFLT-type validation pulse, it may be more useful to work with the number of pulses issued. They can be counted by using the *Veto* input as the source for GATE PULSEs, which are counted in the variable GCOUNT.

6.7.3 Count Rates

Besides the count time and dead times, the Pixie-4 Express counts the numbers of triggers in each channel, FASTPEAKS, the number of valid single channel events, NUMEVENTS, and the number of valid pulses stored for each channel, NOUT. To accommodate dead time correction for pileup even in cases where events are not recorded for other reasons (e.g. not matching coincidence or veto requirements), a counter NPPI counts the number of locally triggered events passing pileup inspection. In addition, it counts the number of gate pulses for each channel, GCOUNT. FASTPEAKS and GCOUNT are inhibited when the COUNT TIME counter is not incrementing. NUMEVENTS and NOUT by nature only count events captured when the COUNT TIME counter is incrementing.

Count rates are then computed in the C library as follows:

Input count rate	ICR	=	FASTPEAKS / (COUNT TIME – FTDT)
Event rate	ER	=	NUMEVENTS / RUN TIME
Channel output count rate	OCR	=	NOUT / COUNT TIME
Channel Pass Pileup Rate	PPR	=	NPPI / COUNT TIME
Gate count rate	GCR	=	GCOUNT / COUNT TIME

Users are free to use the reported values to compute rates and time better matching their preferred definitions.

Notes:

- Output pulse counters are updated whenever an event has been processed; input, gate and all time counters are updated every ~7ms. Therefore reading rates at random times, e.g. clicking *Update* in the Pixie Viewer, might return slight inconsistencies between input rates and output rates. At the end of the run, all rates are updated and these effects should disappear.
- NOUT is counted for each event a channel is processed no matter if the channel had a valid hit or not. Thus a channel that is processed in "group trigger" mode may have an output count rate even though its input count rate is zero.
- Since COUNT TIME counters are paused when SDRAM or local 500-event buffers are full, the input and output count rates should be considered as "rates while active" as opposed to actual rates per elapsed lab time. For input count rates, this is the more intuitive case, since the detector will not stop generating pulses when the channel becomes inactive due to a full SDRAM and the input count rate should closely correspond to the detector's rate. For output count rates, it is a matter of perspective – should it mean the total number of counts per acquisition lab time or the number of counts processed while the Pixie module is taking date? The former would produce unreasonably low count rates when e.g. the signal goes out of range periodically, since it will not account for the duty cycle of the signal source. The latter would produce unreasonably high rates if the system is near its processing limit and often paused for SDRAM readout, though it will better reflect the pileup rejection statistics. The choices made in the current firmware select the latter case, but by

qualifying the output count rate with the COUNT TIME / TOTAL TIME, the former can be recovered.

6.7.4 Dead time correction in the Pixie-4 Express

Historically, dead time correction in analog systems relied on the system dead time measurements taken directly from the acquisition system and the recorded output count rate. For example, a peak sensing ADC module might output a "dead time" signal during the several microseconds it would require to capture the peak value. Thus reconstruction of true count rate required the knowledge of dead times associated with various stages of acquisition and the subsequent mathematical modeling to tie this quantity to the input rate. The classic paralyzable and non-paralyzable models of pulse acquisition do exactly that. For example, the dead time from a non-paralyzable ADC conversion process simply "takes away" active counting time (Td for each output count) and so one can use the classical model of OCR = ICR/(1+Td*ICR), derive[8] OCR/ICR = (real time – dead time)/(real time), and solve for ICR as a function of measured OCR, real time and dead time.

In the Pixie-4 Express, the input count rate is measured directly with the trigger filter, and so the system dead time bears only theoretical or diagnostic value. For any measurements where accurate determination of true (source) counts are required (activity measurements), the empirical ratio ICR/OCR is the only really unbiased quantity for dead time correction. No matter what the actual dead time the acquisition process incurs on the system, the ICR/OCR ratio applied to any region of interest in the energy spectrum correctly reconstructs the true counts in this region, assuming a random source so that pulses are lost with equal probability in each region or in time. **For cases where events are added by group trigger or removed by coincidence or veto requirements, PPR should be used instead of OCR.**

In the Pixie-4 Express firmware design, the counter SFDT attempts to independently account for the system dead time. SFDT counts the time during which the pileup rejection will reject this and any subsequent pulse that are too close in time. Essentially it measures – cumulative for all pulses – the time from the first trigger until a new trigger is again allowed, extended by any trigger during the interval. This is a paralyzable dead time with pileup rejection, closely matching the classical model[9]. SFDT correctly measures the *time* during which pulses are not recorded, but unless simplified to the assumptions in the classical model, it is not trivial to compute from that the *number* of pulses lost. The detailed mathematical treatment is beyond the scope of this writing.

Please also see a related application note: XIA Pixie-4 Dead Time Correction

[8] OCR(1+Td*ICR) = ICR can be written as OCR = ICR(1-OCR*Td). The measured cumulative dead time DT is DT = OCR*Td*RT. Therefore OCR/ICR = RT-DT/RT.

[9] In the Pixie-4, there was an additional contribution due to DSP readout, which also prohibited the recording of new events, non-paralyzable in nature. The combined effect of these two contributions on SFDT as a mixture of paralyzable and non-paralyzable dead time does *not* follow the simple text book models. This limited the use of SFDT to theoretical and diagnostic purposes.

7 Synchronized Data Acquisition

The Pixie modules are designed for coincidence acquisition, and as such there are several aspects to synchronize:

- Synchronize the channels within a module,
- Synchronize multiple modules within a chassis, and
- Synchronize Pixie modules with external signals.

For example, it is usually required to synchronize clocks and timers between modules and to distribute triggers between channels and modules. It will also be necessary to ensure that runs are started and stopped synchronously in all modules. All these signals are distributed through the PXIe backplane.

7.1 Clock Distribution

Unlike the Pixie-4, the Pixie-4 Express uses the 10 MHz and 100 MHz clocks provided by the PXI Express chassis. These clocks are routed on the backplane to tight tolerances and ensure that all modules receive the same clock with very little phase skew. Every module in the chassis is therefore a clock slave to the backplane, and no jumpers or switches are required to change the clock mode.

7.2 Trigger Distribution

7.2.1 Trigger Distribution Within a Module

Within a module, each channel can be enabled to issue triggers. Such a *Fast Trigger* indicates that the trigger filter just crossed the threshold at the rising edge of a pulse, and is used to start pileup inspection and to latch time stamps, among other things.

Each trigger-enabled channel issues triggers to the central "system logic", which builds an OR of all triggers and sends it back to all channels. If a channels is set to "group trigger" mode, it uses the distributed fast trigger instead of its own local triggers to capture data. In this way, one channel can cause data to be acquired at the same time in all channels of the trigger group. There are then two ways in which the DSP assembles such data: In Run Types 0x400 and 0x401, the DSP reads data from all participating channels individually and stores it as one event record per channel. In Run Type 0x402, the DSP reads data from all 4 channels and stores it as a 4-channel event record. The latter requires a number of specific settings, as listed in section 4.1.2.

Notes:

1. Each channel, trigger enabled or not, always also generates a "hit" flag for a coincidence test when it detects a rising edge. To disable this, mark the channel as not "good" or set the trigger threshold to zero.
2. In group trigger mode, all data is captured based on the distributed trigger. This may cause slight shifts in waveforms and timestamps due to the extra delay of routing signals through the central logic. Run Type 0x402 records local time stamps latched at the locally detected rising edge in addition to the event time stamp at which the distributed trigger latched the event data.

3. If pulses in different channels have slight delays, and all such channels are in group trigger mode, each channel will acquire multiple records, one for each distinct trigger.
4. In systems where a common live time is important (for example, a detector array that should never be partially disabled because that would change its efficiency from event to event), it is recommended to set all channels to group trigger mode and run in Run Type 0xc402. The 500-event buffers will then fill up equally in all channels. Any SDRAM full condition also affects all channels equally, which means the live time will be the same. (Only out-of-range conditions may be different, which can be avoided by reducing the gain to a safe value.) Duplicate records from delayed triggers (if more than one channel is trigger enabled) can then be sorted out offline. Check time stamps for very closely following events in the same channel. Duplicate record should only affect throughput at very high rates. (See also coincidence, below)

7.2.2 Trigger Distribution Between Modules

Fast triggers can also be distributed over the PXIe backplane. The fast trigger signal uses a common backplane line for all modules, which is set up to work as a wired-OR. Normally pulled high, the signal is driven low by the module that issues a trigger. All other modules detect the lines being low and send the triggers to all of their channels. In other words, the backplane line carries a system-wide trigger that essentially acts as a 5th input to the trigger OR in the central "system logic" of each module.

Each module can be enabled to share triggers over the backplane lines or not. In this way, a trigger group can be extended over several modules or each module can form its local sub-group.

7.2.3 Trigger Distribution between PXI chassis

In principle it is possible to distribute triggers between several chassis with Pixie-4 Express modules with a suitable PXIe module to bring out signals from the backplane. Please contact XIA for details.

7.2.4 External Triggers

External trigger signals (3.3V TTL standard) can be connected to a Pixie-4 Express through the PXI backplane. A falling edge on backplane line TRIG0 is recognized as a trigger and can be used to capture waveforms and timestamps in any channel using the group trigger logic. (Sharing triggers over the backplane must be enabled.) Since the Pixie module uses this line to distribute its own triggers, external signals should only drive the line low briefly; a backplane pullup resistor brings it back high (wired-OR).

Starting in release 4.2A, the Pixie-4 Express front panel input can be routed to this backplane line or used as an external trigger within a single module only. The setup is as follows:

<u>Physical Setup</u>

- Connect the external trigger signal to the "In/Out" MMCX connector on the front panel
- The trigger signal must be a 3.3V TTL signal.
- The trigger is recognized at the rising edge of the external signal (0 -> 3.3V transition); the signal must stay high for at least 200ns.

Parameter Setup

- In the **Chassis Setup** panel, enter a nonzero number for **Validation delay for external fast trigger**.
  (There is no actual validation, but to stay consistent with the Pixie-4, any non-zero number turns on external triggering.)
- Optionally, in the **Chassis Setup** panel, enter "1" for Trigger Share mode. This will distribute the external trigger over the chassis backplane to other modules (ORed with any other triggers).
- In the **Trigger** tab, check the **respond to group trigger only** checkbox to enable a channel for distributed (=external) triggering.
- In the **Coincidence** tab, ensure the option to accept events with hitpattern **0 0 0 0** is enabled.
- Optionally, in the **Coincidence** tab, specify the appropriate **channel delays (ns)** to match timing of detector signals with the external trigger.
- If the intention of the external trigger is to capture unconventional detector pulses (with multiple rising edges, going out of range, etc), we recommend
  - In the **Trigger** tab, uncheck the **enable trigger** checkbox.
  - In the **Energy** tab, check the box for **Estimate E if not hit.**
  - In the **Advanced** tab, check the boxes **Allow out of range** and **Disable pileup**.

7.3 Run Synchronization

It is possible to make all Pixie-4 Express modules in a system start and stop runs at the same time by using a wired-OR SYNC line on the PXIe backplane. The feature is enabled by checking the corresponding checkbox in the **Run Control** tab of the Pixie Viewer.

The run synchronization works as follows: When the host computer requests a run start, the Pixie-4 Express's DSP will first execute a run initialization sequence (clearing memory etc). At the beginning of the run initialization the DSP causes the SYNC line to be driven low. At the end of the initialization, the DSP enters a waiting loop, and allows the SYNC line to be pulled high by pullup resistors. As long as at least one of all modules is still in the initialization, the SYNC line will be low. When all modules are done with the initialization and waiting loop, the SYNC line will go high. The low->high transition will signal the DSP to break out of the loop and begin taking data.

If the timers in all modules are to be synchronized at this point, check the corresponding checkbox in the **Run Control** tab of the Pixie Viewer. This instructs the DSP to reset all timers to zero when coming out of the waiting loop. This is implemented as a pulse on an additional backplane line distributed to all modules. From then on they will remain in synch until the next power cycle or reboot. Timers are also synchronized at boot time.

Whenever a module encounters an end-of-run condition and stops the run it will also drive the SYNC line low. This will be detected in all other modules, and in turn stop the data acquisition.

7.4 External *Gate* and *Veto*

7.4.1 External Gating Scenarios

In the current firmware, we accommodate 3 scenarios of external gating:

- It is common in larger applications to have dedicated external electronics to create event triggers to accept events or vetoes to reject events. For example, external logic may sum the "multiplicity" from all channels that fired at the same time and issue a short validation pulse to accept an event if the multiplicity is high enough). For this purpose, the Pixie-4 Express accepts a global *Veto* signal[10], essentially a pulse for each event which has to be timed correctly for the event acceptance decision in the Pixie-4 Express logic.
- In a second scenario, it may be desirable to reject pulses that occur while a channel specific *Gate* signal is on (or off), for example if a pulse derived from a BGO shield around the detector indicates Compton scattering. When the BGO shield sees a pulse, not all of the energy was deposited in the detector, and therefore this event should be rejected. A digital *Gate* signal derived from the BGO detector is thus coincident with the rising edge of the detector pulse (give or take a cable delay). Usually it would have a final duration, corresponding to the coincidence window with the BGO signal for Compton scattering.
- In a third scenario, the acquisition may have to be inhibited for certain intervals. An example is the on/off cycle of a neutron generator, and events may only be of interest if the generator is off. Both the *Veto* and the *Gate* inputs are available for this purpose, *Veto* as a global signal for the whole system, Gate as a dedicated signal for each channel. *Veto* acts at the time of pulse validation and therefore is not appropriate when nanosecond precision is required; *Gate* acts at the time of the rising edge of the pulse.

As a connection convenience, the *Veto* input path can be used for the *Gate* signal, and the *Gate* signal can be reshaped by delays and re-pulsing to apply timing windows and compensate cable delays. *Veto/Gate* pulses and times can be counted in a variety of ways, as described in section 6.7.1.2. The following sections describe the options for signal shaping, marking of events and rejection of events[11]. They are specified in the **Gate** tap of the PARAMETER SETUP panel (Figure 7-1

[10] In the Pixie-4 this was also called Global First Level Trigger (GFLT)

[11] In the Pixie-4 gate logic, the approach was to reject and count while *Gate*/*Veto* is high and invert the incoming signal to be high at the right time. In the Pixie-4 Express, the approach is reversed: there are now options to reject and count while high or low, and the *Veto* signal is used directly (the *Gate* signal can still be delayed and re-pulsed).

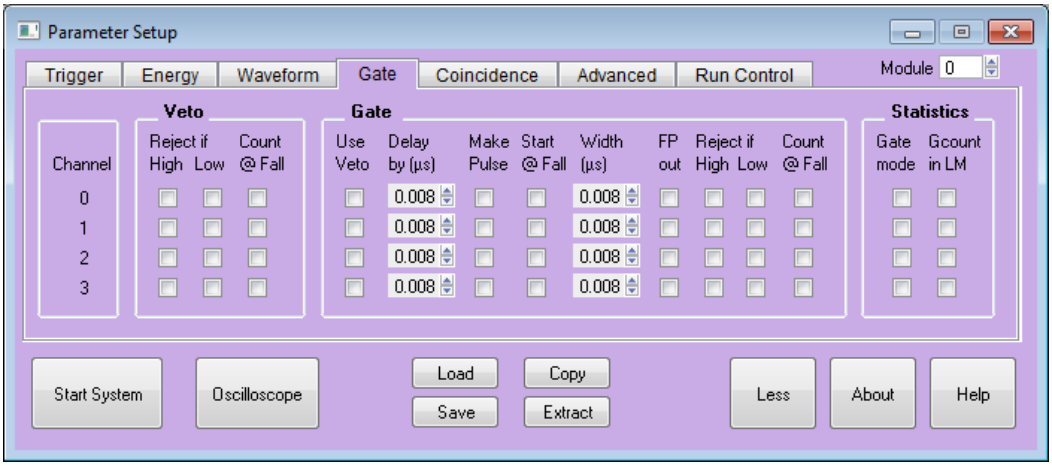


Figure 7-1: **Gate** tab of the PARAMETER SETUP Panel.

7.4.2 Shaping of External Signals

The *Veto* signal is used without any modification as fed into the front panel MMCX connector of the Pixie-4 Express. The *Veto* signal is distributed on the PXI backplane as an inverted wired OR – 3.3V (logic high) on the MMCX connector drives the backplane low, 0V (logic low) releases the backplane to be pulled up by a resistor.

The *Gate* signal is normally taken from the 10-pin front panel connector of the Pixie-4 Express, one pin for each channel. The pinout is described in the appendix. As a firmware option, the *Veto* signal can be used as an alternative input. This is intended to be a simpler connection option for those cases where the same *Gate* signal is used for all channels and the *Veto* logic is not used.

The *Gate* signal can then be delayed by a user defined time. Further there is an option to create a pulse of user defined length, starting at either the rising or the falling edge of the delayed signal. This serves to define a coincidence window for *Gate*. To facilitate setup of the delays and windows, the final delayed and re-pulsed *Gate* signal and the (global) fast trigger of a channel can be made visible on the output pins of the 10-pin front panel connector. (Note that the detector signal can also be delayed using the parameter **Coinc Delay**.)

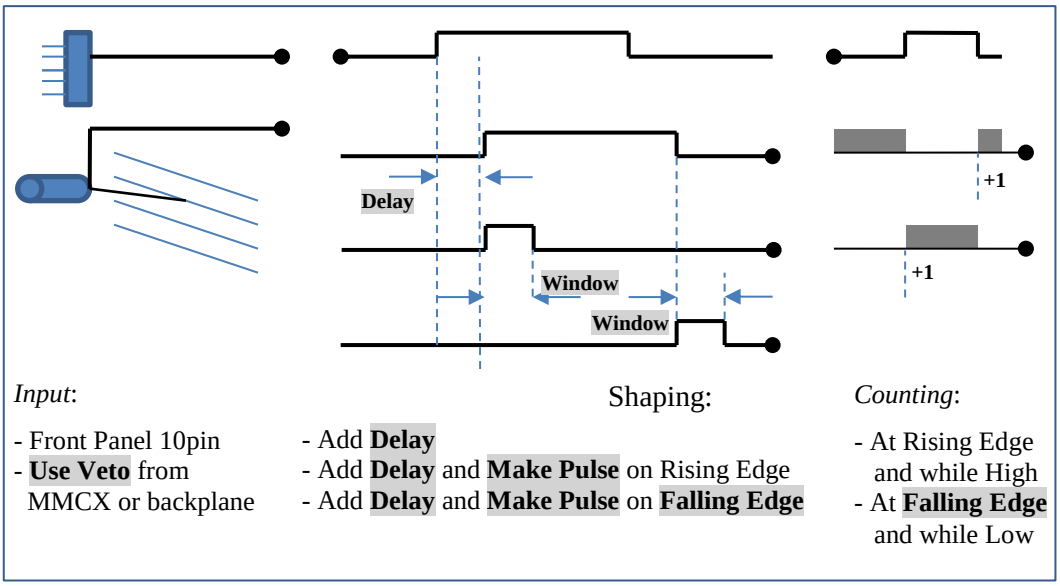


Figure 7-2: Shaping of the Gate signal. The input can be either the front panel 10-pin connector (individually for each channel) or the global Veto line from the MMCX front panel connector or the PXI packplane. The signal can then be delayed and a pulse can be created from the rising or falling edge. Further options allow counting the rising edges and the time high or the falling edges and the time low. The highlighted text corresponds to control variables in the Pixie software.

7.4.3 Marking Events

The status of the *Veto* signal is latched for each event at the time the pileup inspection is finished (energy filter rise time plus flat top after rising edge). This information is recorded in the event status flags of the list mode data

The status of the *Gate* signal (after delay and re-pulsing) is latched for each event at the trigger time (rising edge). This information is recorded in the event status flags of the list mode data

7.4.4 Rejecting Events

While event data is captured for every detected rising edge, not all event data is recorded. User specified options determine which events are acceptable, not only due to pileup and out-of-range conditions, but also due to *Veto* and *Gate*. The rejection is based on the latched *Veto/Gate* status described in the previous paragraph. There are four options to reject events based on *Veto/Gate* status:

- Reject if *Veto* status is low
- Reject if *Veto* status is high
- Reject if *Gate* status is low
- Reject if *Gate* status is high

Obviously, if both low and high rejection are enabled for one signal, nothing is ever recorded.

7.4.5 Counting *Veto/Gate* Pulses and Times

There are two counters in the firmware for *Veto/Gate* related run statistics: GCOUNT and GDT. The basic assumption is that either *Veto* or *Gate* is used, not both at the same time. GCOUNT counts rising (optionally falling) edges of *Veto* OR *Gate**. GDT counts the time during which *Veto* OR *Gate** are high (optionally low). *Gate** here means the signal after delay and re-pulsing. If the option to use the *Veto* input for *Gate* is enabled, only *Gate** is counted, not *Veto* OR *Gate**. For more details, see section 6.7.1.2.

7.4.6 Timing Diagrams for Application Examples

For the first scenario in section 7.4.1, assume external logic sums the multiplicity of the 4 detector channels and issues an "accept" pulse with appropriate delay for the pileup inspection if more than 2 channels fire at the same time; logic high for accept. That "accept" pulse is connected to the Veto input. Pixie-4 Express settings are set to count at the rising edge and reject when low.

For the first event in Figure 7-3, there are 2 pulses in 2 channels so the external logic issues an "accept" pulse to the *Veto* logic. The status of *Veto* at the end of the pileup inspection is thus latched as logic high and the event is marked accordingly. GCOUNT is incremented by one and GDT is increased by the time *Veto* was high. In the event acceptance stage, the event is accepted since *Veto* is high. There is an entry in the list mode data and NOUT is incremented by one.

For the second event, there is only 1 pulse in 1 channel so the external logic issues no "accept" pulse. The status of *Veto* at the end of the pileup inspection is thus latched as logic low and the event is marked accordingly. GCOUNT and GDT are not incremented. In the event acceptance stage, the event is not accepted since *Veto* was low. There is no entry in the list mode data and NOUT is not incremented.

The COUNT TIME is on throughout and all rising edges are counted in FASTPEAKS.

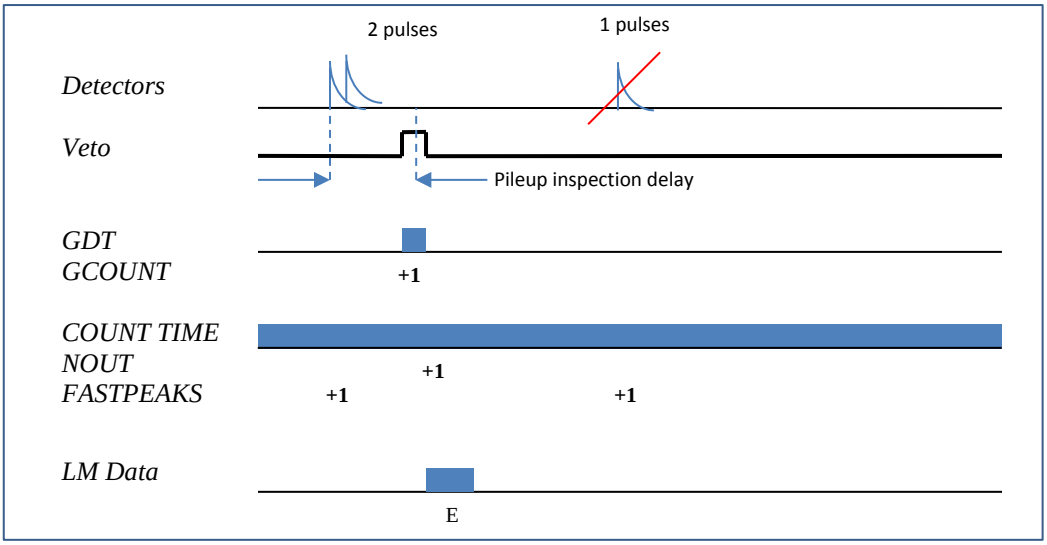


Figure 7-3: Timing for scenario 1 described in the text

7.5 External Status

An external "Status" signal can be distributed to all modules on the PXI backplane. The status of that line is also latched at the time of trigger and included in the event status flags of the list mode data.

7.6 Coincident Events

7.6.1 Coincidences Within a Module

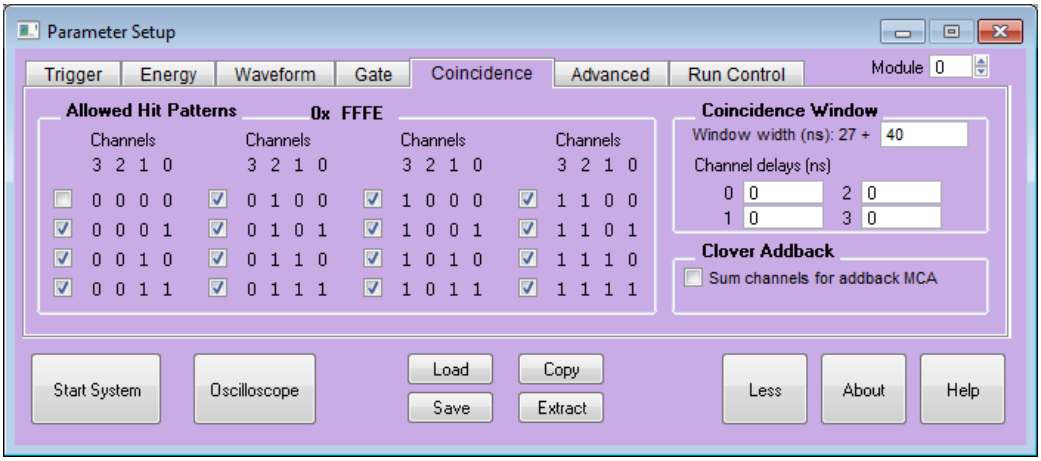


Figure 7-4: Coincidence Pattern and Coincidence Window Settings in the Pixie Viewer

In any given event, a single Pixie module will have up to four channels with a "hit", i.e. a rising edge of a pulse detected in the signal of a channel's ADC. The four channels thus form one of 16 possible Hit Patterns, described in a 4-bit word. In this representation, the Hit Pattern ranges from "no channel hit" [0000] over "only channel 1 hit" [0010] to "all four channels hit" [1111].

The user can define a combination of allowed Hit Patterns, the Coincidence Pattern, to accept one or more Hit Patterns. Usually this is done to reduce the recorded event rate, omitting non-coincident events that are of no interest. In the **Coincidence** Tab of the PARAMETER SETUP Panel, there are 16 checkboxes for the 16 possible hit patterns, and selecting one sets the corresponding bit in the Coincidence Pattern. For example, allowing all except for Hit Pattern [0000] makes the Coincidence Pattern 0xFFFE. Allowing only [0011] makes the Coincidence Pattern 0x0008. Several of the check boxes can be set at the same time, for instance to accept any pattern with two or more channels. If all checkboxes are set, any possible Hit Pattern is acceptable and the Coincidence Pattern is 0xFFFF[12].

Each channel with a pulse above threshold, whether trigger enabled of not, contributes to the hit pattern. A channel hit flag is set to logic high for a user specified time, the Coincidence Window, after the fast trigger. The hit flags from all four channels of a module are continuously tested against the coincidence pattern (local coincidence test). Each channel latches the 4 Hit flags from all 4 channels and the result of the coincidence test in

[12] The case of accepting events with Hit Pattern [0000] is for distributed triggers from another module, to allow recording of data in this module even though none of the local channels saw a pulse.

the middle of its coincidence window. This data become part of the event status flags, and the DSP can decide to accept or reject events based on this information.

The plot below shows the time of arrival difference histogram for an acquisition using one periodic and one quasi-random pulser with a Coincidence Window of 800 ns. The resulting distribution is basically flat with a sharp cutoff at +/- 400 ns. The recorded event rate dropped by a factor ~60 compared to the acquisition without coincidence requirement.

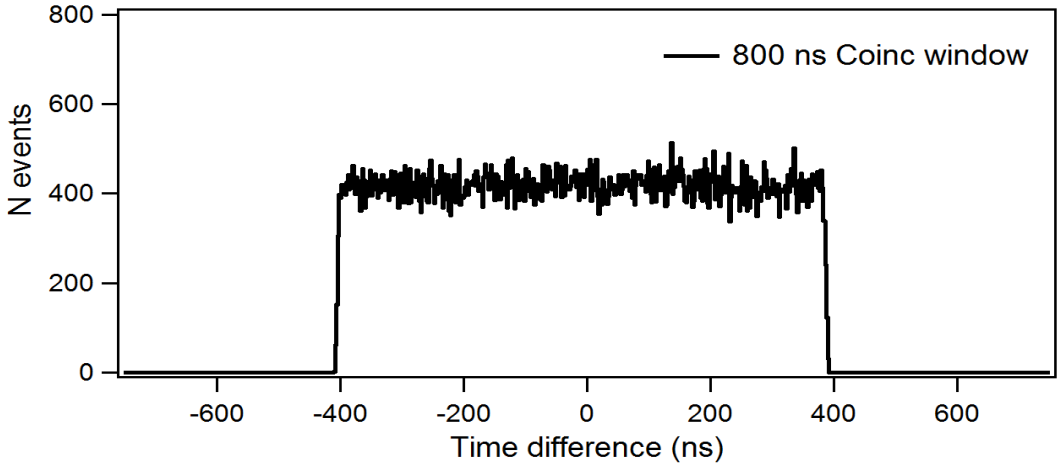


Figure 7-5: Timing distribution for a 800 ns coincidence window

Notes:

- To prevent a channel from contributing to the hit pattern, set the threshold to zero (disables triggers) or uncheck the "good channel" check box.
- The coincidence is based on triggers at the rising edge, but a pulse can subsequently be rejected as piled up. That may lead to coincidence records with missing channels. For example, if channels 0-2 were in coincidence and channel 0 saw a second pulse to be rejected as piled up, then only channels 1 and 2 will be recorded but will show the actual 3-channel hit pattern (bits 8-10 set for channels 0-2). To avoid such missing records, disable pileup rejection.
- The LIST MODE TRACE display applies an independent coincidence window for viewing events. Under the option "show 4 pulses within", events within that range are extracted from the specified file, whereas the Coincidence Window described above limits the acquisition of events into the file. To view coincidence events properly, the range in the LIST MODE TRACE display should be set at least as large as the acquisition coincidence window.
- Coincidence acquisitions can be conducted with independent triggers (EACH channel is recorded when IT is hit AND all channels match the coincidence pattern) or with distributed triggers (ALL channels with the "group trigger" option set are recorded when ANY trigger enabled channel is hit AND all channels match the coincidence pattern).
    - In the former case, waveforms will not appear significantly shifted relative to each other even though they may be a few hundred ns delayed – waveforms are shown vs time from first sample. The time stamps carry the delay information.
    - The latter case will lead to multiple records per coincidence if the delays between channels are greater than a few dozen nanoseconds so triggers are

recognized separately. Event info bit 4 identifies such “group trigger without local hit” records, also their energy is set to zero unless the “estimate energy” option is set. In Run Type 0x402, both the local time and the event time are recorded

- If waveforms are of interest, it is advised to make *Trace Lengths* long enough to cover at least half of the coincidence window. That way, waveforms in event N (triggered by the first channel and recorded for the first channel) and in event N+1 (triggered by the first channel and recorded for the second channel) will start at the same time and will contain the rising edge of trigger and delayed pulse, respectively. Otherwise, matching events becomes somewhat more difficult.

7.6.2 Coincidences Between Modules

Not yet implemented

8 Using Pixie-4 Express Modules with Clover detectors

When working with clover detectors, the Pixie-4 Express can be operated in a specific "clover mode". In this mode, the DSP will calculate the pulse height for each channel as in normal operation, and in addition – for events with hits in more than one channel – calculate the sum of individual channel energies. The result, the full energy of gamma rays scattered within the clover detector, is binned in an additional "addback" spectrum.

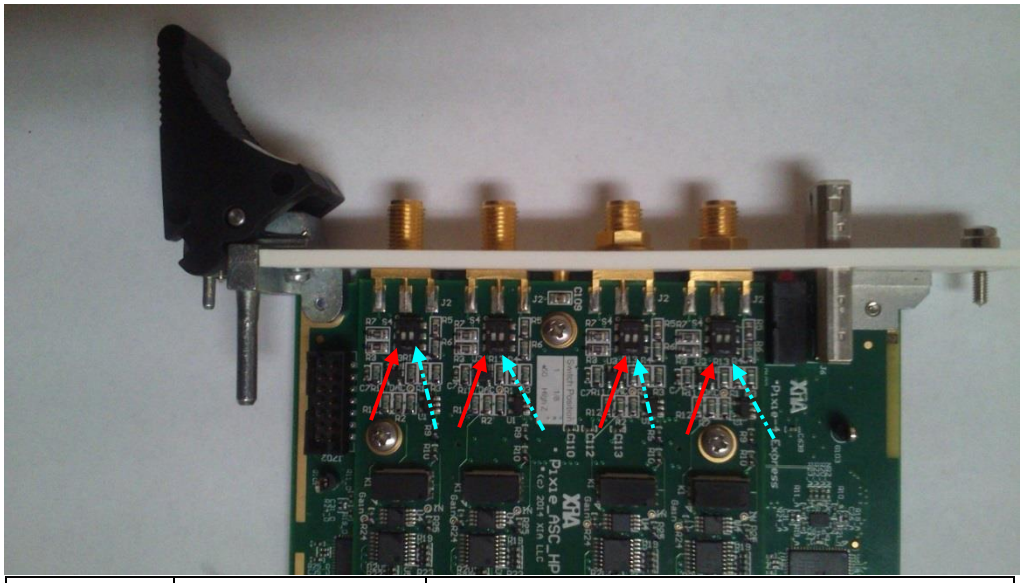
In the current implementation of the clover mode, the spectrum length is fixed to 16Ki. The clover binning mode is designed for run type 0x402. The clover mode is enabled by setting the corresponding checkbox in the **Coincidence** Tab of the PARAMETER SETUP Panel. To automatically gain match the 4 channels (after an initial MCA spectrum has been acquired and a common peak has been fitted), use the button in the CHASSIS SETUP Panel.

Additional clover functions can be developed upon request. XIA maintains a set of Igor functions to plot energies and time differences between channels that can be added to the Pixie Viewer.

Appendices

Appendix A: Hardware information

Front end switches for termination and attenuation



Switch reference	PCB Label	Function
S4 (a)	"1" "1/8"	Attenuation will be 1:1 or 1:8
S4 (b)	"50" … "high Z	Input impedance will be 50Ω. or 2kΩ.

Table 8-1: Analog conditioning selection switches on Pixie-4 Express modules. Switches are marked with solid red (termination) and dashed blue (attenuation) arrows. On the PCB, inverse labels describe the switch positions.

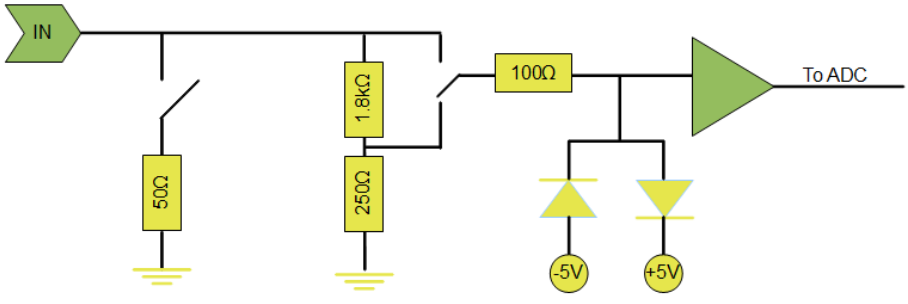


Figure 8-1: Simplified input stage of Pixie-4 Express showing switches, input termination and attenuation, and the overvoltage protection circuit.

Front Panel LEDs

The Pixie-4 Express has 3 LEDs on the front panel.

- A green LED indicates the firmware has been booted correctly and clocks have been programmed.
- A yellow LED indicates that a run is in progress. This can be a very short flash, e.g. for parameter I/O, or continuous, for data acquisition runs.
- A red LED indicates an error occurred. Currently that will be one of two cases:
  a) the module has been powered up, but not booted yet
  b) in a list mode run, the SDRAM has been filled with data so that the acquisition has been paused. Acquisition will resume (and the LED goes off) if the host catches up with data readout.

PXI backplane pin functions

PXI J2 pin	PXIe XJ4 pin	pin name	Connection Type	Pixie pin function
1A		LBL9*	Left neighbor	Event Trigger output (chained OR)
3A		LBR7*	Right neighbor	reserved
16A	7A	TRIG1	Bussed	Clock synchronization
17A	6A	TRIG2	Bussed	Veto
18A	5A	TRIG3	Bussed	Sync
19A		LBL2*	Left neighbor	Sync output (chained OR)
20A		LBR4*	Right neighbor	reserved
21A		LBR0*	Right neighbor	Clock output
16B	7B	TRIG0	Bussed	Fast Trigger
18B	5B	TRIG4	Bussed	Status
20B		LBR5*	Right neighbor	reserved
1C		LBL10*	Left neighbor	Fast Trigger output (chained OR)
3C		LBR8*	Right neighbor	reserved
18C	5C	TRIG5	Bussed	Token
19C		LBL3*	Left neighbor	Control data to PDM (left)
20C		LBL0*	Left neighbor	Clock input
2D		LBL7*	Left neighbor	GATE input channel 3
3D		LBR9*	Right neighbor	Event Trigger input (chained OR)
15D	8D	LBL6	Left neighbor	GATE input channel 2
17D	6D	STAR	Star trigger to slot 2	Hit pattern to slot 2
19D		LBL4*	Left neighbor	GATE input channel 0
21D		LBR2*	Right neighbor	Sync input (chained OR)
2E		LBL8*	Left neighbor	reserved
3E		LBR10*	Right neighbor	Fast Trigger input (chained OR)
15E	8E	LBR6	Right neighbor	reserved
16E	7E	TRIG7	Bussed	Bussed Clock
17E	6E	CLK10	Clock	PXI Clock
19E		LBL5*	Left neighbor	GATE input channel 1
21E		LBR3*	Right neighbor	reserved

Table 8-2: Pins of the J2 or XJ4 backplane connector defined in the PXI(e) standard used by the Pixie-4 Express. A * indicates the pin is not available in PXIe standard

High Density Front Panel Digital Connector

L E E D F G T E	Hit 3 (out)	Hit 2 (out)	Hit 1 (out)	Hit 0 (out)	Timer Clear (out)
	Gate 3 (in)	Gate 2 (in)	Gate 1 (in)	Gate 0 (in)	Gobal trigger (out)

Table 8-3: Pinout of the h10-pin high density front panel connector as seen from the front of the module. The Hit output can be changed to the delayed and re-pulsed Gate signal.

Default IO standard: LVTTL (0..3.3V). Some lines can be configured as LVDS.

By setting the "FP out" bit (ChanCSRC bit 13), the Hit output changes to the final Gate signal applied to the processing logic, after delay, inversion, and windowing.

Matching cable:

- Harting p/n 33 27 243 0500 001 (Harlink 10P MA DB CABLE ASSY, L=0.5m)
- Custom cable assemblies with 2mm pin pitch are possible.

MMCX Coaxial Front Panel Digital Connector

IO standard: LVTTL (0..3.3V).

Default use: input to backplane VETO signal

Matching cables:

- MMCX to SMA: Cinch connectivity 415-0071-006
- MMCX to BNC: Provided by XIA

