

User's Manual

Digital Gamma Finder (DGF)

PIXIE-4

Version 1.61, May 2008

XIA LLC

31057 Genstar Road
Hayward, CA 94544 USA

Phone: (510) 401-5760; Fax: (510) 401-5761
<http://www.xia.com>



Disclaimer

Information furnished by XIA is believed to be accurate and reliable. However, XIA assumes no responsibility for its use, or for any infringement of patents, or other rights of third parties, which may result from its use. No license is granted by implication or otherwise under the patent rights of XIA. XIA reserves the right to change the DGF product, its documentation, and the supporting software without prior notice.

Table of Contents

1	Overview.....	1
1.1	Features.....	1
1.2	Specifications.....	2
2	Setting Up.....	3
2.1	Scope of Document.....	3
2.2	Installation.....	3
2.2.1	Hardware Setup.....	3
2.2.2	Drivers.....	3
2.2.3	Pixie User Interface.....	4
2.3	Getting Started.....	4
3	Navigating the Pixie Viewer.....	8
3.1	Overview.....	8
3.2	Settings.....	8
3.3	Calibrate.....	9
3.4	Run.....	10
3.5	Analyze.....	10
3.6	Optimizing Parameters.....	11
3.6.1	Noise.....	11
3.6.2	Energy Filter Parameters.....	11
3.6.3	Threshold and Trigger Filter Parameters.....	12
3.6.4	Decay Time.....	12
4	Data Runs and Data Structures.....	14
4.1	Run Types.....	14
4.1.1	MCA Runs.....	14
4.1.2	List Mode Runs.....	14
4.1.3	Fast List Mode Runs.....	16
4.2	Output Data Structures.....	17
4.2.1	MCA Histogram Data.....	17
4.2.2	List Mode Data.....	17
5	Hardware Description.....	20
5.1	Analog Signal Conditioning.....	20
5.2	Real-time Processing Units.....	21
5.3	Digital Signal Processor (DSP).....	22
5.4	PCI Interface.....	22
6	Theory of Operation.....	24
6.1	Digital Filters for γ -ray Detectors.....	24
6.2	Trapezoidal Filtering in the Pixie-4.....	26
6.3	Baselines and Preamplifier Decay Times.....	27
6.4	Thresholds and Pile-up Inspection.....	29
6.5	Filter Range.....	31
7	Operating Multiple Pixie-4 Modules Synchronously.....	32
7.1	Clock Distribution.....	32
7.1.1	Individual Clock mode.....	33

7.1.2	Daisy-chained Clock Mode.....	33
7.1.3	Bussed Clock Mode	33
7.1.4	PXI Clock Mode	34
7.2	Trigger Distribution	34
7.2.1	Trigger Distribution Within a Module.....	34
7.2.2	Trigger Distribution Between Modules	36
7.3	Run Synchronization.....	37
7.4	External Gate — GFLT (Veto).....	38
7.5	External Status	38
7.6	Coincident Events	39
7.6.1	Coincidences Within a Module.....	39
7.6.2	Coincidences Between Modules	40
8	Using Pixie-4 Modules with Clover detectors	43
9	Troubleshooting	44
9.1	Startup Problems	44
10	Appendix A	45
10.1	Jumpers	45
10.2	PXI backplane pin functions	46
10.3	Control and Status Register Bits	47

1 Overview

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 is a 4-channel all-digital waveform acquisition and spectrometer card based on the CompactPCI/PXI standard for fast data readout to the host. It combines spectroscopy with waveform digitizing and on-line pulse shape analysis. The Pixie-4 accepts signals from virtually any radiation detector. Incoming signals are digitized by 14-bit 75 MSPS ADCs. Waveforms of up to 13 μ s in length for each event can be stored in a FIFO. The waveforms are available for onboard pulse shape analysis, which can be customized by adding user functions to the core processing software. Waveforms, timestamps, and the results of the pulse shape analysis can be read out by the host system for further off-line processing. Pulse heights are calculated to 16-bit precision and can be binned into spectra with up to 32K channels. The Pixie-4 supports coincidence spectroscopy and can recognize complex hit patterns.

Data readout rates through the CompactPCI/PXI backplane to the host computer can be over 100Mbytes/s. The PXI backplane is also used to distribute clocks and trigger signals between several Pixie-4 modules for group operation. With a large variety of CompactPCI/PXI 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 Features

- Designed for high precision γ -ray spectroscopy with HPGe detectors.
- Directly compatible with scintillator/PMT combinations: NaI, CsI, BGO, and many others.
- Simultaneous amplitude measurement and pulse shape analysis for each channel.
- Input signal decay time: as fast as 150ns and up to 10ms, exponentially decaying.
- Wide range of filter rise times: from 27ns to 109 μ s, equivalent to 12ns to 50 μ s shaping times.
- Programmable gain and input offset.
- Excellent pileup inspection: double pulse resolution of 50 ns. Programmable pileup inspection criteria include trigger filter parameters, threshold, and rejection criteria.
- Digital oscilloscope and FFT for health-of-system analysis.
- Triggered synchronous waveform acquisition across channels, modules and crates.
- Dead times as low as 1 μ s per event are achievable (limited by DSP algorithm complexity). Events at even shorter time intervals can be extracted via off-line ADC waveform analysis.

- Digital constant fraction algorithm measures event arrival times down to a few ns accuracy.
- Supports 32-bit 33 MHz PCI data transfers (>100 Mbytes/second).

1.2 Specifications

Front Panel I/O	
Signal Input (4x)	4 analog inputs. Selectable input impedance: 50 Ω , and 5k Ω , \pm 5V pulsed, \pm 2V DC. Selectable input attenuation 1:7.5 and 1:1.
Logic Input/Output	General Purpose I/O connected to programmable logic (Rev. C, D only). Currently can be either used as input for global backplane signals <u>Veto</u> or <u>Status</u> , or as an input for module specific logic level reported in the data stream
Logic Output	General Purpose output from Digital Signal Processor. Function to be determined.
Backplane I/O	
Clock Input/Output	Distributed 37.5 MHz clock on PXI backplane.
Triggers	Two wired-or trigger buses on PXI backplane. One for synchronous waveform acquisition, one for event triggers.
Status	Global logic level from backplane reported for each event
Token	Global logic level from backplane used for coincidence tests
Synch	Wired-or SYNC signal distributed through PXI backplane to synchronize timers and run start/stop to 50ns.
Veto	Global logic level to suppress event triggering.
Channel Gate	Individual GATE to suppress event triggering for each channel with use of PXI-PDM (Rev. D only)
Interface	
PCI	32-bit, 33MHz Read/Write, memory readout rate to host over 100 Mbytes/s.
Digital Controls	
Gain	Analog switched gain from 0.97 to 11.25 in max. 10% steps. Digital gain adjustment of up to \pm 10% in 15ppm steps.
Offset	DC offset adjustment from $-2.5V$ to $+2.5V$, in 65535 steps.
Shaping	Digital trapezoidal filter. Rise time and flat top set independently: 27ns – 109 μ s in small steps.
Trigger	Digital trapezoidal trigger filter with adjustable threshold. Rise time and flat top set independently from 26ns to 413ns.
Data Outputs	
Spectrum	1024-32768 channels, 32 bit deep (4.2 billion counts/bin). Additional memory for sum spectrum for clover detectors.
Statistics	Real time, live time, input and throughput counts.
Event data	Pulse height (energy), timestamps, pulse shape analysis results, waveform data and ancillary data like hit patterns.

2 Setting Up

2.1 Scope of Document

The scope of this document is Pixie-4 modules with serial numbers 105 through 129 (Revision B), serial numbers 130 through 249 (Revision C), and serial numbers 250-299 (Revision D).

2.2 Installation

2.2.1 Hardware Setup

The Pixie-4 modules can be operated in any standard 3U CompactPCI or PXI chassis. Chassis following only the CompactPCI standard can be used to operate modules individually. To operate several modules together, a backplane following the PXI standard has to be present. Put the host computer (or remote controller) in the system slot of your chassis. Place the Pixie-4 modules into any free slots with the chassis still powered down, then power up the chassis (Pixie-4 modules are not hot swappable). If using a remote controller, be sure to boot the host computer *after* powering up the chassis.

2.2.2 Drivers

When the host computer is powered up the first time after installing the controller and Pixie-4 modules in the chassis, it will detect new hardware and try to find drivers for it. (A Pixie-4 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. Unless directed otherwise by the manufacturer of the controller, this can be done with or without the controller and Pixie-4 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. Install Igor Pro
5. Install the Pixie-4 software provided by XIA (see section 2.2.3)
6. Unless already installed, power down the host computer, install the Pixie-4 modules in the chassis, and power up the system again (chassis first).
7. Windows will detect new hardware (the Pixie-4 modules) and should find the drivers automatically. If not, direct it to the “drivers” directory in the Pixie-4

software distribution installed in step 5. Verify in Window's device manager that the modules are properly installed as "Custom (OEM) PCI 9054 Boards" and have no "resource conflicts". Currently, the driver must be version 4.1.

2.2.3 Pixie User Interface

The Pixie Viewer, XIA's graphical user interface to set up and run the Pixie-4 modules, is based on WaveMetrics' IGOR Pro. To run the Pixie Viewer, you have to have IGOR Version 4.0 or higher installed on your computer. By default, the IGOR Pro should be installed at C:\Program Files\WaveMetrics\IGOR Pro Folder.

The CD-ROM with the Pixie-4 software distribution contains

- 1) an installation program Setup.exe,
- 2) the Pixie-4 software in the folder XIA\Pixie4 and its subfolders.

The Pixie-4 software can be installed by running its installation program. 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. This folder will contain the IGOR control program (Pixie4.pxp), online help files and 5 subfolders (Configuration, DSP, Firmware, MCA, and PulseShape). Make sure you keep this folder organization intact, as the IGOR program and future updates rely on this. Feel free, however, to add folders and subfolders at your convenience.

2.3 Getting Started

To start the Pixie Viewer, double-click on the file *Pixie4.pxp* in the installation folder. After IGOR loaded the Pixie Viewer, the Start Up panel should be prominently displayed in the middle of the desktop.

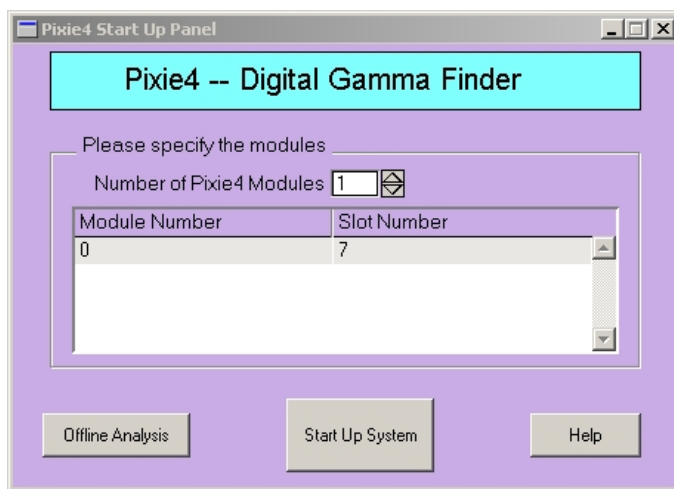


Figure 2.1: The Pixie-4 Start Up Panel.

In the panel, first select the number of Pixie-4 modules in the system. Then specify the slot number in which each module resides.

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-4 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 the “Offline Analysis” button.

After the system is initialized successfully, you will see the main Pixie Control Panel from which all work is conducted. The tabs in the Control Panel are arranged in logical order from left to right. 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 right corner of the Control Panel (inside the “Module” control). Next to the “Module” control is the “Channel” control which indicates the current channel the Pixie Viewer is interacting with. Proceed with the steps below to configure your system.

Note: The “More/Less” button next to the “Help” button on the bottom of the Pixie Control Panel can be used to hide some controls. This may be helpful to those first-time Pixie users who only need to use those major controls in the beginning.

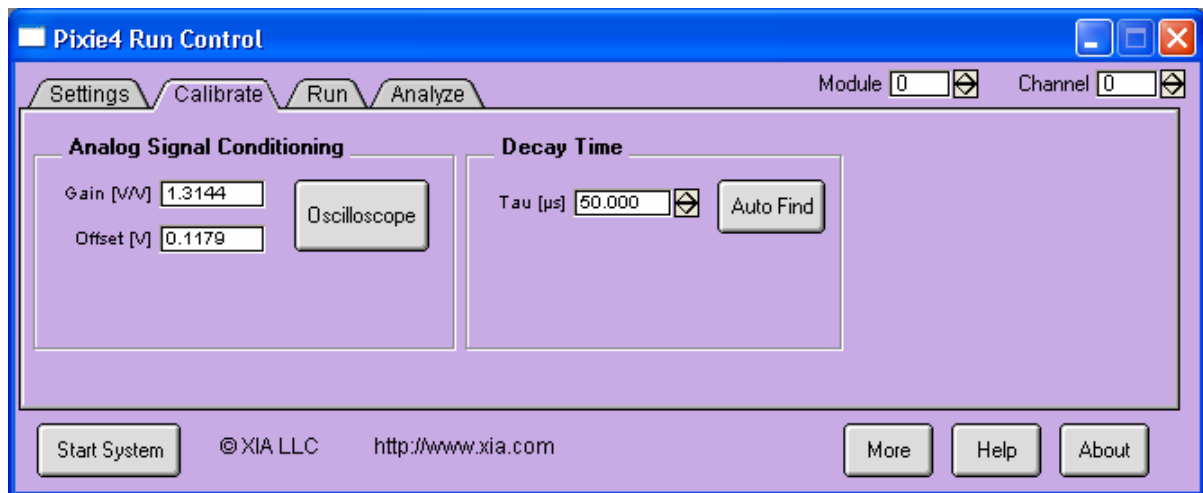


Figure 2.2: The Pixie-4 Run Control Panel, *Calibrate* tab shown

1. In the *Calibrate* tab, click on the Oscilloscope button. This opens a graph that shows the untriggered signal input. Click "Refresh" to update the display. The pulses should fall in the display range. If no pulses are visible or if they are cut off out of the display range, click "Adjust Offsets" to automatically set the DC offset. There is a control called “Offset [%]” on the Oscilloscope which can be used to set the DC-offset level for each channel. If the pulse amplitude is too large to fall in the display range, decrease the "Gain" in the *Calibrate* tab of the Pixie Control Panel. If the pulses are

negative, toggle the “Trigger positive” checkbox in the Channel CSRA Edit Panel which can be accessed by clicking on the Edit button next to “Channel” on the *Settings* tab.

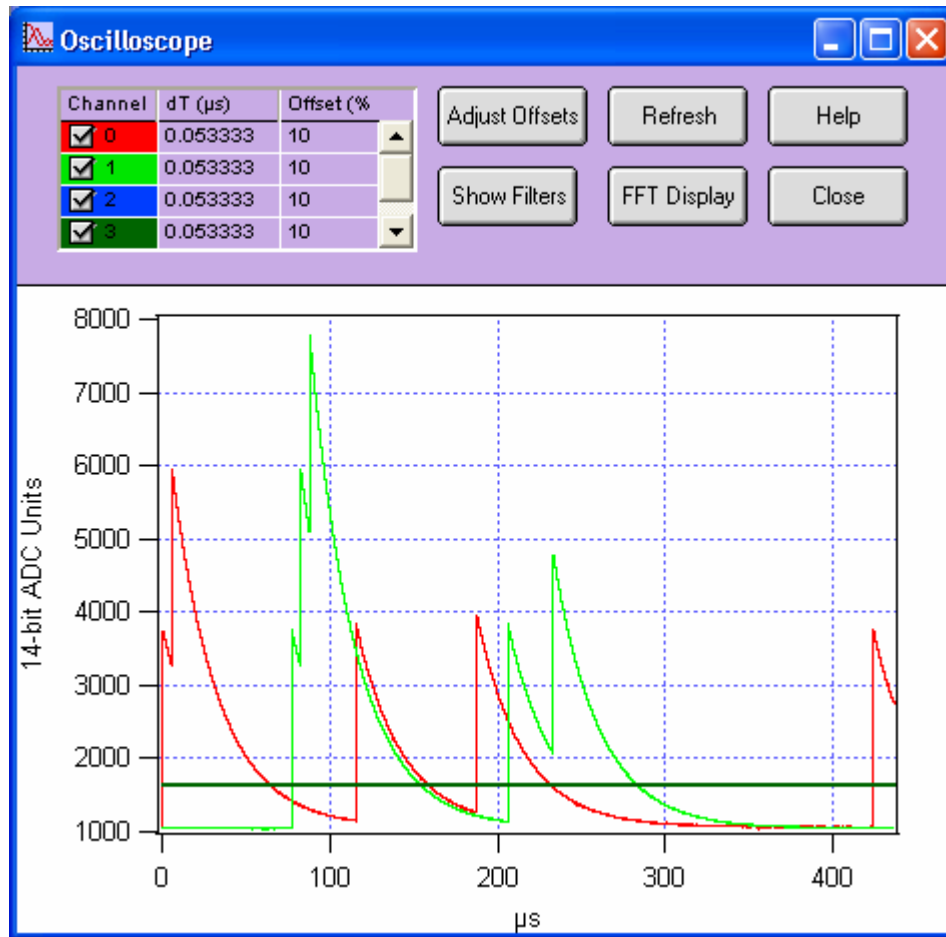


Figure 2.3: Oscilloscope Panel with typical pulses from a pulser.

2. In the *Calibrate* tab, input an estimated preamplifier exponential RC decay time for Tau, and then click on "Auto Find" to determine the actual Tau value for the current channel of the current module. Repeat this for other channels if necessary. You can also enter a known good Tau value directly in the control.
3. Save the modified parameter settings to file. To do so, click on the “Save” button on the *Settings* tab to open a save file dialog. Create a new file name to avoid overwriting the default settings file.
4. Click on the *Run* tab, set "Run Type" to 0x301 MCA Mode, “Poll time” to 1 second, and “Run time/time out” 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. After the run is complete, select the *Analyze* tab and click on the “MCA Spectrum” button. The MCA spectrum shows the

MCA histograms for all four channels. You can deselect other channels while working on only one channel. You can do a Gauss fit on a peak by entering values in the "Min" and "Max" fields as the limits for a Gauss fit. You can also use the mouse to drag the Cursor A and B in the MCA spectrum to the limits of the fit. Click "Fit" to select one or all channels and to perform the fit. Enter the true energy value in the "Peak" field to calibrate the energy scale.

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 etc. as described in section 3.6.

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 “Pixie Run Control” panel. The “Pixie Run Control” panel is divided into 4 tabs, corresponding to the 4 topics summarized below. The *Settings* tab contains controls used to initialize the module, and the file and directory settings. The *Calibrate* tab contains controls to adjust parameters such as gain, DC-offset, preamplifier decay time, histogram control. The *Run* tab is used to start and stop runs, and in the *Analyze* tab are controls to analyze, save and read spectra or event traces.

At the bottom of the panel is a “More” button, which will make all advanced panel controls visible as well. For initial setup and normal operation, these controls are not important.

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.

3.2 Settings

The Pixie Viewer comes up in exactly the same state as it was when last saved to a file using *File->Save Experiment*. However, the Pixie module itself loses all programming when it is switched off. When the Pixie is switched on again, all programmable components need code and configuration files to be downloaded to the module. Clicking on the *Start System* button in the main “Pixie Run Control” panel performs this download.

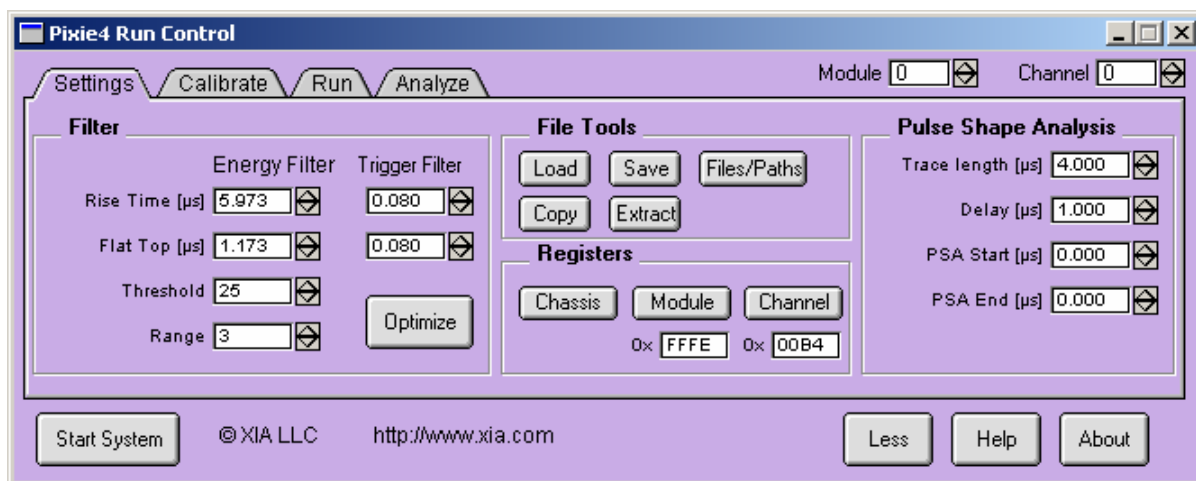


Figure 3.1: The *Settings* tab of the Pixie-4 Run Control Panel.

The Pixie-4 being a digital system, all parameter settings are stored in a settings file. This file is separate from the main 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 in the *Settings* tab. 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 current settings file.

Internally, the module parameters are handled as binary numbers and bitmasks. The *Settings* tab gives access to user parameters in meaningful physical units. Values entered by the user are converted by the Pixie Viewer to the closest value in internal units. Refer to the Online Help for detailed descriptions of the parameters.

3.3 Calibrate

The *Calibrate* tab is used to calibrate or diagnose the system. You can adjust the Gain and DC-Offset on the channel by channel basis. You can use the automatic Tau-Finder routine to find the "Decay Time" of the preamplifier. You can also control the histogram by setting the cut-off energy and binning factor.

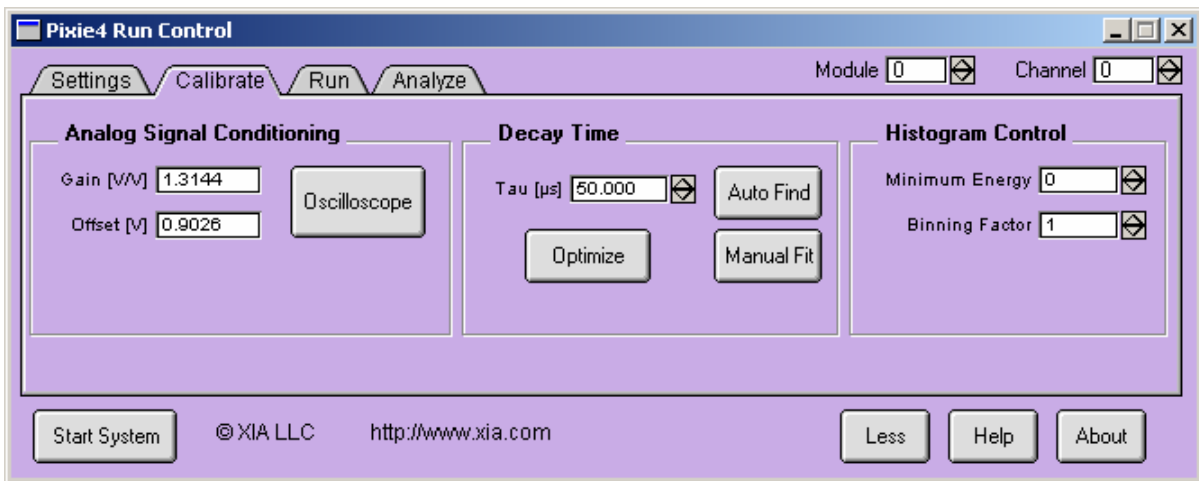


Figure 3.2: The *Calibrate* tab of the Pixie-4 Run Control Panel.

The *Calibrate* tab also has an *Oscilloscope* button linking to a diagnostic graph. The *Oscilloscope* shows a graph of ADC samples which are untriggered pulses from the signal input. The time intervals between the samples can be adjusted; for intervals greater than $0.147\mu\text{s}$ the samples will be averaged over the interval. The main purpose of the *Oscilloscope* is to make sure that the signal is in range in terms of gain and DC-offset. The *Oscilloscope* is also useful to estimate the noise in the system. Clicking on the *FFT Display* button opens the "ADC Trace FFT", where the noise spectrum can be investigated as a function of frequency. This works best if the *Oscilloscope* trace contains no pulses, i.e. with the detector attached but no radioactive sources present.

3.4 Run

The *Run* tab is used to start and stop runs. There are two major run types: List mode runs and MCA runs. MCA runs only collect spectra, List mode runs acquire data on an event-by event basis. List mode runs come in several variants. Before you start a run, you need to select the run type, polling time (the time interval for polling the run status), run time for MCA runs and time out limit and the number of spills (repeated runs) for list mode runs.

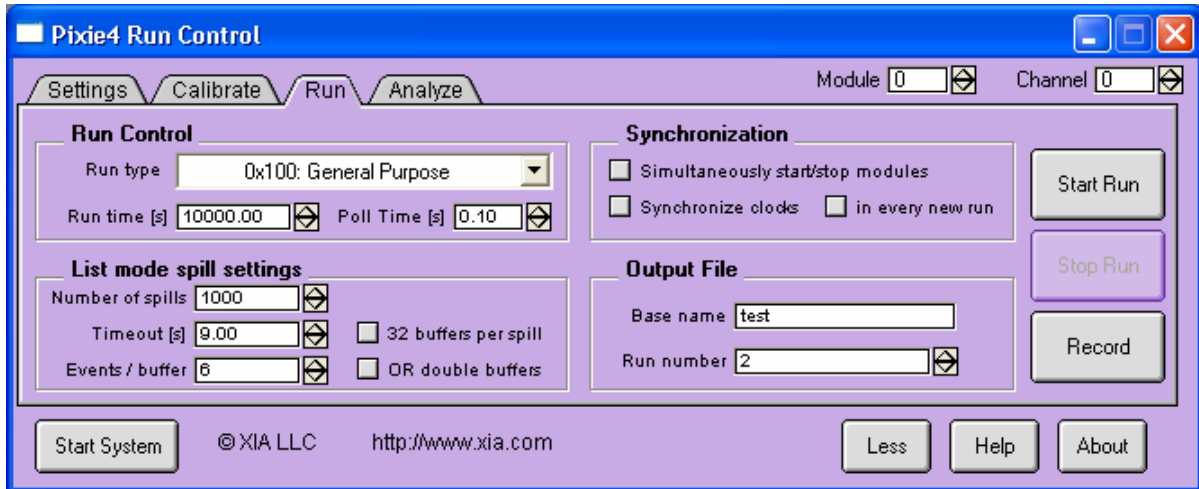


Figure 3.3: The *Run* tab of the Pixie-4 Run Control Panel.

In a multi-module system, you can set all modules to start and stop simultaneously and to reset the timers in all modules with the start of the next data acquisition run by selecting the two options in the *Synchronization* group.

You can choose a base name and a run number in order to form an output file name. The run data will be written to a file whose name is composed of both. The run number is automatically incremented at the end of each run if you select “Auto update run number” on the *Record* panel, but you can set it manually as well. Data are stored in files in either the MCA folder if the run is a MCA run or the PulseShape folder if the run is a List mode run. These files have the same name as the output file name but different extension. For list mode runs, buffer data are stored in a file with name extension “.bin”. For both list mode runs and MCA runs, MCA spectrum data are stored in a file with name extension “.mca” if you select “Auto store spectrum data” on the *Record* panel. Module settings are stored in a file with name extension “.set” after each run if you select “Auto store settings” on the *Record* panel.

3.5 Analyze

The *Analyze* tab is used to investigate the spectrum or to view list mode traces. It also shows the run statistics such as run time, event rate and live time and input count rate for each channel. You can perform Gauss fits on peaks to find the resolution, and calibrate the energy

spectrum by entering a known energy value for a fitted peak. You can also view individual event trace and its energy from a standard list mode run.

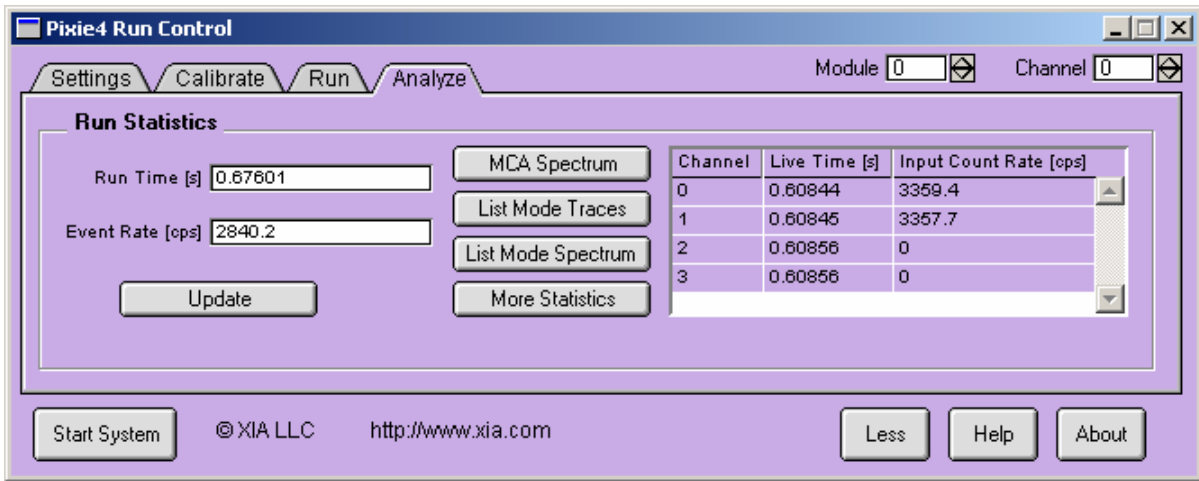


Figure 3.4: The *Analyze* tab of the Pixie-4 Run Control Panel.

3.6 Optimizing Parameters

Optimization of the Pixie-4'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.

3.6.1 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 Display* in the *Calibrate* tab. The graph shows the FFT of the untriggered input signal 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.

3.6.2 Energy Filter Parameters

The main parameter to optimize energy resolution is the rise time of the energy filter. 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 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 1.2 μ s. For a small detector (20% efficiency) a flat top of 0.8 μ s is a good choice. For larger detectors flat tops of 1.2 μ s and 1.6 μ s 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 0.853 μ s, 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 which automatically scans all possible combinations of energy filter rise time and flat top and finds the combination that gives the best energy resolution. This tool can be accessed by clicking the *Optimize* button on the Settings tab. Please refer to the Online Help documentation for more details.

3.6.3 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 minimum edge 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 on slow rising pulses and thus result in a sharper cut off at the threshold in the spectrum.

3.6.4 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 *Calibrate* tab. 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 in the *Calibrate* tab, take a short run, and note the value of τ that gives the best resolution.

Pixie users can also use the *Manual Fit* routine to manually find the decay time through exponentially fitting the untriggered input signals. Another tool is the *Optimize* routine. Similar to the routine for finding the optimal energy filter times, this routine can be used to automatically scan a range of decay times and find the optimal one. Please refer to the Online Help documentation for more details.

4 Data Runs and Data Structures

4.1 Run Types

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

The output data are available in three different memory blocks. The multichannel analyser (MCA) block resides in memory external to the DSP. There is a local I/O data buffer for list mode data located in the DSP, consisting of 8192 16-bit words, and an extended I/O data buffer for list mode runs in the external memory, holding up to 32 local buffers.

4.1.1 MCA Runs

If all you want to do is to collect spectra, you should start an MCA run. For each event, this type of run collects the data necessary to calculate pulse heights (energies) only. The energy values are 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). Run statistics, such as live time, run time, and count rates are kept in the Pixie module.

4.1.2 List Mode Runs

If, on the other hand, you want to operate the Pixie in multi-parametric or list mode and collect data on an event-by-event basis, including energies, time stamps, pulse shape analysis values, and wave forms, you should start a list mode run. In list mode, you can still request histogramming of energies, e.g. for monitoring purposes. In the current standard software, one pulse shape analysis value is a constant fraction trigger time calculated by the DSP, the other is reserved for user-written event processing routines. Other routines exist to e.g. calculate rise times and/or to characterize pulses from phoswich detectors.

4.1.2.1 Compressed Data Formats

The output data of list mode runs can be reduced by using one of the compressed formats described below. The key difference is that as less data is recorded for each event, there is room for more events in the I/O data buffer of the Pixie-4 module and less time is spent per event to read out data to the host computer. For example, if you need individual energies and time stamps, but no waveforms, select “energy and time only” instead of “general purpose” in the Pixie Viewer. In this mode, raw data from the energy filters and waveforms are temporarily stored in an intermediate buffer, and only results are written to the output buffer.

In compressed list mode runs, the following points are to consider:

- When using a runtime that computes results of pulse shape analysis (PSA) computations. make sure the total combined trace length from all four channels is less than 52

microseconds because the intermediate buffer used to temporarily store the trace data is limited to 4k samples.

- If no PSA is required, reduce the trace length to zero to avoid unnecessary data transfer time.
- When the intermediate buffer is filled with events not yet processed for output data, new events are rejected. Whenever a new event occurs, the DSP first checks if there is enough room left in the intermediate buffer, then transfers the data from the FPGAs into its intermediate buffer or rejects it. Consequently, if the combined trace length is more than 26 microseconds, only one event at a time can be stored. This means that the effective dead time for an event is increased by the processing time. If the combined trace length is such that $N \geq 2$ events fit into the intermediate buffer, the processing time does not add to the dead time as long as the *average* event rate is smaller than the processing rate and no bursts of more than N events occur.

4.1.2.2 Data Readout Options

List mode runs halt data acquisition either when the local I/O data buffer is full, or when a preset number of events are reached. The data has then to be read out by the host PC. (Runs can be resumed for longer data acquisitions as described in 4.1.2.3). There are three options for the data readout mode, with different consequences for the readout dead time. (Modes 2 and 3 are only available for module revisions C and D.)

1. The default readout option available for all module revisions is to simply fill and read the local I/O buffer. However, data readout from the local I/O buffer is relatively slow, and since acquisition is halted during the readout, the readout dead time is relatively large (~30ms per module and buffer).
2. A more efficient readout mode is to transfer the data to the external memory when the local buffer is full, and resume the run right away. This is repeated 32 times until the external memory is full, and only then the run is halted and data is read out by the host PC in a fast block read (~30ms per 32 buffers and module, about 550µs between buffers).
3. A third readout mode is to transfer the data to the external memory when the local buffer is full, and resume the run as in mode 2, but after 16 times a flag is raised to the host PC to read out the external memory while new data is stored in the other half of the external memory. This allows (almost) uninterrupted data acquisition. Readout dead time between buffers is about 550 µs, and readout time for 16 buffers is again ~30ms.

Note that at high cont rates and/or for uncompressed data runs with long waveforms, 16 buffers may fill up faster than 30ms. Readout mode 3 is only efficient as long as the time to half fill the external memory is longer than the readout time.

4.1.2.3 Multiple Spills

Runs can be “resumed” by the host after the data is read out. In a resumed run, run statistics are not cleared at the beginning of the run, i.e. it is possible to combine several buffer readouts (“spills”) into one extended run. In the Pixie Viewer this is done automatically when requesting several spills.

Note that for runs with several modules, the buffer ordering in the data depends on the readout option. In mode 1, the data file begins with the first buffer readout of module 0, followed by first buffers of module 1, module 2, ... module N, then the second buffers of modules 0 to N, and so forth. In readout mode 2, list mode runs are repeated 32 times before readout. Therefore the data file will begin with the first **32** buffer readouts of module 0, followed by the first **32** buffers of module 1, module 2, ... module N, then a second **32** buffers of module 0 to N and so forth. In mode 3, list mode runs are repeated 16 times before readout. Therefore the data file will begin with the first **16** buffer readouts of module 0, followed by the first **16** buffers of module 1, module 2, ... module N, then a second **32** buffers of module 0 to N and so forth.

4.1.3 Fast List Mode Runs

In Fast List Mode, data are collected on an event-by-event basis, but waveforms are not read. The output can again be stored in standard or compressed format. This is mainly a legacy run type, and only marginally faster (slightly shorter dead time from event processing) than a normal list mode run recording waveforms of zero length.

To make it faster, a number of steps are omitted during data acquisition. For example, the live time and other run statistics are not updated during the run, and no channel or module coincidences are tested. Also, at high count rate, when the intermediate buffer for event processing fills faster with raw data than being emptied by the processing routine, output data can be corrupted as checks for this condition are omitted. It is therefore important to always keep the input rate below the average processing rate.

Table 4.1: Summary of run types and data formats.

Run Type	Output data	DSP Variables
List Mode (standard)	Energies, time stamps, 6 PSA values, and wave forms in List mode block. Spectra in MCA block	RUNTASK = 256 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
List Mode Compression 1	Energies, time stamps, and 6 PSA values in List mode block. Spectra in MCA block	RUNTASK = 257 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
List Mode Compression 2	Energies, time stamps, and 2 PSA values in List mode block. Spectra in MCA block	RUNTASK = 258 MAXEVENTS = <calculate> (CHANHEADLEN = 4)
List Mode Compression 3	Energies and time stamps in List mode block. Spectra in MCA block	RUNTASK = 259 MAXEVENTS = <calculate> (CHANHEADLEN = 2)
Fast List Mode (standard)	Energies, time stamps, and 6 PSA values in List mode block. Spectra in MCA block	RUNTASK = 512 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
Fast List Mode Compression 1	Energies, time stamps, and 6 PSA values in List mode block. Spectra in MCA block	RUNTASK = 513 MAXEVENTS = <calculate> (CHANHEADLEN = 9)
Fast List Mode Compression 2	Energies, time stamps, and 2 PSA values in List mode block. Spectra in MCA block	RUNTASK = 514 MAXEVENTS = <calculate> (CHANHEADLEN = 4)

Fast List Mode Compression 3	Energies and time stamps in List mode block. Spectra in MCA block	RUNTASK = 515 MAXEVENTS = <calculate> (CHANHEADLEN = 2)
MCA Mode	Spectra in MCA block	RUNTASK = 769 MAXEVENTS=0

4.2 Output Data Structures

4.2.1 MCA Histogram Data

The MCA block is fixed to 32K words (32-bit deep) per channel, i.e. total 128K words. The MCA block resides in the external memory which can be read out via the PCI data bus at rates over 100Mbytes/s. If spectra of less than 32K length are requested, only part of the 32K will be filled with data. This data can be read even when a run is in progress, to get a spectrum update.

In clover mode, spectra for each channel are 16K long and compressed into the first 64K of the external memory. An additional 16K addback spectrum containing the sum of energies in events with multiple hits is accumulated in the second 64K of the external memory.

In 2D mode, spectra for each channel are 16K long and compressed into the first 64K of the external memory. The remaining 64K of the external memory contains a 256 x 256 bin two dimensional spectrum (custom code required)

4.2.2 List Mode Data

The list mode data in external memory consists of 32 local I/O data buffers. The local I/O data buffer can be written by the DSP in a number of formats. User code should access the three variables BUFHEADLEN, EVENTHEADLEN, and CHANHEADLEN in the configuration file of a particular run to navigate through the data set. It should only be read when the run has ended.

The 32 buffers in external memory follow immediately one after the other. The data organization of one I/O buffer is as follows. The buffer content always starts with a buffer header of length BUFHEADLEN. Currently, BUFHEADLEN is six, and the six words are:

Table 4.2: Buffer header data format.

Word #	Variable	Description
0	BUF_NDATA	Number of words in this buffer
1	BUF_MODNUM	Module number
2	BUF_FORMAT	Format descriptor = RunTask + 0x2000
3	BUF_TIMEHI	Run start time, high word
4	BUF_TIMEMI	Run start time, middle word
5	BUF_TIMELO	Run start time, low word

Following the buffer header, the events are stored in sequential order. Each event starts out with an event header of length EVENTHEADLEN. Currently, EVENTHEADLEN=3, and the three words are:

Table 4.3: Event header data format.

Word #	Variable	Description
0	EVT_PATTERN	Hit pattern. Bit [15..0] = [gate pattern hit pattern status read pattern]
1	EVT_TIMEHI	Event time, high word
2	EVT_TIMELO	Event time, low word

The hit pattern is a bit mask, which tells which channels were read out plus some additional status information, as listed in table 4.4. After the event header follows the channel information as indicated by the hit pattern, in order of increasing channel numbers. For example, if bits[3:0] = 1001, the event header is followed by data from channel 0, then channel 3.

Table 4.4: Hit pattern bit description.

Bit #	Description
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 coincidence 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	Logic level of the GATE input of channel 0..3 (for rev. D modules only).

The data for each channel are organized into a channel header of length CHANHEADLEN, which may be followed by waveform data. CHANHEADLEN depends on the run type and on the method of data buffering, i.e. if raw data is directed to the intermediate Level-1 buffer or directly to the linear buffer. Offline analysis programs should therefore check the value of RUNTASK, which is reported in the buffer header. All currently supported data formats are defined below.

1. For List Mode or Fast List Mode, either standard or compression 1, (RUNTASK = 256, 257, 512 or 513), CHANHEADLEN=9, and the nine words are

Table 4.5: Channel header, possibly followed by waveform data.

Word #	Variable	Description
0	CHAN_NDATA	Number of words for this channel
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy
3	CHAN_XIAPSA	XIA PSA value
4	CHAN_USERPSA	User PSA value
5	Unused	N/A
6	Unused	N/A
7	Unused	N/A
8	CHAN_REALTIMEHI	High word of the real time

Any waveform data for this channel would then follow this header. An offline analysis program can recognize this by computing

$N_WAVE_DATA = CHAN_NDATA - 9$.

If N_WAVE_DATA is greater than zero, it indicates the number of waveform data words to follow.

In the current software version, the XIA PSA value contains the result of the constant fraction trigger time computation (CFD). The format is as follows: the upper 8-bit of the word point to the ADC sample before the CFD, counted from the beginning of the trace. The lower 8 bits give the fraction of an ADC sample time between the sample and the CFD time. For example, if the value is 0x0509, the CFD time is $5 + 9/256$ ADC sample steps away from the beginning of the recorded trace.

2. For compression 2 List Mode or Fast List Mode ($RUNTASK = 258$ or 514), $CHANHEADLEN=4$, and the four words are:

Table 4.6: Channel header for compression 2 format.

Word #	Variable	Description
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy
3	CHAN_XIAPSA	XIA PSA value
4	CHAN_USERPSA	User PSA value

3. For compression 3 List Mode or Fast List Mode ($RUNTASK = 259$ or 515), $CHANHEADLEN=2$, and the two words are:

Table 4.7: Channel header for compression 3 format.

Word #	Variable	Description
1	CHAN_TRIGTIME	Fast trigger time
2	CHAN_ENERGY	Energy

5 Hardware Description

The Pixie-4 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 1 shows the functional block diagram of the Pixie-4.

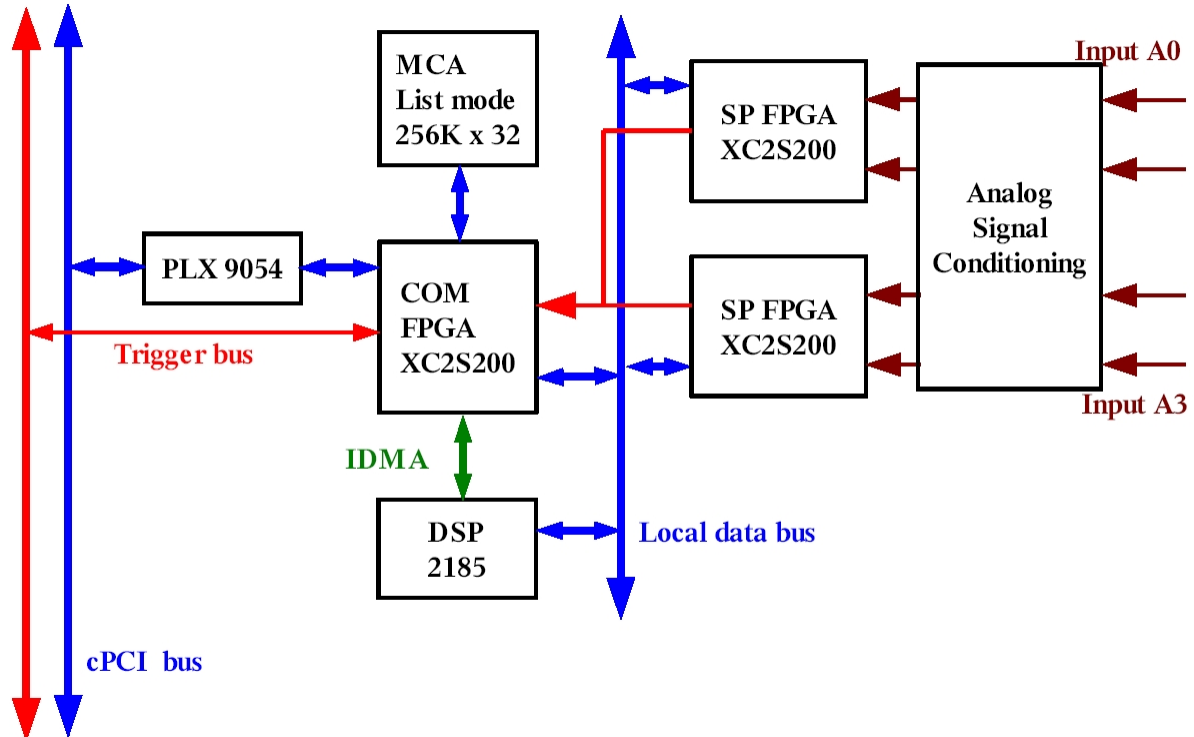


Figure 5.1: Functional block diagram of the Pixie-4 front-end data acquisition and signal processing card.

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 to the input voltage range of the ADC, which spans 2V. Input signals are adjusted for offsets, and there is a computer-controlled gain stage of switched relays. This helps to bring the signals into the ADC's voltage range and set the dynamic range of the channel. A fine tuning of the gain is achieved by multiplying the calculated energy values with digital gain factors in the digital signal processor (DSP).

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, an active Sallen-Key filter, cuts off sharply at the Nyquist frequency, namely half the ADC sampling frequency.

Though the Pixie-4 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 without any further shaping.

5.2 Real-time Processing Units

The real time processing units (RTPUs), one per two channels, consist of a field programmable gate array (FPGA) which also incorporates a FIFO memory for each channel. The data stream from the ADCs is sent to these units at the full ADC sampling rate. Using a pipelined architecture, the signals are also processed at this high rate, without the help of the on-board DSP.

Note that the use of one RTPU for two channels allows sampling the incoming signal at twice the regular ADC clock frequency. If both channels are fed the same input signal, the RTPU can give the second ADC a clock with a 180 degree phase shift, thus sampling the signal twice in one clock cycle. Special software is required to combine the two input streams into one; contact XIA for details.

The real-time processing units apply 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.

Secondly, the RTPUs contain 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 rejected 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.

If a pulse was detected and passed the pileup inspector, a trigger may be issued. That trigger would notify the DSP that there are raw data available now. If a trigger was issued the data remain latched until the RTPU has been serviced by the DSP.

The third component of the RTPU is a FIFO memory, which is controlled by the pile up inspector logic. The FIFO memory is continuously being filled with waveform data from the ADC. On a trigger it is stopped, and the read pointer is positioned such that it points to the

beginning of the pulse that caused the trigger. When the DSP collects event data, it can read any fraction of the stored waveform, up to the full length of the FIFO.

5.3 Digital Signal Processor (DSP)

The DSP controls the operation of the Pixie-4, reads raw data from the RTPUs, reconstructs true pulse heights, applies time stamps, prepares data for output to the host computer, and increments spectra in the on-board memory.

The host computer communicates with the DSP, via the PCI interface, using a direct memory access (DMA) channel. Reading and writing data to DSP memory does not interrupt its operation, and can occur even while a measurement is underway.

The host sets variables in the DSP memory and then calls DSP functions to program the hardware. Through this mechanism all gain and offset DACs are set and the RTPUs are programmed in this way.

The RTPUs process their data without support from the DSP, once they have been set up. When any one or more of them generate a trigger, an interrupt request is sent to the DSP. It responds with reading the required raw data from the RTPUs and storing those in an intermediate buffer. It then returns from the interrupt routine without processing the data to minimize the DSP induced dead time. The event processing routine works from the data in the buffer to generate the requested output data. There are different implementations of the intermediate buffer for the different run types. In standard list mode runs, intermediate and I/O buffer are the same, to avoid moving long waveforms inside the DSP. In other run types, the intermediate buffer is circular, old data is overwritten once it has been processed. If the circular buffer fills up before the data can be processed, no further raw data is read from the RTPUs.

In this scheme, the greatest processing power is located in the RTPUs. Implemented in FPGAs each of them processes the incoming waveforms from its associated ADC in real time and produces, for each valid a 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 Interface

The PCI interface through which the host communicates with the Pixie-4 is implemented in a PCI slave IC together with an FPGA. The configuration of this PCI IC is stored in a PROM, which is placed in the only DIP-8 IC-socket on the Pixie-4 board. The interface conforms to the commercial PCI standard. It moves 32-bit data words at a time.

The interface does not issue interrupt requests to the host computer. Instead, for example to determine when data is ready for readout, the host has to poll a Control and Status Register (CSR) in the communication FPGA.

The communication FPGA links the PCI slave 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 while a run is in progress. The communication FPGA also distributes triggers and coincidence signals to other modules using the PXI backplane connections.

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, HgI₂, 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 $Q_x = E_x/\epsilon$, where ϵ is a material constant. Q_x is integrated onto C_f , to produce the voltage $V_x = Q_x/C_f = E_x/(\epsilon C_f)$. 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).

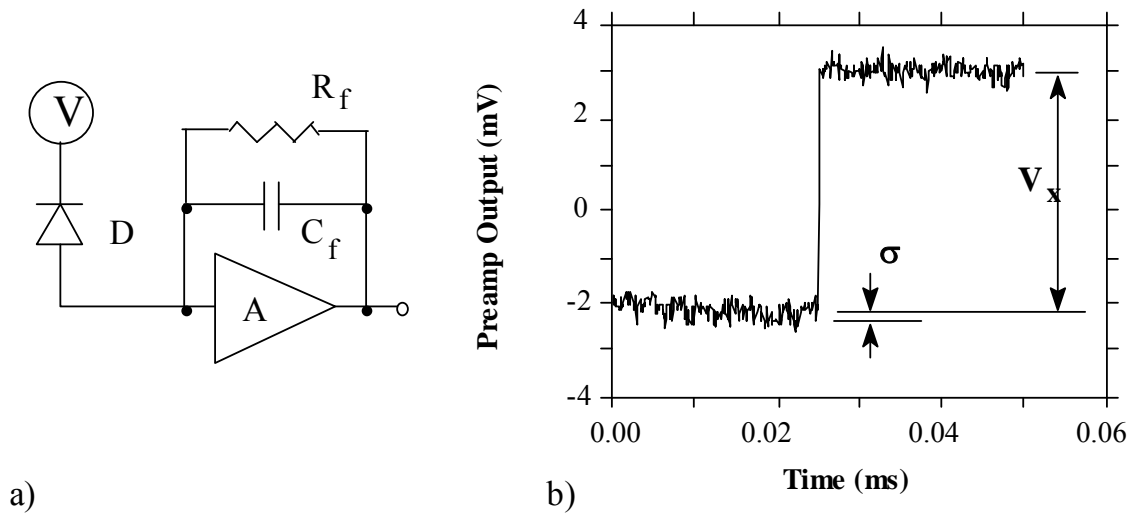


Figure 6.1: (a) Charge sensitive preamplifier with RC feedback; (b) Output on absorption of an γ -ray.

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 MSA (megasamples/sec). 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:

$$V_{x,k} = - \sum_{i(before)} W_i V_i + \sum_{i(after)} W_i V_i \quad (6.1)$$

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.

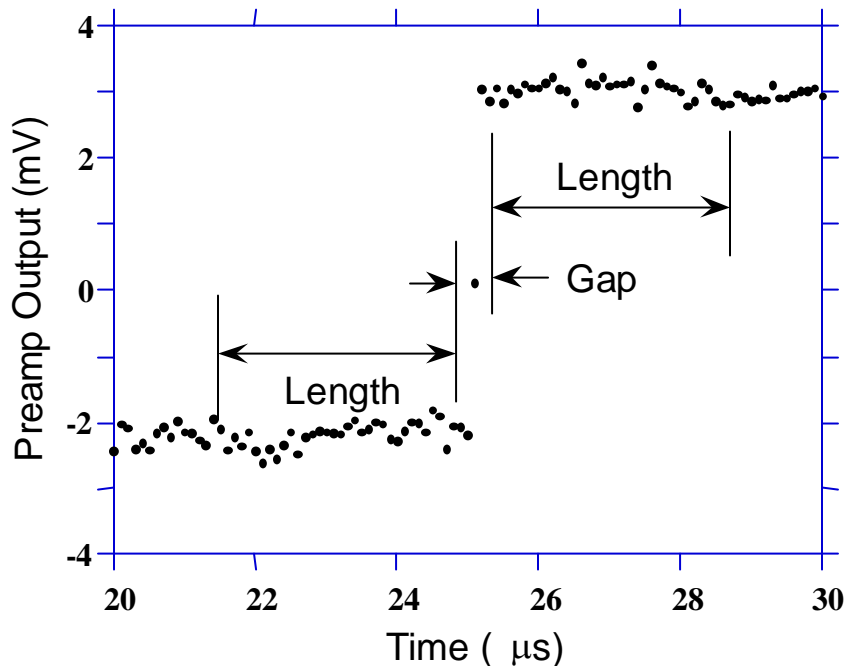


Figure 6.2: Digitized version of the data of **Figure 6.1** (b) in the step region.

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. 6.1. 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. 6.1 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 principal, 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 takes a different approach because it was optimized for very 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:

$$LV_{x,k} = - \sum_{i=k-2L-G+1}^{k-L-G} V_i + \sum_{i=k-L+1}^k V_i \quad (6.2)$$

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. 6.2 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 the Pixie-4

From this point onward, we will only consider trapezoidal filtering as it is implemented in the Pixie-4 according to Eqn. 6.2. The result of applying such a filter with Length $L=1\mu s$ and Gap $G=0.4\mu s$ 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 flat top equal to G , and a symmetrical fall time equal to L . The base width, 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-4 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.

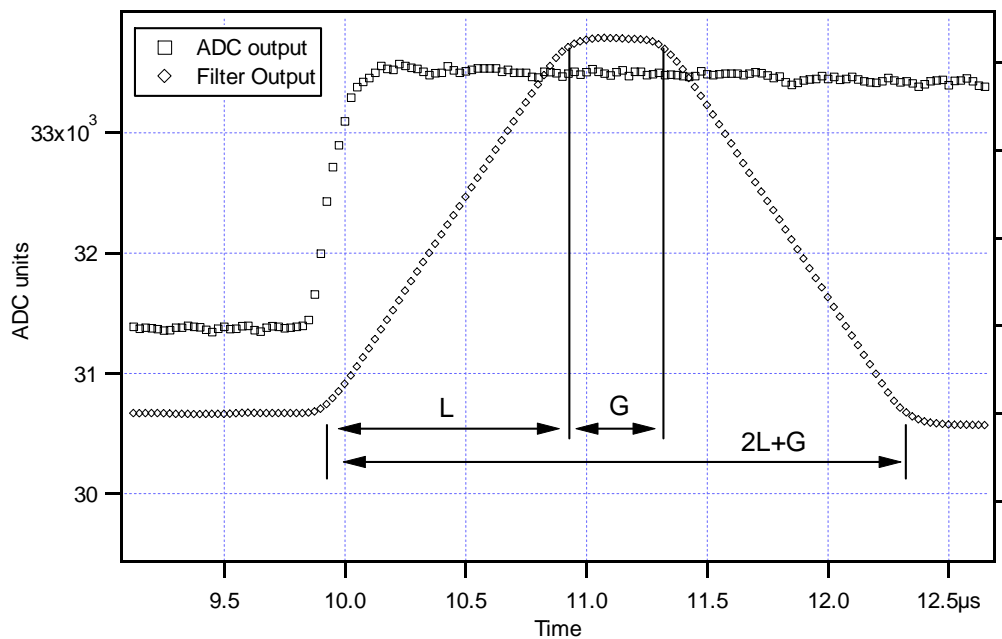


Figure 6.3: Trapezoidal filtering of a preamplifier step with $L=1\mu\text{s}$ and $G=0.4\mu\text{s}$.

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 signal is termed 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

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

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.

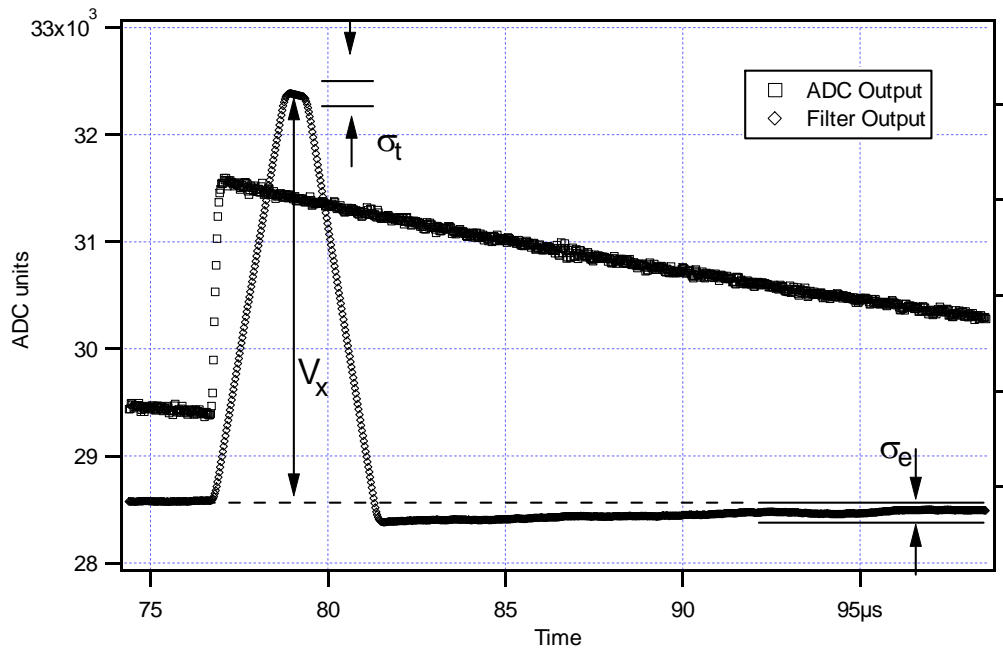


Figure 6.4: A γ -ray event displayed over a longer time period to show baseline noise and the effect of preamplifier decay time.

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.

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, the filter sums are continuously updated by the RTPU (see section 5.2), and only have to be read out by the DSP when an event occurs. Reconstructing the energy and incrementing the spectrum is done by the DSP, so that the RTPU is ready to take new data immediately after the readout. This usually takes much less than one filter rise time, so that no system deadtime is produced by a “capture and store” operation. This is a significant source of the enhanced throughput found in digital systems.

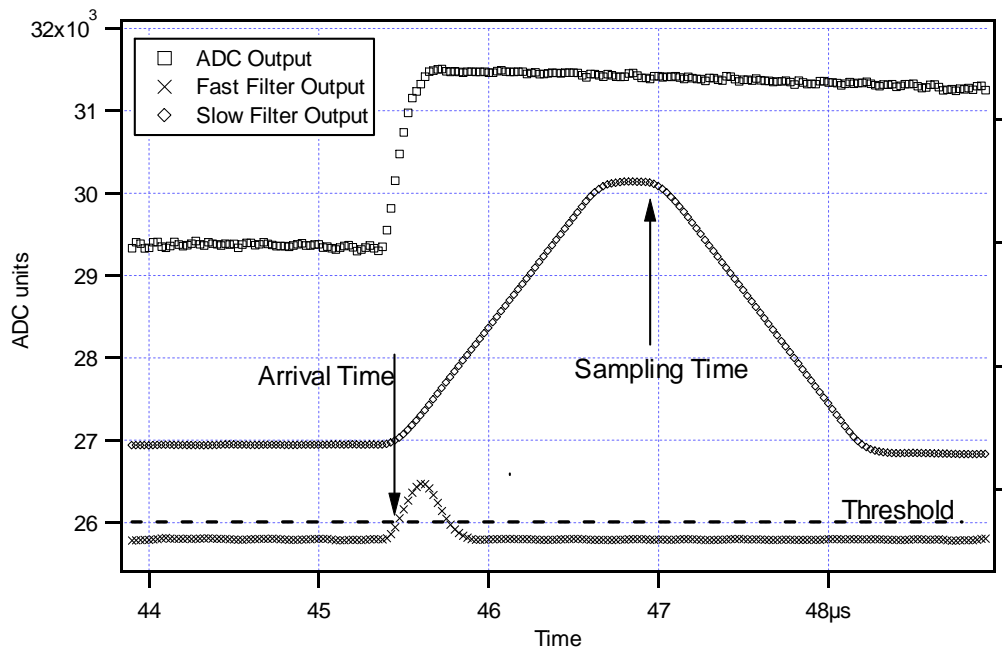


Figure 6.5: Peak detection and sampling in the Pixie-4.

The peak detection and sampling in the Pixie-4 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 $L_f = 0.1\mu\text{s}$ and a gap $G_f = 0.1\mu\text{s}$. The slow filter has $L_s = 1.2\mu\text{s}$ and $G_s = 0.35\mu\text{s}$.

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 counter to count 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 and the rise time of the preamplifier pulses. The slow filter value captured following PEAKSAMP is then the slow digital filter's estimate of V_x .

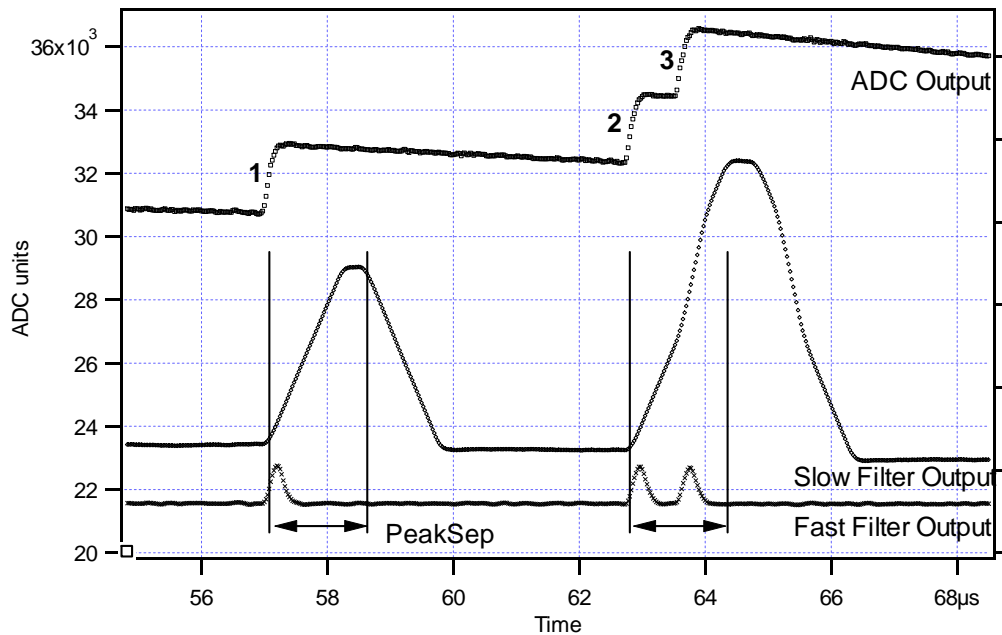


Figure 6.6: A sequence of 3 γ -ray pulses separated by various intervals to show the origin of pileup and demonstrate how it is detected by the Pixie-4.

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 $L_f = 0.1\mu\text{s}$ and a gap $G_f = 0.1\mu\text{s}$. The slow filter has $L_s = 1.2\mu\text{s}$ and $G_s = 0.35\mu\text{s}$.

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 $1.0\ \mu\text{s}$, are thus seen to pileup in the present example with a $1.2\ \mu\text{s}$ 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 $0.1\ \mu\text{s}$, these γ -ray pulses do not pileup in the fast filter channel. The Pixie-4 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. 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 $1.0\ \mu\text{s}$. 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 the wide range of filter rise times from $0.053\ \mu\text{s}$ to $106\ \mu\text{s}$, the filters are implemented in the RTPUs or FPGA configurations with different clock decimations (filter ranges). The ADC sampling rate is always 13.3ns , but in higher clock decimations, several ADC samples are averaged before entering the 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 are listed in Table 6.1.

Table 6.1: RTPU clock decimations and filter time granularity

Filter range	Filter granularity	max. $T_{\text{rise}} + T_{\text{flat}}$	min. T_{rise}	min. T_{flat}
1	$0.0267\ \mu\text{s}$	$3.387\ \mu\text{s}$	$0.0533\ \mu\text{s}$	$0.08\ \mu\text{s}$
2	$0.0533\ \mu\text{s}$	$6.773\ \mu\text{s}$	$0.1067\ \mu\text{s}$	$0.16\ \mu\text{s}$
3	$0.1067\ \mu\text{s}$	$13.547\ \mu\text{s}$	$0.2133\ \mu\text{s}$	$0.32\ \mu\text{s}$
4	$0.2133\ \mu\text{s}$	$27.093\ \mu\text{s}$	$0.4267\ \mu\text{s}$	$0.64\ \mu\text{s}$
5	$0.4267\ \mu\text{s}$	$54.187\ \mu\text{s}$	$0.8533\ \mu\text{s}$	$1.28\ \mu\text{s}$
6	$0.8533\ \mu\text{s}$	$108.373\ \mu\text{s}$	$1.7067\ \mu\text{s}$	$2.56\ \mu\text{s}$

All filter ranges are implemented in the same FPGA configuration. Only the “FILTERRANGE” parameter of the DSP has to be set to select a particular filter range.

7 Operating Multiple Pixie-4 Modules Synchronously

When many Pixie-4 modules are operating as a system, it may be required to synchronize clocks and timers between them and to distribute triggers across 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 PXI backplane.

7.1 Clock Distribution

In a multi-module system, there will be one clock master and a number of clock slaves or repeaters. The clock function of a module can be selected by setting jumpers JP1, JP2 and JP3 near the back of the board. Pin 2 of JP2 is the input to the board's clock distribution circuitry. It can be connected with a shunt to several other pins, thus choosing a particular clock distribution mode. The preferred clock distribution mode is the PXI clock mode if the chassis supports it, else the daisy-caned clock mode.

These jumpers differ slightly for modules of Revision B and Revision C/D. The clock functions themselves as described below are identical and compatible for both revisions.

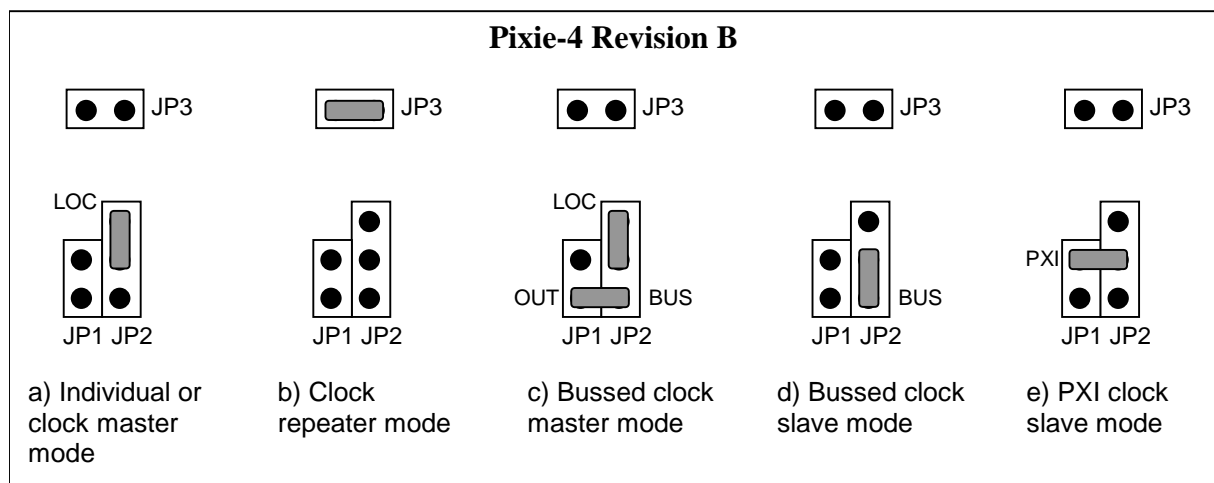


Figure 7.1. Jumper Settings for different clock distribution modes of Revision B modules. In a group of modules, there will be one daisy-chained clock master (a) in the leftmost position and several repeaters (b) OR one bussed clock master (c) and several bussed clock slaves (d). Modes (a/b) and (c/d) can not be mixed. Mode (e) can only be used with a backplane providing 37.5 MHz instead of the usual 10MHz. Jumper JP3 is located near the chip U2; it has no label.

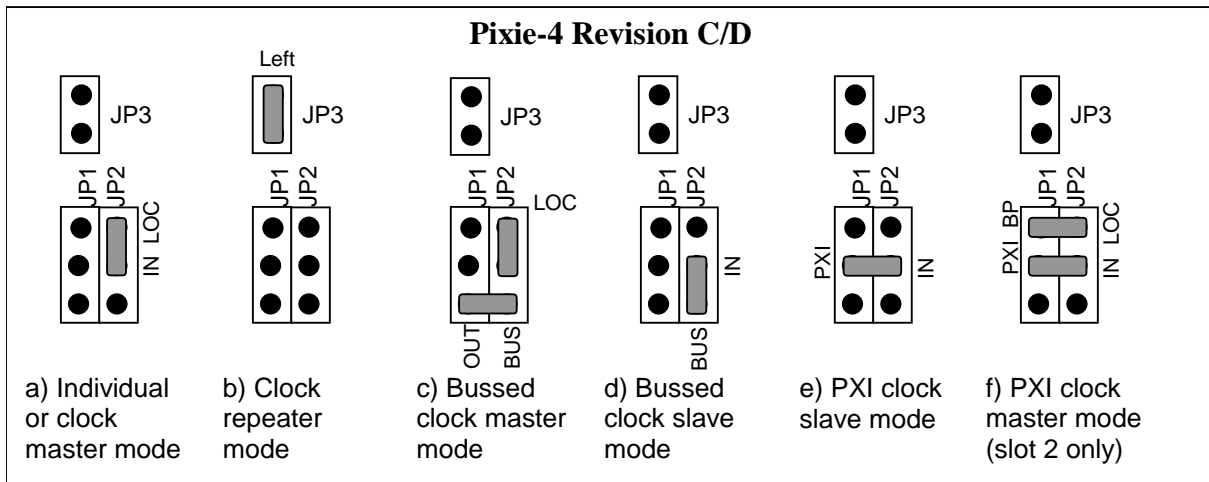


Figure 7.2. Jumper Settings for different clock distribution modes of Revision C modules. In a group of modules, there will be one daisy-chained clock master (a) in the leftmost position and several repeaters (b) OR one bussed clock master (c) and several bussed clock slaves (d) OR one PXI clock master (f) in slot 2 and several PXI clock slaves (e).

Modes (a/b), (c/d), or (e/f) can not be mixed. Mode (e) can also be used with a custom module in slot 2 or a backplane providing 37.5 MHz instead of the usual 10MHz.

7.1.1 Individual Clock mode

If only one Pixie-4 module is used in the system, or if clocks between modules do not have to be synchronized, the module should be set into individual clock mode, as shown in Figures 7.1 (a) and 7.2 (a). Connect pin 2 of JP2 (the clock input) with a shunt to pin 3 of JP2, which is labeled “LOC”. This will use the on-board clock crystal as the clock source.

7.1.2 Daisy-chained Clock Mode

The preferred way to distribute clocks between modules is to daisy-chain the clocks from module to module, where each module repeats and amplifies the signal. This requires one master module, located in the leftmost slot of the group of Pixie-4 modules, with the same jumper settings as an individual module, see Figures 7.1 (a) and 7.2 (a). Configure the other modules in the chassis as clock repeaters by setting the jumpers as shown in Figures 7.1 (b) and 7.2 (b); i.e. remove all shunts from JP 1 and JP2 and set a shunt on JP3, located on top of the clock crystal U2.

Note that the clock output is always enabled, i.e. every board, independent of its clock mode, sends out a clock to its right neighbor as long as it has a clock itself. Thus make sure that no other module sits to the right of a Pixie module that uses the PXI_LBR0 line on the PXI backplane for other purposes.

7.1.3 Bussed Clock Mode

If there have to be gaps between a group of Pixie-4 modules, the daisy-chained clock distribution will not work since the chain is broken. In this case, the modules can be configured for bussed clock mode. To do so, configure one module (in any slot) as the bussed

clock master as shown in Figures 7.1 (c) and 7.2 (c), i.e. set one shunt to connect pins 2 and 3 on JP2 and a second shunt to connect Pin1 of JP1 and JP2 together (“OUT” to “BUS”). The clock master will drive the clock signal to a line that is bussed to all others slots in the chassis. The drive strength is limited to 3-4 modules. All other modules should be configured as clock slaves, i.e. set a shunt to connect pin 1 and 2 of JP2 (“BUS” to clock input) as shown in Figures 7.1 (d) and 7.2 (d).

Note that the bussed clock line does usually not connect over a PCI bridge in chassis with more than 8 slots, whereas daisy-chains usually do.

Always make sure that there is no shunt on JP3, in order to disconnect from the incoming daisy-chained clock, else there will be a conflict between the two clock signals.

7.1.4 PXI Clock Mode

A further option for clock distribution is to use the PXI clock distributed on the backplane. The PXI clock is driven by default by the PXI backplane at a rate of 10 MHz - too slow to run the Pixie 4 modules. However, clock signals are individually buffered for each slot and clock skew between slots is specified to be less than 1ns, making it the preferred distribution path for sensitive timing applications. If a custom backplane provides 37.5 MHz, or if a module in slot 2 overrides the default signal from the backplane, this clock can be used as an alternative setting for Pixie-4 modules. Such a module can be a Revision C Pixie module configured as PXI clock master, shown in Figure 7.2 (f), a XIA PXI-PDM power and logic module, or any other suitable custom module.

The PXI clock master has to be configured as shown in Figure 7.2 (f). The PXI clock slaves are configured by connecting pin 2 on JP1 and JP2 together (“PXI” to clock input), as shown in Figures 7.1 (e) and 7.2 (e).

Always make sure that there is no shunt on JP3, in order to disconnect from the incoming daisy-chained clock, else there will be a conflict between the two clock signals.

7.2 Trigger Distribution

7.2.1 Trigger Distribution Within a Module

Within a module, each channel can be enabled to issue triggers. Two kinds of triggers are distributed: First, a *Fast Trigger* indicating the trigger filter just crossed the threshold, which is used to start pileup inspection and to stop the FIFOs for waveform acquisition, among other things. Second, an *Event Trigger* indicating that pileup inspection was passed, i.e. validating the event as acceptable. The Event Trigger also tells the DSP that data is ready for readout in the Trigger/Filter FPGA.

If channels are put in “group trigger” mode, each trigger enabled channel issues both kinds of triggers to the central Communication FPGA, which builds an OR of all triggers and sends it

back to all channels. The channels then use the distributed fast triggers and event triggers instead of their own local triggers to capture data. In this way, one channel can cause data to be acquired at the same time in all other channels of the trigger group. The DSP then reads data from all participating channels and stores it as one event record. Each channel, trigger enabled or not, always also generates a “hit” flag if pileup inspection was passed, and DSP readout is conditional to this flag.

Even in group trigger mode, some data is captured based on the channel’s local trigger. For example, the pulse height of a pulse is best determined based on the trigger from the pulse itself, not from a common group trigger that may be delayed – if several channels in a group are trigger enabled, always the *last* fast trigger before the *first* event trigger in this event is the one that counts. The following table lists which quantities are based on local or group trigger in group trigger mode. (If not in group trigger mode, all quantities are based on local triggers.)

Note that for the trigger timing to be correct, all channels in a trigger group should be set to the same energy filter rise time and flat top. Furthermore, to capture the energy of delayed channels, ensure that the coincidence window (see section 7.6) is long enough to include the maximum expected delay.

Channel	Energy	Waveform	Timestamp
Hit and trigger enabled	Based on local trigger	Based on (last) group trigger	Based on (last) group trigger unless “local trigger only” option selected
	Always read out	Always read out	Always read out
Hit, but not trigger enabled	Based on local trigger	Based on (last) group trigger	Based on (last) group trigger unless “local trigger only” option selected
	Always read out	Always read out	Always read out
Not hit (no pulse or pileup)	Based on group trigger if “estimate energy” option selected, else zero. Only read if “read always” option selected	Based on (last) group trigger Only read if “read always” option selected	Based on (last) group trigger unless “local trigger only” option selected Only read if “read always” option selected

Energy: The pulse height of a pulse is best determined based on the trigger from the pulse itself, not from a common group trigger that may be delayed. Even if a channel is not trigger enabled, energy filter values are latched some time after the local trigger filter crosses the user defined threshold. The channel is then marked as “hit” for the DSP to read out.

Even channels without “hit” are read out if the “read always” option is set for this channel. In this case, the channel’s energy is usually reported as zero since there was no valid local trigger to capture the value of the energy filter. However, since the energy filter is computed continuously, setting the “estimate energy” option cause the energy filter value to be captured based on the (last) group trigger. This might be useful for channels with occasional very small pulses (below the threshold), or possibly to obtain energy estimates on piled up pulses.

Note that since the timing of the group trigger is not precise with respect to the non-triggering pulse, the energy reported is only a rough estimate. It might help to set the flat top time to a large value to make the capturing of the energy filter less time sensitive.

Waveforms: The waveforms are always captured based on the (last) group trigger for this event. Channels without “hit” are read out only if the “read always” option is set for this channel.

Timestamp: Normally, in acquisitions with shared group triggers, all channels record the (identical) timestamp of the (last) group trigger for this event. Since waveforms are captured based on the group trigger, this ensures that within a data record traces and timestamps are correlated. However, if no waveforms are recorded, there is no time-of-arrival information of possible delays between channels from the timestamps alone. In this case, setting the “local trigger only” option (a module-wide option) preserves the time difference information by recording timestamps for each channel based on its local trigger only.

7.2.2 Trigger Distribution Between Modules

Both fast triggers and event triggers can also be distributed over the PXI backplane. Each trigger 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 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 Communication FPGA 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 across PXI segment boundaries

In PXI chassis with more than 8 slots, the PCI bus as well as the PXI bussed backplane lines are divided into segments with not more than 8 slots. While the PCI bus is bridged between the segments, the PXI bussed backplane lines are usually only buffered from one segment to the next; i.e. the line in one segment drives the line in the neighboring segment. Since this buffer is essentially a one-way communication (though the direction may be selectable), no wire-OR can be build across the segment boundary. (Note: Sometimes there is no connection at all.)

For applications with more than 7 modules, the Pixie-4 have to be operated in a chained OR mode, where trigger signals are passed from module to module using the PXI nearest neighbor lines which are not interrupted by the segment boundaries. In this mode, each module ORs the trigger signal from its right neighbor with its own contributions and passes it to the left. The leftmost module issues the combined OR to a bussed PXI line. The chassis has to be configured such that the leftmost segment drives all other segments to the right. The Pixie-4 modules can be set up to operate in this mode using the chassis control panel of the

Pixie Viewer. The PXI backplane buffering has to be set up with the tools provided by the chassis manufacturer: the lines named PXI_TRIG0 (fast trigger), PXI_TRIG1 (event trigger) and PXI_TRIG2 (synchronization) have to be set up to be driven from the leftmost segment.

7.2.4 Trigger Distribution between PXI chassis

In principle it is possible to distribute triggers between several chassis with Pixie-4 modules using XIA's PXI PDM module. Please contact XIA for details.

7.3 Run Synchronization

It is possible to make all Pixie-4 modules in a system start and stop runs at the same time by using a wired-OR SYNC line on the PXI backplane. In all modules the variable SYNCHWAIT has to be set to 1. If the run synchronization is not used SYNCHWAIT must be set to 0. The variable is set by checking the corresponding checkbox in the *Run* tab of the Pixie Viewer.

The run synchronization works as follows. When the host computer requests a run start, the Pixie-4'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, set the variable INSYNCH to 0 by checking the corresponding checkbox in the *Run* tab of the Pixie Viewer. This instructs the DSP to reset all timers to zero when coming out of the waiting loop. From then on they will remain in synch if the system is operated from one master clock.

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.

Note that if the run synchronization is not operating properly and there was a run start request with SYNCHWAIT=1, the DSP will be caught in an infinite loop. This can be recognized by reading the variables INSYNCH and SYNCHWAIT. If after the run start request the DSP continues to show INSYNCH=0 and SYNCHWAIT=1, it is stuck in the loop waiting for an OK to begin the run. Besides rebooting, there is a software way to force the DSP to exit from that loop and to lead it back to regular operation. See the Programmer's Manual for details.

7.4 External Gate — GFLT (Veto)

It is common in larger applications to have dedicated external electronics to create event triggers or vetoes. While the Pixie-4 does not currently accept an external fast trigger, it does accept a global first level trigger (GFLT). This signal acts as a validation for an event already recognized by the Pixie-4. Based on multiplicities and other information the dedicated trigger logic needs to make the decision whether to accept or reject a given event. If that decision can be made within a filter rise time of all Pixie-4 channels involved, then the GFLT input can be used. To accept signals, the GFLT signal applied to the Pixie-4 must be logic 0 (0V) at the time when the event data are latched in the RTPUs. This happens not before a filter rise time has passed since the event arrival and not later than a rise time plus the flat top. Therefore the trigger logic should generate a GFLT pulse that is logic 0 during the filter flat top.

The GFLT signal is distributed through the PXI backplane. Using XIA's PDM module or a custom board, external signals can be connected to the backplane. In addition, the Pixie-4 front panel input labeled "DSP-OUT" [sic] can be used to send the signal to the backplane. The input signal must be LVTTTL, i.e. logic 0 = 0V, logic 1 = 3.3V. Only one module within a chassis may use this option to avoid conflicts in driving the backplane. The option is enabled in software by setting the corresponding checkbox in the Pixie Viewer's *Chassis Setup Panel*. Setting it for one module will automatically disable it for all other modules.

Each channel can be programmed individually to require the presence of a GFLT in order to latch event data. This is achieved by setting the corresponding checkbox in the Pixie Viewer's Channel Control Register panel.

7.5 External Status

A second function for the Pixie-4's front panel input is to contribute to a wired-OR backplane line called "Status". Several modules can be enabled to contribute to the Status line. The backplane status line will be logic 1 whenever the "DSP-OUT" input of any enabled module is high (3.3V). The status of this line is read as part of the event acquisition and is stored in the list mode data.

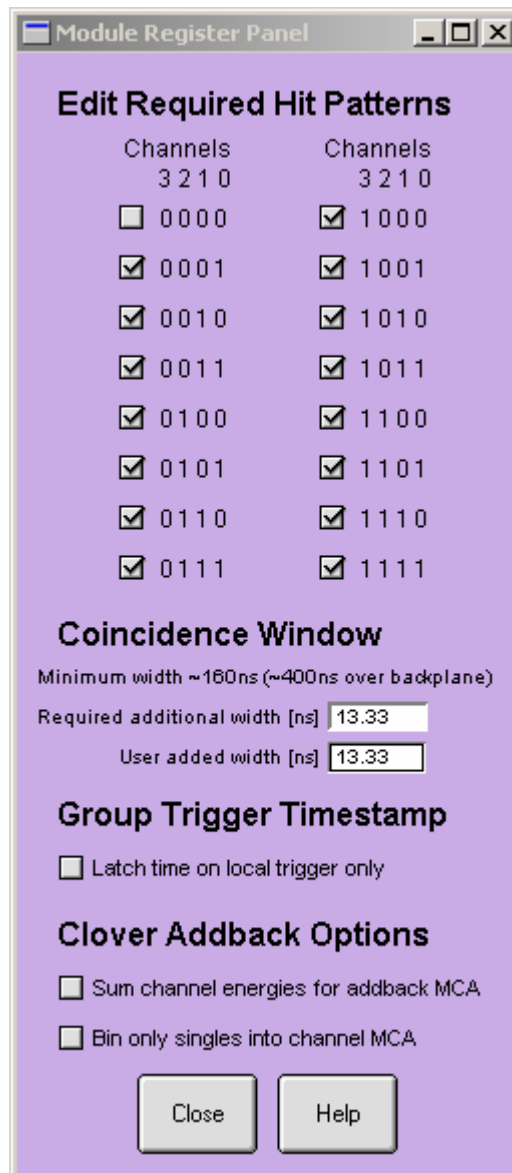
The third function for the Pixie-4's front panel input is to contribute one "Front" bit in the event hit pattern. If the front panel is not used as Veto or Status input, this allows recording of an externally created logic level in each module individually. For example, each module may be assigned to a detector or radiation source that is enabled/disabled individually, and so the status of that detector is recorded in the event data stream.

Notes: The front panel input can be used for Veto, Status, and Front at the same time, if necessary. The wired-OR backplane lines are of type active low, i.e. logic 1 is 0V.

7.6 Coincident Events

7.6.1 Coincidences Within a Module

In any given event, a single Pixie-4 module will have up to four channels with a “hit”, i.e. a valid local pulse without pileup. The four channels thus form one of 16 possible Hit Patterns, stored in the lower 4 bits of a DSP parameter. In this representation, the Hit Pattern ranges from “no channel hit” [0000] over “only channel 1 hit” [0010] to “all four channels hit” [1111]. For each event, the Hit Pattern is checked against the user defined “Coincidence Pattern” to determine if it is acceptable (Local Test). If acceptable, the event is recorded and processed, if not, the event is rejected and data acquisition continues.



The screenshot shows a software window titled "Module Register Panel" with a purple background. It contains several sections for configuring coincidence events:

- Edit Required Hit Patterns:** A table with two columns of checkboxes and 4-bit binary patterns (0000 to 1111). The left column has checkboxes for 0001, 0010, 0011, 0100, 0101, 0110, and 0111. The right column has checkboxes for 1000, 1001, 1010, 1011, 1100, 1101, 1110, and 1111.
- Coincidence Window:** Includes a note "Minimum width ~160ns (~400ns over backplane)", a text field for "Required additional width [ns]" with the value 13.33, and a text field for "User added width [ns]" with the value 13.33.
- Group Trigger Timestamp:** Includes a checkbox labeled "Latch time on local trigger only".
- Clover Addback Options:** Includes two checkboxes: "Sum channel energies for addback MCA" and "Bin only singles into channel MCA".
- At the bottom are "Close" and "Help" buttons.

Figure 7.3. Coincidence Pattern and Coincidence Window Settings in the Pixie Viewer

The user can define the Coincidence Pattern to accept one or more hit patterns. For example, in the *Module Register Edit Panel* of the Pixie Viewer, there are 16 checkboxes for the 16 possible hit patterns, and selecting one sets the corresponding bit in the coincidence pattern. In the example shown in Figure 7.3, accepting only Hit Pattern [0001] makes the Coincidence Pattern 0x0002. 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.

Each channel with a pulse above threshold, whether trigger enabled or not, contributes to the hit pattern the moment the pulse is validated as not piled up (i.e. an energy filter time after rising edge of pulse). The hit pattern is read for comparison with the coincidence pattern about 50 ns after the first pulse is validated. If several channels contribute to an event, the minimum coincidence window -- the time period in which (delayed) channels can contribute to the hit pattern -- is thus ~50ns. When sharing triggers over the backplane, it takes longer to route the triggers to all modules, so the minimum width is ~300ns.

A difference in peaking times between channels will cause even channels with simultaneous pulses to contribute to the hit pattern at different times. The Pixie Viewer thus calculates the required additional window width to compensate for any such difference, displays it, and ensures that the additional window width is at least this value. If longer delays between channels are expected from the physics of the experiment, this added width can be increased up to a value of ~870 microseconds (16 bit counter).

If the required additional width is later decreased by reducing the difference in peaking times, a smaller coincidence window is possible. However, to avoid modifying a large coincidence window intentionally set by the user, the value entered by the user is never decreased automatically.

Notes:

- 1) Any added coincidence window width will increase the time required to process an event and thus reduce the maximum count rate.
- 2) In run types 0x100-0x300, pulses contributing during the readout of data (after the end of the coincidence window) are lost. The readout may take several microseconds, longer if waveforms are to be recorded.
- 3) The cut off at the end of the coincidence window is precise to within $13.3\text{ns} * 2^{(\text{Filter Range})}$, e.g. ~100ns in range 3.

7.6.2 NEW: Coincidences Between Modules

If more than one module is operated in the same PXI chassis, acceptance of events can also be subject to the results of a system-wide (“global”) coincidence test. The result of the global test is distributed over the TOKEN backplane line. This module coincidence test takes place in the following steps:

1. After receiving a valid event trigger and waiting for the user defined coincidence window, each module sends its channel hit pattern to slot 2 of the chassis using the PXI STAR trigger line
2. Each module determines the result of the local coincidence test based on its own 4 channels. If enabled to do so, it signals the test result on the TOKEN line. If the local test passed, the TOKEN line is left pulled up (3.3V, logic 1) else, the TOKEN line is driven low (logic 0).
3. The module in slot 2, typically XIA's PXI-PDM module, uses the up to 48 bit hit pattern from up to 12 modules (slots 3-14) to make an accept/reject decision. If the hit pattern is acceptable, the TOKEN line is left pulled up. If not acceptable, the TOKEN line is driven low (logic 0). The decision criteria is based on a user defined control word, downloaded to the PXI-PDM by its neighboring Pixie-4 module.
The acceptance decisions implemented in the current PDM firmware are described in detail in the Pixie Viewer online help. For example, if the control word is 0x13 (0x14, 0x15, etc) events are only acceptable if at least 3, (4, 5, etc) channels are hit. In case of 0x0200, channel 0, but not 1, 2, and 3 must be hit in each module 0 and 1. The current firmware does not claim to cover all cases. Please contact XIA to request additional cases or to obtain verilog source code to write custom PDM firmware.
4. Each module, after waiting ~100ns for the global accept/reject decision to be made, captures the status of the TOKEN line and puts it in the event hit pattern. The hit pattern also contains the status of the backplane STATUS line, the result of the local coincidence test, and the status of the front panel input at this moment. Depending on user settings, the event will be recorded or discarded if the TOKEN and/or LOCAL bits are set in the hit pattern.

A full implementation of this feature thus requires an additional module in slot 2 of the chassis, receiving hit patterns over the PXI STAR Trigger lines, making a coincidence decision, and signaling the result on the TOKEN line. This can be XIA's PXI-PDM module or any other compatible PXI module. A limited coincidence decision can be made with Pixie-4 modules only, e.g. one or more "master modules" inhibiting acquisition in all other modules based on their local hit pattern.

In the Pixie Viewer, the module coincidence is configured in the Chassis Register Panel (Fig. 7.4). With the checkboxes in the "Module Coincidence Setup" block, each module can be set to accept events if

- a) only the local coincidence test is passed (check "local")
- b) only the global coincidence test is passed (check "global")
- c) either the local OR the global coincidence test is passed (check "global" and "local")
- d) the global test AND all relevant local tests pass (check "global" and "local adds to global")

Other checkboxes and controls define if a module sends its hit pattern to slot 2, if a module is writing the global coincidence control word to a neighboring PDM, and the control word to write.

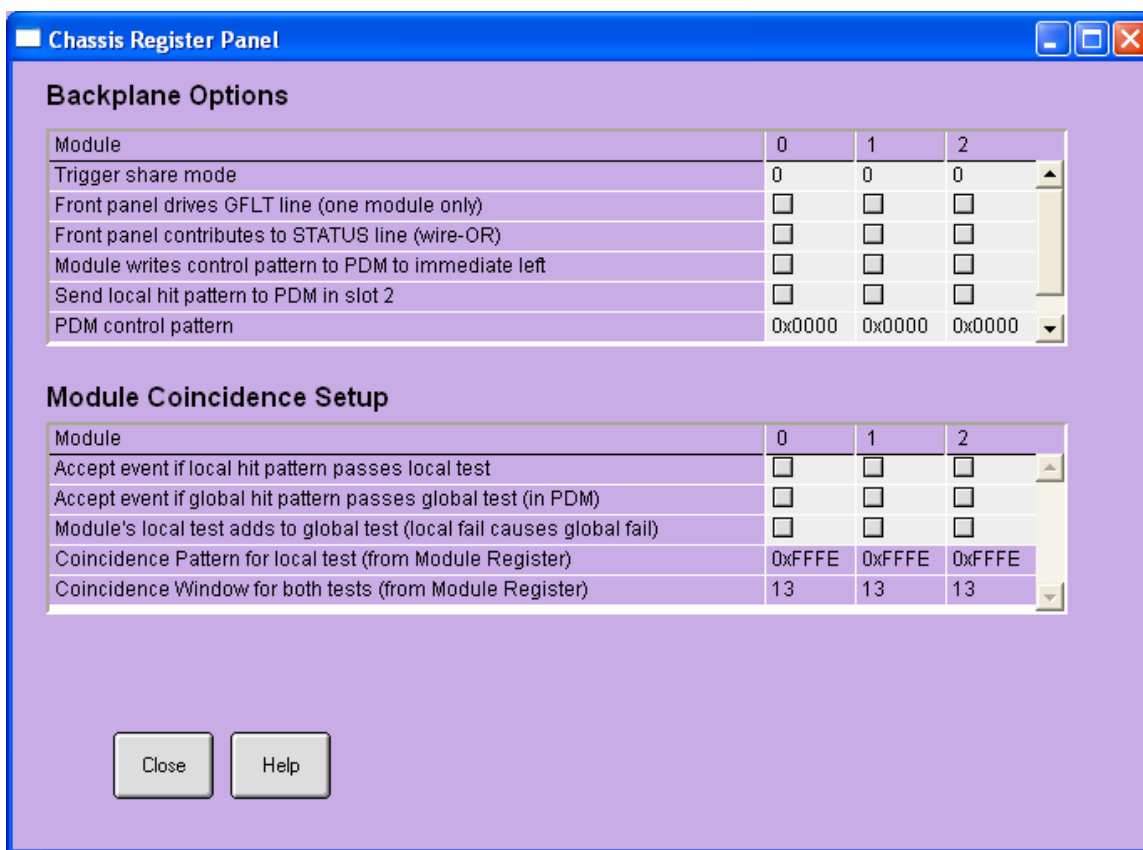


Figure 7.4 Module coincidence setup in the Pixie Viewer

Examples:

1. To require a local coincidence of channels 0-1, 2-3, or both (as above), set the coincidence pattern to 0x9008 in the ModuleRegisterPanel and check only the "local test" box in the Chassis RegisterPanel
2. To require coincidence of channels 0 and 1 in Module 0, and no other channel/module matters, in the ModuleRegisterPanel set the coincidence pattern in Module 0 to 0x8888 and in all other modules to 0xFFFF. In the Chassis Register Panel, check the "global test" box for all modules and the "local adds to global" box for module 0. No PXI PDM is required.
3. To require at least 3 channels to be active in all modules, use a PDM module in slot 2 and set the [PDM control pattern] to 0x0013 for the module in slot 3. Make sure the [Module writes control pattern ...] box is checked for this module and the [Send local hit pattern to PDM] box is checked for all modules. Then check the "global test" box for all modules

8 Using Pixie-4 Modules with Clover detectors

When working with clover detectors, the Pixie-4 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 16K. The clover mode applies only to MCA runs, not list mode runs. The clover mode is enabled by setting the corresponding checkbox in the Pixie Viewer’s Module Control Register panel. There is also the option of binning only those events in the individual channel spectra that do not have multiple hits.

Additional clover functions are under development.

9 Troubleshooting

9.1 Startup Problems

The following describes solutions to common startup problems.

1. **Computer does not boot when Pixie module is installed in chassis**
This is usually caused by an incorrect clock setting on the Pixie module. The module needs to have a valid clock to respond to the computer's scanning of the PCI bus.
2. **Computer reports new hardware found, needs driver files**
Whenever a Pixie module is installed in a slot of the chassis for the first time, it is detected as new hardware, even if Pixie modules have been installed in other slots previously. Point Windows to the driver files provides with the software distribution.
3. **Drivers are installed, but module does not boot**
The Modules require driver version 3.20.0.0 or 4.1.0.0 provided by XIA, not 4.4.0.0.
4. **When starting the Pixie Viewer, IGOR reports compile error**
For IGOR to start up properly, a number of driver files have to be in the correct locations. In particular, the file pixie.xop has to be located in the "Igor Extensions" folder – usually C:\Program Files\Wavemetrics\Igor Pro\Igor Extensions in a default installation.
5. **When starting up modules in the Pixie Viewer, downloads are not "successful"**
This can have a number of reasons. Verify that
 - The files and paths point to valid locations (run the "UseHomePaths" macro)
 - The slot numbers entered in the Startup panel match the location of the modules.
6. **After starting up modules, the ADC traces show only rectangular waves.**
This problem can be caused by downloading the wrong Communication FPGA file (e.g. a Rev. B file to a Rev. C module). Verify the file names are correct.

10 Appendix A

This section contains hardware-related information.

10.1 Jumpers

Table 9.1: Analog conditioning selection jumpers on Pixie-4 modules. x=1..4 for channel 0..3.

JPx01	Remove only if you require attenuation. Attenuation will be 1:7.5 if JPx02 is set.
JPx02	Set for input impedance of 50 Ω . If not set, input impedance is 5K Ω .
JPx05	(Revision B only) Set to “VGA” to enable contributions of the variable gain amplifier to the overall system gain for fine tuning of the gain. This might increase the signal noise and is currently not supported by software.

Table 9.2: On-board jumper settings for the clock distribution on Pixie-4 modules.

Clock mode	JP1 and JP2	JP3	PCB Reference
Single Module	Connect pins 2 and 3 of JP2	not set	LOC to IN
Daisy-Chained Clock Master	Connect pins 2 and 3 of JP2	not set	LOC to IN
Daisy-Chained Clock Repeater	Not set	set	Left
Bussed Clock Master	Connect pins 2 and 3 of JP2 Connect pin1, JP1 to pin 1, JP2	not set	LOC to IN OUT to BUS
Bussed Clock Slave	Connect pins 1 and 2 of JP2	not set	BUS to IN
Clock Slave with PXI clock	Connect pin2, JP1 to pin 2, JP2	not set	PXI to IN
Clock Master for PXI clock (Revision C only)	Connect pin2, JP1 to pin 2, JP2 Connect pin3, JP1 to pin 3, JP2	not set	PXI to IN LOC to BP

10.2 PXI backplane pin functions

Table 9.3: Pins of the J2 backplane connector defined in the PXI standard used by the Pixie-4. Pins not listed are not connected except for pull-ups to 5V recommended by the PXI standard.

J2 pin number	PXI pin name	Connection Type	Pixie pin function (revised in version 1.5.6)
1A	LBL9	Left neighbor	Event Trigger output (chained OR)
3A	LBR7	Right neighbor	reserved
16A	TRIG1	Bussed	Event Trigger
17A	TRIG2	Bussed	Veto
18A	TRIG3	Bussed	Sync
19A	LBL2	Left neighbor	Sync output (chained OR)
20A	LBR4	Right neighbor	reserved
21A	LBR0	Right neighbor	Clock output
16B	TRIG0	Bussed	Fast Trigger
18B	TRIG4	Bussed	Status
20B	LBR5	Right neighbor	reserved
1C	LBL10	Left neighbor	Fast Trigger output (chained OR)
3C	LBR8	Right neighbor	reserved
18C	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	LBL6	Left neighbor	GATE input channel 2
17D	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	LBR6	Right neighbor	reserved
16E	TRIG7	Bussed	Bussed Clock
17E	CLK10	Clock	PXI Clock
19E	LBL5	Left neighbor	GATE input channel 1
21E	LBR3	Right neighbor	reserved

10.3 Control and Status Register Bits

Table 9.4: Control and Status Register of the Pixie-4 System FPGA

0x0001	Bit 0	RunEna	Set to 1 to start data acquisition or 0 to stop. Automatically cleared when DSP de-asserts Active to end run.
0x0002	Bit 1	Unused	Reserved for future use.
0x0004	Bit 2	PCIactive	Set to reserve external memory I/O for host
0x0008	Bit 3	Unused	Reserved for future use.
0x0010	Bit 4	DSPReset	Write only. Set to reset DSP processor to initiate program download
0x0020	Bit 5	SynchCtrl	Read only. If low, module is busy with run initialization, has filled its I/O buffer with data, or is finished with the run.
0x0040	Bit 6	Unused	Reserved for future use.
0x0080	Bit 7	Unused	Reserved for future use.
0x0100	Bit 8	SynchFlag	Read only. Reserved for future use.
0x0200	Bit 9	Live*	Read only. If zero, DSP is taking data.
0x0400	Bit 10	Unused	Reserved for future use.
0x0800	Bit 11	Unused	Reserved for future use.
0x1000	Bit 12	Unused	Reserved for future use.
0x2000	Bit 13	Active	Read only. If set, there is a run in progress.
0x4000	Bit 14	LAMState	Read only. If set, LAM is set internally.
0x8000	Bit 15	Unused	Reserved for future use.