mvme - mesytec VME Data Acquisition

Release 0.9.4.1

mesytec GmbH & Co. KG <info@mesytec.com>

Jul 11, 2018

CONTENTS:

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

CHAPTER

ONE

INTRODUCTION

mvme is a VME data acquisition solution by mesytec aimed at nuclear physics experiments involving a single VME controller. The goal of this project is to provide an easy to setup, easy to use, cross-platform data acquisition system with basic data visualization and analysis capabilities.

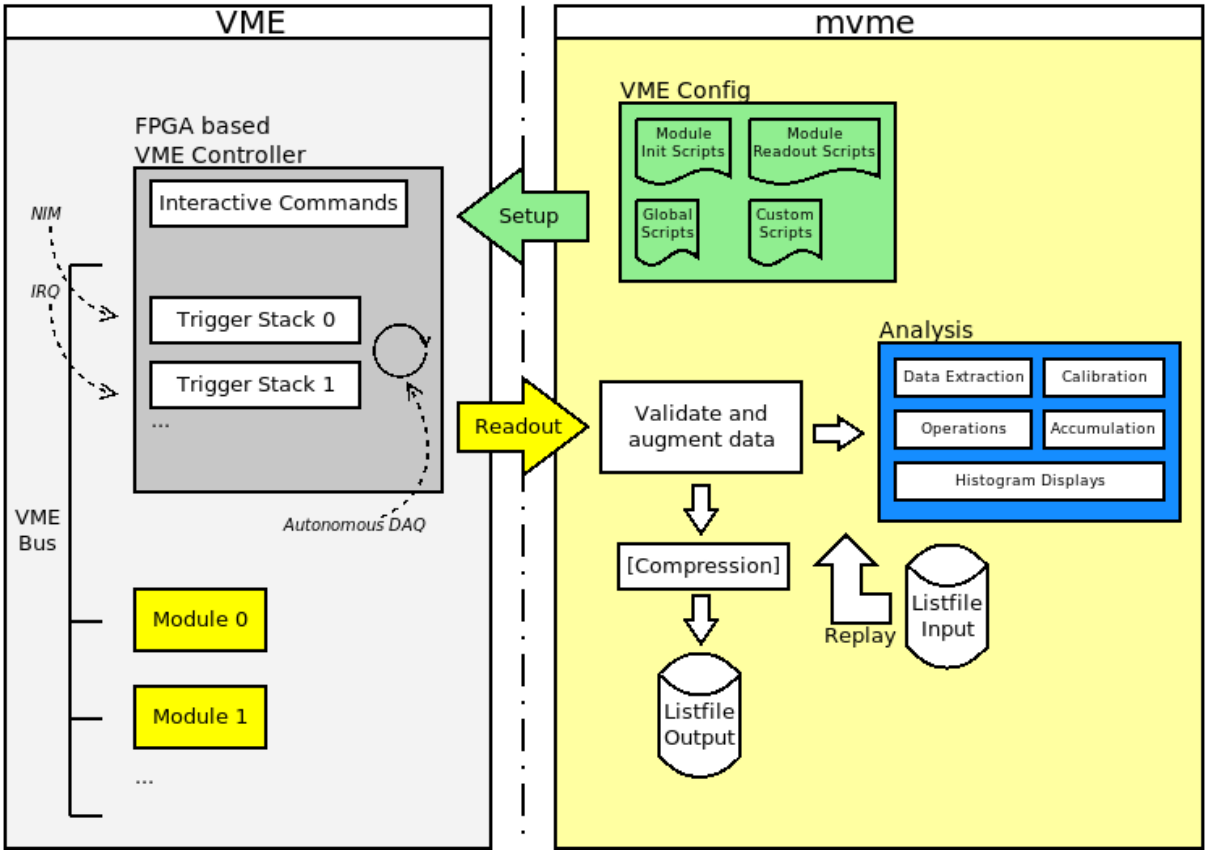


Fig. 1.1: mvme overview

1.1 Features

- High-rate, low-latency VME module readout
- Supports the WIENER VM-USB Controller:
  - Readout rates of up to 15 MB/s
  - USB2 connection
- Supports the Struck SIS3153 Controller over **GBit/s ethernet**:

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

  - Readout rates of up to 35 MB/s from a single module.
  - Optimized setups can yield around 52 MB/s.
  - GBit/s Ethernet connection required
- Easy creation and configuration of the VME setup
  - Multiple event triggers are possible (NIM, IRQ, periodic readout)
  - Multiple modules can be read out per trigger
  - Flexible VME module setup using configuration scripts
- Live histogramming of readout data (1D and 2D)
- Rate Monitoring of internal system rates and external rates generated from readout data (e.g. scaler modules, event counters).
- Flexible VME module data extraction using matching and filtering on the bit level
- Graphical analysis UI
- Optional compression of output listfiles
- Replays of recorded listfile data
- JSON-RPC remote control interface over TCP

  Allows to remotely start and stop DAQ runs and request status information.

1.2 High-level overview

1.2.1 The VME side

mvme achieves high data rate, low-latency VME readout by using the VME controllers **autonomous DAQ mode**. In this mode the controller executes lists of commands (**trigger stacks**) upon activation of a specific trigger condition. Data generated by the execution of trigger stacks is buffered and then sent over USB (VM-USB) or Ethernet (SIS3153) to the controlling computer.

In mvme the physical VME setup is described as a tree of objects with **events** as top-level nodes. Each event has a trigger condition (e.g. *Interrupt* or *NIM*) and contains the **modules** to be read out on every activation of the trigger.

A **module** in mvme has a collection of *VME Scripts*. These scripts contain the module configuration (module specific parameters, multicast setup, etc.) and the commands required to perform the readout. The readout commands of all modules belonging to the same event are combined to form the trigger stacks uploaded to the VME controller.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

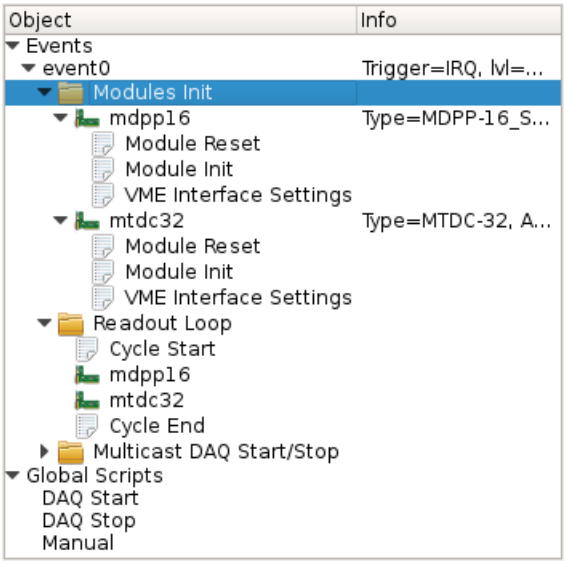


Fig. 1.2: VME setup with one event containing two modules

The VME setup also describes the structure of the data that is expected to be read out and thus allows the software to validate the received data stream.

For mesytec modules fully commented initialization files are bundled with mvme and can be loaded as templates.

1.2.2 Analysis

mvme contains an analysis system that allows parameter extraction (e.g. ADC values per channel), calibration, accumulation and visualization of data both during a DAQ run and while replaying from file. Additionally a set of built-in operators can be used to perform calculations and transformations on the data as it flows through the system.

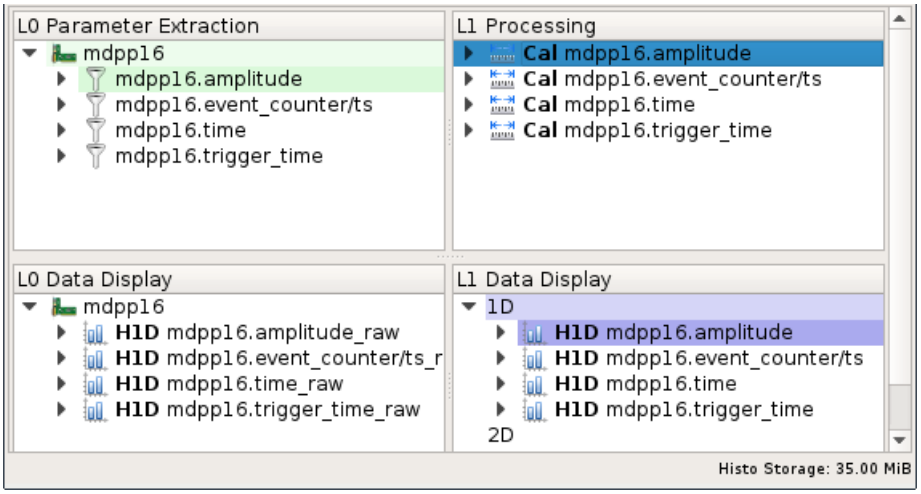


Fig. 1.3: Analysis UI with MDPP-16 default objects

The structure defined by the VME configuration is also present in the analysis: modules which are read out as a result of the same trigger condition are grouped together.

The system itself models dataflow from **sources**, through **operators**, into **sinks**. Data is transported in the form of **parameter arrays** with each element carrying the parameters numeric value and additional meta information.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

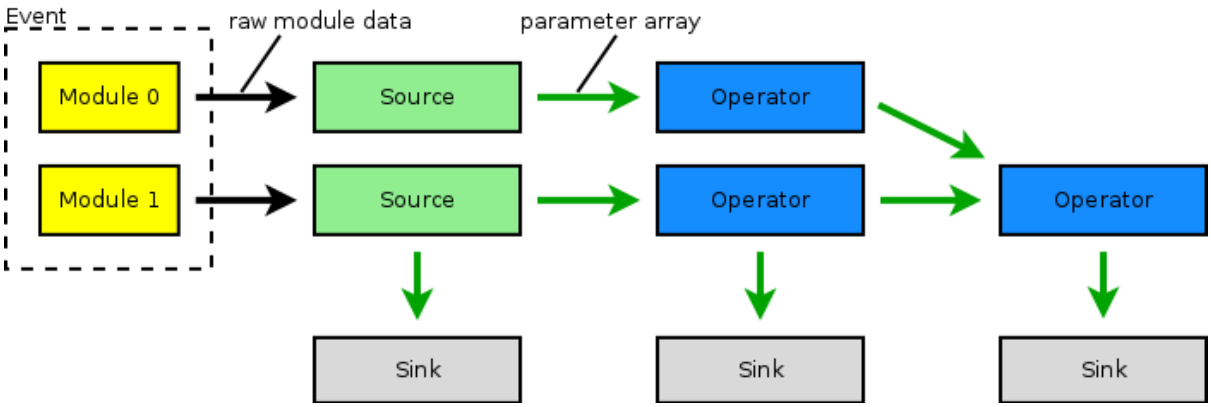


Fig. 1.4: Example analysis dataflow

Sources are data extractors that are directly attached to a VME module. A source receives each data word that was read out from the module in response to a trigger condition. Sources are used to split the data into logical parts, e.g *Amplitude* and *Time* data and to extract the corresponding raw values.

Operators are logic pieces used to perform calculations on the data (e.g. calibrate raw ADC values to voltage). Operators can have multiple inputs and produce a single output array.

Sinks are data accumulators that do not produce any output parameters. Currently 1D and 2D histogram sinks are implemented.

Output parameters of sources and operators can be inspected at runtime. Objects can be added, removed and modified even while the DAQ or a replay is running. Changes are effective immediately.

1.2.3 The DAQ process

When requested to start a data acquisition run mvme performs the following steps:

- Initialize the VME controller using information from the VME configuration
- Setup modules using the module VME scripts
- Switch the controller into DAQ mode
- Repeat until DAQ is stopped:
  - Read a data buffer from the VME controller
  - Validate the structure of the received data
  - Augment the data with mvme specific meta data
  - Write data to the listfile (optionally using compression)
  - Pass data to the *Analysis*
- Tell the controller to leave DAQ mode
- Close the listfile

Note: Data acquisition and writing the data to file have the highest priority in mvme. If during a DAQ run the analysis system cannot keep up with the incoming data rate some buffers will not be passed on to the analysis.

The number of buffers not passed on is shown as *Buffers dropped* in the statistics area at the bottom of the main window.

When replaying from file *all* buffers are passed to the analysis.

---

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

CHAPTER

TWO

INSTALLATION

2.1 System Requirements

- Any recent Linux distribution or a version of Windows 7 or later.

  Both 64- and 32-bit systems are supported but the 64-bit version is recommended due to the larger address space.

- If using the WIENER VM-USB VME Controller:
  - WIENER VM-USB VME Controller with a recent firmware

    The VM-USB firmware can be updated from within mvme. See *VM-USB Firmware Update* for a guide.
  - Latest USB chipset driver for your system.

    Updating the driver is especially important for Windows versions prior to Windows 10 in combination with a NEC/Renesas chipset (frequently found in laptops). The driver shipped by Microsoft has a bug that prevents libusb from properly accessing devices. See the libusb wiki for more information.
  - USB Driver: libusb-0.1 (Linux) / libusb-win32 (Windows)

    The windows installer can optionally run Zadig to handle the driver installation.
- No additional drivers are required when using the Struck SIS3153 Controller. Just make sure you're using a **GBit/s** ethernet connection to the controller.
- At least 4 GB RAM is recommended.
- A multicore processor is recommended as mvme itself can make use of multiple cores: readout, analysis and GUI (which includes histogram rendering) run in separate threads.

2.2 Linux

The mvme archives for Linux include all required libraries. The only external dependency is the GNU C Library glibc. When using a modern Linux distribution no glibc version errors should occur.

Installation is simple: unpack the supplied archive and execute the *mvme* startup script:

```
$ tar xf mvme-x64-1.0.tar.bz2
$ ./mvme-x64-1.0/mvme.sh
```

2.2.1 VM-USB Device Permissions

To be able to use the VM-USB controller as a non-root user a udev rule to adjust the device permissions needs to be added to the system.

Create a file called `/etc/udev/rules.d/999-wiener-vm_usb.rules` with the following contents:

```
# WIENER VM_USB
SUBSYSTEM=="usb", ATTRS{idVendor}=="16dc", ATTRS{idProduct}=="000b", MODE="0666"
```

This will make the VM-USB usable by *any* user of the system. A more secure version would be:

```
# WIENER VM_USB
SUBSYSTEM=="usb", ATTRS{idVendor}=="16dc", ATTRS{idProduct}=="000b", MODE="0660",␣
↪GROUP="usb"
```

which requires the user to be a member of the *usb* group.

Reload udev using `service udev reload` or `/etc/init.d/udev reload` or `service systemd-udev reload` depending on your distribution or simply reboot the machine.

2.3 Windows

Run the supplied installer and follow the on screen instructions to install mvme.

At the end of the installation process you are given the option to run Zadig to install the driver required for VM-USB support to work. Refer to the description text in the installer and *VM-USB Driver Installation* for details.

2.3.1 VM-USB Driver Installation

To be able to use the VM-USB VME Controller the *libusb-win32* driver needs to be installed and registered with the device. An easy way to install the driver is to use the Zadig USB Driver Installer which comes bundled with mvme. You can run Zadig at the end of the installation process or at a later time from the mvme installation directory.

In the Zadig UI the VM-USB will appear as *VM-USB VME CRATE CONTROLLER*. If it does not show up there's either a hardware issue or another driver is already registered to handle the VM-USB. Use *Options -> List All Devices* to get a list of all USB devices and look for the controller again.

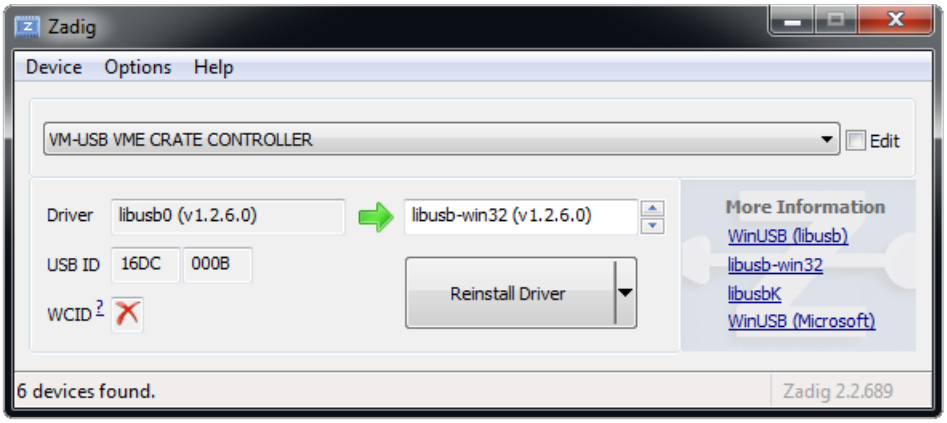


Fig. 2.1: Zadig with VM-USB and libusb-win32 selected

Make sure *libusb-win32* is selected as the driver to install, then click on *Install Driver*. Zadig will generate a self-signed certificate for the driver and start the installation process.

It is highly recommended to restart your system after driver installation, especially if you replaced an existing driver. Otherwise USB transfer errors can occur during VME data acquisition!

In case you want to manually install the driver a ZIP archive can be found here: libusb-win32.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

2.4 SIS3153 Hostname/IP-Address configuration

2.4.1 Using DHCP

On powerup the SIS3153 tries to get an IP address and a hostname via DHCP. The requested hostname is of the form `sis3153-0DDD` where `DDD` is the decimal serial number as printed on the board. For example my controller with S/N 042 will ask for the hostname `sis3153-0042`. During this phase the L-LED will flash quickly and turn off once the DHCP assignment succeeded.

2.4.2 Using a static ARP entry

In case DHCP with hostname assignment should not or cannot be used an alternative approach is to manually associate the MAC-address of the controller with an IP-address.

The MAC-address of the SIS3153 is `00:00:56:15:3x:xx` where `x:xx` is the serial number in hexadecimal. So for my development controller with S/N 42 the serial becomes `0x2a` and the resulting MAC-address is `00:00:56:15:30:2a`.

- Creating the ARP entry under linux:

  With root permissions an ARP entry can be addded this way:

  ```
  # arp -s 192.168.100.42 00:00:56:15:30:2a
  ```

  To make the entry permanent (at least on debian and ubuntu systems) the file /etc/ethers can be used. Add a line like this to the file:

  ```
  00:00:56:15:30:2a 192.168.100.42
  ```

  This will take effect on the next reboot (or when restarting the networking services I think).

- Creating the ARP entry under windows:

  Open a `cmd.exe` prompt with **administrator** permissions and use the following command to create the ARP entry:

  ```
  arp -s 192.168.100.42 00-00-56-15-30-2a
  ```

To verify that the connection is working you can ping the controller. It will send out ICMP replies and for each received packet the L-LED will flash briefly.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

CHAPTER

THREE

QUICKSTART GUIDE

The quickstart guide explains how to create a simple setup using the WIENER VM-USB VME controller and one mesytec VME module. The modules internal pulser is used to generate test data. Data readout is triggered by the module using IRQ1.

Note: In this example an MDPP-16 with the SCP firmware is used but any **mesytec** VME module should work. For other modules the value written to the pulser register (0x6070) might need to be adjusted. Refer to the modules manual and the VME templates for details.

- Start mvme and create a new workspace directory using the file dialog that should open up. This directory will hold all configuration files, recorded listfiles, exported plots, etc.
- Three windows will open:
  - A main window containing DAQ controls, the VME configuration tree and a statistics area.
  - The analysis window. As there are no VME events and modules defined yet the window will be empty.
  - A log view where runtime messages will appear.

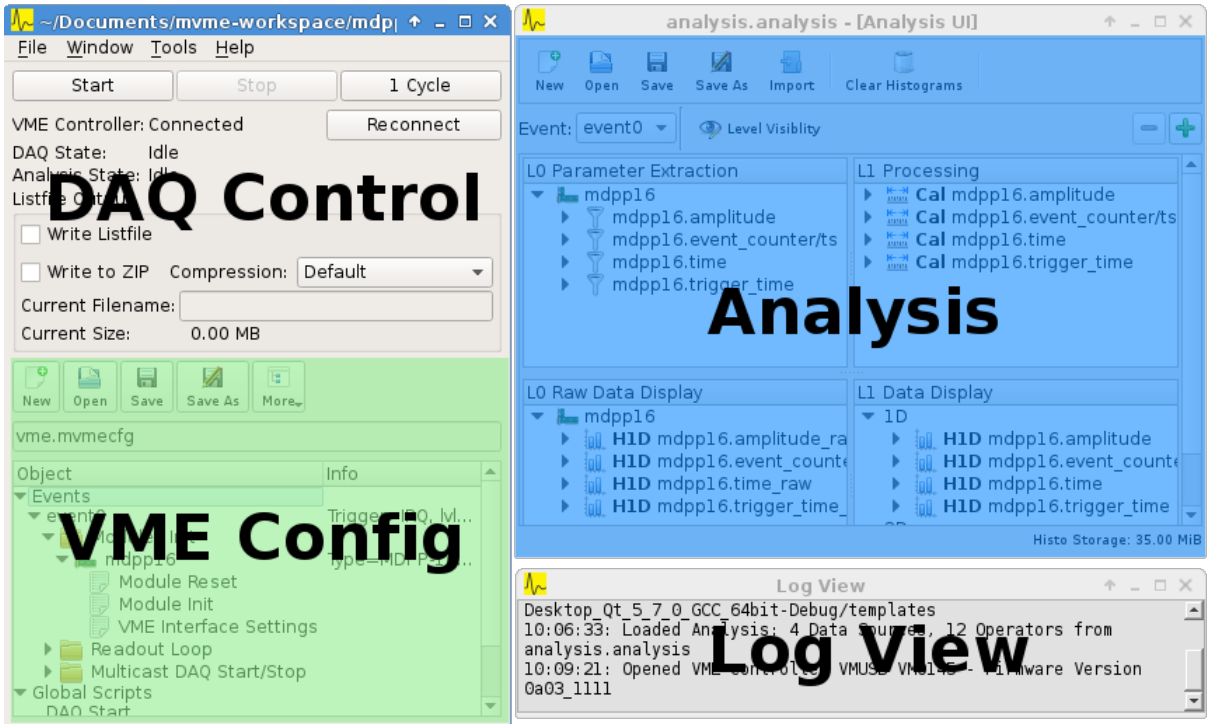


Fig. 3.1: GUI overview

If a VM-USB VME controller is connected to the PC and powered on mvme should automatically find and use it.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

VME Controller in the DAQ Control area should show up as *Connected*. Also the VM-USB firmware version will be printed to the Log View.

3.1 VME Setup

- Select the mvme main window containing the *VME Config* area.
- Create a VME event:
  - Right-click the *Events* entry in the VME tree and select *Add Event*.
  - Select *Interrupt* in the *Condition* combobox. Keep the defaults of *IRQ Level = 1* and *IRQ Vector = 0*.

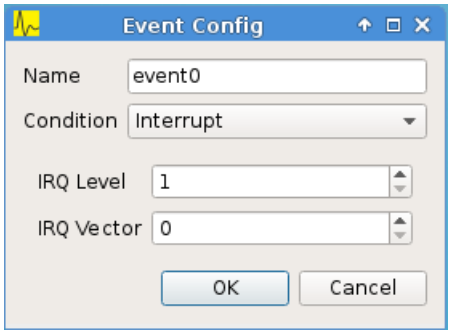


Fig. 3.2: Event Config Dialog

- Create a VME module:
  - Right-click the newly created event (called “event0” by default) and select *Add Module*.
  - Select *MDPP-16_SCP* from the module type list. If you changed the modules address encoders adjust the *Address* value accordingly (the address encoders modify the 4 most significant hex digits).

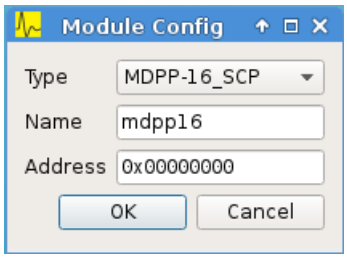


Fig. 3.3: Module Config Dialog

The VME GUI should now look like shown in *VME Config Tree*.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

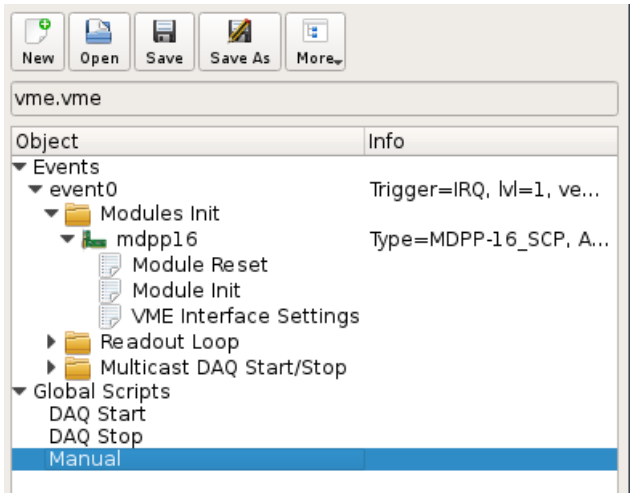


Fig. 3.4: VME Config Tree

- Double-click the *Module Init* entry to open a VME Script Editor window. Scroll to the bottom of the editor window and adjust the register value for the modules internal pulser:

  ```
  0x6070 3
  ```

  This line tells mvme to write the value `3` to register address `0x6070`. The address is relative to the module base address.

- Click the *Apply* button on the editors toolbar to commit your changes to the VME configuration. Close the editor window.

- Start editing the *VME Interface Settings* VME Script. Near the top of the script set the *irq level* to 1:

  ```
  0x6010 1
  ```

  This makes the module send IRQ 1 if it has data to be read in its internal buffer. The other parameters can be left at their default values. Click *Apply* and close the editor window.

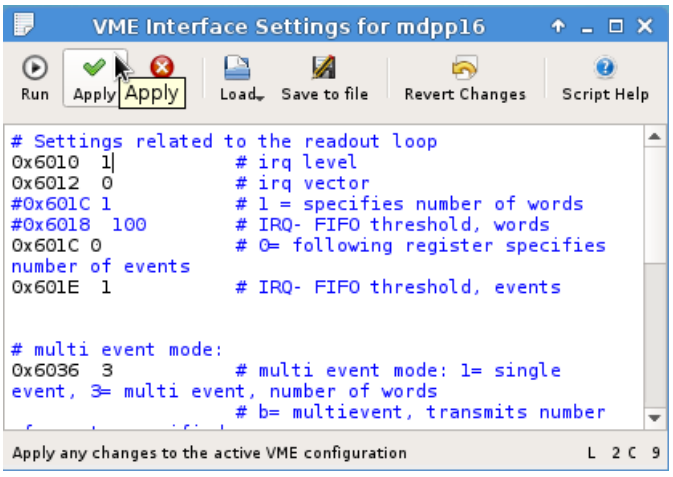


Fig. 3.5: VME Interface Settings with IRQ Level set to 1

3.2 Analysis Setup

- Activate the *Analysis UI* window (the shortcut is `Ctrl+2`). The event containing the module just created should be visible in the UI.

- Right-click the module and select *Generate default filters*. Choose *Yes* in the messagebox that pops up. This will generate a set of data extraction filters, calibration operators and histograms for the module.

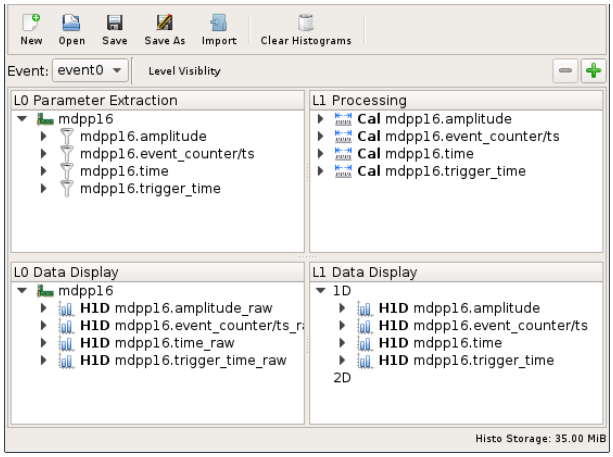


Fig. 3.6: Analysis UI with MDPP-16 default objects

3.3 Starting the DAQ

Activate the main window again (`Ctrl+1`). Make sure the *VME Controller* is shown as *Connected* in the top part of the window.

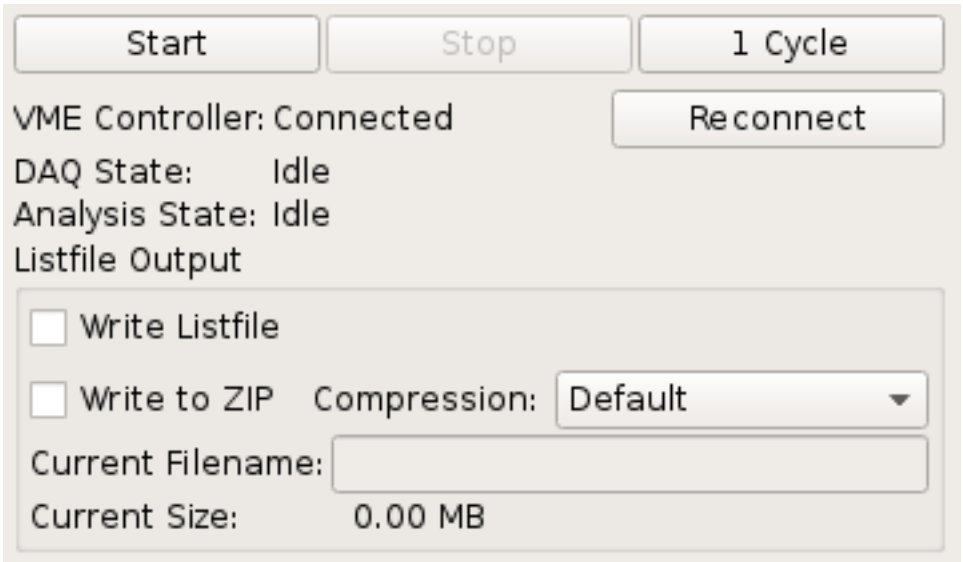


Fig. 3.7: DAQ control

Optionally uncheck the box titled *Write Listfile* to avoid writing the test data to disk. If the option is set the raw data will be written to a *.mvmelst* file inside the *listfile* subdirectory of the workspace. For each run a new filename based on the current timestamp is generated. If writing a ZIP archive both the current analysis and the text log file produced during the run will be added to the resulting archive.

Press the *Start* button to start the DAQ. Check the *Log View* (`Ctrl+3`) for warnings and errors.

In the *Analysis UI* double-click the histogram entry called *amplitude_raw* (bottom-left corner in the *L0 Data Display* tree) to open a histogram window.

If data acquisition and data extraction are working properly you should see new data appear in the histogram. Use the spinbox at the top right to cycle through the individual channels.

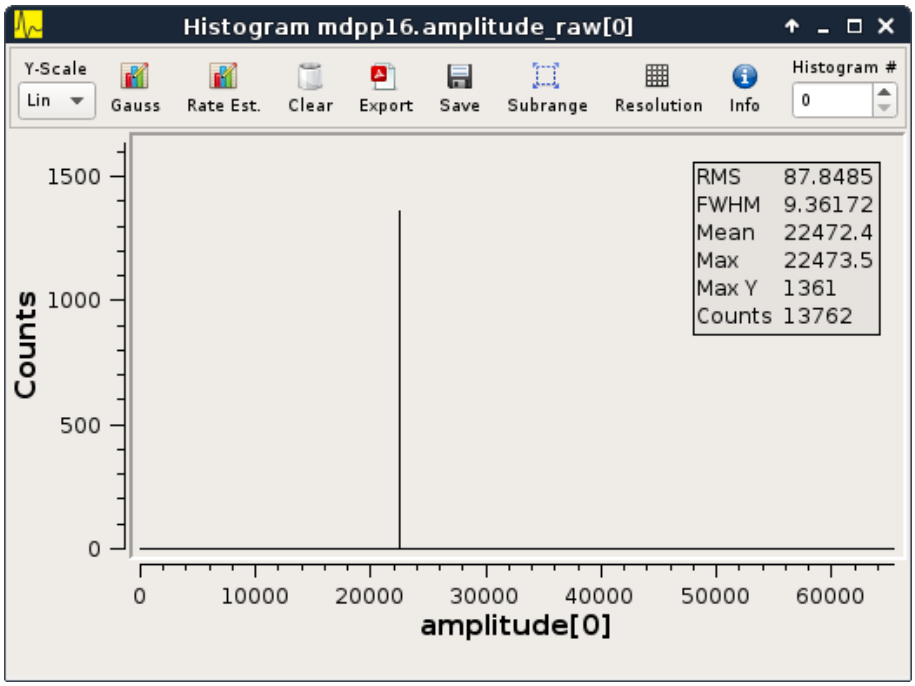


Fig. 3.8: Amplitude histogram

You can pause and/or stop the DAQ at any time using the corresponding buttons at the top of the main window.

3.4 Event Counter readout

Optionally a second event used to read out the modules event counter registers can be created. This event will be triggered periodically by the VME controller.

- Right-click *Events*, choose *Add Event*
- Set *Condition* to *Periodic* and the period to `1.0s`
- Right-click the newly created event, choose *Add Module*
- Select *MesytecCounter* as the module type
- Enter the same address as used for the MDPP-16 above

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

CHAPTER

FOUR

REFERENCE

4.1 DAQ / Replay controls

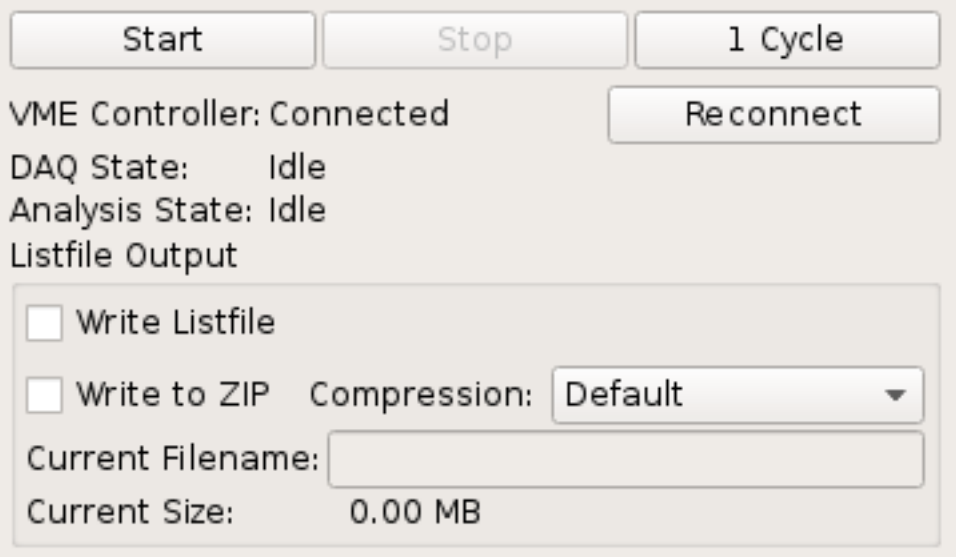


Fig. 4.1: DAQ controls

The effects of the buttons depend on the current mode - DAQ or replay - and the current state of the system:

Table 4.1: DAQ control actions

Action	DAQ mode	Replay mode
Start	• Run *DAQ startup procedure* • Open new listfile	Start replay from beginning of file
Stop	• Run *DAQ stop procedure* • Close listfile	Stop and rewind to beginning of file
Pause	• Leave DAQ mode • No special procedures are run	Pause replay
1 Cycle / Next Event	• Start DAQ for one cycle • Dump received data to log view	• Replay the next event from the file • Dump event data to log view

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

The *Start* and *1 Cycle / Next Event* buttons allow to choose what should happen with existing histogram data. Selecting *Clear* will clear all histograms in the current analysis before accumulating new data. Using *Keep* allows to accumulate the data from multiple replays or DAQ runs into the same histograms.

The *Reconnect* button will attempt to reconnect to the current VME controller. This is needed when using the VM-USB controller and power-cycling the VME crate as USB disconnects are currently not detected.

If *Write Listfile* is checked a new output listfile will be created when starting a DAQ run. The file will be created in the *listfiles* subdirectory of the current workspace. The filename is based off the current timestamp to make it unique.

If *Write to ZIP* is checked a ZIP archive with optional compression will be created instead of a flat *.mvmelst* file.

4.1.1 Replaying from listfile

To replay data from a listfile use *File -> Open listfile* and choose a *.zip* or *.mvmelst* file.

When opening a listfile the VME config included inside the file is loaded and will replace the current config. The global mode will be switched to *Listfile*. To go back to DAQ mode use *File -> Close Listfile*.

4.2 VME configuration

4.2.1 Structure

The VME configuration in mvme models the logical VME setup. *Modules* that should be read out as a result of the same trigger condition are grouped together in an *Event*:

```
Event0
    Module0.0
    Module0.1
    ...

Event1
    Module1.0
    Module1.1
    ...
```

The type of available trigger conditions depends on the VME controller in use. With the WIENER VM-USB the following triggers are available:

- NIM

  One external NIM input

- IRQ1-7

  The standard VME interrupts

- Periodic

  VM-USB supports one periodic trigger which is executed every `n * 0.5s` (*Period*) or on every `m-th` data event (*Frequency*). If both values are set both internal counters are reset on each activation of the trigger. Refer to section 3.4.3 of the VM-USB manual for details.

4.2.2 Module and Event configuration

The module and event configuration is done using *VME scripts* which contain the commands necessary to initialize and readout each module.

At the module level the following phases are defined:

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

- Reset

  Reset the module to a clean default state.

- Init

  Setup the module by writing specific registers.

- Readout

  The code needed to readout the module whenever the trigger condition fires.

The event level distinguishes between the following phases:

- Readout Cycle Start / End

  Inserted before / after the readout commands of the modules belonging to this event. The *Cycle Start* script is currently empty by default, the *Cycle End* script notifies the modules that readout has been performed. By default this is done by writing to the multicast address used for the event.

- DAQ Start / Stop

  Executed at the beginning / end of the a DAQ run. The purpose of the *DAQ Start* script is to reset module counters and tell each module to start data acquisition. *DAQ End* is used to tell the modules stop data acquisition. By default both scripts again use the multicast address of the corresponding event.

4.2.3 DAQ startup procedure

- Reset and setup the VME controller
- Assemble readout code from configured Events

  For each Event do:

  - Add *Cycle Start* script
  - For each Module:
    - Add Module readout script
    - Add “Write EndMarker” command
  - Add *Cycle End* script
- Upload the readout code to the controller and activate triggers
- Execute global *DAQ Start* scripts
- Initialize Modules

  For each Event do:

  - For each Module do:
    - Run *Module Reset*
    - Run all *Module Init* scripts
  - Run the Events *Multicast DAQ Start* script
- Set the controller to autonomous DAQ mode

Control is handed to the VME controller. mvme is now reading and interpreting data returned from the controller.

4.2.4 DAQ stop procedure

- Tell the VME controller to leave autonomous DAQ mode
- Read leftover data from the VME controller
- Run the *DAQ Stop* script for each Event

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

- Execute global *DAQ Stop* scripts

4.3 VME Scripts

4.3.1 Overview

VME Scripts are plain text files with one command per line. Comments may be started using the # character. They extend to the end of the line.

Scripts belonging to a module (**Module Init**, **VME Interface Settings**, **Module Reset** and the readout code) will have the **module base address** added to most of the commands. This allows writing scripts containing module-relative addresses only. An exception is the *writeabs* command which does not modify its address argument. The base address can also be temporarily replaced with a different value by using the *setbase* and *resetbase* commands.

The commands below use the following values for address modifiers and data widths:

Table 4.2: VME Address Modes

Address Mode (*<amode>*)
a16
a24
a32

Table 4.3: VME Data Widths

Data Width (*<dwidth>*)
d16
d32

The combination of amode, dwidth and BLT/MBLT yields a VME address modifier to be sent over the bus. Internally these non-privileged (aka user) address modifiers will be used:

Table 4.4: VME address modifiers used by mvme

amode	single	BLT	MBLT
A16	0x29		
A24	0x39	0x3b	
A32	0x09	0x0b	0x08

Numbers in the script (addresses, transfer counts, masks) may be specified in decimal, octal or hex using the standard C prefixes (`0x` for hex, `0` for octal). Additionally register values may be written in binary starting with a prefix of `0b` followed by `0`s and `1`s, optionally separated by `'` characters.

Example: `0b1010'0101'1100'0011` is equal to `0xa5c3`

4.3.2 Commands

4.3.2.1 Writing

- **write** <*amode*> <*dwidth*> <*address*> <*value*>
- **writeabs** <*amode*> <*dwidth*> <*address*> <*value*>

writeabs uses the given <*address*> unmodified, meaning the module base address will not be added.

There is a short syntax version of the write command: if a line consists of only two numbers separated by whitespace, a write using 32-bit addressing (a32) and 16-bit register width (d16) is assumed. The address is the first number, the value to be written is the second number.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Example: `0x6070 3` is the same as `write a32 d16 0x6070 3`

4.3.2.2 Reading

- **read** <*amode*> <*dwidth*> <*address*>

Reads a single value from the given <*address*>.

4.3.2.3 Block Transfers

mvme supports the following read-only block transfer commands:

- **blt** <*amode*> <*address*> <*count*>
- **bltfifo** <*amode*> <*address*> <*count*>
- **mblt** <*amode*> <*address*> <*count*>
- **mbltfifo** <*amode*> <*address*> <*count*>

blt and **bltfifo** transfer <*count*> number of 32-bit words, **mblt** and **mbltfifo** transfer 64-bit words.

The ***fifo** variants do not increment the given starting address.

4.3.2.4 Variable sized Block Transfers

- **bltcount** <*reg_amode*> <*reg_dwidth*> <*reg_addr*> <*reg_count_mask*> <*block_amode*> <*block_addr*>
- **bltfifocount** <*reg_amode*> <*reg_dwidth*> <*reg_addr*> <*reg_count_mask*> <*block_amode*> <*block_addr*>
- **mbltcount** <*reg_amode*> <*reg_dwidth*> <*reg_addr*> <*reg_count_mask*> <*block_amode*> <*block_addr*>
- **mbltfifocount** <*reg_amode*> <*reg_dwidth*> <*reg_addr*> <*reg_count_mask*> <*block_amode*> <*block_addr*>

These commands read the number of transfers to perform from the register at <*reg_addr*>, using the given <*reg_amode*> and <*reg_dwidth*> as access modifiers. The value read is then AND'ed with <*reg_count_mask*>. The resulting value is the number of block transfers to perform, starting at <*block_addr*>.

4.3.2.5 Miscellaneous

- **wait** <*waitspec*>

Delays script execution for the given amount of time. <*waitspec*> is a number followed by one of `ns`, `ms` or `s` for nanoseconds, milliseconds and seconds respectively. If no suffix is given milliseconds are assumed.

Note: When creating a command stack to be executed by the VMUSB Controller in DAQ Mode the resolution of the waitspec is **200 ns** and the maximum value is **51000 ns**.

Example: `wait 500ms # Delay script execution for 500ms`

- **marker** <*marker_word*>

The marker command adds a 32-bit marker word into the data stream. This can be used to separate data from different modules.

- **setbase** <*address*>
- **resetbase**

These commands can be used to temporarily replace the current base address with a different value. **setbase** sets a new base address, which will be effective for all following commands. Use **resetbase** to restore the original base address.

4.3.3 Example

```
# BLT readout until BERR or number of transfers reached
bltfifo a32 0x0000 10000

# Write the value 3 to address 0x6070. If this appears in a module specific
# script (init, readout, reset) the module base address is added to the
# given address.
0x6070 3

# Same as above but explicitly using the write command.
write a32 d16 0x6070 3

# Set a different base address. This will replace the current base address
# until resetbase is used.
setbase 0xbb000000

# Results in an a32/d16 write to 0xbb006070.
0x6070 5

# Restore the original base address.
resetbase

# Binary notation for the register value.
0x6070 0b0000'0101
```

4.4 Analysis

4.4.1 User Guide

4.4.1.1 UI Overview

The Analysis system in mvme is designed to allow

- flexible parameter extraction from raw readout data.
- calibration and additional processing of extracted parameters.
- accumulation and visualization of processed data.

The user interface follows the structure of data flow in the system: the modules for the selected *Event* are shown in the top-left tree.

The bottom-left tree contains the *raw histograms* used to accumulate the unmodified data extracted from the modules.

The next column (*Level 1*) contains calibration operators in the top view and calibrated histograms in the bottom view.

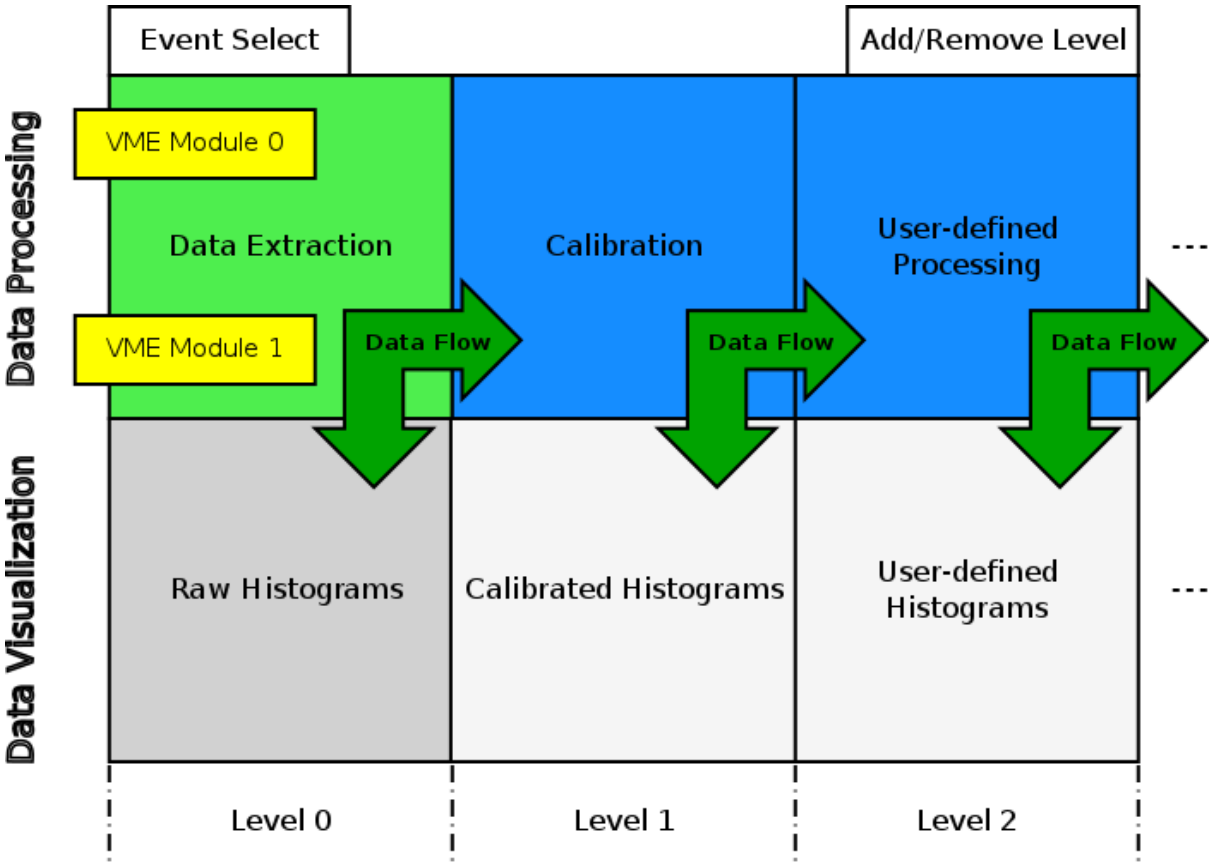


Fig. 4.2: Analysis UI Block Diagram

Note: To get a set of basic data extraction filters, calibration operators and histograms right-click on a module an select *Generate default filters*.

User Levels are used to structure the analysis and it's completely optional to have more than two of them. Additional levels can be added/removed using the + and - buttons at the top-right. Use the *Level Visibility* button to select which levels are shown/hidden.

The UI enforces the rule that operators can use inputs from levels less-than or equal to their own level. This means data should always flow from left to right.

Operators can be moved between levels by dragging and dropping them.

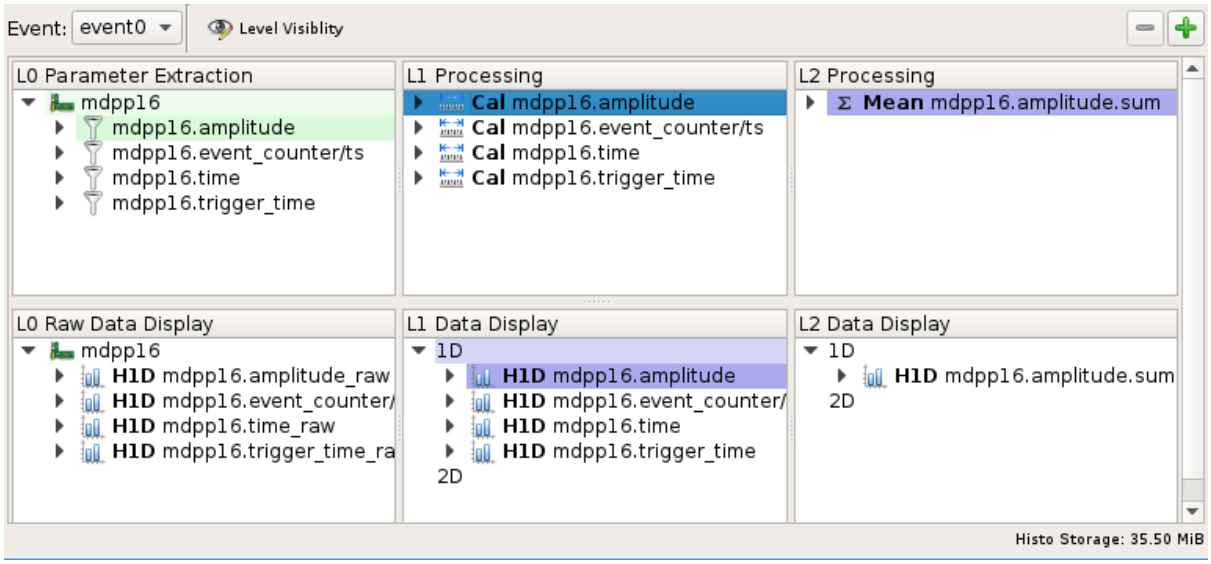


Fig. 4.3: Analysis UI Screenshot
The Calibration for *mdpp16.amplitude* is selected. Its input is shown in green. Operators using the calibrated data are shown in a light blueish color.

Selecting an object will highlight its input data sources in green and any operators using its output in blue.

4.4.1.2 Adding new objects

Right-click in any of the views and select *New* to add new operators and histograms. A dialog will pop up with input fields for operator specific settings and buttons to select the operators inputs.

Clicking any of the input buttons will make the user interface enter “input select mode”. In this mode valid outputs for the selected input are highlighted.

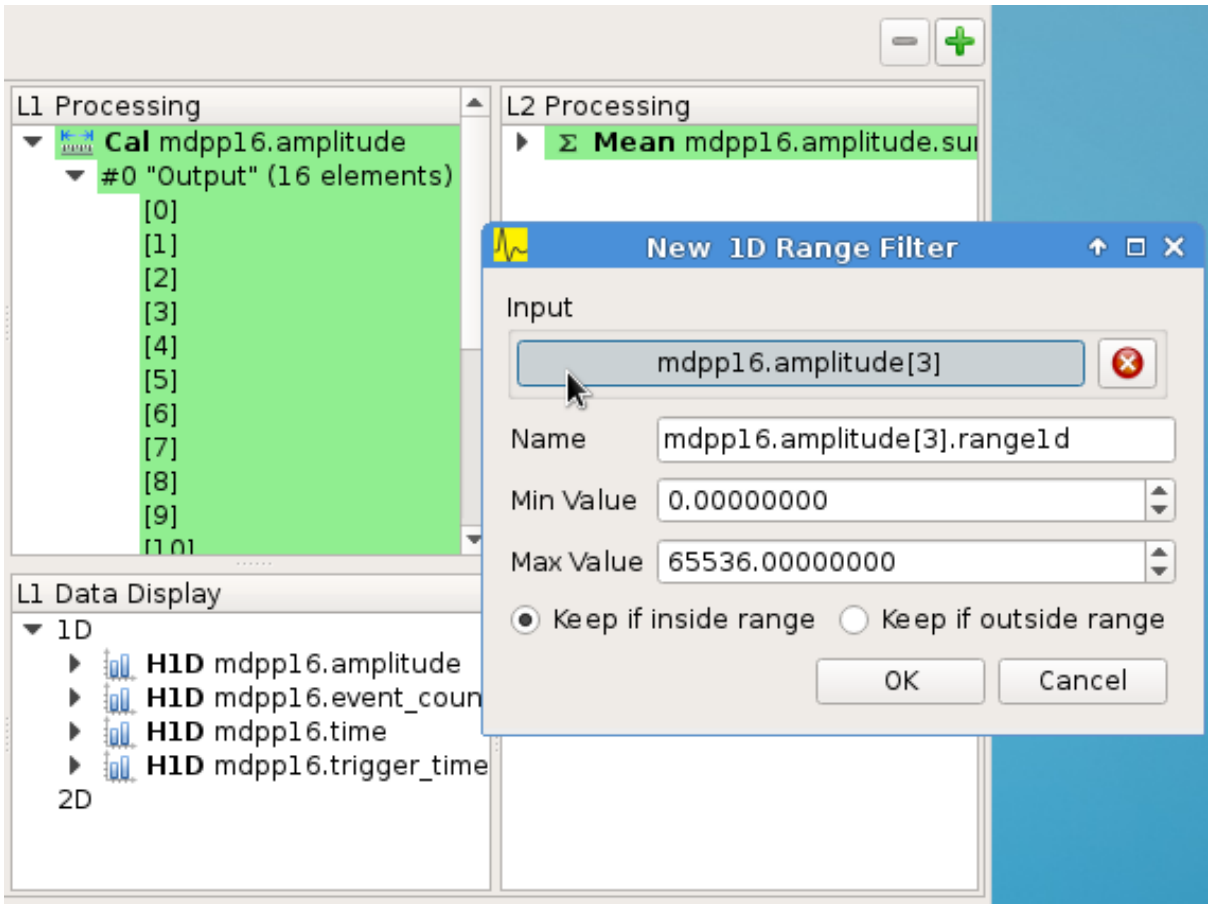


Fig. 4.4: Input select mode
Adding a *1D Range Filter* to Level 2. Valid inputs are highlighted in green.

Click an input node to use it for the new operator. If required fill in any additional operator specific data and accept the dialog. The operator will be added to the system and will immediately start processing data if a DAQ run or replay is active.

For details about data extraction refer to *Data Sources*. Descriptions of available operators can be found in *Operators*. For details about 1D and 2D histograms check the *Data Sinks* section.

4.4.1.3 Working with histograms

1D and 2D histograms are shown in the bottom row of the user interface. Raw 1D histograms are grouped by module in the bottom-left *L0 Raw Data Display* area. Higher level data displays are grouped by histogram type.

New histograms can be added by right-clicking in one of the data display areas, selecting *New* and choosing the histogram type.

1D

1D histograms can take full arrays as input parameters. Internally an array of histograms of the same size as the input array will be created.

Double-click on the *H1D* node to open the histogram array widget:

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

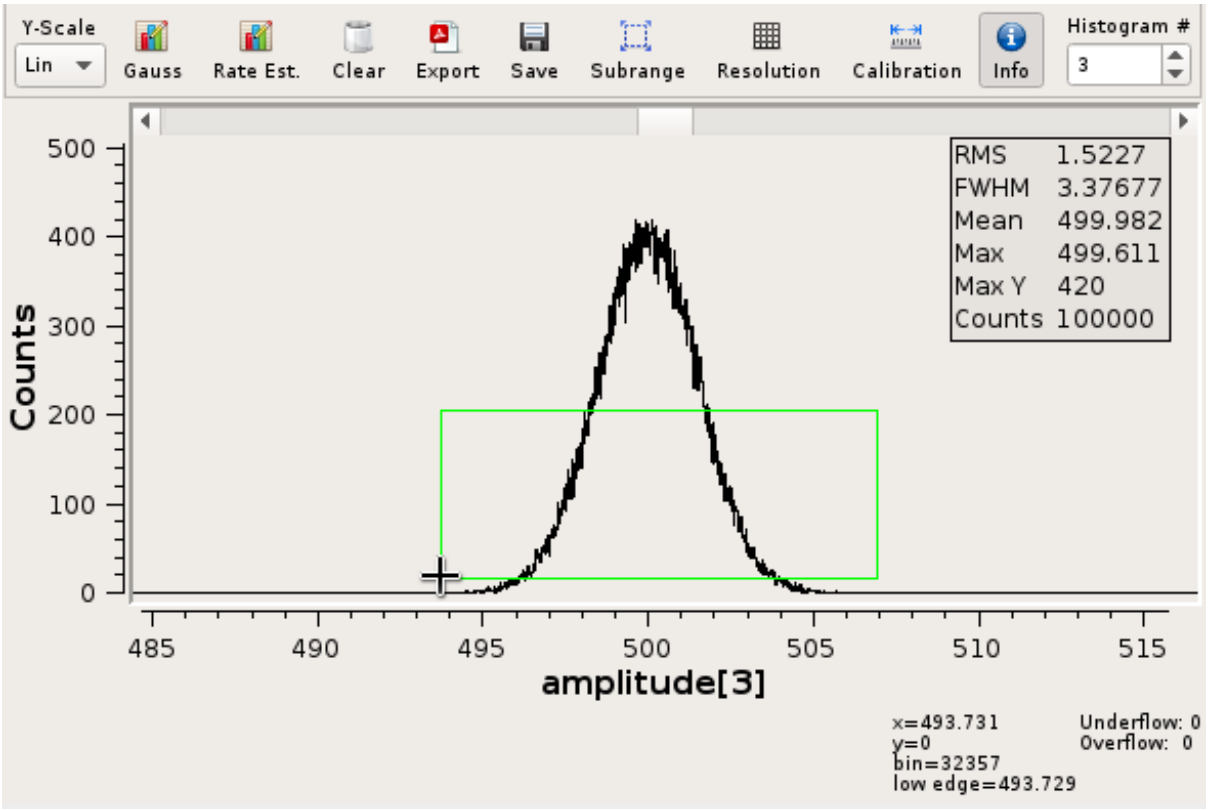


Fig. 4.5: 1D Histogram Array Widget

- The histogram index can be changed using the spinbox in the top-right corner.
- Zooming is achieved by dragging a rectangle using the left mouse button. Zoom levels are stacked. Click the right mouse button to zoom out one level.
- Press the *Info* button to enable an info display at the bottom-right of the window. This will show the current cursor coordinates and the corresponding bin number.
- Y-Scale

  Toggle between linear and logarithmic scales for the Y-Axis.
- Gauss

  Fit a gauss curve through the currently visible maximum value.
- Rate Est.

  Rate Estimation feature.

  Refer to *Rate Estimation Setup* for a how-to guide.
- Clear

  Clears the current histogram.
- Export

  Allows exporting to PDF and various image formats. Use the file type selection in the file dialog to choose the export format.
- Save

  Saves the histogram data to a flat text file.
- Subrange

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Allows limiting the range of data that's accumulated. Only input values falling within the specified interval will be accumulated.

This does not affect the histogram resolution: the full range of bins is still used with the limits given by the subrange.

- Resolution

  Change the resolution of the histogram in powers of two from 1 bit to 20 bits.

  This will not rebin existing data. Instead the histogram is cleared and new data is accumulated using the newly set resolution.

- Calibration

  This button is enabled if the histograms input is a *Calibration Operator* and allows to directly modify the calibration information from within the histogram:

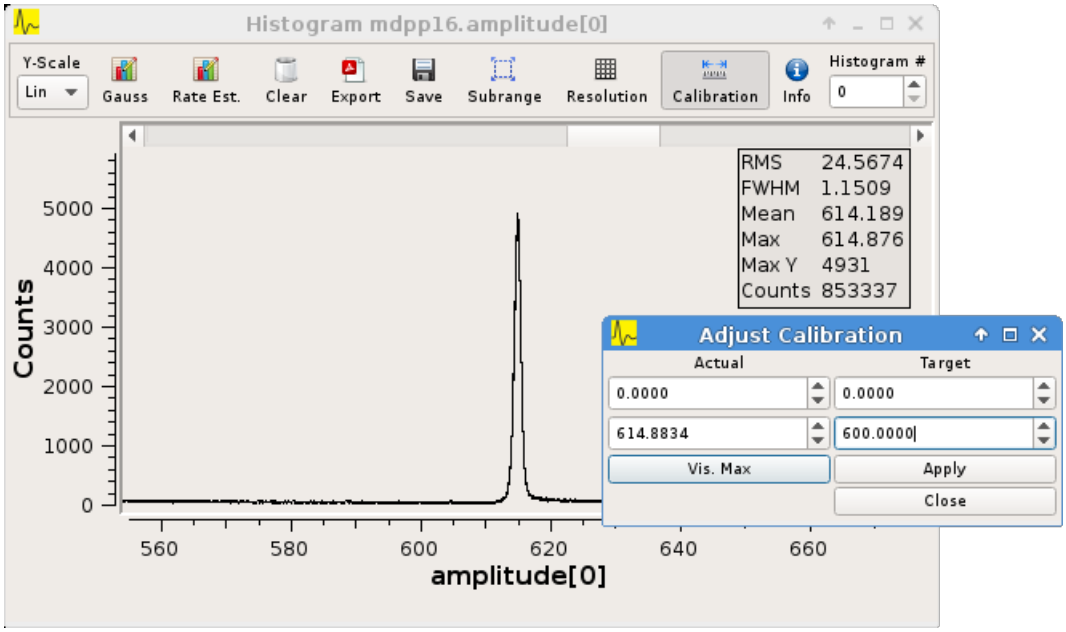


Fig. 4.6: Calibration adjustment from within the histogram display

The two inputs in the *Actual* column refer to the current x-axis scale. The inputs in the *Target* column are used to specify the desired x-axis values.

Click on one of the *Actual* inputs and then press the *Vis. Max* button to fill in the x-coordinate of the currently visible maximum value. Then enter the new x-coordinate value in the *Target* box and press *Apply*.

In the example above it is known that the peak should be at `x = 600.0`. The current x-coordinate of the peak was found using the *Vis. Max* button. Pressing *Apply* will modify the calibration for that particular histogram.

To see a list of calibration values for each channel open the Analysis UI (`Ctrl+2`), right-click the *Calibration Operator* and select *Edit*.

- 2D combined view

  A combined view of the histograms of an array of parameters can be opened by right-clicking a **H1D** node and selecting *Open 2D Combined View*. This option will open a 2D histogram with one column per 1D histogram in the array.

  The X-axes of the 1D histograms are plotted on the combined views Y-axis, the values of the histograms are plotted in Z.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

This view allows to quickly see if any or all channels of a module are responding.

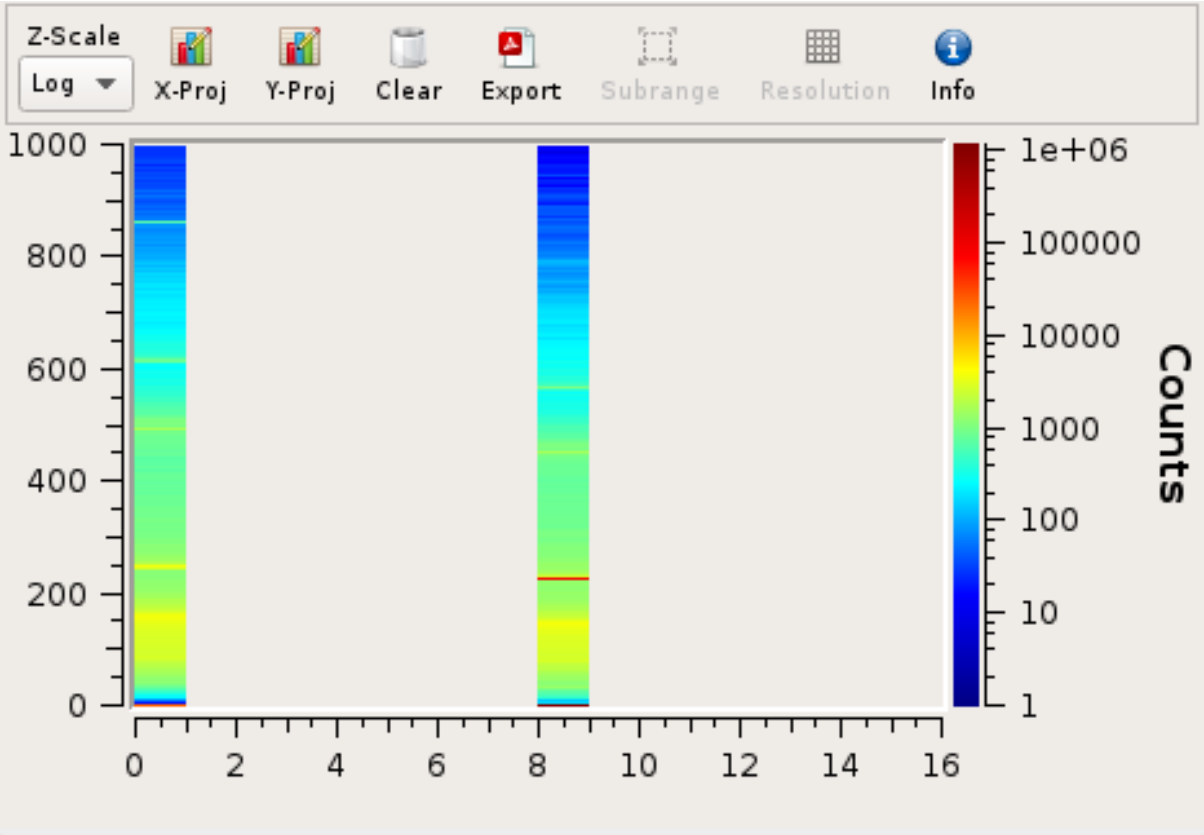


Fig. 4.7: 2D Combined View of MDPP-16_SCP amplitude values
Channels 0 and 8 are producing data with visible peaks at around 0 and 230.

2D

2D histograms take two single values as their inputs: the X and Y parameters to accumulate. When selecting the inputs you will need to expand other operators and select the desired index directly.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

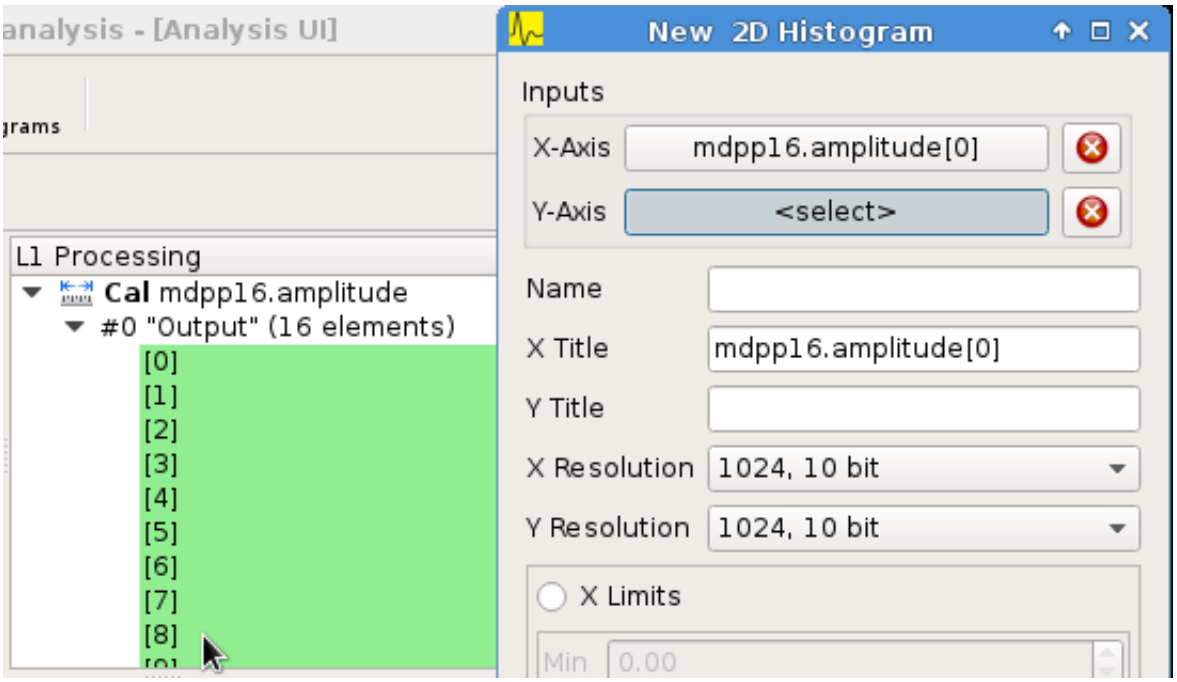


Fig. 4.8: Adding a 2D Histogram
Expand operator outputs and select individual indices for both axes.

Optional range limits can be specified for the axes. If enabled only values falling within the given interval will be accumulated.

Double-click on a *H2D* node to open the histogram widget:

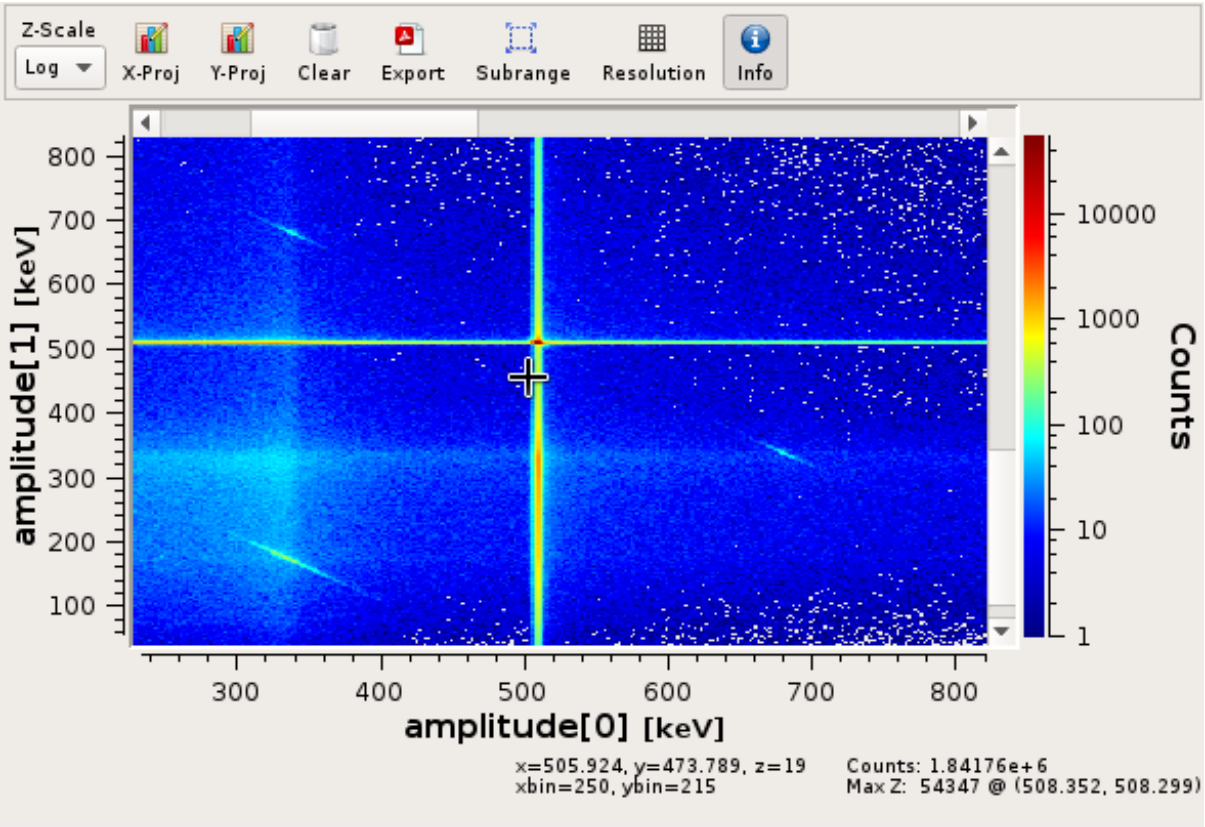


Fig. 4.9: 2D Histogram Widget

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

- Zooming is achieved by dragging a rectangle using the left mouse button. Zoom levels are stacked. Click the right mouse button to zoom out one level.
- Press the *Info* button to show histo and cursor coordinate information at the bottom of the window.
- Z-Scale

  Toggle between linear and logarithmic scales for the Z-Axis.
- X- and Y-Proj

  Create the X/Y-Projection and open it in a new 1D histogram window. The projection will follow any zooming/scrolling done in the 2D histogram.
- Clear

  Clears the histogram.
- Export

  Allows exporting to PDF and various image formats. Use the file type selection in the file dialog to choose the export format.
- Subrange

  Allows limiting the range of data that's accumulated. Only input values falling within the specified interval will be accumulated.

  This does not affect the histogram resolution: the full range of bins is still used with the limits given by the subrange.

  Can optionally create a new histogram with the specified limits instead of modifying the current one. The newly created histogram will be added to the analysis.
- Resolution

  Change the resolution of the histograms axes in powers of two from 1 bit to 13 bits.

  This will not rebin existing data. Instead the histogram is cleared and new data is accumulated using the newly set resolution.

4.4.2 System Details

As outlined in the *introduction* the analysis system is a set of interconnected objects with data flowing from *Sources* through *Operators* into *Sinks*.

The system is structured the same way as the VME Configuration: VME modules are grouped into events. An event contains the modules that are read out on activation of a certain trigger condition. The result of the readout is the modules event data (basically an array of 32-bit words). This module event data is the input to the analysis system.

When processing data from a live DAQ run or from a listfile replay the analysis system is "stepped" in terms of events: in each step all the *Data Sources* attached to a module get passed the modules event data. The task of each source is to extract relevant values from its input data and make these values available to subsequent operators and sinks.

After all sources have processed the module event data, the dependent operators and sinks are stepped in order. Each object consumes its input and generates new output or in the case of sinks accumulates incoming data into a histogram.

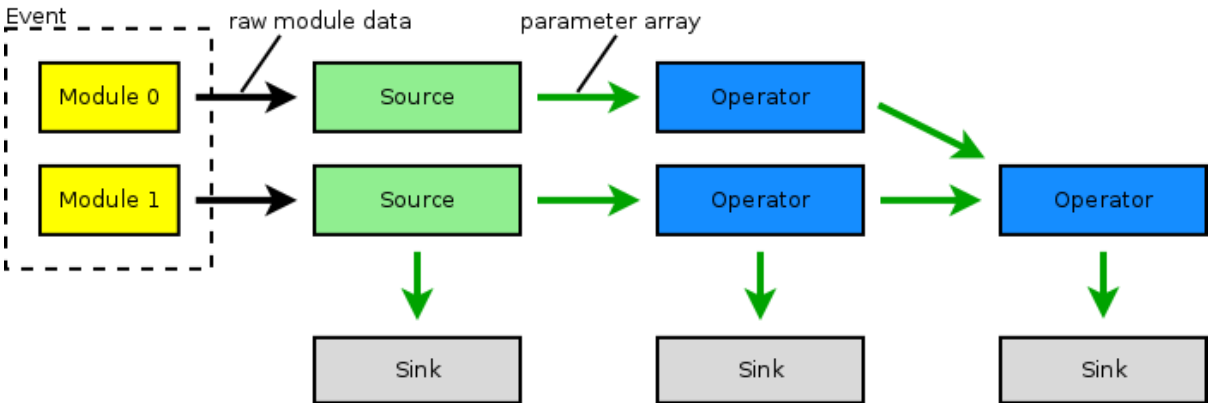


Fig. 4.10: Example analysis dataflow

4.4.2.1 Parameter Arrays

The transport container carrying data between objects is the Parameter Array:

Parameter Array			
size	unit label		
Parameters			
0	value	valid	limits
1	value	valid	limits
2	value	valid	limits
...			
size-1	value	valid	limits

The *size* of parameter arrays is determined at analysis startup time and is constant throughout the run. The *unit label* is a string which currently can be set through the use of the *Calibration Operator*. The index of a parameter in the array is usually the channel address that was extracted from the modules data.

Each parameter has the following attributes:

- *value* (double)

  The parameters data value.

- *valid* (bool)

  True if the parameter is considered valid, false otherwise.

  A parameter can become invalid if for example a data source did not extract a value for the corresponding channel address or an operator wants to explicitly filter out the address or could not calculate a valid result for the input value.

- *limits* (two doubles)

  Two double values forming the interval `[lowerLimit, upperLimit)` that the parameters value should fall into. This is used by histogram sinks and calibration operators to determine the parameters range and thus calculate the binning.

4.4.2.2 Connection types

Different operators have different requirements on their input types. The *Calibration Operator* for example can use whole parameter arrays as its input, transforms each data value and produces an output array of the same size as the input size.

Other operators can only act on individual values and thus connect directly to a specific *index* into the parameter array. An example is the *2D Histogram Sink*: it requires exactly two input values, X and Y, neither of which can be an array.

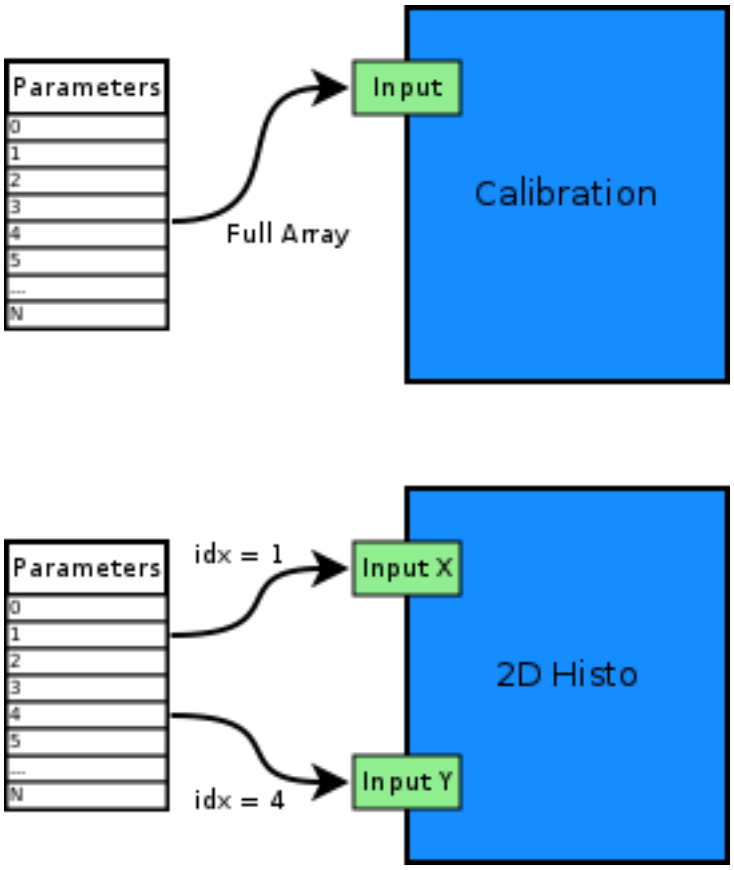


Fig. 4.11: Example of different input types

Each Operator implementation decides which types of input connections it accepts. Some operators even change the type of inputs they accept based on the first input type that is connected (they either accept full arrays for all their inputs or single values for all their inputs).

The *Analysis UI* will highlight valid input nodes in green when selecting an operators input.

4.4.3 Data Sources

Analysis Data Sources attach directly to a VME module. On every step of the analysis system they're handed all the data words produced by that module in the corresponding readout cycle. Their job is to extract data values from the raw module data and produce an output parameter array. Currently there's one Source implemented: The *Filter Extractor*

4.4.3.1 Filter Extractor

The Filter Extractor uses a list of bit-level filters to classify input words and extract address and data values.

Filter Basics

A single filter consists of 32 characters used to match a 32-bit data word. The filter describes the static parts of the data used for matching and the variable parts used for data extraction. The first (leftmost) character of a filter line matches bit 31, the last character bit 0.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

The following characters are used in filter strings:

Character	Description
`0`	bit must be cleared
`1`	bit must be set
`A`	address bit
`D`	data bit
others	don't care

The following conventions are used in the default filters that come with mvme:

- `X` is used if any bit value is allowed.
- `O` (the letter) is used to denote the position of the *overflow* bit.
- `U` is used to denote the position of the *underflow* bit.
- `P` is used to denote the position of the *pileup* bit.

These characters are merely used to make it easier to identify certain bits when editing a filter. With regards to matching any character other than `0` or `1` means that any bit value is allowed.

Example: The default *Amplitude* filter for the MDPP-16_SCP:

```
0001 XXXX PO00 AAAA DDDD DDDD DDDD DDDD
```

The filter above contains a 4-bit address and a 16-bit data value. The positions of the pileup and overflow bits are marked using `P` and `O`. This helps when adjusting the filter to e.g. match only pileup data (replace the `P` with a `1`).

The number of address bits (`A`) determine the size of the Filter Extractors output array.

Data extraction from an input data word is done by keeping only the bits matching the address or data mask and then right shifting to align with the 0 bit.

Note: Address and data bit masks do not need to be consecutive. `A0AA` will produce 3-bit address values by gathering all extracted `A` bits on the right: `0AAA`.

Each filter has an optional *word index* attached to it. If the word index is set to a value >= 0, then the filter can only produce a match on the module data word with the same index.

Multiple filter words

The Filter Extractor implementation allows combining multiple 32-bit filters to match and extract data from multiple input words.

Filters are tried in order. If a previously unmatched filter produces a match no further filters will be tried for the same data word.

Once all individual filters have been matched the whole combined filter matches and address and data values can be extracted.

When extracting values the filters are again used in order: the first filter produces the lowest bits of the combined result, the result of the next filter is left-shifted by the amount of bits in the previous filter and so on.

Note: The maximum number of bits that can be extracted for address and data values is limited to 64.

See *Timestamp extraction* for an example of how a multiword filter can be used.

Matching and data extraction

During a DAQ run or a replay the Filter Extractor gets passed all the data that was produced by a single module readout (*Event Data*). Each data word is passed to the internal filter.

Once the filter has completed *Required Completion Count* times address and data values will be extracted.

The data value is cast to a double and a uniform random value in the range `[0, 1)` is added. This resulting value is stored in the output parameter array at the index specified by the extracted address value.

User Interface

In the Analysis UI right-click a Module and select *New -> Filter Extractor* to add a new filter.

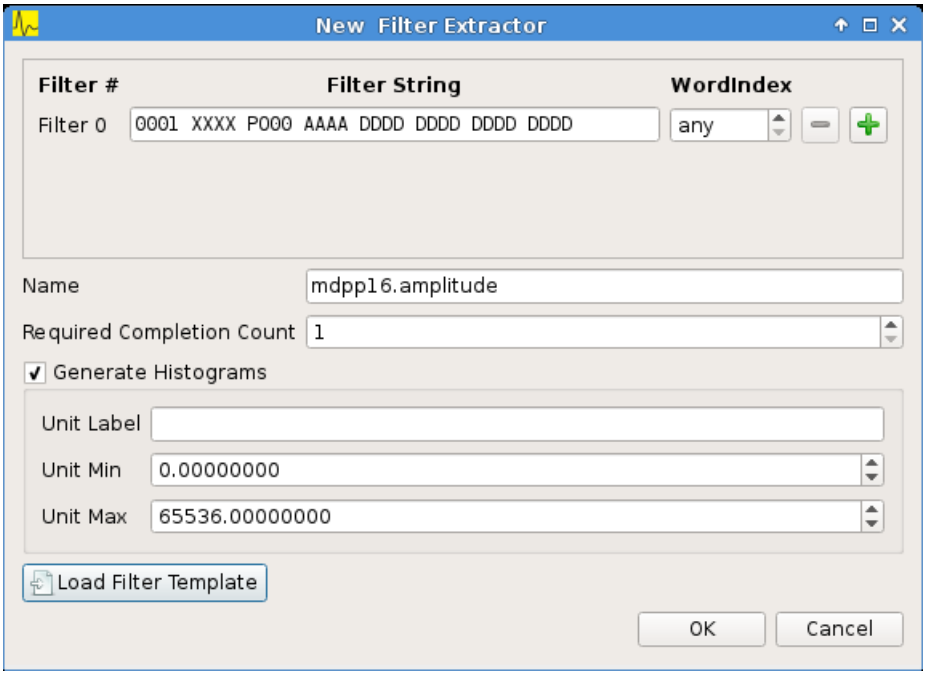


Fig. 4.12: Filter Extractor UI

Use the + and - symbols to add/remove filter words. The spinbox right of the filter string lets you specify a word index for the corresponding filter.

Required Completion Count allows you to specify how many times the filter has to match before it produces data. This completion count starts from 0 on every module event and is incremented by one each time the complete filter matches.

If *Generate Histograms* is checked raw and calibrated histograms will be created for the filter. *Unit Label*, *Unit Min* and *Unit Max* are parameters for the *Calibration Operator*.

Predefined filters can be loaded into the UI using the *Load Filter Template* button.

4.4.4 Operators

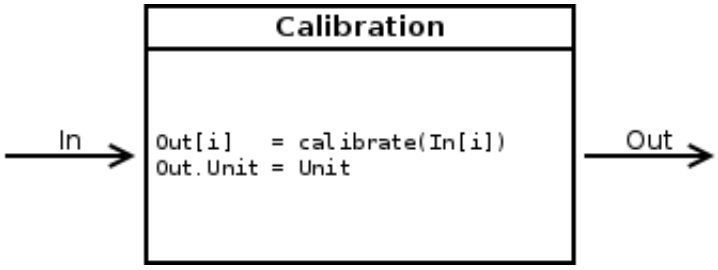
The following operators are currently implemented in mvme:

4.4.4.1 Calibration

The calibration operator allows to add a unit label to a parameter array and to calibrate input parameters using *unitMin* and *unitMax* values.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Each input parameters `[lowerLimit, upperLimit)` interval is mapped to the outputs `[unitMin, unitMax)` interval.



With *calibrate()*:

```
Out = (In - lowerLimit) * (unitMax - unitMin) / (upperLimit - lowerLimit) + unitMin
```

Limits can be specified individually for each address in the input array. Use the *Apply* button to set all addresses to the global min and max values.

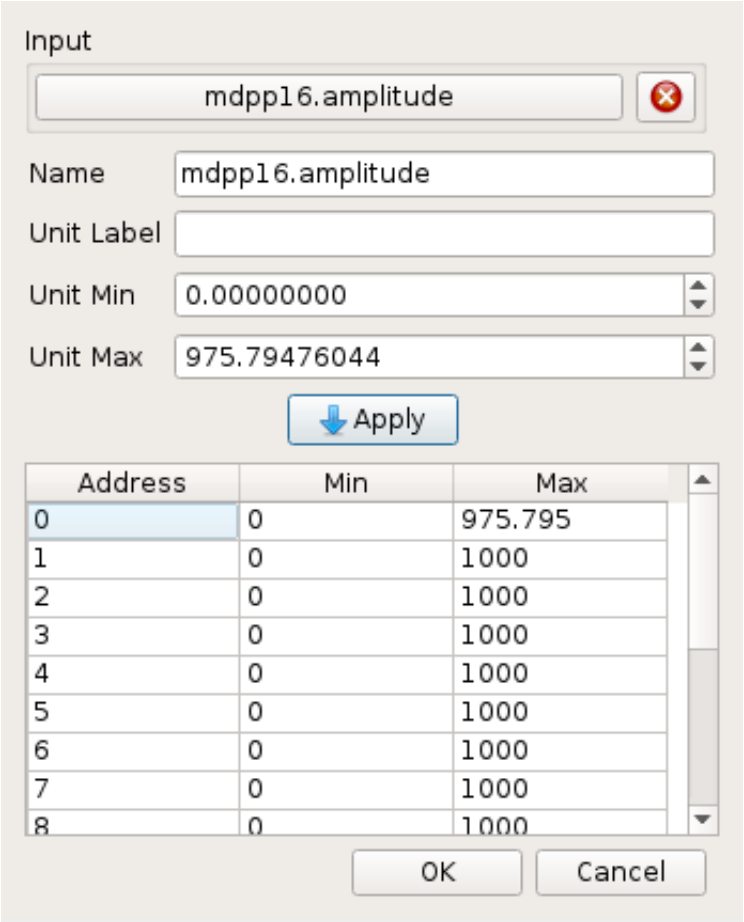


Fig. 4.13: Calibration UI

Note: Calibration information can also be accessed from adjacent 1D histograms. Refer to *Working with 1D Histograms* for details.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

4.4.4.2 Previous Value

Outputs the input array from the previous event. Optionally outputs the last input that was valid.

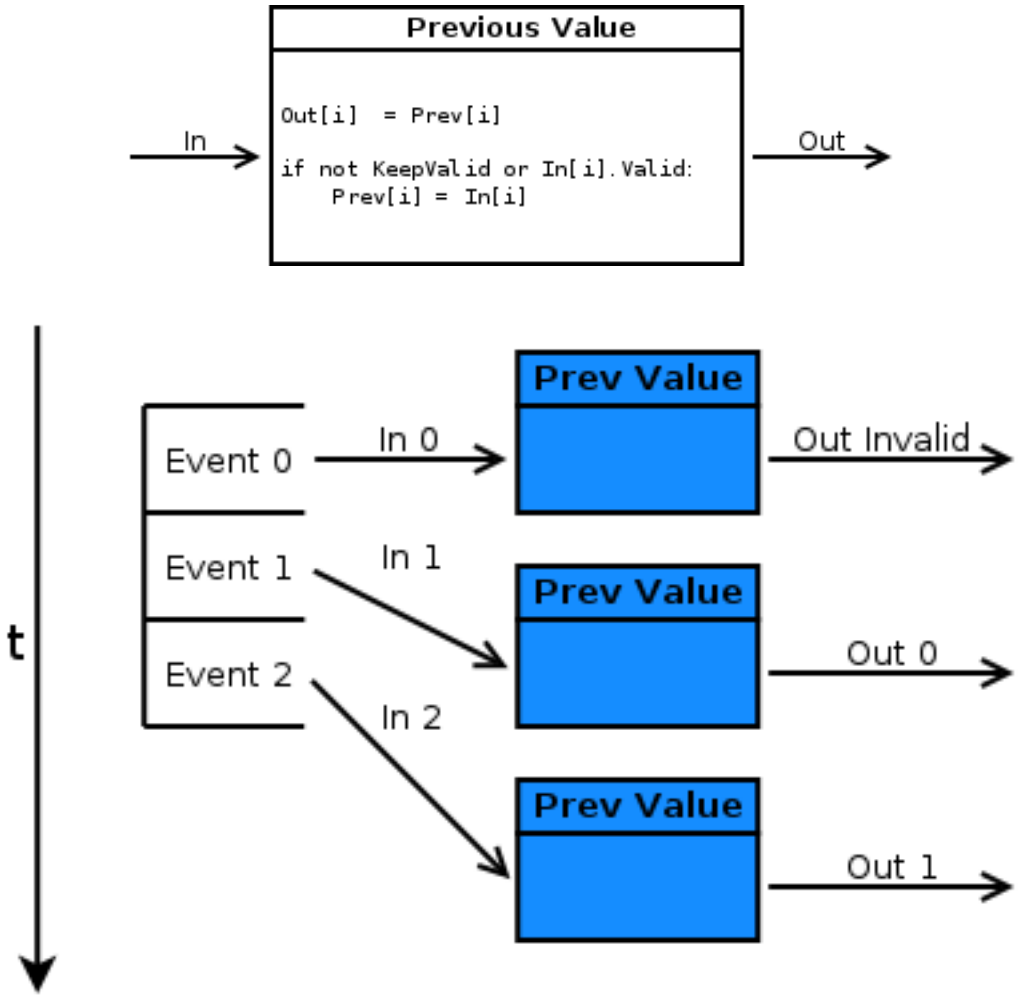


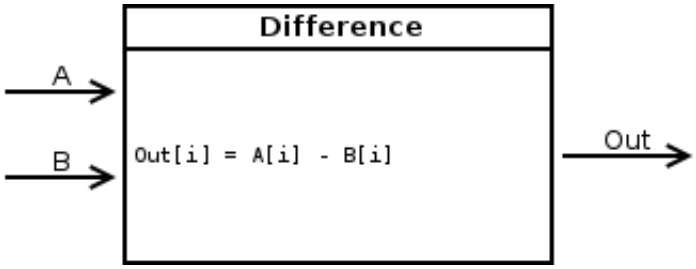
Fig. 4.14: Behaviour of Previous Value over time.

If *keepValid* is set the output will always contain the last valid input values.

This operator can be combined with the *Difference Operator* to accumulate the changes of a parameter across events. See *Rate Estimation Setup* for an example.

4.4.4.3 Difference

Produces the element-wise difference of its two inputs *A* and *B*:



The output unit label is taken from input *A*. If `A[i]` or `B[i]` are invalid then `Out[i]` will be set to invalid:

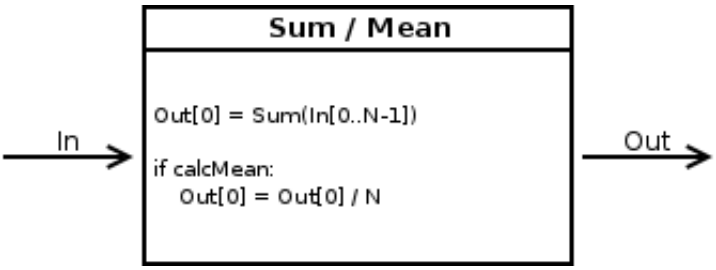
mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

```
Out.Unit = A.Unit
Out[i].lowerLimit = A[i].lowerLimit - B[i].upperLimit
Out[i].upperLimit = A[i].upperLimit - B[i].lowerLimit
Out[i].value      = A[i].value - B[i].value
```

4.4.4.4 Sum / Mean

Calculates the sum (optionally the mean) of the elements of its input array.

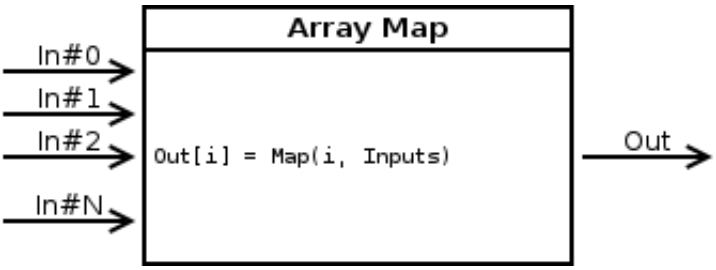
This operator produces an output array of size 1.



When calculating the mean the number of *valid* input values is used as the denominator.

4.4.4.5 Array Map

Allows selecting and reordering arbitrary indices from a variable number of input arrays.



The mappings are created via the user interface:

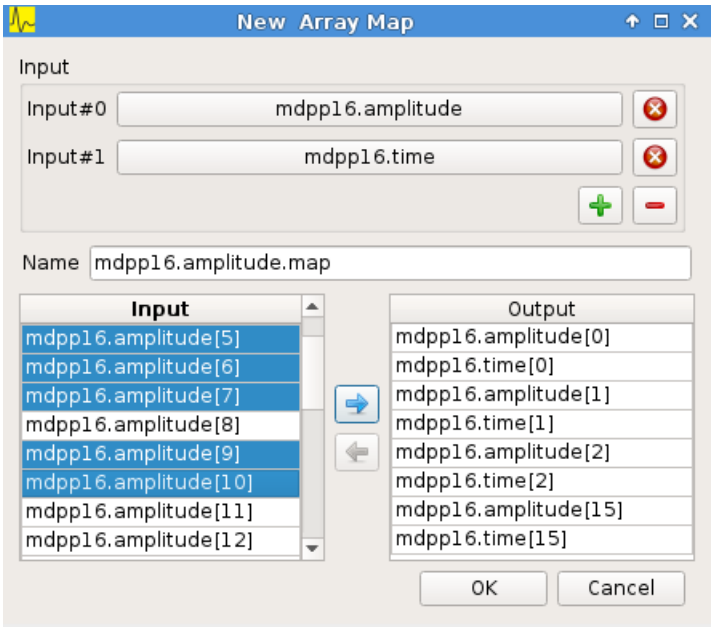
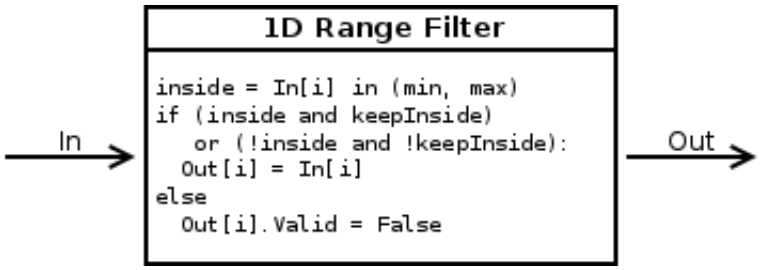


Fig. 4.15: Array Map UI

- Use the + and - buttons to add/remove inputs. The elements of newly added inputs will show up in the left *Input* list.
- Select elements in the *Input* and *Output* lists and use the arrow buttons to move them from one side to the other.

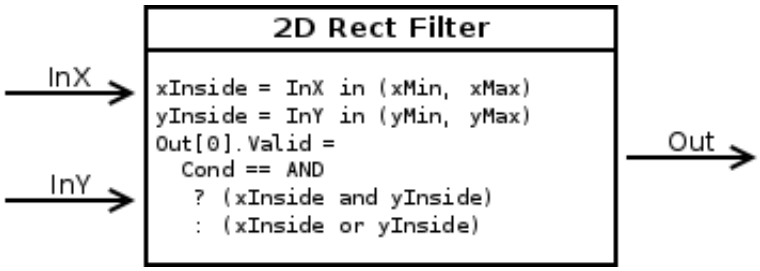
Multiple items can be selected by control-clicking, ranges of items by shift-clicking. Both methods can be combined to select ranges with holes in-between them. Focus a list and press `Ctrl-A` to select all items.

4.4.4.6 1D Range Filter



Keeps values if they fall inside (optionally outside) a given interval. Input values that do not match the criteria are set to *invalid* in the output.

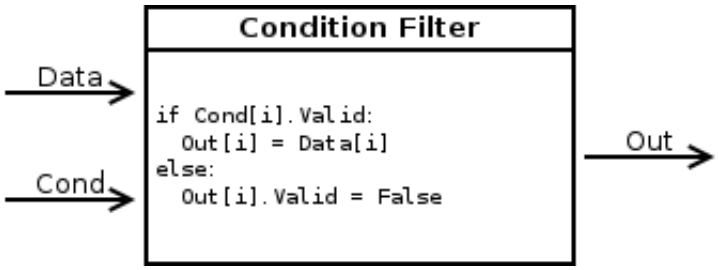
4.4.4.7 2D Rectangle Filter



mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Produces a single *valid* output value if both inputs satisfy an interval based condition.

4.4.4.8 Condition Filter



Copies data input to output if the corresponding element of the condition input is valid.

4.4.4.9 Expression Operator

This operator uses the exprtk library to compile and evaluate C-like, user-defined expressions.

The operator supports multiple inputs and outputs. The definition of the outputs is done using an exprtk script, which means arbitrary calculations can be performed to calculate the number of outputs, their sizes and their parameter limits.

During analysis runtime a second script, the *step script*, is evaluted each time event data is available. The script calculates and assigns parameter values to the operators output arrays.

Details about the syntax and semantics are provided in the online help in the Expression Operator user interface.

4.4.5 Data Sinks

mvme currently implements the following data sinks:

4.4.5.1 1D Histogram

Accumulates incoming data into 1D histograms. Accepts a full array or an individual value as input. If given a full array the number of histograms that will be created is equal to the array size.

See *Working with 1D histograms* for usage details.

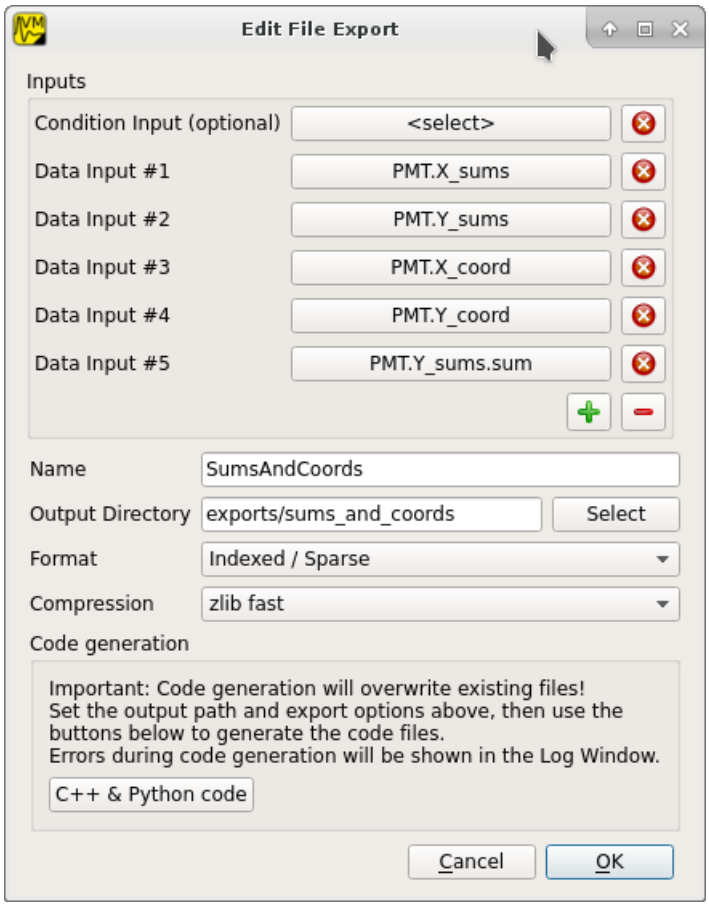
4.4.5.2 2D Histogram

Accumulates two incoming parameters into a 2D histogram. On each event input data will only be accumulated if both the X- and Y inputs are *valid*.

See *Working with 2D histograms* for details.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

4.4.5.3 Export Sink



Implements data export to binary files and C++/Python example code generation.

The Export Sink has a variable number of data input arrays that will be written to disk. Additionally a single parameter condition input can be used to pre-filter data: output data will only be generated if the condition input is *valid*.

For each DAQ run or replay an export file named *data_<runid>.bin* is generated and the data from each event is appended to that file. If zlib compression is enabled the extension *.bin.gz* is used.

The inputs define the layout of the exported data (in the case of the "Plain/Full" format the export file contains plain, packed C-structs).

Use the "C++ & Python Code" button to generate code examples showing how to read and work with export data.

To compile the C++ code run `cmake . && make` inside the export directory. The CMake file will try to find a ROOT installation using the environment variable `${ROOTSYS}` and will search for **zlib** in the standard system paths.

Most of the generated executables take an export binary file as their first command line argument, e.g:

```
./root_generate_histos my_run_001.bin.gz
```

4.4.5.4 Rate Monitor

The rate monitor uses its input values as precalculated rates or calculates the rate using the difference of successive input values. Rate values are kept in a circular buffer and a plot of the rate over time can be displayed.

Details can be found in the Rate Monitor user interface.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

4.4.6 Loading an Analysis / Importing Objects

Internally VME modules are uniquely identified by a UUID and the module type name. This information is stored in both the VME and analysis configs.

When opening (or importing from) a “foreign” analysis file, module UUIDs and types may not match. In this case auto-assignment of analysis objects to VME modules is tried first. If auto-assignment is not possible the “Module assignment” dialog will be shown.

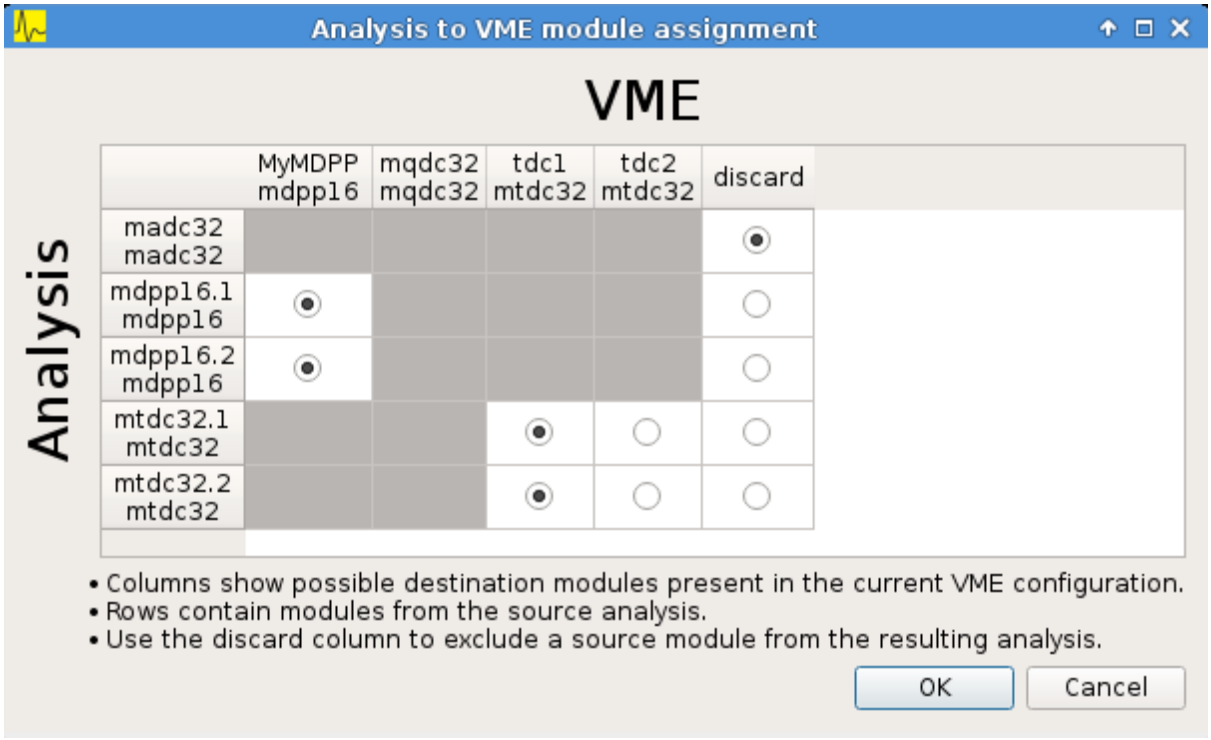


Fig. 4.16: Module assignment dialog

Each row contains one module present in the analysis that’s being opened/imported. The columns contain the modules present in the local VME configuration.

Use the radio buttons to assign analysis objects to VME modules. Select *discard* to completely remove the corresponding module from the analysis.

If the dialog is accepted the source objects UUIDs will be rewritten to match the VME object ids.

In addition to opening an analysis file and thus replacing the current analysis objects, it is possible to add objects to the current analysis. Use the *Import* action fronm the Analysis UI toolbar to import from another file.

Directly importing for a specific module is possible by right-clicking the module in the Analysis UI and selecting *Import*. Only objects matching the selected module type will be imported from the selected analysis.

Newly imported operators will be placed in a new User Level in the UI. Use drag and drop to reorder them as needed.

4.5 JSON-RPC remote control support

MVME uses the JSON-RPC specification to implement basic remote control functionality. To enable/disable the RPC-Server use the “Workspace Settings” button in the DAQ Controls Window and follow the instructions there.

If the RPC-Server is enabled mvme will open a TCP listening socket and accept incoming connections. By default mvme binds to all interfaces and listens on port 13800.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

Requests are currently not length-prefixed, instead data is read until a full JSON object has been received. The received JSON is then interpreted according to the JSON-RPC spec.

Note: The JSON-RPC batch feature is not supported by the server implementation.

A command line client written in Python3 can be found in the mvme distribution under `extras/mvme_jsonrpc_client.py`:

```
$ python3 extras/mvme_jsonrpc_client.py localhost 13800 getDAQState
---> {"id": "0", "jsonrpc": "2.0", "method": "getDAQState", "params": []}
<--- {"id": "0", "jsonrpc": "2.0", "result": "Running"}
```

4.5.1 Examples

- Requesting DAQ State:

```
---> {"id": "0", "jsonrpc": "2.0", "method": "getDAQState", "params": []}
<--- {"id": "0", "jsonrpc": "2.0", "result": "Running"}
```

- Starting data acquisition:

```
---> {"id": "0", "jsonrpc": "2.0", "method": "startDAQ", "params": []}
<--- {"id": "0", "jsonrpc": "2.0", "result": true}
```

- An error response:

```
---> {"id": "0", "jsonrpc": "2.0", "method": "startDAQ", "params": []}
<--- {"error": {"code": 102, "message": "DAQ readout worker busy"}, "id": "0",
↪"jsonrpc": "2.0"}
```

- Requesting DAQ stats:

```
---> {"id": "0", "jsonrpc": "2.0", "method": "getDAQStats", "params": []}
<--- {"id": "0", "jsonrpc": "2.0", "result": {"analysisEfficiency": 1,
↪"analyzedBuffers": 4644, "buffersWithErrors": 0, "currentTime": "2018-06-
↪14T11:45:21", "droppedBuffers": 0, "endTime": null, "listFileBytesWritten":␣
↪0, "listFileFilename": "", "runId": "180614_114412", "startTime": "2018-06-
↪14T11:44:13", "state": "Running", "totalBuffersRead": 4644, "totalBytesRead
↪": 6366924, "totalNetBytesRead": 5851300}}
```

4.5.2 Methods

4.5.2.1 getVersion

Returns the version of MVME running the JSON RPC services.

- Parameters

  None

- Returns

  String containing version information.

4.5.2.2 getLogMessages

Returns the messages buffered up in the MVME log.

- Parameters

  None

- Returns

  StringList - List containing the log messages from oldest to newest.

4.5.2.3 getDAQStats

Returns information about the current DAQ run including counter values and VME controller statistics.

- Parameters

  None

- Returns

  Object

4.5.2.4 getVMEControllerType

Returns the type of VME controller in use.

- Parameters

  None

- Returns

  String

4.5.2.5 getVMEControllerStats

Returns detailed VME controller statistics if available for the current controller type.

- Parameters

  None

- Returns:

  Object

4.5.2.6 getVMEControllerState

Returns the connection state of the VME controller.

- Parameters

  None

- Returns:

  String - “Connected”, “Disconnected” or “Connecting”

4.5.2.7 reconnectVMEController

Starts a reconnection attempt of the VME controller. The operation is asynchronous, thus the result will not be directly available. Instead the controller state needs to be polled via `getVMEControllerState` to see the result of the reconnection attempt.

Note: this might in the future be changed to a synchronous version, which immediately returns the result or any errors that occured.

- Parameters

  None

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

- Returns:

  String - “Reconnection attempt initiated”

4.5.2.8 getDAQState

Returns the current state of the DAQ.

- Parameters

  None

- Returns String - “Idle”, “Starting”, “Running”, “Stopping” or “Paused”

4.5.2.9 startDAQ

Starts a new DAQ run. The system must be idle, meaning any previous DAQ runs must have been stopped.

- Parameters

  None

- Returns

  true on success, error status and additional information otherwise.

4.5.2.10 stopDAQ

Stops the current DAQ run.

- Parameters

  None

- Returns

  true on success, error status and additional information otherwise.

CHAPTER

FIVE

HOW-TO GUIDES

5.1 Rate Estimation Setup

This example shows how to use the rate estimation feature built into 1D histograms. Rate estimation works with statistical data that forms an exponential function.

In this example and MDPP-16_SCP is used but any mesytec VME module should work. The rate estimation is setup for channel 8 of the MDPP.

To make use of the rate estimation feature a 1D histogram accumulating the *difference* between timestamp values of incoming events needs to be created.

Note: For the module to produce timestamps instead of event counter values the register `0x6038` needs to be set to `1`. mvme does this by default for newly created modules in the *VME Interface Settings* script.

Steps for creating the histogram:

5.1.1 Timestamp extraction

Add a new Extraction Filter with two filter words to the mdpp16:

```
0001 XXXX XX00 1000 XXXX XXXX XXXX XXXX     any
11XX XXXX XXXX XDDD DDDD DDDD DDDD DDDD     any
```

The first filter word matches on the data word for channel 8: the prefix `0001` identifies a data word, the pattern `00 1000` is used to select channel 8 specifically.

The second filter word is used to extract the timestamp value. The prefix `11` marks an *End of Event* word which contains the 30-bit timestamp. Using the `D` character 19-bits of the timestamp value are extracted.

Set the name of the filter to *ts_c8* and click *Ok* to create the filter.

The complete filter will thus match if there was a data word for channel 8 and the event contained a timestamp data word.

5.1.2 Timestamp calibration

By default the timestamp is generated using the 16 MHz VME SYSCLK. This means the 19-bit timestamp value is in units of 16 MHz.

The goal is to convert the value to μs which is easier to use: $2^{19}/16 = 2^{19}/2^4 = 2^{15} = 32768$.

Thus instead of ranging from $[0, 2^{19}[$ the timestamp should use the range $[0, 2^{15}[$.

To transform the value create a *Calibration Operator* by right-clicking inside the *L1 Processing* tree (or any higher level processing tree) and selecting *New -> Calibration*. Use the green + button in the top-right corner to add a new user level if necessary.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

As input select the *ts_c8* node created in the previous step. Keep the default name *ts_c8.cal*. Type `µs` in the *Unit Label* field, set *Unit Max* to `32768` and press the *Apply* button. Accept the dialog by pressing *Ok*.

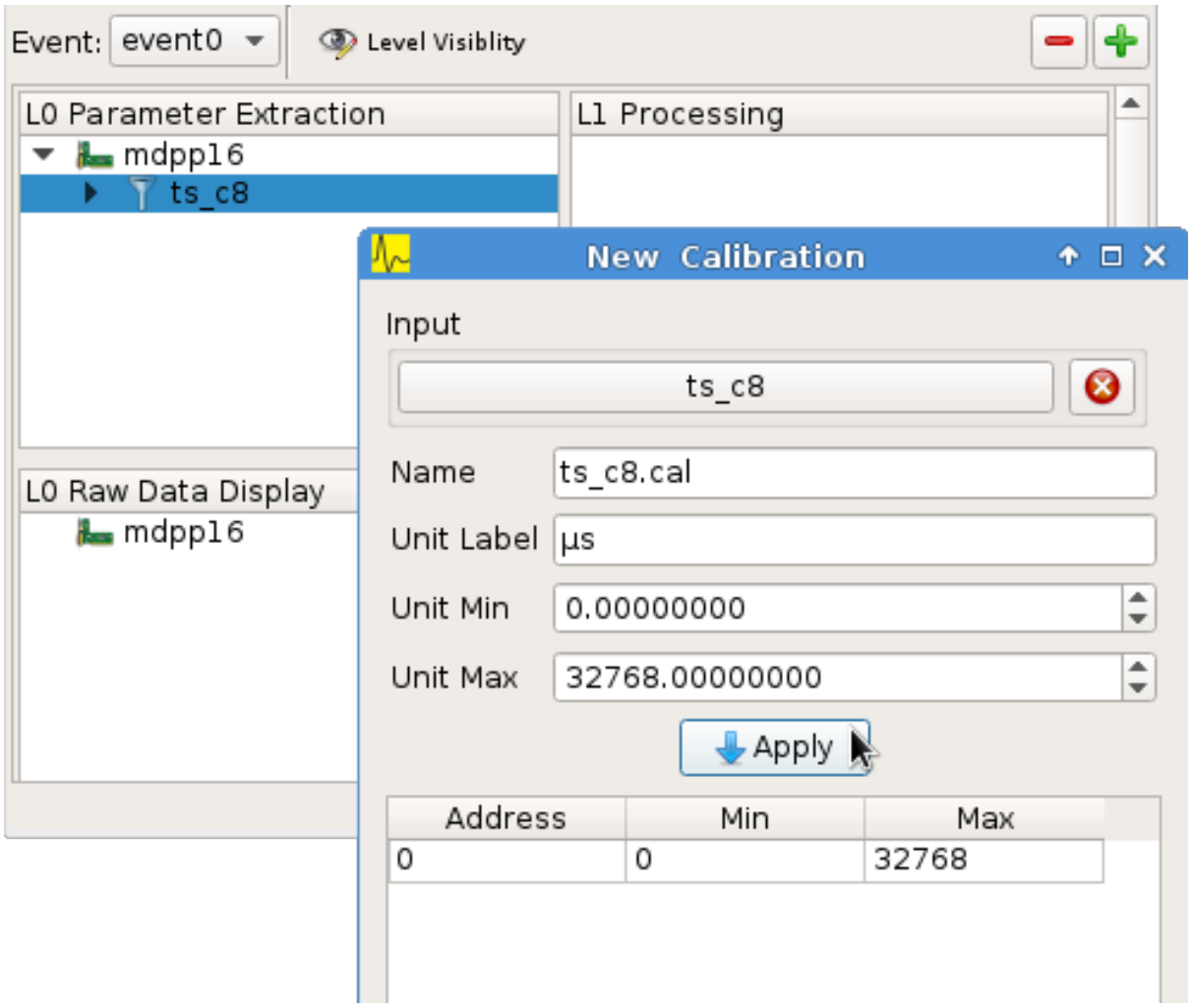


Fig. 5.1: Timestamp calibration

5.1.3 Making the previous timestamp available

Next right-click in the *L1 Processing* tree (or any higher level processing tree) and select *New -> Previous Value*.

As input select the *ts_c8.cal* node created in the previous step. Make sure the *Keep valid parameters* box is checked. Set the name to *ts_c8.cal.prev*.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

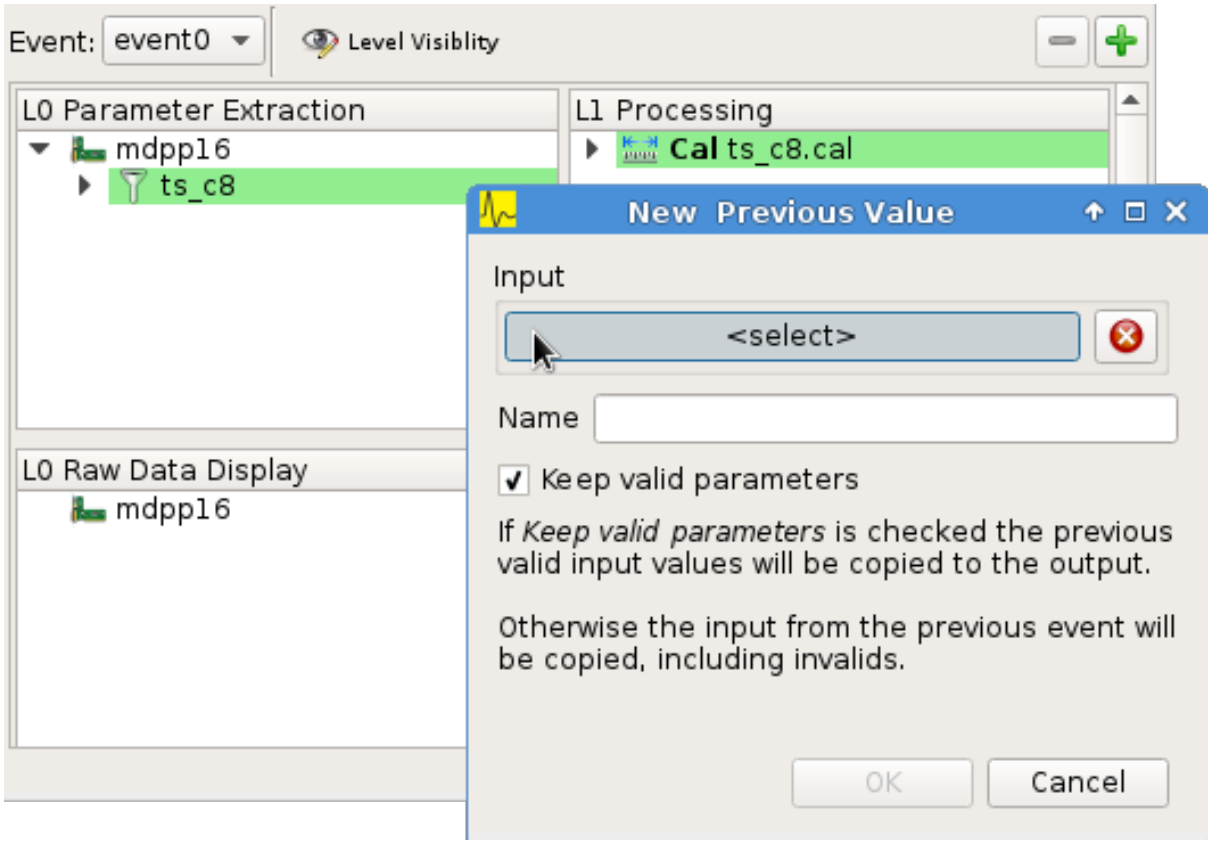


Fig. 5.2: Adding the PreviousValue operator

5.1.4 Histogramming the timestamp difference

Again right-click in one of the processing trees and choose *New -> Difference*. Set input A to *ts_c8.cal* and input B to *ts_c8.cal.prev*. Set the name to *ts_c8.diff*.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

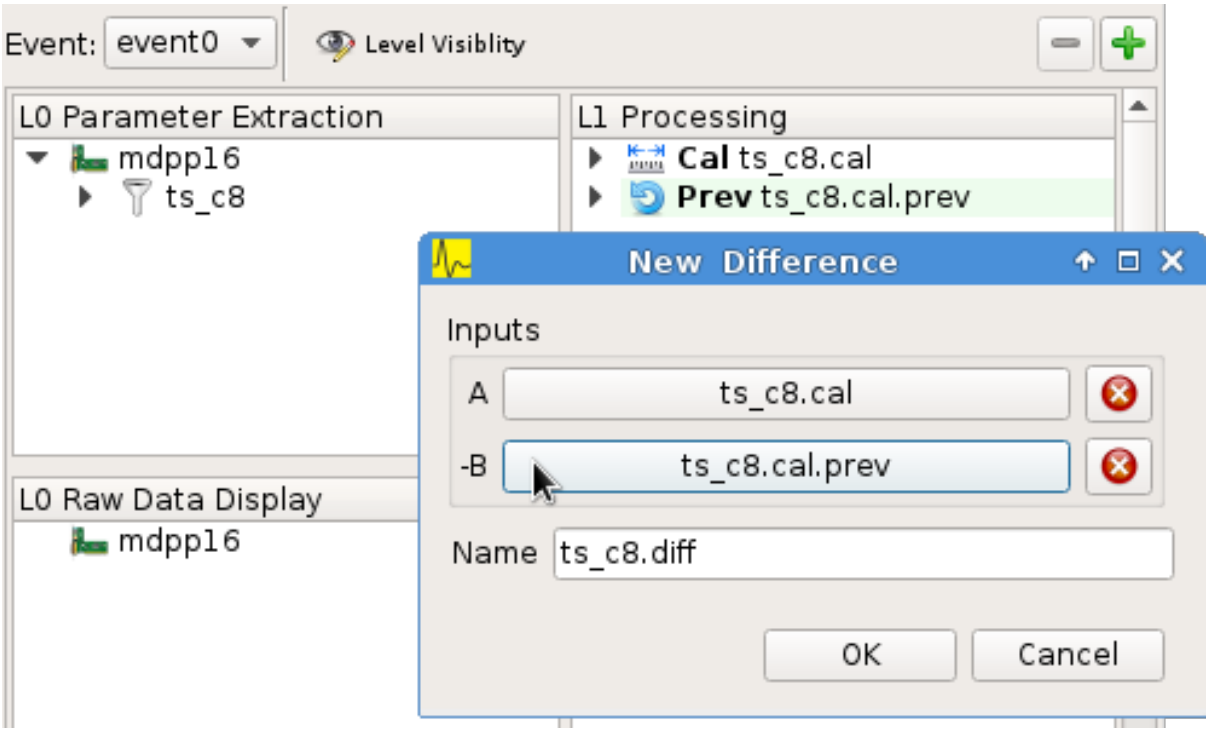


Fig. 5.3: Calculating the timestamp difference

Right-click in the display tree below and add a new 1D histogram using the difference *ts_c8.diff* as the input.

Open the newly created histogram click on *Subrange*, select *Limit X-Axis* and enter `(0.0, 200.0)` as the limits. This step limits the large default parameter range calculated by the *Difference operator*.

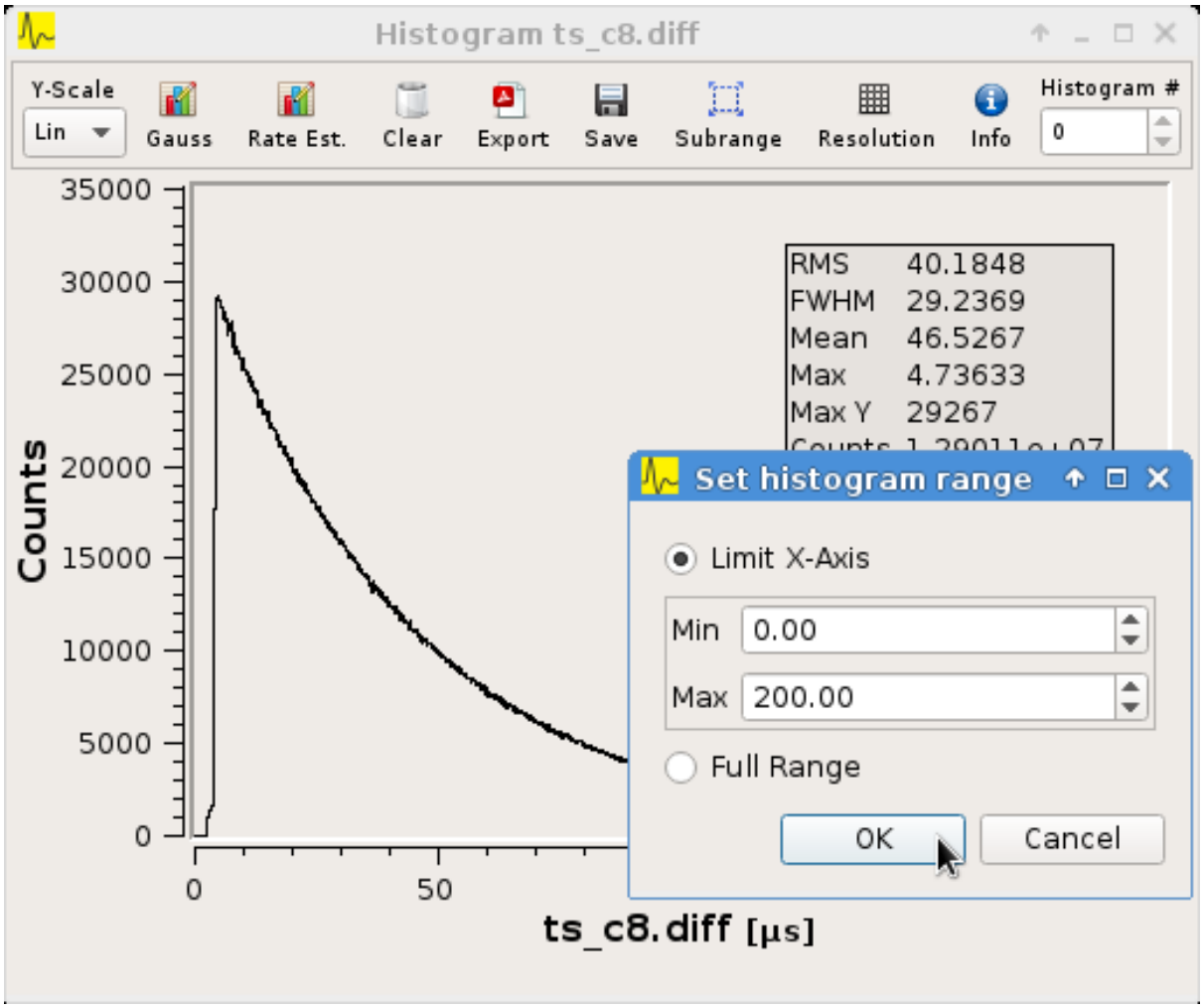


Fig. 5.4: Setting the histogram subrange

Next click the *Rate Estimation* button in the toolbar and then select two points on the x-axis to use for the rate estimation.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

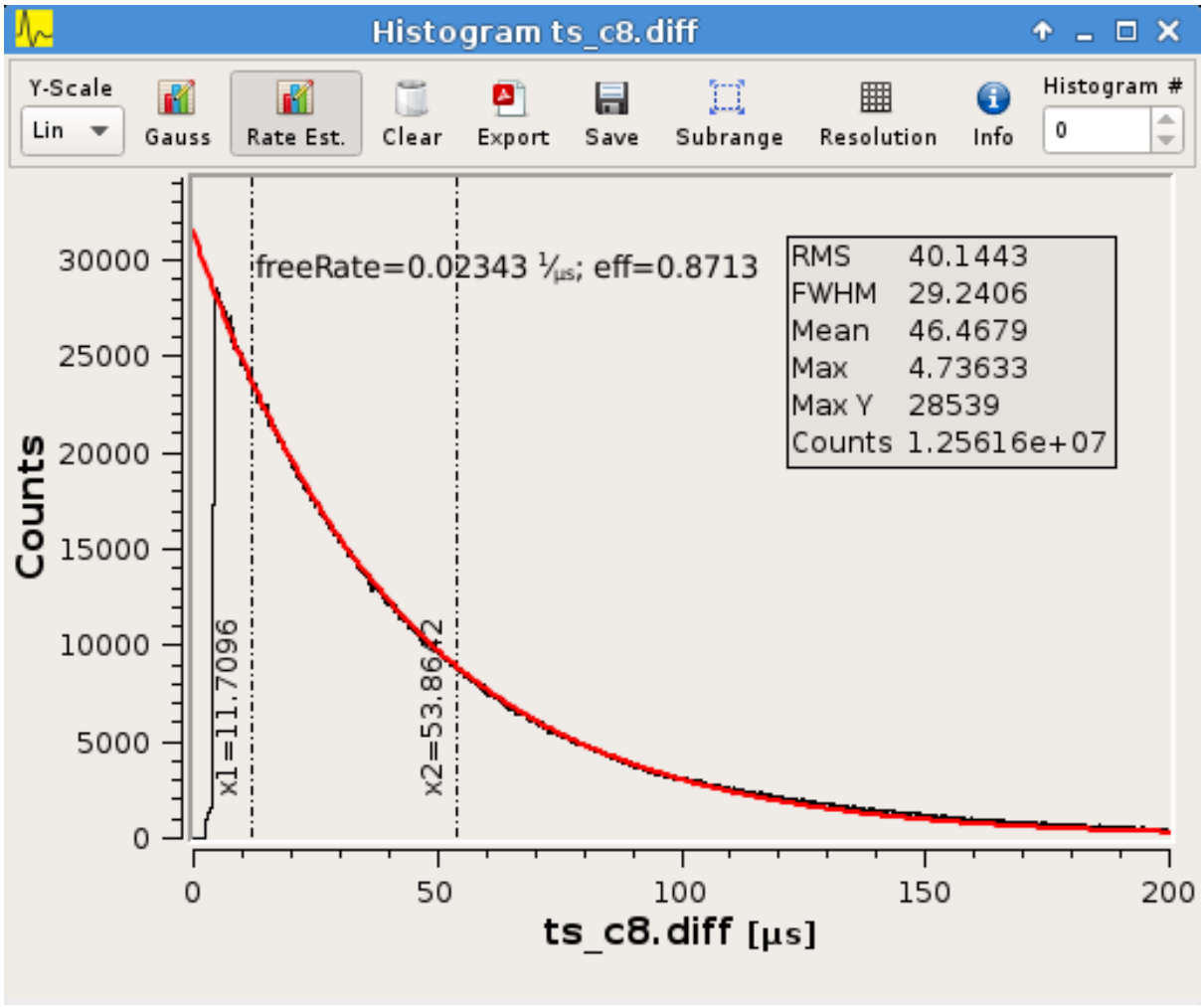


Fig. 5.5: Rate estimation data and curve visible

The calculation performed is:

$$\tau = (x_2 - x_1)/log(y_1/y_2)$$
$$y = y_1 * (e^{-x/\tau}/e^{-x_1/\tau})$$
$$freeRate = 1.0/\tau$$

5.2 VM-USB Firmware Update

The VM-USB firmware update functionality can be found in the mvme main window under *Tools -> VM-USB Firmware Update*. The latest firmware file is included in the mvme installation directory under *extras/vm-usb*.

Before starting the update set the *Prog* dial on the VM-USB to one of the programming positions P1-P4.

The controller will start the newly written firmware immediately after writing completes. Reset the *Prog* dial to C1-C4 to make the controller start the correct firmware on the next power cycle.

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

CHAPTER

SIX

CHANGELOG

6.1 0.9.4.1

- Fix expression operator GUI not properly loading indexed parameter connections
- Split Histo1D info box into global and gauss specific statistics. Fixes to gauss related calculations.

6.2 0.9.4

- New: *Analysis Expression Operator*

  This is an operator that allows user-defined scripts to be executed for each readout event. Internally exprtk is used to compile and evaluate expressions.

- New: *Analysis Export Sink*

  Allows exporting of analysis parameter arrays to binary files. Full and sparse data export formats and optional zlib compression are available.

  Source code showing how to read and process the exported data and generate ROOT histograms can be generated.

- New: *Analysis Rate Monitor*

  Allows to monitor and plot analysis data flow rates and rates calculated from successive counter values (e.g. timestamp differences).

- Moved the MultiEvent Processing option and the MultiEvent Module Header Filters from the VME side to the analysis side. This is more logical and allows changing the option when doing a replay.
- General fixes and improvements to the SIS3153 readout code.
- New: JSON-RPC interface using TCP as the transport mechanism.

  Allows to start/stop DAQ runs and to request status information.

6.3 0.9.3

- Support for the Struck SIS3153 VME Controller using an ethernet connection
- Analysis:
  - Performance improvments
  - Better statistics
  - Can now single step through events to ease debugging
  - Add additional analysis aggregate operations: min, max, mean, sigma in x and y

mesytec GmbH & Co. KG
Wernher-von-Braun-Str.1, D-85640 Putzbrunn, Germany
phone: +49 - 89 / 456007-30 fax: +49 - 89 / 456007-39
info@mesytec.com - www.mesytec.com

  - Save/load of complete analysis sessions: Histogram contents are saved to disk and can be loaded at a later time. No new replay of the data is neccessary.
  - New: rate monitoring using rates generated from readout data or flow rates through the analysis.
- Improved mesytec vme module templates. Also added templates for the new VMMR module.
- More options on how the output listfile names are generated.
- Various bugfixes and improvements

6.4 0.9.2

- New experimental feature: multi event readout support to achieve higher data rates.
- DataFilter (Extractor) behaviour change: Extraction masks do not need to be consecutive anymore. Instead a “bit gather” step is performed to group the extracted bits together and the end of the filter step.
- UI: Keep/Clear histo data on new run is now settable via radio buttons.
- VMUSB: Activate output NIM O2 while DAQ mode is active. Use the top yellow LED to signal “USB InFIFO Full”.
- Analysis performance improvements.
- Major updates to the VME templates for mesytec modules.

6.5 0.9.1

- Record a timetick every second. Timeticks are stored as sections in the listfile and are passed to the analyis during DAQ and replay.
- Add option to keep histo data across runs/replays
- Fixes to histograms with axis unit values >= 2^31
- Always use ZIP format for listfiles