VME Data Acquisition with MVLC Controller and mvme Software

(Application note 004.0)
Application note

Scope of this application note is, to show the concept of data readout of several modules while keeping synchronous events.

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

Table of Contents

VME Overview

VME crates provide power supply and a high speed parallel bus for data transfer between up to 20 standard VME modules and one VME crate controller. Data transfer rate is up to 320 Mbytes/s in the fastest Transfer mode.

Data and Address Lines

VME provides a parallel bus with **32 address** and **32 data lines**. In nuclear physics applications the bus is controlled by one readout controller, which is the bus master. All digitizer modules are bus slaves.

The data lines are bidirectional and are used for reading and writing the addressed registers in the modules or reading the data. In MBLT and 2ESST mode address lines are also used as data lines, forming a 64 wide data bus.

The 16 high bits of the address lines are usually used to give the VME modules a unique base address in the 32 bit address space, while the lower 16 bits address the registers inside a module.

Address Modifier Lines to Control the Data Flow

- There are **6 address modifier** lines (AM).

  They are set by the controller and code for the four basic data transfer modes of VME:

  1. single word data read/write (typically 200..400ns per cycle, 16 to 32 bits).
  2. block transfer: up to 256 words with width of 32 bits can be read in one cycle from a module (typically 100 to 200ns per cycle, 32 bits)
  3. Multiplexed block transfer: read 256 words with a width of 64 bits from a module. In this case the address lines are also used for data transfer (66 to 200ns per cycle).
  4. 2ESST (two edge source synchronous mode), transmits 64 bits, down to 20ns/cycle. This mode is specified for VME64x, but also works for good VME64 backplanes, and small VME64 crates.

Interrupt lines

- Then there are **7 interrupt lines** to signal states from a module to the controller. They can signal interrupts from 1 to 7. This is useful to signal to the controller that a module has data, or that a buffer is almost full.

  Which interrupt number to send is configured in the module registers. VME also allows to send an 8 bit interrupt vector when interrupt acknowledge is sent. Mesytec modules do not use those data, also interrupt acknowledge is not necessary after an interrupt (but possible).

  The action to take after an interrupt, is left to the controller.
- For mesytec modules with MVLC controller, IRQ lines they can also be “abused” to signal triggers to the MVLC-TrigIO to feed into a trigger generation logic.

Addressing modes

Typical addressing for modern modules is 32 bit addressing (A32). (Also 24 bit addressing (A24) is supported, but not recommended.)

The **base address**, which are the leading 16 bits in A32 mode, can be set via rotary coders on the motherboards of the modules or via address registers.

For VME64x crates, also geo-addressing is available. When setting the address coders of mesytec modules to 0xFFFF, geo-addressing is used (the high byte gets the number of the slot).

A second base address can be set by register for Multi Cast access (MCST).

Multi Cast (MCST)

Multicast means that data words can be written to more than one module simultaneously. The data transfer is identical to the standard data transfer, but addresses a second base address which is set common for more than one module. A good application is a readout reset to all modules after the data is read out.

MCST is also essential to reset synchronous timer counters at the start of a run. Mesytec modules provide counters, which count the backplane clock of 16 MHz (or an external clock) and so allow to transmit a time stamp with each event. This time stamp is synchronous to all modules in a crate.

Readout to Berr

For nuclear physics setups a very efficient readout termination is widely used. Every controller used in nuclear physics should be able to handle it.

As the amount of data to be read is usually unknown to the bus controller, it may just read with block transfers at the module FIFO. When there are no more data, or a full event is read out, the module emits a bus error "Berr". This is not an error condition, but simply a signal showing the termination of the data transfer. The controller can then directly proceed to the next module readout.

Many useful features like transferring a fixed number of events (with unknown length), can only be used when reading to Berr.

VME64 / 64x

VME-64 provides +5 V, and ±12 V supply

VME64x is an advanced standard. It provides +3.3, + 5V, ±12 V supply, geo addressing and additional ground lines for better signal integrity on the backplane.

VME64 modules can always be operated in VME64x crates.

Mesytec modules can always be operated in both standards.

Special concern has to be taken when using VME64x crates and modules. Push in the modules completely and fix them with the locker screws. Insertion force may be up to 300N ! First time users of VME64x very commonly fail to insert the modules completely.

Modules tend to disconnect (jump out) if they are not fixed.

Readout modes

Introduction, Sequence of a data acquisition

For each module

Reset

In the first step modules are set to their default initialisation (reset).

The modules may take up to 500ms to get ready again. In the mvme scripts, module type and the actual software type and revision is read. For some modules the software is checked if it is correct for the intended use.

Initialisation, Frontend settings

The required registers of all modules are set, defining resolution, window of interest, integration times thresholds...

VME interface Setting

The VME transfer mode, IRQ signalling and multi cast addresses are initialised.

Simultaneous for all modules

Multicast start:

the counters are reset (synchronised)

Buffers are reset

The modules are started and then accept triggers

The Start Daq bit is set in MVLC, which is available in Trig-IO for Experiment trigger gating → start triggers.

Readout Loop

When the modules are signalling **data available** via VME IRQ, the readout loop in MVLC is started

It reads data from first module until the module shows “Berr” then the next module is read out.

When all modules were read, a multicast command writes to the modules (reg 0x6034) to allow the next readout cycle.

At this point the readout loop waits for the next “data available” signalling from modules.

Event by event readout (mode 0)

For mesytec modules set mode register 0x6036 of all modules to 0 to enter event by event mode.

For the classical analogue modules (MADC-32):

- A discriminator which whatches the analogue signals after preamplifier generates a trigger
- An external Logic may decide to accept the trigger and forwards it to the digitizer module.
- The module gets a trigger or gate,
- Immediately blocks further triggers,
- Converts the input signals,
- Sends them to its local buffer, (→ conversion time)
- Signals data in the buffer to Data acquisition,
- Data acquisition reads event data from module buffer via VME bus (→ readout time)
- Data acquisition signals to the module to allow for the next trigger.

For the digital modules (MDPP-16/32, VMMR, MTDC-32)

Analogue signals at the inputs are converted as they come, digital data are stored in intermediate memories.

An incoming trigger can be shifted in time and a gate width can be specified (window of interest). So after the trigger, data with the specified time tag are selected and sent to the output buffer. The “conversion time” of analogue modules translates to a trigger dead time of the digital modules. The trigger dead time is the minimum time between two triggers. This depends on the specified width of time window and a constant data processing time. For single event readout the time the selected data need to travel to the output buffer (latency time) have to be added to the trigger dead time. All digital modules include a full processing chain, so can also produce a trigger when data at the input are detected. They can also be operated free triggering without an external trigger. Then they generate an internal trigger and open the time window for data selection internally.

Readout for the digital modules:

- The built in discriminator in the digital module generates a trigger
- An external Logic may decide to accept the trigger and forwards it back to the digitizer module.
- The module gets a trigger,
- Immediately blocks further triggers,
- selects the data with matching time stamp,
- Sends them to its local buffer, (→ conversion time)
- Signals data in the buffer to Data acquisition,
- Data acquisition reads event data from module buffer via VME bus (→ readout time)
- Data acquisition signals to the module to allow for the next trigger.

This mode is very easy to implement, but has a large dead time from signal arriving at the module input to digitised data ready for readout. In this mode module data buffers are only used for one event.

When more than one Module is used, also for this type of readout, a single experiment trigger should be generated (see next chapter), to trigger all modules with an identical trigger.

If this is not done, there will be no clean coincidence definition of data from different detectors and modules.

A part of data from different modules may not belong to the same physical event.

In this mode, the module buffers are only used for the data of one event. After conversion all modules are read out.

The procedure is:

- Wait a fixed time after trigger for the slowest module to convert, then read out all modules.
  Or Wait for a data ready interrupt of the slowest module, then start readout.
- Read the converted event from each module forming a “Crate event”
- write event reset 0x6034 via MCST to all modules to allow modules to accept the next trigger.

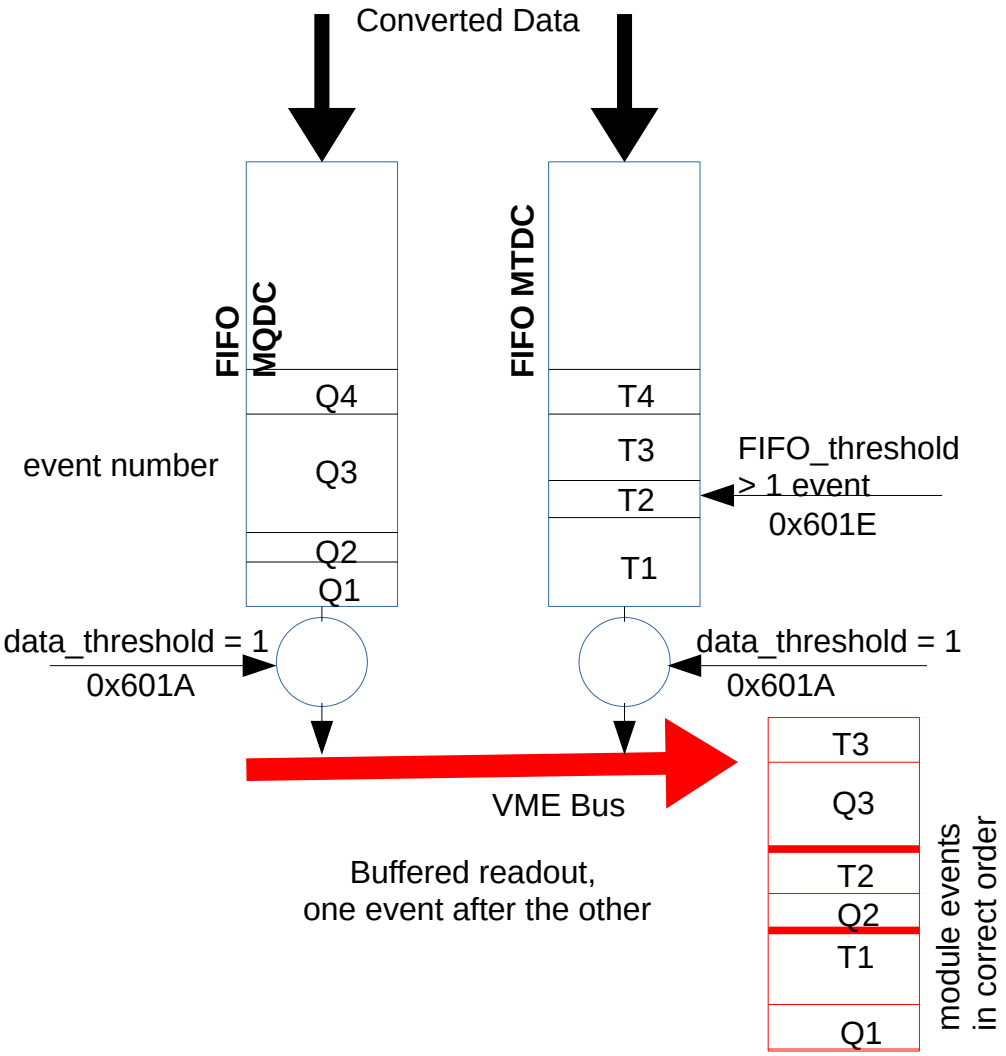
The advantage of this mode: very easy implementation, very low requirement for the modules, almost all modules can be integrated.

Disadvantage: low data throughput, high dead time.

Multi event readout , Readout using the module buffers (mode 0xB),.

Set mode register 0x6036 to 0xB. The event data conversion and readout are decoupled, so no longer synchronous. Readout time does no longer add to the conversion time.

The challenge is to synchronise the data sets of different modules to get a common data set belonging to the same physical event.



Essential for this type of readout:

1) all modules must create event data packets when they get a trigger and send them to their buffer. When all modules get the same trigger (= experiment trigger) and are not busy when they get the trigger, this condition is fulfilled.

So a logic has to decide from a pattern of several detector triggers that there is a “good event” and send this signal (= experiment trigger) to all trigger inputs of the modules. The experiment trigger must be inhibited for the longest time needed in a module to process the trigger (trigger dead time).

For the digital mesytec modules the “trigger dead time” can be calculated from “window width” of the window of interest + trigger recovery time.

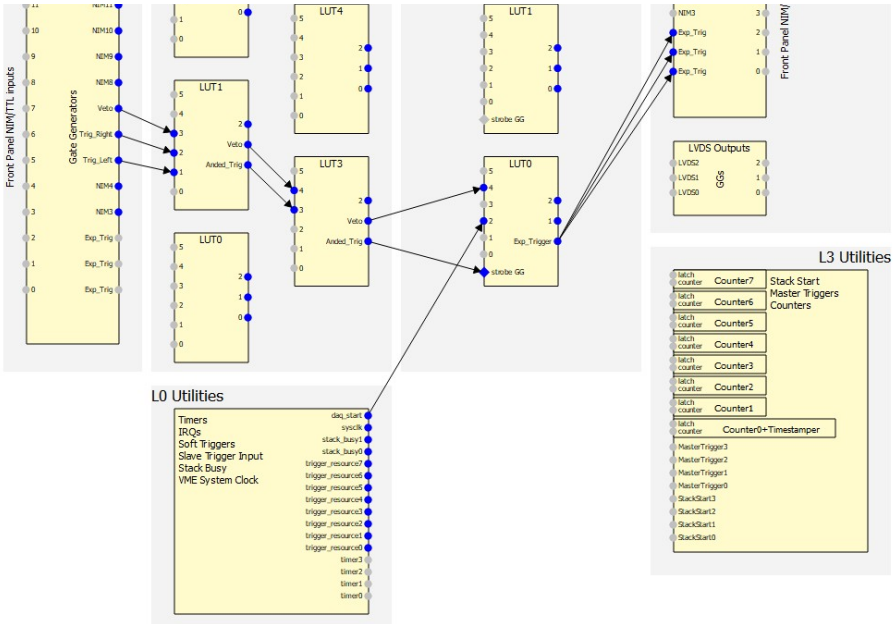
Module	MTDC	MDPP16	MDPP32	VMMR
Trigger recovery time	800ns	500ns	900ns	400ns

2) a readout cycle has to read the same number of events from each module. This can be easily fulfilled when more than the required number of events are in all module buffers (0x601C = 0, 0x601E > number of events to transmit). And each module releases the same number of events in a readout cycle (0x6036 = 0xb, and 0x601A = number of events to release)

When there are enough events in the buffer, this can be signalled by VME-IRQ.

Some modules like VMMR may have a relatively long latency due to large events and internal processing. This means that several event data packets may be on the way to the buffer. This doesn’t increase dead time, but readout of data is significantly delayed to the trigger. VMMR is well suited as the module signalling the readout IRQ (enough events in buffer).

Realisation of Experiment trigger generation in MVLC-TrigIO

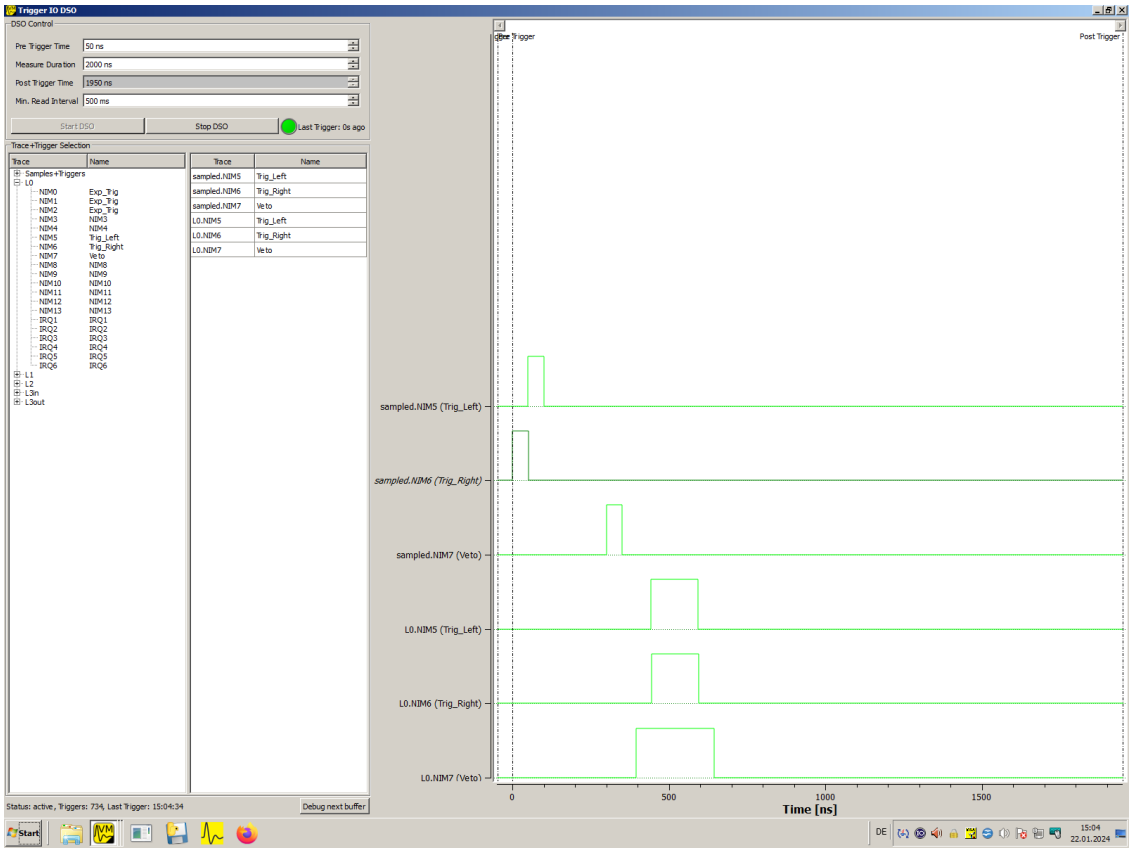


Example: a setup with 3 detectors, one is left another right, and a veto detector.

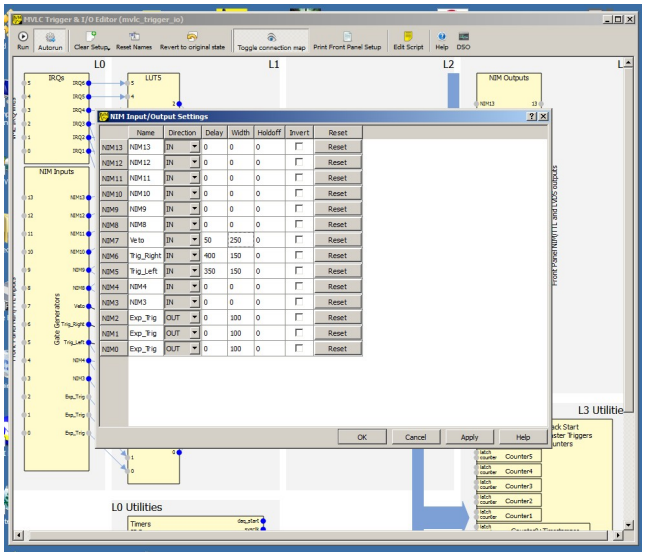
When detector left and detector right respond, and the veto detector has no signal, the event is classified as good, and an experiment trigger is generated. It signals all modules to store this event in their buffer.

In this example each detector is on one MDPP-16. So 3 triggers are available at the module trigger outputs. They are connected to the MVLC Trig-IO inputs (left input side of Trig-IO in the picture above). Each input provides a gate generator, so an independent delay and width can be added to the input signals to compensate for different delays of the detectors and trigger generation.

The following picture shows the 3 input signals at the top, followed by the 3 delayed and widened signals after passing the gate generators.

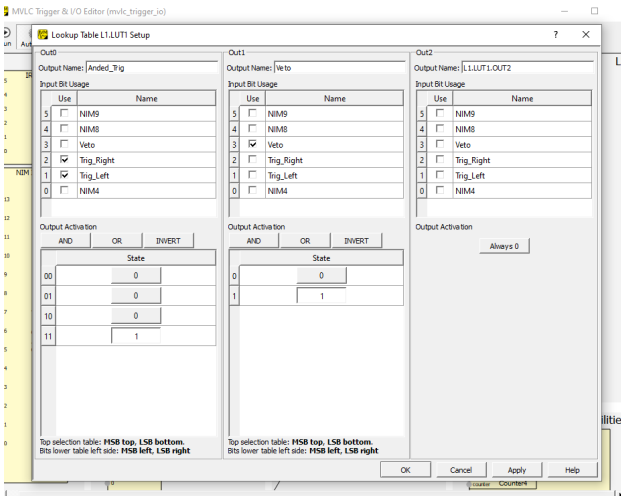


Following picture shows the editor window to set the gate generators:



The three trigger signals shifted to the same time can now be used to create coincidences.

The logic unit “LUT1” creates an overlap AND of the two trigger signals, and feeds through the “Veto” signal. The LUT is a lookup table with 6 inputs and 3 outputs. It works as a universal logic between inputs and outputs. The sum of the width of the two triggers signals define the coincidence time.



The picture shows the editor window for the LUT. First column establishes the AND between the two trigger inputs. The second column is a route through for the VETO signal.

The next LUT3 is only a route through. The anded trigger signal and the veto signal are now fed to the final LUT. This one has an additional optional “Strobe” input. We use it here for demonstration.

The strobe input samples the logic state generated in the LUT at the time the edge of strobe signal rises. It has an additional gate generator, which allows to delay the strobe input, and a “hold off”, which is a blocking time after the edge. The hold off time is used to block the trigger input at strobe for the time modules are busy after a trigger (gate dead time). For a setup with MDPP-32 modules with a window of interest width of 1us, a hold off time of 2000ns will be required.

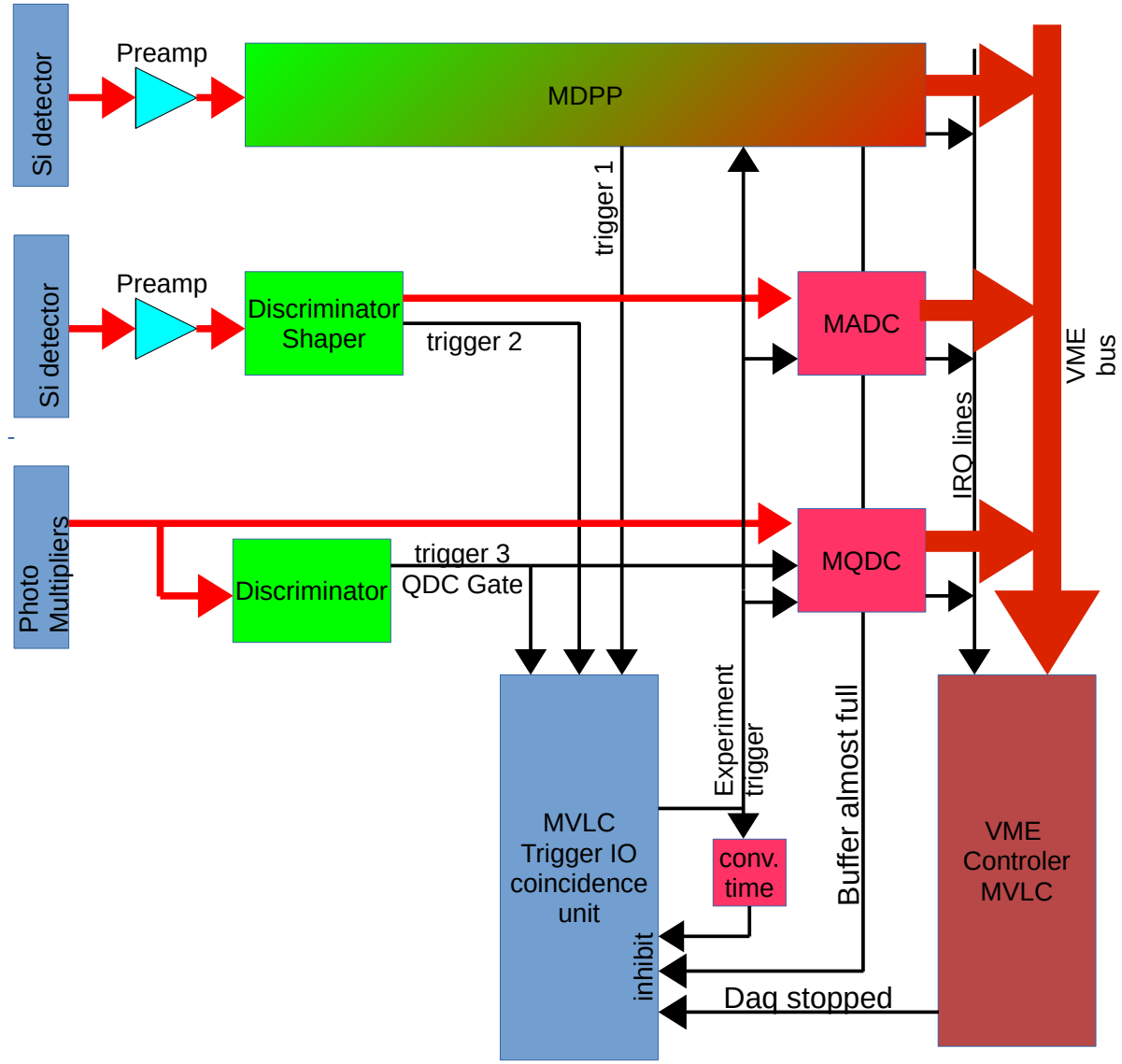
In this LUT logic also “data acquisition start”, “buffer full” and external signals to block the triggers may be included.

The output of this final logic will be the experiment trigger.

Also free triggers can be generated for counting. The experiment trigger can be connected to several outputs to feed trigger inputs of several modules. Multiple outputs generated from the same source have a jitter versus each other of less than 50ps.

That’s important because the experiment trigger is used as a time basis for the timing within an event.

The following picture shows an example readout with three different modules. All of them get the experiment trigger to generate an event in their buffer.



In the readout cycle it is also possible to read a fixed number of events per module. This increases the amount of data read in one cycle, and reduces the time overhead for handling interrupts.

The data then have to be split into the elementary crate events before the analysis. In the mvme software the “Multi Event Processing” feature has to be switched on. See Icon “Event Settings” in Analysis window.

Advantage of the multi event readout is a much shorter dead time and higher rate capability. Only the trigger dead time for the digital modules is relevant. For analogue modules like MADC-32 the gate width, gate delay and the conversion time will determine the dead time. The readout also runs while modules are receiving detector signals.

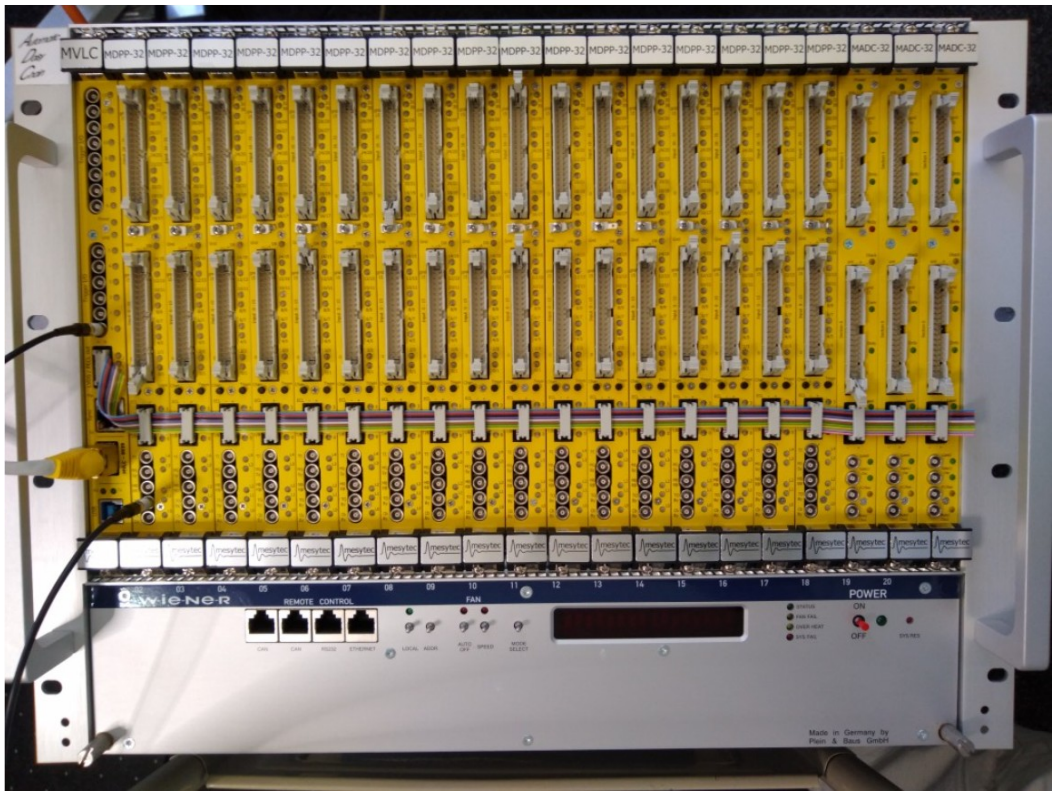
MVLC crate controller with mvme software can transmit up to 120 Mbytes/s via Ethernet connection and up to 150MBytes/s via USB3 and store it in a list file.

Risk of multi event readout is, that at any error in trigger blocking, so when ever one module can not accept a trigger, data get asynchronous. To overcome this risk, all mesytec modules should have the time stamping activated (usual default). Then in “Event Settings” in the analysis window you can select “event builder”, “enable event builder”. The “main reference module” may be any module in the readout chain.

Now the module events are matched for coincident time stamps. Only matched crate events are forwarded to the analysis.

For ease of use with many modules, it may be convenient to distribute the experiment trigger via flat cable from MVLC PECL output to all modules in parallel. Only the highest three input pin input pairs and the two ground lines should be connected. The lower two output pins should be left free.

ECL direction	MADC		MTDC		MDPP16/32	
input	**T0**		**T0**		**T0**	
input	**T1**	Sync	**T1**	Sync	**T1**	Sync
input	**Fast clear**	Res		Res		Res
output	Busy		Busy		Busy	



Event Synchronisation via Time stamp

In this mode all modules emit a “data available” signal. The readout loop reads all modules, not necessarily the same amount of events per module. All modules must add a time stamp to their event data.

In extreme case the modules can be triggered free running, so each one with its own trigger.

In the data acquisition events are sorted by their time stamp and synchronous crate events are extracted.

For this mode, choose “Event Settings” in the analysis window then select “event builder”, “enable event builder”.

One module, which always responds when there is an interesting event, has to be selected as “main reference module”.

This mode is very simple, but produces a lot of uncorrelated data and increases the dead time. That’s because a module with high rate produces data all the time. It may not be ready when an interesting, correlated event arrives coincident on all modules.

So also for this mode, a common Experiment trigger should be generated to select reasonable interactions from background.

The experiment trigger is also used as a time basis for ps timing within a correlated event (for modules MDPP16/32, MTDC32).

The event time stamp only provides a timing resolution within the clock period (62.5ns for the backplane clock).

Using Backplane IRQ lines to Forward Triggers.

There is a powerful mechanism for complex coincidences. In the "VME interface settings" of the modules (MDPP-16 / 32, MTDC, VMMR), at the top of the script you can find this entry:

```
# Trigger output via IRQ-lines at Backplane; MDPP-16
# Implemented since March 2021.
# Each bit corresponds to one of the MDPP-16 channels.
# 0x6300 0b1111'1111'1111'1111  # IRQ 1
# 0x6304 0b1111'1111'1111'1111  # IRQ 2
# 0x6308 0b1111'1111'1111'1111  # IRQ 3
# 0x630C 0b1111'1111'1111'1111  # IRQ 4
# 0x6310 0b1111'1111'1111'1111  # IRQ 5
# 0x6314 0b1111'1111'1111'1111  # IRQ 6
# 0x6318 0b1111'1111'1111'1111  # IRQ 7  ** not supported by MVLC Trig-IO

# ######################################
```

Or for 32 channel modules MDPP-32 and MTDC-32

```
# Trigger output via IRQ-lines at Backplane; MDPP-32
# Implemented since March 2021.
# 0x6300 0b1111'1111'1111'1111 #IRQ 1 lower 16 channels
# 0x6302 0b1111'1111'1111'1111 # upper 16 channels
# 0x6304 0b1111'1111'1111'1111 #IRQ2 lower 16 channels
# 0x6306 0b1111'1111'1111'1111 # upper 16 channels
# 0x6308 0b1111'1111'1111'1111 #IRQ3 lower 16 channels
# 0x630A 0b1111'1111'1111'1111 # upper 16 channels
# 0x630C 0b1111'1111'1111'1111 #IRQ4 lower 16 channels
# 0x630E 0b1111'1111'1111'1111 # upper 16 channels
# 0x6310 0b1111'1111'1111'1111 #IRQ5 lower 16 channels
# 0x6312 0b1111'1111'1111'1111 # upper 16 channels
# 0x6314 0b1111'1111'1111'1111 #IRQ6 lower 16 channels
# 0x6316 0b1111'1111'1111'1111 # upper 16 channels
# 0x6318 0b1111'1111'1111'1111 #IRQ7** not supported by MVLC Trig-IO
# 0x631A 0b1111'1111'1111'1111 #** not supported by MVLC Trig-IO
########################################
```

Don't use IRQs which are used to trigger readout.
The other free lines can be configured to create a pulse when any group of channels respond.
So for example IRQ 1 and 7 is used for triggering, then IRQ2..IRQ6 are available to form 5 coincidence groups.

Example:
Put a detector on chan 0..3 to IRQ2 : 0x6304 0b0000'0000'0000'1111 # IRQ 2.

Now the VME back plane IRQ will respond when any of the four channels responds. Responding channels create a 50ns pulse on the back plane line, timing resolution is 12.5 ns for MDPP modules. When you want detectors (channels) in other modules to join the same coincidence group, just put them to the same IRQ line (only reasonable when the timing is similar). IRQ lines provide a “wired or” between the modules.

The IRQ lines can be evaluated in the trigger-IO of MVLC. They arrive at the top left corner in Trigger IO and can be evaluated with the trigger-IO logic to create the experiment trigger.
At the input they should be shifted with the input gate generators to arrive at the same time.
You can use the built in DSO to inspect and shift signals.

The experiment trigger is distributed to the trigger input of all modules (you can connect the signal to several trig-io outputs to multiply it for several modules, they keep absolutely synchronous).
The modules are set to be triggered external (0x6058 0x001 for trigger 0 input).
The Window of interest in the modules has to be adjusted for the delayed experiment trigger, so shifted negative in time.